João Pedro Carvalho Leal Mendes Moreira

Travel Time Prediction for the Planning of Mass Transit Companies: a Machine Learning Approach

Tese apresentada à Faculdade de Engenharia da Universidade do Porto
para obtenção do grau de Doutor em Ciências de Engenharia,
realizada sob orientação científica do
Prof. Doutor Jorge Rui Guimarães Freire de Sousa,
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto,
e co-orientação científica do
Prof. Doutor Alípio Mário Guedes Jorge,
Professor Auxiliar da Faculdade de Economia da Universidade do Porto

Departamento de Engenharia Informática
Faculdade de Engenharia da Universidade do Porto

Novembro de 2008

Dedico esta tese à minha mãe.
Nada fez excepto o que foi
necessário para me permitir
realizar este trabalho.
E foi imenso.

Resumo

Nesta tese é realizado um estudo sobre como utilizar previsões de tempos de viagem nas tarefas de planeamento de empresas de transportes rodoviários de passageiros. São identificados dois problemas: a definição dos tempos de viagem a utilizar (1) nos horários e (2) na definição de serviços quer de viaturas quer de motoristas. Todos estes estudos assumem a existência de dados sobre as viagens realizadas, obtidos tipicamente através de sistemas de localização automática de viaturas.

O primeiro problema não é novo, havendo vários estudos relacionados na literatura. A nossa abordagem não é analítica. Em alternativa desenhamos e desenvolvemos um sistema de apoio à decisão que utiliza dados históricos do mesmo percurso e representativos do período em que os horários estarão em vigor. Este problema foi o menos estudado.

A abordagem ao segundo problema, previsão de tempos de viagem com três dias de antecedência, consiste em testar quanto se pode aumentar em precisão se se utilizarem na definição dos serviços dos veículos e dos motoristas previsões feitas tão próximo da data desses serviços quanto possível, em vez dos tempos definidos nos horários. A motivação para a realização deste estudo é que, caso o aumento de precisão seja significativo, é possível que se consiga reduzir os custos operacionais e/ou aumentar o nível de satisfação dos clientes.

Neste segundo problema, utilizamos abordagens de aprendizagem automática. Não obstante, iniciamos pela definição de um método simples (para permitir avaliar comparativamente os resultados obtidos com métodos mais complexos) e de um outro baseado no conhecimento existente no início desta investigação quer da nossa parte quer da parte dos especialistas de trânsito da empresa STCP. De seguida experimentamos três algoritmos com bons resultados em diferentes problemas. São eles: máquinas com vectores de suporte, florestas aleatórias e regressão por projecções (*projection pursuit regression)*. Para cada um destes algoritmos foram realizados testes exaustivos para afinação de parâmetros. Foram realizados mais testes sobre: selecção de instâncias, selecção dos domínios dos valores e selecção das variáveis de entrada. A precisão aumentou com as novas abordagens testadas.

O passo seguinte foi experimentar abordagens com modelos múltiplos de forma a melhorar ainda mais os resultados por comparação com a utilização de um só modelo. É feita uma revisão bastante completa sobre a utilização de modelos múltiplos para regressão. São também apresentadas diversas experiências utilizando a abordagem de selecção dinâmica. A utilização de modelos múltiplos permite melhorar os resultados comparativamente à utilização de um só modelo.

As últimas experiências sobre o segundo problema comparam a abordagem simples, a abordagem com base no conhecimento existente, o melhor algoritmo

(com respectivos parâmetros e técnicas de selecção de instâncias, domínios de valores e variáveis), a abordagem utilizando modelos múltiplos e os tempos de viagem definidos nos horários. Os resultados dão uma pequena vantagem à utilização de modelos múltiplos por comparação com a abordagem baseada no conhecimento existente. No entanto, esta última abordagem necessita de menos dados e é bastante mais rápida de afinar. O método actualmente utilizado pela STCP (a utilização dos tempos dos horários) é competitiva para percursos circulares. No entanto, este resultado pode ser, pelo menos parcialmente, explicado pela forma como esses percursos são controlados. Nos restantes percursos, esta abordagem foi claramente ultrapassada em termos de precisão das previsões (mesmo sabendo que é esta a previsão que os motoristas tentam cumprir).

Tentamos ainda dar respostas práticas sobre a forma de utilizar as previsões dos tempos de viagem nas tarefas de planeamento de empresas de transportes colectivos de passageiros, utilizando o caso da STCP como caso de estudo. O impacto da previsão dos tempos de viagem nos objectivos de negócio, nomeadamente, o aumento do grau de satisfação dos clientes e a diminuição dos custos operacionais, não é alvo deste estudo apesar de ser, naturalmente, o passo seguinte desta investigação.

Abstract

In this thesis we undertook a study in order to know how travel time prediction can be used in mass transit companies for planning purposes. Two different problems were identified: the definition of travel times (1) for timetables and (2) for bus and driver duties. All these studies assume the existence of data on the actual trips, typically obtained from Automatic Vehicle Location (AVL) systems.

The first problem is a well-known problem with several related studies in the literature. Our approach is not analytical. Instead, we have designed and developed a decision support system that uses past data from the same line and representative of the period the timetable will cover. This problem was the least studied.

With respect to the second problem, travel time prediction three days ahead, we focused on how much we can increase in accuracy if we predict travel times for the definition of bus and driver duties as near the date as possible, instead of using the scheduled travel times (STT). The reason for doing this is that, if the increment is important, it is expected to reduce operational costs and/or increase passengers' satisfaction.

In this second problem we used machine learning approaches. However, we started by defining a baseline method (in order to evaluate comparatively the results obtained with more sophisticated methods) and an expert based method using the knowledge we had at the time together with the traffic experts from the STCP company. Then, we tried three different algorithms with reported good results in different problems. They were: support vector machines, random forests and projection pursuit regression. For each of these algorithms, exhaustive tests were done in order to tune parameters. Other tests were done using the three focusing tasks: example selection, domain values selection and feature selection. Accuracy was improved using these approaches.

The next step was to experiment heterogeneous ensemble approaches in order to ameliorate further the results by comparison with the use of just one model. An extensive survey on ensemble methods for regression was undertaken. Several experiments using the dynamic selection approach were executed. Approaches using ensembles have improved results consistently when compared to the use of just one model.

Experiments on the second problem finished by comparing the baseline, the expert based, the best single algorithm (with the respective tuned parameters and focusing tasks), and the ensemble approach, against the use of STT, on various routes. Results gave a small advantage in terms of accuracy to the ensemble approach when compared to the expert based method. However, the expert based approach needs less data and is much faster to tune. The actual

method used by STCP (the use of STT) was competitive for circular routes. However, this result can be explained, at least partially, by how these routes are controlled. On the rest of the routes tested, it was clearly beaten.

We also try to give practical answers in using travel time predictions for the planning of mass transit companies, using the STCP company as study case. The impact of travel time prediction in the business goals, namely clients' satisfaction and operational costs, is not addressed despite it is the natural step forward of this research.

Résumé

Cette thèse porte sur une étude de l'utilisation qui est faite, par les diverses entreprises de transports, des prévisions du temps des voyages dans les planifications des transports collectifs de passagers. Deux problèmes ont été identifiés: (1) la définition des temps de voyages inscrits sur les horaires transmis au public; et (2) la définition des services.

L'ensemble des recherches est fait sur des voyages ayant déjà été effectués. Les données ont été obtenues grâce au système de localisation automatique de véhicule.

Le premier problème n'est pas nouveau, plusieurs ouvrages ont déjà abordé ce sujet. Dans cette thèse, l'approche de la question n'est pas analytique. Autrement, nous avons conçu et développé un système d'aide à la décision utilisant les données historiques représentatives d'une période donnée sur un même parcours.

Le deuxième problème c'est la prévision du temps de parcours à trois jours d'intervalle pour la définition des services. L'idée principale s'agit de quantifier l'augmentation de la précision des prévisions horaires si on utilise pour la définition des services: des voitures et des conducteurs, précis au lieu des temps de transport des horaires publics.

La motivation de cette étude réside dans le fait que plus la précision est grande, plus les coûts opérationnels sont réduits et plus la satisfaction des passagers est garanti.

Pour ce deuxième volet d'étude, nous avons utilisé une méthode d'apprentissage automatique. Il a fallu, tout d'abord, définir une méthode "naïve" (afin d'avoir une référence de base) puis ensuite une autre méthode basée sur la connaissance (celle des experts du trafic routier de l'entreprise STCP et la nôtre). Ces deux méthodes constituent le départ de notre travail de recherche.

Ensuite nous avons testé trois algorithmes d'apprentissage automatique à savoir: machines à vecteur de support, forêt aléatoire et *projection pursuit regression*. Nous avons testé intensément les paramètres pour chacune des méthodes. Nous avons aussi fait des tests sur la sélection des instances, la sélection des domaines de valeurs et la sélection des variables. La précision augmentait à chaque nouvelle approche testée.

Nous avons, ensuite, expérimenté l'approche d'ensembles hétérogènes afin d'améliorer les résultats en les comparant avec la utilization d'un modèle unique. Nous avons fait une révision de l'état de l'art compréhensif sur les méthodes d'ensemble par régression. Puis nous avons présenté les différentes expériences à l'approche de sélection dynamique. L'utilisation d'ensembles a amélioré les résultats en comparaison du modèle unique.

Enfin, les dernières expériences sur le deuxième problème sont la comparaison de l'approche naïve avec l'approche basée sur la connaissance, le meilleur méthode d'apprentissage automatique (avec paramètres respectifs, méthode de choix d'instances et domaines de valeurs et variables), l'approche d'ensembles et des temps de voyages des horaires. Les résultats révèlent un léger avantage avec l'utilisation de l'approche d'ensembles plutôt qu'à l'approche basée sur la connaissance. Par contre, cette dernière nécessite beaucoup moins de données et est donc plus vite configurée que celle des ensembles. Le méthode qui est actuellement utilisée par la STCP (les temps de voyages des horaires) est d'ailleurs compétitive pour les parcours circulaires. En revanche, pour les autres types de parcours cette approche est clairement mauvaise. Nous avons aussi fait un essai sur l'utilisation des prévisions des temps de voyages pour la planification dans les entreprises de transports collectifs en utilisant l'entreprise STCP comme cas d'étude.

L'effet de l'utilisation des prévisions des temps de voyages sur la réduction des coûts opérationnels ainsi que sur l'augmentation du niveau de satisfaction des passagers ne sont pas encore étudiés mais ils le seront, bien évidemment, dans la deuxième phase de cette recherche.

Agradecimentos

Começo por agradecer aos meus orientadores, Jorge Freire de Sousa e Alípio Jorge, todo o apoio e profissionalismo com que orientaram esta tese. Neste grupo incluo o Carlos Soares que, não sendo meu orientador, também cumpriu essa tarefa com grande benefício para a qualidade do trabalho desenvolvido. Ao Alípio e ao Carlos agradeço ainda o terem contribuído para uma melhoria significativa na qualidade da minha escrita científica.

À STCP agradeço a participação nesta parceria que, espero, tenha tido para ela pelo menos tantos benefícios como os que teve para mim. Agradeço em particular aos dois conselhos de administração que conheci durante este período as condições de trabalho oferecidas. Mais uma vez, ao Jorge Freire de Sousa, na sua qualidade de vogal do conselho de administração da STCP, a sua grande disponibilidade. Agradeço também o bom acolhimento por parte do departamento de operações. Um agradecimento especial aos engenheiros José Miguel Magalhães e Pedro Gonçalves toda a disponibilidade manifestada.

Ao Mykola Pechenizkiy agradeço todos os conselhos dados que foram de grande utilidade para validar e perspectivar o trabalho até então desenvolvido.

Um obrigado muito especial às bolseiras de investigação Eva Duarte e Francisca Castro Martins a dedicação e o empenho sempre manifestados.

Ao DEI a confiança depositada na minha pessoa.

Ao CICA as facilidades dadas para a realização dos testes computacionais.

Ao gabinete de tradução Phala a revisão do inglês da tese.

À Aude a revisão amiga do resumo em francês.

À Henriqueta e à Lia o nunca se terem esquecido de mim.

Vitória (1454 km), Ferraz (632 km), Eduardo (580 km), Paula (492 km), Nuno (491 km), Barros Basto (412 km), Maria João (369 km), Vítor (264 km), José Eduardo (258 km), Yá (203 km), Paulo (174 km), Lígia (164 km), Joana (156 km), André (152 km), Rachel (152 km), e os demais que tantos são. Eles bem sabem os caminhos que trilho, mas mal sabem o bem que me fazem ...

Às matrecadas em casa do Luis. Às paisagens enche-alma. Aos longos cafés com o João. Ao 'tá-se bem' da Paula. À minha família. Aos serões gastronómicos. Ao riso das crianças dos outros. Às árvores de minha casa que cada vez estão mais bonitas. Aos longos almoços Domingueiros. Ao companheirismo do Ferraz. E do Barros Basto. E de tantos outros. Ao jogo de bola da Lira. À cumplicidade da Vitória. Aos meus amigos. Às 2000 cabras de Covas do Monte. Ao esconde-esconde do Bernardo e do Pedro. Aos concertos. Aos filmes. Às piscinadas. Aos Gaiteiros as gaiteiradas. Ao nosso país bonito. Aos mimos caseiros. Às noites de refúgio. Ao refúgio das noites.

Este trabalho foi apoiado pela Fundação para a Ciência e a Tecnologia (projecto POCTI/TRA/61001/2004).

Contents

List of Figures

List of Tables

Part I

The problem

Chapter 1

Introduction

In the last two/three decades, passenger transport companies have made important investments in information systems, such as Automatic Vehicle Location (AVL), automatic passenger counting, automated ticketing and payment, multimodal traveller information systems, operations software and data warehouse technology, among others. As a consequence of this effort in Advanced Public Transportation Systems (APTS), passenger transport companies have been able to collect massive data. As in other areas of activity, all this information was not proving to be particularly helpful in supporting companies to accomplish their mission in a significantly better way. The quote "we are drowning in information and starving for knowledge" from Rutherford D. Rogers, a librarian from Yale University, summarizes these moments in the company lives. The Sociedade de Transportes Colectivos do Porto, SA (STCP), the largest urban public transport operator in greater Oporto, Portugal, has made important investments in APTS in the last few years. In 2003, the Sistema de Apoio à Exploração e à Informação (SAEI), a Bus Dispatch System, began to store the first data regularly, although in an experimental period. The SAEI was able, among other functionalities, to incorporate AVL data, namely the location given by radio every 30 seconds and all the relevant data about the bus trips. In 2003, the operations department of the STCP began the intensive data collection process. Since then, the STCP has been making an important effort in data consolidation and data exploration. This thesis attempts to give a contribution, albeit small, towards this effort of feeding STCP with knowledge, in particular its department of operations.

The goal of this thesis is to study how to better predict travel time for operational purposes. The final goal is to reduce operational costs and increase client satisfaction by improving operational planning and service reliability due to the use of more adequate travel time predictions (TTP).

This chapter begins with the presentation of the problem. Then, in Sect. 1.2, we state the objectives of the thesis followed by the presentation of the methodology used in Sect. 1.3. Finally, in Sect. 1.4, the structure of the thesis is presented.

Table 1.1: Extra work from 2004 to 2007 in hours and thousands of €.

	2004	2005	2006	2007	
All employees	157695	168184	174187	103124	In hours
Crew staff	135723	152586	154515	84557	
All employees	1220	1320	1379	890	In thousands of €
Crew staff	998	1158	1185	704	

1.1 Description of the problem

The STCP company serves Greater Oporto, an area with 1.3 million inhabitants using a network of 533 km. In 2007, they offered 83 lines, 51 of which are bus lines operated using the company's own resources. In 2007, they fulfilled 30 million km using a working force of 1623 employees (at the end of of the year), with 65.4% of them being crew staff. In the STCP company, the work force costs are an important component of the operational costs. In 2007, this ratio was 46.0%, the work force costs represented 39.6% of the total costs [Gomes *et al.*, 2008].

Although costs with extra time represent typically from 2% to 4% of the total costs with the personnel (2.32% in 2007), this value is, in absolute terms, an important one (table 1.1) mainly because there is the perception that this value is, at least in part, due to an inefficiency in the construction of the crew duties because of the use of long horizon TTPs. These predictions are the ones used to construct the timetables. The timetables are typically used for several months. It is expected that by using a shorter horizon for the predictions, they can be more accurate and, consequently the duties can be more adapted to the reality.

Last, but definitely not least, client satisfaction depends on the reliability of the service. More accurate travel times for timetable construction are expected to increase clients' satisfaction [Strathman *et al.*, 1998].

1.2 Objectives of the thesis

The question we try to answer in this thesis is: is it possible to improve the accuracy of travel time prediction/definition for planning purposes in comparison with the present approach in use at STCP through the possible reduction in the horizon of prediction?

This goal assumes that increasing the accuracy of travel time used for planning purposes will reduce operational costs and will increase clients' satisfaction. It is not an objective of this thesis to quantify the reduction in operational costs or the increase in clients' satisfaction. The only objective is to prove that we are able to predict/define travel times for operational purposes more accurately. This objective will be better defined in Sect. 2.4, after the introduction to main concepts of operational planning at a mass transit company.

In addition to this, some other objectives were defined but to better understand them some previous thoughts are needed. This thesis is obviously on applied research in the sense that the goal is to use tools to solve a given problem. Generally, these tools already exist. Our objective in this thesis is

not to develop new tools. Instead, we aim to fully understand the tools which are more adapted to solve the particular problem and to be able to generalize results obtained with this particular problem. In other words, it is an important issue to contextualize the lessons learned and to explore them in order to make these lessons available to a broader research community. The effort in contextualization and generalization of the lessons learned is an effort to which all researchers must be committed.

The secondary objectives of this thesis are:

1. To contextualize each of the used methodologies in the corresponding state of the art of the problem to be solved;
2. To contextualize each of the used methodologies in the respective state of the art of the methods being used;
3. To design and develop the necessary model(s) to accomplish the main objectives of this thesis;
4. To implement a prototype using the relevant aforementioned model(s);
5. To generalize, when appropriate, the results obtained using the different methodologies to other problems.

1.3 Methodology

The methodology used in this thesis for approaching the travel time prediction problem follows the CRISP-DM (CRoss-Industry Standard Process for Data Mining) model [Shearer, 2000]. This model was proposed in 1996 by four leading data mining companies at the time. Although CRISP-DM is defined as a data mining model, it is easily adapted to projects involving forecasting models, in general. It has six phases (Fig. 1.1, adapted from [Shearer, 2000]):

1. Business understanding: firstly, the objectives of the project are defined from the business point of view. Then, these objectives must be defined in a quantitative way, in order that they can be measured.
2. Data understanding: this phase comprises the data assessment; the first analysis of the data, namely by visual inspection or by the use of descriptive statistics; and the quality assessment of the data.
3. Data preparation: it comprises the necessary tasks to obtain the final data set, namely the definition of the features and examples to use, the substitution of NULL values if needed, feature construction, among other necessary data preparation tasks.
4. Modelling: in this phase the specific techniques (algorithms) are selected, the experimental setup is defined, the parameter sets are tuned and the final results are assessed and compared to each other.
5. Evaluation: in this phase, how the results meet the business objectives is firstly evaluated, secondly the process is reviewed in order to adjust some tasks, if necessary, and finally it is decided whether the project should

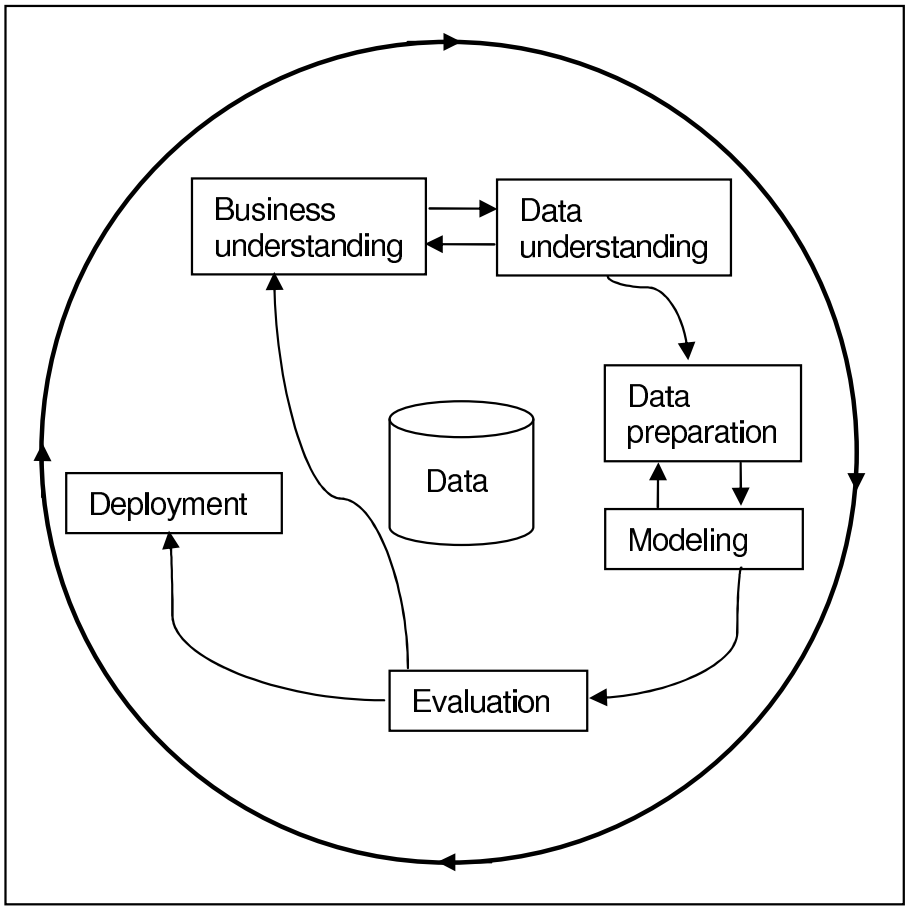

Figure 1.1: Phases of the CRISP-DM reference model.

be finished and, consequently, whether the project should pass to the deployment phase or if it is necessary to do further iterations or to set up new projects.

6. Deployment: it is the final phase, where the project is deployed to be used by the end user, the tasks for monitoring and maintenance are defined, the final report is written as well as the evaluations and recommendations for further improvements.

The CRISP-DM model is fully described in [Chapman *et al.*, 2000], namely the description and outputs for all the tasks of each phase.

1.4 Structure of the thesis

The remainder of this thesis is organized into four parts as follows:

- Part I: The problem

  - Chap. 2 describes the problem. It comprises a brief overview of the main planning tasks in a mass transit company, how those tasks are accomplished in STCP, and how travel time prediction can be used to improve it, namely, the definition of both timetables and duties. Then, a brief state of the art review on TTP is described, addressing these two different problems. In order to better define the problem, a brief overview on evaluation and deployment issues is carried on. Then the problem is restated. Finally, the data is described and analyzed. This chapter comprises the business understanding, data understanding and data preparation phases of the CRISP-DM model.

- Part II: Travel times for timetabling

  - Chap. 3 presents a decision support system for timetable adjustments that allows the test of different scenarios, namely, different frequencies for the offer of trips, scheduled travel times and slack times. How to use this tool for different situations is explained, namely short and large headways, where headway is the inverse of the frequency. It comprises the modeling phase of the CRISP-DM model. However, this chapter does not use data mining approaches. Consequently, the modeling phase does not use the CRISP-DM model but the evolutionary prototype as described in [Nunes *et al.*, 2000].

- Part III: TTP three days ahead (for the definition of duties)

  - Chap. 4: after a first discussion about the model that better addresses the problem, inductive learning methods versus time series forecasting methods, the main data mining concepts on supervised learning and particularly in regression are introduced, namely the description of the main issues on regression, a brief review of the three focusing tasks (example, feature and domain value selection) is made and the main inductive learning algorithms for regression are presented. This review helps to accomplish the second secondary objective defined in Sect. 1.2, namely to contextualize the different methodologies used throughout this thesis in the corresponding state of the art. This is part of the CRISP-DM modeling phase.
  - Chap. 5 describes the modeling phase of the CRISP-DM model. The regression algorithms (support vector regression, random forests and projection pursuit regression) and the input variables to use are selected, the experimental setup is defined, a baseline and an expert-based methods are presented and tested, the parameter sets for the regression algorithms are tuned and some experiments using different focusing approaches are carried on. For example selection, two different approaches are tested and compared to the use of all training examples: the selection of examples from equivalent days and the selection of examples that fall into the same leaf node of a CART decision tree as the test example. For domain value selection, two different domain values (numeric and symbolic) are tested for a particular input variable, the weekday variable. For feature selection, the irrelevant features were detected using the RReliefF algorithm

[Robnik-Šikonja and Kononenko, 2003]. Then, five pre-selected subsets of the original feature set were tested using random forests.

- Chap. 6: like Chap. 4 it is a review work, in this case on ensemble methods for regression. The experiments described in Chap. 5 motivated the study of methodologies that might improve our initial results. We have decided to use ensemble methods. In this chapter, we present a review work of ensemble methods for regression. Here, we make the first comprehensive state of the art review on ensemble methods for regression. It comprises main definitions and concepts. Different state of the art methods are described for the three phases of the ensemble learning model: generation of the pool of models, ensemble selection by pruning the pool, and integration of the results from each model in the ensemble in order to obtain the final ensemble prediction.
- Chap. 7 describes another modeling phase according to the CRISP-DM model. Different tests on the dynamic selection approach of ensemble learning are described. Firstly, the methods used for each one of the three phases of the ensemble learning model described in Chap. 6 (generation, pruning and integration) are described. In the generation phase three different pools are described. In the pruning phase, two different search algorithms are presented as well as two evaluation functions, according to the taxonomy presented in Sect. 4.3.1. In the integration phase, different approaches for each task of the dynamic selection approach are described, namely: the selection of similar data, the selection of models from the ensemble, and the combination of their results to obtain the final ensemble prediction. Three different experiments are carried out: testing the different methods to obtain similar data; testing the impact of the size of the data set used for pool generation on the results of the ensemble; and testing ensembles selected from the pool using pruning algorithms with different evaluation functions. Additionally, in all three experiments: different sizes for the similar data set are tested; different methods for the selection of models from the ensemble are tested as well as the combination of their results; the results for the base learners and the constant weighting function, simple average are shown.
- Chap. 8 evaluates comparatively in new routes the different approaches tested in the previous chapters namely, the baseline and the expert based methods, the use of the best algorithm with respective parameter set from the pool, an ensemble approach using the dynamic selection method and the approach in use at STCP. A final statistical evaluation of the experiments on travel time prediction three days ahead is presented. This chapter comprises the evaluation phase of the CRISP-DM model.

- Part IV: concluding remarks
  - Chap. 9 concludes this thesis summarizing the work done and describing how the initial objectives were accomplished. Finally, ideas for future research are pointed out.

Chapter 2

Description of the problem

The idea of this thesis, to improve the accuracy of travel time predictions for operational purposes in order to reduce costs and/or increase client satisfaction, is appealing and easily understandable. However, the way the predictions can be useful to a mass transit company is neither simple nor obvious. This happens because the operational planning processes are complex, especially at larger companies as in the case of STCP, the case study used throughout this thesis.

This chapter starts with the description of the main operational planning tasks (Sect. 2.1) followed by a in-depth analysis of travel time prediction - TTP (Sect. 2.2). Then, a first overview of the evaluation and deployment phases of the CRISP-DM model is outlined (Sect. 2.3) in order to make decisions on how to follow up (Sect. 2.4). Finally, the available data is described and analyzed (Sect. 2.5).

2.1 Operational planning

Operational planning in mass transit companies is, by its own complexity, a well established research area: all the tasks are well described in the literature. In this section, we firstly describe them for a medium/large size bus company (Sect. 2.1.1), then we describe how they are accomplished at STCP (Sect. 2.1.2).

2.1.1 Main concepts of operational planning

The main tasks of operational planning are usually done in the following sequential way [Ceder, 2002; Lourenço *et al.*, 2001; Dias, 2005]:

1. The network definition: it is, obviously, a planning task for the long/very long term. It comprises the definition of the lines, routes and bus stops. We define route as an ordered sequence of directed road stretches and bus stops. Lines are a set of routes, typically two routes that use roughly the same road stretches but in opposite directions.

2. The trips definition: it is a medium term task, with an horizon much shorter than the network definition. There are typically two different methods for trip definition: (1) headway-based, defining the time between

two successive trips on the same route [Vuchic, 2005]; or (2) schedule-based, defining timetables by explicitly setting the departure time and the time of passage at the main bus stops. The supply of trips is defined by route even if they are articulated between groups of routes/lines [Ceder *et al.*, 2001].

3. The definition of the duties of the drivers and buses: they are medium term (several months) tasks. The goal of both tasks is to define duties. A duty is the work a bus / driver must do. When a duty is defined, in both cases, we do not know which driver or bus will do it. We are just making a logic assignment. The case of bus duties is much simpler than driver duties for obvious reasons: drivers must stop for lunch, cannot drive every day of the week, etc., i.e., they have many more constraints than buses. According to [Dias, 2005], "each driver duty is subject to a set of rules and constraints defined by the government legislation, union agreements and some internal rules of the company". Typically, bus duties are defined before drivers duties.

4. The assignment of duties: it is the task where the driver duties are assigned to drivers and bus duties are assigned to buses. We are now making a physical assignment. Assignment for driver duties is more complex than for bus duties, for similar reasons to the ones explained above. The assignment of driver duties to drivers is called rostering. It can vary significantly from one company to another.

The process described is typical for a large mass transit company. Smaller companies typically use simpler approaches [Dias, 2005].

2.1.2 Operational planning at STCP

In this section we describe how operational planning is accomplished at STCP.

The bus network

The bus network at STCP was completely redesigned in 2007. This event, which happens very rarely in the life of a public transport company, was a consequence of the creation and growth of the Metro do Porto network. Metro do Porto is the light train company from Greater Oporto. At the end of 2002 it commenced commercial operations on the first line. In September 2005 it finished the so-called first phase network composed of 4 lines [Porto, unknown]. In order to adjust its bus network to the new reality of the public transport supply in the region, STCP redesigned its bus network using a complementary philosophy in relation to the Metro do Porto network. The new STCP network has fewer and shorter bus lines, 51 operated using the company's own resources, with, on average, shorter headways.

The definition of trips

In all Portuguese bus companies, trips are defined by schedule. At STCP, the supply is different for Saturdays, Sundays/Holidays and working days. The supply for working days is also different for summer holidays and for special

days, namely All Saints' Day. Although the supply of trips is done by schedule, the company favors stable headways, i.e., they try to offer trips at constant time lags in order to facilitate the creation of routines for their clients.

The definition of trips, as well as the definition of duties and the rostering process, is done with a decision support system named GIST [Lourenço *et al.*, 2001; Dias, 2005; Ferreira, 2005], which is used in the main mass transit companies in Portugal.

The definition of duties

The bus scheduling is done for a set of lines (typically sharing part of the route or using the same terminal node). This phase consists of the definition of the bus duties (named running boards) in order to cover all the trips. The algorithms used try to minimize both the number of buses needed and the running costs. Each running board can be seen as a sequence of trips separated by slack times, where slack time is the scheduled time between trips in order to accommodate delays (Sect. 2.2.1). It includes not only the trips offered to the public but also the connection trips. Fig. 2.1 shows an image of the GIST system representing running boards. Each running board has a different color. It is composed of a set of trips (with respective slack times) and connection trips, typically beginning and ending the duties for connection with the depot (the dashed lines). The main bus stops are at the vertical axis (they are ordered according to their position on the route) and the hour of the day is at the horizontal axis (at the top of the image). That is why this kind of chart is called a time-distance one [Vuchic, 2005].

Then, the driver scheduling is defined. Each driver schedule is a set of driver duties within the existing constraints and rules. Each driver duty is a sequence of pieces-of-work where each one is part of a running board between two relief opportunities. A relief opportunity is a bus stop where drivers can be replaced. At this phase the duties are not assigned to the drivers. This phase is fully described in [Dias, 2005], including the approaches used by STCP. Fig. 2.2 shows an image of the GIST system representing driver duties. Each line represents a different running board using a horizontal representation (instead of the time-distance representation used in Fig. 2.1). Driver duties are represented with different colors and numbers. The pieces-of-work are easily identifiable because they are the stretches between two consecutive relief opportunities (the vertical black traces).

The rostering process

After being defined, the duties are assigned to the drivers in the rostering phase. In this phase, some drivers are organized into groups, named rostering groups. Each group has a set of drivers and a set of duties defined for a set of lines. This phase has three different stages according to the planning horizon:

1. In the long term rostering, assignments are defined in order to guarantee that personal constraints (days off and holidays) and skills of the drivers (such as skills for driving different types of vehicles) are respected.

2. In the regular rostering phase the assignments are updated according to the drivers' availability. These updates are typically done in a daily basis.

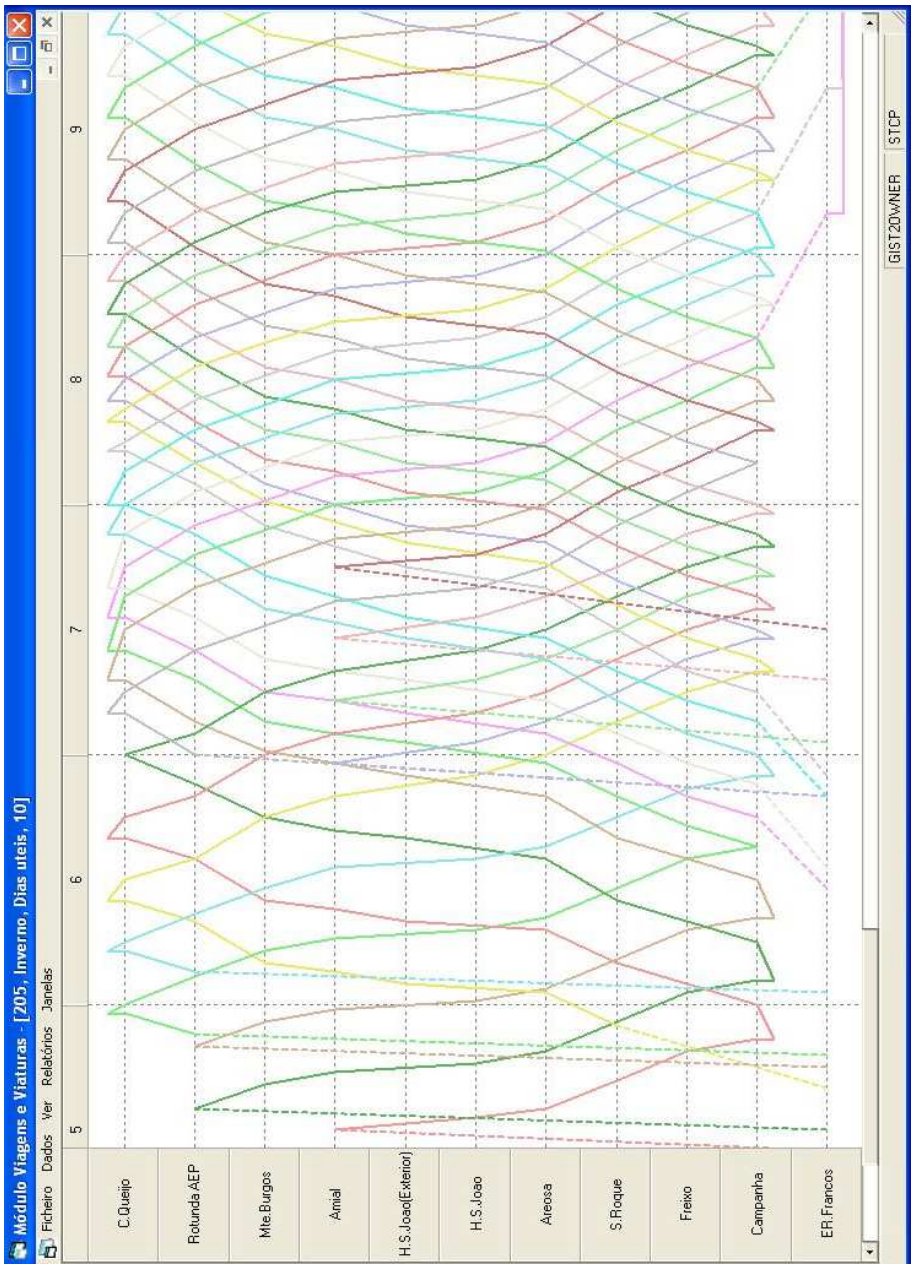

Figure 2.1: Running boards identified by colors (image from the GIST system).

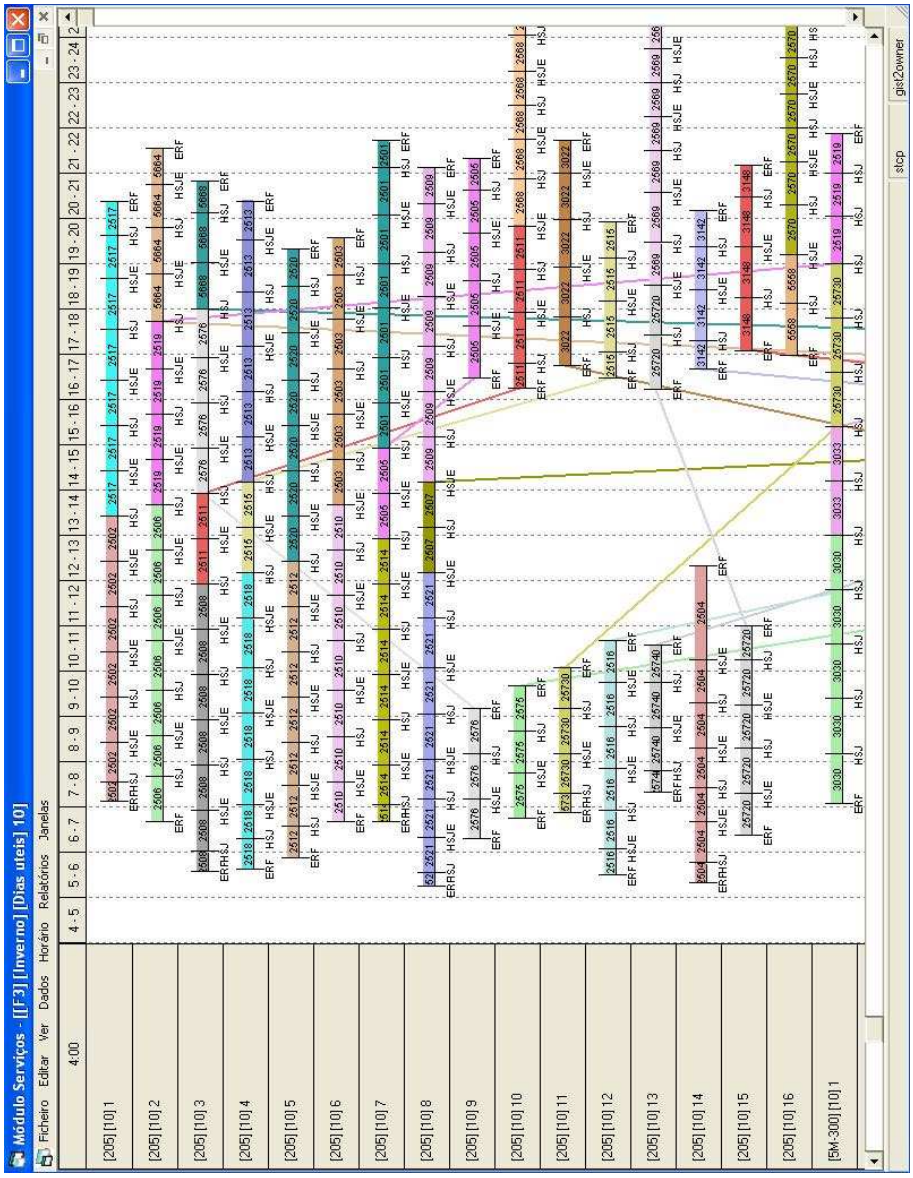

Figure 2.2: Driver duties identified by colors and duty numbers (image from the GIST system).

3. In the post rostering phase last minute changes are settled. The actual work is registered.

The rostering process is fully described in [Ferreira, 2005], including the approaches used by STCP.

2.2 Travel time prediction (TTP)

According to [Strathman *et al.*, 2001], in 1978 Abkowitz suggested the classification of the methods to improve transit service reliability in three groups:

- Priority: are those methods that give priority to transit vehicles, such as, bus lanes and traffic signal prioritization, among others;
- Operational: are those methods previously described, namely, route restructuring, trip definition, duties definition and rostering processes;
- Control: take place in real time and include vehicle holding, short-turning, stop-skipping and speed modification.

TTP can be used for operational and control methods. We study just the operational issues of TTP.

We begin this section by discussing main concepts and definitions of travel time. Then, we make a brief description of TTP at STCP detailing TTP for timetabling and for duties definition. We conclude by briefly reviewing the state of the art in TTP.

2.2.1 Main definitions of travel time

In order to clarify concepts we present the main definitions related to travel time used throughout this thesis:

- Travel time: is the time of a trip, from terminal to terminal. However, in the state of the art review on TTP (Sect. 2.2.3), we use the travel time definition given by Turner et al., "the time necessary to traverse a route between any two points of interest" [Turner *et al.*, 1998].
- Slack time: is the time that, according to the schedule, the bus is at the terminal between the end of a trip and the beginning of the next one.
- Cycle time: is the interval between the two consecutive times a bus in regular service leaves the same terminal (adapted from [Vuchic, 2005]).

These times are represented in Fig. 2.3 using a time-distance chart for the most common situation, i.e., lines with two routes (go and return). At STCP just four lines, the circular ones, do not have two routes.

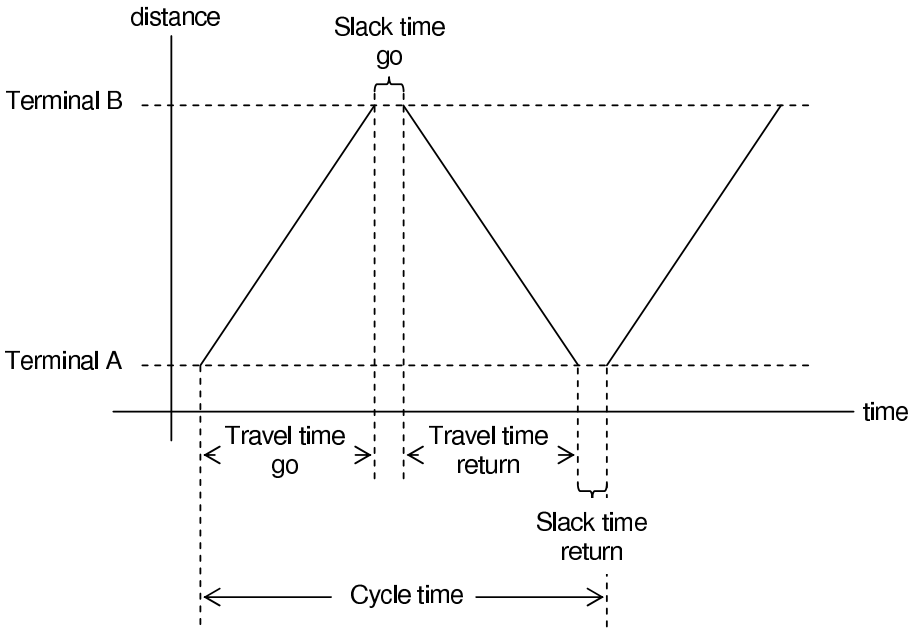

Figure 2.3: Time-distance chart on different concepts of travel time.

2.2.2 Travel time prediction at STCP

There are four different moments where TTP is or should/could be used at STCP (as well as in other medium/large mass transit companies):

1. To define timetables;
2. To define duties for both buses and drivers;
3. To be used in the control center to anticipate intra-day corrections;
4. To inform the public of the bus passing time at a specific bus stop for the short run.

At STCP, the first situation is accomplished as described below, the second situation uses the travel times defined in the timetables, the third is not accomplished, and the last one is already implemented using Kalman filters [Kalman, 1960] [1]. STCP informs the public by request using Short Message Service (SMS).

As we said previously, we only study the operational issues of TTP, i.e., for timetabling and for duties definition. These problems are discussed in detail next.

Travel time prediction for timetabling

Schedules are typically defined for periods of several months and travel times are always the same while these schedules are in use.

[1] The algorithm used was not implemented at STCP and, consequently, we do not have information on the details.

The way travel times to be used in the timetables are defined at STCP changed in March 2007. Before this date, travel time was defined by analyzing two or three days considered empirically representative. Fig. 2.4 shows a comparison between the planned travel times and their actual values for the trips made on working days during school time for the first 7 months of 2004 using the described approach. The planned travel times are the red ones. The large dispersion of travel times for the same departure time is apparent (the vertical width of the data points cloud). The lack of criteria to define the planned travel times is also apparent. In fact, at the beginning and end of each day, most trips are ahead of the scheduled travel time, while in the remaining periods most trips are behind the scheduled times.

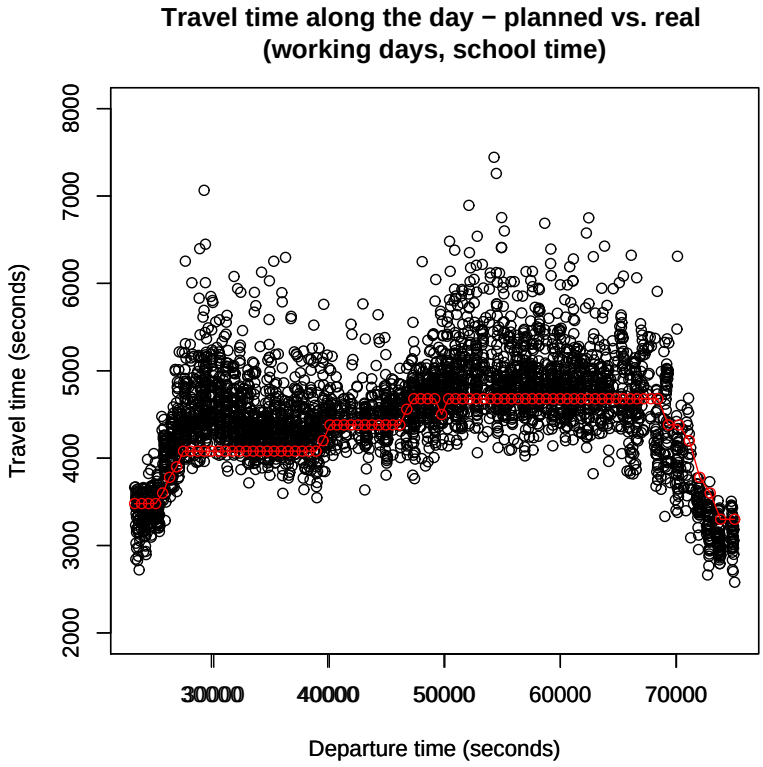

Figure 2.4: Travel time along the day for the trips defined in the schedule of route 78-1-1 for working days during school time for the first half of 2004.

Approximately in March 2007 a new tool appeared in order to support travel time definition. This tool, developed at STCP, uses the data of the actual trips obtained from the SAEI system (Sect. 2.5.1) to calculate the average travel time for each scheduled trip of a given line. This time is used as a reference when the scheduled travel times are defined during the process of scheduling definition. Fig. 2.5 presents an analysis for line 205, for the period of 7 to 11 of April 2008 between 9.00 and 12.00. For each scheduled trip (the dashed lines) the actual average time (the full lines) is presented.

The department of operations aims to update schedules every three months at least for the most busy lines. However, in March 2008, i.e., one year after

the introduction of this new procedure, there was still a meaningful number of lines that had not been analyzed. Only around 60% of the necessary timetables for working days during school time have been reformulated. All the remaining timetables have not been reformulated at the time.

When the trips are defined by schedule, the respective slack times are also defined. The trips are usually defined so that the cycle time is a multiple of the scheduled headway.

It is important to notice that the new approach uses an approximate mean travel time value for each trip, giving equal weighting to advances and delays. It is expected that, using this approach, global error will fall but the percentage of trips ahead of schedule will be roughly the same as delays. For lines with lower frequency, the impact of a bus passing ahead of schedule from the clients' point of view, is higher than for lines with higher frequency, i.e., if the average (or the median) from past travel times is used, the largest is the headway (lower frequency) the lowest is clients' satisfaction [Zhao *et al.*, 2006]. Furthermore, this tool does not give any information on travel time dispersion.

In short, the definition of travel times for timetabling is a two-objective problem where one of the objectives is to minimize the cost for the company and the other one is to minimize the cost for the clients, usually measured as the time passengers must wait at the bus stop for the bus. According to [Strathman *et al.*, 1998], the practice is to consider that the difference between the actual and the scheduled travel time is acceptable for the passengers when it is inside the interval $[-1, 5]$ (in minutes). From this point of view, the travel times used in timetables should not be seen as a prediction but, instead, as a compromise between these two objectives: costs for both the company and the passengers.

Travel time prediction for duties definition

The definition of travel times for timetabling and for duties definition are different problems because not only have prediction horizons typically different, but also because they have different objective functions to optimize.

The prediction time horizon for duties definition depends on how flexible the definition of duties is. The flexibility of duties definition depends on:

- The process itself: at STCP, in order to use TTPs for duties definition, it is necessary to re-define the duties for both buses and drivers since different travel times have consequences on how the trips are ordered (when defining running boards) and, consequently, on the length of pieces-of-work (when defining driver duties).

- The rostering schemas: at STCP, rostering schemas are defined in such a way that the process only restarts after a certain period. For example, in order to guarantee that all drivers can have a day off on Sundays, the rostering process has a rotation schema to assure this and other internal rules. The process only restarts from the initial period after a certain number of weeks.

- The company agreements: at STCP there are four unions for the drivers and there is more than one company agreement, i.e., a set of compromises between the company and the unions. One of the rules is how many days in advance the drivers should know their duties. Nowadays, the

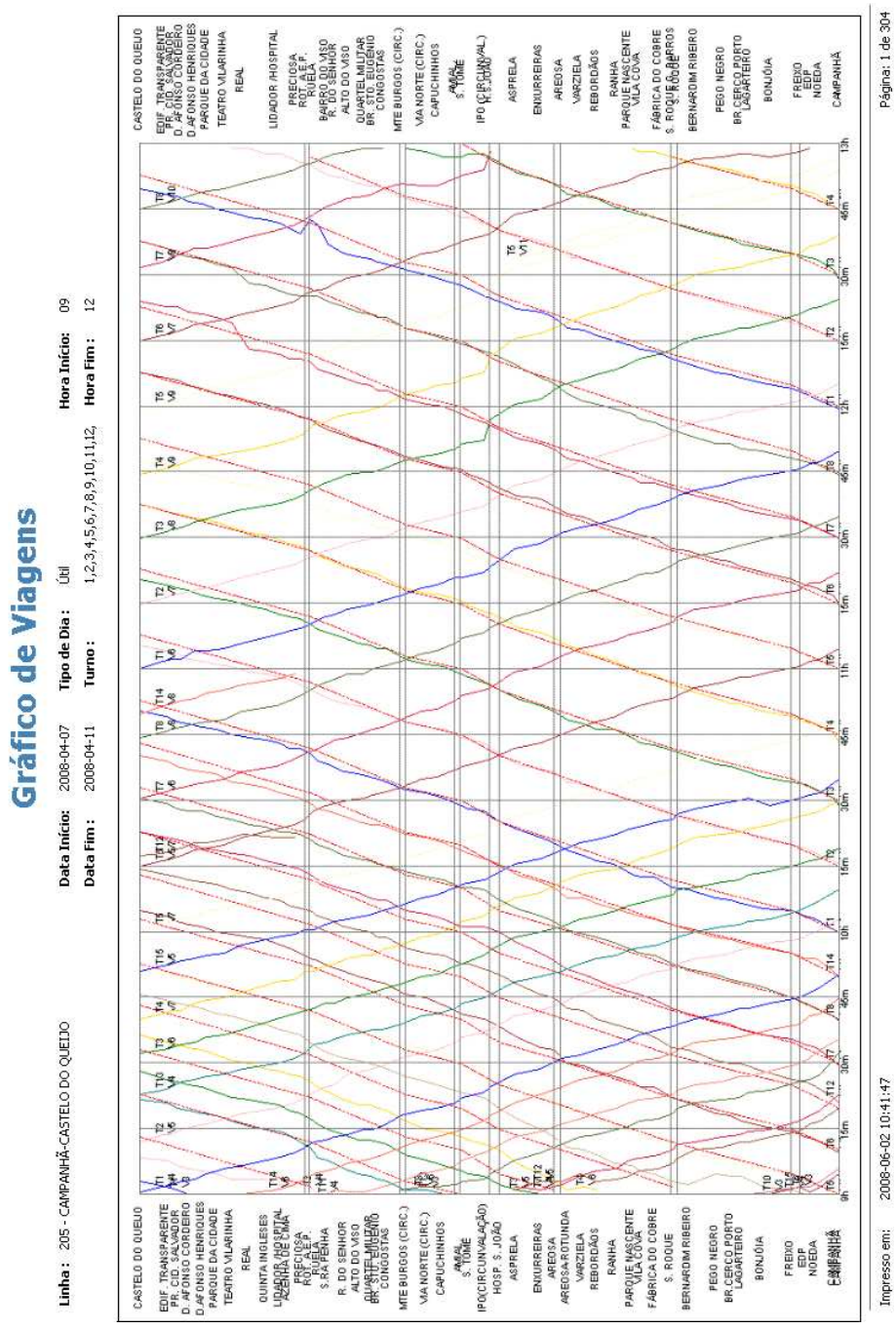

Figure 2.5: A tool to support trip analysis (the source of this image is a software application owned by STCP).

agreements define a limit of two days to announce which duty is assigned to each driver. The drivers in the rostering groups (Sect. 2.1.2) are able to know their duties several weeks or even months in advance. However, the rostering schema is only confirmed two weeks (with exceptions) before the duty day. The remaining drivers, i.e., the ones that are not in rostering groups, only know their duties (for the first time) two days in advance.

From the resources optimization point of view, namely the definition of duties for vehicles and drivers, the best TTP is the one that minimizes the generalization error, typically the mean squared error (Sect. 4.2.3).

It is important to mention that solving the problem of TTP does not solve the problem of how to use those predictions. The aim of this study is to predict travel times as accurately as possible. If the prediction of travel times with a shorter prediction horizon can increase the accuracy of those predictions, then the following step is to study how to make the process for duties definition more flexible. Despite the fact that this second part is not an objective of this thesis, it is necessary to study how short the prediction horizon can be, a subject to discuss in Sect. 2.3.

2.2.3 A brief state of the art review

There is extensive literature on TTP. However, TTP can be used to answer different questions, such as: how long do we need to go to the town center right now; how long does a truck need to do a long delivery tour departing tomorrow; or even, how long does a bus need, on average, to make a scheduled trip? Different approaches are needed to answer each of these questions.

In order to better understand some of the concepts usually found in the literature on TTP, we firstly discuss some of these concepts roughly following [Lint, 2004]:

- Prediction horizon: it is a common practice to classify the prediction horizon as short when it does not exceed 60 minutes [Ben-Akiva *et al.*, 1992]. According to [Lint, 2004] accuracy is smaller as the prediction horizon gets longer. We only study predictions a few days ahead, clearly long-term ones. The prediction horizon is strongly related to the input data. A simple example is that current traffic flow is relevant for short term prediction but it is not for predictions three days ahead. The prediction horizon is also related to the business goal. Advanced Traveller Information Systems (ATIS) are systems that, among other functions, inform transit users of TTP. Consequently they use short term predictions. However, TTP to plan long distance deliveries uses long term (several days) prediction horizons. The vast majority of the studies on travel time prediction that exist in the literature are for the short term.

- Modeling approach: (1) some methods are data-driven in the sense that they build the models from the data, (2) some others are model-based because travel time is seen as part of a traffic-flow simulation model to predict traffic conditions in the route of interest, and (3) a last group are instantaneous ones [Lint, 2004] because they measure some input variables to obtain instant travel time predictions.

- Data collection methods: the way data is collected has a direct influence on the available data. There are many different methods of collecting data [Turner *et al.*, 1998], namely, test vehicles, probe vehicles, license plate matching, or extrapolation methods (such as the loop detectors), among others.

In the remainder of this section, we discuss related works on TTP a few days ahead (TTP horizon for duties definition is discussed in Sect. 2.3) and on the specific task of travel time definition for timetabling that is necessarily a task with a horizon of several months. The definition of travel times for timetabling also has a long duration horizon, i.e., while travel times to be used for driver duties definition are just for one trip occurring on a specific day, for timetabling it is for a scheduled trip, i.e., a trip that will occur in a period of several months (while the timetable is being used).

Related works on TTP for timetabling

An important difference between the existing approaches on timetable creation concerns the variables used. These variables depend on the purpose. If, for instance, a timetable for a new line is needed, a variable such as the population density of the served area is important [Ceder, 1986]. However, if the goal is to make small adjustments to the timetables, this variable is not relevant. We describe the variables for the latter case, i.e., for regular planning tasks (line creation is a sparse event in a mass transit company). Some typical variables are: scheduled travel time (STT), slack time (SlT), scheduled headway (SH), fleet size (N) and scheduled departure time. Let us assume that SCT is the scheduled cycle time,

$$SCT = STT_g + SlT_g + STT_r + SlT_r, \tag{2.1}$$

where the indexes g and r represent the go and return trips (Fig. 2.3 represents graphically this equation). For urban areas, the departure times are usually defined by headway instead of irregularly spaced. Irregularly spaced departure times are typically used for long distance trips or trips in rural areas. We focus on the definition of the timetables' departure times by headway as it is used at STCP.

According to [Zhao *et al.*, 2006],

$$SCT = N \times SH. \tag{2.2}$$

Fixing N, and assuming that travel times are exponentially distributed, slack times can be optimized [Zhao *et al.*, 2006]. Using this approach, the shorter the slack time is, the shorter the scheduled headway is. The objective function used is the passengers' expected waiting time function,

$$E[\omega] = \frac{SH}{2} \times \left(1 + \frac{2Var[l]}{SH^2}\right), \tag{2.3}$$

where l represents the delay of the bus and $Var[l]$ the variance of l. This function assumes that passengers arrive uniformly at the bus stops, which is acceptable for short headways. Headways equal or shorter than 10 minutes are

usually defined as short headways in the literature [Zhao *et al.*, 2006]. Formally, the probability density function for the passengers' arrival at the bus stop is,

$$f_A(t) = SH^{-1}, \tag{2.4}$$

where t is the time of the passenger arrival at the bus stop.

However, the authors argue that by using the function defined in [Bowman and Turnquist, 1981] for the passengers' arrival at the bus stops, it is possible to adapt the solution to problems with large headways. In this case,

$$f_A(t) = \frac{exp(U(t))}{\int_0^{SH} exp(U(\tau))d\tau}, \tag{2.5}$$

where $U(t) = aE[\omega(t)]^b$ is the utility function, $E[\omega(t)]$ is the expected waiting time of an arrival at time t and a and b are constants that must be defined from empirical data. In any case, the derivation of the passengers' expected waiting time for large headways has not yet been done.

Using an economic perspective, [Carey, 1998] defines as objective function, the cost expressed in terms of STT, lateness and earliness unit costs. Using this approach it is possible to define the optimal STT and SlT for given ratios between the STT unit cost and both the lateness and earliness unit costs. Another contribution of this work is the inclusion in the model of the effect of relaxation when the SlT is larger, i.e., it is known that when the schedule is tight, the actual travel time is shorter than when it is large. Carey calls it the behavioral response. What Carey shows is that the timetable definition should be neither too tight, to avoid delays in departures, nor too large, to avoid behavioral inefficiency.

In all the studies on the definition of travel times, it is not explicit how global cost is defined, i.e., the cost for the passengers and for the company. In Carey's approach, these costs are implicit in the unit costs, but the author does not explore how to estimate them (it is not the goal of the paper). The work by Zhao et al. uses just the passengers' cost, i.e., the expected time the passengers must wait at the bus stop. The operational costs are not considered.

The above mentioned approaches assume that the purpose is to adjust STT and SlT, i.e., the timetable is already defined (even if roughly) and the goal is just to tune it. However, there are several studies on methods for the creation of bus timetables, with different purposes. For instance, [Ceder, 1986] addresses the definition of the frequency related to the problem of the efficient assignment of trips to running boards (i.e., bus duties). Input variables such as the population density of the area served by the line, bus capacity and single mean round-trip time, including slack time, are used. This work was extended in order to address the synchronization of certain arrivals [Ceder *et al.*, 2001]. In [Palma and Lindsey, 2001] the goal is to minimize total schedule delay costs for the users. In [Salzborn, 1972], the goal is to define the bus departure rate as a function of passengers' arrival rate. Firstly, the author minimizes the fleet size and, secondly, he minimizes passenger waiting time. In all these works [Ceder, 1986; Ceder *et al.*, 2001; Palma and Lindsey, 2001; Salzborn, 1972], it is assumed that the travel time is deterministic.

The approaches used by Carey and by Zhao et al. benefit from the existence of abundant archived data from AVL systems, in particular the one by Zhao et al.. In fact, the calculus of $Var[l]$ needs it. This work has the appeal of being

an analytical approach. However, for the schedulers, rather than a method that solves the (partial) problem in a deterministic way, they need a tool to give them insights into the best solution, at least while there are no answers to questions such as "what are the optimal ratios between STT and lateness unit costs and between STT and earliness unit costs (in Carey's approach)?", or, "when should the scheduler put on an additional bus, i.e., how does passengers' waiting time compare with the operational cost of an additional bus?", or even, "what is the impact of reducing the SCT on operational costs?".

Related works on TTP a few days ahead

Surprisingly, TTP a few days ahead is hardly mentioned in the literature. The only reference we found is a data-driven method used in the internet route planner ANWB since September 2007 [Klunder *et al.*, 2007]. It uses k-nearest neighbors (see Sect. 4.4.2) with the input variables departure time, day of the week and date. Other factors, namely precipitation and school holidays were being studied to be included in the model but results on these experiments are not yet known. The authors report a value of 10% for the mean relative error, i.e., the absolute value of the difference between the reported and the actual values divided by the actual value. This value was obtained using data from freeways. According to the authors, the k-nearest neighbor method was selected from a set of other previously tested methods, however no references were found about those experiments.

2.3 Outlining the evaluation and the deployment phases

The definition of travel times for both timetabling and duties definition has necessarily different evaluation and deployment issues.

For timetabling, the main evaluation problem is the previously mentioned issue of defining the objective function. A priori, the deployment phase is not expected to have special difficulties.

For duties definition, the evaluation criterion of the prediction method is known. However, the comparison between duties using different travel times (the timetabled ones and the shorter horizon prediction ones) is a difficult issue. The problem is the impossibility of testing different planned duties in the same situation. Obviously, it is not possible to execute two different definitions of duties under the same circumstances. Additionally, it is not reliable to reconstruct the duties using past trips for two different planning scenarios because the past data is conditioned by the used planned duties, i.e., it is expected that the actual duties have a smaller gap to its base plan than to an alternative plan.

Consequently, a plan was outlined, in cooperation with the operations department, to evaluate driver duties using different travel times. This plan, defined at the end of 2004/ beginning of 2005, was:

1. to have three different versions of driver schedules (a driver schedule is a set of driver duties for a day and a rostering group) using different travel times (assuming different traffic conditions) defined in the long term;

2. to predict travel time three days ahead; and

3. to choose one of the three schedules according to the predictions.

The prediction horizon of three days was the shortest one possible to allow the department of operations time to make the necessary adjustments before the announcement of the duties to the drivers and simultaneously, respect the two days of anticipation for the announcement of duties to the crew staff, imposed by the existing company agreements.

This methodology should be tested alternatively to the current one (one day, one method, next day, another method) during a previously defined time period of 15 days or 1 month. To do this, it is necessary to have the agreement of the rostering group to be used in the test because during the test the drivers in the group will know their duties just two days before the day of the duty instead of the usual two weeks. Due to the sensitivity of the relationship with the drivers and their unions, it was decided to do the validation process described above just once. Firstly, the prediction tool is developed and secondly, the impact of the new predictions in the costs of the company is evaluated. This method was defined without the compromise of being implemented in time for this thesis due to the mentioned opportunity for the company. The business evaluation of the impact of this research is the obvious step forward for this research, whichever the approach is used. However, it could not be included in the scope of this thesis due to the difficulty of coordinating the research and the business interests of the company.

2.4 Restating the problem

In this thesis we try to define travel times for planning purposes. Two different subproblems are studied:

- Definition of travel times for timetabling: it is a multi-objective problem (it intends to reduce both passengers' expected waiting time and operational costs) with a long term horizon. The goal of this study is not to define timetables for new lines but for existing lines and, consequently, lines with archived data of actual travel times. The reason for this option is that the adjustment of timetables is a much more frequent problem at STCP than timetable creation, which is, in practice, a sparse planning task. This study is discussed in Part II (Chap. 3).

- TTP for duties definition: it is a three-day horizon prediction problem. From a mass transit company's point of view, the usefulness of TTP three days ahead depends on the possibility of showing how these predictions can be used to reduce costs and/or increase service reliability. This depends on how the predictions can be used in the different operational planning tasks in a mass transit company. As mentioned earlier, this is not the scope of this thesis. The scope of this study on TTP three days ahead is just the prediction problem, i.e., to predict travel times as accurately as possible three days ahead. The relevance of this study is that with the amount of data that mass transit companies have, due to the investments they made in Advanced Public Transportation Systems, there is increasing pressure to adopt more flexible planning approaches. Moreover, this study is relevant for logistics and delivery companies. The lack of studies on

travel time prediction for the long term is simultaneously a surprise and a research opportunity. For this purpose, the availability of data is still a problem (deliveries do not follow predefined routes and, consequently, there is not the same type of data there is in this study) but it is expected that in the near future there will be. This problem is the one we study in more detail in Part III (from Chap. 4 to Chap. 8).

2.5 The data

The next step is to know what data is available to solve the problem (Sect. 2.5.1) and analyze it (Sect. 2.5.2).

2.5.1 Relevant data sources

STCP has made, in the last few years, important investments in information systems in order to support the decisions of both top and operational managers. For the department of operations there are two major systems: (1) the GIST system, to support the planning tasks (partially described in Sect. 2.1.2); and (2) the SAEI system, a bus dispatching system for real time control purposes.

The main components of the SAEI include:

- Automatic vehicle location based upon dead reckoning sensors supplemented by differential GPS technology;
- Voice and data communication system using TETRA (Trans European Trunked Radio);
- On-board computer and control head displaying schedule adherence information to drivers, and detection and reporting of schedule and route adherence to dispatchers;
- Automatic passenger counters on front and rear doors of twenty vehicles;
- Computer-aided dispatch center.

A test version of the SAEI system began to store data on a regular basis in 2003. Sometime later, the control center started to be used regularly. Nowadays, at rush hour, it has 8 dispatchers to coordinate any problems that might occur. Each dispatcher is responsible for six or seven lines. Despite the fact that the SAEI was designed for control purposes, nowadays it is a major source of the data warehouse owned by STCP. The data warehouse is the main source of information for management purposes. The most impressive characteristic of this data warehouse is its very low level of granularity [Inmon, 1996]: in fact, information is stored at the bus stop level, allowing a very detailed level of analysis. The data from the SAEI system is that used in this thesis. Since the goal is to predict travel times of trips, for this study just the information for beginning and end times of the trips was used. Information about passing times at bus stops exists in the SAEI system but it was not used. The reason for using only the data registered at the beginnings and endings of the trips is that trips are defined by route and route-based prediction (prediction for the full trip) is reported to have less variance and, consequently, as being more accurate than

link-based prediction (prediction for each road stretch that composes the route) [Chien and Kuchpudi, 2003].

2.5.2 Gathering and analyzing data

Firstly, the main issues of the gathering task are described, then the collected data is analyzed by visual inspection.

Gathering data

The data used throughout this section and Chap. 5 is from line 78, variant 1, direction 1, i.e., route 78-1-1. The time period under analysis goes from January the 1st to August the 30th, 2004. This route was chosen because it was a large one crossing areas of the town with different traffic characteristics. Additionally, it was the one with most trips stored in the SAEI database when this data was collected. Line 78 was cancelled at the end of 2006 after network restructuring.

The relevant trip data is stored in two different tables from SAEI: trip beginnings and trip endings. The obtention of trip data is not direct. This is due to the lack of a trip identifier, typically a primary key, in the SAEI database. Consequently, it is not possible to relate records of both tables in a direct way. In order to know which trip ending is related to a given trip beginning, one needs to sort all trip beginnings and endings and to match pairs of trip beginnings and endings. From more than 18000 records of trip beginnings and more than 17000 records of trip endings, it was possible to obtain around 8000 full trip records. Fig. 2.6 presents the trips obtained after this first data cleaning task.

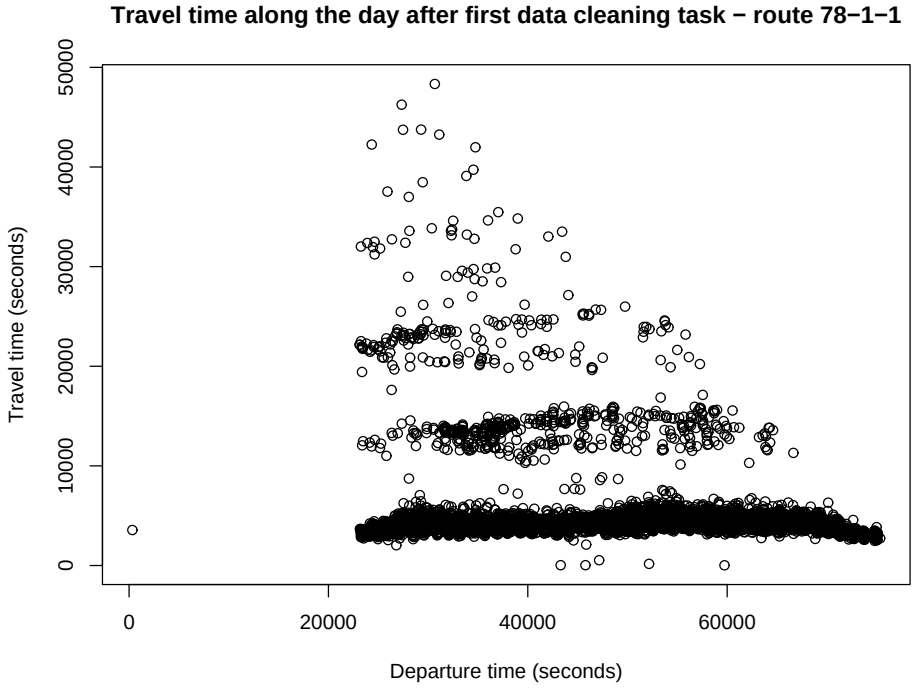

Figure 2.6: Travel time (after the first data cleaning task).

At a first glance Fig. 2.6 looks quite bizarre. The trips are organized in horizontal stretches. The higher the stretches, the shorter they are. The first stretch has correct trip records, while others have aggregated trip times, i.e., each point from the second stretch has the added sum of two trip times, the third stretch has the added sum of three trip times, and so on. In the said cases, the trips are aggregated because some trip endings were not recorded. Possible causes for the existence of these aggregated trips might be:

- Failure in the communication via radio between the bus and the control center [2];
- Lack of trip ending signalization by the driver. This situation happens typically in the last trip before the return of the bus to the depot.

The trip data set used along this chapter (Fig. 2.7) was obtained after some more data cleaning tasks, in order to maintain just the first stretch.

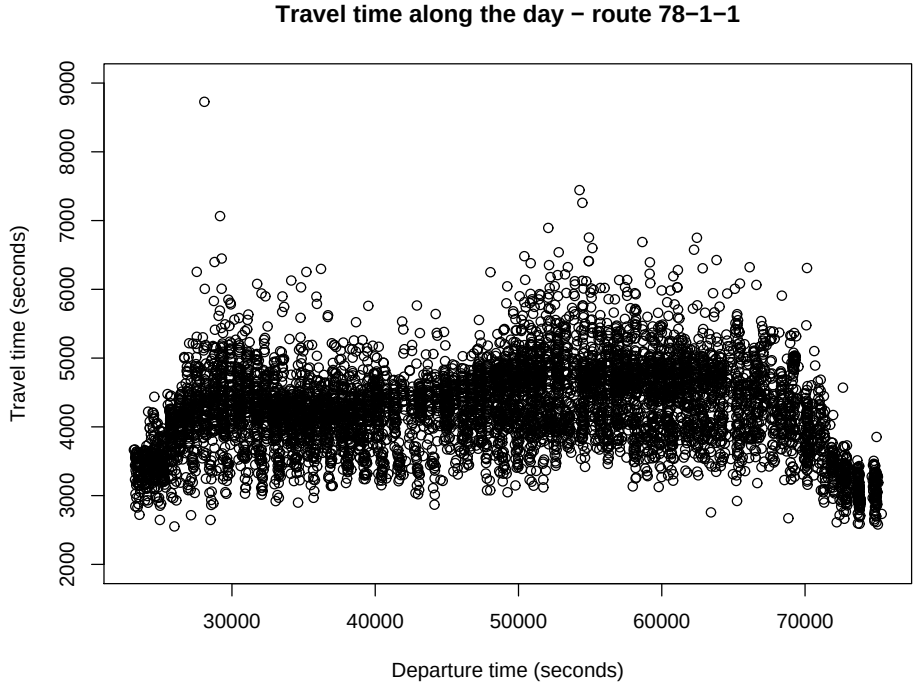

Figure 2.7: Travel time along the day.

A detailed study on the quality of the data about the trips obtained from the SAEI system is described in [Castro, 2008]. The main conclusions are that if the goal is to obtain a representative sample of trips, the data is sufficient and it is possible to obtain representative samples. The problem is to compare actual duties with planned duties. The only valid comparisons are between the complete ones. For all the others it is not possible to guarantee whether the trip is missing because it was not done or because it was not correctly registered. Consequently, the sample of duties obtained in this way is necessarily biased.

[2] In 2004 this problem was reported by STCP to the company responsible for the SAEI project.

Analyzing data

In this section we describe the process of input variables identification by visual inspection of the data and also by suggestions given by STCP traffic experts.

Some previous concepts on time series forecasting are introduced just to help the reader [Makridakis *et al.*, 1983; Nóvoa, 1994]:

- Seasonality: periodic repetition of a pattern;
- Cycle: irregularly spaced repetition of a pattern;
- Trend: global tendency of the time series;
- Impact factor: irregularly spaced event.

Several seasonal / impact components can be identified by plotting the data:

- The daily seasonality: by analyzing figure 2.7, one can identify this type of seasonality.
- The weekly seasonality: Fig. 2.8 (left-hand side) shows the differences between the daily average of travel time for different week days. The days of the week with the shortest average travel times are Sundays, whilst the days of the week with the longest average travel times are working days. Average travel times for Saturdays fall between these two extremes. The weekly seasonality is obviously due to these differences. Fig. 2.9 shows the different behavior of travel time on Mondays and Saturdays.

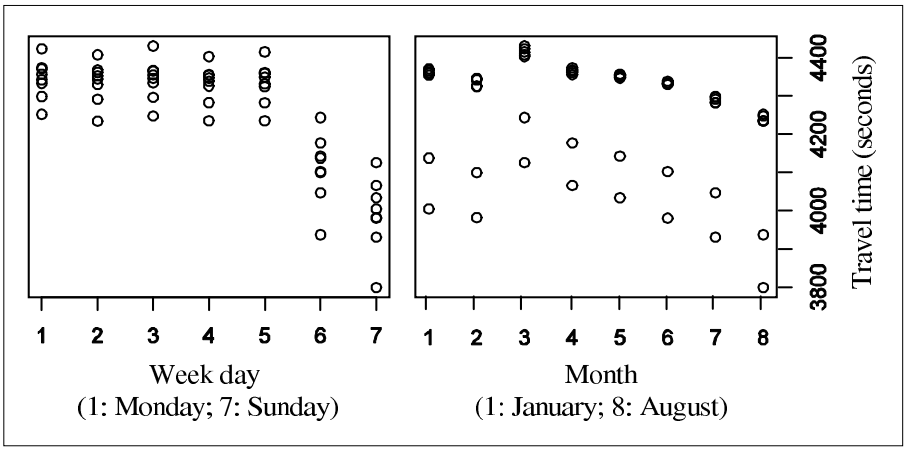

Figure 2.8: Daily average of travel times grouped by week days and months.

- The seasonality of the year: there is not enough data to obtain reliable conclusions, however the expected variations throughout the year are apparent on the right-hand side of both figures 2.8 and 2.10.
- The holiday impact: this impact is expected to change according to the day of the week. If the holiday is on a Tuesday or Thursday, it is likely that workers take respectively Monday or Friday off, potentially increasing the impact of holidays. From the period of time covered by this study it is not possible to evaluate the extent of this impact.

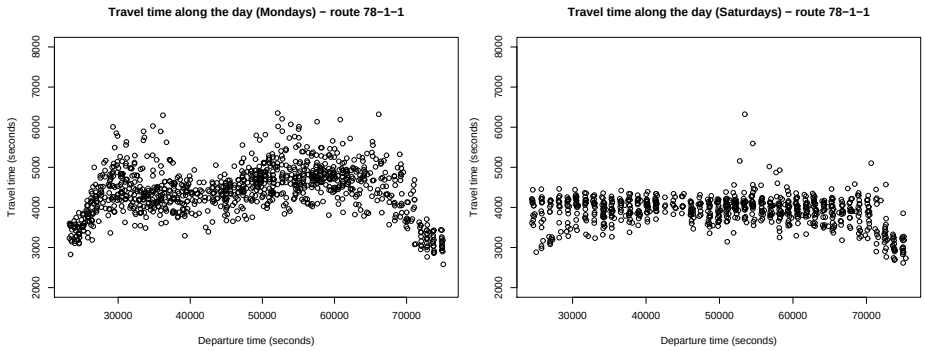

Figure 2.9: Travel time along the day on Mondays and Saturdays.

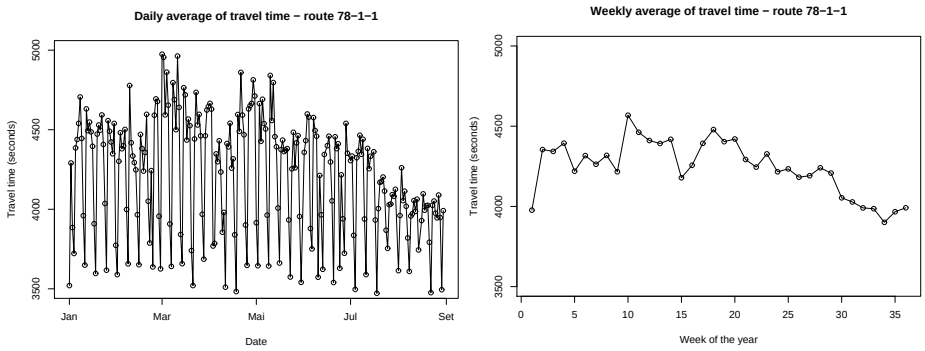

Figure 2.10: Daily and weekly average of travel time.

- The school breaks impact: there are four main school vacations per year in Portugal, namely, Carnival, Easter, summer and Christmas breaks. Fig. 2.11 shows obvious differences in travel time throughout working days for school vacations and for school periods. In Fig. 2.10 (right-hand side), shorter travel times during weeks 15 - 16 (Easter holidays) and after week 30 (summer holidays) are evident.
- The pay day impact: on Sundays previous to a typical pay day [3] the travel times appear to be shorter (Fig. 2.12). This can be explained because people have less money at this period of the month combined with the circumstance that on Sundays, the traffic is mainly for leisure purposes and not for work.

No cycles or trends were observed. They could not be observed because there is not enough data to isolate them from the apparent seasonality of the year.

The seasonalities of the day, week and year (as much as can be observed) and the impact of school breaks are, globally, as described in the literature [Vuchic, 2005; Turner *et al.*, 1998].

[3] In Portugal, the state pays on the 23rd day of the month, except when the 23rd is not a working day. When this happens, the pay day is brought forward to the previous working day. In private companies, pay day is different for each company but is usually at the end of each month.

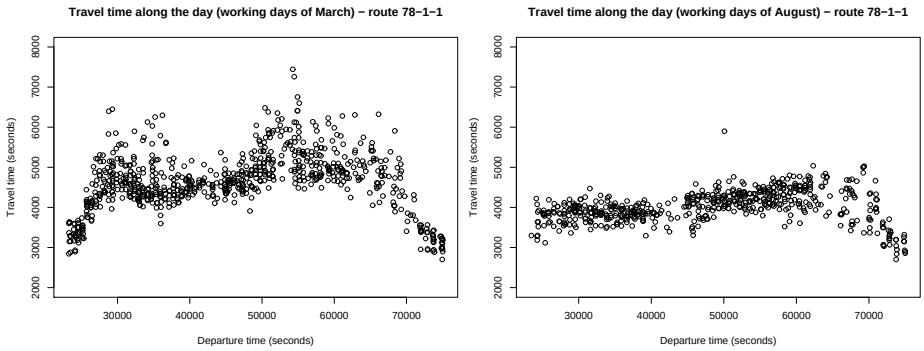

Figure 2.11: Travel time along the day (working days - March and August).

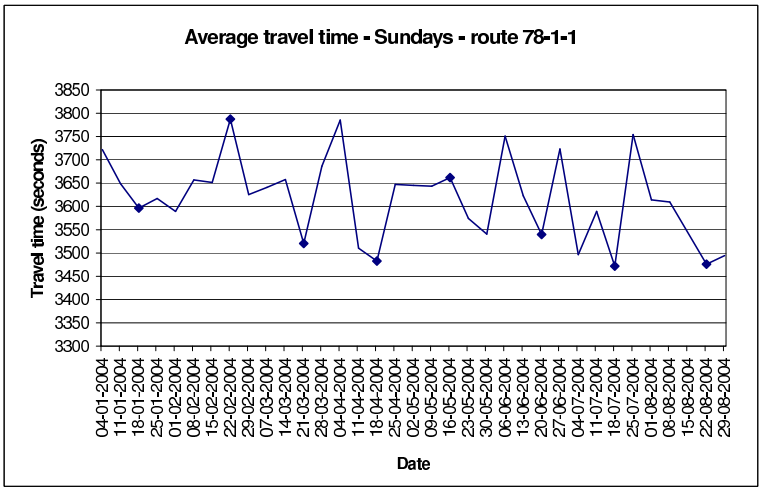

Figure 2.12: Daily average of travel time (Sunday) - the Sundays immediately before pay day are marked.

STCP traffic experts identified other factors that can explain travel time, namely: the driver, the type of the vehicle, the service, weather conditions and special dates that can potentially have an impact on traffic flow and, consequently on travel time, such as, the end of holidays, the beginning of long weekends, etc. These variables are better explained in Sect. 5.1.

Part II

Travel times for timetabling

Chapter 3

A DSS for timetable adjustments

The planning process at a medium/large size mass transit company has different decision moments needing travel time predictions. These moments were identified in Sect. 2.2.2 for the STCP case: the definition of timetables and the definition of duties for both buses and drivers. The studies for the definition/prediction of travel times to use for each one of these moments are done separately in this chapter and in part III because they are indeed two different problems. However they are both part of the same planning process and, consequently, the adequacy of the planning to the reality depends of the quality of both the definition of travel times for timetabling and the prediction of travel times for duties definition. That is the reason why we study both problems in this thesis.

In this chapter we describe our solution to the first problem as stated in Sect. 2.4. We present a decision support system (DSS) for timetable adjustments. The solution found favors usability rather than scientific novelty. Nevertheless, there is no known solution in the scientific literature identical to this one, as far as we know.

Firstly, the reasons for developing a DSS for timetable adjustments are explained (Sect. 3.1), then the DSS is described (Sect. 3.2), and finally how to use it in typical situations is described for short and long headways (Sect. 3.3).

3.1 The reason for using a DSS

In Sect. 2.2.3 the limitations of the existing methods for the creation of bus timetables with regard to the value of travel time were pointed out. One of the difficulties is the inherent multi-objective nature of the problem of finding the optimal value, namely the minimization of both the expected passengers' waiting time at the bus stop and the operational costs. In this chapter we propose a decision support system which allows the person in charge of timetable planning to assess the impact of different scenarios in both objectives. He or she can test different values for the scheduled travel time, slack time and headway, and obtain a set of descriptive statistics that allow this person to evaluate the impact of

this scenario using data from a past period similar to the one that the timetable is going to cover.

The reason for using this approach is that the existing ones can give optimized solutions when some of the variables are fixed but do not allow the planner to easily evaluate the sensitivity of the solution to each decision variable. Furthermore, the objective functions used by these approaches do not simultaneously cover the two above mentioned objectives. This DSS is compatible with the use of optimized solutions like the one described in [Zhao *et al.*, 2006]. In fact, such solutions can always be developed and integrated in this DSS as a default solution for fixed given values. This DSS can be seen as an integrated environment for analysis.

3.2 Description of the DSS

The data used by the DSS is obtained from the data warehouse, but its original source is the SAEI system (Sect. 2.5.1). The reason to do so is that when the DSS was developed (end of 2007) the migration processes from the SAEI database to the data warehouse have been already implemented by STCP. These processes include part of the cleaning tasks described in Sect. 2.5.2.

The person in charge of planning (the planner) starts by selecting the data to be used for the analysis of travel times. It is expected that the planner will choose past data that might be representative of the period (in the future) that is going to use the new version of the timetable. The planner is able to choose the characteristics of the analysis, such as period of time (days), the time of the day and the line/route. There are three types of analysis that depend on the characteristics of the line and the objectives of the analysis the planner wants to perform: single direction, double direction and circular route analysis.

3.2.1 Single direction analysis

In a single direction analysis (Fig. 3.1) the information provided is:

- A time plot of travel times: it provides visual information about travel times. It allows the user to observe the dispersion of travel times during the period of analysis. This can show, for instance, the difference in travel times along the seasons of the year or days of the week. It is also possible to identify outliers and to obtain information on the trips by clicking the mouse.

- A plot of the accumulated relative frequency of travel times: it allows the user to have an idea of the route performance, just by looking at it. The steeper is the slope of the graph, the less is the variability of travel times. We can observe that the duration of the majority of the selected past trips for line 602 is between 50 and 62.5 minutes.

- Information about the sample ('amostra' in Portuguese): it characterizes the sample past data being used in the analysis.

- A table with statistical information on travel times: it provides the user with statistical information for the 25th, 50th and 75th percentile. For these three travel times, say travel time t, one presents the percentage

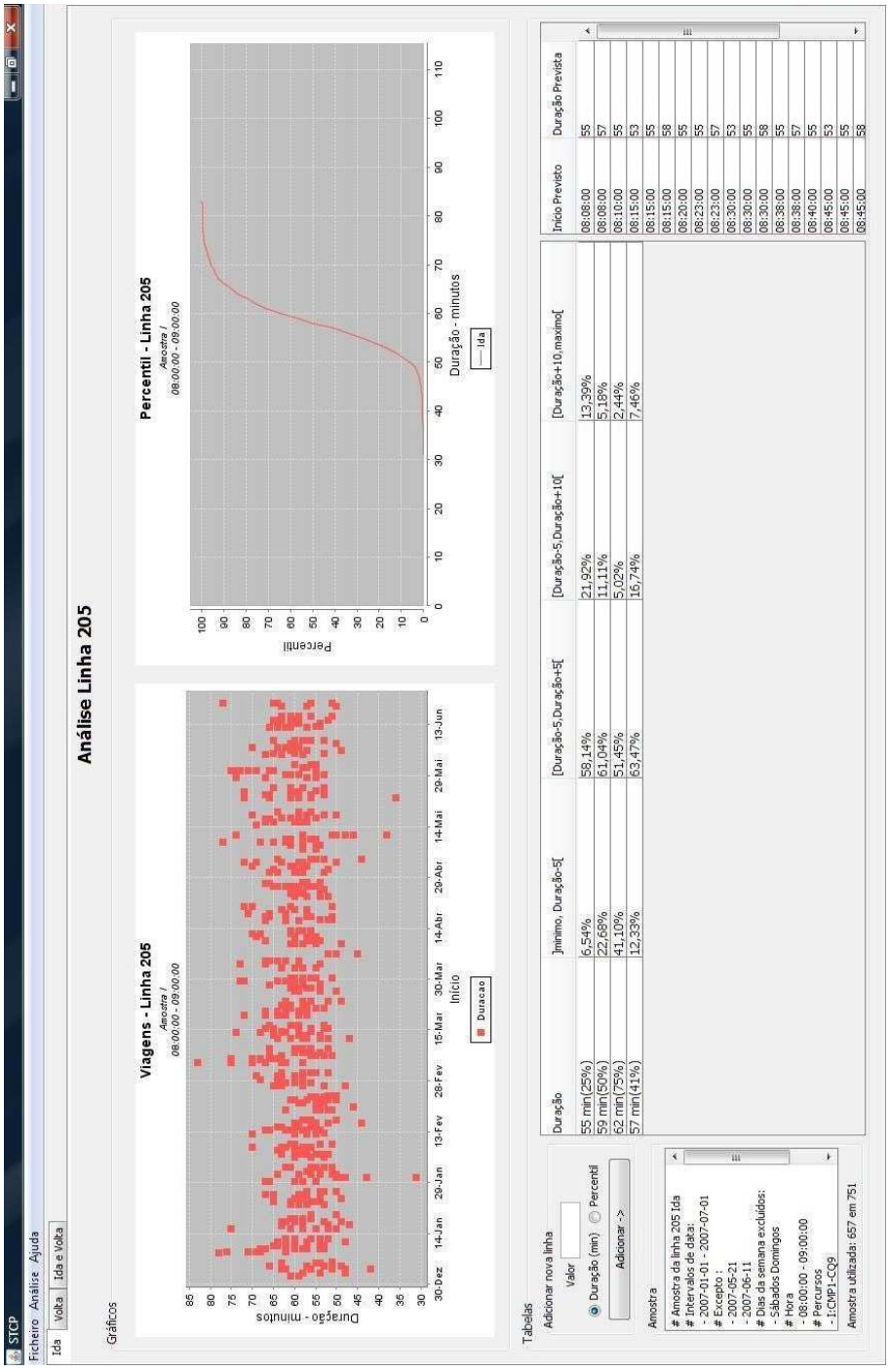

Figure 3.1: DSS for single direction analysis.

of travel times in the intervals: $]-\infty, t-5min[$, $[t-5min, t+5min[$, $[t+5min, t+10min[$, $[t+10min, +\infty[$. The value of t that maximizes the percentage of travel times in the interval $[t-5min, t+5min[$ is also calculated and presented in the table. This is the initial information provided in the table. The percentiles and the intervals being used were chosen by the planners at STCP. This information complements the plot of the accumulated relative frequency of travel times. However, the user may add lines to the table by choosing the values of percentile or duration he wants to analyze. The information provided in this table is very useful for the transit planners because it helps them to estimate the effects on delays and early arrivals, when choosing the duration for a trip (or set of trips).

- A table with the STT used in the selected past period: it provides information about the STT in the time interval and dates selected. It is useful to compare the STT being used with the actual past travel times, which allows the detection of STT that are not correctly defined.

3.2.2 Double direction analysis

In a double direction analysis (Fig. 3.2), all the information provided for single direction is also shown separately for each of the two directions, as described in the previous section. One also provides an additional tab with statistical information of travel times, possible slack times and the minimum number of vehicles needed, according to the scheduled headway (a value to be introduced by the planner). This kind of information couldn't be shown in a single direction analysis because the definition of slack times and the scheduling of vehicles depend on the entire cycle (go and return).

Fig. 3.2 shows the sceen of the application for a double direction analysis. The information provided is:

- A plot of the accumulated relative frequency of travel times for both routes (go and return): it is identical to the plot of the accumulated relative frequency of travel times described in Sect. 3.2.1 but with two data series, one for each route;

- A table with statistical information on cycle times: the information provided in this table is applied to the lines with period trips in a defined period of time, in which vehicle scheduling is done together for both directions. For each travel time presented in the table (go and return), one presents three possible slack times and the percentile for the sum of travel time and slack time. The table also presents the minimum number of vehicles needed to cover the trips with those travel and slack times. The total slack time is calculated based on the headway of the service. Imagine the case of a headway of 11 minutes, STT of the first trip equal to 57 minutes and STT of the returning trip equal to 55 minutes. The sum of both durations is 112 minutes. This means that the three lowest values for the duration of the cycle are 121, 132 and 143 minutes, because the duration of the cycle has to be a multiple of the headway (Eq. 2.2). In the first case we have 121 - 112 = 9 minutes of slack time to distribute by the two trips. The appropriate slack time to add to each trip is calculated

Figure 3.2: DSS for double direction analysis.

based on the aim of minimizing the difference between the percentiles of the total times for each direction (travel time plus slack time). In this example, the solution is to give 6 minutes of slack time for the first trip and 3 minutes of slack time for the returning trip. The percentile values for the sum of STT and slack time are 79.76% and 80.65%, for the go and return trips respectively. This is the solution that minimizes the differences between them. The minimum number of vehicles needed to cover this demand is 11. However, if this solution is applied, we can expect that around 20% of trips, in each direction, won't be finished in a time less or equal to the duration of travel time plus slack time. This may cause an important number of delays in the departures of sequential trips and poor line performance. So, the transit planner should analyze all the possibilities suggested in the table, trying to maximize the percentage of trips covered by the travel time and slack time, but at the same time, minimizing the costs for the company. The planner is also able to add lines to the table in order to analyze different solutions. He just has to introduce the desired travel times or percentile values, for both directions.

3.2.3 Circular route analysis

In circular routes, there is no slack time added to trips because the bus must always be in service. Besides, the cycle is composed of just one trip. Because of this, the information provided is similar to that in single direction analysis with the addition of the minimum number of vehicles needed for each tested scenario.

3.2.4 Additional features

The DSS has the following additional features:

- The user can modify the data used for analysis at any time;
- He can also save the analysis that he is performing at the time and reopen it whenever he wants;
- The plots can be zoomed in and out to better analyze details;
- A help file is provided;
- The application can easily use a different idiom because all sentences on the interface can be modified in a file used for configuration [1].

3.3 How to use the DSS

The criteria for defining the timetable depend necessarily on the headway [Zhao *et al.*, 2006]. In this section we describe how to use the DSS for timetable adjustments for typical situations: short headways, large scheduled headways and circular routes.

[1] Despite that, the figures in this chapter are in Portuguese because we have used the examples from the STCP company.

3.3.1 For short headways

It is known that for short headways (those shorter to 10 minutes) customers arrive roughly uniformly at the bus stops. They do not care about the timetable because they know that they will wait no more than the headway time. Additionally, there is an inherent instability of headways when they are short. This happens because when a bus is delayed, it has, typically, to pick up more customers, increasing the delay after the previous bus even further. At the same time, the following bus tends to go faster because it will stop less and less time since the number of passengers tends to decrease. In other words, for short headways, the actual travel times are very sensitive to the maintenance of the headways. If the actual headways are irregular, the buses tend to bunch [Newell, 1974; Zhao *et al.*, 2006]. Fig. 3.3 presents a real example of this situation visible in the middle of the figure for the return trips (the top-down ones). In this situation, i.e., for short headways, the concern of the planner should be to guarantee that the time between two buses on the same route is, as much as possible, equal to the scheduled headway [Zhao *et al.*, 2006]. The controllers should use the same principle.

The question is: how to address the planning issues using the DSS for timetable adjustments for short scheduled headways? In order to minimize the variance of the headways, it is advisable to use the mean travel time (or the median, which is more robust to outliers). The sum of STT and the slack time for each direction should be a high percentile value $p.max$. This value is, necessarily, strongly conditioned by the satisfaction of Eq. 2.2. The possible values are represented in the table with statistical information on cycle times (Fig. 3.2).

3.3.2 For large headways

For large headways (those of over 10 minutes), the behavior of customers is different. They tend to arrive sometime before the scheduled passing time. In this case, the planner should guarantee the adherence of the actual trips to the scheduled ones and the controllers should act accordingly. In this case it is important to be aware that, from the passengers' point of view, an offset between the actual and the scheduled passing times is accepted as in time if in the interval $[-1, 5]$ (in minutes) [Strathman *et al.*, 1998]. Another important issue is that a delay is more acceptable than an advance of the same amount of time.

In this case, a low percentile should be chosen for the STT (represented by $p.min$) in order to reduce the number of trips passing ahead of schedule. The slack time should be the difference between $p.max$ and STT (the $p.min$). Again, this value should respect Eq. 2.2 and, consequently, the options are limited.

3.3.3 For circular routes

For circular routes, whatever the scheduled headway, the STT should comprise both the expected travel time and the slack time. The problem is that on circular routes it is not possible to have slack times because the buses are always in service. The time to accommodate delays must be incorporated in the STT. Consequently, a high percentile value $p.max$ should be chosen, in the same way

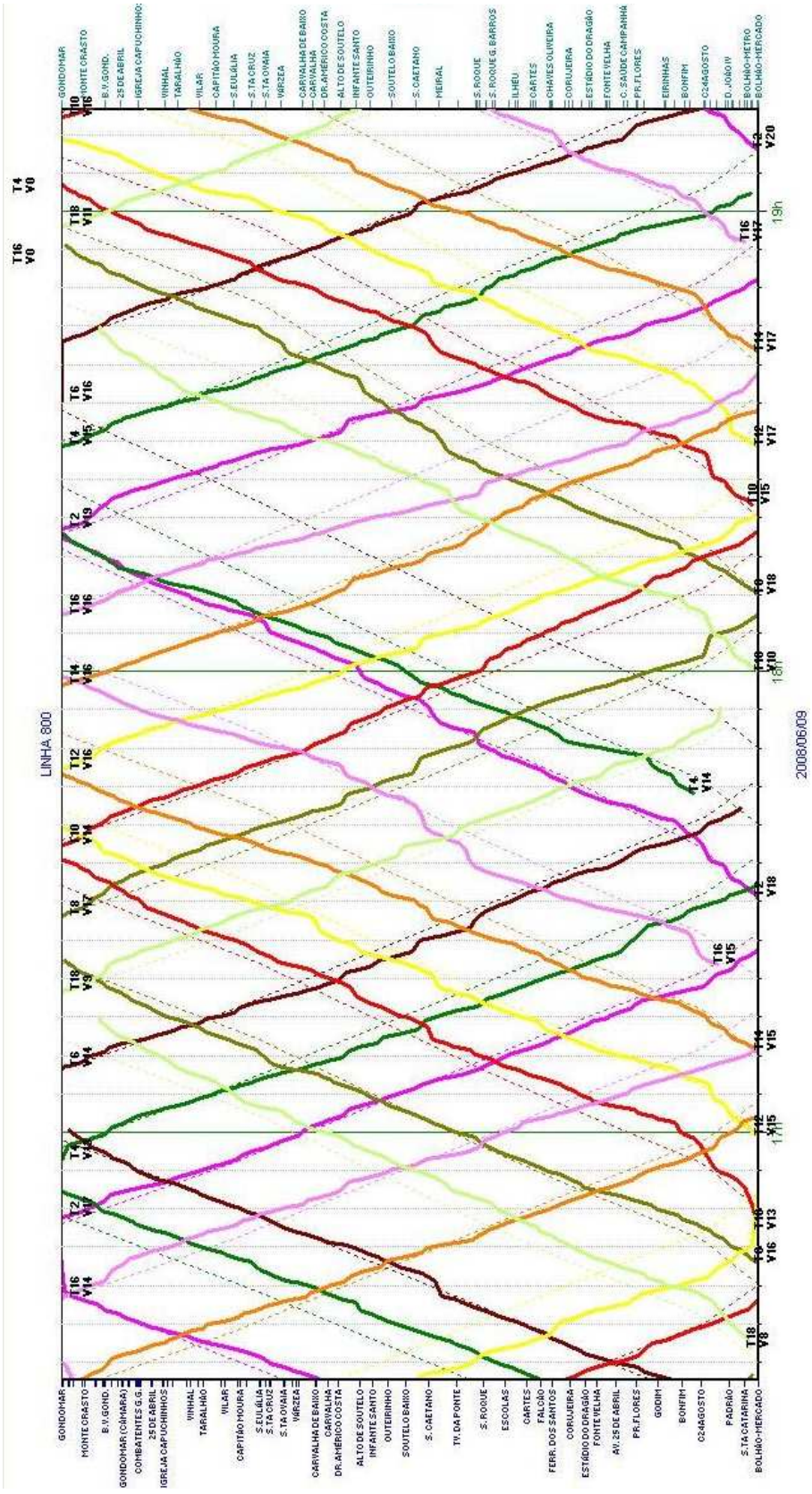

Figure 3.3: An example of a bunch situation (the source of this image is a software application owned by STCP).

as previously described. In this case, controllers tend to have more intervention in order to guarantee that the actual travel time respects the STT.

3.3.4 A discussion on the use of the DSS

It is expected that the values of $p.min$ and $p.max$ can be obtained empirically by the planners. Their values can be different for different routes but it is expected that, at least for routes with identical characteristics, they will not be too different.

An important advantage of this DSS is that it allows the planner to learn the advantages and disadvantages of the different scenarios. Another good characteristic of this DSS is the possibility of testing different scenarios and evaluating them by estimating the impact of each one using the different indicators, namely:

- Number of vehicles needed: an important indicator of the operational costs;
- Estimated percentage of trips passing ahead of schedule: an indicator of the level of passengers' satisfaction;
- Estimated percentage of trips starting delayed: another indicator of the level of passengers' satisfaction.

Additional information that could be useful is the estimation of the number of passengers expected for a given scenario. This information was not included because it is not fully operational. There are many situations where the increase of the headway by two or three minutes does not meaningfully degrade the quality of the service but meaningfully reduces its cost (for example, because the number of vehicles needed can be reduced). When the planner does this type of experiment, he needs to estimate how many passengers are expected in order to avoid the buses getting too overcrowded.

Indicators for short headways were not included because of the existence of many missing and incomplete trips, as discussed in Sect. 2.5.2. In [Strathman *et al.*, 1998; Wang *et al.*, 2006] indicators for this situation are suggested.

This tool is being used successfully at STCP. It can also be used in the future as an experimental prototype for the test of different analytical approaches, like the one suggested by [Zhao *et al.*, 2006]. This is a natural step forward for this research.

Part III

TTP three days ahead (for the definition of duties)

Chapter 4

Regression: a review

In 1877 Francis Galton observed reversion toward the mean in experiments on seed size in successive generations of sweet peas. He renamed this phenomenon regression in the mid 1880s [Bulmer, 2003]. Since then, regression has been the task of creating a mathematical function that explains a numerical output variable from a set of input variables. The obtained function can then be used for prediction of unknown output values for given new input vectors.

This chapter is organized as follows: firstly, the question of whether to use time series forecasting methods or inductive learning ones is discussed. The remainder of the chapter is about regression mainly focused on the inductive learning approach: the main issues on regression are described; then data manipulation tasks in order to improve accuracy are presented; finally, the main algorithms for regression are enumerated.

4.1 The approach to use: inductive learning methods vs time series forecasting methods

This section does not aim to describe in detail either the concept of inductive learning (this will be discussed in Sect. 4.2), or the concept of time series forecasting. However, the main characteristics of each one of these families of methods, even if informally, must be given.

Time series forecasting methods [Makridakis *et al.*, 1983] assume that data are spaced equally over time. The sequence of the data is crucial for data understanding. The model is built by previous knowledge of seasonalities (periodic repetitions of a pattern), cycles (irregularly spaced repetitions of a pattern), trend (global tendency of the time series) and impact factors (irregularly spaced events). This previous knowledge is obtained using appropriate data analysis techniques. Some of the most commonly used families of forecasting methods are ARIMA models and smoothing models.

Of the smoothing models, the most adequate one for this particular problem is the Holt-Winters method because it is able to deal with both trend and seasonality. The seasonality is obvious from Sect. 2.5.2. The apparent seasonality of the year should be treated as trend, since there is just one year of data, not allowing this seasonality to be captured. In any case, the original method must be adapted to deal with several seasonalities and different impact factors. This

Table 4.1: Regularly spaced time series.

Time lag	Expected values	Filled values	Unfilled values
15 min	14337	6394	7943
30 min	7290	4796	2494
60 min	3645	3310	335

approach has been used to predict daily sales of hypermarkets [Nóvoa, 1994]. The main differences between TTP and Nóvoa's work are: (1) the travel time data is irregularly time spaced; and (2) there are more potential causes to explain travel time, namely the ones identified by the traffic experts. A usual approach to solving the first issue is to convert the irregularly spaced time series into a regularly spaced one, by averaging travel times that fall in the same time interval. However, experiments done in our data set show that when using small time lags, there is an important percentage of unfilled data (Table 4.1), and when enlarging the period there is a process of averaging that meaningfully reduces the detail of the data. The second issue can be, on its own, a good motivation for the use of inductive learning approaches as suggested in [Petridis *et al.*, 2001].

ARIMA models comprise three basic models: AR (autoregressive), MA (moving average) and a combined ARMA in addition to RD (regular differencing). When RD is applied, together with AR and MA, they are referred to as ARIMA, with the I indicating 'integrated'. The two reasons pointed out to not use the Holt-Winters method are also applicable to the ARIMA models. Additionally, for the particular problem of TTP, the ARIMA models are not the most adequate because they do not deal well with more than one seasonal component [Nóvoa, 1994]. Despite the work by Nóvoa, forecasting time series with multiple seasonal patterns used to be considered a difficult problem. However, recently (2008) a promising approach to dealing with this problem appeared even if just for regularly spaced time series [Gould *et al.*, 2008].

Inductive learning for prediction aims to learn a model from a data sample, i.e., given a data set with V input variables and one output variable, it returns a model that is used to predict new data (data for which the output variable is unknown). These methods do not have the difficulties previously referred. Both issues (1) and (2) are solved by a proper choice of the input variables, a subject discussed in Sect. 5.1. Using this approach, the time domain is explicit through the inclusion of input variables to model time [Kindermann and Trappenberg, 1999] and the TTP problem is solved as a regression problem. The concept of regression is discussed along this chapter.

4.2 Regression

This section begins with main definitions followed by the description of the regression process. The concept of generalization error is introduced just before the description of methods for evaluating supervised learning models. Finally, statistical tests for model comparison are discussed.

4.2.1 Definition

Regression is a widespread concept that crosses different areas of knowledge. The concept of regression used throughout this thesis is restricted to regression as a subarea of inductive learning.

According to Brandyn Webb,

> "Inductive learning is the inference of a generalized conclusion from particular instances" [Webb, 1995].

Inductive learning techniques are usually classified as supervised or unsupervised learning.

> "Supervised learning can be formalized as the problem of inferring a function $y = f(\mathbf{x})$, based on a training set $D = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$. Usually, the inputs are V-dimensional vectors, $\mathbf{x}_i = [x_{i,1}, \cdots, x_{i,V}]^T \in \Re^V$. When y is continuous (e.g., $y \in \Re$), we are in the context of regression, whereas in classification problems, y is of categorical nature (e.g., binary, $y \in \{-1, 1\}$). The obtained function is evaluated by how well it generalizes, i.e., how accurately it performs on new data assumed to follow the same distribution as the training data [...]" [Figueiredo, 2003].

Given a set of V input variables $x_1, x_2, \cdots, x_v$, also called predictor variables or attributes, the goal of supervised learning is to predict the values of the response variable y, also called output variable, concept or target. Since travel time prediction is a regression problem, only regression is discussed.

4.2.2 Process of regression

For regression, inductive learning consists ideally in the inference of a function

$$\hat{f} : X \to \Re, \text{such that} \hat{f}(\mathbf{x}) = f(\mathbf{x}), \forall \mathbf{x} \in X, \tag{4.1}$$

where f represents the unknown true function. The algorithm used to obtain the $\hat{f}$ function is called induction algorithm or learner. The particular values for the input parameters of the induction algorithm are often referred to as the parameter set. The result of the induction algorithm is the $\hat{f}$ function, called model, predictor or regressor.

In practice, the output of the learned function $\hat{f}$ will not coincide with the output of the real function f for every input value $mathbfx$. The typical result of the learner is an approximate function that minimizes the difference between $\hat{f}$ and f.

4.2.3 Generalization error

A main issue of supervised learning is to measure how good the learned models are. To better understand this problem, let us introduce the concept of error. The true function f can be decomposed as

$$f(\mathbf{x}) = g(\mathbf{x}) + \epsilon \tag{4.2}$$

where $g(\mathbf{x})$ is a deterministic function and ϵ is the noise [Cherkassky and Mulier, 1998]. If the learned function $\hat{f}(\mathbf{x}) = f(\mathbf{x}), \forall \mathbf{x} \in X1$, where $X1 \subseteq X$, i.e., $X1$ is a sample of the population X, it means that $\hat{f}(\mathbf{x})$ is also learning the noise ϵ overfitting the data. Another problem can be that $\hat{f}(\mathbf{x})$ does not learn $g(\mathbf{x})$ underfitting the data. In both situations $\hat{f}$ will have lower capacity to predict new unlabelled data. In order to measure the quality of $\hat{f}$, the data is split into two subsets: (1) the training set, used to train $\hat{f}$; and (2) the test set for assessment of the generalization error, i.e., it measures the capacity of $\hat{f}$ to predict unlabelled data.

For regression, the most common generalization error functions, also known as loss functions, are the mean squared error (*mse*) or its squared root. Others exist, namely, the relative mean squared error (*rmse*). It expresses the relative predictions' variance in relation to the variance of the output variable. Its main advantage is that its interpretation is independent of the unit measure used to express the target variable. The *variation index* has the same properties as *rmse*. It expresses the root of the predictions' variance in relation to the average of the output variable. It is a measure similar to the ratio between the standard deviation and the average. Another well known loss function for regression is the mean absolute deviation (*mad*). The main difference between the squared root of *mse* and *mad* is that *mad* linearly weighs the values according to their distance to the average, while the squared root of *mse* quadratically increases the weights according to the distance. Many other generalization error functions exist for numerical prediction [Witten and Frank, 2000].

$$mse = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}(\mathbf{x_i}) - f(\mathbf{x_i}))^2, \tag{4.3}$$

$$rmse = \frac{mse}{\frac{1}{n}\sum_{i=1}^{n}(\bar{y} - f(\mathbf{x_i}))^2}, \tag{4.4}$$

$$varIndex = \frac{\sqrt{mse}}{\bar{y}}, \tag{4.5}$$

$$mad = \frac{1}{n}\sum_{i=1}^{n} \mid \hat{f}(\mathbf{x_i}) - f(\mathbf{x_i}) \mid, \tag{4.6}$$

where $\bar{y} = \frac{1}{n}\sum_{i}^{n} f(\mathbf{x_i})$, i.e., it is the average of the output variable.

Another important problem with inductive learning is the curse of dimensionality. It concerns the increase of the generalization error when the dimensionality (number of features) of the problem is increased and the size of the data set is maintained. This happens because the examples from the training set are spread by a larger input space, reducing the density of the data in the input space. For further details see [Hastie *et al.*, 2001].

4.2.4 Experimental Setup

The most applied approach in order to guarantee that the estimation of the generalization error does not use data already used to induce the model is the resampling approach. The choice of the experimental setup has a direct influence on the accuracy of the estimation of the generalization error, i.e., when choosing

a model, the estimation of the generalization error of each one of the models depends on the experimental setup used. The main resampling methods are:

- Hold out: randomly split the data into two disjoint sets, one for training and the other for testing. Typically 1/3 of the data is used as test set.
- υ-fold cross-validation: randomly split the data in υ disjoint subsets with sizes as similar as possible. Then, each fold is used as test set and the remaining $\upsilon - 1$ subsets are used as training set. The process is repeated υ times (quite often $\upsilon = 10$) such that each subset is used once as test set [Stone, 1974].
- Jack-knife (or leave-one-out): is a particular case of υ-fold cross-validation with $\upsilon =$ data size [Efron and Tibshirani, 1993].
- Monte-Carlo cross-validation: is similar to the hold out method but the process is repeated several times (typically between 30 and 50) [Picard and Cook, 1984].
- Bootstrap: random selection with replacement of a sample of the size of the data set. On average 63.2% of the data are selected (some with repetitions). The selected examples are used as training set. The remainder out of bag examples (36.8% in average) are used as test set [Efron and Tibshirani, 1993].
- Sliding window: it is particularly suitable for time-varying data sets. It aims to guarantee that the most recent data are those used for training, i.e., it is expected that recent data better explain the near future than old data. The window can be sequence-based or time stamp-based [Babcock *et al.*, 2002]. The former uses the last n examples for training, where n is a pre-defined parameter while the time stamp-based window uses all examples within a time interval t of the current time, where t must also be pre-defined. Whenever new data arrives (sequence-based) or a new time period goes on (time stamp-based), the data used for training changes. Currently there is on-going research. Other approaches, under the name of change detection and concept drifting on data streams, try to adapt the window according to the characteristics of the incoming data [Hulten *et al.*, 2001; Wang *et al.*, 2003]. This topic is not studied in this thesis even though it is a promising area of research.

4.2.5 Statistical tests

A common problem in inductive learning processes is to statistically validate the results. When comparing two algorithms, the reason why the direct comparison of their results may not be statistically meaningful is because part of that difference is of a random nature. Only one component of the difference is due to the algorithms. In a typical inductive learning algorithm, four different sources of the random variation can exist [Dietterich, 1998]:

- Selection of the test set;
- Selection of the training set;

- Randomization introduced by the learning process (deterministic algorithms do not have this source of variation);

- Random component of the target variable.

The choice of the experimental setup, as discussed in the previous section, must be chosen in order to guarantee that the sample used in the experiments is representative of all the sources of variation. Representative samples must have independent examples according to each variation source. However, in many practical problems, it is not possible to guarantee that all the examples are independent. This happens mainly to obtain a representative sample for the training set.

Let us assume that there are 500 examples. If we use, for example, 10-fold cross validation, at each different training, 400/450 examples of each of the 10 training sets are common to any other of those 10 training sets. Independency is not, obviously, guaranteed. Another option would be to use just 50 examples for training. In this case it would be possible to obtain 10 independent sets for both test and training tasks. The problem with this solution is that the accuracy performance of the model obtained with just 50 examples could be seriously reduced.

A well known method in statistics to isolate the variation one intends to study (in this case the variation due to the algorithms) is the use of pairwise hypothesis tests [Groebner and Shannon, 1985]. Pairwise hypothesis tests for the difference between two population means testing under the same conditions (the same training sets, test sets and models) if that difference is different from zero. Pairwise tests reduce the effect of dependency in the sample [Pizarro *et al.*, 2002] and also reduce the variance of the sample mean, increasing the power of the test when compared to a standard hypothesis test (unpairwise test) for the difference between two population means.

Let us assume two algorithms (represented by the suffix A and B) tested under the same training and test sets, with n being the total number of tested examples. The formulation of a pairwise hypothesis test in order to test whether the *mse* population mean of algorithm B is lower than the *mse* population mean of algorithm A is (under the normality assumption of $\overline{\Delta}$):

$$
\begin{aligned}
H0 : \mu_\Delta &= 0 \\
H1 : \mu_\Delta &> 0 \\
\text{If H0 } = \text{ TRUE } \Rightarrow ts &\text{ has a Student's } t_{n-1} \text{ distribution,} \\
&\text{where} \\
ts &= \frac{\overline{\Delta} - 0}{s_\Delta / \sqrt{n}}, \\
\Delta_i &= \hat{f}_A(\mathbf{x}_i) - \hat{f}_B(\mathbf{x}_i), i \in \{1, 2, \cdots, n\}, \\
\overline{\Delta} &= \frac{1}{n} \times \sum_{i=1}^{n} \Delta_i \text{ and} \\
s_\Delta &= \frac{1}{n-1} \sum_{i=1}^{n} (\Delta_i - \overline{\Delta})^2. \qquad (4.7)
\end{aligned}
$$

When multiple comparison tests are executed, the possibility of committing the type I error (to reject wrongly the null hypotheses) increases. The larger the the number of tests is, the greater the possibility of committing this error is. This is known in statistics as the 'multiplicity effect' [Salzberg, 1997]. The approach used to compare more than two population means is the analysis of variance. This name is due to the way the method works. To test whether there are meaningful differences between the algorithms being tested, it compares whether there is a meaningful difference between the global variation and the variation among the means of these algorithms. Formally, let us assume that each target value ($y_{i,j}$) is decomposed as $y_{i,j} = \mu + \alpha_i + E_{i,j}$, where μ is the global average, α_i is the effect of each algorithm ($i \in \{1, 2, \cdots, I\}$, I being the number of algorithms compared) and $E_{i,j}$ is the random error. The parametric analysis of variance, known as ANOVA (fully described in any introductory book on statistics, for example [Groebner and Shannon, 1985]) assumes that $E_{i,j}$:

- has 0 population mean and constant variance σ^2. The constant variance means, when comparing the population statistic for several algorithms, that the population variance of the results is assumed to be equal for the different algorithms. This is known in statistics as homoscedasticity.
- is independent.
- has normal distribution for all the populations being tested.

These three conditions are represented saying that $E_{i,j}$ has $\text{IN}(0, \sigma^2)$ distribution. The comparison of several algorithms is a one factor, fixed effects ANOVA problem. It is a problem with one factor because the algorithms are tested under the same conditions, i.e., the differences among them are due to just one factor: the algorithm. It is a fixed effects problem because the goal is to compare those specific algorithms. They are not intended to be representative of a population of algorithms.

The assumption of normality is not too critical because the ANOVA method is fairly robust to deviations from normality for balanced samples, i.e., equally sized samples. Anyway it can be tested using the Lilliefors variation of the Kolmogorov-Smirnov nonparametric test [Lilliefors, 1967]. When the deviations from normality are strong, the nonparametric analysis of variance known as the Kruskal-Wallis test is used. Analysis of variance is also robust to the heterogeneity of variances [Guimaraes and Cabral, 1997] for balanced samples. Anyway, homogeneity of the within-group variances (called homoscedasticity, as above mentioned) can be tested using the Bartlett test, if the normality assumption is guaranteed. For very different variances, the Kruskal-Wallis test can be used instead of the ANOVA test [Pizarro *et al.*, 2002]. Although the Kruskal-Wallis test can always be used, even when the ANOVA assumptions are fulfilled, it should only be used when the ANOVA cannot, because this method is more powerful.

Both methods, ANOVA and Kruskal-Wallis, test whether there is one group (in the present case, one algorithm) with a population mean meaningfully different from the others. If there is statistical significance for the existence of differences among the means, then a multiple comparison test should be used to test whether there is a meaningful difference among the population means

for each combination of two algorithms. These tests should be used as a second step after the analysis of variance step. Multiple comparison tests are different when the first step uses the ANOVA or the Kruskal-Wallis methods. In [Pizarro *et al.*, 2002] several multiple comparison tests for both cases, the parametric and nonparametric tests, are described.

All these methods assume that instances are independently drawn. However, in many situations such as time series and others, data can be strongly correlated. For such situations the normality assumption does not hold, not allowing the use of the ANOVA method and even nonparametric methods like that of the Kruskal-Wallis test are not applicable. These cases are much less referred to in the literature. There is some work on multiple comparisons of asymptotically stationary time series [1] [Nakayama, 1997], however we did not find information on multiple comparisons for non-stationary irregularly spaced time series. The only information we found was on the comparison of the predictive accuracy of two prediction methods [Diebold and Mariano, 1995]. In this work the usual assumption of independent errors is not necessary. Unfortunately, this work was not extended to multiple comparisons.

4.3 Focusing

Despite the resampling approach being used (discussed in Sect. 4.2.4) results can be improved by data manipulation. In fact, each induction algorithm has a set of parameters. *A&ps* refers to the set of an algorithm with a vector of parameters (a parameter set). For the same *a&ps* it is possible to obtain different models by data manipulation, i.e., given a training set: to use just a subset of the instances, to use a subset of the features or to use different domain of values for each input variable. The first case is called example selection (also called instance selection), the second one is called feature selection and the last one is the domain values definition. These three tasks are the focusing tasks [Reinartz, 2002].

In other words, given a table, the row selection (example selection), the column selection (feature selection) and the domain values definition are the three focusing tasks.

4.3.1 Feature selection

Several potential benefits of feature selection can be pointed out [Guyon and Elisseeff, 2003]:

- facilitation of data visualization and data understanding;
- reduction of measurement and storage requirements;
- reduction of training and utilization times;
- defying the curse of dimensionality to improve prediction performance.

[1] asymptotically means that it converges to stationarity (in this case) for larger sample sizes, where stationarity means that a certain statistic, for example the mean, stays constant along time [Makridakis *et al.*, 1998].

It is possible to add to this list the use of feature selection as a technique to generate different models in order to obtain an ensemble of predictors [Ho, 1998], a subject to discuss in section 6.2.1. Some of these benefits/goals are usually contradictory. For example, the best feature subset in order to obtain the best prediction can discard important variables for data understanding. The accuracy of the predictions will always be the main goal of this thesis.

The algorithms for feature selection are typically composed of the following three components [Aha and Bankert, 1996]:

- The search algorithm: it searches the space of feature subsets.
- The evaluation function: it computes a numerical evaluation. The goal of the search algorithm is to find the features subset that minimizes (or maximizes) this function (depending on the chosen function).
- The performance function: the performance task in this thesis is accomplished by one of the regression algorithms discussed in Sect. 4.4.

Search algorithms

In [Doak, 1992] three categories of search algorithms are identified: exponential, randomized and sequential.

The first one consists of a full search of the input space of feature subsets. It has exponential complexity $O(2^V)$ being, consequently, most often, prohibitively expensive.

The randomized ones include genetic algorithms [Vafaie and Jong, 1993; Skalak, 1994; Yang and Honavar, 1997; Oh *et al.*, 2004], simulated annealing [Loughrey and Cunningham, 2005] and ant colony optimization [Al-Ani, 2005]. The randomized algorithms are claimed to obtain high accuracies [Skalak, 1994; Oh *et al.*, 2004; Al-Ani, 2005].

The sequential search algorithms have polynomial complexity [Aha and Bankert, 1994]. The most common sequential algorithms [Aha and Bankert, 1996] are forward sequential selection (FSS) and backward sequential selection (BSS). FSS begins with zero features and at each iteration adds the one from the available set of features that optimizes the evaluation function. BSS does the same but begins with the full set of features and at each iteration it takes off the one that optimizes the evaluation function. It stops when the addition (or removal) of one more feature does not improve the evaluation function. This type of search is called hill-climbing, greedy search or steepest ascent [Kohavi and John, 1997]. The main problem of the sequential approach is that it can stop at a local optimum instead of the global one. There are other methods that try to reduce the limitations of the greedy approaches by using more intensive search strategies. Examples of such algorithms are the adaptive floating search method (the Plus-l-Minus-r search method) [Stearns, 1976; Pudil *et al.*, 1994; Somol *et al.*, 1999] and the beam search [Xu and Fern, 2007]. The forward version of the floating methods as proposed by Pudil *et al.* is claimed in [Jain and Zongker, 1997] to be the most effective suboptimal method.

All these methods evaluate subsets of features. Relief algorithm [Kira and Rendell, 1992] and their improvements ReliefF for classification and RReliefF for regression [Robnik-Šikonja and Kononenko, 2003] evaluate features instead of subsets of features. They randomly select a pre-defined number of examples

and measure the relevance of each feature in the target. The result is a list of weights. Each feature has a weight. The higher the weight, the more relevant the feature. The features with a weight above a pre-defined threshold are selected (Hall suggests 0.01 [Hall, 2000]).

In the last few years the research on feature selection has been driven mainly to problems with hundred or thousand of features [Guyon and Elisseeff, 2003]).

Evaluation functions

The evaluation functions are strongly related with the architecture of the feature selection algorithm [Blum and Langley, 1997; Al-Ani, 2005]:

- Embedded: the feature selection is embedded within the induction algorithm (an example is the CART induction algorithm [Breiman *et al.*, 1984]);
- Filter: the evaluation function does not use the results of the induction algorithm;
- Wrapper: the selection of the features is done testing them at each iteration according to the induction algorithm performance for a given evaluation criterion.

The wrapper approach is expected to give better results than the filter one [Doak, 1992; John *et al.*, 1994]. This is empirically expectable since the selection process in the wrapper case takes into account the induction algorithm to use, i.e., the evaluation and the performance functions have the same bias [John *et al.*, 1994]. The filter approach selects the features regardless of the induction algorithm to be used. The drawback of the wrapper approach, when compared with the filter one, is its computational cost.

4.3.2 Example selection

According to Liu & Motoda:

> "The ideal outcome of instance selection is a model independent, minimum sample of data that can accomplish tasks with little or no performance deterioration, i.e., for a given data mining algorithm M, its performance P on a sample s of selected instances and on the whole data w is roughly $P(Ms) \doteq P(Mw)$. By model independence, we mean that for any two data mining algorithms Mi and Mj, let $\triangle P$ be the performance difference in using data s with respect to using data w. $\triangle P(Mi) \doteq \triangle P(Mj)$" [Liu and Motoda, 2002].

This definition of example/instance selection is not consensual. According to Blum & Langley,

> "Researchers have proposed at least three reasons for selecting examples used during learning. One is if the learning algorithm is computationally intensive; in this case, if sufficient training data is available, it makes sense to learn only from some examples for purposes of computational efficiency. [...] Yet a third reason for example

> selection is to increase the rate of learning by focusing attention on informative examples, thus adding search through the space of hypotheses [...]" [Blum and Langley, 1997].

While the definition by Liu & Motoda assumes a filter approach, the approach by Blum & Langley opens space for wrapper approaches (Sect. 4.3.1).

Throughout this thesis, example selection is understood in the same way of Blum & Langley.

In the literature, example/instance selection is much less referred to than feature selection (the book [Liu and Motoda, 2001] and the survey [Jankowski and Grochowski, 2004] are two good starting references on example selection). A possible explanation is that many of the induction algorithms embed methods for example selection. For several methods the search of the most promising examples is part of the learning process. Local regression, instance based, decision trees, regression rules and support vector machines are examples of such methods. These and other methods are discussed in Sect. 4.4.

4.3.3 Domain value selection

The domain values selection can include the choice of the data type for each variable, the discretization of continuous variables, or the choice of appropriate values for a symbolic variable. The best domain values for each variable depends on the model to train [Witten and Frank, 2000].

4.4 Regression algorithms

This section presents the main regression algorithms. It does not aim to describe in detail the state of the art of each method from a researcher point of view, but simply to obtain insights from an end user point of view.

4.4.1 Parametric regression models

The basic idea under parametric regression models is, given a real function of a given form, to adjust the parameters of the function so that it fits the input data. Assuming the simplest case with just one input variable and the linear regression model, the problem consists of finding the β in the formula

$$\hat{f}(x) = \beta_0 + \beta_1 \times x, \tag{4.8}$$

in order to minimize an error function, typically the *mse*,

$$\sum_x (f(x) - \hat{f}(x))^2. \tag{4.9}$$

This is the simplest case known as simple linear regression. When using V input variables, instead of one, the equation is,

$$\hat{f}(x_1, x_2, ..., x_V) = \beta_0 + \sum_{i=1}^{V} (\beta_i \times x_i), \tag{4.10}$$

and the approach is known as multiple linear regression.

Parametric regression approaches can also use nonlinear regression, such as the polynomial or the exponential regression, among others. Any introductory book on statistics explains linear regression and how to calculate the necessary parameters. For more advanced topics on parametric regression [Kleinbaum *et al.*, 1998] is a good reference. Parametric regression approaches are characterized by having a known model. The goal is to learn the parameters of the given model, not the model itself. The following sections on regression algorithms describe methods whose goal is to learn the unknown model, not just the parameters.

4.4.2 Local regression and instance based methods

The term nonparametric is usually used by the statistical community in the context of local modelling. These methods are, typically, locally parametric but not globally parametric. They have many connections with the instance based methods, as they are known by the machine learning community.

The main idea of both local regression and instance based methods is to use regression in one point looking just for a neighborhood of this point. These methods can be classified according to the definition of this local region (the neighborhood) and, also, according to the locally used model. K-nearest neighbor is well known as a method to define the local region. Weighted average and locally weighted regression are typical local models. The former uses weighted average, whose weights are inversely proportional to the distance to the regression point. The latter applies parametric regression (usually linear regression) to the nearby instances [Atkeson *et al.*, 1997]. The local choice of the weights when parametric regression is used is known as kernel regression [Watson, 1964; Nadaraya, 1964]. Kernel regression is a particular case of local polynomial regression since kernel regression fits a polynomial of degree zero to the local instances [Torgo, 1999].

The well known instance based learner, k-nearest neighbor method defines the local region as the nearest k neighbors, according to a given distance function. One of the main issues of k-nearest neighbor is the choice of the distance function. The Euclidean distance is a standard but others exist [Wilson and Martinez, 1997]. The prediction is typically obtained by simply averaging its values. Weighted averaging is also used in the k-nearest neighbor context [Cost and Salzberg, 1993]. The K-nearest neighbors approach may degrade its performances on high dimensional data sets due to their sparsity. Some improved k-nearest neighbor methods reduce this problem. In particular, appropriate feature selection and feature weighting. Principal components [Jolliffe, 2002] can also be used in this context.

These methods can be classified as lazy learning methods in opposition to eager learning since "it is a memory-based technique that postpones all the computation until an explicit request for a prediction is received" [Aha, 1997]. One of the main drawbacks of local regression and instance based learning is their limited interpretability. References [Cleveland and Loader, 1996; Atkeson *et al.*, 1997; Mitchell, 1997] give good insights into this kind of methods.

4.4.3 Generalized additive model

The generalized additive model for regression can be expressed as,

$$\hat{f}(x_1, x_2, ..., x_V) = \alpha + \sum_{i=1}^{V}(f_i(x_i)), \tag{4.11}$$

where α is a constant and $f_i, i \in \{1, 2, ..., V\}$ are univariate basis functions [Hastie *et al.*, 2001].

The additive model, like local regression, uses the divide and conquer approach. Instead of splitting the instances (as in local regression / instance based methods), splitting is done by features, i.e., instead of solving a problem with V attributes, V simpler subproblems, each one with just one attribute, are solved.

One of the most appealing characteristics of additive models is the interpretability. In fact, it is possible to estimate the contribution of each attribute to the final prediction.

The main drawback of additive models is their computational complexity due to the large amount of possible basis functions and the respective parameter setting that is different for different basis functions [Torgo, 1999]. A second problem is the possibility of overfitting that must be solved by each particular additive model. Some interesting additive models are MART - Multiple Additive Regression Trees [Friedman, 2001], groves of trees [Sorokina *et al.*, 2007] or BART - Bayesian Additive Regression Trees [Chipman *et al.*, 2007].

Some of the models presented in the following sections can be formulated, with some adaptations, as additive models namely, decision trees, MARS - Multivariate Adaptive Regression Splines, PPR - Projection Pursuit Regression and ANN - Artificial Neural Networks.

4.4.4 Decision trees

Decision trees is an appealing family of methods, which are conceptually very simple yet very useful, mainly due to their interpretability and accuracy. In addition, their predictive performance has been recently enhanced by the use of ensemble methods (a subject to be discussed in Chap. 6). The main idea of decision trees is the use of the divide-and-conquer approach to split the data into more homogeneous subsets. The principle is to find one property that splits the data set into two distinct parts according to the output variable. One common way to proceed is to minimize the sum of the variances of the two resulting subsets. By iterating this process for the resulting subsets, the model grows as a tree-like structure, known as a binary tree because it always splits the current data set into two subsets [Breiman *et al.*, 1984]. The described algorithm is the recursive partitioning algorithm. Following the branches of the tree until a leaf node is achieved allows us to follow the criteria used by the algorithm, which can be very useful for the user. The described method, known as CART - Classification And Regression Trees, uses the simple average in the output values of the leaf nodes to obtain the prediction for regression problems.

Another approach, known as model trees, uses different models in the leaves, instead of simple average, in order to increase prediction accuracy [Quinlan, 1992; Torgo, 1999; Malerba *et al.*, 2004].

4.4.5 Regression rules

Regression rules models have two components: a set of IF-THEN rules and a consequent model to use when the rules are satisfied.

The rules are typically ordered to avoid the possibility of different results for the same input values. Ordering the rules guarantees that the model is deterministic. The set of regression rules is usually more compact than the implicit rules of a decision tree [Witten and Frank, 2000]. Like decision trees, regression rules are interpretable models. This is why this method is used to obtain explanatory insights from black box models, such as artificial neural networks (Sect. 4.4.8) [Saito and Nakano, 2002].

The consequent model can be the simple average, k-nearest neighbors [Weiss and Indurkhya, 1995] or any other regression method [Torgo, 1995].

4.4.6 MARS - Multivariate Adaptive Regression Splines

MARS is a tree based method. It uses a modified version of the recursive partitioning algorithm (Sect. 4.4.4) in order to improve the accuracy of predictions. The main focus is to surpass the lack of continuity of the resultant $\hat{f}$ function of the decision tree methods at the subregion boundaries [Friedman, 1991]. The use of adaptive regression splines surpasses this problem. Splines are a series of locally defined low order polynomials used to approximate data. Each one of the polynomial of the series is defined in order to guarantee continuity in both the function and its higher derivatives at the borders of the local regions [Cherkassky and Mulier, 1998]. The ANOVA expansion [2] of the final $\hat{f}(\mathbf{x})$ function can be expressed as,

$$\hat{f}(\mathbf{x}) = \alpha + \sum_i f_i(x_i) + \sum_{i,j} f_{i,j}(x_i, x_j) + \sum_{i,j,k} f_{i,j,k}(x_i, x_j, x_k) + \cdots . \quad (4.12)$$

4.4.7 PPR - Projection Pursuit Regression

Projection Pursuit Regression (PPR) is also an additive model. However, instead of the original features, it uses a linear combination of these. The V dimensional original space is linearly projected and the image of this projection in the univariate f_i basis functions is added, this is,

$$\hat{f}(x_1, x_2, \cdots, x_V) = \sum_{i=1}^{t} f_i \left(\sum_{j=1}^{V} (\beta_{i,j} \times x_j) \right), \quad (4.13)$$

where t is the number of iterations. The value of t is obtained on the fly, when the criterion of fit is smaller than a pre-specified threshold.

PPR was presented in 1981 [Friedman and Stuetzle, 1981] and maybe this can explain the relative lack of use of this promising method. One of the main drawbacks of PPR is its computational cost. For modern-day computers, the effort is reasonable, in contrast to the past. Another disadvantage of PPR is its difficult interpretability. However, the prediction accuracy and the ability to

[2] The ANOVA expansion is obtained by decomposing orthogonally a given function [Karlin and Rinott, 1982].

bypass the curse of dimensionality are two of the most important characteristics of PPR [Huber, 1985]. An important result of PPR is that it is a universal approximator, i.e., for t arbitrarily large and for an appropriate choice of the f_i basis functions, PPR can approximate any continuous function in $\Re^V$ [Hastie *et al.*, 2001].

4.4.8 ANN - Artificial Neural Networks

Although they have been developed in different research communities, PPR and Artificial Neural Networks - ANN have much in common. Both are additive models in the derived features as well as universal approximators [Hastie *et al.*, 2001]. ANN was developed with the intention of artificially reproducing the mechanisms of biological nervous systems. In Fig. 4.1 a typical ANN architecture for regression is presented:

- A set of V input variables;
- One hidden layer with P derived features;
- One output variable, as usual for regression (for classification, one output variable can be used per each different value of the target variable).

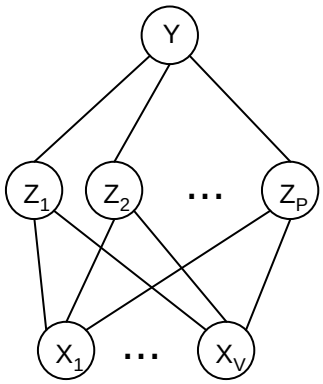

Figure 4.1: A typical ANN architecture for regression using one hidden layer.

The derived features (Z) are calculated as,

$$Z_p = \sigma\left(\alpha_{0p} + \sum_{v=1}^{V} \alpha_{pv} x_v\right), \tag{4.14}$$

and y is

$$y = f(\mathbf{x}) = g\left(\beta_0 + \sum_{p=1}^{P} \beta_p Z_p\right), \tag{4.15}$$

where $\mathbf{x} = (x_1, x_2, \cdots, x_V)$. The two most common activation functions σ are the sigmoid function and the Gaussian radial basis function. The output function (g) for regression is typically the identity function. A good introduction to ANN can be found in [Basheer and Hajmeer, 2000], for example. In [Hastie *et al.*, 2001] there is an interesting comparison between PPR and ANN.

4.4.9 SVM - Support Vector Machines

Support vector machines (SVM) is a novel family of successful algorithms for both regression and classification. Based on the statistical learning theory (VC theory) developed during the sixties/seventies, it was mainly in the nineties, when Vladimir Vapnik began to work at AT&T Bell Laboratories, that SVM were largely developed. Nowadays SVM are an area of research in there own right with a large number of successful applications. This section follows closely [Smola and Scholkopf, 2004].

The basic idea of $\varepsilon-$SVM, the most used SVM algorithm, is to learn $\hat{f}$ such that the error for the training data is the minimum possible higher than a predefined ε and, simultaneously, keeping $\hat{f}$ as flat as possible. This is achieved using the $\varepsilon-$insensitive loss function:

$$|y - \hat{f}(\mathbf{x})|_\varepsilon = \begin{cases} 0 & , \text{ if } |y - \hat{f}(\mathbf{x})| \leq \varepsilon \\ |y - \hat{f}(\mathbf{x})| - \varepsilon & , \text{ otherwise} \end{cases} . \tag{4.16}$$

The examples that minimize the second part of the loss function, thus implicitly defining the margin, are called support vectors.

For the case of linear functions (see Eq. 4.10), the problem can be written as a linear programming one:

$$\begin{aligned} &\text{minimize } \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C \sum_{i=1}^{M} (\xi_i + \xi_i^*) \\ &\text{subject to: } \begin{cases} ((\mathbf{w} \cdot \mathbf{x}_i) + \beta_0) - y_i \leq \varepsilon + \xi_i \\ y_i - ((\mathbf{w} \cdot \mathbf{x}_i) + \beta_0) \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} , \end{aligned} \tag{4.17}$$

where $\mathbf{w} = (\beta_1, \beta_2, \cdots, \beta_V)$ (according to Eq. 4.10), ξ_i and ξ_i^* are slack variables, and ε and C are input parameters. ε is the limit for non-penalized errors and C determines the trade-off between the flatness of $\hat{f}$ and the tolerance to errors larger than ε. The flatness of $\hat{f}$ is guaranteed by the first term of the objective function.

After some reformulation, using the dual problem and generalization for the nonlinear case, the problem takes the form:

$$\begin{aligned} \text{minimize } &\frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \text{Kn}(\mathbf{x}_i, \mathbf{x}_j) \\ &+ \varepsilon \sum_{i=1}^{M} (\alpha_i - \alpha_i^*) - \sum_{i=1}^{M} y_i (\alpha_i - \alpha_i^*) \\ &\text{subject to: } \begin{cases} \sum_{i=1}^{M} (\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \in [0, C] \end{cases} , \end{aligned} \tag{4.18}$$

where Kn is a kernel function. This function is of major importance in the SVM algorithm since it guarantees the generalization for the nonlinear space. Research exists to define appropriate kernels for specific types of problems.

A more complete overview on support vector regression can be obtained in [Smola and Scholkopf, 2004; Vapnik, 1995].

4.4.10 Ensemble methods

Another successful approach is the use of an ensemble of models, instead of just one, for the prediction task. The main idea is to generate different models in such a way that the combination of their results reduces the generalization error, when compared to the use of just one model. Bagging [Breiman, 1996a] and Random Forest [Breiman, 2001a] are two successful ensemble regression methods using decision trees as base learners. Negative correlation learning [Liu *et al.*, 2000] is an example of neural networks ensembles. Ensemble methods for regression are discussed in detail in Chap. 6.

Chapter 5

Experiments using different algorithms

The purpose of this chapter is to evaluate different prediction methods in order to maximize accuracy (as stated in Sect. 2.4). The objective is to use the predictions for duties' definition. For this reason, the problem has a three-day prediction horizon, as explained in Sect. 2.3. Due to both the complexity of the problem and limitations in the evaluation of the predictions impact on the business objectives (Sect. 2.3), this chapter, as well as all the third part of this thesis, addresses only the accuracy issue. The impact of the predictions on the business objectives is not evaluated in this thesis.

Firstly, the choice of input variables (Sect. 5.1) and tested algorithms (Sect. 5.2) is discussed, next the experimental setup used along this chapter is presented (Sect. 5.3), followed by the presentation of a baseline method to use as reference for comparison (Sect. 5.4) and an expert-based method (Sect. 5.5). Then, four different sets of experiments are described. The first one tests each one of the algorithms with different parameter sets. The other tests consist of experiments for each one of the focusing tasks described in Sect. 4.3: example selection, domain value selection and feature selection (Sect. 5.7). Finally, we discuss possible areas of research with the aim of reducing the generalization error even further.

5.1 Choosing the input variables

Some of the input variables chosen are a direct consequence of the data analysis task discussed in Sect. 2.5.2. The variables were obtained as follows:

- Seasonality: for each seasonality a variable is used that identifies the relevant seasonality indexes, in other words, variables that can identify the position in the seasonality cycle;
- Impact factor: a variable is used for each impact factor.

In Table 5.1 the input variables for each one of the seasonalities/impact factors are presented. For the seasonality of the year, we use two variables, because, initially, it was not clear which of them would be more valuable.

Table 5.1: Variables detected by visual inspection.

Seasonality / Impact factor	**Input variable**	**Data type**
Daily seasonality	Departure time	Numeric (in seconds)
Weekly seasonality	Week day	Symbolic
Seasonality of the year	Day of the year	Numeric
	Week of the year	Numeric
Holiday impact	Day type	{holiday, bridge day, tolerance day, normal }
School breaks impact	School break	{break, normal}
Pay day impact	Sundays until next pay day	Numeric

Since the SAEI database only stores data during one year, it is not possible to identify the seasonality of the year. In any case, the variables *day of the year* and *week of the year* are used in the expectation of identifying variations along the year.

The remaining input variables were suggested by experts (Table 5.2):

- Bus type: it is the name of the bus type. It implicitly identifies the bus size and the type of energy used.
- Driver: it is an identifier of the driver. If there are reliefs, i.e., the trip has more than one driver, it is assigned the 0 value.
- Service: the service is a concept used in the SAEI. Whenever changes occur in the schedule or in the route, the service changes. However, it is not possible to quantify how relevant the changes are.
- Entrance flow: this variable tries to identify potential situations of increment of traffic flow in the direction of the town entrances. 'we3', 'we4' and 'holidays' identify, respectively, the end of a long weekend of 3 days, a long weekend of 4 days, and holidays.
- Exit flow: it is similar to the entrance flow but for traffic flow in the direction of the town exits.
- Weather variables: the variables temperature, precipitation and wind speed try to identify weather conditions with influence on travel time [1].

The data types presented in Tables 5.1 and 5.2 were used as default. Other data types could have been used. This is a subject to be discussed in Sect. 5.7.2.

5.2 Choosing the regression methods

A first step prior to the choice of the induction algorithms to be used is to know what characteristics they should have. Often, the learners are classified

[1] These variables were suggested by the expert on meteorology, Alfredo Rocha (PhD) from Aveiro University.

Table 5.2: Variables suggested by experts.

Input variable	**Data type**
Bus type	Symbolic
Driver	Symbolic
Service	Symbolic
Entrance flow	{we3, we4, holidays, normal}
Exit flow	{we3, we4, holidays, normal}
Wind speed	Numeric (in m/s)
Temperature	Numeric (in ° Celsius)
Precipitation	Numeric (in mm, $1mm = 1lt/m^2$)

according to their interpretability and predictive powers. Other characteristics exist to classify them, as pointed out in [Hastie *et al.*, 2001]. Anyway, for TTP, the predictive capability is the one that shall be optimized. In fact, from the point of view of the goal of this research, the usefulness of interpretability is to get insights into the contribution of each input variable to the prediction performance. The interpretability from the end users point of view is not important because the variables that could be useful for them are not the ones that are important for the improvement of the prediction's accuracy. An example is the existence of bus lanes. The impact of bus lanes is relevant for business negotiations between the STCP company and the Oporto authorities, namely the town hall, but this variable is not interesting from the point of view of the prediction's accuracy because it is implicit in the choice of the route. Several other variables exist in the same situation.

In [Hastie *et al.*, 2001], ANN (Sect. 4.4.8), SVM (Sect. 4.4.9) and instance local regression / instance based methods (Sect. 4.4.2) are the ones reported to be the most accurate when compared against decision trees (Sect. 4.4.4) and MARS (Sect. 4.4.6).

In any case, the main reference for the choice of the induction algorithms to use for TTP was the work by Meyer et al. done on nine real and three artificial benchmark data sets. They test SVM, linear regression (Sect. 4.4.1), decision trees (the recursive partitioning algorithm), ANN, MARS, BRUTO (similar to MARS but cannot handle categorical input variables), MART (Sect. 4.4.3), PPR (Sect. 4.4.7) and two decision tree-based ensemble algorithms, Random Forests and bagging (to be discussed in Sect. 6.2). Globally ANN, SVM, PPR and Random Forest are the best [Meyer *et al.*, 2003].

These kinds of benchmark tests are never fully convincing because there are several choices that must be made that can influence the results. An example is the set of features used in the training process. The set of features can be optimal for a particular induction algorithm but suboptimal for others. Although many other examples exist, choices must be made. Furthermore, when the results obtained agree with the overall perception of the literature on regression methods, the confidence over the decisions is stronger, as is the present case [Smola and Scholkopf, 2004].

From the four induction algorithms previously referred, ANN was not used. The reason is the strong time consuming requirements needed to train and test the different algorithms. After the first experiments with the other three

methods, experiments with ANN were postponed. Its stochastic nature helped in this decision. From the benchmark test by Meyer et al. there is no evidence of the advantage of any of the four methods mentioned over the other three.

5.3 Experimental setup

As previously mentioned in Sect. 2.5.2, the experiments described in this chapter use data from route 78-1-1. In this chapter, all the experiments use data just from January 1st to March 31st of 2004. As planned, we are using data from just one route. A similar approach is used by Petridis et al., for sugar beet yield prediction. In their case, data from each different geographical region is treated as a different data set [Petridis *et al.*, 2001].

The set of variables used were {departure time, week day, day of the year and day type}, i.e., one variable for each of the first four seasonalities / impact factors presented in Table 5.1. In Sect. 5.7.3, we present a study on the variables to use.

The generalization error is assessed using the *variation index* function (Eq. 4.5). The reason for using this function is three-fold, the quadratic measure of the distance (which penalizes the distance more than a linear measure), its interpretability and the fact that it is a known measure in the context of operations planning at mass transit companies [Strathman *et al.*, 1998; Turner *et al.*, 1998]. The *variation index* gives the ratio between a dispersion measure and the average.

The resampling method (Sect. 4.2.4) is a sliding window with a 30-day time stamp (Fig. 5.1). This sliding window is used because it is expected that the last days are the ones that can give more information on what will happen three days ahead. The use of a time stamp-based window against a sequence-based one, as well as the use of 30 days of data for training, has no scientific support. Since the SAEI database keeps just one year of data, it is not possible to capture the seasonality of the year. The maximum window size that could be used is one year. But as we have data just from January 1 2004, and the following experiments were done at the end of 2004, we have decided to use just 30 days of data for each window. For this particular data set, the average number of trips for each 30-day window is around 9 hundred. Nevertheless, the most adequate window size can and should be studied also because fixing the number of days is oriented towards constraining the computational complexity instead of improving the accuracy. This problem is discussed in [Wang *et al.*, 2003]. However, it is not the focus of this thesis.

All the experiments use the R-project [Team, 2006].

5.4 A baseline method

In the spirit of a common recommendation on prediction methods [Makridakis *et al.*, 1998], we start the experiments using a naive approach to be used as a baseline method. The goal is, obviously, to obtain the first results for comparison with more sophisticated approaches.

Baseline methods are often used to measure the gains in accuracy using more advanced methods. Makridakis et al. propose two methods, named Naive

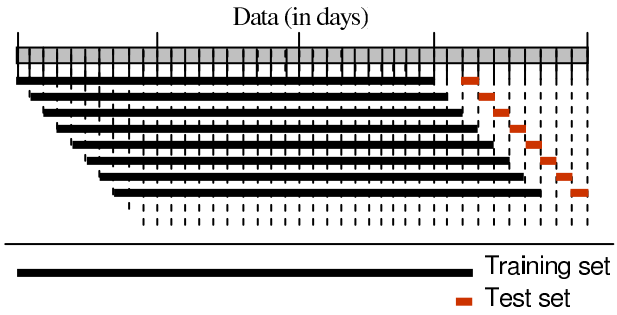

Figure 5.1: Experimental setup using a time stamp sliding window.

Forecast 1 (NF1) and Naive Forecast 2 (NF2) [Makridakis *et al.*, 1998]. NF1 uses as prediction the target value of the most recent example. When the data set is not stationary (Sect. 4.2.5), NF1 typically gives very inaccurate predictions. NF2 previously removes the seasonality of the data and then it applies the NF1 [Makridakis *et al.*, 1998]. NF2 is a more accurate method when compared to NF1. NF2 cannot be used for TTP because the data set is irregularly time spaced.

The baseline method we use has two steps: (1) the first step filters data regarded as equivalent to the day we want to predict; and (2) the second one uses the filtered data set from which to select the nearest past example.

The first component filters the data according to the equivalent day's group on which the data to predict falls. If the equivalent day's group is empty then all data is used at the second step. These groups are:

- Day Type = Normal and working days (from Monday to Friday)
- Day Type = Normal and Saturdays
- Sundays
- Day Type = bank holiday and weekday on Monday, Friday
- Day Type = bank holiday and weekday on Tuesday, Thursday
- Day Type = bank holiday and Wednesday
- Day Type = bank holiday and Saturdays
- Day Type = bridge day
- Day Type = tolerance day

The equivalent day's groups were defined by visual inspection and using experts knowledge.

The second component uses a distance measure to obtain the nearest neighbor. The selected distance measure is the Heterogeneous Euclidean-Overlap Metric (HEOM) because it is one of the mentioned measures that handle numeric and nominal attributes simultaneously [Wilson and Martinez, 1997]:

$$heom(\mathbf{x}, \mathbf{t}) = \sqrt{\sum_{a=1}^{v} d_a(x_a, t_a)^2}, \tag{5.1}$$

where

$$d_a(x,t) = \begin{cases} 1 & , \text{ if } x \text{ or } t \text{ is unknown, else} \\ overlap(x,t) & , \text{ if } a \text{ is nominal, else} \\ rn\_diff_a(x,t) & , \end{cases}, \tag{5.2}$$

$$overlap(x_1, x_2) = \begin{cases} 0 & , \text{ if } x_1 = x_2 \\ 1 & , \text{ otherwise} \end{cases}, \tag{5.3}$$

and

$$rn\_diff_a(x_1, x_2) = \frac{|x_1 - x_2|}{max_a - min_a}. \tag{5.4}$$

v is the number of input variables (also named attributes), min_a and max_a are, respectively, the minimum and the maximum values of attribute a.

The baseline method calculates the HEOM distance between the test instance (the $\mathbf{t}$ variable in Eq. 5.1) and each element $\mathbf{x}$ of the training set that belongs to the same equivalent day. The target value of the instance with the shortest heom distance is the prediction of the baseline method.

Using the experimental setup described in Sect. 5.3, the *variation index* obtained with the baseline method was 12.32%. This value will be the reference for comparison from now on. However, it is important to notice that this value depends on the set of features used for training.

5.5 An expert-based method

We also developed an algorithm after the data analysis and before the use of any inductive learning algorithm. We name it the expert-based algorithm. While the baseline method is a simple one, the expert-based method aims (with all the obvious limitations) to test inductive learning algorithms against experts.

Our expert-based algorithm tries to represent the existing expert knowledge on TTP three days ahead. To call it 'the expert-based algorithm' is necessarily arguable because each expert would write a different algorithm. This one was the result of some months of experience working at the operational department of the STCP company and it was developed before the experiments with machine learning approaches. It is also necessarily the result of our previous background of nearly 10 years working on planning projects for the operations of public transport companies. The algorithm uses the four input variables referred to in Sect. 5.3.

The expert-based algorithm (Fig. 5.2) has three parameters:

- *min.ex*: is the minimum number of examples to retrieve;
- *margin*: is the maximum allowed difference (in seconds) between the travel times of the test example and of the similar examples;
- *max.incr*: maximum allowed increments of the margin.

Require: *data*, the training data set
Require: *test.ex*, the test example
Require: *min.ex*, the minimum number of examples to search for
Require: *margin*, the maximum allowed difference between the departure times of *test.ex* and of the similar examples
Require: *max.incr*, maximum allowed increments of *margin*
1: *sel.ex* := *first.search(data, test.ex, min.ex, margin, max.incr)* {the function *first.search* is defined in Fig. 5.3}
2: **if** *size(sel.ex)*>0 **then**
3: *result* := *mean(*travel time from *sel.ex)*
4: **else**
5: *sel.ex* := *second.search(data, test.ex, min.ex, margin, max.incr)* {the function *second.search* is defined in Fig. 5.4}
6: **if** *size(sel.ex)*>0 **then**
7: *result* := *mean*(travel time from *sel.ex*)
8: **else**
9: *result* := *mean*(travel time from *data*)
10: **end if**
11: **end if**
12: **return** *result*

Figure 5.2: The expert-based algorithm.

The main idea of the algorithm is to select data similar to the test example *test.ex* and to use this data to obtain the prediction by averaging their travel time. The retrieval of similar data is done in three steps using, at each successive step, a softer criterion to find similar data if the previous one was not able to find any similar example. The first step uses the examples from the group where *test.ex* falls, as defined in Sect. 5.4 (Fig. 5.3); the second one uses the examples from Saturdays if the test example is from a Saturday, from Sundays if the test example is from a Sunday and from day types $\neq$ 'normal' if the test example is from one of these days (Fig. 5.4); and the last one uses all the examples from the training set.

Require: *data*, the training data set
Require: *test.ex*, the test example
Require: *min.ex*, the minimum number of examples to search for
Require: *margin*, the maximum allowed difference between the departure times of *test.ex* and of the similar examples
Require: *max.incr*, maximum allowed increments of *margin*
1: *data* := *equivalent.day.data(data, test.ex)* {the function *equivalent.day.data* returns the examples from the group where the *test.ex* falls, as defined in Sect. 5.4}
2: *sel.ex* := *nearby.examples(data, test.ex, min.ex, margin, max.incr)* {the function *nearby.examples* is defined in Fig. 5.5}
3: **return** *sel.ex*

Figure 5.3: The function *first.search*.

In the first two steps, there is an additional selection of the examples by

Require: *data*, the training data set
Require: *test.ex*, the test example
Require: *min.ex*, the minimum number of examples to search for
Require: *margin*, the maximum allowed difference between the departure times of *test.ex* and of the similar examples
Require: *max.incr*, maximum allowed increments of *margin*
1: **if** weekday of *test.ex* $\in$ {Saturday, Sunday} **then**
2: *data* := select from *data* the examples with the same weekday
3: *sel.ex* := *nearby.examples(data, test.ex, min.ex, margin, max.incr)* {the function *nearby.examples* is defined in Fig. 5.5}
4: **else if** day type of *test.ex* $\neq$ 'normal' **then**
5: *data* := select from *data* the examples with day type $\neq$ 'normal'
6: *sel.ex* := *nearby.examples(data, test.ex, min.ex, margin, max.incr)*
7: **end if**
8: **return** *sel.ex*

Figure 5.4: The function *second.search*.

selecting just the ones whose departures times are inside an interval centered in the departure time of *test.ex* $\pm$ *margin*. The value of *margin* is initially the input parameter *margin*, and is increased by the quantity *margin* until there is at least *min.ex* in the interval or the number of times *margin* was increased is larger than *max.incr*. Then the selected examples are ordered according to the data proximity to *test.ex*. The final selected examples are the first *min.ex* examples, if the number of selected examples is $>$ *min.ex*, or the current selected examples, otherwise. The pseudo-code of this last phase described in the current paragraph is presented in Fig. 5.5.

Require: *data*, the training data set
Require: *test.ex*, the test example
Require: *min.ex*, the minimum number of examples to search for
Require: *margin*, the maximum allowed difference between the departure times of *test.ex* and of the similar examples
Require: *max.incr*, maximum allowed increments of *margin*
1: $i := 0$
2: *new.margin* := *margin*
3: **repeat**
4: *sel.ex* := examples from *data* with departure time in the interval: departure time of *test.ex* $\pm$ *new.margin*
5: *new.margin* := *new.margin* + *margin*
6: $i := i + 1$
7: **until** *size(sel.ex)* $\geq$ *min.ex* AND $i \geq$ *max.incr*
8: order *sel.ex* by nearest days to the *test.ex*
9: *sel.ex* := select the first *min(size(sel.ex),min.ex)* from *sel.ex*
10: **return** *sel.ex*

Figure 5.5: The function *nearby.examples*.

The results for the expert-based method are expected to vary according to the used input parameters. This is true for all the methods that have input

Table 5.3: Input parameters for the expert-based method.

min.ex	**margin**	**max.incr**
$6 + 2\text{idx}_1$	$100 + 200\text{idx}_1$	idx_2
$\text{idx}_1 = 1, 2, \cdots, 7; \text{idx}_2 = 1, 2, \cdots, 10$		

parameters. For this reason, in the following section experiments are presented to tune the parameter set for each one of the algorithms being tested, including the expert-based one. Only then will it be possible to compare them.

5.6 Tuning parameter sets

This section describes experiments for tuning the parameter set for each of the selected algorithms that have input parameters. Only the baseline method does not have parameters to tune. Consequently, experiments are described using the expert-based method, Support Vector Machines (SVM), Random Forests (RF) and Projection Pursuit Regression (PPR).

5.6.1 Tuning parameter sets for the expert-based method

The experiments use the parameters presented in Table 5.3. Each *a&ps* (as defined in Sect. 4.3) is represented by m[*min.ex* index]g[*margin* index]i[*max.incr* index]. In the results (Fig. 5.6), the straight line represents the baseline prediction (Sect. 5.4). It is represented just for comparison. Despite the existence of three input parameters, the algorithm does not seem to be very sensitive to any of them, at least for the used range.

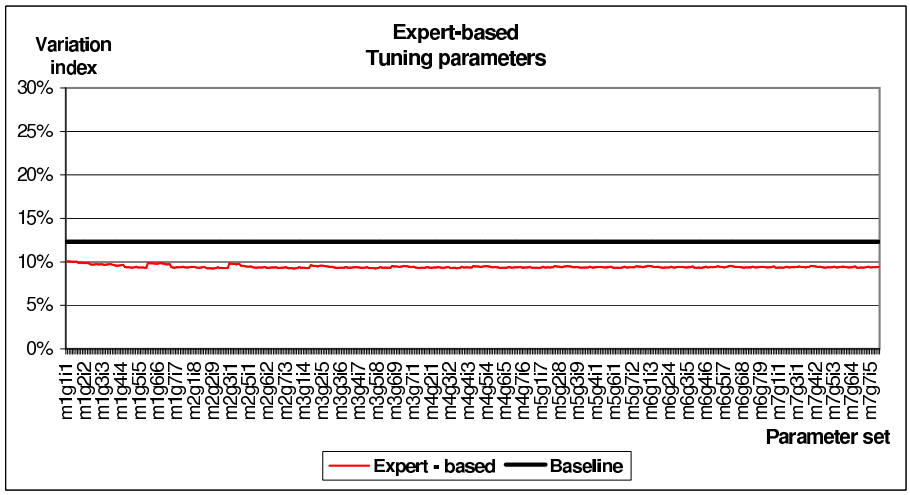

Figure 5.6: The *variation index* for the expert-based method using different parameter sets.

5.6.2 Tuning parameter sets for SVM

[Scholkopf *et al.*, 2000] propose an alternative formulation to the one given by Eq. 4.18. The motivation for this new formulation, called $\nu-$SVM, is that $\varepsilon-$SVM needs to give the desired accuracy of the approximation as an input parameter (the ε parameter) while in most cases the goal is to obtain a $\hat{f}$ function as accurate as possible. $\nu-$SVM automatically minimizes ε. However, it is not expected that one of the formulations will give better results than the other. The ν parameter is an upper bound on the fraction of errors and a lower bound on the fraction of support vectors. An important difference for tuning the parameter set is that $\nu \in [0, 1]$ while $\varepsilon \in [0, +\infty[$. Consequently, it is easier to tune ν than ϵ. The $\nu-$SVM formulation for regression is

$$\text{minimize } \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\text{Kn}(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{M} y_i(\alpha_i - \alpha_i^*) \qquad (5.5)$$

$$\text{subject to: } \begin{cases} \sum_{i=1}^{M}(\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \in \left[0, \frac{C}{M}\right] \\ \sum_{i=1}^{M}(\alpha_i + \alpha_i^*) \leq C \cdot \nu \end{cases} .$$

The kernels used were [Dimitriadou *et al.*, 2006]:

- linear: $\text{Kr}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$;
- Gaussian radial basis function: $\text{Kr}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \parallel \mathbf{x}_i - \mathbf{x}_j \parallel^2)$;
- sigmoid: $\text{Kr}(\mathbf{x}_i, \mathbf{x}_j) = tanh(\gamma \mathbf{x}_i \cdot \mathbf{x}_j + coef0)$.

It is not our goal to define what is an algorithm in the context of SVM, but we assume that different kernels induce different algorithms.

SVM are sensitive to the scale used by the numeric variables. In order to avoid this problem, a common approach is to standardize those variables to have zero mean and standard deviation one [Hastie *et al.*, 2001]. This is done by default in the SVM implementation used in the R language.

While ν and C are needed whatever the kernel is, other parameters exist that are kernel dependent, namely, γ (radial and sigmoid kernels) and *coef0* (sigmoid kernel).

The experiments use the parameters presented in Table 5.4. The used values were obtained firstly from [Meyer *et al.*, 2003] and secondly by preliminary tests in order to get some insights into the range of values to use. For SVM, each *a&ps* is represented by c[C index]n[μ index]g[γ index]f[*coef0* index] (Table 5.4).

The results for each kernel are presented in Figs. 5.7, A.1 and A.2. The straight line represents the baseline prediction (Sect. 5.4).

Since the goal is to find the best parameter set for each algorithm, there is no interest in a deep analysis of the sensitivity of each algorithm to the parameters. Additionally, the range of values tested in those experiments was selected from previous experiments using a larger and wider grid (in particular for unbounded parameters). Consequently, the results are already conditioned by these previous selections. However, some general conclusions are drawn:

- Results are clearly worse than the baseline method whatever the kernel or the parameter set used;

Table 5.4: Input parameters for SVM.

algorithm	cost (C)	nu (ν)	gamma (γ)	coef0
SVM - linear	2^{2idx_1}	$\frac{\text{idx}_2}{10}$		
SVM - radial	$2^{2\text{idx}_3} \times 1000$	$\frac{\text{idx}_4}{5}$	$6\text{idx}_5/100000$	
SVM - sigmoid	$2^{2\text{idx}_3} \times 1000$	$\frac{\text{idx}_4}{5}$	$(2 + 12\text{idx}_6)/1000000$	-0.5idx_7

$\text{idx}_1 = -2, -1, \cdots, 6; \text{idx}_2 = 1, 2, \cdots, 10; \text{idx}_3 = 1, 2, \cdots, 5; \text{idx}_4 = 1, 2, \cdots, 4; \text{idx}_5 = 1, 2, 3; \text{idx}_6 = 0, 1, \cdots, 4; \text{idx}_7 = -1, 0, \cdots, 4$

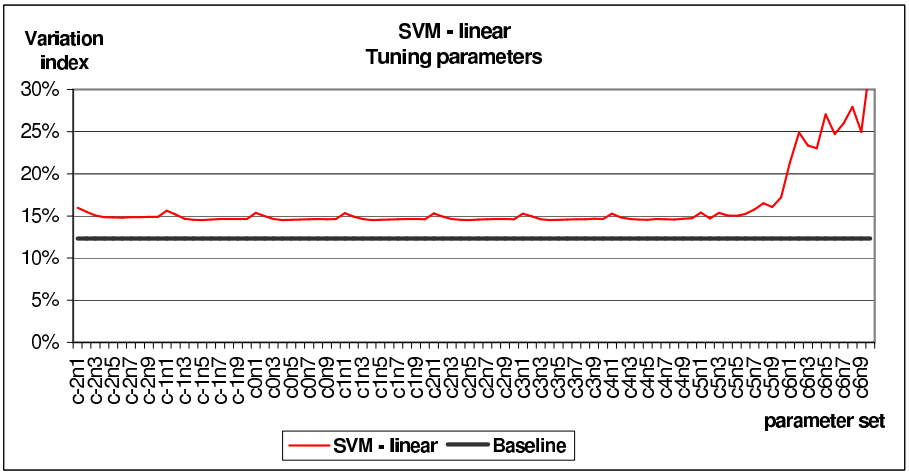

Figure 5.7: The *variation index* for SVM - linear using different parameter sets.

- Since C determines the trade-off between the flatness of $\hat{f}$ and the tolerance to errors, it is expected that the larger C is, the larger the number of support vectors will be. Also, the larger ν is, the larger the number of support vectors is. As the SVM computational complexity is given by $O(N_{SV}^3 + N_{SV}^2 \times M + N_{SV} \times V \times M)$ [Tao and Tang, 2004], (N_{SV} is the number of support vectors and M and V are, respectively, the training set size and the number of input variables, as usual) the larger C and ν are the larger the computational time for training is.

5.6.3 Tuning parameter sets for RF

Random Forest (RF) is an ensemble model (Sect. 4.4.10). It constructs a predefined number of trees (it is one of the input parameters of the method, called *ntree*) and averages the result obtained by the set of trees. The CART algorithm described in Sect. 4.4.4, is manipulated by selecting randomly, at each split, the set of variables from where the split variable is selected [Breiman, 2001a]. The number of such variables is another input parameter, named *mtry*. Several other input parameters exist but are not relevant for the prediction accuracy [Breiman, 2003; Liaw and Wiener, 2002]. They are important for the interpretability and consequently they are not discussed here. A property of RF is that its result

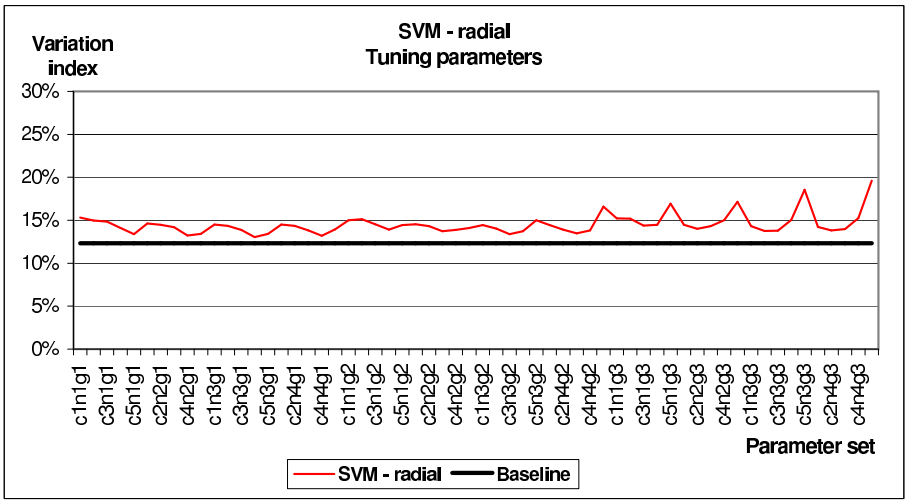

Figure 5.8: The *variation index* for SVM - radial using different parameter sets.

converges with the increasing of *ntree*. In all the experiments with RF, *ntree* is set to 1000 as suggested in [Breiman, 2003]. The values for *mtry* are, obviously, $mtry \in \{1, 2, \cdots, V\}$. In the present case, $mtry \in \{1, 2, 3, 4\}$.

The results are presented in Fig. 5.10. Some lessons can be drawn from their analysis:

- Random Forest is a very appealing 'off-the-shelf' method because parameter tuning is easy (there is only one relevant parameter that is two side bounded) and performs well. Additionally it is interpretable (in this case some more parameters are needed).
- RF is not very sensitive to the value of *mtry*. This is apparent from Fig. 5.10 and is confirmed in [Breiman, 2001a].
- Despite the fact that RF is not a deterministic model, due to its random nature, just one result of each experience is shown and the variance of the result is ignored. This happens because the *variation index* for different runs of the same experience, using 1000 trees, never differed more than four units in the fourth decimal number.

5.6.4 Tuning parameter sets for PPR

Using the description given by the R project about the *ppr* function,

> "The algorithm first adds up to *max.terms* ridge terms one at a time; it will use less if it is unable to find a term to add that makes sufficient difference. It then removes the least 'important' term at each step until *nterms* terms are left.
>
> The levels of optimization (argument *optlevel*) differ in how thoroughly the models are refitted during this process. At level 0 the existing ridge terms are not refitted. At level 1 the projection directions are not refitted, but the ridge functions and the regression

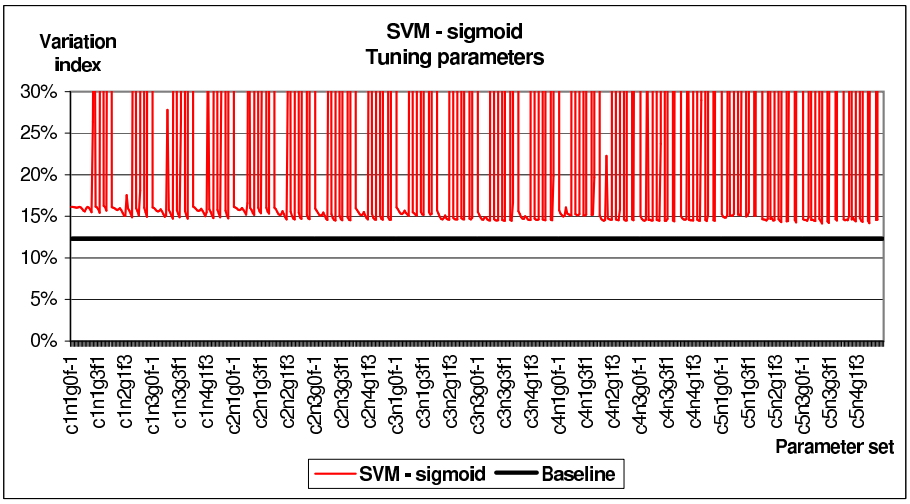

Figure 5.9: The *variation index* for SVM - sigmoid using different parameter sets.

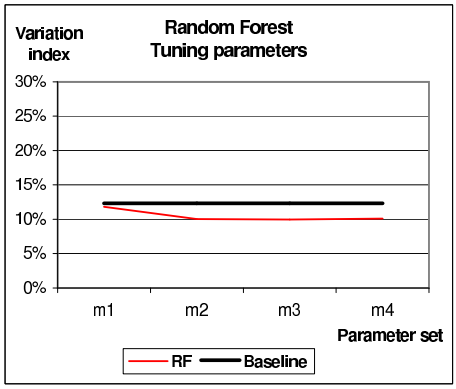

Figure 5.10: The *variation index* for RF using different parameter sets.

> coefficients are. Levels 2 and 3 refit all the terms and are equivalent for one response; level 3 is more careful to re-balance the contributions from each regressor at each step and so is a little less likely to converge to a saddle point of the sum of squares criterion" [Team, 2006].

In all the experiments done with PPR we use *max.terms* as the number of input variables (in this case, *max.terms*= 4). For experiments on parameter tuning, $nterms \in \{1, 2, 3, 4\}$ and $optlevel \in \{0, 1, 2, 3\}$.

Furthermore, PPR has a method for smoothing the ridge functions, called smoothers. As for SVM with the kernels, we assume that different smoothers induce different algorithms. The *ppr* function from the R-project [Team, 2006] has three different options for smoothers:

- The super smoother (*supsmu*), proposed by Friedman, runs a moving

Table 5.5: Input parameters for PPR.

algorithm	bass	span	df	gcvpen
PPR - supsmu	idx_1	0		
	0	$\text{idx}_2/10$		
PPR - spline			2^{idx_1}	
PPR - gcvspline				2^{2*idx_3}

$\text{idx}_1 = 0, 1, \cdots, 10; \text{idx}_2 = 1, 2, \cdots, 10; \text{idx}_3 = -2, -1, \cdots, 6$

average on a pre-defined span, where the input parameter *span* $\in [0, 1]$. If *span* is 0, then span is set by local cross validation. In this case, there is another parameter called bass tone control, where *bass* in an integer from 0 to 10. The higher is *bass* the smoother is the function.

- The splines (*spline*) are, as already mentioned in Sect. 4.4.6, a series of locally defined low order polynomials used to approximate data. They have an input parameter, the degrees of freedom number (*df*), which controls the smoothness of the ridge functions.

- The Generalized Cross-Validation (GCV) spline (*gcvspline*) is identical to *spline* but the degrees of freedom are obtained using GCV. GCV is an approximation for Jack-nife (Sect. 4.2.4), for linear fitting under a squared loss function [Hastie *et al.*, 2001]. It has an input parameter (*gcvpen*) to set "the penalty used in the GCV selection for each degree of freedom used" [Team, 2006].

The set of values tested for those input parameters that depend on the smoother is presented in Table 5.5.

The results for each one of the smoothers are presented in Figs. 5.11, A.3 and A.4.

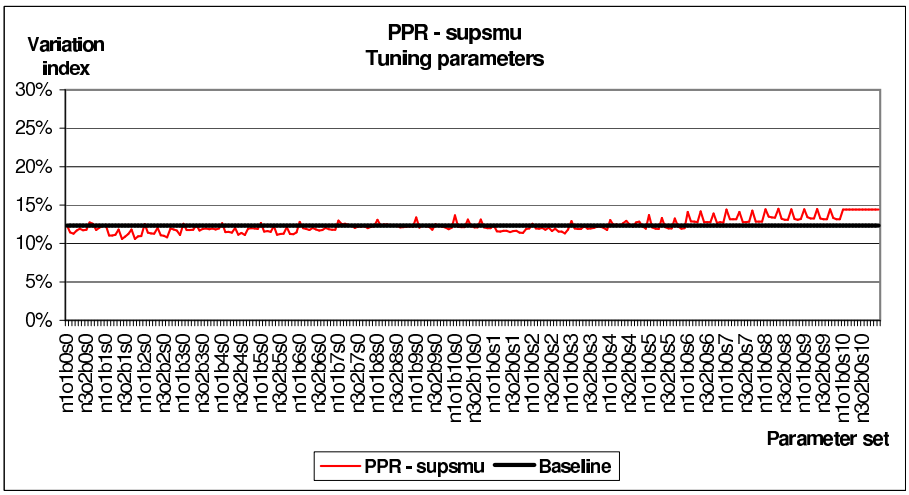

Figure 5.11: The *variation index* for PPR - supsmu using different parameter sets.

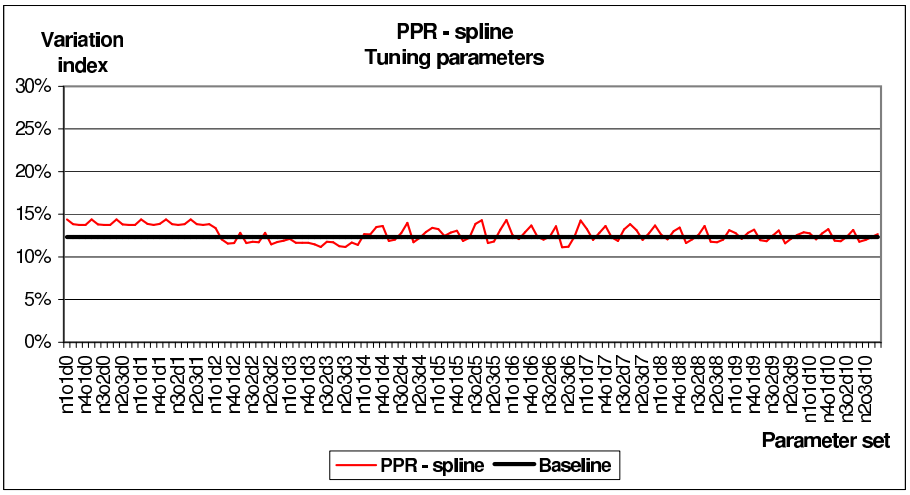

Figure 5.12: The *variation index* for PPR - spline using different parameter sets.

Some comments on the results with PPR are necessary:

- Results for PPR - supsmu have two distinct parts. The first one is easily identifiable because the determination of the parameter set finishes with $s0$ (i.e., *span*$= 0$). It corresponds to set span by local cross validation. When the identification of the parameter set finishes with an index for s larger than 0, the span is directly set. Results using local cross validation to set span are better.
- The super smoother is the one that gives the best result among the three available smoothers in R.

5.7 Data manipulation

As explained in Sect. 4.3, the data can be manipulated in order to increase accuracy and reduce the computational cost, among other benefits. All the three focusing tasks can be used to increase accuracy, while example and feature selection can also be used to address the computational cost issue. In fact, the computational complexity for each of the learners is a function of M, the number of examples in the training set, and V, the number of features/variables:

- SVM: $O(N_{SV}^3 + N_{SV}^2 \times M + N_{SV} \times V \times M)$ [Tao and Tang, 2004], where N_{SV} is the number of support vectors;
- RF: $O(\sqrt{V} \times M \times \log M)$ [Rudnicki *et al.*, 2006];
- PPR: $O(t \times V \times M \times \log(M))$ [Friedman and Stuetzle, 1981] where t is the number of iterations (Sect. 4.4.7).

This section follows the structure of Sect. 4.3 by this order: (1) example selection; (2) domain values selection; and (3) feature selection.

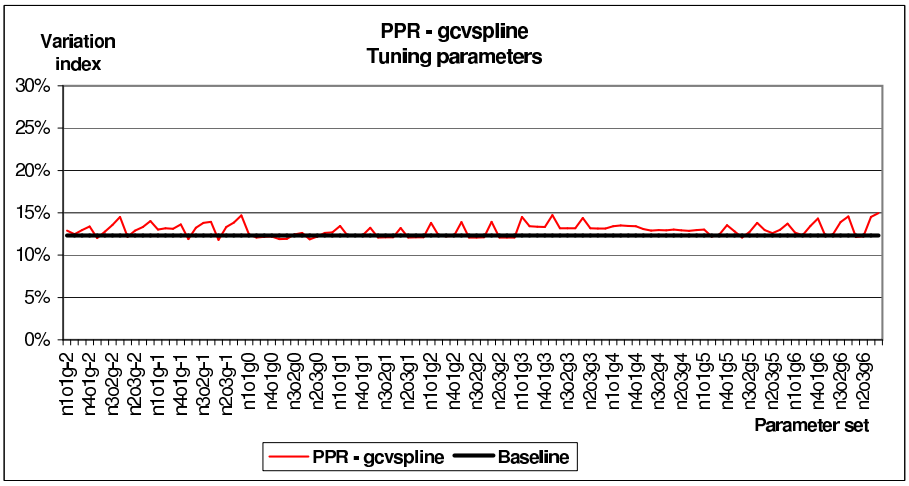

Figure 5.13: The *variation index* for PPR - gcvspline using different parameter sets.

5.7.1 Example selection

As stated in Sect. 4.3.2, the main idea of example selection is to increase accuracy by selecting from the training set just a subset of the examples for the training task.

Two approaches were tested:

- Equivalent days (ed): uses the examples from identical past days according to the groups defined in Sect. 5.4. If there is not a minimum of examples from equivalent days, according to the input parameter *min members*, all the examples from the training set are used. The value of *min members* was always set to 10 in all the experiments using the equivalent days approach.
- Leaf node (ln): use the examples from the same leaf node of a CART (Sect. 4.4.4) according to the pseudo-code presented in Fig. 5.14.

Require: *trainingSet*, the training set
Require: *testExample*, the test example
1: *tree* := *rpart(trainingSet)*
2: *leafNode* := *findleafnode(tree, testExample)*
3: *members* := *getmembers(tree, leafNode)*
4: **return** *members*

Figure 5.14: The leaf node approach: using CART to select similar data

The 'naive' idea behind these experiments can be stated in this way: as happens with autoregressive methods [Makridakis *et al.*, 1998] that use data from equivalent past periods to predict the unknown future (using typically a weighted average of the past equivalent data), it is expected that by using only similar past data for training, the methods can predict better because there

will be less 'noise'. Both approaches use this principle: to select only the most promising past data. In the first case, the approach could be easily improved by using just past trips from nearby departure times. The advantage of the leaf node approach is that it is completely independent of the existing expertise on the problem and, consequently, it can easily be adapted to other problems, as discussed later in this section.

Results for SVM linear, RF and PPR supsmu are presented in Figs. 5.15, 5.16 and 5.17. Results for the remaining algorithms are in Appendix A. In all these figures, the series 'All' presents the results using all the examples from the training set, i.e., without example selection. These series are the same as the ones presented in Sect. 5.6 under the name of the respective algorithm.

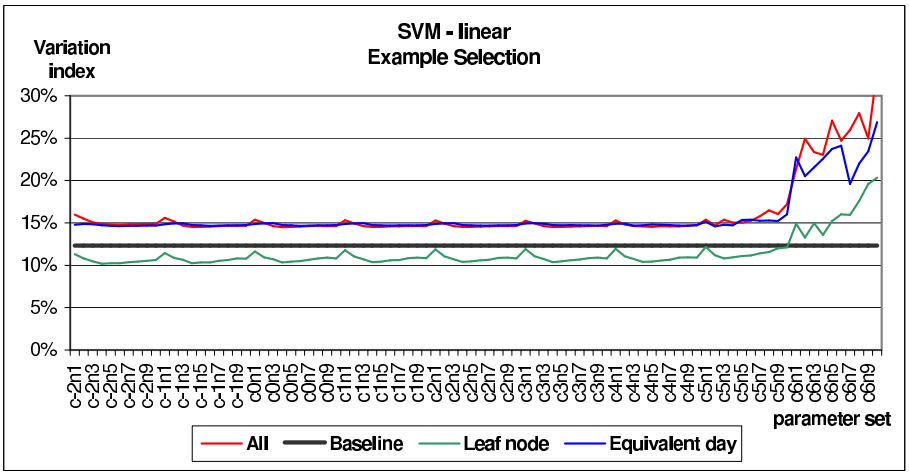

Figure 5.15: The *variation index* for SVM - linear using different methods for example selection.

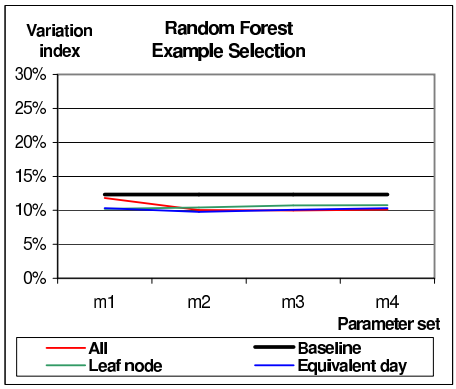

Figure 5.16: The *variation index* for RF using different methods for example selection.

The two approaches behave differently for each of the three methods. Results for RF are not surprising. In fact, the leaf node approach uses a filter identical

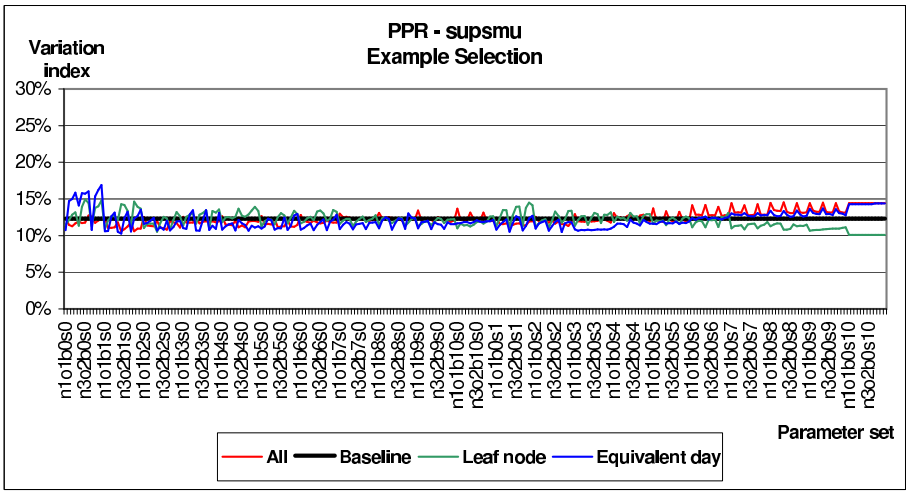

Figure 5.17: The *variation index* for PPR - supsmu using different methods for example selection.

to the one embedded in CART, and the equivalent day approach applies a kind of filter that can be, potentially, applied in CART when it is statistically relevant. In fact, the sequence of splits, implemented by CART, splits the data according to a set of rules in order to minimize, with some constraints (such as the minimum number of examples per leaf node), the sum of the variances of the resulting subsets. The equivalent day approach uses a pre-specified set of such rules as a previous step to the use of the CART approach.

For PPR, both approaches perform better than the all approach, even if for some parameter sets they perform worse. Anyway, it is clear that both approaches for example selection can ameliorate the accuracy.

For SVM, the leaf node approach is the best one. This result is particularly impressive. As previously mentioned, the leaf node approach is not problem dependent and, consequently, there is an open question: is the leaf node approach promising in other domains? The answer to this question is given in [Moreira *et al.*, 2006a]. The leaf node approach is tested in eleven regression data sets [Torgo, unknown] just for SVM - linear. The regression data sets are not time varying and, consequently, the experimental setup is necessarily different. We use 10-fold cross validation. The leaf node approach increases accuracy in seven data sets, draws in two and loses in the remaining two data sets. These results were obtained with 5% statistical significance. This approach has obvious connections to model trees (Sect. 4.4.4). In fact, we apply SVM - linear in the leaf nodes. This method can be improved by the use of functional trees [Gama, 2004] in order to reduce the computational complexity. The research community working on SVM has a known problem called working set selection. The goal is to select a subset of examples from the training set in order to reduce the computational complexity of the resolution of a quadratic optimization problem [Fan *et al.*, 2005]. This problem is different from the example selection task, as previously discussed, because the goal is to solve that particular quadratic problem faster but not a different problem. The leaf node approach changes the

Table 5.6: The best *Variation index* for each example selection method × algorithm.

algorithm	all	ln	ed
Baseline	12.32%		
Expert-based	9.23%		
SVM - linear	14.50%	**10.18%**	14.59%
SVM - radial	13.03%	**10.68%**	13.95%
SVM - sigmoid	14.15%	**10.20%**	14.28%
RF	9.92%	10.29%	**9.80%**
PPR - supsmu	10.60%	**10.09%**	10.27%
PPR - spline	11.15%	**10.30%**	10.40%
PPR - gcvspline	11.80%	10.39%	**10.00%**

initial problem.

In Table 5.6 the best results for each algorithm and for each different method to select the training set are presented. It is interesting to observe that after the introduction of a method to select examples, the best results for each different algorithm are much closer to each other, and are apparently better than the baseline method (these results are not statistically validated).

5.7.2 Domain values selection

As mentioned in Sect. 4.3.3, domain values selection can include: (1) the choice of the data type for each variable; (2) the discretization of continuous variables; or (3) the choice of appropriate values for a symbolic variable. Any of these tasks could be tested for this particular data set. Analyzing Table 5.7, we have identified the following possible experiments:

- Some variables, namely, day of the year, week day and week of the year, can be defined as numeric or as symbolic;
- Some continuous variables can be discretized by the use of the integer part when expressed in different scales, namely:
  - departure time: seconds or minutes;
  - travel time: seconds or minutes.
- Other variables can cluster their values. Then, the cluster identifier is used instead of the original value [Benjamini and Igbaria, 1991]. This is particularly relevant for variables with higher cardinality. Possible variables to be clustered are: driver, service, day of the year and week of the year.

Of all these tests, we have only done the one on the data type for the variable week day. Results for SVM using different kernels present the same behavior. Results for the linear kernel are shown in Fig. 5.18. For the remaining kernels, see Appendix A. The use of the numeric data type for the variable week day when using SVM is not promising.

For RF the results are similar (Fig. 5.19) for any value of *mtry*.

Table 5.7: Variables: data type and cardinality.

Variable	**Data type**	**Cardinality †**
Departure time	Numeric (in seconds)	6466
Week day	Symbolic	7
Day of the year	Numeric	237 ‡
Week of the year	Numeric	36
Day type	{holiday, bridge day, tolerance day}	4
School break	{break, normal}	2
Sundays unpd §	Numeric	5
Bus type	Symbolic	5
Driver	Symbolic	206
Service	Symbolic	79
Entrance flow	{we3, we4, holidays, normal}	4
Exit flow	{we3, we4, holidays, normal}	4
Wind speed	Numeric (in m/s)	13
Temperature	Numeric (in ° Celsius)	203
Precipitation	Numeric (in mm, $1mm = 1lt/m^2$)	55
Travel time	Numeric (in seconds)	2410

† The cardinality refers to the period from January 1st to August 30th of 2004.
‡ For the said period there are 244 days, but due to some errors in the backup files of the SAEI from where the data was collected, some days are missing.
§ Unpd means until next pay day.

For PPR (Fig. 5.20 and Appendix A) the behavior between the two approaches is clearly more erratic than for SVM and RF. For this reason, the same tests were done using the leaf node and the equivalent day approaches for example selection. Results for the super smoother are presented in Fig. 5.21. Results for the other smoothers are in Appendix A.

In all these figures on domain values selection, the series 'Symbolic' are the same as the ones presented in Sect. 5.6 under the name of the respective algorithm. The reason is that in all the experiments on parameter tuning, we used the symbolic data type for the variable week day.

After example selection and the choice of the data type for the week day variable, the best results for PPR improved for all the smoothers (Table 5.8).

5.7.3 Feature selection

Some authors classify the features as relevant or irrelevant according to how the inclusion of these features in the training set improves the prediction task (considering just supervised learning) [John *et al.*, 1994; Molina *et al.*, 2002]. Even knowing that the best feature subset in order to optimize accuracy depends on the induction algorithm to be used [Kohavi and John, 1997], in this section the goal is just to remove the irrelevant features. The goal is not to obtain the best feature subset for each one of the induction algorithms being tested. Using the definition given by Molina et al. for irrelevant features, they are the ones "not having any influence on the output, and whose values are generated at random for each example" [Molina *et al.*, 2002]. Molina et al. compare some of the most well known algorithms for feature subset selection in order

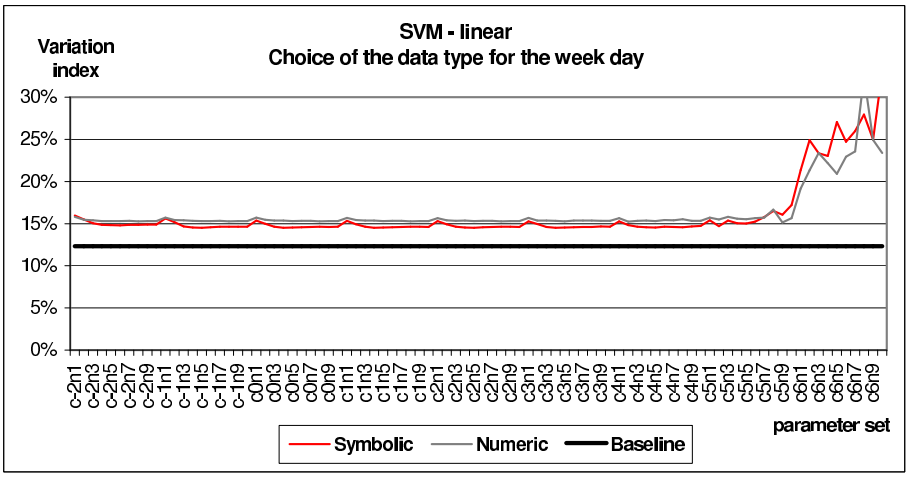

Figure 5.18: The *variation index* for SVM - linear using different data types for the variable week day.

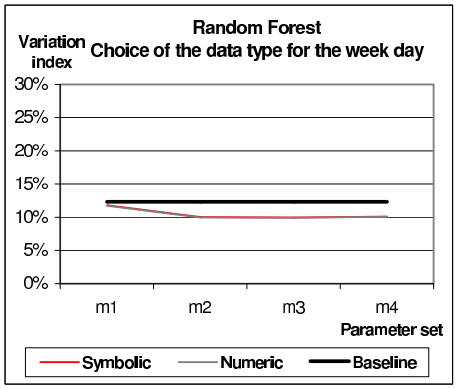

Figure 5.19: The *variation index* for RF using different data types for the variable week day.

to evaluate some particularities, one of which is irrelevance. The method with best results for detecting irrelevance is the Relief algorithm. According to Dietterich, "[ReliefF] is one of the most successful preprocessing algorithms to date" [Dietterich, 1997]. One decade has passed but the Relief family of algorithms continue to be frequently used for data mining applications [Cukjati *et al.*, 2001; Huang *et al.*, 2004; Liu *et al.*, 2004; Jin *et al.*, 2005; 2007].

The Relief family of algorithms [Kira and Rendell, 1992] weighs the features according to "how well their values distinguish between instances that are near to each other" [Robnik-Šikonja and Kononenko, 2003]. Originally the Relief algorithm only addressed the classification problem. The regression version, called RReliefF, was introduced later.

Our implementation of RReliefF follows Ref. [Robnik-Šikonja and Kononenko, 2003] (Fig. 5.22). The variable V in the algorithm is the number of input variables of the data set. The parameter t is the number of iterations. Its choice is

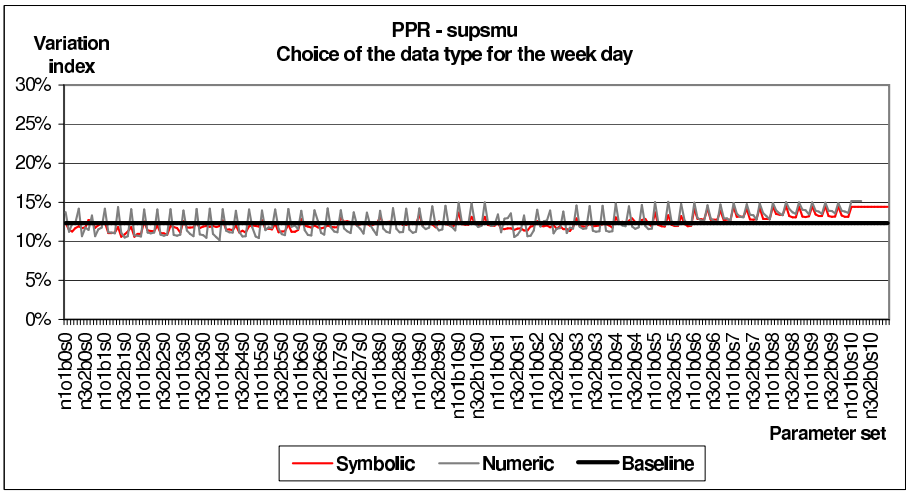

Figure 5.20: The *variation index* for PPR - supsmu using different data types for the variable week day.

problem dependent. The larger the t value is, the more stable the weights estimations are. Additionally, the computational cost also increases. The authors state that the stability of the results is obtained, typically, for values between 20 and 50 iterations. We use $t = 50$. The value used for k (the number of nearest examples) was 10, as suggested by the authors. From the several distance functions proposed by the authors, we use one that quadratically increases the cost of the distance:

$$d(i,j) = \frac{d1(i,j)}{\sum_{l=1}^{k}(d1(i,l))}, \tag{5.6}$$

$$d1(i,j) = \frac{1}{(\sum_{l=1}^{V} diff(A_l, R_i, I_j))^2}, \text{ and} \tag{5.7}$$

$$diff(A, I_1, I_2) = \begin{cases} 0 & : \quad d \leq t_{eq} \\ 1 & : \quad d > t_{diff} \\ \frac{d - t_{eq}}{t_{diff} - t_{eq}} & : \quad t_{eq} \leq t_{diff} \end{cases}, \tag{5.8}$$

where the values of t_{eq} and t_{diff} are, respectively, 5% and 10% of the length of the input variable's value interval, as suggested by the authors; and the value d represents the absolute difference of the input variable A for the two examples, I_1 and I_2.

RReliefF is a filter algorithm (Sect. 4.3.1), i.e., the feature selection is independent of the induction algorithm, implying a step prior to the use of the learner.

The first experiment on feature selection uses RReliefF in order to select, from the complete set of features identified in Sect. 5.1, a subset of them. The selected subset will be the complete working feature subset for future experiments. The other features will be discarded. A feature is selected if its RReliefF weight is larger than 0.01, as suggested in [Hall, 2000]. However, remembering

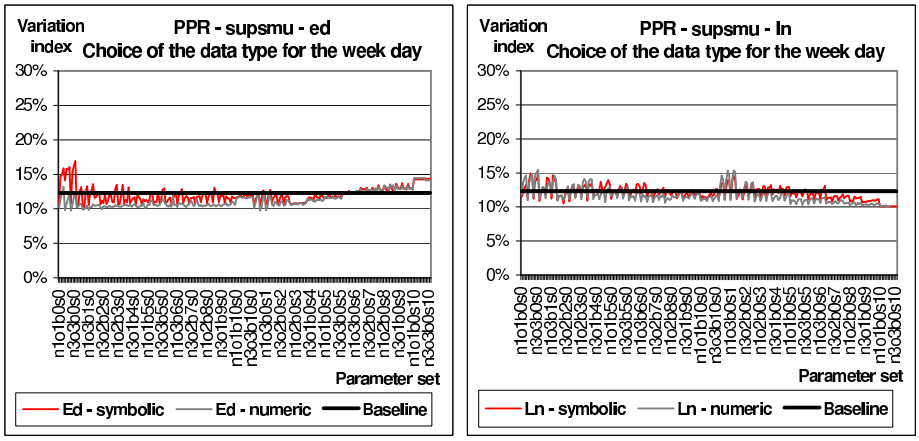

Figure 5.21: The *variation index* for PPR - supsmu (with example selection) using different data types for the variable week day.

Table 5.8: The *variation index* using example selection (ES) and different data types for the variable week day (CDT).

algorithm	**all**	**CDT**	**ES**	**ES & CDT**
Baseline	12.32%			
Expert-based	9.23%			
SVM - linear	14.50%	15.14%	**10.18%**	
SVM - radial	13.03%	13.37%	**10.68%**	
SVM - sigmoid	14.15%	14.43%	**10.20%**	
RF	9.92%	9.88%	**9.80%**	
PPR - supsmu	10.60%	10.15%	10.09%	**9.73%**
PPR - spline	11.15%	10.24%	10.30%	**9.79%**
PPR - gcvspline	11.80%	10.21%	10.00%	**9.55%**

The best result using ES & CDT is, for the three PPR smoothers, obtained under the equivalent days approach.

the experimental setup (Sect. 5.3), for 60 days of data, for instance, there are $60 - 30$ (the number of days in the time stamp) $+ 1$ different training sets. Results (Table 5.9) present five statistics: the percentage of days where the RReliefF weight is larger than 0.01 and the minimum, maximum, mean and standard deviation of the RReliefF weight values.

Some comments can be drawn:

- The low values for day type, entrance flow and exit flow can be explained by the use of 30 days in the training set and the previous knowledge that these values are rarely different to the standard ones.
- The low value for Sundays until next pay day can be due to the use of just one seasonal-cycle of this event. This variable could be useful if using a larger training set.
- Of the two variables used to capture the seasonality of the year: day of

Require: $\{R_i : i = 1, 2, ..., M\}$, the M input vectors from the training set
Require: τ, the correspondent M target values
Require: t, the number of iterations
Require: k, the number of nearest examples
set all $N_{dC}, N_{dA}[A], N_{dC\&dA}[A], W[A]$ to 0, where $A = 1, 2, \cdots, a$
for $i := 1$ to t **do**
  randomly select example R_i
  select k examples I_j nearest to R_i
  for $j := 1$ to k **do**
    $N_{dC} := N_{dC} + diff(\tau(.), R_i, I_j) \times d(i, j)$
    for $A := 1$ to V **do**
      $N_{dA}[A] := N_{dA}[A] + diff(A, R_i, I_j) \times d(i, j)$
      $N_{dC\&dA}[A] := N_{dC\&dA}[A] + diff(\tau(.), R_i, I_j) \times diff(A, R_i, I_j) \times d(i, j)$
    end for
  end for
end for
for $A = 1$ to V **do**
  $W[A] := N_{dC\&dA}[A]/N_{dC} - (N_{dA}[A] - N_{dC\&dA}[A]) \times (t - N_{dC})$
end for
return W

Figure 5.22: The RReliefF algorithm

the year and week of the year, the first one is the most relevant. Due to the dependence between these two variables, the respective weights are expected to be lower than they should be [Robnik-Šikonja and Kononenko, 2003].

- A natural doubt is if a feature with a low percentage of days with a weight larger than 0.01 has, in those days, a weight much larger than 0.01. 'Day type' is one of those cases (even if not too evident), i.e., when a test day is of a day type not 'normal', the most common is that in the training set there are no examples from the same day type. However, if there are, the feature day type can be relevant, specially when the test day is a working day.

For the above reasons, the variables week of the year, school break, Sundays until next pay day, entrance flow and exit flow are classified as irrelevant.

The following step was to select the feature subset that maximizes accuracy. The used approach takes advantage of the embedded feature selection used in random forests. Taking advantage of RF capacity to ignore irrelevant features, the approach we use runs RF using pre-specified subsets of input variables. Using this approach it is expected that the best result is obtained using all the variables. It is also expected that removing irrelevant variables, performance keeps stable. The tested subsets are:

- RedSet: the reduced set, i.e., the set of four variables used along this paper;
- RedSet+Meteo: the reduced set with the meteorologic variables;

Table 5.9: Statistics on RReliefF weights using 207 training windows.

variable	% of days where RW † > 0.01	min	max	mean	sd
Departure time	96.62%	0.08%	30.36%	11.56%	5.90%
Week day	58.45%	0.01%	5.78%	1.57%	1.26%
Day of the year	14.98%	0.00%	2.87%	0.54%	0.48%
Week of the year	0.97%	0.00%	1.44%	0.11%	0.21%
Day type	1.45%	0.00%	2.48%	0.09%	0.25%
School break	0.00%	0.00%	0.35%	0.01%	0.04%
Sundays unpd	0.97%	0.00%	1.30%	0.15%	0.23%
Bus type	64.25%	0.01%	6.05%	1.80%	1.42%
Driver	88.89%	0.01%	17.74%	5.14%	3.84%
Service	89.86%	0.04%	15.60%	5.04%	3.59%
Entrance flow	0.00%	0.00%	0.78%	0.03%	0.12%
Exit flow	0.00%	0.00%	0.82%	0.02%	0.09%
Wind speed	66.18%	0.01%	10.66%	2.45%	2.18%
Temperature	63.29%	0.01%	7.64%	1.80%	1.57%
Precipitation	16.91%	0.00%	5.92%	0.60%	0.98%

† RW means RReliefF weight.

- AllRRF-Meteo: all variables excluding the irrelevant and the meteorological ones;
- AllRRF: all variables excluding the irrelevant ones;
- All15: all the 15 variables.

For each of these subsets, all possible values of the *mtry* parameter are tested.

The best subset is, as expected, All15. However, AllRRF-Meteo uses just 7 variables and obtains a nearby result. Since the meteorological variables are the only ones that are obtained from outside the company, it is relevant to get insights into how valuable these variables are for prediction. Comparing results for AllRRF and AllRRF-Meteo, the difference among the best results for each of these two subsets is around 0.3%. It is important to note that the meteorological variables used as input variables have the actual values, instead of the predictions. This happens for all the variables used in the experiments. However, if inspecting the promising subset RedSet+Meteo, three variables (weekday, day type and day of the year) are certain, departure time is uncertain but it is the one that we hope to keep unchanged when predicting travel time three days ahead. The meteorological variables are the ones whose predictions three days ahead are expected to be more erratic. If we use predicted values for the input variables instead of the actual ones, the difference among the results for AllRRF and AllRRF-Meteo is expected to be lower than 0.3%. If we use meteorological variables, RedSet+Meteo seems to be a good departure subset for wrapper feature selection. If we do not use meteorological variables, the RedSet is a good start. Anyway, we did not study wrapper approaches for feature selection in this thesis. The use of RF for feature selection can be further explored using

Table 5.10: The *variation index* for RF using different subsets (the best value per subset is in bold face).

mtry	All15	AllRRF	AllRRF-Meteo	RedSet+Meteo	RedSet
1	14.03%	12.93%	12.31%	12.68%	11.85%
2	12.34%	10.96%	10.21%	10.67%	10.06%
3	11.24%	10.11%	**9.93%**	9.93%	**9.96%**
4	10.51%	9.74%	9.99%	9.66%	10.11%
5	10.11%	9.64%	10.09%	**9.62%**	
6	9.84%	**9.64%**	10.11%	9.70%	
7	9.69%	9.73%	10.11%	9.78%	
8	9.63%	9.79%			
9	**9.59%**	9.78%			
10	9.63%	9.79%			
11	9.64%				
12	9.65%				
13	9.66%				
14	9.65%				
15	9.64%				

the interpretation capabilities of the algorithm. Some work already exists in this area [Rudnicki *et al.*, 2006]. It is surely a subject to explore in the future.

5.8 Final comments and lessons learned

A possible doubt for someone who reads this chapter is whether it was necessary to test all the parameter sets at each experiment. Wouldn't it be enough to pick the best one for each algorithm and pursue the following experiments with just that parameter set? We try to answer this question by presenting a table similar to table 5.8 but using the best parameter set for the initial configuration as described in Sect. 5.6, in all the experiments (CDT, ES and ES & CDT). This is done for each algorithm (table 5.11). To better understand these results, they must be compared with table 5.8, where each value is obtained by testing the same parameter sets in all the experiments. The presented result is the one with the lower *variation index*. The results are a little bit erratic. For example, results for CDT are different for different smoothers. Also for SVM-radial, the used parameter set gives a worse result when using example selection. If we used the results of the last table, it would not be possible to understand the influence of the parameters at each new experiment. This is obvious, for example, for SVM-radial when comparing the result for ES in table 5.11 with the figure for SVM-radial presented in Sect. A.1.

All the results are presented without any kind of statistical comparison between the different methods. The reason is simple: if a method presents a better result than another one, even if statistically the difference is not meaningful, it will be the one to be chosen. The statistical validation is done at the end of the machine learning process in Chap. 8.

During the experiments carried out along this chapter, we asked ourselves several times whether the error is homogeneous along the input space or whether it is different for different regions of the input space. To answer this question,

Table 5.11: *Variation index* for the best parameter set for 'all', using example selection (ES) and different data types for the variable week day (CDT).

Algorithm	**Cod**	**all**	**CDT**	**ES**	**ES & CDT**
Baseline		12.32%			
Expert-based	m2g2i9	9.23%			
SVM - linear	c0n4	14.50%	15.36%	**10.34%**	
SVM - radial	c4n3g1	**13.03%**	13.48%	13.95%	
SVM - sigmoid	c5n3g2f1	14.15%	14.84%	**11.23%**	
RF	m3	**9.92%**	**9.92%**	10.09%	
PPR - supsmu	n2o3b1s0	10.60%	10.77%	12.31%	**10.25%**
PPR - spline	n1o3d6	11.15%	13.28%	10.57%	**9.87%**
PPR - gcvspline	n1o3g-1	11.80%	13.23%	10.57%	**9.74%**

Each algorithm uses the best parameter set for 'all' (identified by cod as explained in Sect. 5.6.2).
The algorithms that obtain the best result using example selection (ES and ES & CDT) use the leaf node approach (SVM-linear and SVM-sigmoid) and the equivalent day approach (PPR).

we trained a regression tree using CART [Breiman *et al.*, 1984] and calculated the *variation index* for the examples from each leaf node of the tree (table 5.12). The *variation index* was calculated for the best parameter set from each of the algorithms when using the 'all' approach, i.e., using the initial configuration without focusing tasks (these parameter sets are the ones identified in column 'Cod' in table 5.11). The global *variation index* uses a weighted mean squared error (weighted by the size of each leaf node). Comparing this result to the 'all' column of table 5.11, there is an improvement with respect to the one obtained when we use just one algorithm. However, it is necessary to be able to choose the most appropriate algorithm to be used in each leaf node. The idea of using an ensemble of models, instead of just one in order to reduce the error, appeared as a natural step after the analysis of table 5.12. The idea was still to reduce the generalization error as much as possible. As explained in Sect. 2.3, the business evaluation of the usefulness of the predictions three days ahead would be a second step that is not studied in this thesis. We only study the prediction problem.

The obtained results justify a certain flavor of unfinished work in this chapter, i.e., some tasks, namely the selection of the domain values (Sect. 5.7.2), the wrapper approaches for feature selection (Sect. 5.7.3) and the use of concept drift or active learning approaches in order to select from larger training sets (instead of 30 days time stamp, as discussed in Sect. 5.3), the most interesting examples, were not explored as they could have been. The study of ensemble approaches to solve the travel time prediction problem was the direction we took, i.e., how the use of an ensemble of models, instead of just one, could reduce the error. This is the subject of the following two chapters. Chap. 6 surveys ensemble approaches for regression and Chap. 7 describes different experiments on the use of ensemble approaches to solve TTP three days ahead.

Table 5.12: *Variation index* obtained by the algorithms on the leaf nodes of the regression tree. Column 'n' represents the number of examples in each node.

RF	**PPR supsmu**	**PPR spline**	**PPR gcvspline**	**SVM linear**	**SVM radial**	**SVM sigmoid**	**n**
21.84%	7.01%	**5.83%**	7.49%	7.46%	6.97%	6.85%	11
9.52%	9.46%	9.47%	9.62%	9.39%	9.49%	**9.13%**	649
16.76%	9.46%	6.93%	6.76%	8.52%	7.43%	**6.47%**	15
10.30%	**7.09%**	9.92%	11.01%	8.93%	8.69%	8.48%	28
7.44%	9.64%	8.03%	7.71%	7.80%	8.54%	**7.08%**	217
6.15%	8.09%	10.18%	10.66%	25.73%	13.96%	22.39%	104
11.10%	**8.24%**	11.98%	12.10%	13.01%	11.17%	12.48%	45
13.45%	**11.02%**	12.88%	14.69%	16.54%	15.96%	15.45%	265
6.66%	12.73%	7.48%	6.81%	7.66%	7.83%	7.64%	335
5.69%	7.28%	11.19%	12.67%	21.69%	10.89%	18.94%	82
7.10%	9.05%	**6.47%**	8.05%	10.59%	8.48%	8.15%	37
				global variation index:		**8.38%**	1 788

Chapter 6

An ensemble regression survey

Ensemble learning typically refers to methods that generate several models which are combined to make a prediction, either in classification or regression problems. This approach has been the object of a significant amount of research in recent years and good results have been reported (e.g., [Liu *et al.*, 2000; Breiman, 2001a]). The advantage of ensembles with respect to single models has been reported in terms of increased robustness and accuracy [García-Pedrajas *et al.*, 2005; Tan *et al.*, 2008]. Taking these advantages of ensemble methods into consideration, the study of ensemble approaches was a natural step forward in the sequence of the experiments described in Chap. 5.

Most work on ensemble learning focuses on classification problems. However, techniques that are successful for classification are often not directly applicable to regression. Therefore, although both are related, ensemble learning approaches have been developed somewhat independently. Consequently, existing surveys on ensemble methods for classification [Kuncheva, 2004; Ranawana and Palade, 2006] are not suitable for providing an overview of existing approaches for regression.

This chapter surveys existing approaches to ensemble learning for regression. Its relevance is strengthened by the fact that ensemble learning is an object of research in different communities, including pattern recognition, machine learning, statistics and neural networks. These communities have different conferences and journals and often use different terminology and notation, which makes it quite hard for a researcher to be aware of all contributions that are relevant to his/her own work. Therefore, besides attempting to provide a thorough account of the work in the area, we also organize these approaches independently of the research area they were originally proposed in.

In the next section, we provide a general discussion of the process of ensemble learning. This discussion will lay out the basis according to which the remaining sections of the chapter will be presented: ensemble generation (Sect. 6.2), ensemble pruning (Sect. 6.3) and ensemble integration (Sect. 6.4). Sect. 6.5 concludes the chapter with a summary.

6.1 Ensemble Learning for Regression

In this section we provide a more accurate definition of ensemble learning and provide terminology. Additionally, we present a general description of the process of ensemble learning and describe a taxonomy of different approaches, both of which define the structure of the rest of the chapter. Finally, we analyze the error decomposition of ensemble learning methods for regression.

6.1.1 Definition

First of all we need to define clearly what ensemble learning is, and to define a taxonomy of methods. As far as we know, there is no widely accepted definition of ensemble learning. Some of the existing definitions are partial in the sense that they focus just on the classification problem or on part of the ensemble learning process [Dietterich, 1997]. For these reasons we propose the following definition:

> Ensemble learning is a process that uses a set of models, each of them obtained by applying a learning process to a given problem. This set of models (ensemble) is integrated in some way to obtain the final prediction.

This definition has important characteristics. In the first place, it covers ensembles in supervised learning (both classification and regression problems). However, it does not cover, for example, the emerging research area of ensembles of clusters [Strehl and Ghosh, 2003] because clustering is not a predictive method.

Additionally, it clearly separates ensemble and divide-and-conquer approaches. This last family of approaches splits the input space into several subregions and separately trains each model in each one of the sub-regions. With this approach the initial problem is converted into the resolution of several simpler subproblems.

Finally, it does not separate the combination and selection approaches as usually happens. According to this definition, selection is a special case of combination where the weights are all zero except for one (to be discussed in Sect. 6.4).

More formally, an ensemble $\mathcal{F}$ is composed of a set of predictors of a function f denoted as $\hat{f}_i$.

$$\mathcal{F} = \{\hat{f}_i, i = 1, ..., k\}. \tag{6.1}$$

The resulting ensemble predictor is denoted as $\hat{f}_f$.

6.1.2 The Ensemble Learning Process

The ensemble process can be divided into three steps [Roli *et al.*, 2001] (Fig. 6.1), usually referred to as the overproduce-and-choose approach. The first step is ensemble generation, which consists of generating a set of models. It often happens that, during the first step, a number of redundant models are generated. In the ensemble pruning step, the ensemble is pruned by eliminating some of the models generated earlier. Finally, in the ensemble integration step,

a strategy to combine the base models is defined. This strategy is then used to obtain the prediction of the ensemble for new cases, based on the predictions of the base models.

Figure 6.1: Ensemble learning model.

Our characterization of the ensemble learning process is slightly more detailed than the one presented by Rooney et al. [Rooney *et al.*, 2004]. For those authors, ensemble learning consists of the solution of two problems: (1) how to generate the ensemble of models (ensemble generation); and (2) how to integrate the predictions of the models from the ensemble in order to obtain the final ensemble prediction (ensemble integration). This last approach (without the pruning step) is named direct, and can be seen as a particular case of the model presented in Fig. 6.1, named overproduce-and-choose.

Ensemble pruning has been reported, at least in some cases, to reduce the size of the ensembles obtained without degrading the accuracy. Pruning has also been added to direct methods successfully increasing the accuracy [Zhou *et al.*, 2002; Martínez-Muñoz and Suárez, 2006]. A subject to be discussed further in Sect. 6.3.

6.1.3 Taxonomy and Terminology

Concerning the categorization of the different approaches to ensemble learning, we will follow mainly the taxonomy presented by the same authors [Rooney *et al.*, 2004]. They divide ensemble generation approaches into homogeneous, if all the models were generated using the same induction algorithm, and heterogeneous, otherwise.

Ensemble integration methods are classified by some authors [Rooney *et al.*, 2004; Kuncheva, 2002] as combination (also called fusion) or as selection. The former approach combines the predictions of the models from the ensemble in order to obtain the final ensemble prediction. The latter one selects from the ensemble the most promising model(s) and the prediction of the ensemble is based on the selected model(s) only. Here, we use, instead, the classification of constant vs. non-constant weighting functions given by Merz [Merz, 1998]. In the first case, the predictions of the base models are always combined in the same way. In the second case, the way the predictions are combined can be different for different input values.

As mentioned earlier, research on ensemble learning is carried out in different communities. Therefore, different terms are sometimes used for the same concept. In Table 6.1 we list several groups of synonyms, extended from a previous list by Kuncheva [Kuncheva, 2004]. The first column contains the most frequently used terms in this thesis.

Table 6.1: Synonyms.

ensemble	committee, multiple models, multiple classifiers (regressors)
predictor	model, regressor (classifier), learner, hypothesis, expert
example	instance, case, data point, object
combination	fusion, competitive classifiers (regressors), ensemble approach, multiple topology
selection	cooperative classifiers (regressors), modular approach, hybrid topology

6.1.4 Understanding the generalization error of ensembles

To accomplish the task of ensemble generation, it is necessary to know the characteristics that the ensemble should have. Empirically, it is stated by several authors that a good ensemble is the one with accurate predictors and making errors in different parts of the domains of the input variables. For the regression problem it is possible to decompose the generalization error in different components, which can guide the process to optimize the ensemble generation. In this section, as well as in the remaining parts of this chapter, we assume a typical regression problem as defined in Sect. 4.2.2.

Here, the functions are represented, when appropriate, without the input variables, just for the sake of simplicity. For example, instead of $f(\mathbf{x})$ we use f. We closely follow Brown [Brown, 2004].

Understanding the ensemble generalization error enables us to know which characteristics the ensemble members should have in order to reduce the overall generalization error. The generalization error decomposition for regression is straightforward. What follows is about the decomposition of the *mse* (Eq. 4.3). Despite the fact that most works were presented in the context of neural network ensembles, the results presented in this section are not dependent on the induction algorithm used.

Geman et al. present the bias/variance decomposition for a single neural network [Geman *et al.*, 1992]:

$$E\{[\hat{f} - E(f)]^2\} = [E(\hat{f}) - E(f)]^2 + E\{[\hat{f} - E(\hat{f})]^2\}. \tag{6.2}$$

The first term on the right hand side is called the bias and represents the distance between the expected value of the estimator $\hat{f}$ and the unknown population average. The second term, the variance component, measures how the predictions vary with respect to the average prediction. This can be rewritten as:

$$mse(f) = bias(f)^2 + var(f). \tag{6.3}$$

Krogh & Vedelsby describe the ambiguity decomposition, for an ensemble of k neural networks [Krogh and Vedelsby, 1995]. Assuming that $\hat{f}_f(\mathbf{x}) = \sum_{i=1}^{k}[\alpha_i \times \hat{f}_i(\mathbf{x})]$ (see Sect. 6.4.1) where $\sum_{i=1}^{k}(\alpha_i) = 1$ and $\alpha_i \geq 0, i = 1, ..., k$, they show that the error for a single example is:

$$(\hat{f}_f - f)^2 = \sum_{i=1}^{k}[\alpha_i \times (\hat{f}_i - f)^2] - \sum_{i=1}^{k}[\alpha_i \times (\hat{f}_i - \hat{f}_f)^2]. \tag{6.4}$$

This expression shows explicitly that the ensemble generalization error is less than or equal to the generalization error of a randomly selected single predictor. This is true because the ambiguity component (the second term on the right) is always non negative. Another important result of this decomposition is that it is possible to reduce the ensemble generalization error by increasing the ambiguity without increasing the bias. The ambiguity term measures the disagreement among the base predictors on a given input $\mathbf{x}$ (omitted in the formulae just for the sake of simplicity, as previously mentioned). Two full proofs of the ambiguity decomposition [Krogh and Vedelsby, 1995] are presented in [Brown, 2004].

Later, Ueda & Nakano presented the bias/variance/covariance decomposition of the generalization error of ensemble estimators [Ueda and Nakano, 1996]. In this decomposition it is assumed that $\hat{f}_f(\mathbf{x}) = \frac{1}{k} \times \sum_{i=1}^{k} [\hat{f}_i(\mathbf{x})]$:

$$E[(\hat{f}_f - f)^2] = \overline{bias}^2 + \frac{1}{k} \times \overline{var} + (1 - \frac{1}{k}) \times \overline{covar}, \tag{6.5}$$

where

$$\overline{bias} = \frac{1}{k} \times \sum_{i=1}^{k} [E_i(f_i) - f], \tag{6.6}$$

$$\overline{var} = \frac{1}{k} \times \sum_{i=1}^{k} \{E_i\{[\hat{f}_i - E_i(\hat{f}_i)]^2\}\}, \tag{6.7}$$

$$\overline{covar} = \frac{1}{k \times (k-1)} \times \sum_{i=1}^{k} \sum_{j=1, j \neq i}^{k} E_{i,j}\{[\hat{f}_i - E_i(\hat{f}_i)][\hat{f}_j - E_j(\hat{f}_j)]\}. \tag{6.8}$$

The indexes i, j of the expectation mean that the expression is true for particular training sets, respectively, $\mathcal{L}_i$ and $\mathcal{L}_j$.

Brown provides a good discussion on the relation between ambiguity and covariance [Brown, 2004]. An important result obtained from the study of this relation is the confirmation that it is not possible to maximize the ensemble ambiguity without affecting the ensemble bias component as well, i.e., it is not possible to maximize the ambiguity component and minimize the bias component simultaneously.

The discussion of the present section is usually referred to in the context of ensemble diversity, i.e., the study on the degree of disagreement between the base predictors. Many of the above statements are related to the well known statistical problem of point estimation. This discussion is also related to the multicollinearity problem that will be discussed in Sect. 6.4.

6.2 Ensemble generation

The goal of ensemble generation is to generate a set of models, $\mathcal{F} = \{\hat{f}_i, i = 1, ..., k\}$. If the models are generated using the same induction algorithm, the ensemble is called homogeneous, otherwise it is called heterogeneous.

Homogeneous ensemble generation is the best covered area of ensemble learning in the literature. See, for example, the state of the art surveys from Dietterich [Dietterich, 1997; 2002], or Brown et al. [Brown *et al.*, 2005]. In this

section we followed mainly [Dietterich, 1997]. In homogeneous ensembles, the models are generated using the same algorithm. Thus, as explained in the following sections, diversity can be achieved by manipulating the data (Section 6.2.1) or by the model generation process (Section 6.2.2).

Heterogeneous ensembles are obtained when more than one learning algorithm is used. This approach is expected to obtain models with higher diversity [Webb and Zheng, 2004; Wichard *et al.*, 2003]. The problem is the lack of control over the diversity of the ensemble during the generation phase. In homogeneous ensembles, diversity can be systematically controlled during their generation, as will be discussed in the following sections. Conversely, when using several algorithms, it may not be so easy to control the differences between the generated models. This difficulty can be solved by the use of the overproduce-and-choose approach. Using this approach the diversity is guaranteed in the pruning phase [Caruana *et al.*, 2004]. Another approach, commonly followed, combines the two approaches, by using different induction algorithms mixed with the use of different parameter sets [Merz, 1996; Rooney *et al.*, 2004] (Sect. 6.2.2).

In the context of this thesis, the existence of an extensive pool of results using different induction algorithms (from the experiments done in Chap. 5) and the possibility to reduce the generalization error from the base predictors (one of them, random forest, is a state of the art homogeneous ensemble regression method, see Sect. 6.5) lead us to the study of heterogeneous approaches (a subject to discuss in Sect. 7.1).

6.2.1 Data manipulation

Data can be manipulated in three different ways: subsampling from the training set, manipulating the input features and manipulating the output targets.

Subsampling from the training set

These methods have in common that the models are obtained using different subsamples from the training set, where subsampling means resampling with replacement [Minaei-Bigdoli *et al.*, 2004]. This approach generally assumes that the algorithm is unstable, i.e., small changes in the training set imply important changes in the result. Decision trees, neural networks, rule learning algorithms and MARS are well known unstable algorithms [Breiman, 1996b; Dietterich, 1997]. However, some of the methods based on subsampling (e.g., bagging and boosting) have been successfully applied to algorithms usually regarded as stable, such as Support Vector Machines (SVM) [Kim *et al.*, 2003].

One of the most popular of such methods is bagging [Breiman, 1996a]. It uses randomly generated training sets to obtain an ensemble of predictors. If the original training set $\mathcal{L}$ has m examples, bagging (bootstrap aggregating) generates a model by sampling uniformly m examples with replacement (some examples appear several times while others do not appear at all). Both Breiman [Breiman, 1996a] and Domingos [Domingos, 1997] give insights on why bagging works.

Based on [Schapire, 1990], Freund & Schapire present the AdaBoost (ADAptive BOOSTing) algorithm, the most popular boosting algorithm [Freund and Schapire, 1996]. The main idea is that it is possible to convert a weak learning

algorithm into one that achieves arbitrarily high accuracy. A weak learning algorithm is one that performs slightly better than random prediction. This conversion is done by combining the estimations of several predictors. Like in bagging [Breiman, 1996a], the examples are randomly selected with replacement but, in AdaBoost, each example has a different probability of being selected. Initially, this probability is equal for all the examples but, in the following iterations, examples with more inaccurate predictions have a higher probability of being selected. In each new iteration there are more 'difficult examples' in the training set. Although boosting has been originally developed for classification, several algorithms have been proposed for regression but none has emerged as being the appropriate one [Granitto *et al.*, 2005].

Parmanto et al. describe the cross-validated committees technique for neural networks ensemble generation using v-fold cross validation [Parmanto *et al.*, 1996]. The main idea is to use the models obtained by the use of the v training sets as ensemble on the cross validation process.

Manipulating the input features

In this approach, different training sets are obtained by changing the representation of the examples. A new training set j is generated by replacing the original representation $\{(\mathbf{x_i}, f(\mathbf{x_i}))$ into a new one $\{(\mathbf{x'_i}, f(\mathbf{x_i}))$. There are two types of approaches. The first one is feature selection, i.e., $\mathbf{x'_i} \subset \mathbf{x_i}$. In the second approach, the representation is obtained by applying some transformation to the original attributes, i.e., $\mathbf{x'_i} = g(\mathbf{x_i})$.

A simple feature selection approach is the random subspace method, consisting of a random selection [Ho, 1998]. The models in the ensemble are independently constructed using a randomly selected feature subset. Originally, decision trees were used as base learners and the ensemble was called decision forests [Ho, 1998]. The final prediction is the combination of the predictions of all the trees in the forest.

Alternatively, iterative search methods can be used to select the different feature subsets. Opitz uses a genetic algorithm approach that continuously generates new subsets starting from a random feature selection [Opitz, 1999]. The author uses neural networks for the classification problem. He reports better results using this approach than using the popular bagging and AdaBoost methods. In [Zenobi and Cunningham, 2001] the search method is a wrapper-like hill-climbing strategy. The criteria used to select the feature subsets are the minimization of the individual error and the maximization of ambiguity (Sect. 6.1.4).

A feature selection approach can also be used to generate ensembles for algorithms that are stable with respect to the training set but unstable with respect to the set of features, namely the nearest neighbors induction algorithm. In [Domeniconi and Yan, 2004] the feature subset selection is done using adaptive sampling in order to reduce the risk of discarding discriminating information. The authors use the ADAMENN algorithm [Domeniconi *et al.*, 2002] to estimate feature relevance. Compared to random feature selection, this approach reduces diversity between base predictors but increases their accuracy.

A simple transformation approach is input smearing [Frank and Pfahringer, 2006]. It aims to increase the diversity of the ensemble by adding Gaussian noise to the inputs. The goal is to improve the results of bagging. Each input

value x is changed into a smeared value x' using:

$$x' = x + p * N(0, \hat{\sigma}_x) \tag{6.9}$$

where p is an input parameter of the input smearing algorithm and $\hat{\sigma}_x$ is the sample standard deviation of x, using the training set data. In this case, the examples are changed, but the training set keeps the same number of examples. In this work just the numeric input variables are smeared even if the nominal ones could also be smeared using a different strategy. Results compare favorably to bagging. A similar approach called BEN - Bootstrap Ensemble with Noise, was previously presented by Raviv & Intrator [Raviv and Intrator, 1996].

Manipulating the output targets

The manipulation of the output targets can also be used to generate different training sets. However, not much research follows this approach and most of it focuses on classification.

An exception is the work of Breiman, called output smearing [Breiman, 2000]. The basic idea is to add Gaussian noise to the target variable of the training set, in the same way as is done for input features in the input smearing method (Sect. 6.2.1). Using this approach it is possible to generate as many models as desired. Although it was originally proposed to use with CART trees as base models, it can be used with other base algorithms. The comparison between output smearing and bagging shows a consistent generalization error reduction, even if not outstanding.

An alternative approach consists of the following steps. First it generates a model using the original data. Second, it generates a model that estimates the error of the predictions of the first model and generates an ensemble that combines the prediction of the previous model with the correction of the current one. Finally, it iteratively generates models that predict the error of the current ensemble and then updates the ensemble with the new model. The training set used to generate the new model in each iteration is obtained by replacing the output targets with the errors of the current ensemble. This approach was proposed by Breiman, using bagging as the base algorithm and was called iterated bagging [Breiman, 2001b]. Iterated bagging reduces generalization error when compared with bagging, mainly due to the bias reduction during the iteration process.

6.2.2 Model generation manipulation

As an alternative to manipulating the training set, it is possible to change the model generation process. This can be done by using different parameter sets, by manipulating the induction algorithm or by manipulating the resulting model.

Manipulating the parameter sets

Each induction algorithm is sensitive to the values of the input parameters. The degree of sensitivity of the induction algorithm is different for different input parameters. To maximize the diversity of the models generated, one should focus on the parameters to which the algorithm is most sensitive.

Neural network ensemble approaches quite often use different initial weights to obtain different models. This is done because the resulting models vary significantly with different initial weights [Kolen and Pollack, 1990]. Several authors, like Rosen, for example, use randomly generated seeds (initial weights) to obtain different models [Rosen, 1996], while other authors mix this strategy with the use of different number of layers and hidden units [Perrone and Cooper, 1993; Hashem, 1993].

The k-nearest neighbors ensemble proposed in [Yankov *et al.*, 2006] has just two members. They differ on the number of nearest neighbors used. They are both sub-optimal. One of them because the number of nearest neighbors is too small, and the other because it is too large. The purpose is to increase diversity (see Sect. 6.4.2).

Manipulating the induction algorithm

Diversity can also be attained by changing the way induction is done. Therefore, the same learning algorithm may have different results on the same data. Two main categories of approaches for this can be identified: sequential and parallel. In sequential approaches, the induction of a model is influenced only by the previous ones. In parallel approaches it is possible to have more extensive collaboration: (1) each process takes into account the overall quality of the ensemble and (2) information about the models is exchanged between processes.

Rosen generates ensembles of neural networks by sequentially training networks, adding a decorrelation penalty to the error function, to increase diversity [Rosen, 1996]. Using this approach, the training of each network tries to minimize a function that has a covariance component, thus decreasing the generalization error of the ensemble, as stated in [Ueda and Nakano, 1996]. This was the first approach using the decomposition of the generalization error made by Ueda & Nakano [Ueda and Nakano, 1996] (Sect. 6.1.4) to guide the ensemble generation process. Another sequential method for generating ensembles of neural networks is called SECA (Stepwise Ensemble Construction Algorithm) [Granitto *et al.*, 2005]. It uses bagging to obtain the training set for each neural network. The neural networks are trained sequentially. The process stops when adding another neural network to the current ensemble increases the generalization error.

The Cooperative Neural Network Ensembles (CNNE) method [Islam *et al.*, 2003] also uses a sequential approach. In this work, the ensemble begins with two neural networks and then, iteratively, CNNE tries to minimize the ensemble error firstly by training the existing networks, then by adding a hidden node to an existing neural network, and finally by adding a new neural network. As in Rosen's approach, the error function includes a term representing the correlation between the models in the ensemble. Therefore, to maximize the diversity, all the models already generated are trained again at each iteration of the process. The authors test their method not only on classification data sets, but also on one regression data set, with promising results.

Tsang et al. [Tsang *et al.*, 2006] propose an adaptation of the CVM (Core Vector Machines) algorithm [Tsang *et al.*, 2005] that maximizes the diversity of the models in the ensemble by guaranteeing that they are orthogonal. This is achieved by adding constraints to the quadratic programming problem that is solved by the CVM algorithm. This approach can be related to AdaBoost

because higher weights are given to instances which are incorrectly classified in previous iterations.

Note that the sequential approaches mentioned above add a penalty term to the error function of the learning algorithm. This sort of added penalty has also been used in the parallel method Ensemble Learning via Negative Correlation (ELNC) to generate neural networks that are learned simultaneously so that the overall quality of the ensemble is taken into account [Liu and Yao, 1999].

Parallel approaches that exchange information during the process typically integrate the learning algorithm with an evolutionary framework. Opitz & Shavlik [Opitz and Shavlik, 1996] present the ADDEMUP (Accurate anD Diverse Ensemble-Maker giving United Predictions) method to generate ensembles of neural networks. In this approach, the fitness metric for each network weighs the accuracy of the network and the diversity of this network within the ensemble. The bias/variance decomposition presented by Krogh & Vedelsby [Krogh and Vedelsby, 1995] is used. Genetic operators of mutation and crossover are used to generate new models from previous ones. The new networks are trained emphasizing misclassified examples. The best networks are selected and the process is repeated until a stopping criterion is met. This approach can be used on other induction algorithms. A similar approach is the Evolutionary Ensembles with Negative Correlation Learning (EENCL) method [Liu *et al.*, 2000], which combines the ELNC method with an evolutionary programming framework. In this case, the only genetic operator used is mutation, which randomly changes the weights of an existing neural network. The EENCL has two advantages in common with other parallel approaches. First, the models are trained simultaneously, emphasizing specialization and cooperation among individuals. Second, the neural network ensemble generation is done according to the integration method used, i.e., the learning models and the ensemble integration are part of the same process, allowing possible interactions between them. Additionally, the ensemble size is obtained automatically in the EENCL method.

A parallel approach in which each learning process does not take into account the quality of the others, but in which there is exchange of information about the models is given by the cooperative coevolution of artificial neural network ensembles method [García-Pedrajas *et al.*, 2005]. It also uses an evolutionary approach to generate ensembles of neural networks. It combines a mutation operator that affects the weights of the networks, as in EENCL, with another which affects their structure, as in ADDEMUP. As in EENCL, the generation and integration of models are also part of the same process. The diversity of the models in the ensemble is encouraged in two ways: (1) by using a coevolution approach, in which sub-populations of models evolve independently; and (2) by the use of a multiobjective evaluation fitness measure, combining network and ensemble fitness. Multiobjective is a quite well known research area in the operational research community. The authors use a multiobjective algorithm based on the concept of Pareto optimality [Deb, 1999]. Other groups of objectives (measures) besides the cooperation ones are: objectives of performance, regularization, diversity and ensemble objectives. The authors do a study on the sensitivity of the algorithm to changes in the set of objectives. The results are interesting but they cannot be generalized to the regression problem, since the authors just studied the classification one. This approach can be used for regression, but with a different set of objectives.

Finally we mention two other parallel techniques. In the first one the learning algorithm generates the ensemble directly. Lin & Li formulate an infinite ensemble based on the SVM (Support Vector Machines) algorithm [Lin and Li, 2005]. The main idea is to create a kernel that embodies all the possible models in the hypothesis space. The SVM algorithm is then used to generate a linear combination of all those models, which is, in fact, an ensemble of an infinite set of models. They propose the stump kernel that represents the space of decision stumps.

Breiman's random forests method [Breiman, 2001a] uses an algorithm for induction of decision trees which is also modified to incorporate some randomness: the split used at each node takes into account a randomly selected feature subset. The subset considered in one node is independent of the subset considered in the previous one. This strategy based on the manipulation of the learning algorithm is combined with subsampling, since the ensemble is generated using the bagging approach (Sect. 6.2.1). The strength of the method is the combined use of boostrap sampling and random feature selection.

Manipulating the model

Given a learning process that produces one single model M, it can potentially be transformed into an ensemble approach by producing a set of models M_i from the original model M. Jorge & Azevedo have proposed a post-bagging approach for classification [Jorge and Azevedo, 2005] that takes a set of classification association rules (CAR's), produced by a single learning process, and obtains n models by repeatedly sampling the set of rules. Predictions are obtained by a large committee of classifiers constructed as described above. Experimental results on 12 data sets show a consistent, although slight, advantage over the singleton learning process. The same authors also propose an approach with some similarities to boosting [Azevedo and Jorge, 2007]. Here, the rules in the original model M are iteratively reassessed, filtered and reordered according to their performance on the training set. Again, experimental results show minor but consistent improvement over using the original model, and also show a reduction in the bias component of the error. Both approaches replicate the original model without relearning and obtain very homogeneous ensembles with a kind of jittering effect around the original model. Model manipulation has only been applied in the realm of classification association rules, a highly modular representation. Applying it to other kinds of models, such as decision trees or neural networks, does not seem trivial because the way these models are represented is not suitable for being post-bagged. It could, however, be easily tried with regression rules.

6.2.3 A discussion on ensemble generation

Two relevant issues arise from the discussion above. The first is how can the user decide which method to use for a given problem.? The second, which is more interesting from a researcher's point of view, is what are the promising lines for future work?

In general, existing results indicate that ensemble methods are competitive when compared to individual models. For instance, random forests are consistently among the best three models in the benchmark study by Meyer et al.

[Meyer *et al.*, 2003], which included many different algorithms.

However, there is little knowledge about the strengths and weaknesses of each ensemble method, given that the results reported in different papers are not comparable because of the use of different experimental setups [Islam *et al.*, 2003; García-Pedrajas *et al.*, 2005].

It is possible to distinguish the most interesting/promising methods for some of the most commonly used induction algorithms. For decision trees, bagging [Breiman, 1996a] because of its consistency and simplicity, and random forest [Breiman, 2001a] because of its accuracy, are the most appealing ensemble methods for regression.

For neural networks, methods based on negative correlation are particularly appealing, due to their theoretical foundations [Brown, 2004] and good empirical results. EENCL is certainly an influential and well studied method on neural network ensembles [Liu *et al.*, 2000]. Islam et al. [Islam *et al.*, 2003] and Garcia-Pedrajas et al. [García-Pedrajas *et al.*, 2005] also present interesting methods.

One important line of work is the adaptation of the methods described here to other algorithms, namely support vector regression and k-nearest neighbors. Although some attempts have been made, there is still much work to be done.

Additionally, we note that most research focuses on one specific approach building the ensemble (e.g., subsampling from the training set or manipulating the induction algorithm). Further investigation is necessary into the gains that can be achieved by combining several approaches.

In this thesis, due to the experiments on parameters' tuning (Sect. 5.6) and on data manipulation, namely example (Sect. 5.7.1) and domain value selection (Sect. 5.7.2), there are available models obtained through subsampling from the training set (even if deterministic since the methods we use to select examples do not use randomization) and through manipulation of the input features (through the use of different representations for the variable week day).

6.3 Ensemble pruning

Ensemble pruning consists of eliminating models from the ensemble (also named pool in this circumstance), with the aim of improving its predictive ability or reducing costs. In the overproduce and choose approach it is the chosen step. In the direct approach, ensemble pruning is also used to reduce computational costs and, if possible, to increase prediction accuracy [Zhou *et al.*, 2002; Bakker and Heskes, 2003]. Bakker & Heskes claim that clustering models (later described in Sect. 6.3.5) summarize the information on the ensembles, thus giving new insights into the data [Bakker and Heskes, 2003]. Ensemble pruning can also be used to avoid the multi-collinearity problem [Perrone and Cooper, 1993; Hashem, 1993] (to be discussed in Sect. 6.4).

The ensemble pruning process has many common aspects with feature selection (Sect. 4.3.1) namely the search algorithms that can be used. In this section, the ensemble pruning methods are classified and presented according to the used search algorithm: exponential, randomized and sequential; plus the ranked pruning and the clustering algorithms. It finishes with a discussion on ensemble pruning, where experiments comparing some of the algorithms described along this chapter are presented.

6.3.1 Exponential pruning algorithms

When we have to select a subset of models from a pool of K models, the searching space has $2^K - 1$ non-empty subsets. The search of the optimal subset is a NP-complete problem [Tamon and Xiang, 2000]. According to Martínez-Muñoz & Suárez it becomes intractable for values of $K > 30$ [Martínez-Muñoz and Suárez, 2006]. Perrone & Cooper suggest this approach for small values of K [Perrone and Cooper, 1993].

Aksela presents seven pruning algorithms for classification [Aksela, 2003]. One of them can also be used for regression. It calculates the correlation of the errors for each pair of predictors in the pool and then it selects the subset with minimal mean pairwise correlation. This method implies the calculation of the said metric for each possible subset.

6.3.2 Randomized pruning algorithms

Partridge & Yates describe the use of a genetic algorithm for ensemble pruning but with poor results [Partridge and Yates, 1996].

GASEN (Genetic Algorithm based Selective ENsemble), a work on neural network ensembles [Zhou *et al.*, 2002], starts with the assignment of a random weight to each one of the base models. Then it employs a genetic algorithm to evolve those weights in order to characterize the contribution of the corresponding model to the ensemble. Finally it selects the networks whose weights are bigger than a predefined threshold. Empirical results on ten regression problems show that GASEN outperforms bagging and boosting both in terms of bias and variance. Following this work, Zhou & Tang successfully applied GASEN to building ensembles of decision trees [Zhou and Tang, 2003].

Ruta & Gabrys use three randomized algorithms to search for the best subset of models [Ruta and Gabrys, 2001]: genetic algorithms [Davis (ed.), 1991], tabu search [Glover and Laguna, 1997] and population-based incremental learning [Baluja, 1994]. The main result of the experiments on three classification data sets, using a pool of $K = 15$, was that the three algorithms obtained most of the best selectors when compared against exhaustive search. These results may have been conditioned by the small size of the pool.

6.3.3 Sequential pruning algorithms

The sequential pruning algorithms iteratively change one solution by adding or removing models. Three types of search algorithms are used:

- Forward: if the search begins with an empty ensemble and adds models to the ensemble in each iteration;
- Backward: if the search begins with all the models in the ensemble and eliminates models from the ensemble in each iteration;
- Forward-backward: if the selection can have both forward and backward steps.

Forward selection

Forward selection starts with an empty ensemble and iteratively adds models with the aim of decreasing the expected prediction error.

Coelho & Von Zuben describe two forward selection algorithms called Cw/oE - constructive without exploration, and CwE - constructive with exploration [Coelho and Von Zuben, 2006]. However, to use a more conventional categorization (Sect. 4.3.1), the algorithms will be renamed Forward Sequential Selection with Ranking (FSSwR) and Forward Sequential Selection (FSS), respectively. The FSSwR ranks all the candidates with respect to its performance on a validation set. Then, it selects the candidate at the top until the performance of the ensemble decreases. In the FSS algorithm, each time a new candidate is added to the ensemble, all candidates are tested and the one that leads to the maximal improvement of the ensemble performance is selected. When no model in the pool improves the ensemble performance, the selection stops. This approach is also used in [Roli *et al.*, 2001]. These algorithms were firstly described for ensemble pruning by Perrone & Cooper [Perrone and Cooper, 1993].

Partridge & Yates present another forward selection algorithm similar to the FSS [Partridge and Yates, 1996]. The main difference is that the criterion for the inclusion of a new model is a diversity measure. The model with higher diversity than the ones already selected is also included in the ensemble. The ensemble size is an input parameter of the algorithm. Prodromidis et al. present a similar algorithm with a different diversity measure [Prodromidis *et al.*, 1999].

Another similar approach is presented in [Hernández-Lobato *et al.*, 2006]. At each iteration it tests all the models not yet selected, and selects the one that reduces most the ensemble generalization error on the training set. Experiments to reduce ensembles generated using bagging are promising even if overfitting could be expected since the minimization of the generalization error is done on the training set.

Backward selection

Backward selection starts with all the models in the ensemble and iteratively removes models with the aim of decreasing the expected prediction error.

Coelho & Von Zuben describe two backward selection algorithms called Pw/oE - pruning without exploration, and PwE - pruning with exploration [Coelho and Von Zuben, 2006]. Like in the forward selection methods, they will be renamed Backward Sequential Selection with Ranking (BSSwR) and Backward Sequential Selection (BSS), respectively. In the first one, the candidates are previously ranked according to their performance in a validation set (like in FSSwR). The worst is removed. If the ensemble performance improves, the selection process continues. Otherwise, it stops. BSS is related to FSS in the same way BSSwR is related to FSSwR, i.e., it works like FSS but using backward selection instead of forward selection.

Mixed forward-backward selection

In the forward and backward algorithms described by Coelho & Von Zuben, namely the FSSwR, FSS, BSSwR and BSS, the stopping criterion assumes that the evaluation function is monotonic [Coelho and Von Zuben, 2006]. However, in practice, this cannot be guaranteed. The use of mixed forward and backward

steps aims to avoid the situations where the fast improvement at the initial iterations does not allow solutions with slower initial improvements but with better final results to be explored.

Moreira et al. describe an algorithm that begins by randomly selecting a pre-defined number of k models [Moreira *et al.*, 2006b]. At each iteration, one forward step and one backward step are given. The forward step is equivalent to the process used by FSS, i.e., it selects the model from the pool that most improves the accuracy of the ensemble. At this step, the ensemble has $k + 1$ models. The second step selects the k models with higher ensemble accuracy, i.e, in practice, one of the $k + 1$ models is removed from the ensemble. The process stops when the same model is selected in both steps.

Margineantu & Dietterich present an algorithm called reduce-error pruning with back fitting [Margineantu and Dietterich, 1997]. This algorithm is similar to the FSS in the two first iterations. After the second iteration, i.e., when the third candidate and the following ones are added, a back fitting step is given. Consider C_1, C_2 and C_3 as the included candidates. Firstly, it removes C_1 from the ensemble and tests the addition of each of the remaining candidates $C_i (i > 3)$ to the ensemble. It repeats this step for C_2 and C_3. It chooses the best of the tested sets. Then it executes further iterations until a pre-defined number of iterations is reached.

6.3.4 Ranked pruning algorithms

The ranked pruning algorithms sort the models according to a certain criterion and generate an ensemble containing the top k models in the ranking. The value of k is either given or determined on the basis of a given criterion, namely, a threshold, a minimum, a maximum, etc.

Partridge & Yates rank the models according to accuracy [Partridge and Yates, 1996]. Then, the k most accurate models are selected. As expected, results are not good because there is no guarantee of diversity. Kotsiantis & Pintelas use a similar approach [Kotsiantis and Pintelas, 2005]. For each model a t-test is done with the aim of comparing its accuracy with the most accurate model. Tests are carried out using randomly selected 20% of the training set. If the p-value of the t-test is lower than 5%, the model is rejected. The use of heterogeneous ensembles is the only guarantee of diversity. Rooney et al. use a metric that tries to balance accuracy and diversity [Rooney *et al.*, 2004].

Perrone & Cooper describe an algorithm that removes similar models from the pool [Perrone and Cooper, 1993]. It uses the correlation matrix of the predictions and a pre-defined threshold to identify them.

6.3.5 Clustering algorithms

The main idea of clustering is to group the models into several clusters and choose representative models (one or more) from each cluster.

Lazarevic uses the prediction vectors made by all the models in the pool [Lazarevic, 2001]. The k-means clustering algorithm [Kaufman and Rousseeuw, 1990] is used over these vectors to obtain clusters of similar models. Then, for each cluster, the algorithms are ranked according to their accuracy and, beginning with the least accurate, the models are removed (unless their disagreement with the remaining ones overcomes a pre-specified threshold) until the ensemble

accuracy on the validation set starts decreasing. The number of clusters (k) is an input parameter of this approach, i.e., in practice this value must be tested by running the algorithm for different k values or, as in Lazarevic's case, an algorithm is used to obtain a default k [Lazarevic, 2001]. The experimental results reported are not conclusive.

Coelho & Von Zuben [Coelho and Von Zuben, 2006] use the ARIA - Adaptive Radius Immune Algorithm [Bezerra *et al.*, 2005], for clustering. This algorithm does not require a pre-specified k parameter. Only the most accurate model from each cluster is selected.

6.3.6 A discussion on ensemble pruning

Partridge & Yates compare three of the approaches previously described [Partridge and Yates, 1996]: (1) Ranked according to accuracy; (2) FSS using a diversity measure; and (3) a genetic algorithm. The results are not conclusive because just one data set is used. The FSS using a diversity measure gives the best result. However, as pointed out by the authors, the genetic algorithm result, even if not very promising, can not be interpreted as being less adapted to ensemble pruning. The result can be explained by the particular choices used for this experiment. Ranked according to the accuracy gives the worst result as expected since this method does not evaluate the contribution of each model to the overall diversity.

Roli et al. compare several pruning algorithms using one data set with three different pools of models [Roli *et al.*, 2001]. In one case, the ensemble is homogeneous (they use 15 neural networks trained using different parameter sets), in the other two cases they use heterogeneous ensembles. The algorithms tested are: FSS selecting the best model in the first iteration and randomly selecting a model for the first iteration, BSS, tabu search, Giacinto & Roli's clustering algorithm [Giacinto and Roli, 2001], and some others. The tabu search and the FSS selecting the best model in the first iteration give good results for the three different pools of models.

Coelho & Von Zuben also use just one data set to compare FSSwR, FSS, BSSwR, BSS and the clustering algorithm using ARIA [Coelho and Von Zuben, 2006]. Each of these algorithms is tested with different integration approaches. Results for each of the tested ensemble pruning algorithms give similar results, but for different integration methods. Ensembles obtained using the clustering algorithm and BSS have higher diversity.

The ordered bagging algorithm by Martínez-Muñoz & Suárez is compared with FSS using, also, just one data set [Martínez-Muñoz and Suárez, 2006]. The main advantage of ordered bagging is the meaningfully lower computational cost. The differences in accuracy are not meaningful.

Ruta & Gabrys compare a genetic algorithm, a population-based incremental learning algorithm [Baluja, 1994] and tabu search [Glover and Laguna, 1997] on three classification data sets [Ruta and Gabrys, 2001]. Globally, differences are not meaningful between the three approaches. The authors used a pool of just fifteen models, not enough to explore the differences between the three methods.

All of these benchmark studies discussed are for ensemble classification. It seems that more sophisticated algorithms like the tabu search, genetic algorithms, population based incremental learning, FSS, BSS or clustering algorithms are able to give better results, as expected. However, all studies use a

very small number of data sets, limiting the generalization of the results.

The evaluation function used for ensemble pruning has not yet been discussed in this thesis. This will be discussed in Sect. 6.4.

6.4 Ensemble integration

Now that we have described the process of ensemble generation, we move to the next step: how to combine the strengths of the models in one ensemble to obtain one single answer, i.e., ensemble integration.

For regression problems, ensemble integration is done using a linear combination of the predictions. This can be stated as

$$\hat{f}_f(\mathbf{x}) = \sum_{i=1}^{k} [h_i(\mathbf{x}) * \hat{f}_i(\mathbf{x})], \tag{6.10}$$

where $h_i(\mathbf{x})$ are the weighting functions.

Merz divides the integration approaches into constant and non-constant weighting functions [Merz, 1998]. In the first case, the $h_i(\mathbf{x})$ are constants (in this case α_i will be used instead of $h_i(\mathbf{x})$ in Eq. 6.10) while, in the second one, the weights vary according to the input values $\mathbf{x}$.

When combining predictions, a possible problem is the existence of multi-collinearity between the predictions of the ensemble models. As a consequence of the multi-collinearity problem, the confidence intervals for the α_i coefficients will be larger, i.e., the estimators of the coefficients will have higher variance [Merz, 1998]. This happens because we must determine the inverse of a linearly dependent matrix to obtain the α_i's.

A common approach is to handle multicollinearity in the ensemble generation (Sect. 6.2) or in the ensemble pruning (Sect. 6.3) phases. If the principles referred to in Sect. 6.1.4, namely those of accuracy and diversity, are assured, then it is possible, if not to completely avoid, at least to ameliorate this problem.

This section follows the Merz classification. It finishes with a discussion on ensemble integration methods.

6.4.1 Constant weighting functions

Constant weighting integration functions always use the same set of coefficients, regardless of the input for prediction. They are summarized in Fig. 6.2. Some methods use the test data (td) to obtain the α_i weights of the integration function (Eq. 6.10).

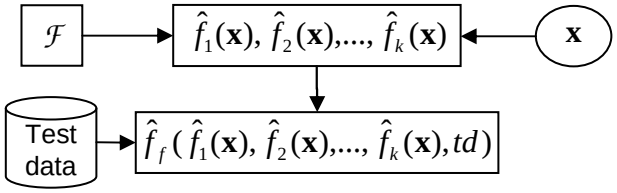

Figure 6.2: Constant weighting functions model.

Next, we describe the main constant weighting functions closely following Merz [Merz, 1998].

The BEM - Basic Ensemble Method [Perrone and Cooper, 1993] uses as estimator for the target function

$$\hat{f}_{BEM}(\mathbf{x}) = \sum_{i=1}^{k} \left[\frac{1}{k} * \hat{f}_i(\mathbf{x}) \right]. \tag{6.11}$$

This formula can be written as

$$\hat{f}_{BEM}(\mathbf{x}) = f(\mathbf{x}) - \frac{1}{k} \sum_{i=1}^{k} m_i(\mathbf{x}), \tag{6.12}$$

where

$$m_i(\mathbf{x}) = f(\mathbf{x}) - \hat{f}_i(\mathbf{x}). \tag{6.13}$$

BEM assumes that the $m_i(\mathbf{x})$ are mutually independent with zero mean.

To address this issue, Perrone & Cooper propose the GEM - Generalized Ensemble Method [Perrone and Cooper, 1993]. For GEM, the estimator is

$$\hat{f}_{GEM}(\mathbf{x}) = \sum_{i=1}^{k} [\alpha_i * \hat{f}_i(\mathbf{x})] = f(\mathbf{x}) + \sum_{i=1}^{k} [\alpha_i * m_i(\mathbf{x})], \tag{6.14}$$

where

$$\sum_{j=1}^{k} \alpha_i = 1,$$
$$\alpha_i = \frac{\sum_{j=1}^{k} C_{ij}^{-1}}{\sum_{l=1}^{k} \sum_{j=1}^{k} C_{lj}^{-1}},$$
$$C_{ij} = E[m_i(\mathbf{x}) * m_j(\mathbf{x})].$$

The drawback of this method is the multicollinearity problem, since it is necessary to calculate the inverse matrix C^{-1}. The multicollinearity problem can be avoided by pruning the ensemble [Perrone and Cooper, 1993] (Sect. 6.3). A similar approach to the GEM method, which ignores some of the constraints was proposed by Hashem & Schmeiser [Hashem and Schmeiser, 1995]. It also suffers from the multicollinearity problem.

The well-known linear regression (LR) model is another possible combination method. The predictor is the same as in the GEM case but without the constraint $\sum_{j=1}^{k} \alpha_i = 1$. The use of a constant in the LR formula is not relevant in practice (the standard linear regression formulation uses it) because $E[\hat{f}_i(\mathbf{x})] \simeq E[f(\mathbf{x})]$ [LeBlanc and Tibshirani, 1996]. It would only be necessary if predictors were meaningfully biased.

All the methods discussed so far suffer from the multicollinearity problem with the exception of BEM. Next, we discuss methods that avoid this problem.

Caruana et al. embed the ensemble integration phase in the ensemble selection one [Caruana *et al.*, 2004]. By selecting with replacement, from the pool,

the models to include in the ensemble, and using the simple average as the weighting function, the α_i coefficients are implicitly calculated as the number of times that each model is selected over the total number of models in the ensemble (including repeated models).

Breiman presents the stacked regression [Breiman, 1996c] method based on the well-known stacked generalization framework [Wolpert, 1992] firstly presented for the classification problem. Given a $\mathcal{L}$ learning set with M examples, the goal is to obtain the α_i coefficients that minimize

$$\sum_{j=1}^{M}[f(\mathbf{x}_j) - \sum_{i=1}^{k} \alpha_i * \hat{f}_i(\mathbf{x}_j)]^2. \tag{6.15}$$

To do so, the learning set used to obtain the α_i coefficients will be the same one used to train the $\hat{f}_i$ estimators, over-fitting the data. This problem is solved by using υ-fold cross-validation,

$$\sum_{j=1}^{M}[f(\mathbf{x}_j) - \sum_{i=1}^{k} \alpha_i * \hat{f}_i^{(-\upsilon)}(\mathbf{x}_j)]^2. \tag{6.16}$$

The second problem is the possible existence of multicollinearity between the $\hat{f}_i$ predictors. Breiman presents several approaches to obtain the α_i coefficients but concludes that a method that gives consistently good results is the minimization of the above equation under the constraints $\alpha_i \geq 0, i = 1, ..., k$ [Breiman, 1996c]. One of the methods tried by Breiman to obtain the α_i coefficients is ridge regression, a regression technique for solving badly conditioned problems, but results were not promising [Breiman, 1996c]. An important result of Breiman is the empirical observation that, in most cases, many of the α_i weights are zero. This result supports the use of ensemble pruning as a second step after the ensemble generation (Sect. 6.3).

Merz & Pazzani use principal component regression to avoid the multicollinearity problem [Merz and Pazzani, 1999]. The PCR* method obtains the principal components (PC) and then it selects the number of PCs to use. Once the PCs are ordered as a function of the variation they can explain, the search for the number of PCs to use is much simplified. The choice of the correct number of PCs is important to avoid under-fitting or over-fitting.

Evolutionary algorithms have also been used to obtain the α_i coefficients [Ortiz-Boyer *et al.*, 2005]. The used approach is globally better than BEM and GEM on twenty-five classification data sets. Compared to BEM,it wins nineteen, draws one and loses five. Compared to GEM, it wins seventeen, draws one and loses seven. These results were obtained by direct comparison, i.e., without statistical validation.

The main study comparing constant weighting functions is presented by Merz [Merz, 1998]. The functions used are: GEM, BEM, LR, LRC (the LR formula with a constant term), gradient descent, EG, EG^{+}_{-} (the last three methods are gradient descent procedures discussed in [Kivinen and Warmuth, 1997]), ridge regression, constrained regression (Merz [Merz, 1998] uses the bounded variable least squares method from [Stark and Parker, 1995]), stacked constrained regression (with ten partitions) and PCR*. Two of the three experiments reported by Merz are summarized next. The first experiment used an ensemble of twelve models on eight regression data sets: six of the models were

generated using MARS [Friedman, 1991] and the other six using the neural network back-propagation algorithm. The three globally best functions were constrained regression, EG and PCR*. The second experiment tests how the functions perform with many correlated models. The author uses neural network ensembles of size ten and fifty. Just three data sets were used. The PCR* function presents more robust results. See [Merz, 1998] to get details on the experiments and their results.

6.4.2 Non-constant weighting functions

The non-constant weighting functions use different coefficients according to the input for prediction. They can be static (defined at learning time) or dynamic (defined at prediction time). The static ones can use two different approaches: (1) to assign models to predefined regions, this is the divide-and-conquer approach already discussed in Sect. 6.2.1; or (2) to define the areas of expertise for each model, also called static selection [Kuncheva, 2002], i. e., for each predictor from the ensemble, the input subspace where the predictor is expert is defined. In the dynamic approach, the $h_i(\mathbf{x})$ weights from Eq. 6.10 are obtained on the fly based on the performances of the base predictors on data, from the training set, which is similar to $\mathbf{x}$.

The approach by areas of expertise

The work on meta decision trees [Todorovski and Dzeroski, 2003] for classification induces meta decision trees using as variables meta-attributes that are previously calculated for each example. The target variable of the meta tree is the classifier to recommend. In practice, the meta decision tree, instead of predicting, recommends a predictor. Although this work has been developed for classification, it could be easily adapted for regression by an appropriate choice of the meta attributes.

Yankov et al. use support vector machines with the Gaussian kernel to select the predictor to use from an ensemble with two models [Yankov *et al.*, 2006].

The dynamic approach

In the dynamic approach, the predictor(s) selection is done on the fly, i.e., given the input vector for the prediction task, it chooses the expected best predictor(s) to accomplish this task. While in the approach by areas of expertise these areas are previously defined, in the dynamic approach the areas are defined on the fly. Selection can be seen as a kind of pruning on the fly.

Figure 6.3 summarizes the dynamic approach. Given an input vector $\mathbf{x}$, it first selects similar data. Then, according to the performance of the models on this similar data, a number k_1 of models are selected from the ensemble $\mathcal{F}$ (Eq. 6.1). Merz describes in detail this approach, namely the use of a performance matrix to evaluate the models locally [Merz, 1996]. It consists of a $m \times k$ matrix, where m is the number of past examples and k is the number of models in $\mathcal{F}$. Each cell has a performance measure of each model for each example obtained from the models' past predictions and the respective actual target values for these examples. In regression, this measure can be, for instance, the squared error, the absolute error, or other performance measure. If $k_1 = 1$ then $\hat{f}_f$ is the

identity function. If $k_1 > 1$ then the integration method uses the performances of similar data (sd) obtained from the test data (td) to estimate the $h_i(\mathbf{x})$ weights.

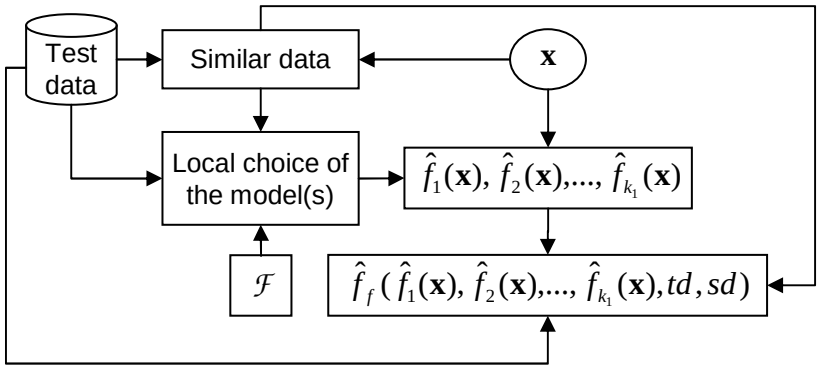

Figure 6.3: Dynamic approach model.

This approach consists of the following tasks (assuming that the ensemble models are already available):

1. given an input value $\mathbf{x}$, find similar data from the validation set;
2. select the model(s) from the ensemble according to their performance for the selected similar data;
3. obtain the prediction $\hat{f}_i(\mathbf{x})$ for the given input value, for each selected i model;
4. obtain the ensemble prediction $\hat{f}_f$. This is straightforward if just one model is selected, otherwise, it is necessary to combine results.

While task (3) is straightforward, the others are not. In this section related works concerning the remaining three tasks are reviewed.

The standard method for obtaining similar data (task 1) in the context of ensemble learning is the well-known k-nearest neighbors with the Euclidean distance [Woods, 1997]. One limitation of this method is that it weights equally all the input variables even if there are input variables with different levels of relevance in the explanation of the target variable. If the data set has many variables with a small impact on the target variable and a small subset of dominant input variables, this potential problem can become a real one. Known solutions already exist in the context of random forests - RF.

RF, a decision tree based method, builds several trees independently and obtains the final prediction by averaging the results of these trees, for regression, and by simple majority voting, for classification (see [Breiman, 2001a] for details). RF uses bootstrap samples and random feature selection at each split (during the process of tree construction) in order to obtain diversity in the produced trees. When a bootstrap sample is generated 36.8% of the examples, on average, are left out. These are called the out-of-bag examples, i.e., examples that were not used to train a particular tree. This particular characteristic of RF allows the use of a natural validation set for each tree: the set of out-of-bag

examples. These examples can be used to obtain the weights for the dynamic voting approach, by measuring the distance of similar examples in the out-of-bag set [Robnik-Šikonja, 2004; Tsymbal *et al.*, 2006]. Tsymbal et al. test two different distance metrics, RFIS and HEOM, to obtain the similar example set [Tsymbal *et al.*, 2006]. RFIS gives, for each example, the proportion of trees from the forest where the example appears together with the test example in the same leaf node. HEOM - Heterogeneous Euclidean/Overlap Metric [Wilson and Martinez, 1997] is a successful distance function for mixed numeric and nominal attributes. The authors use, as integration method, dynamic voting with selection, adopting equally weighted and locally weighted voting schemes instead of the majority voting used in RF. They obtained poor results with dynamic selection, possibly due to the increased variance introduced by this technique. The tests using dynamic voting with selection improve accuracy in 12 out of 27 data sets when compared with majority voting. These results can be explained, at least partially, by the circumstance that, in RFIS, the similarity between two examples is measured by the comparison of the target values similarity, rather than by the proximity of the input variables. This procedure implicitly measures in a different way the distances for each input variable.

Didaci & Giacinto also test a kind of similarity measure according to the outputs [Didaci and Giacinto, 2004] embedded in DANN - Discriminant Adaptive Nearest Neighbor [Hastie and Tibshirani, 1996]. DANN locally reshapes the nearest neighborhood. In practice, some of the explanatory variables are discarded, reducing the dimensionality of the problem. This method can be used for both classification and regression. Experiments done by Didaci & Giacinto show that this approach as well as a dynamic choice of the number k of neighbors can meaningfully improve the results when compared with the standard Euclidean distance [Didaci and Giacinto, 2004].

The simplest method of selecting models (task 2) is to pick the one with the best performance for a given metric [Woods, 1997; Giacinto and Roli, 1997]. However, the dynamic selection approach can use more than one model [Merz, 1996]. The dynamic voting with selection [Tsymbal and Puuronen, 2000] and its regression version, called dynamic weighting with selection [Rooney *et al.*, 2004], use the 50% locally more accurate models from the ensemble.

When more than one model is selected, their results are combined (task 4). Each one of the methods discussed in Sect. 6.4.1 can be used to accomplish this task. Anyway, the approaches used in the context of dynamic selection are described.

Wang et al. use weights $h_i(\mathbf{x})$ (see Eq. 6.10) inversely proportional to $\hat{f}_i(\mathbf{x})$ expected error [Wang *et al.*, 2003]. This approach is similar to the variance based weighting presented in [Tresp and Taniguchi, 1995].

The dynamic integration methods originally presented by Puuronen et al. [Puuronen *et al.*, 1999] for classification are adapted for regression in [Rooney *et al.*, 2004]. Dynamic selection (DS) selects the predictor with less cumulative error on the k-nearest neighbors set. Dynamic weighting (DW) assigns a weight to each base model according to its localized performance on the k-nearest neighbors set [Rooney *et al.*, 2004] and the final prediction is based on the weighted average of the predictions of the related models. Dynamic weighting with selection (DWS) is similar to DW but the predictors with cumulative error in the upper half of the error interval are discarded. From the three methods tested, the DWS using just the subset of the most accurate predictors for the

final prediction gets the best results.

6.4.3 A discussion on ensemble integration

The studies on how to handle the multicollinearity problem, quite common in the nineties, have become less frequent in the last few years, apparently because most of the effort shifted to the generation phase. The approach seems to have changed from "what integration function to use for a given ensemble" to "how to generate the ensemble for a given integration function". In all the studies highlighted in Sect. 6.2.3, constant weighting functions were used. It is already known from the decomposition of the generalization error (Sect. 6.1.4) what characteristics of the ensemble are the expected when using constant weighting functions. The question now is: "how to generate ensembles for non-constant weighting functions". We try to give a contribution on this topic (Sect. 7.2) by testing evaluation functions for the pruning algorithms potentially better adapted to non-constant weighting functions. It is, in practice, an indirect way of ensemble generation for non-constant weighting functions.

The experiments described by Merz (Sect. 6.2) were published in 1998 [Merz, 1998]. However, since 1995, maybe due to the advances in the studies on the generalization ensemble error (Sect. 6.1.4), the ensemble generation research has been driven towards the quality of the ensembles [Rosen, 1996; Liu *et al.*, 2000; Zhou *et al.*, 2002]. These examples show that an important part of the problems at the integration phase can be solved by a joint design of the generation, pruning (when appropriate) and the integration phases.

The main disadvantage of constant weighting functions is that the α_i weights, being equal for all the input space, can, at least theoretically, be less adequate for some parts of the input space. This is the main argument for using non-constant weighting functions [Verikas *et al.*, 1999]. This argument can be particularly true for time changing phenomena [Wang *et al.*, 2003]. Both approaches are compared in Chap. 7.

Ensemble integration approaches can also be classified as selection or combination ones [Kuncheva, 2002]. In the selection approach the final prediction is obtained by using just one predictor, while in the combination one the final prediction is obtained by combining predictions of two or more models. Kuncheva presents a hybrid approach between the selection and the combination ones [Kuncheva, 2002]. It uses paired t-hypothesis test [Groebner and Shannon, 1985] to verify whether one predictor is meaningfully better than the others. If positive, it uses the best predictor, if not it uses a combination approach.

An approach that can be explored and seems to be promising is to combine different ensemble integration methods. The method wMetaComb [Rooney and Patterson, 2007] uses a weighted average to combine stacked regression (described in Sect. 6.4.1) and the DWS dynamic method (Sect. 6.4.2). The cocktail ensemble for regression [Yu *et al.*, 2007] combines different ensemble approaches, whichever they are, using a combination derived from the ambiguity decomposition. The authors leave for a future paper the full description of that derivation. They combine different ensembles using a forward approach by selecting the one that reduces the combined estimated error the most. The same ensemble can be selected more than once.

Table 6.2: Main homogeneous ensemble generation methods.

Method	Reference	Algorithm	Class/Regr
Bagging	[Breiman, 1996a]	Unst. learners	yes / yes
AdaBoost	[Freund and Schapire, 1996]	Unst. learners	yes / ?
Random forests	[Breiman, 2001a]	Decis. trees	yes / yes
EENCL	[Liu *et al.*, 2000]	ANN	yes / yes
CNNE	[Islam *et al.*, 2003]	ANN	yes / yes
Coop. Coev.	[García-Pedrajas *et al.*, 2005]	ANN	yes / ?

6.5 Conclusions

The use of ensemble methods has as its main advantage the increase in accuracy and robustness, when compared to the use of a single model. This makes ensemble methods particularly suitable for applications where small improvements in the predictions have an important impact.

For ensemble learning, as for other research areas, methods for regression and for classification have different solutions, at least partially. The methods for ensemble learning have, typically, three phases: generation, pruning (not always) and integration.

The generation phase aims to obtain an ensemble of models. It can be classified as homogeneous or as heterogeneous. This classification depends on the number of induction algorithms used. If just one is used, the ensemble is classified as homogeneous, otherwise it is classified as heterogeneous. The most successful methods for ensemble generation are developed for unstable learners, i.e., learners that are sensitive to changes in the training set, namely decision trees or neural networks. Table 6.2 summarizes some of the most important methods on homogeneous ensemble generation. The mark ? means that there are currently no promising versions of this algorithm for regression (the case of AdaBoost), or that these methods are not even tested for regression (the case of cooperative co-evolution), and consequently it is not known how these methods could work for regression.

Ensemble pruning aims to select a subset from a pool of models in order to reduce computational complexity and, if possible, to increase accuracy. It has many similarities with the well-known feature subset selection task. This happens because in both cases the goal is to select, from a set, a subset in order to optimize a given objective function. As in the feature subset selection case, randomized heuristics, such as evolutionary algorithms or tabu search, seem to be very effective.

Ensemble integration functions use the predictions made by the models in the ensemble to obtain the final ensemble prediction. They can be classified as constant or non-constant weighting functions. As previously underlined, constant weighting functions are the most used, possibly because it is easier to generate ensembles in order to minimize known generalization error functions. Since non-constant weighting functions seem to be attractive in order to increase accuracy, further research is needed to obtain ensemble methods that take advantage of such integration functions.

This chapter described the complete process for ensemble based regression. As shown, at each step there are many challenging problems to be solved, and

many ideas still need theoretical and experimental development. We believe that this survey provides a thorough road map that can serve as a stepping stone to new ideas for research.

Chapter 7

Experiments on heterogeneous ensembles

In the sequence of the analysis of the results using regression models in order to address the problem of travel time prediction (TTP) three days ahead (Sect. 5.8), we decided to study the use of an ensemble of models in order to increase accuracy comparing to the use of only one model. With this purpose, we did a comprehensive state of the art review on ensemble methods for regression (Chap. 6). Using the lessons from the survey, we explore in this chapter the ensemble approach:

1. Taking advantage of the multiplicity of results from the experiments done in Chap. 5 complemented with the good indications on heterogeneous ensembles [Wichard *et al.*, 2003] and the fact that one of the methods tested in Chap. 5 is a state of the art homogeneous ensemble method for regression (it is the case of Random Forests), we decided to explore the use of heterogeneous ensembles using the overproduce-and-choose approach. The pools used along this chapter, i.e., the initial ensembles from which the final ensembles are selected using a pruning method, are presented in Sect. 7.1.

2. For ensemble pruning, we study the selection of ensembles more skilled for different types of integration functions (constant and nonconstant). Some experiments are done in Sect. 7.2 exploring two different search schemas and two different evaluation functions. The evaluation functions are the simple average (*avg*) and the sum of minimum squared errors (*smse*). The *avg* is certainly more skilled for the constant weighting function simple average while the *smse* is potentially more skilled for nonconstant weighting functions.

3. For ensemble integration we aim to evaluate both constant (Sect. 6.4.1) and nonconstant weighting functions. From the existing nonconstant weighting functions, we study the dynamic approach discussed in Sect. 6.4.2. This option seems well adapted for data streams problems [Tsymbal *et al.*, 2008] and has been reported as giving better results than constant weighting functions [Verikas *et al.*, 1999]. TTP can also be seen as a data

stream problem despite this was not the approach used along this thesis. In order to study the best setup for the dynamic approach we discuss different methods to obtain similar data, to select the models and to combine their predictions (Sect. 7.3).

4. In order to test the different possibilities discussed so far they must be tested in an ensemble learning framework. We use a framework based in stacked generalization [Wolpert, 1992] (Sect. 7.4).

5. This framework is used to do three different experiments. Despite we address different issues in all the experiments (see Sect. 7.5), the main motivation for each one of the experiments is:

   - The first experiment aims to evaluate the different methods to obtain similar data (as discussed in the third item and Sect. 7.3.1). The one with best results selects the examples that fall in the same leaf node of a CART tree together with the test example and then, it chooses from these examples the k nearest according to the HEOM distance (Sect. 7.5.1). How this new method performs in other domains is an open issue for research.
   - The second experiment aims to evaluate how sensitive the ensemble is to the data set used in the ensemble generation phase (Sect. 7.5.2. With this purpose we test how ensembles learned on a 3 month data set (January to March 2004) performs on a 8 month data set (January to August 2006) by comparison against the performance on a 3 month data set (January to March 2006). Results show that some a&ps in the ensemble degrade performance on the larger data set. This result is important because it shows that the data set used in the ensemble generation should be representative of all the input space.
   - The third experiment tests ensembles selected from the pool using the two different evaluation functions discussed in the second item and Sect. 7.2, *avg* and *smse*. Results show that *avg* performs globally better than *smse* for all the integration functions tested.

The main result from all the experiments done using the ensemble approach is the reduction of the generalization error when compared against the error of the base learners. Anyway, the main goal of this chapter is to study the best setup for the ensemble learning approach. The comparison of this approach with the other ones discussed in Chap. 5 is the subject of Chap. 8.

7.1 A&ps used

The generated pool is not a pool of models but, instead, a pool of algorithms & parameter sets (*a&ps*). The reason is that, once we use a time stamp-based sliding window for resampling (Sect. 4.2.4) and we do not keep the models obtained with past windows, what we have in the ensembles are not models but the information on the algorithms and respective parameter sets. Indeed, using three month of data, the experiment with a particular *a&ps* generates several models, one for each training set in the sliding window process. The predictions

obtained for that *a&ps* were obtained using different models (one model for each different test day).

The best pool is the one that allows the highest possible accuracy. However, pool generation is strongly computer intensive. When we increase the number of *a&ps* in the pool, the accuracy potentially increases, but the time needed to generate it also increases. In practice, the pool should have as many *a&ps* as possible, giving priority to *a&ps* able to learn accurate and diverse models. The diversity can be obtained using different algorithms and using some of the methods discussed in the previous chapter.

Three different pools of *a&ps* were obtained using the results from Chap. 5. Given the circumstances, the pools were not generated but, instead, selected in a previous pruning step. The reason for the use of several pools was to measure the sensitivity of the accuracy of the ensemble prediction to the pool. This method to obtain the pools was already used by other authors, namely [Caruana *et al.*, 2004]. The used pools are:

- The pool 1538. It is the largest one with 1538 *a&ps*, as the name suggests. It uses a selection of the *a&ps* tested in Chap. 5 (Table 7.1). In the table, ES identifies the method used for example selection: 'all' uses all the examples for training, 'ln' uses the leaf node approach and 'ed' uses the equivalent days approach, as explained in Sect. 5.7.1.
- The pool 1234 (Table 7.2) does not use example selection, i.e., it uses always the 'all' approach.
- The pool 130 (Table 7.3) has less *a&ps* from each one of the algorithms comparing to the other pools. The only algorithm that is not used in this pool is SVM sigmoid. The expanded description of this pool is in Appendix B.

In all the pools, SVM uses the numeric data type for the variable week day while PPR uses the symbolic data type. RF uses the numeric data type in all but pool 1234.

7.2 Choosing the ensemble pruning techniques

The goal of ensemble pruning is to select a set of models (in this case, of *a&ps*) from a pool. Two main choices must be made (using the categorization for feature selection presented in Sect. 4.3.1): the search algorithm; and the evaluation function. As already discussed in the previous chapter, ensembles are constructed, typically, for the use of constant weighting functions. However, the study on how the ensemble of models/a&ps should be in order to take the most of nonconstant weighting functions, is almost not discussed in the literature. The only known work on this subject presents a study on the best set of pruning algorithm and integration function in order to optimize the ensemble accuracy [Coelho and Von Zuben, 2006]. The results presented are not very impressive. The best pair (pruning and integration algorithms) is problem dependent (they use four classification data sets). The authors test several pruning algorithms differing only in the searching schema. This fact can explain the results, at least partially. In fact, it is not expected that a searching method is more skilled for a particular integration method than another one. Conversely,

Table 7.1: *A&ps* in the pool 1538.

Algorithm	**ES**	**Par 1**	**Par 2**	**Par 3**	**Par 4**
SVM - linear	ln	2^{2idx_1}	$\frac{\text{idx}_2}{10}$		
SVM - radial	all	$2^{2\text{idx}_3} \times 1000$	$\frac{\text{idx}_4}{5}$	$\frac{6\text{idx}_5}{100000}$	
SVM - radial	ln	$2^{2\text{idx}_3} \times 1000$	$\frac{\text{idx}_4}{5}$	$\frac{6\text{idx}_5}{100000}$	
SVM - sigmoid	ln	$2^{2\text{idx}_3} \times 1000$	$\frac{\text{idx}_4}{5}$	$\frac{12\text{idx}_3-10}{1000000}$	-0.5idx_6
RF	ln	idx_4			
RF	ed	idx_4			
PPR - supsmu	ed	idx_4	idx_5	idx_7	0
PPR - supsmu	ed	idx_4	idx_5	0	$\frac{\text{idx}_1+3}{10}$
PPR - spline	all	idx_4	idx_5	2^{idx_7}	
PPR - spline	ed	idx_4	idx_5	2^{idx_7}	
PPR - gcvspline	all	idx_4	idx_5	2^{2idx_1}	
PPR - gcvspline	ed	idx_4	idx_5	2^{2idx_1}	

$\text{idx}_1 = -2, -1, \cdots, 6; \text{idx}_2 = 1, 2, \cdots, 10; \text{idx}_3 = 1, 2, \cdots, 5; \text{idx}_4 = 1, 2, \cdots, 4; \text{idx}_5 = 1, 2, 3; \text{idx}_6 = -1, 0, \cdots, 4; \text{idx}_7 = 0, 1, \cdots, 10.$

For SVM: Par 1 = C, Par 2 = ν, Par 3 = γ and Par 4 = coef0.

For RF: Par 1 = mtry.

For PPR: Par 1 = nterms and Par 2 = optlevel.

For PPR-supsmu: Par 3 = bass and Par 4 = span.

For PPR-spline: Par 3 = df.

For PPR-gcvspline: Par 3 = gcvpen.

Table 7.2: *A&ps* in the pool 1234.

Algorithm	**Par 1**	**Par 2**	**Par 3**	**Par 4**
SVM - linear	2^{2idx_1}	$\frac{\text{idx}_2}{10}$		
SVM - radial	$2^{2\text{idx}_3} \times 1000$	$\frac{\text{idx}_4}{5}$	$\frac{6\text{idx}_5}{100000}$	
SVM - sigmoid	$2^{2\text{idx}_3} \times 1000$	$\frac{\text{idx}_4}{5}$	$\frac{12\text{idx}_3-10}{1000000}$	-0.5idx_6
RF	idx_4			
PPR - supsmu	idx_4	idx_5	idx_7	0
PPR - supsmu	idx_4	idx_5	0	$\frac{\text{idx}_1+3}{10}$
PPR - spline	idx_4	idx_5	2^{idx_7}	
PPR - gcvspline	idx_4	idx_5	2^{2idx_1}	

$\text{idx}_1 = -2, -1, \cdots, 6; \text{idx}_2 = 1, 2, \cdots, 10; \text{idx}_3 = 1, 2, \cdots, 5; \text{idx}_4 = 1, 2, \cdots, 4; \text{idx}_5 = 1, 2, 3; \text{idx}_6 = -1, 0, \cdots, 4; \text{idx}_7 = 0, 1, \cdots, 10.$

For SVM: Par 1 = C, Par 2 = ν, Par 3 = γ and Par 4 = coef0.

For RF: Par 1 = mtry.

For PPR: Par 1 = nterms and Par 2 = optlevel.

For PPR-supsmu: Par 3 = bass and Par 4 = span.

For PPR-spline: Par 3 = df.

For PPR-gcvspline: Par 3 = gcvpen.

Table 7.3: *A&ps* in the pool 130.

Algorithm	**ES**	**Par 1**	**Par 2**	**Par 3**	**Par 4**
SVM - linear	all	2^{-14+10idx_1}	$\frac{2\text{idx}_2-1}{10}$		
SVM - linear	ln	2^{-14+10idx_1}	$\frac{2\text{idx}_2-1}{10}$		
SVM - radial	all	$2^{4\text{idx}_3-2} \times 1000$	$\frac{4\text{idx}_2-2}{10}$	$\frac{12\text{idx}_2-6}{100000}$	
SVM - radial	ln	$2^{4\text{idx}_3-2} \times 1000$	$\frac{4\text{idx}_2-2}{10}$	$\frac{12\text{idx}_2-6}{100000}$	
RF	ln	$2\text{idx}_2 - 1$			
PPR - supsmu	all	$3\ \text{idx}_2 - 2$	idx_3	$5\text{idx}_3 - 5$	0
PPR - supsmu	ed	$3\ \text{idx}_2 - 2$	idx_3	$5\text{idx}_3 - 5$	0
PPR - spline	all	idx_2	$2\text{idx}_2 - 1$	2^{5idx_3-5}	
PPR - spline	ed	idx_2	$2\text{idx}_2 - 1$	2^{5idx_3-5}	
PPR - gcvspline	all	idx_2	$2\text{idx}_2 - 1$	2^{8idx_3-12}	
PPR - gcvspline	ed	idx_2	$2\text{idx}_2 - 1$	2^{8idx_3-12}	

$\text{idx}_1 = 1, 2, \cdots, 5; \text{idx}_2 = 1, 2; \text{idx}_3 = 1, 2, 3.$
For SVM: Par 1 = C, Par 2 = ν, Par 3 = γ and Par 4 = coef0.
For RF: Par 1 = mtry.
For PPR: Par 1 = nterms and Par 2 = optlevel.
For PPR-supsmu: Par 3 = bass and Par 4 = span.
For PPR-spline: Par 3 = df.
For PPR-gcvspline: Par 3 = gcvpen.

each evaluation measure is expected to be more adapted to certain integration functions. For that reason we made experiments in order to test two evaluation functions. Additionally we also test two search algorithms.

The search algorithms tested are:

- the well known Forward Sequential Selection (FSS) algorithm (Sect. 6.3.3); and
- the algorithm by Moreira et al. already discussed in Sect. 6.3.3. This algorithm is referred as Mor06 from now on. Despite the name of the algorithm, we believe that this search method was already known, even if we were not able to find a reference for it.

Both search algorithms are expected to stop at a local minima. While FSS is deterministic, Mor06 is not, due to its random seed, i.e., different runs of the algorithm return, typically, different results.

The evaluation functions tested are:

- the *variation index* (Eq. 4.5) for the predictor simple average. When this metric is used, the algorithm gains the suffix '-avg'. For example, FSS-avg is the FSS search algorithm using this evaluation function (Fig. 7.1).
- the same *variation index* but for the predictor with minimum squared error at each data point. This metric is identified by the suffix '-smse' meaning sum of minimum squared errors. Fig. 7.2 presents the pseudo-code for Mor06-smse.

FSS-smse is obtained from FSS-avg by substituting the *avg* evaluation function by the *smse* one (line 5 from Fig. 7.1 with line 6 from Fig. 7.2). Similarly Mor06-avg is obtained from Mor06-smse by substituting the *smse* evaluation function by the *avg* one (lines 6 and 12 from Fig. 7.2 with line 5 from Fig. 7.1).

Require: $M((t+1) \times m)$, a matrix with t predictions (from t *a&ps*) and the actual value (the $t+1$ row) for the m trips
Require: n, the number of *a&ps* to be selected from M
Require: $in.array$, an array with the identification of the lines of M to be included in the final set. This parameter is useful to reduce the computational cost when some of the *a&ps* are already known
1: $in.set := M_{in.array,\cdot}$
2: $out.set := M_{1..t,\cdot} \setminus in.set$
3: **while** $size(in.set) < n$ **do**
4: **for** all the rows of $out.set$ ($cur.row$) **do**
5: $eval.value(cur.row) := \sum_{j=1}^{m}((avg(M_{i \in in.set \cup cur.row,j}) - M_{t+1,j})^2)$
6: **end for**
7: $best.in :=$ $out.set$ row with $min(eval.value)$
8: $in.set := in.set \cup best.in$
9: $out.set := out.set \setminus best.in$
10: **end while**
11: **return** varIndex using as predictor *avg* on the $in.set$ ensemble

Figure 7.1: The pseudo-code for FSS-avg.

The first evaluation function was expected, a priori, to be more adequate to constant weighting functions (in this particular case, the simple average). There is no special reason to choose the simple average for the evaluation, as there would not be any special reason to choose another constant weighting function, because it is not expected that the pool is more skilled for a particular constant weighting function. The simple average was chosen because of its simplicity.

On the other hand, the *smse* evaluation function was expected to be more appropriate for the dynamic selection approach (Sect. 6.4.2). The set of *a&ps* selected, are the ones that are accurate for a subset of the input space. The main assumption of this evaluation function is that the ensemble prediction is able to select the most accurate base predictor for each unlabelled example. The question is to know if these examples form a region of the input space or, instead, are sparse examples. In the last case, this metric will not be useful because it will not be possible to detect the expertise of the base learners for those particular examples when running the task 'local choice of the model(s)' (Fig. 6.3).

While the results of the algorithms using the *avg* evaluation measure are an estimate of the error for the integration function simple average (even if optimistic), the results of the algorithms using the *smse* evaluation measure is an estimate of the oracle when selecting dynamically one predictor. This happens because this evaluation measure uses the best prediction for each example, as previously referred. Consequently, the results of both algorithms cannot be compared.

Each combination of search algorithm and evaluation function was used for each one of the pools described in Sect. 7.1 for the selection of 5, 10, 15, 20 and

Require: $M((t+1) \times m)$, a matrix with t predictions (from t *a&ps*) and the actual value (the $t+1$ row) for the m trips
Require: n, the number of *a&ps* to be selected from M
1: $sn := n$ different random values between 1 and t
2: $in.set := M_{sn,\cdot}$
3: $out.set := M_{1..t,\cdot} \setminus in.set$
4: **repeat**
5: **for** all the rows of $out.set$ ($cur.row$) **do**
6: $eval.value(cur.row) := \sum_{j=1}^{m}(min((M_{i \in in.set \cup cur.row,j} - M_{t+1,j})^2))$
7: **end for**
8: $best.in := out.set$ row with $min(eval.value)$
9: $in.set := in.set \cup best.in$
10: $out.set := out.set \setminus best.in$
11: **for** all the rows of $in.set$, now with $n+1$ rows ($cur.row$) **do**
12: $eval.value(cur.row) := \sum_{j=1}^{m}(min((M_{i \in in.set \setminus cur.row,j} - M_{t+1,j})^2))$
13: **end for**
14: $best.out := in.set$ row with $min(eval.value)$
15: $in.set := in.set \setminus best.out$
16: $out.set := out.set \cup best.out$
17: **until** $best.in = best.out$
18: **return** varIndex using as predictor smse on the $in.set$ ensemble

Figure 7.2: The pseudo-code Mor06-smse.

25 *a&ps*. When using the Mor06 search algorithm, 10 runs were executed due to its random seed, and the run with the lowest error is chosen one (table 7.4).

The comparison between the two different evaluation functions is not useful, as previously referred. Comparing the results of the two different search methods we observe that FSS is worse in every situation except when selecting ensembles of sizes 20 and 25 in pool 1234. These results are better understood by analyzing the ten runs of Mor06-avg (instead of comparing the minimum of the ten runs) for each one of the pools using ensembles of sizes 5 and 25 (table 7.5). Results are surprising because the dispersion is quite small for all the pools except pool 1234. When we select an ensemble of size 5, the dispersion is high even if the minimum of the *variation index* on 10 runs is lower than the value obtained using FSS-avg. For the ensemble of size 25, the dispersion is lower but the minimum increases, being larger than the *variation index* obtained with FSS-avg. Despite the results for ensembles of size 10, 15 and 20 are not displayed, this is a tendency for the results with pool 1234: the larger the ensemble is the lower the dispersion of the results is. On the contrary, the minimum increases. Why does this happens? Why FFS is not affected by this problem? Table 7.6 shows some common statistics to describe the pools. Pool 130 has no outliers, i.e., all the *a&ps* are reasonably accurate. However, the other two pools have some outliers. The percentage of outliers in pool 1234 is larger than in pool 1538. Using the searching algorithm Mor06-avg, an initial subset is randomly selected. Then, the *a&ps* that most improves the evaluation measure (in this case, the simple average) is added to this subset. Next, the *a&ps* from the subset is removed in order to obtain the highest evaluation measure for the remaining *a&ps* in the subset. If the initial subset has one or more outliers, the *a&ps* that most

Table 7.4: The *variation index* for each one of the pruning algorithms on each one of the three used pools.

	smse					
	FSS			Mor06		
Size	P 1538	P 1234	P 130	P 1538	P 1234	P 130
5	5.05%	5.01%	5.34%	4.86%	4.95%	5.16%
10	3.50%	3.47%	4.13%	3.20%	3.39%	4.05%
15	2.78%	2.78%	3.66%	2.61%	2.71%	3.63%
20	2.43%	2.40%	3.47%	2.27%	2.36%	3.45%
25	2.19%	2.17%	3.36%	2.06%	2.13%	3.34%
	avg					
	FSS			Mor06		
Size	P 1538	P 1234	P 130	P 1538	P 1234	P 130
5	8.81%	9.07%	8.90%	8.80%	9.03%	8.90%
10	8.75%	9.10%	8.86%	8.72%	9.10%	8.85%
15	8.72%	9.18%	8.86%	8.71%	9.17%	8.86%
20	8.73%	9.23%	8.88%	8.72%	11.89%	8.88%
25	8.74%	9.27%	8.90%	8.73%	11.09%	8.90%

improves the ensemble's evaluation measure, when added to the initial subset, is the one that attenuates the most the effect of those outliers, probably another outlier negatively correlated to the first one(s). This process can stop in a local minima when the *a&ps* to remove is the last added *a&ps*. The smaller is the size of the ensemble to select the lower is the probability of randomly select an outlier. However, if one of them is selected, its effect on the ensemble evaluation is stronger (this explain the existence of large values of *variation indexes* for pool 1234 in table 7.5). When selecting larger ensembles, the probability of selecting an outlier is larger, but its effect in the *variation index* is smaller due to size of the ensemble. This explains the larger values for the minimum *variation index* in table 7.5 when we use pool 1234 and ensembles of size 20 and 25. FSS does not have this effect because the ensemble is constructed incrementally using forward search. This process is deterministic, there is no possibility of selecting an outlier *a&ps*.

7.3 Ensemble integration techniques used

As it was previously discussed in Sect. 6.4.2, dynamic selection (Fig. 6.3) has the following tasks: (1) find similar data; (2) select the model(s) to use in the prediction; (3) obtain the prediction(s) from the selected model(s); and (4) obtain the final ensemble prediction.

This section presents the methods used to accomplish tasks 1, 2 and 4 (task 3 is straightforward) in the experiments described in Sect. 7.5.

7.3.1 Obtaining similar data

Three methods were used to obtain similar data. Two of them use the well known k-nearest neighbor (knn) algorithm but with different distance functions:

Table 7.5: The *variation index* for the ten runs of Mor06-avg using ensembles of size 5 and 25 on each one of the three used pools.

Ensembles of size 5			Ensembles of size 25		
P 1538	P 1234	P 130	P 1538	P 1234	P 130
8.81%	31.11%	8.90%	8.74%	13.64%	8.90%
8.92%	31.24%	8.90%	8.74%	11.24%	8.90%
8.82%	9.03%	8.92%	8.73%	12.32%	8.90%
8.80%	38.58%	8.90%	8.74%	16.21%	8.90%
8.86%	48.83%	8.93%	8.73%	15.47%	8.90%
8.85%	9.06%	8.90%	8.74%	13.17%	8.90%
8.86%	74.01%	8.94%	8.73%	13.56%	8.90%
8.88%	9.04%	8.92%	8.73%	14.02%	8.90%
8.84%	9.06%	8.93%	8.73%	12.39%	8.90%
8.84%	9.03%	8.94%	8.74%	11.09%	8.90%

Table 7.6: Descriptive statistics for the *variation index* on each one of the three used pools.

		P 1538	P 1234	P 130
Percentil	0%	9.49%	9.82%	9.69%
	25%	10.50%	13.27%	10.68%
	50%	11.30%	14.53%	11.98%
	75%	13.49%	15.59%	14.18%
	80%	14.06%	15.99%	14.53%
	85%	14.94%	95.54%	14.78%
	90%	25.03%	736.25%	15.02%
	95%	359.25%	7365.49%	16.01%
	100%	21743.80%	220981.30%	24.76%
average		195.32%	2415.95%	12.60%
std. deviation		1266.22%	14885.58%	2.33%

the Heterogeneous Euclidean-Overlap Metric (HEOM) distance [Wilson and Martinez, 1997] and the feature weighted distance using the RReliefF algorithm to obtain the weights. The third one selects similar data using the CART algorithm [Breiman *et al.*, 1984] as described in Sect. 5.7.1 (Fig. 5.14).

The HEOM distance was already used in the literature on ensemble learning [Tsymbal *et al.*, 2006]. We use it for comparison against the other two methods, conceptually more complex. HEOM was already described in Sect. 5.4. Its main advantage over other standard distance measures, namely the Euclidean distance, is to handle simultaneously numeric and nominal attributes.

The RReliefF algorithm [Robnik-Šikonja and Kononenko, 2003] (already discussed in Sect. 5.7.3, Fig. 5.22) weighs each one of the input variables. These weights are used to obtain a feature weighted distance. The array of weights W returned by the RReliefF algorithm (one weight for each input variable) is used to calculate the distance between the test example and each one of the examples in the training set using a weighted average. This method guarantees that the similarity is measured weighting differently the attributes according to their relevance to the output variable.

The number of similar examples k is an input parameter when we use the knn method. When the CART approach is used, the number of similar examples has no upper limit. To overcome this limitation, we use the knn with the HEOM distance on the members of the leaf node where the test example falls. Doing this, the number of similar examples that are selected is the minimum between the number of similar examples in the leaf node where the test example falls and the input parameter k. When $k = \infty$ all the elements of the selected leaf node are used. Since the CART tree is built selecting the split that minimizes the sum of the output variable's variances of the new two groups formed, the examples that fall in the same leaf node have reduced variance in the output, i.e., the examples in the same leaf node are similar according to the outputs.

The use of CART in the context of obtaining similar data for ensemble learning is firstly described in [Moreira *et al.*, 2007]. In this work some experiments are done comparing the CART approach to select similar data to three other methods, namely, a version of the kd-tree named approximate nearest neighbor algorithm [Arya *et al.*, 1998], and knn with two different distance measures namely the Euclidean distance and the feature weighted distance using the RReliefF algorithm to obtain the weights. The kd-tree [Bentley, 1975] is a generalization of the binary search tree. It is particularly suited for high-dimensional data sets. The approximate nearest neighbor algorithm differs from the original kd-tree because instead of returning the k nearest neighbors, it returns an approximate number of k, more precisely k_1 nearest neighbors where $k_1 \leq k$. This approach reduces meaningfully the computational cost with a low loss in accuracy. Experiments were done on five regression data sets using a four sized ensemble. The approach using CART for similar data searching uses all the examples from the selected leaf node as the similar data set. This can explain, at least partially, the not very exciting results of this approach. The data sets where the CART approach obtained better results were the ones in which the best value of k for the knn approach was higher. The use of the CART approach can have a smoother effect when the best similar data size is lower. This was the motivation to use the HEOM distance on the examples from the selected leaf node.

7.3.2 Selecting the models and combining their predictions

To accomplish both tasks 2 and 4 we use four different methods:

- selection of the best model on similar data (Best): the best model is the one with minimum sum of the squared errors on the similar data set.

- forward selection with replacement (FSwR): at each iteration one selects the model that, when included in the set of models already selected, maximizes the error reduction for the selected similar data, using the simple average as combination method (task 4). It stops when the simple average of the predictions using the selected models does not improve further on the validation set. The same model can be selected more than once. This approach has been suggested by Caruana et al. in a different context [Caruana *et al.*, 2004]. It is important to note that, even if the combination method seems to be the simple average, in practice it is not, because when the same model is selected more than once, the weight of each model depends on the number of times this model is selected.

- Dynamic weighting (DW), already mentioned in Sect. 6.4.2: it uses the vector w of weights and the matrix sqe with the squared error of the models for each similar data point, in order to obtain the wm weight for each model [Puuronen *et al.*, 1999; Rooney *et al.*, 2004] according to,

$$w_i = \frac{\frac{1}{dist_i}}{\sum_{i1=1}^{I}\left(\frac{1}{dist_{i1}}\right)}, \tag{7.1}$$

$$wm_k = \frac{\frac{1}{\sqrt{\sum_{i=1}^{I}(w_i * sqe_{i,k})}}}{\sum_{k1=1}^{K}\left(\frac{1}{\sqrt{\sum_{i=1}^{I}(w_i * sqe_{i,k1})}}\right)}, \tag{7.2}$$

  where i and $i1$ are the example's indexes, k and $k1$ are the model's indexes, K is the ensemble size, $dist$ is the vector of distances of the similar data examples to the input example and I is the number of examples, i.e., the size of the similar data set. In order to better understand the weights equation, lets read $1/dist$ as a measure of similarity. In this case, w_i is the percentage of similarity due to example i.

- Dynamic weighting with selection (DWS) has a previous step with respect to DW. It selects just the models with a mean squared error not higher than a percentage threshold related to the best model. A similar approach is already known [Tsymbal and Puuronen, 2000; Rooney *et al.*, 2004] but setting the threshold to 50%. We tested 5 different values for the threshold: 10%, 30%, 50%, 70% and 90%.

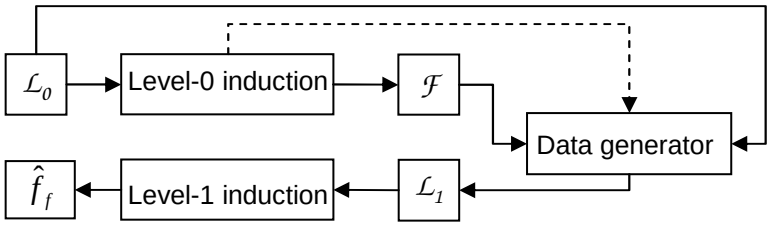

Figure 7.3: Stacked generalization meta-model.

7.4 The experimental setup

The experimental setup is presented at this stage of the chapter because only in Sect. 7.5 we use an ensemble learning framework. The experiments described until now in this chapter, namely the ones on ensemble pruning (Sect. 7.2), do not use this framework.

To better understand the particular experimental setup we have used, we firstly describe the stacked generalization framework, also called stacking [Wolpert, 1992]. Unfortunately, stacking is used with two different meanings: (1) to name the general framework described next; and (2) to refer a particular method for ensemble integration, as discussed in Sect. 6.4.1.

The stacked generalization framework [Wolpert, 1992], defines a meta-model for ensemble learning. This quite general framework establishes two levels of data. The level 0,

$$\mathcal{L}_0 = \{(\mathbf{x_i}, y_i), i = 1, ..., m\}, \tag{7.3}$$

is the data used to train one or more *a&ps*. In our experimental setup this is done using sliding window as explained in Sect. 5.3. The result is an ensemble ($\mathcal{F}$) of k learned models. The level 1 data is obtained by applying these models on the validation data,

$$\mathcal{L}_1 = \{(\hat{f}_1(\mathbf{x_i}), ..., \hat{f}_k(\mathbf{x_i}), y_i), i = 1, ..., m\}. \tag{7.4}$$

The level 1 of induction uses the level 1 data to generate the final predictor. Using this framework for ensemble learning, the level 1 of induction corresponds to the ensemble integration method. Figure 7.3 (adapted from [Merz, 1998]) presents the described framework.

This framework can be seen as a summary of ensemble methods that use the generation and integration methods sequentially with some limitations as it is discussed next.

Dynamic selection typically splits the data into three different data subsets: the training set to build the k base predictors of the ensemble $\mathcal{F}$; the validation set for assessment of the base predictors' generalization error; and the test set for assessment of the ensemble prediction's generalization error.

We also split the data into three subsets: the training set, the validation set for level 0 and the validation set for level 1. The validation set for level 1 is needed because we are testing different integration functions. However, in

this chapter, we do not use a test set. The final evaluation of the framework designed in this chapter is described in the next chapter. The generalization error obtained with the validation set for level 1 is potentially underestimated.

The experimental setup we use is represented in Fig. 7.4. As it can be observed, predictions are made only at the 65th day. This happens because the training sets for both levels 0 and 1 use the same window size of 30 days, i.e., $2 \times (30 + 2) = 64$. The two additional days for each level are due to the prediction lookahead gap. It is not necessary though both training sets use the same window size. As mentioned before (Sect. 5.3), this subject was not studied in this thesis.

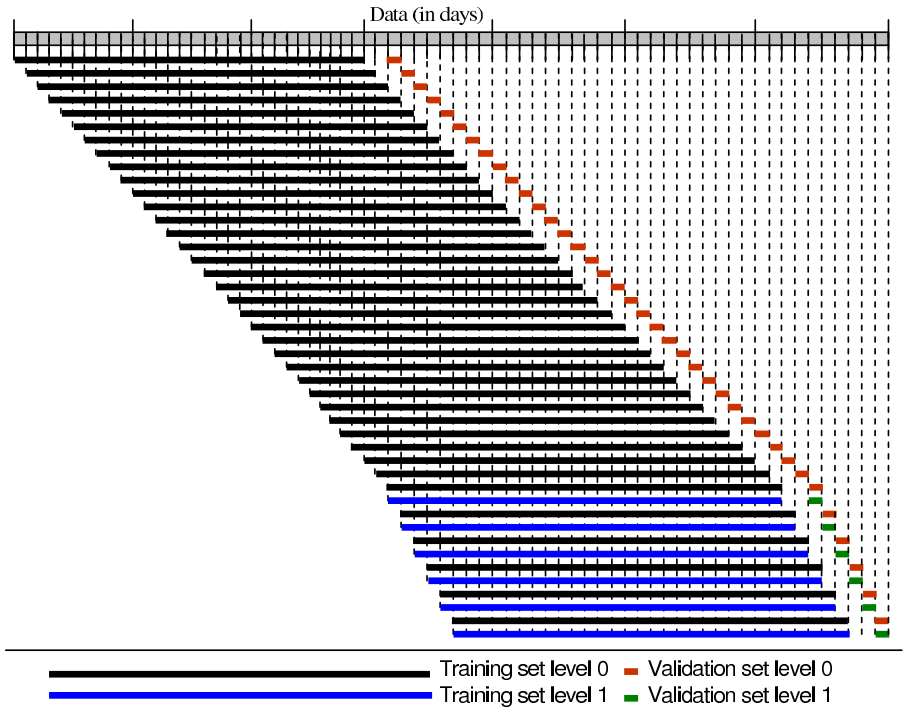

Figure 7.4: Experimental setup for ensemble learning using 70 days of data.

Comparing our approach to Wolpert's framework, level 0 data is given by Eq. 7.3 in both approaches but the level 1 data we use is given by

$$\mathcal{L}_1 = \{(\mathbf{x}_i, \hat{f}_1(\mathbf{x}_i), ..., \hat{f}_k(\mathbf{x}_i), y_i), i = 1, ..., m\}. \tag{7.5}$$

The level 1 data is different because we use a nonconstant weighting function while Wolpert uses a constant one. The definition we use for $\mathcal{L}_1$ is more generic than the one used by Wolpert. The additional $\mathbf{x}_i$ element is necessary for both tasks 1 and 4 of the dynamic approach model.

7.5 The three experiments performed

Three different sets of experiments have been done, each one with a specific goal. These goals are:

- First experiment: to test the three different methods to obtain similar data (Sect. 7.3.1);
- Second experiment: to test how an ensemble selected from a pool learned on a 3 month data set (January to March 2004) performs on a 8 month data set (January to August 2006) by comparison against the performance on a 3 month data set (January to March 2006);
- Third experiment: to test ensembles selected from the pool using two different evaluation functions namely *avg* and *smse* (Sect. 7.2).

Tests on the choice of the similar data size (k) and on methods to select the models and combine their predictions (Sect. 7.3.2) have been done in all the experiments. Additionally, the results for the base learners alone and combined with the constant weighting function simple average (*avg*), are shown. In the first and third experiments we also test ensembles of different sizes.

We notice the use of *avg* in two different situations: (1) as evaluation function of pruning algorithms; and (2) as integration method. Despite this notational repetition, *avg* represents in both situations the simple average and, for that reason, we believe that this repetition do not disturb the lecture of the present section.

7.5.1 First experiment

The first experiment on methods to obtain similar data has the following settings:

- Data set: trips from route 78-1-1, from January the 1st to March the 31st 2006 using the set of input variables {departure time, week day, day of the year and day type};
- Ensemble: selected from pool 130 using the ensemble pruning algorithm Mor06-smse for sizes 5, 10, 15, 20 and 25, where *Mor06* and *smse* represents respectively, the search schema and the evaluation function (see Sect. 7.2);
- Methods for finding similar data: k-nearest neighbor using distance functions based on CART, HEOM and RReliefF approaches for $k \in \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 25, 30\}$;
- Methods for selecting the models to use in prediction: Best, FSwR, DW, DWS-10%, DWS-30%, DWS-50%, DWS-70% and DWS-90%, where FSwR, DW and DWS represent respectively forward selection with replacement, dynamic weighting and dynamic weighting with selection (Sect. 7.3.2).

Analyzing the results (tables from 7.7 to 7.10) for the ensemble with 5 *a&ps*, we can observe that:

Table 7.7: First experiment: the *variation index* for the base learners and simple average using the ensemble of size 5.

a&ps 14	**a&ps 30**	**a&ps 36**	**a&ps 49**	**a&ps 74**	**Avg**
12.47%	11.61%	11.55%	11.41%	11.03%	10.05%

Table 7.8: First experiment: the *variation index* for the knn-CART using the ensemble of size 5.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
30	11.54%	11.36%	9.69%	9.85%	9.64%	9.67%	9.65%	9.66%
25	11.47%	11.44%	9.70%	9.77%	9.66%	9.70%	9.68%	9.66%
20	11.59%	11.46%	9.71%	9.87%	9.68%	9.74%	9.72%	9.69%
18	11.61%	11.44%	9.72%	9.82%	9.70%	9.75%	9.72%	9.69%
16	11.50%	11.46%	9.72%	9.90%	9.73%	9.77%	9.73%	9.71%
14	11.62%	11.54%	9.72%	9.93%	9.73%	9.79%	9.73%	9.71%
12	11.66%	11.56%	9.73%	9.88%	9.80%	9.78%	9.74%	9.72%
10	11.61%	11.55%	9.73%	9.94%	9.83%	9.80%	9.75%	9.75%
8	11.66%	11.59%	9.73%	9.93%	9.84%	9.79%	9.76%	9.75%
6	11.78%	11.68%	9.74%	10.08%	9.94%	9.91%	9.82%	9.80%
4	10.20%	10.14%	9.73%	10.18%	10.04%	9.92%	9.87%	9.88%
2	10.36%	10.23%	9.80%	10.30%	10.20%	10.11%	10.02%	9.97%

Table 7.9: First experiment: the *variation index* for the knn-HEOM using the ensemble of size 5.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
30	11.55%	10.83%	10.10%	10.80%	10.69%	10.35%	10.24%	10.17%
25	10.92%	10.67%	10.09%	10.77%	10.58%	10.36%	10.23%	10.17%
20	10.90%	10.45%	10.08%	10.52%	10.50%	10.31%	10.19%	10.18%
18	11.02%	10.49%	10.07%	10.51%	10.51%	10.31%	10.21%	10.16%
16	10.63%	10.44%	10.07%	10.47%	10.49%	10.32%	10.20%	10.16%
14	10.45%	10.43%	10.06%	10.48%	10.43%	10.35%	10.19%	10.16%
12	10.37%	10.30%	10.06%	10.33%	10.46%	10.32%	10.20%	10.16%
10	10.51%	10.28%	10.04%	10.26%	10.38%	10.33%	10.23%	10.19%
8	10.47%	10.29%	10.03%	10.28%	10.42%	10.36%	10.25%	10.15%
6	10.70%	10.60%	10.03%	10.57%	10.48%	10.42%	10.29%	10.19%
4	10.94%	10.88%	10.02%	10.57%	10.46%	10.40%	10.36%	10.28%
2	11.25%	11.00%	10.10%	10.94%	10.87%	10.73%	10.62%	10.49%

Table 7.10: First experiment: the *variation index* for the knn-RReliefF using the ensemble of size 5.

k	Best	FSwR	DWS 10%	DWS 30%	DWS 50%	DWS 70%	DWS 90%	DW
30	11.46%	10.54%	10.26%	10.89%	10.52%	10.39%	10.35%	10.29%
25	11.07%	10.62%	10.17%	10.95%	10.52%	10.40%	10.37%	10.32%
20	11.31%	10.69%	10.10%	10.86%	10.35%	10.16%	10.13%	10.12%
18	11.31%	10.62%	10.11%	10.96%	10.57%	10.34%	10.29%	10.27%
16	11.35%	10.73%	10.08%	11.14%	10.53%	10.37%	10.28%	10.24%
14	11.29%	10.69%	10.05%	11.09%	10.51%	10.35%	10.27%	10.26%
12	11.09%	10.66%	10.05%	10.93%	10.50%	10.33%	10.24%	10.20%
10	11.25%	10.76%	10.05%	10.98%	10.41%	10.30%	10.21%	10.10%
8	11.26%	10.75%	10.02%	10.81%	10.46%	10.22%	10.16%	10.07%
6	11.24%	10.87%	10.03%	10.94%	10.54%	10.41%	10.24%	10.13%
4	11.35%	11.13%	10.07%	11.06%	10.83%	10.52%	10.36%	10.25%
2	11.52%	11.47%	10.13%	11.32%	11.02%	10.80%	10.63%	10.57%

- Globally, knn-CART is the best of the three tested methods for obtaining similar data.
- Knn-CART performs better than the other two methods for task 1, when DWS or DW are used, whatever is the value of k.
- When DWS or DW are used, the best value of k depends on the method used to obtain similar data. For knn-CART, the accuracy increases as the value of k augments. For knn-HEOM, it is around 10, while for knn-RReliefF is a little bit erratic.
- FSwR performs better than Best in all the situations.
- Just knn-CART beats the constant weighting function *avg*.

These results are not very different when larger ensembles are used (Appendix C.1). However, the main result is that knn-CART is always the best method for obtaining similar data whatever is the size of the ensemble. For knn-CART it is also apparent that the accuracy improves for larger similar data sets, i.e., for larger k values. DS and DWS also compare favorably to Best, FSwR and the simple average. However, it is not very clear what is the best setting for dynamic weighting with or without selection (DWS and DW). It is also important to understand the influence of the ensemble size. The analysis of Fig. 7.5 gives some insights on these issues. Two groups of methods are apparent: the one for lower values of the threshold for DWS and the other ones with larger thresholds (including DW, since DW = DWS-100%). The first group is more unstable than the second one despite the fact that the first group gets the best overall result. Anyway, it is not obvious which is the best ensemble size. Moreover, it is not guaranteed that the best ensemble for a three month period is also the best one for a larger period. It seems more cautious to use at least one complete seasonal-cycle of the seasonality with the widest amplitude, in this case, one year. Since the pool 130 (Sect. 7.1) was trained and tested using just 3 month of data, from January the 1st to March the 31st of 2004, and the first

experiment uses data from an equivalent period, i.e., from January the 1st to March the 31st of 2006, the question is to know how the ensembles selected from this pool behave on different months. The second experiment aims to clarify this issue.

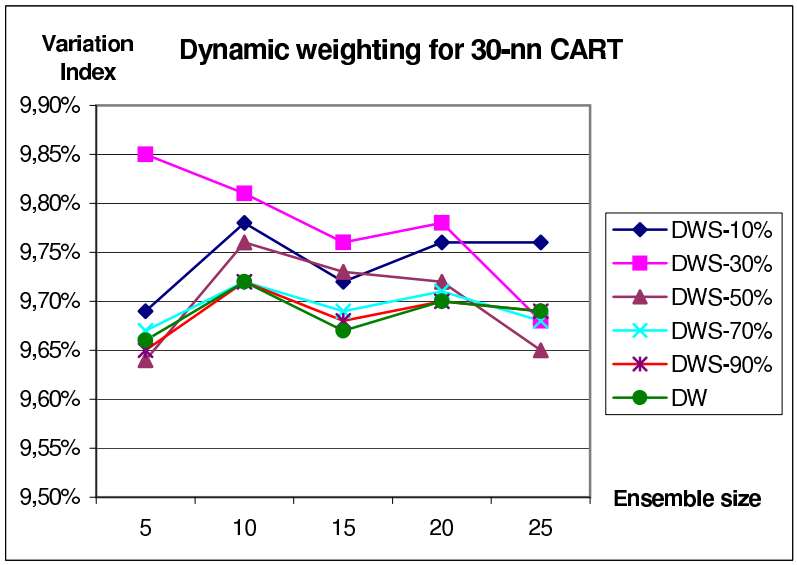

Figure 7.5: DW and DWS for 30-nn CART for different ensembles.

7.5.2 Second experiment

The second experiment has the following settings:

- Data set: trips from route 78-1-1, from January the 1st to August the 31st of 2006 using the set of input variables {departure time, week day, day of the year and day type};
- Ensemble: selected from pool 130 using Mor06-smse for size 5;
- Methods for finding similar data: knn for $k \in \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 25, 30\}$ using the distance function based on CART;
- Methods for selecting the models to use in prediction: Best, FSwR, DW, DWS-10%, DWS-30%, DWS-50%, DWS-70% and DWS-90%.

The main differences for the settings used in the first experiment are: a larger data set; the use of just one method, the knn-CART which obtained the best results in the previous experiments, to obtain similar data; and the use of just one ensemble, the one of size 5.

The analysis of the results (tables 7.11 and 7.12) is done by comparing them to the results obtained using the same setting but on a data set with the same 3 months used in the data set where the pool was trained and tested, but from a different year. These results were already presented in tables 7.7 and 7.8. The difference among those tables are not due to different performance of the

Table 7.11: Second experiment: the *variation index* for the base learners and simple average using the 8 month data set.

a&ps 14	**a&ps 30**	**a&ps 36**	**a&ps 49**	**a&ps 74**	**Avg**
12.84%	13.05%	11.72%	12.14%	16.73%	11.08%

Table 7.12: Second experiment: the *variation index* for the knn-CART using the 8 month data set.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
30	12.43%	11.99%	10.86%	11.40%	10.97%	10.82%	10.80%	10.83%
25	12.45%	12.15%	10.87%	11.47%	10.98%	10.86%	10.83%	10.84%
20	12.44%	12.07%	10.88%	11.48%	11.06%	10.91%	10.88%	10.86%
18	12.37%	12.12%	10.89%	11.56%	11.15%	10.95%	10.92%	10.91%
16	12.37%	12.13%	10.89%	11.64%	11.18%	10.93%	10.96%	10.95%
14	12.52%	12.27%	10.90%	11.70%	11.24%	10.99%	10.99%	10.96%
12	12.46%	12.06%	10.91%	11.71%	11.21%	11.01%	10.98%	10.96%
10	12.47%	12.09%	10.91%	11.78%	11.26%	11.04%	11.01%	10.96%
8	12.40%	12.02%	10.91%	11.79%	11.26%	11.06%	10.99%	10.98%
6	12.49%	12.14%	10.91%	11.76%	11.27%	11.15%	11.00%	10.96%
4	12.23%	11.95%	10.91%	11.91%	11.40%	11.26%	11.07%	11.00%
2	12.52%	12.27%	10.99%	12.34%	11.81%	11.64%	11.49%	11.36%

ensemble methods but to the increase of the error of the base learners when using the 8 month data set. It seems wiser to train and test the ensemble in a data set representative of the population, as understood by the statistical community, i.e., in a time varying problem, using data that cover all the seasons of the largest seasonality. For TTP, the use of one year of data to train the pool of predictors could, potentially, avoid the use of *a&ps* that degrade strongly the performance when used in different seasons (as it happens, for instance, with the *a&ps* 74 from tables 7.7 and 7.11).

7.5.3 Third experiment

The third experiment on evaluation functions for ensemble pruning has the following settings:

- Data set: trips from route 78-1-1, from January the 1st to March the 31st of 2006 using the set of input variables {departure time, week day, day of the year and day type};
- Ensemble: selected from pool 130 using Mor06-smse and Mor06-avg for sizes 5, 10, 15, 20 and 25;
- Methods for finding similar data: knn for $k \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, \infty\}$ using the distance function based on CART (knn-CART);

Table 7.13: Third experiment: the *variation index* for the knn-CART on the 5 size ensemble obtained using Mor06-smse.

k	Best	FSwR	DWS 10%	DWS 30%	DWS 50%	DWS 70%	DWS 90%	DW
∞	11.63%	11.40%	9.70%	9.67%	9.58%	9.66%	9.67%	9.63%
200	11.50%	11.38%	9.70%	9.69%	9.61%	9.67%	9.67%	9.63%
100	11.60%	11.43%	9.70%	9.71%	9.61%	9.66%	9.67%	9.65%
90	11.63%	11.43%	9.70%	9.65%	9.62%	9.66%	9.66%	9.65%
80	11.62%	11.42%	9.70%	9.71%	9.64%	9.66%	9.66%	9.65%
70	11.66%	11.51%	9.70%	9.66%	9.64%	9.65%	9.66%	9.65%
60	11.66%	11.47%	9.70%	9.68%	9.61%	9.66%	9.66%	9.65%
50	11.66%	11.43%	9.70%	9.72%	9.61%	9.68%	9.66%	9.65%
40	11.52%	11.36%	9.70%	9.70%	9.63%	9.67%	9.66%	9.65%
30	11.54%	11.36%	9.69%	9.85%	9.64%	9.67%	9.65%	9.66%
20	11.59%	11.46%	9.71%	9.87%	9.68%	9.74%	9.72%	9.69%
10	11.61%	11.55%	9.73%	9.94%	9.83%	9.80%	9.75%	9.75%

Table 7.14: Third experiment: the *variation index* for the base learners and simple average on the 5 size ensemble obtained using Mor06-avg.

a&ps 36	a&ps 72	a&ps 75	a&ps 96	a&ps 120	Avg
11.56%	12.70%	11.02%	10.91%	10.87%	9.92%

- Methods for selecting the models to use in prediction: Best, FSwR, DW, DWS-10%, DWS-30%, DWS-50%, DWS-70% and DWS-90%.

The main differences for the settings used in the second experiment are: the use of a 3 month data set again; the use of a different set of larger values of k (in the previous experiments the best result of knn-CART was obtained for the larger k value); and the use of two different ensembles (each one obtained using a different evaluation measure in the pruning algorithm).

The evaluation functions used in the pruning algorithms (Sect. 7.2) were chosen expecting that they could get the most of the Best and simple average (*avg*) integration methods. The third experiment evaluates how the ensembles obtained using each one of these evaluation functions perform for the different integration functions being used. Results are shown in tables 7.7 and from 7.13 to 7.15 for ensembles of size 5. The results for larger ensembles are in Appendix C.2. However, in tables 7.16 and 7.17 the results for $k = \infty$ for the different ensemble sizes are presented.

From the analysis of the results we observe that:

- The integration functions Best and FSwR perform better when using the Mor06-avg ensemble.
- The performance of Best and FSwR improve when using larger k in the selection of similar data for the Mor06-avg ensemble.
- The behavior of the integration functions when using ensembles of different

Table 7.15: Third experiment: the *variation index* for the knn-CART on the 5 size ensemble obtained using Mor06-avg.

k	Best	FSwR	DWS 10%	DWS 30%	DWS 50%	DWS 70%	DWS 90%	DW
∞	10.12%	9.91%	9.90%	9.61%	9.80%	9.92%	9.94%	9.95%
200	10.13%	9.92%	9.90%	9.60%	9.81%	9.92%	9.94%	9.95%
100	10.12%	9.95%	9.91%	9.68%	9.85%	9.91%	9.95%	9.95%
90	10.16%	9.97%	9.91%	9.68%	9.86%	9.91%	9.95%	9.95%
80	10.36%	10.03%	9.91%	9.66%	9.86%	9.92%	9.95%	9.94%
70	10.36%	10.05%	9.91%	9.74%	9.82%	9.93%	9.94%	9.94%
60	10.36%	10.14%	9.92%	9.89%	9.82%	9.94%	9.95%	9.94%
50	10.49%	10.22%	9.92%	9.92%	9.83%	9.94%	9.96%	9.95%
40	10.76%	10.51%	9.92%	9.90%	9.84%	9.95%	9.93%	9.93%
30	10.98%	10.87%	9.93%	9.98%	9.81%	9.95%	9.94%	9.92%
20	11.03%	10.79%	9.93%	10.14%	9.85%	9.92%	9.96%	9.95%
10	11.06%	10.92%	9.95%	10.37%	9.99%	9.98%	10.02%	10.00%

Table 7.16: Third experiment: the *variation index* for the knn-CART with $k = \infty$ on ensembles obtained using Mor06-smse.

Size	Avg	Best	FSwR	DWS 10%	DWS 30%	DWS 50%	DWS 70%	DWS 90%	DW
5	10.05%	11.63%	11.40%	9.70%	9.67%	9.58%	9.66%	9.67%	9.63%
10	10.09%	11.74%	11.50%	9.78%	9.78%	9.61%	9.68%	9.72%	9.71%
15	9.98%	11.74%	11.73%	9.68%	9.85%	9.55%	9.59%	9.63%	9.63%
20	10.03%	11.67%	11.70%	9.75%	9.95%	9.61%	9.66%	9.70%	9.70%
25	9.95%	11.41%	11.20%	9.76%	9.54%	9.58%	9.65%	9.67%	9.68%

Table 7.17: Third experiment: the *variation index* for the knn-CART with $k = \infty$ on ensembles obtained using Mor06-avg.

Size	Avg	Best	FSwR	DWS 10%	DWS 30%	DWS 50%	DWS 70%	DWS 90%	DW
5	9.92%	10.12%	9.91%	9.90%	9.61%	9.80%	9.92%	9.94%	9.95%
10	9.68%	9.78%	9.55%	9.62%	9.48%	9.54%	9.62%	9.63%	9.63%
15	9.59%	9.69%	9.50%	9.55%	9.41%	9.46%	9.52%	9.56%	9.55%
20	9.62%	9.76%	9.54%	9.60%	9.44%	9.50%	9.59%	9.61%	9.61%
25	9.56%	9.67%	9.50%	9.55%	9.41%	9.46%	9.53%	9.55%	9.55%

sizes is different for Mor06-avg and Mor06-smse ensembles. It seems that Mor06-avg ensembles take more advantage from the existence of more models (tables 7.16 and 7.17).

- Globally, the best result is obtained using the Mor06-avg ensemble.
- Mor06-smse ensembles do not seem to be valuable whatever is the ensemble integration function.
- Excepting Best, nonconstant weighting functions improve slightly but consistently constant weighting functions when Mor06-avg ensembles are used.

7.6 Final comments

The final phase of Part III of this thesis is to evaluate what is the best method for TTP three days ahead. This is the subject of the next chapter. To do this, it is necessary to test the ensemble learning framework, as well as other approaches, using a set of different routes. To better accomplish this task, we describe the lessons from the experiments carried on along this chapter.

The pool generation described in Sect 7.1 benefits from the many experiments done using the same data set. In fact, the pool generation was done in Chap. 5 when experiments with different algorithms were carried on. The pool generation described in Sect. 7.1 is no more than a previous pruning step using a set of empirical criterion. All the experiments described until now use data from route 78-1-1. It is not guaranteed that a good pool for this route is good for a different one. The pool 130 can be a first starting set but it is important to inspect the pool and add new *a&ps* if necessary.

It would also be important to assure that the data used for the pool generation guarantees one year of data for validation. This remark results directly from the second experiment (Sect. 7.5.2) on ensemble learning.

Ensemble pruning should use the *avg* evaluation measure (Sect. 7.5.3). The forward searching method seems to be a good option because it is much faster than Mor06 with a small loss in accuracy. Furthermore it is not sure that this difference is statistically meaningful, but even if it is, the difference is really minimum. Concerning the optimal size of the ensemble, the results are a little bit erratic (Fig. 7.5). Furthermore, it is not expected that using a data set for the pool generation with more months, the behavior of Fig. 7.5 would be similar. It seems wise to test different ensemble sizes when using different routes, especially if the data set used for the pool generation covers different seasons of the seasonality with the widest amplitude.

On ensemble integration, the use of knn-CART (Sect. 7.5.1) with $k = \infty$ (Sect. 7.5.3) is a natural choice to obtain the similar data set. Moreover, the results using knn-CART are theoretically founded by the variance reduction at each split in the CART recursive partitioning algorithm [Breiman *et al.*, 1984]. However, the method to accomplish tasks 2 and 4, i.e., selection and combination of the base learners, is expected to have an erratic behavior for different sizes of the ensemble (Fig. 7.5), even if the differences are not too large. Furthermore, it seems wiser, when using data sets covering different months, to test at least DW and DWS with different settings (Sect. 7.5.2).

The large number of experiments done on the same data (both in Chap. 5 and in the present one) reduce the confidence on good results since they may have been a consequence of oversearching. This problem is well described in [Salzberg, 1997]. Even if this happens, it will be detected in the following chapter because the chosen ensemble approach will be tested using new data sets from six different routes. Furthermore, each experiment uses different training and test sets according to the sliding window method, reducing this possibility. It is also important to note that Salzberg assumes that experiments are randomly chosen. This is true in the process of parameter tuning described in Sect. 5.6, but in the remaining experiments there is a sense of causality for the choice of each particular experiment, i.e. they were not chosen randomly.

The analysis of the results does not use statistical methods in order to obtain statistically meaningful conclusions. This option was already taken in Chap. 5. The goal of these experiments is to choose one method for future deployment on different routes. Even if a method is not statistically different from another one, one of the methods must be chosen, unless it is expected that this method will have a different behavior on different data sets. In this case, the possible methods for this particular task should be tested for each different data set.

Chapter 8

Evaluating different approaches

The purpose of this chapter is to evaluate the different methods for travel time prediction three days ahead. We start by describing the routes/lines (a line is a set of routes, as described in Sect. 2.1.1) used in the tests (Sect. 8.1). Afterwards, we present the methods to be compared (Sect. 8.2) followed by the description of the experimental setups and first results for each of these methods (Sect. 8.3). The results obtained are presented and statistically evaluated. A discussion on the advantages and disadvantages of the most promising methods for travel time prediction three days ahead is carried out (Sect. 8.4). Finally, we discuss, in a prospective way, how TTP three days ahead could be used in practice (Sect. 8.5).

8.1 Tested routes/lines

We have selected the F3 rostering group, according to the concept of rostering group as defined in Sect. 2.1.2, for the experiments (plan defined in Sect. 2.3). This group was chosen because it is small and its lines are very different from each other. It has four lines: two circular ones (lines 300 and 301) with just one route each, and two lines (205 and 505) with two routes (go and return), totalizing 6 routes. The circular lines are typical urban lines. All four lines go past Hospital de São João, an important bus and metro interface, where there is a relief point for the crew.

Line 205 runs along almost the whole of a peripheral road that roughly defines, together with Douro river and the sea, the limits of Oporto municipality. Crossing some of the main entrances of the city, it serves two important mass transport interfaces, the referred Hospital de São João and the main railway station at Oporto, the Campanhã station, where there is also a metro station. This line is the first to be presented in order to facilitate the reading of the maps.

Lines 300 and 301 are roughly equal but in opposite directions. That is why just one of them is presented (Fig 8.2). Both are arterial urban circular lines that connect the city center, where there is the second railway station (São Bento) and the metro, to Hospital São João.

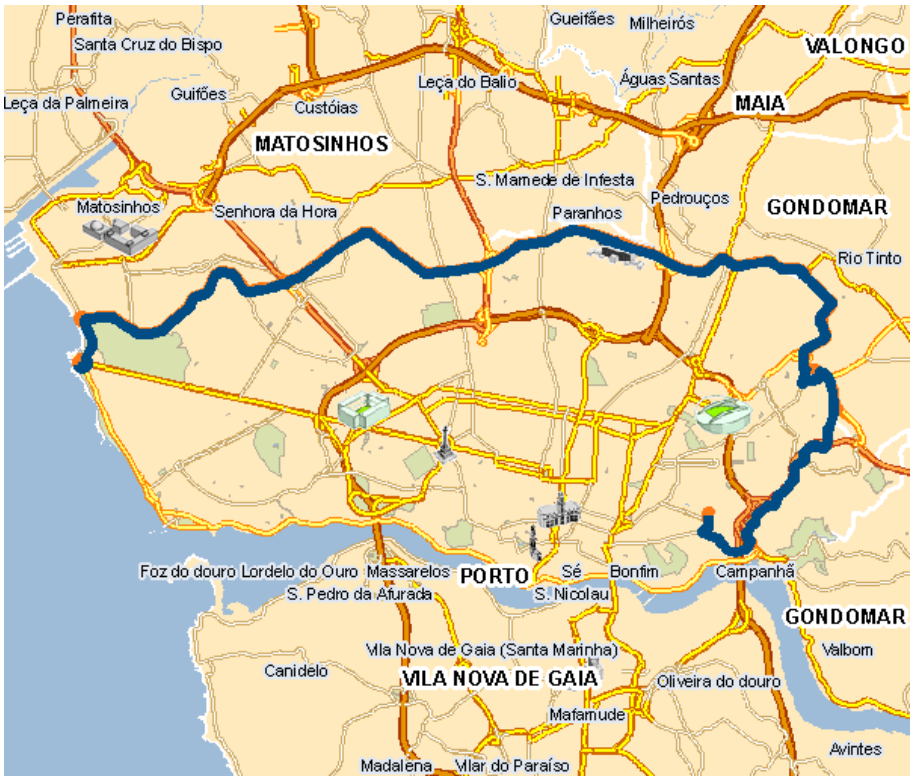

Figure 8.1: Line 205.

Line 505 connects Oporto to the neighboring town of Matosinhos, where there is a sea port. The area served by the line is typically suburban. This line does not go to Oporto city center. It stops at the above mentioned peripheral road at Hospital São João. Despite the fact that this line serves an area quite close to Oporto, it is neither an urban nor a commuter line.

8.2 Tested methods

From the experiments done in chapters 5 and 7, the following methods were chosen for comparison with the method currently in use at STCP:

- The baseline method uses as prediction the travel time of the nearest past example (Sect. 5.4);
- The expert-based method selects past data similar to the test example using a more complete similarity measure than the one used in the baseline method, and uses this data to obtain the prediction by averaging their travel time (Sect. 5.5);
- The single best *a&ps* from the pool used in the ensemble generation task (we used pool 128, Appendix D.1);

Figure 8.2: Line 300.

- An ensemble approach using nonconstant weighting functions following the lessons learned (Sect. 7.6) from the experiments on ensemble learning.

8.3 Experimental setups and preliminary results

In order to compare the different methods, it is necessary to guarantee, as far as possible, comparable experimental setups. In fact, they cannot be fully guaranteed due to the different nature of the methods. The main concern regarding the experiments described in the present chapter was to guarantee both adequate model tuning and the same test set for all the methods. Considering that the ensemble approach is the most demanding method (of the four tested) in terms of data, we begin the description of the experimental setup for this case, followed by the single best *a&ps*, the expert-based and the baseline methods. Before the experimental setups, we briefly describe the data available for the tests.

8.3.1 Available data

Regardless of the routes used to collect the data (see Sect. 8.1), the period covered by the data is important to design the experimental setup. These experiments were executed at the beginning of April 2008. Due to the complete

Figure 8.3: Line 505.

redesign of the STCP bus network (the new network began to be used on January the 1st 2007), there were, for each route, 15 months of data.

8.3.2 Experimental setup for the ensemble method

We describe again the necessary data for the experimental setup of the ensemble approaches, according to the stacked generalization framework [Wolpert, 1992] described in Sect. 7.4:

- A training set for level 0 of induction;
- A validation set to assess the generalization error of the base learners.
- A second validation set to tune parameters of the ensemble framework (this set is not mentioned in the Wolpert's paper because the level 1 of induction used there does not have parameters to tune).
- The test set to assess the final generalization error.

Without restrictions on the amount of available data, the ideal size for the two validation sets and the test set would be one year of data each in order to guarantee that the sample is representative of the population, since the seasonality of the year is the one with the widest seasonal-cycle. Using sliding window (Fig. 7.4), the following would be: (1) 30 days for the initial training

set; (2) at least 2 days plus to keep 3 days of distance for the prediction data; (3) 365 days for the first validation set; (4) 365 days for the second validation set; and (5) 365 days for the test set. This would give around 37 months of data. This estimate is obviously very conservative. With just 15 months of data, and knowing that the first 32 days are necessary for the initial iteration of the sliding window process (the aforementioned sets 1 and 2), the remaining data (around 14 months) should be split into three parts: two validation sets and one test set. We have decided to use approximately 8 months for the first validation set, 3 for the second validation set and the remaining 3 for the test set. The reason to do so was three-fold: (1) the sensitivity of the base learners to level 0 data (Sect. 7.5.2), (2) the lower sensitivity of the ensemble approach to its tuning task, namely to the choice of both the ensemble size and the integration method, and (3) the option to give priority to the primary steps in the ensemble framework because the test set will grow over time and new data can be tested without it being necessary to repeat these primary steps. Following closely the considerations made in Sect. 7.6, the experiments using the ensemble approach have the following settings:

- Pool generation:
  - Data set: trips from January the 1st to September the 29th 2007 using the set of input variables {departure time, week day, day of the year and day type};
  - Pool generated: pool 128 (Appendix D.1);
  - Experimental setup: as described in Sect. 5.3.
- Ensemble pruning:
  - Pruning algorithm: FSS-avg, i.e., the forward search algorithm using the simple average for evaluation (Fig. 7.1);
  - Sizes of the selected ensembles: $Size \in \{5, 10, 15, 20, 25\}$.
- Ensemble integration:
  - Data set: trips from July the 28th to December the 31th 2007 using the set of input variables {departure time, week day, day of the year and day type};
  - Ensemble: selected from pool 128 using FSS-avg for $Size \in \{5, 10, 15, 20, 25\}$;
  - Experimental setup: as described in Sect. 7.4;
  - Tasks of the dynamic approach (Sect. 6.4.2):
    - Obtaining similar data: knn-CART with $k = \infty$ (see Sect. 7.3.1 and 7.6);
    - Selecting the models and combining their predictions: Best, FSwR, DW, DWS-10%, DWS-30%, DWS-50%, DWS-70% and DWS-90%, where FSwR means forward selection with replacement, DW means dynamic weighting and DWS means dynamic weighting with selection (Sect. 7.3.2 and 7.6).
- The final test:

Table 8.1: The best setting and respective *variation index* for each route using the ensemble approach on the second validation set.

route	size	integration function	var. index
205-1-1	5	DWS-70%	8.87%
205-1-2	5	DWS-50%	9.28%
300-1-1	5	DWS-90%	7.00%
301-1-1	5	DWS-90%	10.30%
505-1-1	5	DWS-50%	12.19%
505-1-2	5	DWS-30%	10.07%

- Data set: trips from October the 31st 2007 to March the 31th 2008 using the set of input variables {departure time, week day, day of the year and day type};
- Ensemble: selected from pool 128 using FSS-avg and using the setting with best results in the experiment on ensemble integration;
- Experimental setup: as described in Sect. 7.4;
- Tasks of the dynamic approach (Sect. 6.4.2):
  - Task 1: knn-CART with $k = \infty$ (see Sect. 7.6);
  - Tasks 2 and 4: using the integration function with best results in the experiment on ensemble integration.

In table 8.1 the best settings of ensemble integration in the second validation set are presented. The complete preliminary results of the ensemble approach are in Appendix D.

8.3.3 Experimental setup for the single best *a&ps*

The necessary data for the selection of the best *a&ps* does not require a second validation set as in the previous case. It requires:

- A training set to generate the pool of models;
- A validation set to assess the generalization error of the base learners;
- The test set to assess the final generalization error.

For this case, the necessary data would be: (1) 30 days for the initial training set size; (2) at least 2 days plus to keep 3 days of distance for the prediction data; (3) 365 days for the validation set; and (4) 91 days for the test set (to guarantee the use of the same test set for all the four tested methods, since it was the test set used in the ensemble approach). This would require around 16 months of data where there is just 15. In order to use the same test set as for the ensemble method, the only option would be to reduce the validation set to around 11 months. However, in order to computationally benefit from the experiments made for the ensemble method, we have decided to use only 9 months of data for validation. This guarantees that the ensemble generation process described in the previous section could be used for the selection of the single best *a&ps*.

The settings for the experiments are:

Table 8.2: Best *a&ps* and respective *variation index* (VI) for each route on the validation set.

Route	Algorithm	ES	WDDT	P1	P2	P3	P4	VI
205-1-1	svm linear	ln	Sym.	0.0625	0.9			8.89%
205-1-2	ppr spline	all	Num.	1	3	32		10.19%
300-1-1	RF	ed	Sym.	3				7.89%
301-1-1	svm linear	ln	Sym.	0.0625	0.3			12.28%
505-1-1	svm linear	ln	Sym.	0.0625	0.3			12.66%
505-1-2	svm sigmoid	ln	Sym.	16000	0.6	0.000002	-2	10.96%

- Pool generation:
  - Data set: trips from January the 1st to September the 29th 2007 using the set of input variables {departure time, week day, day of the year and day type};
  - Pool generated: pool 128 (Appendix D.1);
  - Experimental setup: as described in Sect. 5.3.
- The final test:
  - Data set: trips from November the 30th 2007 to March the 31st 2008 using the set of input variables {departure time, week day, day of the year and day type};
  - *a&ps*: the one with minimum generalization error (*mse*) from pool 128;
  - Experimental setup: as described in Sect. 5.3.

Table 8.2 presents the best *a&ps* for each route. ES identifies the used example selection method, as described in Sect. 5.7.1. WDDT identifies the data type used for the variable weekday, as discussed in Sect. 5.7.2. The parameters P1, P2, P3 and P4 should be read as follows:

- For SVM: P1 = C, P2 = ν, P3 = γ and P4 = coef0;
- For RF: P1 = mtry;
- For PPR-supsmu: P1 = nterms, P2 = optlevel, P3 = bass and P4 = span;
- For PPR-spline: P1 = nterms, P2 = optlevel and P3 = df;
- For PPR-gcvspline: P1 = nterms, P2 = optlevel and P3 = gcvpen.

8.3.4 Experimental setup for the expert-based method

The expert-based method requires the same sets as the single best predictor. The options on the size of each one of the sets are identical. However, in the case of the expert-based method, we used an 11-month validation set. Consequently, the experiments using the expert-based method have the following settings:

Table 8.3: Best parameter sets for the expert-based method and respective *variation index* for each route on the validation set.

route	min.ex	margin	max.incr	var. index
205-1-1	24	600	7	8.79%
205-1-2	24	600	7	9.24%
300-1-1	21	600	7	6.91%
301-1-1	21	1200	3	8.81%
505-1-1	27	300	1	12.10%
505-1-2	15	600	7	9.99%

- Tuning parameter set:
  - Data set: trips from January the 1st to December the 31th 2007;
  - The 125 tested parameter sets were the combination of:
    - $min.ex \in \{15, 18, 21, 24, 27\}$;
    - $margin \in \{300, 600, 900, 1200, 1500\}$;
    - $max.incr \in \{1, 3, 5, 7, 9\}$.
  - Experimental setup: as described in Sect. 5.3.
- The final test:
  - Data set: trips from November the 30th 2007 to March the 31th 2008;
  - Parameter set: the one with minimum generalization error (mse) from the 125 tested;
  - Experimental setup: as described in Sect. 5.3.

The best parameter set in the validation set for each route is presented in table 8.3.

8.3.5 Experimental setup for the baseline method

The baseline method does not have parameters to tune. Consequently, the only experiment is the final test. It has the following settings:

- Test set: trips from November the 30th 2007 to March the 31th 2008;
- Experimental setup: as described in Sect. 5.3.

8.4 Results and Analysis

The results obtained for the five methods using the same test set (from January the 1st 2008 to March the 31th 2008), are presented in table 8.4 and Fig. 8.4. The method in use at STCP, identified by STT (Scheduled Travel Time), uses the timetabled values for the days covered by the test set. The STTs used in these experiments were obtained using the two different approaches as described in Sect. 2.2.2: the old one for Saturdays and Sundays (for all four lines) and for working days (lines 300 and 301); the new one for working days on lines 505

Table 8.4: The *variation index* for different methods on the test set. The methods with statistically meaningful lower error than all the others for $\alpha^* = 0.005$ are in bold-face.

Route	STT	BL	EXP	BST	ENS	n
205-1-1	11.87%	12.14%	9.08%	9.49%	9.00%	5210
205-1-2	12.13%	12.42%	9.33%	10.03%	9.21%	5372
300-1-1	**13.53%**	19.78%	14.06%	16.38%	14.08%	3668
301-1-1	14.06%	18.79%	14.15%	15.18%	14.08%	3671
505-1-1	12.82%	15.83%	11.52%	11.62%	**10.92%**	2161
505-1-2	14.04%	12.95%	9.18%	9.26%	**8.78%**	1547

STT: Scheduled Travel Time; BL: BaseLine predictor; EXP: EXPert-based predictor; BST: single BeST predictor; ENS: ENSemble predictor.

(from January 21st) and 205 (however, the method used to define the STT for line 205 was still under study and, consequently, it still had margin to improve using that approach). Column n identifies the number of trips in the test set.

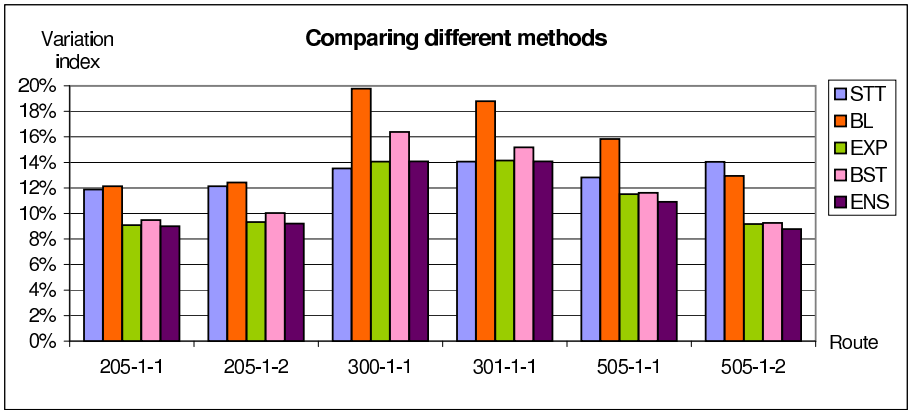

Figure 8.4: The *variation index* for different methods in the test set.

Given the fact that the goal is to compare every pair of methods (this task is named all-pairwise comparisons in the statistical literature [Hsu, 1996; Nakayama, 1997]), a multiple comparison test must be done. In Sect. 4.2.5 we have already discussed the main issues on multiple comparison tests and the difficulties in comparing methods when the instances are not independent. The lack of specific methods for this type of problem has led us to adopt an existing method knowing a priori that the obtained results must be analyzed with caution because the assumption of independence of the examples does not hold.

The method we use is an approximate procedure described in [Hochberg and Tamhane, 1987]. Its use is justified by the fact that its only assumption is that the examples are independent (that assumption also exists in all known alternative methods) and also because any error measure can be used [Feelders and Verkooijen, 1995]. The error measure we use is that used in [Feelders and Verkooijen, 1995], i.e., the square error calculated as $\hat{fe}_i(\mathbf{x}) = \left(\hat{f}_i(\mathbf{x}) - f(\mathbf{x})\right)^2$.

We denote the sample average of $\hat{f}e_i$ as $\overline{\hat{f}}e_i$, the sample variance of $\hat{f}e_i$ as Se_i^2, the sample covariance of $\hat{f}e_i$ and $\hat{f}e_j$ as $Se_{i,j}$, the number of test examples as n, and the number of predictors being compared as k. The method consists of the following definition of $(1-\alpha)\times 100$ simultaneous confidence intervals:

$$\mu e_i - \mu e_j \in \left[\overline{\hat{f}}e_i - \overline{\hat{f}}e_j \pm T_{n-1}^{\alpha^*/2} \times \sqrt{\frac{Se_i^2 + Se_j^2 - 2 \times Se_{i,j}}{n}} \right], (1 \leq i < j \leq k), \tag{8.1}$$

where $\alpha^* = \alpha/k^*$, $k^* = k \times (k-1)/2$, and T_{df}^a is the upper a point of Student's t-distribution with df degrees of freedom [Hochberg and Tamhane, 1987]. The use of α^* instead of α is due to the use of the Bonferroni method in order to correct the multiplicity effect described in Sect. 4.2.5. For $\alpha = 0.05$ and $k = 5$, $k^* = \frac{5\times(5-1)}{2} = 10$, $\alpha^* = 0.05/10 = 0.005$ and $T_{\infty}^{0.005} = 3.17$ [1].

The multiple intervals of confidence are presented in table 8.5 for each of the six routes. Intermediary results can be seen in Appendix E.

The analysis of the intervals of confidence must be made knowing that when both limits of the interval are positive it means that the predictor in the line has a larger error, for $\alpha = 0.05$, than the one in the column. If both limits are negative then it means that the predictor in the column has a larger error, for $\alpha = 0.05$, than the one in the line. If one of the limits is positive and the other is negative then the test is inconclusive for $\alpha = 0.05$.

The results obtained deserve an analysis:

- Results are clearly different for the circular routes (300-1-1 and 301-1-1) and for all the others (205-1-1, 205-1-2, 505-1-1 and 505-1-2). On circular routes the lack of slack times urges the accomplishment of the schedule. Consequently it is expected that for this type of route, the STTs are more accurate than any other method because the control is performed in order to guarantee its accomplishment. For these routes, the use of speed modification and short turning is common (see Sect. 2.2). These orders are communicated from the control center to the drivers by radio. An open question is to know what price is payed by the passengers when the STT are not the most appropriate. For circular routes, the usual measure for passengers' satisfaction, the time they must wait for the bus at the bus stop, does not seem enough. The type of complaints made by the passengers on circular routes has some particularities when compared to other routes, namely complaints about the low speed of the trips. The use of *mse* alone does not capture this problem, i.e., the inefficiency of the planning. How to evaluate the quality of the planning is an open question for future research.

- Comparing these results with the preliminary ones (Sect. 8.3), i.e., the ones obtained using the validation sets, routes 300-1-1 and 301-1-1 present surprisingly larger *variation indexes* in the test sets. To better understand how the error behaves along time in the test sets, weekly moving *variation indexes* for each one of the routes are presented in figures from 8.5 to 8.10. It is interesting to observe that the relative accuracy between the different

[1] ∞ is acceptable due to the large sizes of the test sets.

Table 8.5: Multiple intervals of confidence for the pairwise difference of *mse*. The statistically meaningful differences for $\alpha^* = 0.005$ are in bold-face.

		route 205-1-1		
	BL	EXP	BST	ENS
STT	[-20962, 6505]	**[58633, 74198]**	**[50228, 65288]**	**[60183, 76320]**
BL		**[62065, 85222]**	**[52988, 76985]**	**[64043, 86917]**
EXP			**[-13359, -3955]**	[-2015, 5688]
BST				**[5990, 14997]**
		route 205-1-2		
	BL	EXP	BST	ENS
STT	**[57662, 83345]**	**[48369, 63970]**	**[47117, 65222]**	**[48623, 63716]**
BL		**[59652, 81355]**	**[44541, 67797]**	**[62502, 83123]**
EXP			**[-19351, -9318]**	[-722, 5340]
BST				**[12622, 20664]**
		route 300-1-1		
	BL	EXP	BST	ENS
STT	**[-435850, -237883]**	**[-33746, -13203]**	**[-177985, -97873]**	**[-40873, -8082]**
BL		**[214463, 412322]**	**[105514, 292361]**	**[214113, 410665]**
EXP			**[-150348, -78562]**	[-16010, 14003]
BST				**[81597, 145306]**
		route 301-1-1		
	BL	EXP	BST	ENS
STT	**[-354709, -175278]**	[-23043, 14135]	**[-88588, -23340]**	[-136242, 134140]
BL		**[192565, 328513]**	**[140598, 277461]**	**[144308, 383577]**
EXP			**[-74330, -28690]**	[-95499, 102305]
BST				[-46526, 156352]
		route 505-1-1		
	BL	EXP	BST	ENS
STT	**[-116994, -48231]**	**[13974, 46921]**	**[12327, 44166]**	**[29639, 57202]**
BL		**[81103, 145016]**	**[77591, 144125]**	**[94479, 157486]**
EXP			[-15317, 10915]	**[2749, 23198]**
BST				**[8695, 21654]**
		route 505-1-2		
	BL	EXP	BST	ENS
STT	**[-76287, -8998]**	**[117913,156319]**	**[113049, 157429]**	**[125739, 166029]**
BL		**[152589, 206928]**	**[148811, 206952]**	**[160367, 216686]**
EXP			[-14076, 10322]	**[104, 17432]**
BST				**[2241, 19049]**

STT: Scheduled Travel Time; BL: BaseLine predictor; EXP: EXPert-based predictor; BST: single BeST predictor; ENS: ENSemble predictor.

methods does not change meaningfully with time. Although a deeper analysis would be needed to get more insights into the variation of the generalization error, this study is not made in this thesis. It is a possible subject for future research.

- From figures 8.5 to 8.10 it is apparent that BST performs well in most situations but in some situations performs worse. On all the routes it is very clear that there are periods when BST has quite different behavior from EXP and ENS.
- From the four tested predictors, the expert-based method and the ensemble approach are the most accurate and robust ones. The advantages and disadvantages of each of these methods are outlined:
  - From the accuracy point of view, the ensemble approach has a small advantage. Furthermore, it can be argued that the expert-based method can be included in the ensemble and, in that way, it is expected that the ensemble approach will improve the expert-based results. In fact, the experiments done in Chap. 7 show (even if without statistical validation) that the ensemble approach is able to improve the results of all the individual models in the ensemble.
  - Despite the fact that we have used just four input variables in the experiments done in this chapter, in Sect. 5.7.3 it was apparent that the inclusion of meteorologic variables could improve results. The inclusion of these variables in the expert-based method is not straight because it must be explicitly defined how they are used. In the ensemble approach this is not a problem since the base models are data driven.
  - Much more time and data is necessary to configure the ensemble model when comparing with the expert-based method. In the experiments presented in this chapter, two validation sets and one test set have been used for the ensemble approach. The expert-based method used one validation set and one test set but, if necessary, the validation set can be discarded with a slight loss in accuracy because the algorithm is very stable regarding the input parameters, as shown in Sect. 5.5. This advantage of the expert-based method can be particularly important for the use in practical situations where time is often an important constraint.
  - Without knowing what the business value of the increase in accuracy is, it is not possible to know what price we are able to pay for it.

8.5 Prospecting the use of TTP 3 days ahead

Let us assume that a mass transit company has the right conditions to redefine the planning, namely the duties for the buses and drivers and respective assignment tasks, just three days before the date of the duties. How could the company use TTP results 3 days ahead?

The first thing to do would be to make a new timetable just for internal planning purposes, i.e., not known to the public. Let us denote the values of

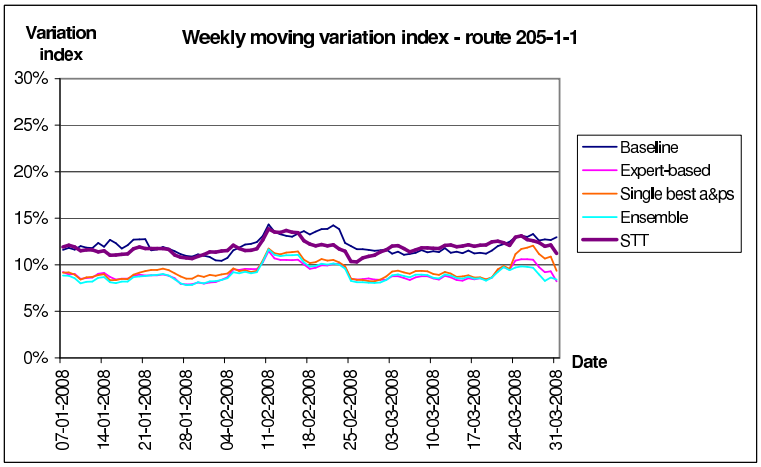

Figure 8.5: Weekly moving *variation index* for route 205-1-1.

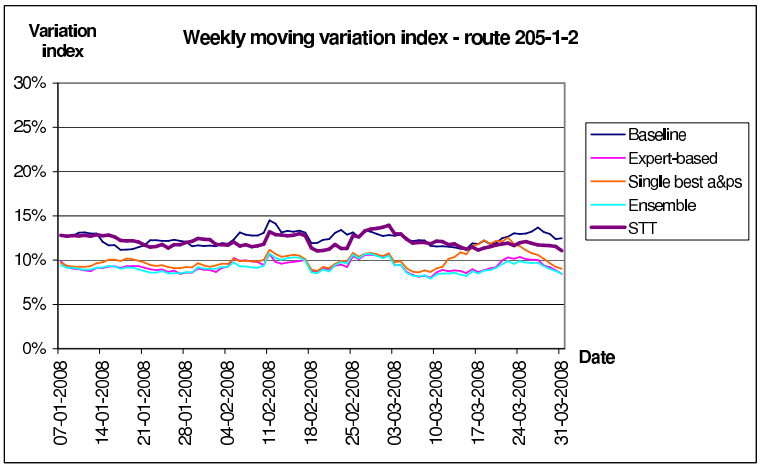

Figure 8.6: Weekly moving *variation index* for route 205-1-2.

this new timetable with the suffix * and let us give to the new scheduled travel time (STT*) the predicted travel time. It is important to note that for planning purposes the only information that is used from this new timetable is the sum of each STT* with the respective SlT*, where SlT* represents the new slack time. A possible approach to estimate a lower limit for STT* + SlT* is to use the decision support system for timetable adjustments (as described in Sect. 3.3.1, using STT* + SlT* = $p.max$) or to define an algorithm to choose one of the possible values for this sum. The process of defining SlT* would be facilitated due to the constraint imposed by Eq. 2.2, which strongly limits the number of acceptable solutions, as discussed in Sect. 3.2.2. These new values for the sum of STT* + SlT* are those needed to define new schedules for both buses and drivers. For control purposes, the timetable known by the public is the one that should be used, together with the new duties for the buses and drivers. The controllers should act according to the headway, as discussed in Sect. 3.3.

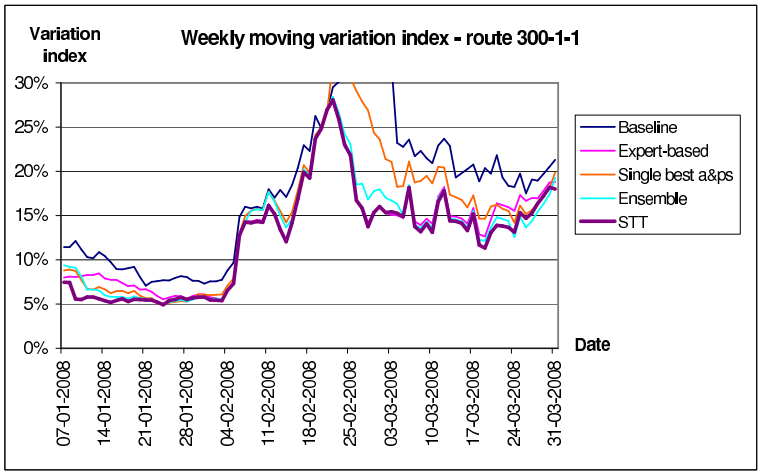

Figure 8.7: Weekly moving *variation index* for route 300-1-1.

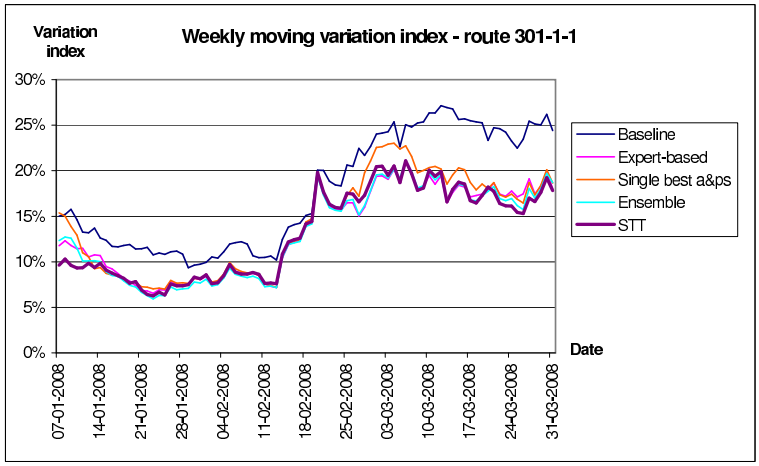

Figure 8.8: Weekly moving *variation index* for route 301-1-1.

Is it possible to do all this just three days before? Technically, we think that it is possible. Even if it is not possible in three days, it could probably be in four or five days. The number of days of anticipation is not the most important although it may be important for the quality of the prediction. What is important is that there is the opportunity to use different schedules for the public and for the planning tasks, giving us the opportunity to reduce the gap between the planned and the actual duties. The best way to do this is still an open issue for research, just as the question of knowing how much we lose in accuracy by increasing the prediction horizon is also open. Another open issue is the impact of these changes in terms of possible operational costs reduction, passengers' satisfaction increase, or even on internal management issues for the companies. These last questions are critical in evaluating the viability of implementing this approach.

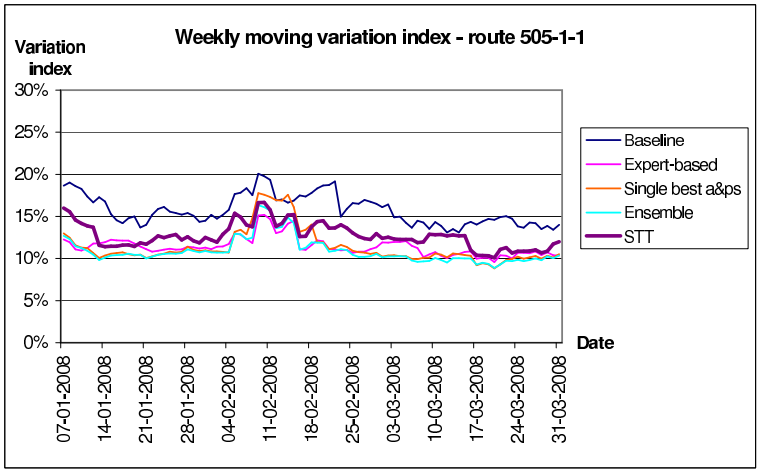

Figure 8.9: Weekly moving *variation index* for route 505-1-1.

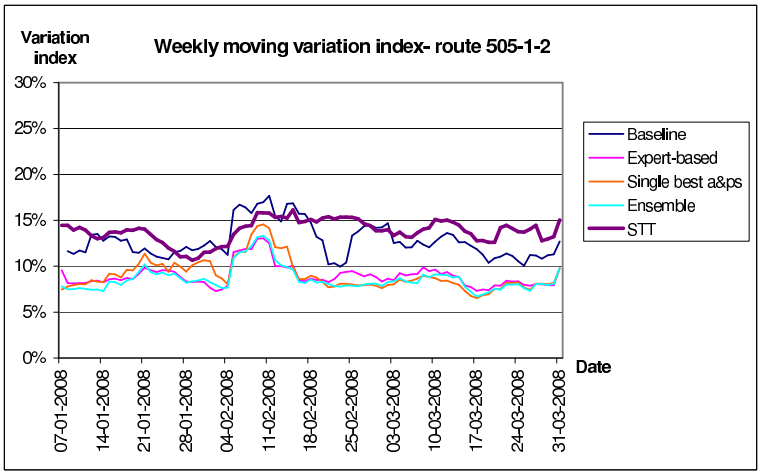

Figure 8.10: Weekly moving *variation index* for route 505-1-2.

Part IV

Concluding remarks

Chapter 9

Conclusions

In this thesis we undertook a study in order to determine how travel time prediction can be used in mass transit companies for planning purposes. Two different problems were identified: the definition of travel times for timetables and the definition of travel times for bus and driver duties. This study assumes, for both problems, the existence of data from actual trips, typically obtained from Automatic Vehicle Location (AVL) systems.

The first problem of timetable definition is a well-known problem with several related studies in the literature. Our approach is not analytical. Instead, we have designed and developed a decision support system that uses past data from the same line and representative of the period the timetable will cover. This problem was the least studied.

With respect to the second problem, the main objective was to find out how much we can increase accuracy if we predict travel times for the definition of bus and driver duties as near the date as possible, instead of using the timetabled travel times. The reason for doing this is that, if the improvement in accuracy is important, it is expected that operational costs reduce and/or passengers' satisfaction increases.

In this second problem we have used machine learning approaches. In order to evaluate such approaches, we started by defining a baseline method to serve as a reference. An expert based method using the knowledge we had at the time together with the traffic experts from the STCP company has also been developed and used for comparison. Then, we tried three different algorithms with reported good results in different problems [Meyer *et al.*, 2003]. They were: support vector machines [Smola and Scholkopf, 2004], random forests [Breiman, 2001a] and projection pursuit regression [Friedman and Stuetzle, 1981]. For each of these algorithms, exhaustive tests were done in order to tune parameters. Other tests were done using the three focusing tasks [Reinartz, 2002]: example selection, domain values selection and feature selection. Accuracy was improved using these approaches.

The next step was to experiment different variants of the dynamic approach using heterogeneous ensembles in order to further improve results with respect to the use of just one model. An extensive survey on ensemble methods for regression was undertaken. Several experiments using the dynamic selection approach were executed. Approaches using ensembles were able to improve results consistently when compared to the use of just one model.

Experiments on the second problem finished by comparing the baseline, the expert based, the best single algorithm (with the respective tuned parameters and focusing tasks), and the ensemble approach, against the use of scheduled travel times (STT), on various routes. Results gave a small advantage in terms of accuracy to the ensemble approach when compared to the expert based method implemented by us. However, the expert based approach needs less data and is much faster to tune. The actual method used by STCP (the use of STT) was competitive for circular routes only. This result can be explained, at least partially, by the way these routes are designed and controlled. On the rest of the routes tested, the method used currently by STCP was clearly beaten by the ensemble approaches.

9.1 Evaluating the initial objectives

We now try to assess to what extent this work has been able to accomplish the main objectives fully described in Sect. 2.4 and the secondary ones described in Sect. 1.2.

9.1.1 Main objectives

Travel time prediction in order to make small adjustments to timetables

It was achieved through the design and implementation of a decision support system that is mainly focused on its utility as a decision support. The solution found is not very relevant in terms of scientific novelty. It is, instead, an easy and effective tool for planners. It is also a framework that can integrate analytical solutions such as the one presented in [Zhao *et al.*, 2006].

Travel time prediction for duties definition

It was extensively studied. It was proved that, at least for the most frequent situations, lines with two routes, the new tested approaches can reduce the mean squared error compared to the use of the scheduled travel times. The average reduction in the *variation index* for the four routes of lines 205 and 505, was around 25% using the ensemble approach and around 23% using the expert-based approach.

9.1.2 Secondary objectives

To contextualize each of the used methodologies in the respective state of art of the problem being solved

A state of the art review on the two addressed problems was undertaken (Sect. 2.2.3).

To contextualize each of the used methodologies in the respective state of art of the methods being used

This was extensively addressed in chapters 4 and 6. In particular, the latter on ensemble methods for regression is, as far as we know, the first comprehensive survey on the subject.

To design and develop the necessary model(s) to accomplish the main objectives of this thesis

Several approaches were tested, some more successfully than others. However, even the failures can be a step forward in the research process. The most successful approaches tested were:

- Selection of examples using those that fall in the same leaf node of a CART tree together with the test example (Sect. 5.7.1). This method gave promising results when used as a filter for support vector regression.
- Selection of similar data for the dynamic approach on ensemble learning using the same CART approach as before (for example selection) but fixing the maximum number of similar examples using the HEOM distance (Sect. 7.3.1). This approach was the most successful for travel time prediction three days ahead among state of the art approaches used in the dynamic selection framework.
- An expert based algorithm (Sect. 5.5) for travel time prediction three days ahead that proved to be competitive due to its simplicity in terms of tuning, lower data requirement and a small loss in accuracy when compared to the ensemble approach, the most accurate of the tested methods.

Testing for the best evaluation function for the pruning algorithms in order to get the most from the dynamic selection approach (Sect. 7.2) is theoretically interesting and is not discussed in the literature. The results obtained using the smse method were not promising (Sect. 7.5.3). However, this was a first step in an open issue for research.

To implement a prototype using the relevant aforementioned model(s)

The DSS for timetabling adjustments was implemented in order to address the first main objective and is being used successfully at STCP.

Two more applications were implemented to address the second main objective: the first one implementing the expert based method [Duarte and Moreira, 2008] and the second one using the ensemble approach used in Chap. 8 [Duarte and Moreira, 2007].

To generalize, when appropriate, the results obtained using the different methodologies to other problems

The good results in example selection for support vector regression using the CART approach (Sect. 5.7.1) were tested in 11 benchmark regression data sets, just for the linear kernel [Moreira *et al.*, 2006a]. Results were promising: with

the exception of two data sets, all the others gave equal or lower mean squared error when using the example selection technique.

Some preliminary experiments on the dynamic selection approach were tested using 5 regression data sets with numeric input variables [Moreira *et al.*, 2007]. These experiments gave important insights into this approach, some of which are described in Sect. 7.3.1.

9.2 Future work

Several research ideas have already been pointed out along this thesis. Here, we summarize the most important ones and some others that seem promising:

- Integrating analytical solutions into the DSS for timetable adjustments: a natural start should be to integrate the solution proposed by [Zhao *et al.*, 2006] extending it to the case of large headways.

- Evaluating the impact of travel time prediction three days ahead on the level of passengers' satisfaction and reduction of operational costs: the way to do this was already pointed out in Sect. 2.3. However, the possibility of conducting this research depends on the STCP's availability to do so, as discussed in the said section.

- Studying the operational implications of using travel time predictions several days ahead in the planning process: this subject was briefly mentioned along this thesis, however, its implications are not fully studied. For example:

  - It is not known the prediction horizon that should be used in order to use this approach on a daily base;
  - It is necessary to study the possible implications of using the predictions in the rostering process (Sect. 2.1.1 and Sect. 2.1.2 for the STCP case);
  - it is still lacking the study on the necessary changes in the control process (Sect. 2.2) due to the use of the predictions in the definition of duties.

- Developing a model to classify the quality of the planning per line: this idea came from the interest of STCP and the interest of a member of our research unit in that problem. She is an expert in Data Envelopment Analysis (DEA), a successful benchmark technique. The use of DEA is not new in mass transit companies [Wang *et al.*, 2006; Barnum *et al.*, 2008]. However, we believe that this is an approach that can contribute to the fulfillment of the business objectives of this research.

- Developing a model that can give insights to the planners into the causes of bus delays: this idea is already being implemented by an MSc. student. She is comparing the use of several methods that can explain the variable offsets. The methods tested are: (1) an algorithm for discovering class association rules [Liu *et al.*, 1998]; (2) an algorithm for associative

classification that is similar to the previous one but with classification capabilities [Liu *et al.*, 1998]; (3) and the C4.5 algorithm [Quinlan, 1993], an algorithm for tree induction.

- Generalizing results on the dynamic selection approach (Sect. 7.5) to other data sets for both regression and classification problems, placing particular emphasis on the similar data search using the CART approach.
- Studying the best training set using concept-drift approaches [Wang *et al.*, 2003].

Appendices

Appendix A

Additional results on focusing tasks

This appendix complements Sect. 5.7 presenting results on example selection (Sect. A.1) and on domain values selection (Sect. A.2). While in Sect. 5.7 we present the results for SVM linear, RF and PPR supsmu, in this appendix we do the same for the remaining algorithms, namely, SVM radial, SVM sigmoid, PPR spline and PPR gcvspline.

A.1 Example selection

In all the figures of the present section, the red line represents the results without example selection, the green line the leaf node approach, and the blue line the equivalent day approach. The straight line represents the baseline method.

A.2 Domain values selection

All figures of this section present results of the tests on the choice of the data type for the variable week day. The red line represents the symbolic (factor) data type and the gray line represents the numeric data type.

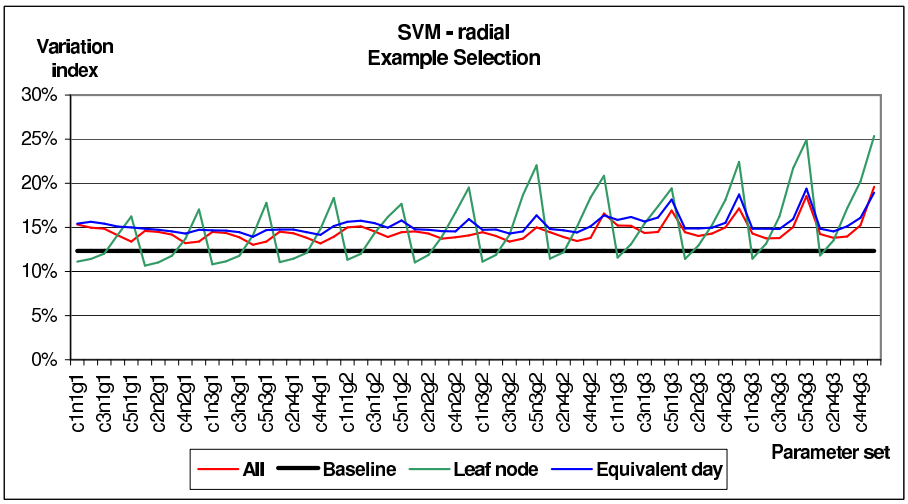

Figure A.1: The *variation index* for SVM - radial using different methods for example selection.

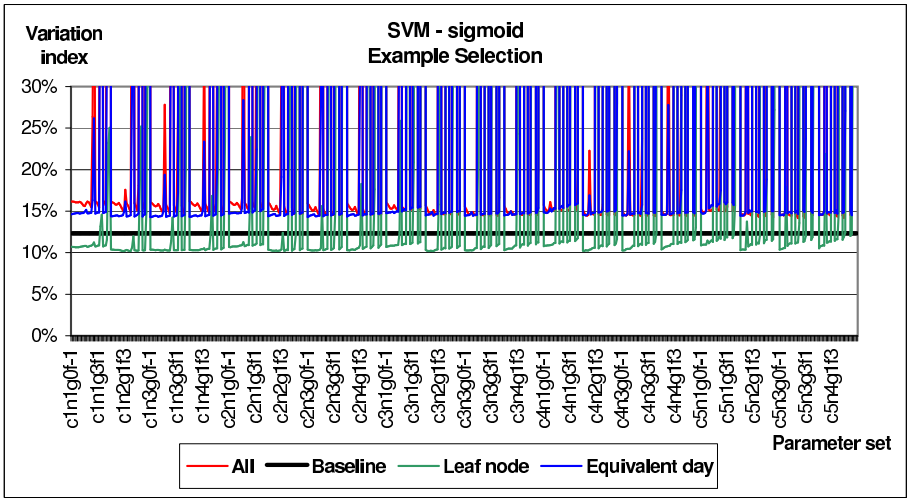

Figure A.2: The *variation index* for SVM - sigmoid using different methods for example selection.

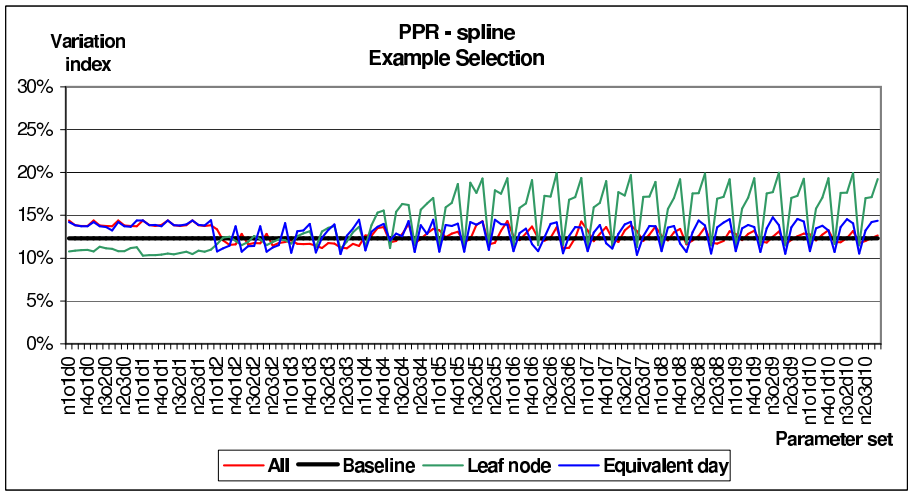

Figure A.3: The *variation index* for PPR - spline using different methods for example selection.

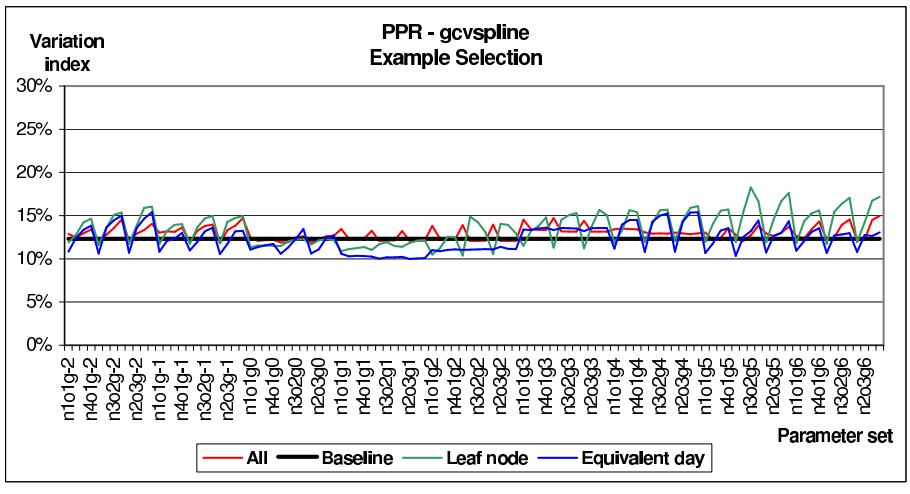

Figure A.4: The *variation index* for PPR - gcvspline using different methods for example selection.

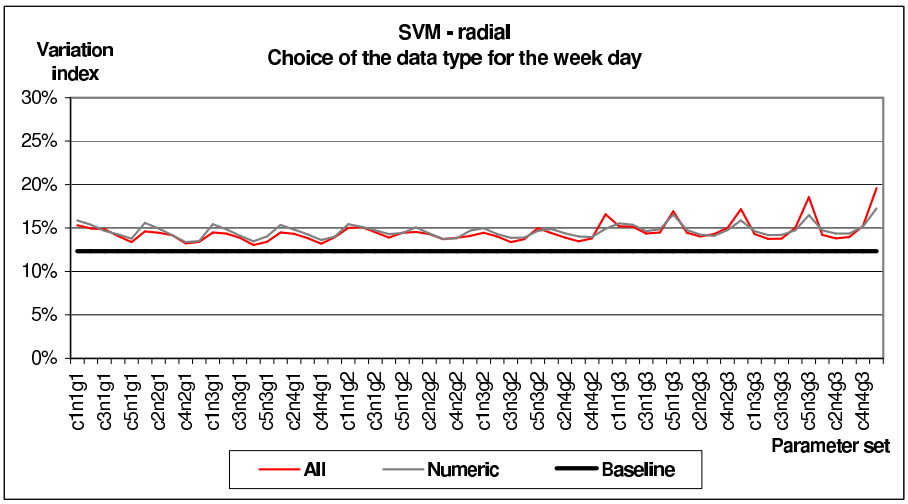

Figure A.5: The *variation index* for SVM - radial using different data types for the variable week day.

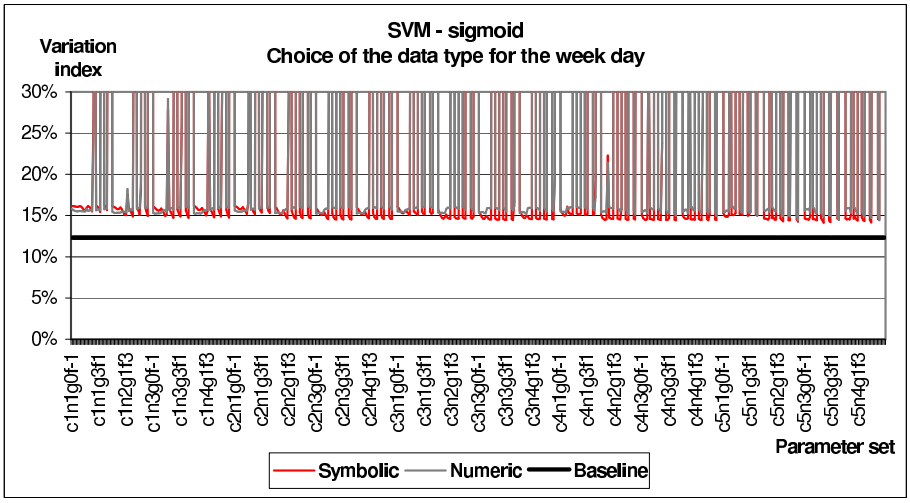

Figure A.6: The *variation index* for SVM - sigmoid using different data types for the variable week day.

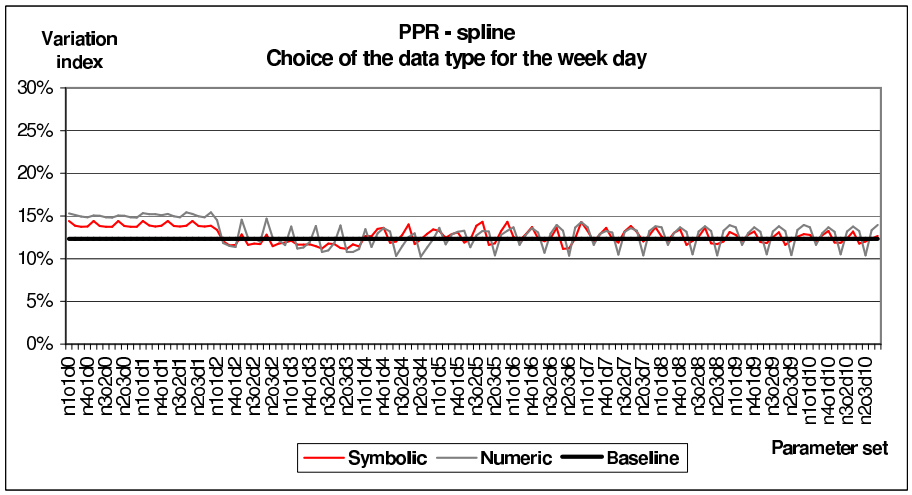

Figure A.7: The *variation index* for PPR - spline using different data types for the variable week day.

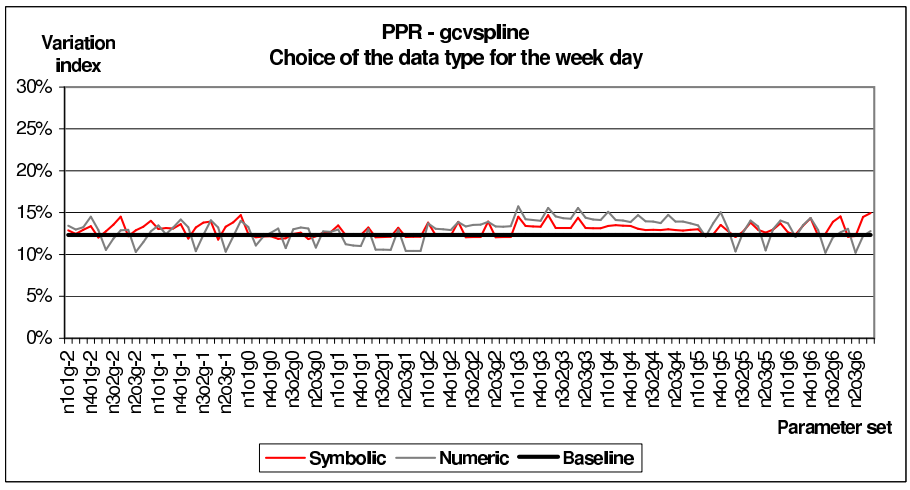

Figure A.8: The *variation index* for PPR - gcvspline using different data types for the variable week day.

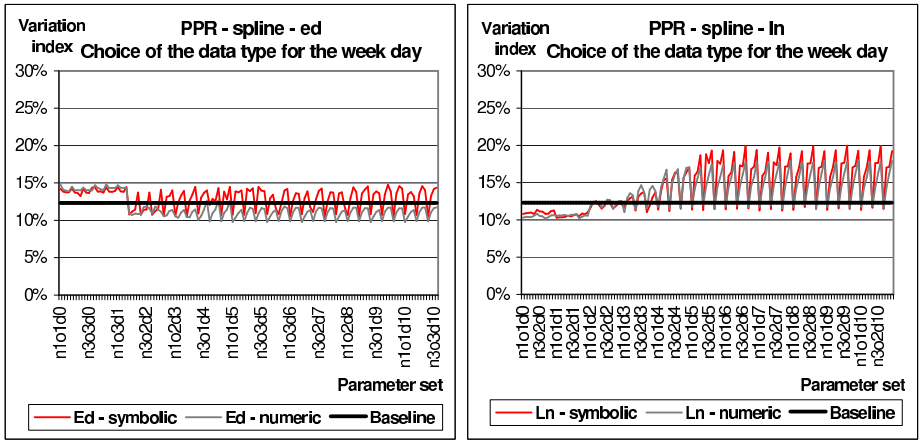

Figure A.9: The *variation index* for PPR - spline (with example selection) using different data types for the variable week day.

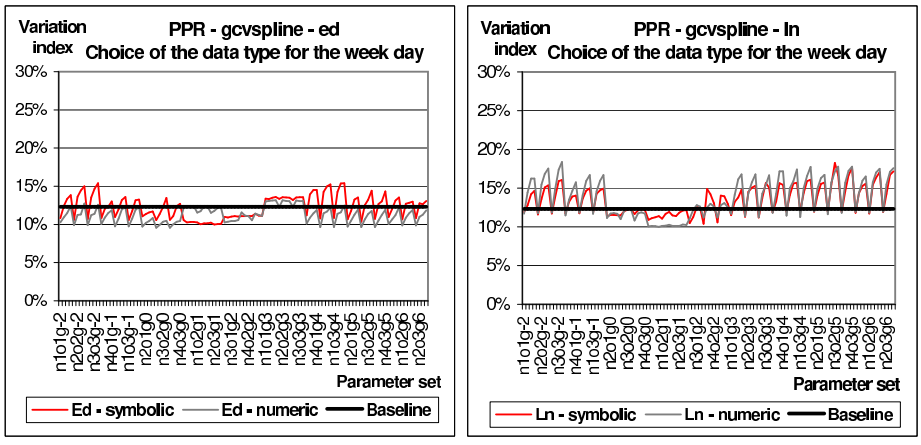

Figure A.10: The *variation index* for PPR - gcvspline (with example selection) using different data types for the variable week day.

Appendix B

Pool 130

This appendix has all the *a&ps* from pool 130. It may help to better understand the results from the experiments on ensemble learning described in Sect. 7.5 and Appendix C. The *a&ps* are presented in table B.1. ES identifies the used example selection method, as described in Sect. 5.7.1. WDDT identifies the data type used for the variable weekday as discussed in Sect. 5.7.2. The parameters Par 1, Par 2, Par 3 and Par 4 should be read as follows:

- For SVM: Par 1 = C, Par 2 = ν and Par 3 = γ;
- For RF: Par 1 = mtry;
- For PPR-supsmu: Par 1 = nterms, Par 2 = optlevel, Par 3 = bass and Par 4 = span;
- For PPR-spline: Par 1 = nterms, Par 2 = optlevel and Par 3 = df;
- For PPR-gcvspline: Par 1 = nterms, Par 2 = optlevel and Par 3 = gcvpen.

Table B.1: The pool 130.

a&ps	**Algorithm**	**ES**	**WDDT**	**Par 1**	**Par 2**	**Par 3**	**Par 4**
1	SVM linear	ln	Symbolic	0.0625	0.1		
2	SVM linear	ln	Symbolic	64	0.1		
3	SVM linear	ln	Symbolic	0.0625	0.3		
4	SVM linear	ln	Symbolic	64	0.3		
5	SVM linear	ln	Symbolic	0.0625	0.5		
6	SVM linear	ln	Symbolic	64	0.5		
7	SVM linear	ln	Symbolic	0.0625	0.7		
8	SVM linear	ln	Symbolic	64	0.7		
9	SVM linear	ln	Symbolic	0.0625	0.9		
10	SVM linear	ln	Symbolic	64	0.9		
11	SVM radial	all	Symbolic	4000	0.2	0.00006	
12	SVM radial	all	Symbolic	64000	0.2	0.00006	
13	SVM radial	all	Symbolic	1024000	0.2	0.00006	
14	SVM radial	all	Symbolic	4000	0.2	0.00018	
15	SVM radial	all	Symbolic	64000	0.2	0.00018	

continued from previous page

a&ps	**Algorithm**	**ES**	**WDDT**	**Par 1**	**Par 2**	**Par 3**	**Par 4**
16	SVM radial	all	Symbolic	1024000	0.2	0.00018	
17	SVM radial	all	Symbolic	4000	0.6	0.00006	
18	SVM radial	all	Symbolic	64000	0.6	0.00006	
19	SVM radial	all	Symbolic	1024000	0.6	0.00006	
20	SVM radial	all	Symbolic	4000	0.6	0.00018	
21	SVM radial	all	Symbolic	64000	0.6	0.00018	
22	SVM radial	all	Symbolic	1024000	0.6	0.00018	
23	SVM radial	ln	Symbolic	4000	0.2	0.00006	
24	SVM radial	ln	Symbolic	64000	0.2	0.00006	
25	SVM radial	ln	Symbolic	1024000	0.2	0.00006	
26	SVM radial	ln	Symbolic	4000	0.6	0.00006	
27	SVM radial	ln	Symbolic	64000	0.6	0.00006	
28	SVM radial	ln	Symbolic	1024000	0.6	0.00006	
29	SVM radial	ln	Symbolic	4000	0.2	0.00018	
30	SVM radial	ln	Symbolic	64000	0.2	0.00018	
31	SVM radial	ln	Symbolic	1024000	0.2	0.00018	
32	SVM radial	ln	Symbolic	4000	0.6	0.00018	
33	SVM radial	ln	Symbolic	64000	0.6	0.00018	
34	SVM radial	ln	Symbolic	1024000	0.6	0.00018	
35	RF	ln	Symbolic	1			
36	RF	ln	Symbolic	3			
37	PPR supsmu	ed	Numeric	1	1	0	0
38	PPR supsmu	ed	Numeric	4	1	0	0
39	PPR supsmu	ed	Numeric	1	2	0	0
40	PPR supsmu	ed	Numeric	4	2	0	0
41	PPR supsmu	ed	Numeric	1	3	0	0
42	PPR supsmu	ed	Numeric	4	3	0	0
43	PPR supsmu	ed	Numeric	1	1	5	0
44	PPR supsmu	ed	Numeric	4	1	5	0
45	PPR supsmu	ed	Numeric	1	2	5	0
46	PPR supsmu	ed	Numeric	4	2	5	0
47	PPR supsmu	ed	Numeric	1	3	5	0
48	PPR supsmu	ed	Numeric	4	3	5	0
49	PPR supsmu	ed	Numeric	1	1	10	0
50	PPR supsmu	ed	Numeric	4	1	10	0
51	PPR supsmu	ed	Numeric	1	2	10	0
52	PPR supsmu	ed	Numeric	4	2	10	0
53	PPR supsmu	ed	Numeric	1	3	10	0
54	PPR supsmu	ed	Numeric	4	3	10	0
55	PPR spline	all	Numeric	1	1	1	
56	PPR spline	all	Numeric	1	1	32	
57	PPR spline	all	Numeric	1	1	1024	
58	PPR spline	all	Numeric	1	3	1	
59	PPR spline	all	Numeric	1	3	32	
60	PPR spline	all	Numeric	1	3	1024	
61	PPR spline	all	Numeric	2	1	1	
62	PPR spline	all	Numeric	2	1	32	
63	PPR spline	all	Numeric	2	1	1024	

continued from previous page

a&ps	**Algorithm**	**ES**	**WDDT**	**Par 1**	**Par 2**	**Par 3**	**Par 4**
64	PPR spline	all	Numeric	2	3	1	
65	PPR spline	all	Numeric	2	3	32	
66	PPR spline	all	Numeric	2	3	1024	
67	PPR spline	ed	Numeric	1	1	1	
68	PPR spline	ed	Numeric	2	1	1	
69	PPR spline	ed	Numeric	1	3	1	
70	PPR spline	ed	Numeric	2	3	1	
71	PPR spline	ed	Numeric	1	1	32	
72	PPR spline	ed	Numeric	2	1	32	
73	PPR spline	ed	Numeric	1	3	32	
74	PPR spline	ed	Numeric	2	3	32	
75	PPR spline	ed	Numeric	1	1	1024	
76	PPR spline	ed	Numeric	2	1	1024	
77	PPR spline	ed	Numeric	1	3	1024	
78	PPR spline	ed	Numeric	2	3	1024	
79	PPR gcvspline	all	Numeric	1	1	0.0625	
80	PPR gcvspline	all	Numeric	2	1	0.0625	
81	PPR gcvspline	all	Numeric	1	3	0.0625	
82	PPR gcvspline	all	Numeric	2	3	0.0625	
83	PPR gcvspline	all	Numeric	1	1	16	
84	PPR gcvspline	all	Numeric	2	1	16	
85	PPR gcvspline	all	Numeric	1	3	16	
86	PPR gcvspline	all	Numeric	2	3	16	
87	PPR gcvspline	all	Numeric	1	1	4096	
88	PPR gcvspline	all	Numeric	2	1	4096	
89	PPR gcvspline	all	Numeric	1	3	4096	
90	PPR gcvspline	all	Numeric	2	3	4096	
91	PPR gcvspline	ed	Numeric	1	1	0.0625	
92	PPR gcvspline	ed	Numeric	2	1	0.0625	
93	PPR gcvspline	ed	Numeric	1	3	0.0625	
94	PPR gcvspline	ed	Numeric	2	3	0.0625	
95	PPR gcvspline	ed	Numeric	1	1	16	
96	PPR gcvspline	ed	Numeric	2	1	16	
97	PPR gcvspline	ed	Numeric	1	3	16	
98	PPR gcvspline	ed	Numeric	2	3	16	
99	PPR gcvspline	ed	Numeric	1	1	4096	
100	PPR gcvspline	ed	Numeric	2	1	4096	
101	PPR gcvspline	ed	Numeric	1	3	4096	
102	PPR gcvspline	ed	Numeric	2	3	4096	
103	SVM linear	all	Symbolic	0.0625	0.1		
104	SVM linear	all	Symbolic	64	0.1		
105	SVM linear	all	Symbolic	0.0625	0.3		
106	SVM linear	all	Symbolic	64	0.3		
107	SVM linear	all	Symbolic	0.0625	0.5		
108	SVM linear	all	Symbolic	64	0.5		
109	SVM linear	all	Symbolic	0.0625	0.7		
110	SVM linear	all	Symbolic	64	0.7		
111	SVM linear	all	Symbolic	0.0625	0.9		

continued from previous page

a&ps	**Algorithm**	**ES**	**WDDT**	**Par 1**	**Par 2**	**Par 3**	**Par 4**
112	SVM linear	all	Symbolic	64	0.9		
113	PPR supsmu	all	Numeric	1	1	0	0
114	PPR supsmu	all	Numeric	4	1	0	0
115	PPR supsmu	all	Numeric	1	2	0	0
116	PPR supsmu	all	Numeric	4	2	0	0
117	PPR supsmu	all	Numeric	1	3	0	0
118	PPR supsmu	all	Numeric	4	3	0	0
119	PPR supsmu	all	Numeric	1	1	5	0
120	PPR supsmu	all	Numeric	4	1	5	0
121	PPR supsmu	all	Numeric	1	2	5	0
122	PPR supsmu	all	Numeric	4	2	5	0
123	PPR supsmu	all	Numeric	1	3	5	0
124	PPR supsmu	all	Numeric	4	3	5	0
125	PPR supsmu	all	Numeric	1	1	10	0
126	PPR supsmu	all	Numeric	4	1	10	0
127	PPR supsmu	all	Numeric	1	2	10	0
128	PPR supsmu	all	Numeric	4	2	10	0
129	PPR supsmu	all	Numeric	1	3	10	0
130	PPR supsmu	all	Numeric	4	3	10	0

Appendix C

Additional results on ensemble learning

This appendix complements Sect. 7.5 by presenting additional results to the first and the third experiments on ensemble learning. In those experiments we show results for ensembles of size 5. In this appendix we present results for ensembles of size 10, 15, 20 and 25.

C.1 First experiment

The goal of the first experiment is to select a method for obtaining similar data. We present four groups of tables, each one with four tables (tables from C.1 to C.16). For each group we use a different ensemble, all of them obtained using the pruning algorithm Mor06-smse (Sect. 7.2) but for different sizes: 10, 15, 20 and 25. For each of of the ensembles, we present four tables: one with the results using the base learners and also, for the constant weighting function, simple average. Each of the other three tables are for a different method to obtain similar data, namely knn-CART, knn-HEOM and knn-RReliefF (Sect. 7.3.1).

Table C.1: First experiment: the *variation index* for the base learners and simple average using the ensemble of size 10.

a&ps **1**	*a&ps* **12**	*a&ps* **16**	*a&ps* **30**	*a&ps* **34**	**Avg**
11.59%	12.22%	12.10%	11.61%	11.92%	10.09%
a&ps **36**	*a&ps* **51**	*a&ps* **58**	*a&ps* **74**	*a&ps* **81**	
11.55%	11.23%	13.50%	11.03%	12.903%	

Table C.2: First experiment: the *variation index* for the knn-CART using the ensemble of size 10.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
30	11.65%	11.48%	9.78%	9.81%	9.76%	9.72%	9.72%	9.72%
25	11.52%	11.48%	9.78%	9.79%	9.77%	9.77%	9.75%	9.73%
20	11.68%	11.51%	9.79%	9.86%	9.81%	9.82%	9.77%	9.77%
18	11.76%	11.58%	9.80%	9.91%	9.84%	9.84%	9.78%	9.77%
16	11.68%	11.56%	9.80%	9.92%	9.89%	9.84%	9.79%	9.78%
14	11.94%	11.68%	9.80%	9.99%	9.90%	9.86%	9.81%	9.79%
12	11.95%	11.68%	9.81%	10.02%	9.94%	9.85%	9.83%	9.81%
10	11.87%	11.67%	9.81%	9.99%	9.93%	9.89%	9.84%	9.83%
8	11.79%	11.67%	9.81%	9.98%	9.93%	9.85%	9.83%	9.83%
6	11.83%	11.74%	9.82%	10.04%	9.94%	9.93%	9.87%	9.85%
4	10.33%	10.17%	9.81%	10.15%	10.09%	9.95%	9.88%	9.87%
2	10.45%	10.30%	9.82%	10.35%	10.21%	10.11%	10.03%	9.96%

Table C.3: First experiment: the *variation index* for the knn-HEOM using the ensemble of size 10.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
30	11.35%	10.67%	10.10%	10.82%	10.72%	10.34%	10.18%	10.15%
25	11.06%	10.51%	10.10%	10.82%	10.67%	10.30%	10.17%	10.14%
20	10.89%	10.47%	10.09%	10.55%	10.56%	10.28%	10.16%	10.14%
18	11.09%	10.40%	10.08%	10.53%	10.58%	10.28%	10.16%	10.14%
16	11.12%	10.32%	10.08%	10.33%	10.54%	10.30%	10.15%	10.14%
14	10.81%	10.42%	10.08%	10.32%	10.48%	10.36%	10.18%	10.14%
12	10.92%	10.44%	10.07%	10.32%	10.49%	10.35%	10.21%	10.14%
10	10.83%	10.51%	10.06%	10.29%	10.45%	10.35%	10.22%	10.14%
8	10.79%	10.50%	10.05%	10.42%	10.43%	10.33%	10.23%	10.14%
6	10.69%	10.58%	10.05%	10.58%	10.46%	10.36%	10.25%	10.15%
4	11.02%	10.79%	10.03%	10.64%	10.54%	10.38%	10.27%	10.18%
2	11.06%	10.72%	10.04%	10.69%	10.54%	10.52%	10.38%	10.32%

Table C.4: First experiment: the *variation index* for the knn-RReliefF using the ensemble of size 10.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
30	11.11%	10.58%	10.09%	11.21%	10.80%	10.56%	10.41%	10.35%
25	11.06%	10.60%	10.10%	10.92%	10.48%	10.35%	10.27%	10.20%
20	11.81%	10.80%	10.06%	10.87%	10.33%	10.11%	10.11%	10.10%
18	11.68%	10.69%	10.08%	10.94%	10.38%	10.20%	10.21%	10.19%
16	11.70%	10.83%	10.01%	11.00%	10.57%	10.32%	10.28%	10.27%
14	11.53%	11.05%	10.09%	11.14%	10.55%	10.29%	10.24%	10.16%
12	11.52%	10.90%	10.10%	11.02%	10.51%	10.28%	10.23%	10.15%
10	11.61%	10.96%	10.11%	11.15%	10.62%	10.38%	10.24%	10.18%
8	11.44%	10.92%	10.08%	11.09%	10.63%	10.34%	10.19%	10.11%
6	11.49%	11.02%	10.10%	11.12%	10.77%	10.48%	10.32%	10.20%
4	11.77%	11.22%	10.12%	11.39%	11.01%	10.61%	10.54%	10.35%
2	11.74%	11.40%	10.13%	11.34%	10.93%	10.76%	10.69%	10.62%

Table C.5: First experiment: the *variation index* for the base learners and simple average using the ensemble of size 15.

a&ps **1**	*a&ps* **4**	*a&ps* **12**	*a&ps* **16**	*a&ps* **30**	**Avg**
11.59%	11.31%	12.22%	12.10%	11.61%	10.00%
a&ps **31**	*a&ps* **34**	*a&ps* **36**	*a&ps* **51**	*a&ps* **58**	
12.46%	11.92%	11.53%	11.23%	13.50%	
a&ps **68**	*a&ps* **74**	*a&ps* **77**	*a&ps* **79**	*a&ps* **114**	
12.34%	11.03%	10.03%	11.99%	11.66%	

Table C.6: First experiment: the *variation index* for the knn-CART using the ensemble of size 15.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
30	11.66%	11.45%	9.72%	9.76%	9.73%	9.69%	9.68%	9.67%
25	11.61%	11.44%	9.72%	9.78%	9.71%	9.71%	9.69%	9.67%
20	11.62%	11.42%	9.73%	9.89%	9.72%	9.75%	9.71%	9.70%
18	11.63%	11.49%	9.73%	9.92%	9.77%	9.76%	9.71%	9.71%
16	11.72%	11.49%	9.73%	9.96%	9.79%	9.77%	9.72%	9.72%
14	11.74%	11.52%	9.74%	10.03%	9.84%	9.78%	9.73%	9.73%
12	11.80%	11.51%	9.74%	10.10%	9.85%	9.78%	9.75%	9.74%
10	11.68%	11.49%	9.74%	10.08%	9.88%	9.79%	9.74%	9.73%
8	11.79%	11.57%	9.74%	10.13%	9.84%	9.77%	9.73%	9.73%
6	11.65%	11.51%	9.74%	10.20%	9.89%	9.83%	9.77%	9.75%
4	10.31%	10.15%	9.73%	10.31%	10.07%	9.90%	9.80%	9.79%
2	10.51%	10.12%	9.71%	10.47%	10.07%	9.96%	9.89%	9.81%

Table C.7: First experiment: the *variation index* for the knn-HEOM using the ensemble of size 15.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
30	11.48%	10.65%	10.00%	10.25%	10.25%	10.20%	10.09%	10.06%
25	11.32%	10.41%	10.00%	10.27%	10.20%	10.15%	10.06%	10.05%
20	10.62%	10.26%	9.99%	10.26%	10.22%	10.13%	10.05%	10.04%
18	10.90%	10.47%	9.99%	10.19%	10.20%	10.12%	10.05%	10.04%
16	10.82%	10.46%	9.98%	10.18%	10.22%	10.13%	10.04%	10.04%
14	10.69%	10.41%	9.98%	10.42%	10.22%	10.14%	10.04%	10.05%
12	10.97%	10.67%	9.98%	10.48%	10.23%	10.18%	10.08%	10.04%
10	10.89%	10.60%	9.97%	10.35%	10.22%	10.17%	10.06%	10.02%
8	10.94%	10.50%	9.96%	10.39%	10.26%	10.16%	10.08%	10.03%
6	10.83%	10.49%	9.96%	10.65%	10.35%	10.25%	10.13%	10.07%
4	11.00%	10.69%	9.93%	10.68%	10.34%	10.29%	10.16%	10.10%
2	11.26%	10.62%	9.91%	10.81%	10.39%	10.26%	10.19%	10.18%

Table C.8: First experiment: the *variation index* for the knn-RReliefF using the ensemble of size 15.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
30	10.97%	10.37%	9.98%	10.42%	10.20%	10.07%	10.03%	10.01%
25	11.22%	10.55%	10.01%	10.46%	10.12%	10.05%	9.99%	9.98%
20	11.72%	10.84%	9.98%	10.75%	10.19%	10.08%	10.06%	10.01%
18	11.41%	10.58%	9.97%	10.61%	10.12%	10.08%	10.03%	10.00%
16	11.61%	10.87%	9.97%	10.85%	10.23%	10.05%	10.02%	10.01%
14	11.48%	10.90%	9.97%	10.89%	10.26%	10.12%	10.03%	10.01%
12	11.31%	10.80%	9.95%	10.71%	10.21%	10.06%	10.01%	9.99%
10	11.30%	10.83%	9.95%	10.74%	10.21%	10.08%	10.01%	9.97%
8	11.47%	11.06%	9.95%	10.97%	10.42%	10.18%	10.04%	9.96%
6	11.60%	11.06%	9.97%	11.06%	10.72%	10.38%	10.34%	10.20%
4	11.71%	11.32%	9.98%	11.23%	10.69%	10.42%	10.30%	10.24%
2	11.63%	11.24%	10.01%	11.21%	10.95%	10.69%	10.48%	10.42%

Table C.9: First experiment: the *variation index* for the base learners and simple average using the ensemble of size 20.

a&ps **1**	*a&ps* **2**	*a&ps* **4**	*a&ps* **12**	*a&ps* **15**	**Avg**
11.59%	11.97%	11.31%	12.22%	11.66%	10.04%
a&ps **16**	*a&ps* **22**	*a&ps* **27**	*a&ps* **30**	*a&ps* **31**	
12.10%	11.90%	10.98%	11.61%	12.46%	
a&ps **34**	*a&ps* **36**	*a&ps* **38**	*a&ps* **51**	*a&ps* **57**	
11.92%	11.59%	13.46%	11.23%	12.49%	
a&ps **58**	*a&ps* **68**	*a&ps* **74**	*a&ps* **91**	*a&ps* **114**	
13.50%	12.34%	11.03%	10.64%	11.66%	

Table C.10: First experiment: the *variation index* for the knn-CART using the ensemble of size 20.

k	Best	FSwR	DWS 10%	DWS 30%	DWS 50%	DWS 70%	DWS 90%	DW
30	11.82%	11.57%	9.76%	9.78%	9.72%	9.71%	9.70%	9.70%
25	11.81%	11.64%	9.76%	9.80%	9.73%	9.73%	9.71%	9.71%
20	11.91%	11.63%	9.77%	9.84%	9.76%	9.77%	9.73%	9.74%
18	11.91%	11.69%	9.78%	9.89%	9.79%	9.80%	9.74%	9.75%
16	12.03%	11.75%	9.78%	9.91%	9.79%	9.79%	9.74%	9.75%
14	12.02%	11.82%	9.78%	10.02%	9.82%	9.80%	9.76%	9.75%
12	12.05%	11.78%	9.78%	10.00%	9.87%	9.81%	9.77%	9.77%
10	12.14%	11.97%	9.79%	10.09%	9.90%	9.84%	9.79%	9.78%
8	12.08%	11.95%	9.79%	10.08%	9.86%	9.84%	9.81%	9.79%
6	11.86%	11.83%	9.79%	10.10%	9.84%	9.89%	9.84%	9.84%
4	10.33%	10.30%	9.78%	10.22%	10.00%	9.97%	9.89%	9.87%
2	10.32%	10.30%	9.80%	10.41%	10.26%	10.12%	10.05%	9.96%

Table C.11: First experiment: the *variation index* for the knn-HEOM using the ensemble of size 20.

k	Best	FSwR	DWS 10%	DWS 30%	DWS 50%	DWS 70%	DWS 90%	DW
30	12.44%	11.23%	10.03%	10.49%	10.16%	10.20%	10.12%	10.07%
25	11.17%	10.65%	10.02%	10.61%	10.11%	10.12%	10.08%	10.06%
20	10.97%	10.51%	10.01%	10.42%	10.13%	10.08%	10.04%	10.04%
18	11.12%	10.64%	10.01%	10.49%	10.16%	10.06%	10.04%	10.04%
16	10.94%	10.68%	10.01%	10.44%	10.14%	10.05%	10.05%	10.05%
14	11.00%	10.67%	10.00%	10.46%	10.15%	10.07%	10.03%	10.05%
12	11.18%	10.85%	10.00%	10.43%	10.17%	10.10%	10.04%	10.03%
10	11.20%	10.95%	9.99%	10.40%	10.19%	10.12%	10.05%	10.04%
8	11.17%	10.90%	9.98%	10.49%	10.20%	10.15%	10.04%	10.00%
6	11.19%	10.81%	9.99%	10.59%	10.36%	10.28%	10.18%	10.07%
4	10.98%	10.80%	9.97%	10.57%	10.39%	10.31%	10.21%	10.13%
2	10.77%	10.77%	9.97%	10.69%	10.58%	10.47%	10.42%	10.34%

Table C.12: First experiment: the *variation index* for the knn-RReliefF using the ensemble of size 20.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
30	11.28%	10.50%	9.88%	10.75%	10.49%	10.38%	10.30%	10.27%
25	11.38%	10.62%	9.98%	10.62%	10.28%	10.18%	10.14%	10.11%
20	11.75%	10.71%	9.97%	10.73%	10.27%	10.15%	10.11%	10.12%
18	11.59%	10.58%	10.02%	10.70%	10.20%	10.05%	10.06%	10.05%
16	11.62%	10.78%	10.07%	10.84%	10.27%	10.15%	10.10%	10.07%
14	11.47%	10.84%	10.05%	10.93%	10.37%	10.22%	10.08%	10.04%
12	11.39%	10.90%	10.03%	10.94%	10.38%	10.20%	10.09%	10.03%
10	11.49%	10.99%	10.03%	11.01%	10.43%	10.21%	10.10%	10.03%
8	11.30%	11.08%	10.02%	10.84%	10.51%	10.26%	10.15%	10.08%
6	11.70%	11.32%	10.03%	11.10%	10.65%	10.36%	10.20%	10.12%
4	11.67%	11.33%	10.04%	11.33%	10.81%	10.52%	10.35%	10.25%
2	11.91%	11.47%	10.05%	11.39%	11.02%	10.80%	10.61%	10.47%

Table C.13: First experiment: the *variation index* for the base learners and simple average using the ensemble of size 25.

a&ps **1**	*a&ps* **2**	*a&ps* **4**	*a&ps* **12**	*a&ps* **15**	**Avg**
11.59%	11.97%	11.31%	12.22%	11.66%	9.95%
a&ps **16**	*a&ps* **22**	*a&ps* **27**	*a&ps* **30**	*a&ps* **31**	
12.10%	11.90%	10.98%	11.61%	12.46%	
a&ps **34**	*a&ps* **36**	*a&ps* **38**	*a&ps* **40**	*a&ps* **46**	
11.92%	11.57%	13.46%	13.57%	10.50%	
a&ps **53**	*a&ps* **58**	*a&ps* **68**	*a&ps* **71**	*a&ps* **74**	
11.32%	13.50%	12.34%	11.18%	11.03%	
a&ps **79**	*a&ps* **95**	*a&ps* **103**	*a&ps* **114**	*a&ps* **116**	
11.99%	11.06%	13.74%	11.66%	11.76%	

Table C.14: First experiment: the *variation index* for the knn-CART using the ensemble of size 25.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
30	11.45%	11.31%	9.76%	9.68%	9.65%	9.68%	9.69%	9.69%
25	11.49%	11.29%	9.76%	9.67%	9.65%	9.69%	9.70%	9.69%
20	11.52%	11.29%	9.77%	9.75%	9.66%	9.71%	9.71%	9.71%
18	11.62%	11.38%	9.77%	9.72%	9.68%	9.72%	9.71%	9.71%
16	11.57%	11.40%	9.77%	9.75%	9.69%	9.70%	9.71%	9.72%
14	11.61%	11.48%	9.77%	9.87%	9.68%	9.71%	9.71%	9.72%
12	11.60%	11.50%	9.78%	9.93%	9.69%	9.71%	9.71%	9.72%
10	11.69%	11.63%	9.77%	9.93%	9.72%	9.73%	9.71%	9.72%
8	11.64%	11.58%	9.77%	9.95%	9.73%	9.72%	9.69%	9.70%
6	11.72%	11.60%	9.77%	9.97%	9.76%	9.77%	9.71%	9.74%
4	11.91%	11.89%	9.75%	10.07%	9.88%	9.83%	9.74%	9.75%
2	11.98%	11.87%	9.78%	10.53%	10.30%	10.10%	9.99%	9.89%

Table C.15: First experiment: the *variation index* for the knn-HEOM using the ensemble of size 25.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
30	10.75%	10.23%	9.89%	10.06%	9.82%	9.92%	9.91%	9.91%
25	10.52%	10.25%	9.89%	10.10%	9.83%	9.86%	9.88%	9.90%
20	10.25%	10.24%	9.88%	9.98%	9.83%	9.82%	9.84%	9.88%
18	10.41%	10.19%	9.88%	10.13%	9.90%	9.80%	9.84%	9.87%
16	10.54%	10.20%	9.87%	10.14%	9.90%	9.80%	9.84%	9.87%
14	10.49%	10.34%	9.87%	10.14%	9.90%	9.80%	9.82%	9.86%
12	10.59%	10.37%	9.86%	10.22%	9.94%	9.84%	9.82%	9.85%
10	10.83%	10.48%	9.86%	10.25%	10.01%	9.84%	9.82%	9.82%
8	10.71%	10.34%	9.84%	10.25%	9.93%	9.85%	9.80%	9.80%
6	11.08%	10.69%	9.84%	10.39%	10.10%	9.98%	9.91%	9.87%
4	11.08%	10.68%	9.83%	10.49%	10.26%	10.09%	10.00%	9.94%
2	11.18%	10.87%	9.82%	10.78%	10.44%	10.29%	10.22%	10.12%

Table C.16: First experiment: the *variation index* for the knn-RReliefF using the ensemble of size 25.

k	Best	FSwR	DWS 10%	DWS 30%	DWS 50%	DWS 70%	DWS 90%	DW
30	10.94%	10.34%	9.97%	10.16%	10.11%	10.00%	9.99%	9.99%
25	10.92%	10.43%	9.93%	10.30%	10.16%	10.02%	10.03%	10.02%
20	11.20%	10.48%	9.90%	10.06%	10.06%	10.01%	9.95%	9.96%
18	11.21%	10.30%	9.90%	10.22%	10.09%	10.03%	9.98%	9.97%
16	11.43%	10.53%	9.90%	10.29%	10.10%	10.00%	9.94%	9.93%
14	11.38%	10.64%	9.92%	10.42%	10.08%	9.97%	9.93%	9.92%
12	11.36%	10.81%	9.91%	10.47%	10.05%	9.97%	9.93%	9.92%
10	11.58%	11.08%	9.89%	10.42%	10.13%	9.95%	9.90%	9.88%
8	11.42%	11.24%	9.89%	10.58%	10.18%	10.00%	9.90%	9.86%
6	11.87%	11.35%	9.89%	11.04%	10.47%	10.20%	10.04%	9.97%
4	11.94%	11.30%	9.90%	11.17%	10.38%	10.23%	10.08%	10.03%
2	11.69%	11.45%	9.92%	11.14%	10.85%	10.63%	10.47%	10.35%

C.2 Third experiment

The goal of the third experiment is to compare the two evaluation functions used in the pruning algorithms, namely smse and avg (Sect. 7.2). We present four groups of tables (tables from C.17 to C.28). Each group refers to the size of the ensemble: 10, 15, 20 and 25. For each group we show three tables: one of them using the ensemble obtained with the evaluation function smse, and the other two for the evaluation function avg. The first table shows the results for knn-CART. The last two tables contain: the results using the base learners and the constant weighting function, the simple average, in the first table; and the results using knn-CART in the second one. The results of the base learners and the simple average for the smse evaluation function are shown in tables C.1, C.5, C.9 and C.13.

Table C.17: Third experiment: the *variation index* for the knn-CART on the 10 size ensemble obtained using Mor06-smse.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
∞	11.74%	11.50%	9.78%	9.78%	9.61%	9.68%	9.72%	9.71%
200	11.71%	11.49%	9.78%	9.78%	9.63%	9.69%	9.72%	9.71%
100	11.76%	11.50%	9.78%	9.85%	9.65%	9.71%	9.72%	9.72%
90	11.75%	11.53%	9.78%	9.85%	9.65%	9.71%	9.71%	9.72%
80	11.80%	11.58%	9.78%	9.83%	9.65%	9.71%	9.71%	9.72%
70	11.71%	11.55%	9.77%	9.83%	9.64%	9.70%	9.71%	9.72%
60	11.69%	11.50%	9.77%	9.86%	9.65%	9.70%	9.72%	9.72%
50	11.58%	11.48%	9.77%	9.86%	9.68%	9.71%	9.72%	9.72%
40	11.80%	11.59%	9.77%	9.81%	9.72%	9.71%	9.72%	9.73%
30	11.64%	11.47%	9.77%	9.80%	9.75%	9.72%	9.72%	9.72%
20	11.67%	11.47%	9.79%	9.86%	9.81%	9.82%	9.78%	9.77%
10	11.86%	11.66%	9.81%	9.99%	9.93%	9.89%	9.84%	9.83%

Table C.18: Third experiment: the *variation index* for the base learners and simple average on the 10 size ensemble obtained using Mor06-avg.

a&ps **4**	*a&ps* **35**	*a&ps* **36**	*a&ps* **41**	*a&ps* **48**	**Avg**
11.31%	11.40%	11.55%	12.73%	10.45%	9.68%
a&ps **71**	*a&ps* **72**	*a&ps* **75**	*a&ps* **96**	*a&ps* **120**	
11.18%	12.70%	11.02%	10.91%	10.87%	

Table C.19: Third experiment: the *variation index* for the knn-CART on the 10 size ensemble obtained using Mor06-avg.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
∞	9.78%	9.55%	9.62%	9.48%	9.54%	9.62%	9.63%	9.63%
200	9.78%	9.55%	9.62%	9.41%	9.55%	9.63%	9.63%	9.63%
100	9.74%	9.50%	9.63%	9.61%	9.58%	9.63%	9.63%	9.64%
90	9.89%	9.59%	9.63%	9.62%	9.58%	9.62%	9.64%	9.64%
80	9.95%	9.66%	9.63%	9.66%	9.58%	9.63%	9.64%	9.64%
70	10.13%	9.76%	9.63%	9.74%	9.57%	9.63%	9.63%	9.64%
60	10.14%	9.82%	9.63%	9.74%	9.56%	9.64%	9.63%	9.64%
50	10.13%	10.01%	9.63%	9.76%	9.57%	9.64%	9.63%	9.64%
40	10.35%	10.05%	9.63%	9.80%	9.56%	9.63%	9.62%	9.63%
30	10.41%	10.29%	9.63%	9.91%	9.57%	9.62%	9.63%	9.64%
20	10.49%	10.36%	9.64%	10.03%	9.57%	9.64%	9.63%	9.64%
10	10.65%	10.44%	9.65%	10.20%	9.69%	9.64%	9.67%	9.67%

Table C.20: Third experiment: the *variation index* for the knn-CART on the 15 size ensemble obtained using Mor06-smse.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
∞	11.74%	11.73%	9.68%	9.85%	9.55%	9.59%	9.63%	9.63%
200	11.84%	11.70%	9.68%	9.77%	9.57%	9.59%	9.64%	9.63%
100	11.81%	11.67%	9.68%	9.77%	9.56%	9.61%	9.63%	9.63%
90	11.88%	11.69%	9.68%	9.80%	9.57%	9.61%	9.63%	9.63%
80	11.85%	11.62%	9.68%	9.80%	9.58%	9.60%	9.63%	9.64%
70	11.90%	11.53%	9.68%	9.77%	9.57%	9.60%	9.64%	9.64%
60	11.93%	11.59%	9.68%	9.74%	9.57%	9.60%	9.63%	9.64%
50	11.99%	11.47%	9.68%	9.74%	9.60%	9.60%	9.63%	9.64%
40	12.09%	11.69%	9.68%	9.76%	9.62%	9.61%	9.63%	9.64%
30	12.00%	11.59%	9.68%	9.74%	9.68%	9.63%	9.63%	9.63%
20	11.58%	11.42%	9.69%	9.86%	9.70%	9.69%	9.67%	9.66%
10	11.74%	11.52%	9.71%	10.05%	9.90%	9.78%	9.72%	9.70%

Table C.21: Third experiment: the *variation index* for the base learners and simple average on the 15 size ensemble obtained using Mor06-avg.

a&ps **4**	*a&ps* **5**	*a&ps* **35**	*a&ps* **36**	*a&ps* **39**	**Avg**
11.31%	10.90%	11.39%	11.56%	12.06%	9.59%
a&ps **41**	*a&ps* **48**	*a&ps* **71**	*a&ps* **72**	*a&ps* **75**	
12.73%	10.45%	11.18%	12.70%	11.02%	
a&ps **77**	*a&ps* **90**	*a&ps* **96**	*a&ps* **98**	*a&ps* **120**	
10.03%	10.14%	10.91%	10.74%	10.87%	

Table C.22: Third experiment: the *variation index* for the knn-CART on the 15 size ensemble obtained using Mor06-avg.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
∞	9.69%	9.50%	9.55%	9.41%	9.46%	9.52%	9.56%	9.55%
200	9.70%	9.51%	9.55%	9.41%	9.47%	9.53%	9.56%	9.55%
100	9.76%	9.48%	9.55%	9.57%	9.51%	9.53%	9.56%	9.56%
90	9.77%	9.56%	9.55%	9.58%	9.51%	9.53%	9.56%	9.56%
80	9.82%	9.62%	9.55%	9.59%	9.50%	9.53%	9.56%	9.56%
70	10.03%	9.71%	9.55%	9.63%	9.49%	9.54%	9.56%	9.56%
60	9.90%	9.70%	9.55%	9.62%	9.47%	9.54%	9.55%	9.56%
50	9.78%	9.80%	9.55%	9.64%	9.48%	9.53%	9.55%	9.56%
40	10.19%	10.01%	9.55%	9.70%	9.48%	9.54%	9.54%	9.55%
30	10.32%	10.19%	9.56%	9.78%	9.49%	9.53%	9.56%	9.55%
20	10.44%	10.34%	9.56%	9.90%	9.50%	9.55%	9.55%	9.56%
10	10.67%	10.44%	9.57%	10.11%	9.58%	9.56%	9.58%	9.59%

Table C.23: Third experiment: the *variation index* for the knn-CART on the 20 size ensemble obtained using Mor06-smse.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
∞	11.67%	11.70%	9.75%	9.95%	9.61%	9.66%	9.70%	9.70%
200	11.78%	11.65%	9.76%	9.86%	9.62%	9.66%	9.70%	9.70%
100	11.79%	11.61%	9.76%	9.85%	9.63%	9.68%	9.70%	9.71%
90	11.84%	11.65%	9.76%	9.84%	9.63%	9.68%	9.70%	9.71%
80	11.86%	11.71%	9.76%	9.83%	9.64%	9.68%	9.70%	9.71%
70	11.86%	11.62%	9.75%	9.79%	9.63%	9.67%	9.69%	9.71%
60	11.84%	11.54%	9.75%	9.73%	9.63%	9.66%	9.69%	9.71%
50	11.94%	11.54%	9.75%	9.73%	9.67%	9.66%	9.69%	9.70%
40	12.11%	11.70%	9.75%	9.73%	9.69%	9.67%	9.69%	9.70%
30	11.82%	11.57%	9.76%	9.79%	9.72%	9.70%	9.70%	9.70%
20	11.88%	11.62%	9.77%	9.85%	9.75%	9.77%	9.73%	9.74%
10	12.15%	11.98%	9.79%	10.10%	9.91%	9.84%	9.79%	9.78%

Table C.24: Third experiment: the *variation index* for the base learners and simple average on the 20 size ensemble obtained using Mor06-avg.

a&ps **3**	*a&ps* **4**	*a&ps* **5**	*a&ps* **35**	*a&ps* **36**	**Avg**
11.22%	11.31%	10.90%	11.40%	11.56%	9.62%
a&ps **39**	*a&ps* **41**	*a&ps* **44**	*a&ps* **48**	*a&ps* **65**	
12.06%	12.73%	11.08%	10.45%	10.20%	
a&ps **71**	*a&ps* **72**	*a&ps* **75**	*a&ps* **76**	*a&ps* **77**	
11.18%	12.70%	11.02%	12.92%	10.03%	
a&ps **90**	*a&ps* **95**	*a&ps* **96**	*a&ps* **98**	*a&ps* **120**	
10.14%	11.06%	10.91%	10.74%	10.87%	

Table C.25: Third experiment: the *variation index* for the knn-CART on the 20 size ensemble obtained using Mor06-avg.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
∞	9.76%	9.54%	9.60%	9.44%	9.50%	9.59%	9.61%	9.61%
200	9.74%	9.57%	9.60%	9.43%	9.51%	9.59%	9.61%	9.61%
100	9.77%	9.53%	9.61%	9.62%	9.56%	9.60%	9.62%	9.62%
90	9.80%	9.61%	9.61%	9.63%	9.55%	9.59%	9.62%	9.62%
80	9.86%	9.70%	9.61%	9.64%	9.56%	9.59%	9.62%	9.62%
70	9.96%	9.73%	9.61%	9.67%	9.54%	9.60%	9.62%	9.61%
60	9.89%	9.76%	9.61%	9.68%	9.53%	9.59%	9.61%	9.62%
50	9.77%	9.85%	9.61%	9.71%	9.53%	9.59%	9.61%	9.62%
40	10.16%	10.04%	9.61%	9.79%	9.54%	9.59%	9.60%	9.61%
30	10.40%	10.39%	9.62%	9.92%	9.55%	9.60%	9.62%	9.61%
20	10.99%	10.77%	9.62%	9.97%	9.58%	9.61%	9.62%	9.63%
10	11.16%	10.90%	9.63%	10.02%	9.67%	9.63%	9.65%	9.65%

Table C.26: Third experiment: the *variation index* for the knn-CART on the 25 size ensemble obtained using Mor06-smse.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
∞	11.41%	11.20%	9.76%	9.54%	9.58%	9.65%	9.67%	9.68%
200	11.41%	11.19%	9.76%	9.57%	9.59%	9.65%	9.68%	9.68%
100	11.42%	11.28%	9.76%	9.62%	9.59%	9.66%	9.68%	9.69%
90	11.43%	11.28%	9.76%	9.62%	9.58%	9.66%	9.68%	9.69%
80	11.43%	11.27%	9.76%	9.62%	9.59%	9.66%	9.68%	9.69%
70	11.38%	11.24%	9.76%	9.62%	9.58%	9.65%	9.68%	9.69%
60	11.46%	11.27%	9.76%	9.61%	9.57%	9.64%	9.67%	9.69%
50	11.43%	11.23%	9.76%	9.63%	9.60%	9.64%	9.68%	9.68%
40	11.55%	11.25%	9.76%	9.60%	9.61%	9.65%	9.68%	9.69%
30	11.45%	11.29%	9.76%	9.67%	9.65%	9.68%	9.69%	9.69%
20	11.50%	11.27%	9.77%	9.74%	9.65%	9.71%	9.71%	9.71%
10	11.70%	11.63%	9.77%	9.93%	9.72%	9.72%	9.70%	9.71%

Table C.27: Third experiment: the *variation index* for the base learners and simple average on the 25 size ensemble obtained using Mor06-avg.

a&ps **3**	*a&ps* **4**	*a&ps* **5**	*a&ps* **6**	*a&ps* **35**	**Avg**
11.22%	11.31%	10.90%	11.03%	11.41%	9.56%
a&ps **36**	*a&ps* **37**	*a&ps* **39**	*a&ps* **41**	*a&ps* **44**	
11.54%	12.82%	12.06%	12.73%	11.08%	
a&ps **48**	*a&ps* **65**	*a&ps* **71**	*a&ps* **72**	*a&ps* **73**	
10.45%	10.20%	11.18%	12.70%	9.82%	
a&ps **75**	*a&ps* **76**	*a&ps* **77**	*a&ps* **90**	*a&ps* **95**	
11.02%	12.92%	10.03%	10.14%	11.06%	
a&ps **96**	*a&ps* **98**	*a&ps* **101**	*a&ps* **120**	*a&ps* **124**	
10.91%	10.74%	9.86%	10.87%	10.57%	

Table C.28: Third experiment: the *variation index* for the knn-CART on the 25 size ensemble obtained using Mor06-avg.

k	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
∞	9.67%	9.50%	9.55%	9.41%	9.46%	9.53%	9.55%	9.55%
200	9.65%	9.53%	9.55%	9.41%	9.46%	9.53%	9.55%	9.55%
100	9.69%	9.50%	9.55%	9.57%	9.49%	9.54%	9.56%	9.56%
90	9.71%	9.57%	9.55%	9.58%	9.49%	9.54%	9.56%	9.56%
80	9.79%	9.67%	9.55%	9.60%	9.50%	9.54%	9.56%	9.56%
70	9.84%	9.70%	9.55%	9.62%	9.49%	9.55%	9.56%	9.55%
60	9.86%	9.73%	9.55%	9.62%	9.48%	9.54%	9.56%	9.56%
50	9.84%	9.82%	9.56%	9.66%	9.50%	9.54%	9.56%	9.56%
40	10.11%	10.00%	9.56%	9.74%	9.51%	9.55%	9.54%	9.55%
30	10.41%	10.42%	9.56%	9.85%	9.51%	9.56%	9.57%	9.56%
20	11.00%	10.76%	9.57%	9.93%	9.55%	9.57%	9.57%	9.57%
10	11.08%	10.90%	9.58%	10.00%	9.64%	9.59%	9.60%	9.61%

Appendix D

Additional results on ensembling for evaluation

This appendix has the complete preliminary results for the ensemble approach described in Chap. 8. It comprises the three phases of the ensemble process: ensemble generation, ensemble pruning and ensemble integration.

D.1 Ensemble generation

The generated pool (pool 128, table D.1) is quite similar to pool 130 presented in Appendix B. The only differences are: (1) the substitution of the 18 *a&ps* using PPR-supsmu without example selection (the *a&ps* from 113 to 130 having ES='all') for 16 *a&ps* using SVM-sigmoid with the CART approach for example selection (the *a&ps* from 93 to 108 having ES='ln'); and (2) the use of the test values 0.00006, 0.000018 and 0.00003 instead of 4000, 64000 and 1024000 for Par 1 using SVM-radial for ES='ln'. ES identifies the used example selection method, as described in Sect. 5.7.1. WDDT identifies the data type used for the variable weekday, as discussed in Sect. 5.7.2. The parameters Par 1, Par 2, Par 3 and Par 4 should be read as follows:

- For SVM: Par 1 = C, Par 2 = ν, Par 3 = γ and Par 4 = coef0;
- For RF: Par 1 = mtry;
- For PPR-supsmu: Par 1 = nterms, Par 2 = optlevel, Par 3 = bass and Par 4 = span;
- For PPR-spline: Par 1 = nterms, Par 2 = optlevel and Par 3 = df;
- For PPR-gcvspline: Par 1 = nterms, Par 2 = optlevel and Par 3 = gcvpen.

Table D.1: The pool 128.

A&ps	**Algorithm**	**ES**	**WDDT**	**Par 1**	**Par 2**	**Par 3**	**Par 4**
1	svm radial	all	Symbolic	4000	0.2	0.00006	
2	svm radial	all	Symbolic	64000	0.2	0.00006	

continued from previous page

a&ps	**Algorithm**	**ES**	**WDDT**	**Par 1**	**Par 2**	**Par 3**	**Par 4**
3	svm radial	all	Symbolic	1024000	0.2	0.00006	
4	svm radial	all	Symbolic	4000	0.6	0.00006	
5	svm radial	all	Symbolic	64000	0.6	0.00006	
6	svm radial	all	Symbolic	1024000	0.6	0.00006	
7	svm radial	all	Symbolic	4000	0.2	0.00018	
8	svm radial	all	Symbolic	64000	0.2	0.00018	
9	svm radial	all	Symbolic	1024000	0.2	0.00018	
10	svm radial	all	Symbolic	4000	0.6	0.00018	
11	svm radial	all	Symbolic	64000	0.6	0.00018	
12	svm radial	all	Symbolic	1024000	0.6	0.00018	
13	RF	ed	Symbolic	1			
14	RF	ed	Symbolic	3			
15	svm radial	ln	Symbolic	0.00006	0.2	0.00006	
16	svm radial	ln	Symbolic	0.00018	0.2	0.00006	
17	svm radial	ln	Symbolic	0.00030	0.2	0.00006	
18	svm radial	ln	Symbolic	0.00006	0.6	0.00006	
19	svm radial	ln	Symbolic	0.00018	0.6	0.00006	
20	svm radial	ln	Symbolic	0.00030	0.6	0.00006	
21	svm radial	ln	Symbolic	0.00006	0.2	0.00018	
22	svm radial	ln	Symbolic	0.00018	0.2	0.00018	
23	svm radial	ln	Symbolic	0.00030	0.2	0.00018	
24	svm radial	ln	Symbolic	0.00006	0.6	0.00018	
25	svm radial	ln	Symbolic	0.00018	0.6	0.00018	
26	svm radial	ln	Symbolic	0.00030	0.6	0.00018	
27	ppr supsmu	all	Numeric	1	1	0	0
28	ppr supsmu	all	Numeric	4	1	0	0
29	ppr supsmu	all	Numeric	1	2	0	0
30	ppr supsmu	all	Numeric	4	2	0	0
31	ppr supsmu	all	Numeric	1	3	0	0
32	ppr supsmu	all	Numeric	4	3	0	0
33	ppr supsmu	all	Numeric	1	1	5	0
34	ppr supsmu	all	Numeric	4	1	5	0
35	ppr supsmu	all	Numeric	1	2	5	0
36	ppr supsmu	all	Numeric	4	2	5	0
37	ppr supsmu	all	Numeric	1	3	5	0
38	ppr supsmu	all	Numeric	4	3	5	0
39	ppr supsmu	all	Numeric	1	1	10	0
40	ppr supsmu	all	Numeric	4	1	10	0
41	ppr supsmu	all	Numeric	1	2	10	0
42	ppr supsmu	all	Numeric	4	2	10	0
43	ppr supsmu	all	Numeric	1	3	10	0
44	ppr supsmu	all	Numeric	4	3	10	0
45	ppr spline	all	Numeric	1	1	1	
46	ppr spline	all	Numeric	2	1	1	
47	ppr spline	all	Numeric	1	3	1	
48	ppr spline	all	Numeric	2	3	1	
49	ppr spline	all	Numeric	1	1	32	
50	ppr spline	all	Numeric	2	1	32	

continued from previous page

a&ps	**Algorithm**	**ES**	**WDDT**	**Par 1**	**Par 2**	**Par 3**	**Par 4**
51	ppr spline	all	Numeric	1	3	32	
52	ppr spline	all	Numeric	2	3	32	
53	ppr spline	all	Numeric	1	1	1024	
54	ppr spline	all	Numeric	2	1	1024	
55	ppr spline	all	Numeric	1	3	1024	
56	ppr spline	all	Numeric	2	3	1024	
57	ppr spline	ed	Numeric	1	1	1	
58	ppr spline	ed	Numeric	2	1	1	
59	ppr spline	ed	Numeric	1	3	1	
60	ppr spline	ed	Numeric	2	3	1	
61	ppr spline	ed	Numeric	1	1	32	
62	ppr spline	ed	Numeric	2	1	32	
63	ppr spline	ed	Numeric	1	3	32	
64	ppr spline	ed	Numeric	2	3	32	
65	ppr spline	ed	Numeric	1	1	1024	
66	ppr spline	ed	Numeric	2	1	1024	
67	ppr spline	ed	Numeric	1	3	1024	
68	ppr spline	ed	Numeric	2	3	1024	
69	ppr gcvspline	all	Numeric	1	1	0.0625	
70	ppr gcvspline	all	Numeric	2	1	0.0625	
71	ppr gcvspline	all	Numeric	1	3	0.0625	
72	ppr gcvspline	all	Numeric	2	3	0.0625	
73	ppr gcvspline	all	Numeric	1	1	16	
74	ppr gcvspline	all	Numeric	2	1	16	
75	ppr gcvspline	all	Numeric	1	3	16	
76	ppr gcvspline	all	Numeric	2	3	16	
77	ppr gcvspline	all	Numeric	1	1	4096	
78	ppr gcvspline	all	Numeric	2	1	4096	
79	ppr gcvspline	all	Numeric	1	3	4096	
80	ppr gcvspline	all	Numeric	2	3	4096	
81	ppr gcvspline	ed	Numeric	1	1	0.0625	
82	ppr gcvspline	ed	Numeric	2	1	0.0625	
83	ppr gcvspline	ed	Numeric	1	3	0.0625	
84	ppr gcvspline	ed	Numeric	2	3	0.0625	
85	ppr gcvspline	ed	Numeric	1	1	16	
86	ppr gcvspline	ed	Numeric	2	1	16	
87	ppr gcvspline	ed	Numeric	1	3	16	
88	ppr gcvspline	ed	Numeric	2	3	16	
89	ppr gcvspline	ed	Numeric	1	1	4096	
90	ppr gcvspline	ed	Numeric	2	1	4096	
91	ppr gcvspline	ed	Numeric	1	3	4096	
92	ppr gcvspline	ed	Numeric	2	3	4096	
93	svm sigmoid	ln	Symbolic	16000	0.2	0.000002	-1
94	svm sigmoid	ln	Symbolic	256000	0.2	0.000002	-1
95	svm sigmoid	ln	Symbolic	16000	0.6	0.000002	-1
96	svm sigmoid	ln	Symbolic	256000	0.6	0.000002	-1
97	svm sigmoid	ln	Symbolic	16000	0.2	0.000026	-1
98	svm sigmoid	ln	Symbolic	256000	0.2	0.000026	-1

continued from previous page

a&ps	**Algorithm**	**ES**	**WDDT**	**Par 1**	**Par 2**	**Par 3**	**Par 4**
99	svm sigmoid	ln	Symbolic	16000	0.6	0.000026	-1
100	svm sigmoid	ln	Symbolic	256000	0.6	0.000026	-1
101	svm sigmoid	ln	Symbolic	16000	0.2	0.000002	-2
102	svm sigmoid	ln	Symbolic	256000	0.2	0.000002	-2
103	svm sigmoid	ln	Symbolic	16000	0.6	0.000002	-2
104	svm sigmoid	ln	Symbolic	256000	0.6	0.000002	-2
105	svm sigmoid	ln	Symbolic	16000	0.2	0.000026	-2
106	svm sigmoid	ln	Symbolic	256000	0.2	0.000026	-2
107	svm sigmoid	ln	Symbolic	16000	0.6	0.000026	-2
108	svm sigmoid	ln	Symbolic	256000	0.6	0.000026	-2
109	svm linear	all	Symbolic	0.0625	0.1		
110	svm linear	all	Symbolic	64	0.1		
111	svm linear	all	Symbolic	0.0625	0.3		
112	svm linear	all	Symbolic	64	0.3		
113	svm linear	all	Symbolic	0.0625	0.5		
114	svm linear	all	Symbolic	64	0.5		
115	svm linear	all	Symbolic	0.0625	0.7		
116	svm linear	all	Symbolic	64	0.7		
117	svm linear	all	Symbolic	0.0625	0.9		
118	svm linear	all	Symbolic	64	0.9		
119	svm linear	ln	Symbolic	0.0625	0.1		
120	svm linear	ln	Symbolic	64	0.1		
121	svm linear	ln	Symbolic	0.0625	0.3		
122	svm linear	ln	Symbolic	64	0.3		
123	svm linear	ln	Symbolic	0.0625	0.5		
124	svm linear	ln	Symbolic	64	0.5		
125	svm linear	ln	Symbolic	0.0625	0.7		
126	svm linear	ln	Symbolic	64	0.7		
127	svm linear	ln	Symbolic	0.0625	0.9		
128	svm linear	ln	Symbolic	64	0.9		

D.2 Ensemble pruning

The ensembles selected from pool 128 are presented in Table D.2. They have sizes $k \in \{5, 10, 15, 20, 25\}$. Since the pruning phase uses the Forward Sequential Selection (FSS) algorithm (Sect. 6.3.3), the results are presented by blocks of 5 *a&ps*. The first five compose the ensembles of size 5, the first ten, the ensembles of size 10, and so on.

D.3 Ensemble integration

The results on the integration phase for the different tested routes have comparable results to the ones obtained in Sect. 7.5. The settings using 5 sized ensembles with the DWS function for integration were the best for all the 6 routes (table D.3).

Table D.2: Ensembles selected from pool 128 using FSS-avg.

205-1-1	**205-1-2**	**300-1-1**	**301-1-1**	**505-1-1**	**505-1-2**
13	13	13	13	13	14
14	14	14	14	14	29
29	37	36	38	71	37
71	51	104	121	119	95
127	127	128	125	121	103
31	31	80	54	15	13
36	32	95	85	18	31
51	55	122	101	31	34
96	71	123	103	83	67
125	94	127	123	123	104
18	29	18	15	16	28
30	35	55	36	79	35
38	72	67	93	85	55
75	93	101	96	125	96
104	96	125	104	127	125
19	38	19	34	21	27
24	56	38	87	24	33
76	79	72	93	36	63
79	84	87	96	72	121
123	95	96	104	93	127
6	15	24	18	6	18
27	27	52	50	25	32
32	69	93	86	29	36
64	86	103	94	37	79
121	101	121	119	91	123

Table D.3: The *variation index* for tuning ensemble integration using knn-CART.

Size	**Best**	**FSwR**	**DWS 10%**	**DWS 30%**	**DWS 50%**	**DWS 70%**	**DWS 90%**	**DW**
route 205-1-1								
5	9.65%	9.35%	8.91%	8.99%	8.89%	8.87%	8.90%	8.91%
10	9.67%	9.18%	9.12%	9.13%	9.09%	9.08%	9.11%	9.12%
15	9.68%	9.03%	9.14%	9.00%	9.06%	9.08%	9.11%	9.12%
20	9.61%	9.02%	9.00%	8.94%	8.95%	8.96%	8.98%	8.99%
25	9.61%	9.02%	9.10%	8.94%	9.00%	9.01%	9.05%	9.06%
route 205-1-2								
5	10.32%	9.75%	9.30%	9.53%	9.28%	9.28%	9.29%	9.29%
10	10.32%	9.81%	9.34%	9.53%	9.31%	9.33%	9.33%	9.33%
15	10.28%	9.79%	9.35%	9.53%	9.34%	9.34%	9.35%	9.35%
20	10.25%	9.78%	9.40%	9.56%	9.41%	9.41%	9.41%	9.41%
25	10.18%	9.79%	9.37%	9.53%	9.37%	9.37%	9.37%	9.37%
route 300-1-1								
5	7.72%	7.38%	7.02%	7.05%	7.03%	7.03%	7.00%	7.06%
10	9.15%	8.45%	7.13%	7.06%	7.09%	7.08%	7.07%	7.18%
15	9.19%	8.49%	7.16%	7.10%	7.11%	7.11%	7.13%	7.23%
20	9.18%	8.52%	7.16%	7.11%	7.09%	7.09%	7.15%	7.27%
25	8.95%	8.83%	7.17%	7.11%	7.09%	7.10%	7.25%	7.28%
route 301-1-1								
5	10.69%	10.50%	10.33%	10.53%	10.39%	10.38%	10.30%	10.35%
10	11.16%	11.12%	10.36%	10.35%	10.36%	10.31%	10.33%	10.35%
15	11.19%	11.16%	10.52%	10.37%	10.62%	10.45%	10.48%	10.50%
20	11.20%	11.14%	10.54%	10.47%	10.49%	10.45%	10.48%	10.52%
25	11.26%	11.15%	10.61%	10.51%	10.58%	10.55%	10.56%	10.60%
route 505-1-1								
5	13.30%	13.19%	12.19%	12.29%	12.19%	12.19%	12.19%	12.19%
10	13.21%	13.12%	12.24%	12.30%	12.23%	12.25%	12.23%	12.24%
15	12.85%	12.70%	12.24%	12.31%	12.22%	12.24%	12.23%	12.23%
20	14.11%	13.16%	12.27%	13.27%	13.17%	12.32%	12.27%	12.26%
25	14.14%	13.68%	12.30%	13.26%	13.17%	12.31%	12.28%	12.27%
route 505-1-2								
5	10.29%	10.07%	10.07%	10.07%	10.11%	10.12%	10.09%	10.08%
10	10.66%	10.37%	10.22%	10.26%	10.21%	10.25%	10.21%	10.22%
15	10.42%	10.56%	10.38%	10.41%	10.41%	10.42%	10.35%	10.35%
20	10.53%	10.57%	10.33%	10.39%	10.34%	10.35%	10.32%	10.30%
25	10.54%	10.55%	10.38%	10.41%	10.41%	10.42%	10.36%	10.35%

Appendix E

Additional results on statistical tests

In Sect. 8.4 multiple confidence intervals are presented according to a method described in [Hochberg and Tamhane, 1987; Feelders and Verkooijen, 1995]. This appendix complements the information given in the said section by presenting intermediate results for each of the studied six routes, namely, the sample average, the sample variance and the sample covariance of the quadratic error of the predictions. With respect to notation, as presented before, the error measure is the square error calculated as $\hat{f}e_i(\mathbf{x}) = \left(\hat{f}_i(\mathbf{x}) - f(\mathbf{x})\right)^2$. We denote the sample average of $\hat{f}e_i$ as $\overline{\hat{f}}e_i$, the sample variance of $\hat{f}e_i$ as Se_i^2, the sample covariance of $\hat{f}e_i$ and $\hat{f}e_j$ as $Se_{i,j}$, and the number of test examples as n. Formally:

$$\overline{\hat{f}}e_i(\mathbf{X}) = \frac{1}{n}\sum_{l=1}^{n}(\hat{f}e_i(\mathbf{x_l})), \tag{E.1}$$

$$Se_i^2(\mathbf{X}) = \frac{1}{n-1}\sum_{l=1}^{n}(\hat{f}e_i(\mathbf{x_l}) - \overline{\hat{f}}e_i(\mathbf{X}))^2, \tag{E.2}$$

$$Se_{i,j}(\mathbf{X}) = \frac{1}{n-1}\sum_{l=1}^{n}((\hat{f}e_i(\mathbf{x_l}) - \overline{\hat{f}}e_i(\mathbf{X})) \times (\hat{f}e_j(\mathbf{x_l}) - \overline{\hat{f}}e_j(\mathbf{X}))). \tag{E.3}$$

Using the schema presented in table E.2, the sample statistics used to calculate the multiple intervals of confidence for each of the six routes are presented in table E.1.

Table E.1: Sample statistics used to calculate the multiple intervals of confidence.

route 205-1-1

5510	STT	BL	EXP	BST	ENS
STT	318090	264435	256591	269183	253553
BL		376376	257062	262562	258194
EXP			252400	251409	243582
BST				273550	251794
ENS					251181
	160182	167411	93767	102424	91930

route 205-1-2

5372	STT	BL	EXP	BST	ENS
STT	336229	281187	269194	265661	272829
BL		364387	259387	257662	266934
EXP			253791	247857	250762
BST				268158	254492
ENS					257422
	153996	161578	91075	105409	88766

route 300-1-1

3668	STT	BL	EXP	BST	ENS
STT	1990331	2008647	1956359	1916022	1978994
BL		3057072	1986239	2114563	2027883
EXP			1936328	1910399	1955007
BST				2058783	1963578
ENS					2003779
	296225	633091	319699	434154	320702

route 301-1-1

3671	STT	BL	EXP	BST	ENS
STT	1904515	1832394	1826320	1836910	372943
BL		2453022	1768294	1810290	359269
EXP			1780432	1781596	362993
BST				1873530	374031
ENS					1824199
	336540	601533	340994	392503	337591

route 505-1-1

2160	STT	BL	EXP	BST	ENS
STT	400880	366719	362173	346161	349026
BL		601174	389044	359069	368764
EXP			399993	358085	361449
BST				365309	352278
ENS					351819
	157756	240368	127309	129510	114336

route 505-1-2

1505	STT	BL	EXP	BST	ENS
STT	369495	229356	233289	220763	220963
BL		458716	228397	219864	211412
EXP			227045	207741	204612
BST				258337	223466
ENS					219795
	239597	282239	102481	104358	93713

STT: Scheduled Travel Time; BL: BaseLine predictor; EXP: EXPert based predictor; BST: single BeST predictor; ENS: ENSemble predictor.

Table E.2: Schema used to present results on statistical validation.

n	STT	BL	EXP	BST	ENS
STT	Se_{STT}	$\sqrt{Se_{STT,BL}}$	$\sqrt{Se_{STT,EXP}}$	$\sqrt{Se_{STT,BST}}$	$\sqrt{Se_{STT,ENS}}$
BL	-	Se_{BL}	$\sqrt{Se_{BL,EXP}}$	$\sqrt{Se_{BL,BST}}$	$\sqrt{Se_{BL,ENS}}$
EXP	-	-	Se_{EXP}	$\sqrt{Se_{EXP,BST}}$	$\sqrt{Se_{EXP,ENS}}$
BST	-	-	-	Se_{BST}	$\sqrt{Se_{BST,ENS}}$
ENS	-	-	-	-	Se_{ENS}
	$\overline{\hat{f}e}_{STT}$	$\overline{\hat{f}e}_{BL}$	$\overline{\hat{f}e}_{EXP}$	$\overline{\hat{f}e}_{BST}$	$\overline{\hat{f}e}_{ENS}$

STT: Scheduled Travel Time; BL: BaseLine predictor; EXP: EXPert based predictor; BST: single BeST predictor; ENS: ENSemble predictor.

Bibliography

[Aha and Bankert, 1994] David W. Aha and Richard L. Bankert. Feature selection for case-based classification of cloud types: an empirical comparison. In *AAAI Workshop on Case-Based Reasoning*, pages 106–112. AAAI Press, 1994.

[Aha and Bankert, 1996] David W. Aha and Richard L. Bankert. A comparative evaluation of sequential feature selection algorithms. In Doug Fisher and Hans-J. Lenz, editors, *Learning from data*, volume chapter 4, pages 199–206. Springer-Verlag, New York, 1996.

[Aha, 1997] David W. Aha. Lazy learning. *Artificial Intelligence Review*, 11:7–10, 1997.

[Aksela, 2003] Matti Aksela. Comparison of classifier selection methods for improving committee performance. In *International Workshop on Multiple Classifier Systems*, volume LNCS 2709, pages 84–93. Springer, 2003.

[Al-Ani, 2005] Ahmed Al-Ani. Feature subset selection using ant colony optimization. *International Journal of Computational Intelligence*, 2(1):53–58, 2005.

[Arya *et al.*, 1998] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45(6):891–923, 1998.

[Atkeson *et al.*, 1997] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning. *Artificial Intelligence Review*, (11):11–71, 1997.

[Azevedo and Jorge, 2007] Paulo J. Azevedo and Alípio Mário Jorge. Iterative reordering of rules for building ensembles without relearning. In *International Conference on Dicovery Science*, volume LNCS 4755, pages 56–67. Springer, 2007.

[Babcock *et al.*, 2002] Brian Babcock, Mayur Datar, and Rajeev Motwani. Sampling from a moving window over streaming data. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 633–634, 2002.

[Bakker and Heskes, 2003] Bart Bakker and Tom Heskes. Clustering ensembles of neural network models. *Neural Networks*, 16(2):261–269, 2003.

[Baluja, 1994] Shumeet Baluja. Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Technical Report 163, Carnegie Melon University, 1994.

[Barnum *et al.*, 2008] Donald T. Barnum, Sonali Tandon, and Sue McNeil. Comparing the performance of urban bus transit bus routes after adjusting for the environment, using data envelopment analysis. Working paper GCP-08-05, Great Cities Institute, 2008.

[Basheer and Hajmeer, 2000] I. A. Basheer and M. Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43:3–31, 2000.

[Ben-Akiva *et al.*, 1992] Moshe Ben-Akiva, Giulio Cantarella, Ennio Cascetta, Johan de Ruiter, Joe Whitaker, and Eric Kroes. Real-time prediction of traffic congestion. In *International Conference on Vehicle Navigation & Information Systems*, pages 557–562, 1992.

[Benjamini and Igbaria, 1991] Yoav Benjamini and Magid Igbaria. Clustering categories for better prediction of computer resources utilizations. *Applied Statistics*, 40(2):295–307, 1991.

[Bentley, 1975] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[Bezerra *et al.*, 2005] George B. Bezerra, Tiago V. Barra, Leandro N. de Castro, and Fernando J. Von Zuben. Adaptive radius immune algorithm for data clustering. In *ICARIS*, volume LNCS 3627, pages 290–303, Banff - Canada, 2005. Springer.

[Blum and Langley, 1997] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[Bowman and Turnquist, 1981] Larry A. Bowman and Mark A. Turnquist. Service frequency, schedule reliability and passenger wait times at transit stops. *Transportation Research Part A*, 15:465–471, 1981.

[Breiman *et al.*, 1984] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and regression trees.* Chapman and Hall/CRC, 1984.

[Breiman, 1996a] Leo Breiman. Bagging predictors. *Machine Learning*, 26:123–140, 1996.

[Breiman, 1996b] Leo Breiman. Heuristics of instability and stabilization in model selection. *Annals of Statistics*, 24(6):2350–2383, 1996.

[Breiman, 1996c] Leo Breiman. Stacked regressions. *Machine Learning*, 24:49–64, 1996.

[Breiman, 2000] Leo Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40(3):229–242, 2000.

[Breiman, 2001a] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[Breiman, 2001b] Leo Breiman. Using iterated bagging to debias regressions. *Machine Learning*, 45(3):261–277, 2001.

[Breiman, 2003] Leo Breiman. Manual - setting up, using, and understanding random forests v4.0. Technical report, 2003.

[Brown *et al.*, 2005] Gavin Brown, Jeremy L. Wyatt, Rachel Harris, and Xin Yao. Diversity creation methods: a survey and categorisation. *Information Fusion*, 6:5–20, 2005.

[Brown, 2004] Gavin Brown. *Diversity in neural network ensembles*. Phd thesis, University of Birmingham - United Kingdom, 2004.

[Bulmer, 2003] Michael Bulmer. *Francis Galton: pioneer of heredity and biometry*. The Johns Hopkins University Press, 2003.

[Carey, 1998] Malachy Carey. Optimizing scheduled times, allowing for behavioural response. *Transportation Research Part B*, 32(5):329–342, 1998.

[Caruana *et al.*, 2004] Rich Caruana, Alexandru Niculescu-Mozil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *International Conference on Machine Learning*, 2004.

[Castro, 2008] Francisca Leite Castro. *O impacto da previsão dos tempos de viagem na melhoria do planeamento do escalamento dos motoristas*. Msc. thesis, University of Porto, Faculty of Engineering, 2008. In Portuguese.

[Ceder *et al.*, 2001] A. Ceder, B. Golany, and O. Tal. Creating bus timetables with maximal synchronization. *Transportation Research Part A*, 35:913–928, 2001.

[Ceder, 1986] Avishai Ceder. Methods for creating bus timetables. *Transportation Research Part A*, 21:59–83, 1986.

[Ceder, 2002] Avishai Ceder. Urban transit scheduling: framework, review and examples. *Journal of Urban Planning and Development*, 128(4):225–244, 2002.

[Chapman *et al.*, 2000] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth. Crisp-dm 1.0 step-by-step data mining guide. Technical report, SPSS, 2000.

[Cherkassky and Mulier, 1998] Vladimir Cherkassky and Filip Mulier. *Learning from data: Concepts, theory, and methods*. John Wiley and Sons, Inc., 1998.

[Chien and Kuchpudi, 2003] Steven Chien and Chandra Kuchpudi. Dynamic travel time prediction with real-time and historic data. *Journal of Transportation Engineering*, 129(6):608–616, 2003.

[Chipman *et al.*, 2007] Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bayesian ensemble learning. In *Advances in Neural Information Processing Systems*, volume 19, pages 265–272, 2007.

[Cleveland and Loader, 1996] William S. Cleveland and Catherine Loader. Smoothing by local regression: principles and methods. In W. Hardle and M. G. Schimek, editors, *Statistical theory and computational aspects of smoothing*. Physica Verlag, Heidelberg, 1996.

[Coelho and Von Zuben, 2006] Guilherme P. Coelho and Fernando J. Von Zuben. The influence of the pool of candidates on the performance of selection and combination techniques in ensembles. In *International Joint Conference on Neural Networks*, pages 10588–10595, 2006.

[Cost and Salzberg, 1993] Scott Cost and Steven Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10(1):57–78, 1993.

[Cukjati *et al.*, 2001] David Cukjati, Marko Robnik-Šikonja, Stanislav Reberšek, Igor Kononenko, and Damijan Miklavčič. Prognostic factors in the prediction of chronic wound healing by electrical stimulation. *Medical and Biological Engineering and Computing*, 39(5), 2001.

[Davis (ed.), 1991] Lawrence Davis (ed.). *Handbook of genetic algorithms*. Van Nostrand Reinhold, 1991.

[Deb, 1999] K. Deb. Evolutionary algorithms for multicriterion optimization in engineering design. In *EUROGEN*, pages 135–161, 1999.

[Dias, 2005] Maria Teresa Galvão Dias. *A new approach to the bus driver scheduling problem using multiobjective genetic algorithms*. Phd, Universidade do Porto, 2005.

[Didaci and Giacinto, 2004] Luca Didaci and Giorgio Giacinto. Dynamic classifier selection by adaptive k-nearest neighbourhood rule. In Fabio Roli, Josef Kittler, and Terry Windeatt, editors, *International Workshop on Multiple Classifier Systems*, volume LNCS 3077, pages 174–183, Cagliary - Italy, 2004. Springer.

[Diebold and Mariano, 1995] Francis X. Diebold and Roberto S. Mariano. Comparing predictive accuracy. *Journal of Business and Economic Statistics*, 13:253–265, 1995.

[Dietterich, 1997] Thomas G. Dietterich. Machine-learning research: four current directions. *AI Magazine*, 18(4):97–136, 1997.

[Dietterich, 1998] Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923, 1998.

[Dietterich, 2002] Thomas G. Dietterich. Ensemble learning. In M. A. Arbib, editor, *The handbook of brain theory and neural networks*. MIT Press, Cambridge, 2002.

[Dimitriadou *et al.*, 2006] Evgenia Dimitriadou, Kurt Hornik, Friedrich Leisch, David Meyer, and Andreas Weingessel. e1071: Misc functions of the department of statistics (e1071), tu wien, 2006. R package version 1.5-16.

[Doak, 1992] J. Doak. An evaluation of feature selection methods and their application to computer securiy. Technical Report CSE-92-18, University of California, department of computer science, 1992.

[Domeniconi and Yan, 2004] Carlotta Domeniconi and Bojun Yan. Nearest neighbor ensemble. In *International Conference on Pattern Recognition*, volume 1, pages 228–231, 2004.

[Domeniconi *et al.*, 2002] Carlotta Domeniconi, Jing Peng, and Dimitrios Gunopulos. Locally adaptive metric nearest-neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1281–1285, 2002.

[Domingos, 1997] Pedro Domingos. Why does bagging work? a bayesian account and its implications. In *International Conference on Knowledge Discovery and Data Mining*, pages 155–158. AAAI Press, 1997.

[Duarte and Moreira, 2007] Eva Duarte and João Mendes Moreira. Prototriana-ensemble: protótipo para previsão de tempos de viagem com 3 dias de antecedência utilizando múltiplos modelos. Technical report, University of Porto, Faculty of Engineering, 2007. In Portuguese, http://www.liaad.up.pt/˜amjorge/docs/Triana/Duarte07.pdf.

[Duarte and Moreira, 2008] Eva Duarte and João Mendes Moreira. Prototriana: protótipo para previsão de tempos de viagem. Tech. report, University of Porto, Faculty of Engineering, 2008. In Portuguese, http://www.liaad.up.pt/˜amjorge/docs/Triana/Duarte08b.pdf.

[Efron and Tibshirani, 1993] Bradley Efron and Robert J. Tibshirani. *An introduction to the bootstrap*. Monographs on statistics and applied probability. Chapman & Hall, 1993.

[Fan *et al.*, 2005] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using the second order information for training svm. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[Feelders and Verkooijen, 1995] A. Feelders and W. Verkooijen. Which methods learns most from the data? In *International Workshop on Artificial Intelligence and Statistics*, pages 219–225, 1995. Preliminary papers.

[Ferreira, 2005] José António Vasconcelos Ferreira. *Sistema de apoio à decisão para escalamento de tripulantes no transporte colectivo urbano*. Phd, Universidade do Porto, 2005. In Portuguese.

[Figueiredo, 2003] Mário A.T. Figueiredo. Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1150–1159, 2003.

[Frank and Pfahringer, 2006] Eibe Frank and Bernhard Pfahringer. Improving on bagging with input smearing. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 97–106. Springer, 2006.

[Freund and Schapire, 1996] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[Friedman and Stuetzle, 1981] Jerome H. Friedman and Werner Stuetzle. Projection pursuit regression. *Journal of American Statistical Regression*, 76(376):817–823, 1981.

[Friedman, 1991] J.H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–141, 1991.

[Friedman, 2001] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[Gama, 2004] João Gama. Functional trees. *Machine Learning*, 55:219–250, 2004.

[García-Pedrajas *et al.*, 2005] Nicolás García-Pedrajas, César Hervás-Martínez, and Domingo Ortiz-Boyer. Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE Transactions on Evolutionary Computation*, 9(3):271–302, 2005.

[Geman *et al.*, 1992] Stuart Geman, Elie Bienenstock, and Rene Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[Giacinto and Roli, 1997] Giorgio Giacinto and Fabio Roli. Adaptive selection of image classifiers. In *International Conference on Image Analysis and Processing*, volume LNCS 1310, pages 38–45, Florence - Italy, 1997. Springer.

[Giacinto and Roli, 2001] Giorgio Giacinto and Fabio Roli. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, 19(9):699–707, 2001.

[Glover and Laguna, 1997] F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Publishers, 1997.

[Gomes *et al.*, 2008] Fernanda Pereira Noronha Meneses Mendes Gomes, Jorge Rui Guimarães Freire de Sousa, João Rui de Sousa Simões Fernandes Marrana, Rui André Albuquerque Neiva da Costa Saraiva, and António Paulo da Costa Moreira de Sá. Relatório e contas 2007. Technical report, Sociedade de Transportes Colectivos do Porto, SA, 2008. In Portuguese.

[Gould *et al.*, 2008] Phillip G. Gould, Anne B. Koehler, J. Keith Ord, Ralph D. Snyder, Rob J. Hyndman, and Farshid Vahid-Araghi. Forecasting time series with multiple seasonal patterns. *European Journal of Operational Research*, 191:205–220, 2008.

[Granitto *et al.*, 2005] P.M. Granitto, P.F. Verdes, and H.A. Ceccatto. Neural network ensembles: evaluation of aggregation algorithms. *Artificial Intelligence*, 163(2):139–162, 2005.

[Groebner and Shannon, 1985] David F. Groebner and Patrick W. Shannon. *Business statistics*. Merrill, 1985.

[Guimaraes and Cabral, 1997] Rui Campos Guimaraes and Jose A. Sarsfield Cabral. *Estatística*. McGraw-Hill, 1997. In Portuguese.

[Guyon and Elisseeff, 2003] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[Hall, 2000] Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *International Conference on Machine Learning*, pages 359–366, 2000.

[Hashem and Schmeiser, 1995] Sherif Hashem and Bruce Schmeiser. Improving model accuracy using optimal linear combinations of trained neural networks. *IEEE Transactions on Neural Networks*, 6(3):792–794, 1995.

[Hashem, 1993] Sherif Hashem. *Optimal linear combinations of neural networks.* Phd thesis, Purdue University, 1993.

[Hastie and Tibshirani, 1996] Trevor Hastie and Robert Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):607–616, 1996.

[Hastie *et al.*, 2001] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: data mining, inference, and prediction.* Springer series in statistics. Springer, 2001.

[Hernández-Lobato *et al.*, 2006] Daniel Hernández-Lobato, Gonzalo Martínez-Muñoz, and Alberto Suárez. Pruning in ordered regression bagging ensembles. In *International Joint Conference on Neural Networks*, pages 1266–1273, Vancouver - Canada, 2006.

[Ho, 1998] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[Hochberg and Tamhane, 1987] Yosef Hochberg and Ajit C. Tamhane. *Multiple comparison procedures.* Wiley series in probability and mathematical statistics. Wiley, 1987.

[Hsu, 1996] Jason C. Hsu. *Multiple comparisons: theory and methods.* Chapman & Hall/CRC, 1996.

[Huang *et al.*, 2004] Y. Huang, P.J. McCullagh, and N.D. Black. Feature selection via supervised model construction. In *International Conference on Data Mining*, 2004.

[Huber, 1985] Peter J. Huber. Projection pursuit. *Annals of Statistics*, 13(2):435–475, 1985.

[Hulten *et al.*, 2001] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *SigKDD*, pages 97–106, San Francisco - USA, 2001.

[Inmon, 1996] W.H. Inmon. *Building the data warehouse.* Wiley, 1996.

[Islam *et al.*, 2003] Md. Monirul Islam, Xin Yao, and Kazuyuki Murase. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks*, 14(4):820–834, 2003.

[Jain and Zongker, 1997] Anil Jain and Douglas Zongker. Feature selection: evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.

[Jankowski and Grochowski, 2004] Norbert Jankowski and Marek Grochowski. Comparison of instances selection algorithms i. agorithms survey. In L. Rutkowski, J. Siekmann, R. Tadeusiewicz, and L.A. Zadeh, editors, *International Conference on Artificial Intelligence and Soft Computing*, volume LNCS 3070, pages 598–603, Zakopane - poland, 2004. Springer.

[Jin *et al.*, 2005] Xing Jin, Yufeng Deng, and Yixin Zhong. Mixture feature selection strategy applied in cancer classification from gene expression. In *IEEE Annual Conference on Enginnering in Medicine and Biology*, pages 4807–4809, Shanghai - China, 2005.

[Jin *et al.*, 2007] Xin Jin, Rongyan Li, Xian Shen, and Rongfang Bie. Automatic web pages categorization with relieff and hidden naive bayes. In *SAC*, pages 617–621, Seoul - Korea, 2007. ACM.

[John *et al.*, 1994] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*. Morgan Kaufmann Publishers, 1994.

[Jolliffe, 2002] I.T. Jolliffe. *Principal component analysis*. Springer Series in Statistics. Springer, second edition, 2002.

[Jorge and Azevedo, 2005] Alípio M. Jorge and Paulo J. Azevedo. An experiment with association rules and classification: post-bagging and conviction. In *Discovery Science*, volume LNCS 3735, pages 137–149, Singapore, 2005. Springer.

[Kalman, 1960] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82 (D):35–45, 1960.

[Karlin and Rinott, 1982] Samuel Karlin and Yosef Rinott. Applications of anova type decompositions for comparisons of conditional variance statistics including jackknife estimates. *The Annals of Statistics*, 10(2):485–501, 1982.

[Kaufman and Rousseeuw, 1990] Leonard Kaufman and Peter J. Rousseeuw. *Finding groups in data: An introduction to cluster analysis*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, 1990.

[Kim *et al.*, 2003] Hyun-Chul Kim, Shaoning Pang, Hong-Mo Je, Daijin Kim, and Sung-Yang Bang. Constructing support vector machine ensemble. *Pattern Recognition*, 36(12):2757–2767, 2003.

[Kindermann and Trappenberg, 1999] Lars Kindermann and Thomas P. Trappenberg. Modeling time-varying processes by unfolding the time domain. In *International Joint Conference on Neural Networks*, volume 4, pages 2600–2603, 1999.

[Kira and Rendell, 1992] Kenji Kira and Larry A. Rendell. A practical approach to feature selection. In *International Wokshop on Machine Learning*, pages 249–256. Morgan Kaufmann, 1992.

[Kivinen and Warmuth, 1997] Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

[Kleinbaum *et al.*, 1998] David G. Kleinbaum, Lawrence L. Kupper, Keith E. Muller, and Azhar Nizam. *Applied regression analysis and other multivariablte methods*. Duxbury Press, 1998.

[Klunder *et al.*, 2007] Gerdien Klunder, Peter Baas, and Frans op de Beek. A long-term travel time prediction algorithm using historical data. Technical report, TNO, 2007.

[Kohavi and John, 1997] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[Kolen and Pollack, 1990] John F. Kolen and Jordan B. Pollack. Back propagation is sensitive to initial conditions. Technical Report TR 90-JK-BPSIC, The Ohio State University, 1990.

[Kotsiantis and Pintelas, 2005] S.B. Kotsiantis and P.E. Pintelas. Selective averaging of regression models. *Annals of Mathematics, Computing & Teleinformatics*, 1(3):65–74, 2005.

[Krogh and Vedelsby, 1995] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. *Advances in Neural Information Processing Systems*, 7:231–238, 1995.

[Kuncheva, 2002] Ludmila I. Kuncheva. Switching between selection and fusion in combining classifiers: an experiment. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 32(2):146–156, 2002.

[Kuncheva, 2004] Ludmila I. Kuncheva. *Combining pattern classifiers*. Wiley, 2004.

[Lazarevic, 2001] Aleksandar Lazarevic. Effective pruning of neural network classifier ensembles. In *International Joint Conference on Neural Networks*, pages 796–801, 2001.

[LeBlanc and Tibshirani, 1996] Michael LeBlanc and Robert Tibshirani. Combining estimates in regression and classification. *Journal of the American Statistical Association*, 91:1641–1650, 1996.

[Liaw and Wiener, 2002] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[Lilliefors, 1967] Hebert W. Lilliefors. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 62(318):399–402, 1967.

[Lin and Li, 2005] Hsuan-Tien Lin and Ling Li. Infinite ensemble learning with support vector machines. In *European Conference on Machine Learning*, volume LNAI 3720, pages 242–254. Springer, 2005.

[Lint, 2004] J.W.C. Van Lint. *Reliable travel time prediction for freeways*. Phd thesis, Technische Universiteit Delft - Netherlands, 2004.

[Liu and Motoda, 2001] Huan (ed.) Liu and Hiroshi (ed.) Motoda. *Instance selection and construction for data mining.* Kluwer Academic Publishers, 2001.

[Liu and Motoda, 2002] Huan Liu and Hiroshi Motoda. On issues of instance selection. *Data Mining and Knowledge Discovery*, 6(2):115–130, 2002.

[Liu and Yao, 1999] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12:1399–1404, 1999.

[Liu *et al.*, 1998] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *KDD 1997*, pages 80–86, 1998.

[Liu *et al.*, 2000] Yong Liu, Xin Yao, and Tetsuya Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.

[Liu *et al.*, 2004] Huan Liu, Hiroshi Motoda, and Lei Yu. A selective sampling approach to active feature selection. *Artificial Intelligence*, 159(1-2):49–74, 2004.

[Loughrey and Cunningham, 2005] John Loughrey and Pádraig Cunningham. Using early stopping to reduce overfitting in wrapper-based feature weighting. Technical report TCD-CS-2005-41, Trinity College Dublin, Computer Science Department, 2005.

[Lourenço *et al.*, 2001] Helena R. Lourenço, José P. Paixão, and Rita Portugal. The crew-scheduling module in the gist system. Tech. Report UPF Economics Working Paper No. 547, Universitat Pompeu - Spain, 2001.

[Makridakis *et al.*, 1983] Spyros Makridakis, Steven C. Wheelwright, and Victor E. McGee. *Forecasting: methods and applications, 2nd edition.* Wiley, 1983.

[Makridakis *et al.*, 1998] Spyros Makridakis, C. Wheelwright, and Rob J. Hyndman. *Forecasting: methods and applications.* Wiley, 1998.

[Malerba *et al.*, 2004] Donato Malerba, Floriana Esposito, Michelangelo Ceci, and Annalisa Appice. Top-down induction of model trees with regression and splitting nodes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):612–625, 2004.

[Margineantu and Dietterich, 1997] Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting. In *International Conference on Machine Learning*, pages 211–218, 1997.

[Martínez-Muñoz and Suárez, 2006] Gonzalo Martínez-Muñoz and Alberto Suárez. Pruning in ordered bagging ensembles. In *International Conference on Machine Learning*, pages 609–616, 2006.

[Merz and Pazzani, 1999] Cristopher J. Merz and Michael J. Pazzani. A principal components approach to combining regression estimates. *Machine Learning*, 36:9–32, 1999.

[Merz, 1996] Cristopher J. Merz. Dynamical selection of learning algorithms. In D. Fisher and H.-J. Lenz, editors, *International Workshop on Artificial Intelligence and Statistics*, volume Learning from Data: Artificial Intelligence and Statistics V. Springer-Verlag, 1996.

[Merz, 1998] Cristopher J. Merz. *Classification and regression by combining models*. Phd thesis, University of California - U.S.A., 1998.

[Meyer *et al.*, 2003] David Meyer, Friedrich Leisch, and Kurt Hornik. The support vector machine under test. *Neurocomputing*, 55(1-2):169–186, 2003.

[Minaei-Bigdoli *et al.*, 2004] Behrouz Minaei-Bigdoli, Alexander Topchy, and William F. Punch. Ensembles of partitions via data resampling. In *International Conference on Information Technology: Coding and Computing*, volume 2, pages 188–191, 2004.

[Mitchell, 1997] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[Molina *et al.*, 2002] Luis Carlos Molina, Lluís Belanche, and Àngela Nebot. Feature selection algorithms: a survey and experimental evaluation. In *IEEE International Conference on Data Mining*, pages 306–313, 2002.

[Moreira *et al.*, 2006a] João M. Moreira, Alípio M. Jorge, Carlos Soares, and Jorge Freire Sousa. Improving svm-linear predictions using cart for example selection. In *International Symposium on Methodologies for Intelligent Systems*, volume LNAI 4203, pages 632–641. Springer, 2006.

[Moreira *et al.*, 2006b] João M. Moreira, Jorge Freire Sousa, Alípio M. Jorge, and Carlos Soares. An ensemble regression approach for bus trip time prediction. In *Meeting of the EURO Working Group on Transportation*, pages 317–321, 2006. http://www.liaad.up.pt/˜amjorge/docs/Triana/Moreira06b.pdf.

[Moreira *et al.*, 2007] João M. Moreira, Alípio M. Jorge, Carlos Soares, and Jorge Freire de Sousa. Ensemble learning: the end of the selection/fusion dichotomy. Technical report, 2007. http://www.liaad.up.pt/˜amjorge/docs/Triana/Moreira07a.pdf.

[Nadaraya, 1964] E. A. Nadaraya. On estimatig regression. *Theory of Probability and its Applications*, 9(1):141–142, 1964.

[Nakayama, 1997] Marvin K. Nakayama. Multiple-comparison procedures for steady-state simulations. *The Annals of Statistics*, 25(6):2433–2450, 1997.

[Newell, 1974] G.F. Newell. Control of pairing vehicles on a public transportation route, two vehicles, one control point. *Transportation Science*, 8(3):248–264, 1974.

[Nóvoa, 1994] Maria Henriqueta S. Nóvoa. *Previsão das vendas de um hipermercado - uma aplicação prática do modelo Holt-Winters*. Research thesis, Porto University, Portugal, 1994. In Portuguese.

[Nunes *et al.*, 2000] Nuno Jardim Nunes, Marco Toranzo, J. Falcão e Cunha, Jaelson Castro, Srdjan Krovacevic, Dave Roberts, Jean-Claude Tarby, Mark Collins-Cope, and Mark van Haemelen. Interactive system design with object models. In Ana Moreira and S: Demeyer, editors, *ECOOP'99 Workshop*, volume LNCS 1743, pages 267–287. Spinger Verlag, 2000.

[Oh *et al.*, 2004] Il-Seok Oh, Jin-Seon Lee, and Byung-Ro Moon. Hybrid genetic algorithms for feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1424–1437, 2004.

[Opitz and Shavlik, 1996] David W. Opitz and Jude W. Shavlik. Generating accurate and diverse members of a neural-network ensemble. *Advances in Neural Information Processing Systems*, 8:535–541, 1996.

[Opitz, 1999] David W. Opitz. Feature selection for ensembles. In *National Conference on Artificial Intelligence*, pages 379–384, Orlando - U.S.A., 1999. AAAI Press.

[Ortiz-Boyer *et al.*, 2005] Domingo Ortiz-Boyer, César Hervás-Martínez, and Nicolás García-Pedrajas. Cixl2: A crossover operator for evolutionary algorithms based on population features. *Journal of Artificial Intelligence Research*, 24:1–48, 2005.

[Palma and Lindsey, 2001] André de Palma and Robin Lindsey. Optimal timetables for public transportation. *Transportation Research Part B*, 35:789–813, 2001.

[Parmanto *et al.*, 1996] Bambang Parmanto, Paul W. Munro, and Howard R. Doyle. Reducing variance of committee prediction with resampling techniques. *Connection Science*, 8(3, 4):405–425, 1996.

[Partridge and Yates, 1996] D. Partridge and W. B. Yates. Engineering multiversion neural-ney systems. *Neural Computation*, 8(4):869–893, 1996.

[Perrone and Cooper, 1993] Michael P. Perrone and Leon N. Cooper. When networks disagree: ensemble methods for hybrid neural networks. In R.J. Mammone, editor, *Neural Networks for Speech and Image Processing*. Chapman-Hall, 1993.

[Petridis *et al.*, 2001] V. Petridis, A. Kehagias, L. Petrou, A. Bakirtzis, N. Maslaris, S. Kiartzis, and H. Panagiotou. A bayesian multiple models combination method for time series prediction. *Journal of Intelligent and Robotic Systems*, 31(1-3):69–89, 2001.

[Picard and Cook, 1984] Richard R. Picard and R. Dennis Cook. Cross-validation of regression models. *Journal of the American Statistical Association*, 79(387):575–583, 1984.

[Pizarro *et al.*, 2002] Joaquín Pizarro, Elisa Guerrero, and Pedro L. Galindo. Multiple comparison procedures applied to model selection. *Neurocomputing*, 48:155–173, 2002.

[Porto, unknown] Metro do Porto. Company history, unknown. Retrieved July 12, 2008, from http://www.metrodoporto.pt.

[Prodromidis *et al.*, 1999] Andreas L. Prodromidis, Salvatore J. Stolfo, and Philip K. Chan. Effective and efficient pruning of meta-classifiers in a distributed data mining system. Technical Report CUCS-017-99, Columbia University, 1999.

[Pudil *et al.*, 1994] P. Pudil, F.J. Ferri, J. Novovicova, and J. Kittler. Floating search methods for feature selection with nonmonotonic criterion functions. In *IEEE International Conference on Pattern Recognition*, volume 11, pages 279–283, 1994.

[Puuronen *et al.*, 1999] Seppo Puuronen, Vagan Terziyan, and Alexey Tsymbal. A dynamic integration algorithm for an ensemble of classifiers. In *International Symposium on Methodologies for Intelligent Systems*, volume LNCS 1609, pages 592–600. Springer, 1999.

[Quinlan, 1992] J. Ross Quinlan. Learning with continuous classes. In Adams and Sterling, editors, *Australian Joint Conference on Artificial Intelligence*, pages 343–348, 1992.

[Quinlan, 1993] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, 1993.

[Ranawana and Palade, 2006] Romesh Ranawana and Vasile Palade. Multi-classifier systems: review and a roadmap for developers. *International Journal of Hybrid Intelligent Systems*, 3(1):35–61, 2006.

[Raviv and Intrator, 1996] Yuval Raviv and Nathan Intrator. Bootstrapping with noise: an effective regularization technique. *Connection Science*, 8(3, 4):355–372, 1996.

[Reinartz, 2002] Thomas Reinartz. A unifying view on instance selection. *Data Mining and Knowledge Discovery*, 6(2):191–210, 2002.

[Robnik-Šikonja and Kononenko, 2003] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine Learning*, 53(1-2):23–69, 2003.

[Robnik-Šikonja, 2004] Marko Robnik-Šikonja. Improving random forests. In *European Conference on Machine Learning*, volume LNAI 3201, pages 359–370, Poznan - Poland, 2004. Springer.

[Roli *et al.*, 2001] Fabio Roli, Giorgio Giacinto, and Gianni Vernazza. Methods for designing multiple classifier systems. In *International Workshop on Multiple Classifier Systems*, volume LNCS 2096, pages 78–87. Springer, 2001.

[Rooney and Patterson, 2007] Niall Rooney and David Patterson. A weighted combination of stacking and dynamic integration. *Pattern Recognition*, 40(4):1385–1388, 2007.

[Rooney *et al.*, 2004] Niall Rooney, David Patterson, Sarab Anand, and Alexey Tsymbal. Dynamic integration of regression models. In *International Workshop on Multiple Classifier Systems*, volume LNCS 3181, pages 164–173. Springer, 2004.

[Rosen, 1996] Bruce E. Rosen. Ensemble learning using decorrelated neural networks. *Connection Science*, 8(3, 4):373–383, 1996.

[Rudnicki *et al.*, 2006] Witold R. Rudnicki, Marcin Kierczak, Jacek Koronacki, and Jan Komorowski. A statistical method for determining importance of variables in an information system. In *International Conference on Rough Sets and Current Trends in Computing*, volume LNAI 4259, pages 557–566. Springer-Verlag, 2006.

[Ruta and Gabrys, 2001] Dymitr Ruta and Bogdan Gabrys. Application of the evolutionary algorithms for classifier selection in multiple classifier systems with majority voting. In *International Workshop on Multiple Classifier Systems*, volume LNCS 2096, pages 399–408. Springer, 2001.

[Saito and Nakano, 2002] Kazumi Saito and Ryohei Nakano. Extracting regression rules from neural networks. *Neural networks*, 15(10):1279–1288, 2002.

[Salzberg, 1997] Steven L. Salzberg. On comparing classifiers: pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1(3):317–327, 1997.

[Salzborn, 1972] Franz J.M. Salzborn. Optimum bus scheduling. *Transportation Science*, 6(2):137–147, 1972.

[Schapire, 1990] R. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[Scholkopf *et al.*, 2000] Bernhard Scholkopf, Alex J. Smola, Robert C. Williamson, and Peter L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.

[Shearer, 2000] Colin Shearer. The crisp-dm model: The new blueprint for data mining. *Journal of Data Warehousing*, 5(6):13–22, 2000.

[Skalak, 1994] David B Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *International Conference on Machine Learning*, pages 293–301. Morgan Kaufmann, 1994.

[Smola and Scholkopf, 2004] Alex J. Smola and Bernhard Scholkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

[Somol *et al.*, 1999] P. Somol, P. Pudil, J. Novovicova, and P. Paclik. Adaptive floating search methods in feature selection. *Pattern Recognition Letters*, 20(11-13):1157–1163, 1999.

[Sorokina *et al.*, 2007] Daria Sorokina, Rich Caruana, and Mirek Riedewald. Additive groves of regression trees. In *European Conference on Machine Learning*, volume LNAI 4701, pages 323–334. Springer-Verlag, 2007.

[Stark and Parker, 1995] P. Stark and R. Parker. Bounded-variable least squares: an algorithm and applications. *Computational Statistics*, 10(2):129–141, 1995.

[Stearns, 1976] S.D. Stearns. On selecting features for pattern classifiers. In *International Conference on Pattern Recognition*, pages 71–75, 1976.

[Stone, 1974] M. Stone. Cross-validatory choice ans assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B*, 36(2):111–147, 1974.

[Strathman *et al.*, 1998] James G. Strathman, Kenneth J. Dueker, Thomas Kimpel, Rick Gerhart, Ken Turner, Pete Taylor, Steve Callas, David Griffin, and Janet Hopper. Automated bus dispatching, operations control and service reliability: analysis of tri-met baseline service date. Technical report, University of Washington - U.S.A., 1998.

[Strathman *et al.*, 2001] James G. Strathman, Thomas Kimpel, Kenneth J. Dueker, Rick Gerhart, Ken Turner, David Griffin, and Steve Callas. Bus transit operations control: review and an experiment involving tri-met's automated bus dispatching system. *Journal of Public Transportation*, 4(1):1–26, 2001.

[Strehl and Ghosh, 2003] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2003.

[Tamon and Xiang, 2000] Christino Tamon and Jie Xiang. On the boosting pruning problem. In *European Conference on Machine Learning*, volume LNCS 1810, pages 404–412. Springer, 2000.

[Tan *et al.*, 2008] Chao Tan, Menglong Li, and Xin Qin. Random subspace regression ensemble for near-infrared spectroscopic calibration of tobacco samples. *Analytical Sciences*, 24:647–653, 2008.

[Tao and Tang, 2004] Dacheng Tao and Xiaoou Tang. Svm-based relevance feedback using random subspace method. In *IEEE International Conference on Multimedia and Expo*, pages 269–272, 2004.

[Team, 2006] R Development Core Team. R: A language and environment for statistical computing. Technical report, R Foundation for Statistical Computing, 2006. ISBN 3-900051-07-0.

[Todorovski and Dzeroski, 2003] Ljupco Todorovski and Saso Dzeroski. Combining classifiers with meta decision trees. *Machine Learning*, 50(3):223–249, 2003.

[Torgo, 1995] Luis Torgo. Data fiting with rule-based regression. In J. Zizka and P. Brazdil, editors, *International Workshop on Artificial Intelligence Techniques*, Brno - Czech Republic, 1995. Also available in http://www.ncc.up.pt/~ltorgo/Papers/list_pub.html.

[Torgo, 1999] Luis Torgo. *Inductive learning of tree-based regression models.* Phd thesis, Porto University - Portugal, 1999.

[Torgo, unknown] Luis Torgo. Regression data repository, unknown. Retrieved October 20, 2005, from http://www.liacc.up.pt/~ltorgo/Regression/DataSets.html.

[Tresp and Taniguchi, 1995] Volker Tresp and Michiaki Taniguchi. Combining estimators using non-constant weighting functions. *Advances in Neural Information Processing Systems*, 7:419–426, 1995.

[Tsang *et al.*, 2005] Ivor W. Tsang, James T. Kwok, and Kimo T. Lai. Core vector regression for very large regression problems. In *International Conference on Machine Learning*, pages 912–919, 2005.

[Tsang *et al.*, 2006] Ivor W. Tsang, Andras Kocsor, and James T. Kwok. Diversified svm ensembles for large data sets. In *European Conference on Machine Learning*, volume LNAI 4212, pages 792–800. Springer, 2006.

[Tsymbal and Puuronen, 2000] Alexey Tsymbal and Seppo Puuronen. Bagging and boosting with dynamic integration of classifiers. In *Principles of Data Mining and Knowledge Discovery*, volume LNAI 1910, pages 116–125. Springer, 2000.

[Tsymbal *et al.*, 2006] Alexey Tsymbal, Mykola Pechenizkiy, and Pádraig Cunningham. Dynamic integration with random forests. Technical Report TCD-CS-2006-23, The University of Dublin, Trinity College, 2006.

[Tsymbal *et al.*, 2008] Alexey Tsymbal, Mykola Pechenizkiy, Pádraig Cunningham, and Seppo Puuronen. Dynamic integration of classifiers for handling concept drift. *Information Fusion*, 9(1):56–68, 2008.

[Turner *et al.*, 1998] Shawn M. Turner, William L. Eisele, Robert J. Benz, and Douglas J. Holdener. Travel time data collection handbook. Technical Report FHWA-PL-98-035, Texas Transportation Institute, 1998.

[Ueda and Nakano, 1996] Naonori Ueda and Ryohei Nakano. Generalization error of ensemble estimators. In *IEEE International Conference on Neural Networks*, volume 1, pages 90–95, 1996.

[Vafaie and Jong, 1993] Haleh Vafaie and Kenneth De Jong. Robust feature selection algorithms. In *IEEE Conference on Tools for Artificial Intelligence*, pages 356–363, 1993.

[Vapnik, 1995] Vladimir N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag, 1995.

[Verikas *et al.*, 1999] Antanas Verikas, Arunas Lipnickas, Kerstin Malmqvist, Marija Becauskiene, and Adas Gelzinis. Soft combining of neural classifiers: a comparative study. *Pattern Recognition Letters*, 20(4):429–444, 1999.

[Vuchic, 2005] Vukan R. Vuchic. *Urban transit: operations, planning, and economics.* Wiley, 2005.

[Wang *et al.*, 2003] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *ACM International Conference on Knowledge Discovery and Data Mining*, 2003.

[Wang *et al.*, 2006] Peng Wang, Jie (Jane) Lin, and Darold T. Barnum. Data envelopment analysis of bus service reliability using automatic vehicle location data. In *Annual Meeting of Transportation Research Board*, 2006.

[Watson, 1964] Geoffrey S. Watson. Smooth regression analysis. *Sankhya: The Indian Journal of Statistics, Series A*, 26:359–372, 1964.

[Webb and Zheng, 2004] Geoffrey I. Webb and Zijian Zheng. Multistrategy ensemble learning: reducing error by combining ensemble learning techniques. *IEEE Transactions on Knowledge and Data Engineering*, 16(8):980–991, 2004.

[Webb, 1995] Brandyn Webb. Inductive learning, 1995. Retrieved February 11, 2008, from http://sifter.org/˜brandyn/InductiveLearning.html.

[Weiss and Indurkhya, 1995] Sholom M. Weiss and Nitin Indurkhya. Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, (3):383–403, 1995.

[Wichard *et al.*, 2003] Jörg Wichard, Christian Merkwirth, and Maciej Ogorzalek. Building ensembles with heterogeneous models. In *Course of the International School on Neural Nets*, Salerno - Italy, 2003.

[Wilson and Martinez, 1997] D. Randall Wilson and Tony R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.

[Witten and Frank, 2000] Ian H. Witten and Eibe Frank. *Data Mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann Publishers, 2000.

[Wolpert, 1992] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[Woods, 1997] Kevin Woods. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405–410, 1997.

[Xu and Fern, 2007] Yuehua Xu and Alan Fern. On learning linear ranking functions for beam search. In *International Conference on Machine Learning*, pages 1047–1054, Oregon - U.S.A., 2007.

[Yang and Honavar, 1997] Jihoon Yang and Vasant Honavar. Feature subset selection using a genetic algorithm. *IEEE Transactions on Intelligent Systems*, 13(2):44–49, 1997.

[Yankov *et al.*, 2006] Dragomir Yankov, Dennis DeCoste, and Eamonn Keogh. Ensembles of nearest neighbor forecasts. In *European Conference on Machine Learning*, volume LNAI 4212, pages 545–556. Springer, 2006.

[Yu *et al.*, 2007] Yang Yu, Zhi-Hua Zhou, and Kay Ming Ting. Cocktail ensemble for regression. In *IEEE International Conference on Data Mining*, pages 721–726, Omaha NE - U.S.A., 2007.

[Zenobi and Cunningham, 2001] Gabriele Zenobi and Pádraig Cunningham. Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. In *European Conference on Machine Learning*, volume LNCS 2167, pages 576–587. Springer, 2001.

[Zhao *et al.*, 2006] Jiamin Zhao, Maged Dessouky, and Satish Bukkapatnam. Optimal slack time for schedule-based transit operations. *Transportation Science*, 40(4):529–539, 2006.

[Zhou and Tang, 2003] Zhi-Hua Zhou and Wei Tang. Selective ensemble of decision trees. In *International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, volume LNAI 2639, pages 476–483. Springer, 2003.

[Zhou *et al.*, 2002] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial Intelligence*, 137:239–263, 2002.