FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Multiboosting for Regression

Nuno Miguel Rainho Valente

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Prof. Dr. João Pedro Carvalho Leal Mendes Moreira

February 16, 2020

Multiboosting for Regression

Nuno Miguel Rainho Valente

Mestrado Integrado em Engenharia Informática e Computação

February 16, 2020

Abstract

Ensemble learning refers to the procedures used to train various machine learning systems and to combine their results by treating them as if they belonged to a committee of decision-makers. The final decision is done by combining the decisions of each element. And this can bring the power to minimize the errors if, eventually, the decision was made by just one element. So, the committee can, for example, minimize the selection of a bad product, a worker with no skills to make the job, a general task taken unsuccessful.

We recognize that in real-world situations, all models have limitations and there will be errors. Thus, the aim of *ensemble learning* is to find the strengths and weaknesses of the models, leading to better decision making in general. Several theoretical and empirical results show that the accuracy of a set can significantly exceed the one of a single model. The principle of combining forecasts has been of interest to several areas of study over many years.

A cross-cutting issue in this thematic is the existence of a trade-off between the bias and variance of the data set. While bias refers to precision, the variance tells us how sensitive the data are to small changes. The problem arises because, when trying to increase the variance, *bias* decreases and vice versa. It is fundamental to find a balance between these two components in order to find a predictive model of quality that fits the one we want to estimate. We will establish comparisons between bias reduction methods and those of variance reduction. This trade-off works like a benchmark of how robust is our model.

In ensemble learning what interests us is the diversity of predictions and not only the variance of a model in the face of different data. It is obvious that the two concepts are related because unstable algorithms generate models with high variance and simultaneously generate ensembles with high diversity because such variance exists.

One important idea here is that, although we recognize the importance and the motivations behind the trade-off already mentioned, we will focus on try to evaluate different techniques, combine some of them, across several datasets and look to their characteristics in order to contribute for a good ensemble, i.e., try to do even better predictions with ensemble learning.

Categories: *Computing methodologies $\rightarrow$ Machine learning $\rightarrow$ Machine learning algorithms $\rightarrow$ Ensemble methods*

Keywords: Ensemble learning, regression, multiple models

Resumo

Ensemble learning refere-se aos procedimentos usados para treinar várias máquinas de aprendizagem, *machine learning*, e combinar os seus resultados, tratando-os como se pertencessem a uma comissão de decisores. A decisão final é realizada através da combinação das decisões de cada elemento presente no comité. E assim pode trazer o poder de minimizar os erros, se, eventualemente, a decisão fosse tomada apenas por um elemento. O comité está apto, por exemplo, para minimizar a seleção infeliz de um mau produto, um funcionários não qualificado ou de uma tarefa genérica levada a cabo com insucesso.

Reconhecemos que, em situações do mundo real, todos os modelos têm limitações e haverá erros. Assim, o objetivo do *ensemble learning* é encontrar os pontos fortes e fracos dos modelos, levando à melhor tomada de decisão em geral. Vários resultados teóricos e empíricos mostram que a precisão de um conjunto pode exceder significativamente a de um único modelo. O princípio de combinar previsões tem sido de interesse para várias áreas de estudo ao longo de muitos anos.

Uma questão transversal nesta temática é a existência de um *dilema* entre o enviesamento e a variância dos dados dos conjuntos. Enquanto o enviesamento(*bias*) diz respeito à precisão, a variância diz-nos o quanto os dados são sensíveis a pequenas alterações. O problema surge porque, quando se tenta aumentar a variância, o *bias* diminui e vice-versa. Torna-se fundamental encontrar um equilíbrio entre estes dois componentes por forma a encontrarmos um modelo preditivo de qualidade e que se ajuste ao que pretendemos estimar. Iremos estabelecer comparações entre os métodos de redução de bias com os de redução de variância. Esse trade-off funciona como uma referência de quão robusto é o nosso modelo.

Em *ensemble learning*, o que nos interessa é a diversidade de previsões e não apenas a variação de um modelo diante de diferentes dados. É óbvio que os dois conceitos estão relacionados porque algoritmos instáveis geram modelos com alta variação e simultaneamente geram conjuntos com alta diversidade, porque essa variação existe.

Uma ideia importante aqui é que, embora reconheçamos a importância e as motivações por detrás do trade-off já mencionado, nos concentraremos em tentar avaliar diferentes técnicas, combinar algumas delas, em vários conjuntos de dados e analisar as suas características para contribuir para um bom *ensemble*, ou seja, tentar efetuar ainda melhores previsões com *ensemble learning*.

Categorias: *Metodologias Computacionais $\rightarrow$ Machine learning $\rightarrow$ Algoritmos Machine learning $\rightarrow$ Métodos de Ensemble*

Palavras chave: Ensemble learning, regressão, modelos

Acknowledgements

Firstly I want to thank the most important people in my life. To my mother for the initial support she gave me and the willpower she has always given me. To my two children for the understanding they have always shown. To my wife for the support she has secured in my family, patience, affection and the ever-transmitting force throughout this fabulous journey that would not have been possible without her support.

Thanks to my supervisor for the opportunity he gave me to do this dissertation, for the demonstrated professionalism and follow-up given during its realization. Thanks Professor João Moreira.

Special thanks to two officemates, Angela Cardoso and Artur Ferreira, who were with me on this adventure and became my friends. They turned out to be good fellows, helped me a lot along the way with their confidence, support and laughs - good fortune I had in meeting them.

To FEUP, for making me evolve and develop as a person, not only intellectually but also as a human being. Acknowledgment extending to the entire University of Porto.

To all my sincere thanks.

*“A man is like a fraction whose numerator is what he is
and whose denominator is what he thinks of himself.”*

Leo Tolstoy

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

AI	Artificial Intelligence
BG	Bagging
CV	Cross Validation
DT	Decision Tree
EL	Ensemble Learning
GB	Gradient Boosting
IT	Information Technology
IDE	Integrated Development Environment
NC	Negative Correlation
MAE	Mean Absolute Error
MB	Multi Boosting
ML	Machine Learning
PC	Personal Computer
RF	Random Forest
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
R	Root
RL	Reinforcement Learning
SML	Supervised Machine Learning
UML	Unsupervised Machine Learning
WG	Wagging
XGB	XGradient Boosting

Chapter 1

Introduction

In this first chapter of the dissertation, we introduce the frame behind the work and the motivation to pursue this essay. At the same time, we identify and define the problems that this dissertation addresses. We summarise the methodologies used in work and present a brief summary of each subsequent chapters in document structure section.

1.1 Context

During the last decades, ensemble learning has revealed to be a very interesting topic. Many and many papers were written by some scientists from a varying number of fields, like Statistics and Machine learning, for example.

In recent years, much attention has been given to Machine Learning related technologies. Machine Learning, is one of the impetus behind ongoing growth. Instead of having to program and adjust the code to deal with all scenarios, ML, could do some calculations and make some assumptions based on initial information in order to achieve new and adjustable results.

ML is increasingly used in products and services across companies. There are some noteworthy contemplations regarding the users that believe in the Web. Some issues could be identified when trying to use AI but this is out of the scope of this work.

Pierre Simon de Laplace, a recognised mathematician, illustrate, in 1818 [Bro09], that two probabilistic methods perform better than each method alone.

Since then, a large number of issues arisen and many research groups have started to work behind this idea.

Ensemble Learning is a predictive method that combines the predictions done by several base learners in order to obtain a better prediction. Some researchers had the idea of combine these learners in some manner and, in fact, there exist multiple ways of combine those predictions. We found that this kind of family methods is being very successful in many real world problems including predictive contests.

Ensemble methods help some knowledge areas to evolve and arises under the hood because those give an engaging answer to multiple machine learning algorithms.

1.2 Motivation and Objectives

A quotidian task we always need to do is decide about something. But to do that we feel the necessity of having more than one opinion when we want to decide about anything. A simple example will be making further ahead on this document, but for now we think if we can group different opinions in such way that they have different weights associated, because the knowledge about some subject we want to decide can be very distinct, we'll be able to make a more complex decision with much more information. So, the key point here is, grouping different individual opinions, will bring us more information and maybe a better decision.

In an attempt to get increasingly reliable predictive results, scientists have been doing more research in the area and it has undergone a lot of development.

The main idea behind forecasting is what we call accuracy. Instead of making just a simple prediction we need to make a good one, i.e., with a high probability of getting the correct value or category for some study. It is important to get a high value of accuracy and ensemble learning is proven to help on this as we saw, for instance, in one major competition that uses ensemble learning to solve the prediction problem like in Kaggle competition [Kag14].

It is known that ensembles are applied to a large number of fields and areas of interest. As a consequence, the impact of ensemble learning evolves will be substantial and will continue to be an object of research, probably, at least, in the next years. In the event that we can see decisively how, why, and when a specific group of techniques can be connected effectively, we will have faced toward an amazing new tool for Machine Learning because we'll have an impact on the quality and shortcomings of various learning frameworks.

A system that is able to teach itself to solve a given task is a system with auto learn capacity and able to solve every task that appears in. Building this kind of system is a mental idea that is turning very popular among the IT community and if done with accuracy will be even better.

The motivation to do this work are some: if for one hand we see a large utilisation of ensemble techniques in many problems which makes us willing to learn more and search for new materials, for the other hand most work focuses on classification problems and we'll focus on regression, where are a large area of investigation and possible experiments to do. One other thing we want highlight is the fact that some techniques are successful for classification are often not directly applicable for regression, there aren't even adapted and tested some algorithms in regression. So this was the original guidance to pursue this work and we defined three core objectives:

- Study and understand the bias/variance trade-off;
- Identification of which algorithms have a positive impact on the reduction of bias/variance;
- Try to obtain better predictive results working with some techniques, doing some experimentations.

This document describes and uses some techniques as Decision Tree, Bagging, Wagging, Random Forest, Gradient Boosting, XGBoost and three other appearing from the experiments which we name as MB1, MG2 and MBB. The effects of these *N* algorithms on the error, bias and variance of weak learners are explored in a number of experiments.

1.3 Document Structure

In addition to the introduction, this dissertation contains five more chapters.

In chapter 2, the state of the art is described and presented the work already done in the field.

Chapter 3 of this dissertation, we address the main topic, definitions of some concepts such as error model, what is ensemble learning and the pros/cons of ensemble models. We use 4 to explore the trade-off between bias and variance.

This chapter 5 outlines research strategies and techniques employed in the work and show the experiments been taken and the corresponding results.

Lastly, in 6 we show a summary of the work done and appreciated by the satisfaction objectives of the work, a list of contributions and directions for future work.

Chapter 2

State of the art

In this chapter we describe the state of the art and related work to this dissertation and show what exists in the same domain and which issues are open.

2.1 Introduction

We found very interesting the next 2.1 graphic that show us how this particular area of knowledge have been evolved in the last years through the increasing number of published related papers per year is getting, when we look together to the terms “ensemble” and “machine learning”, in a well-known database transversal to several disciplines - “Web of Science”. Looking for the 90’s and over we see a clear emerge around that decade and a continuously growth. We can look into the web of science database, described as the largest accessible citation database [Ana17].

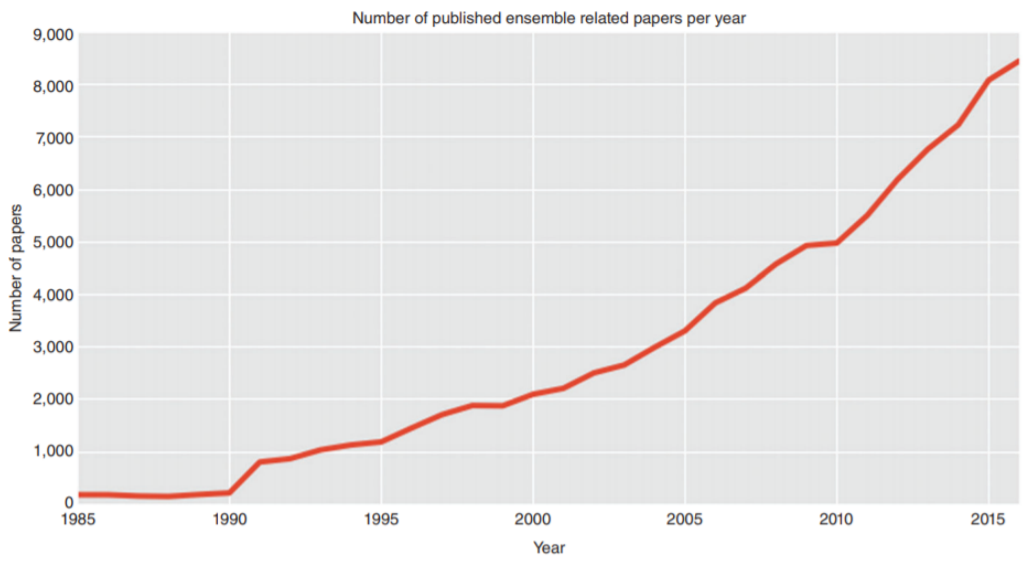
If we try to find papers by looking only for the word "ensemble" per se, we won’t find only relevant papers matching this topic. We need to search together with machine learning - maybe the most important field where ensemble learning is being applied.

2.2 Research already done

Historically, maybe one of the soonest appearances of the composition term ensemble learning was with Dasarathy and Sheela’s 1979 paper [DS79], which previously proposed utilizing ensemble with a divide and conquer style.

Eleven years after, on the beginning of 1990, with Hansen and Salamon [KS90], these two showed the importance, in first time ever, of using ensembles in neural networks. They came to a result that proves using ensembles could reduce the error of neural networks. At the same epoch, appears a work made by Schapire which led to the later appearance of AdaBoost - adaptive boosting algorithm. In his work he showed that is possible combine several weak learners to build one more powerful learner.

Figure 2.1: Number of published papers per year [SR18]



Since that time, more authors showed once and again that using ensemble models is possible to improve the predictive result of individual models in supervised machine learning.

In 1995, Krogh and Vedelsby [KV95] demonstrated that ambiguity and cross-validation gives a reliable estimate of the ensemble generalisation error. Eight years after, Brown and Wyatt [BW03] came to a result where negative correlation is an advancement of the previous work done by Krogh and Vedelsby and is defined as being a derived technique of the decomposition of those last authors.

In 1996, Ueda and Nakano went further and decomposed the generalisation error into three components, a calculation between bias, variance and covariance [UN96].

In 1998 Zheng and Webb appeared with what was at that time a new learning algorithm called MB, multiple boosting for classification [ZW98]. They combined bagging and boosting and demonstrated that, in average, performs better than bagging or boosting acting alone. Two years later, Webb [Web00], used with some success a new combined technique where instead of use bagging, he proposed wagging, where he bag with weighted averaging, following a Poisson distribution.

In 2009, at KDD cup, Xie, Rojkova, Pal and Coggeshall combined boosting and bagging but again, (all binary) classification problems. It was only in 2010, Pavlov, Gorodilov and Brunk, appeared with BagBoo, where bagged the boosting model, for a regression problem [PGB10].

The scientist Geoffrey Hinton (2006) rebranded neural net research as “deep learning” [Bea17] and was in that year the modern machine learning appeared. After that, many other works came up and presented more details about where ensemble learning can be applied, Oza and Tumer (2008) in remote sensing, medicine and recognition in general. In the past few years, other researches were performed. We know EL was successfully applied with great results in object detection [PSV], education [BSH$^+$17] and malware detection [IRC$^+$17].

Nowadays, ensembles are proved to be the state-of-the-art approach when trying to solve numerous machine learning challenges, as problems we can find in real life [FDCB+14]. Not all literature give us a perfect overview of what ensemble is, because in many cases we find ensemble covered in a very restrict way with a specific problem. But some works like (Polikar, 2006), provided a more generic view.

2.3 Other complement ideas

As we can see further in next chapters, will use a specific type of model validation, cross validation, in our experiments and we also will make hyperparameter tuning to our models.

Besides that we think it is good to highlight two other techniques, Negative Correlation Learning and Stacking as they demonstrated to be very effective predictive analysis.

2.3.1 Negative correlation learning

Negative Correlation learning, NC, endeavours to prepare singular networks to train individual systems in an ensemble, and group them in within the learning process. In NC, all the individual systems in the ensemble are trained at the same time and interactively through the punishment terms in their error functions [LY99]. It is flexible once it can works well either in classification or regression approaches. Differently from the *mse* error described in equation 3.1, this type of learning uses a penalty factor in its own formula which is related to the diversity of the ensemble.

2.3.2 Stacking

Stacking predominantly vary from bagging and boosting on two principles:

Table 2.1: Stacking Vs Bagging/Boosting

Stacking	**Bagging/Boosting**
Heterogeneous power learners	Homogeneous weak learners
knows how to aggregate base learners (with meta learner)	Follows deterministic calculations

The patent idea is train some base learners with other ensemble techniques and after, an aggregation algorithm, the *super learner*, is also trained to achieve a new prediction using the already known predictions of each base learners.

2.3.3 Cross validation

Cross validation is a measurable technique for assessing and looking at learning calculations by isolating information into two portions: one used to learn or prepare a model and the other used to approve the model. The main goal is that every datum point gets an opportunity of being approved against second portion. Different types of cross validation exists and we'll use a specific one.

2.3.4 Hyperparameter optimization

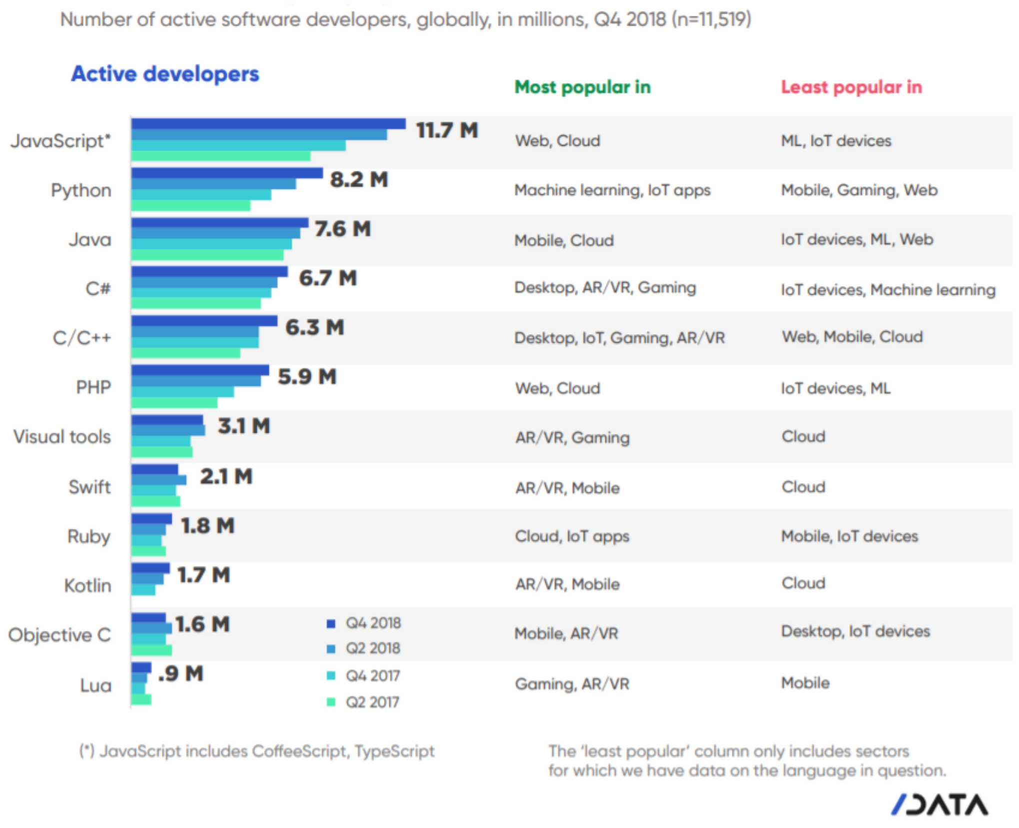
Recent methodologies are appearing in contrast with grid search using cross validation. This grid search has some disadvantages and other alternatives appeared and showed to be better for non-trivial search spaces, like sequential model-based parameter optimization (Hutter et al., 2011), random search (Bergstraand Bengio, 2012), and Bayesian optimization (Snoek etal., 2012), in addition to other researchers [LGS].

Hyperparameters optimization, or tuning as sometimes referred in the literature, are every input parameter we can set up before beginning the training process. Possible example is the number of estimators in Bagging Regressor.

As a comparison, the other parameters, the ones that aren't hyperparameters are used by the model to make predictions. We can have here as an example the weight coefficients in a linear regression model. So, for the sake of distinguish hyperparameters help within the learning process and parameters help us make predictions.

2.4 Technological Review

The main programming language, ordered by number of active developers and where each one is most popular or least popular can be shown in the next figure 2.2.



	Active developers	Most popular in	Least popular in
JavaScript*	11.7 M	Web, Cloud	ML, IoT devices
Python	8.2 M	Machine learning, IoT apps	Mobile, Gaming, Web
Java	7.6 M	Mobile, Cloud	IoT devices, ML, Web
C#	6.7 M	Desktop, AR/VR, Gaming	IoT devices, Machine learning
C/C++	6.3 M	Desktop, IoT, Gaming, AR/VR	Web, Mobile, Cloud
PHP	5.9 M	Web, Cloud	IoT devices, ML
Visual tools	3.1 M	AR/VR, Gaming	Cloud
Swift	2.1 M	AR/VR, Mobile	Cloud
Ruby	1.8 M	Cloud, IoT apps	Mobile, IoT devices
Kotlin	1.7 M	AR/VR, Mobile	Cloud
Objective C	1.6 M	Mobile, AR/VR	Desktop, IoT devices
Lua	.9 M	Gaming, AR/VR	Mobile

Figure 2.2: Number of active developers by programming language[1]

As we can see, the most popular programming language to deal with machine learning challenges was until the end of 2018, Python. It was the programming language elected to develop the practical part of this work. Python has some well known libraries widely used across the machine learning community, like the following ones:

- Numpy;
- Pandas;
- Matplotlib;
- SciKit-Learn;
- TensorFlow;
- Scipy;
- Seaborn

Other than that, it has a very large and active community, it is one of the programming languages growing up, has very nice support for machine learning, is very popular and easy to begin, with a relative small learning curve, if familiar with other programming languages, and use pip, a package installer, where we can find several important and useful libraries to install.

2.5 Conclusion

Besides the fact of ensemble learning is getting more and more recognition, several papers only use ensemble in a restricted manner and don't talk about all the transverse area where this kind of learning could be adopted and how could be done, making the most of the research already done.

For what we understood, EL is intrinsically related with the evolution of machine learning. So, in this case we find very productive use EL to help ML evolve and get predictive results with more accuracy.

There are open areas and lack of knowledge in EL where we can search and learn. One of them will be carried out along this dissertation and is related with multiboosting for regression to try to reduce at the same time, bias and variance, and more important than that, perceive what characteristics models need to have to generate better ensembles.

Let's move on to the next chapter to begin exploring some key concepts.

[1]Retrieved from https://www.developintelligence.com/full-developer-report/

Chapter 3

Ensemble Learning

In this chapter we present some of the fundamentals related with the topic of Ensemble Learning. This way we build a context and an origin where all the works started.

3.1 Machine Learning

The three main topics in machine learning are the supervised machine learning, unsupervised machine learning and reinforcement learning. Supervised Machine Learning, SML is a model building process where given a data set with a target variable and input variables, a modelling algorithm automatically generates a model (a formula or a rule set) which establishes a relationship between the target and some or all of the input variables. We can say SML works for two purposes:

1. Regression;

2. Classification.

The two type of issues have as objective the development of a concise model that can foresee the value of the dependent attribute from the attribute factors. The distinction between the two is, for the first one the dependent attribute needs to be a numerical value because we are trying to predict a value and for the second one a belonging category.

Unsupervised Machine Learning, UML, is related with a search for homogeneous subsets of the data that produce one or more possible segmentations.

Reinforcement learning, RL, is a portion of ML worried about how software agents should take activities in a situation so as to amplify reward.

Ensemble almost always are associated with SML. The reason is that with ensembles we are trying to increase the accuracy of our predictions.

Table 3.1: Ensemble Techniques

Approach	**Example**	**Inspiration**
Sequential	*AdaBoost*	Exploit reliance between learners
Parallel	*Bagging*	Exploit autonomy between learners

3.2 Ensemble

> [MMSMS12] "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."

Ensemble learning is the procedure by which various models, for example, classifiers or learners, are deliberately created and consolidated to take care of a specific computational insight issue. Ensemble learning is principally used to improve the (order, forecast, work estimation, and so on) execution of a model, or lessen the probability of an appalling choice of a poor one. Different uses of ensemble learning incorporate doling out a certainty to the choice made by the model, choosing ideal (or close ideal) features, information combination, incremental learning, non-stationary learning and mistake correcting [Pol09].

3.3 Types of Ensembles

Regarding the types of ensembles, we can categorise them into two categories. Heterogeneous type is related within the use of different kind of models. But for the other hand, we have another type, homogeneous type, and many famous algorithms works on top of this. We can find also different variants using the same algorithm.

We can highlight here the notion that most ensemble strategies utilise a unique type of base learner, creating a homogeneous ensemble.

3.4 Ensemble Methods

Some authors refers to ensemble methods as meta-algorithms who consolidate a few AI procedures into just one final model, the predictive one. It is known we could use bagging to diminish variance or use boosting to reduce bias. These methods can be separated into two ensemble techniques (table 3.1).

3.5 Decision Tree

Decision Trees, DT, are a non-parametric directed learning strategy utilised either for regression either for classification. The objective is to make a model that predicts the value of an objective variable after learning from a training data set.

3.5.1 Advantages and disadvantages of DT's

In the advantages we can talk about how DT's are easy to fit, use, understand and interpret; they also requires little data preparation in comparison with other techniques that may need data normalisation, dummy variables and remove blank values, for instance; but they also have some disadvantages like suffering from overfitting phenomenon and it is common observed if data changed a little it can affect the result itself and produce trees very different.

3.6 Model Evaluation Metrics

Well, focusing in regression will have in list 3.1 some metrics to evaluate the model. The main idea is comparing the actual values against predicted values to determine the accuracy of the regression model. This analysis provides an important evaluation of the model developed as it offers insights at the areas we need to tackle, once we want improvements. Error brings us one measure able to detect how far the data is from the regression value.

Examples are given below:

- *MAE* (Mean Absolute Error)
- *MSE* (Mean Square Error)
- *RMSE* (Root Mean Square Error)
- R^2 (R Squared)
- Adjusted R^2

List 3.1: Model Evaluation Metrics

Our data type and domain of knowledge will have impact on choosing the metric.

3.7 Model Error for Regression Problems

A model error will equip us with one important information and answer the question of why is this significant to us. It has particular interest to know what can be the source of errors, or simply, know why they appears. So, a question: what contributes to generate error and how can we manage that?

The blunder rising up out of any model can be separated into three parts numerically. Following are those parts. Each factor in the sum represents one component, like the following, as presented by Geman et al. [Stu92]:

$$MeanSquaredError = Bias^2 + Variance + IrreducibleError \tag{3.1}$$

Related with noise, we can define noise like the irreducible error and there isn't so much we can do to modify its value.

Thinking more in the previous equation we can work with that on a more mathematical way. Taking in account these three parameters, variance, bias and irreducible error we can bring here a simple equation that defines the value of model error for the regression problems.

Doing some kind of breakdown error we can split the total factor into a mathematically equation where each parcel of the sum is one component [Stu92]:

$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right) + E\left[\hat{f}(x) - E\left[\hat{f}(x)\right]\right]^2 + \sigma_e^2 \tag{3.2}$$

As usual, E means the expected value, i.e., the mean value, $\hat{f}(x)$ is the predicted value and $f(x)$ represents the real observed value.

Later, N. Ueda and R. Nakano., [UN96] presented the generalization error decomposition in three components: variance, covariance and bias, as showed in equation 3.3.

$$E\left[\left(\hat{f} - f\right)^2\right] = bias^2 + \frac{1}{K} var + \left(1 - \frac{1}{K}\right) covar \tag{3.3}$$

The goal is to minimize the value of equation 3.3. We can make some conclusions regarding the objective of the last equation, because it's trivial to see we should have the average of all the subcomponents with lowest possible value. For *bias* the learners should be as accurate as possible and for the *covar* the learners should present negative correlation.

In the equation 3.2, we've defined how we can obtain the generalization error, but we don't have yet a clear vision of how this is closed to an existent trade-off, explained and explored the section 4.2.

We need a simple and target approach to look at our worth estimation models so as to decide whether the outfit that we built speaks well with progress. We could use mean squared error, MSE, for example. More metrics can be found at section 3.6.

After knowing some of the metrics 3.1, we are able to get the ensemble error within a function that works with the errors of learners.

3.8 Advantages of Ensemble Models

There are two noteworthy advantages of ensemble models. They are:

- The result, normally, produces a better forecast;
- Helps on increase a steady model.

The total expression of a different set of models is less loud than just one type of model. In account, we can call this diversity[PMSR14], a blended arrangement of numerous stocks will be substantially less than only one of the stocks alone. And this is additionally the reason why our models will have better performance with ensemble instead of using just one model.

The key point here to make better ensembles is what we call above - model diversity. It could work like a car purchase. Before we buy a car, we may want listen to some bits of advice, as the dealer, family, colleagues, see some reviews in magazines or websites and after all, take a more informed decision. This metaphor works like an ensemble because if we have many different opinions by different peoples, specialized or not, we will, maybe, do an effective final decision, like an ensemble.

3.9 Disadvantages of ensemble methods

Not all are flowers and we want to show that ensemble methods bring some disadvantages. They are:

- Some decrease in interpretability because when we add some complexity it's possible to turn it hard to draw back any essential business bits of knowledge;
- Calculation time is high;
- Models select is a craftsmanship which is extremely difficult to ace.

3.10 Phases of Ensemble Learning

The techniques for ensemble learning have, normally, three stages: generation, pruning and integration. The first stage expects to generate an ensemble of models. The pruning phase, not always applied, expects to select a model from a set of other models, in the way of downward computational complexity, and if conceivable, improve accuracy. The third phase, ensemble integration utilize the forecasts made by the models in the ensemble to acquire the last prediction [MMSMS12].

3.10.1 Ensemble Generation

In this first scenario, we can tackle why this, among others, is an important initial phase. It would be great if we can understand what are the conditions that affects the performance of models. This has a trivial implication because if we will be able to identify the most suitable model for a given problem, we will be ready to recognize which characteristics of each method should be improved [MMSMS12].

Besides the fact of many authors dedicate themselves on one specific theme or idea to generate the ensemble, we have many techniques in doing so. One methodology, called meta learning, allows systems to gain effectiveness through experience (Brazdil et al, 2009). Many other techniques exist but we will not focus on that phase.

3.11 Focus on Regression

Regression is a type of ensemble problems where the goal is find out a model to accurately predict an unknown quantitative value. We need to perform regression evaluation after building the model. Further, we discuss two approaches that can be used to achieve this goal. They are:

- Train and test on the same dataset;
- Train/Test split.

One of the solutions, first in the previous list, is to choose a bit of our dataset for testing. In this case, we utilize the whole dataset for training and we construct a model utilizing this training set. Then, we select a little segment of the dataset, known as a test set. Finally, we analyze the predicted values by our model with the values in the test set.

Training the model on whole dataset, at that point we test it utilizing a segment of the same dataset. In a general sense, when we test with a dataset in which we already know the target value for every data point, we are ready to acquire a level of precise expectations for the model. This kind of approach would lead us to a high training precision and low out-of-sample precision since the model knows the majority of the testing data from the training stage.

A key point here is understand now what is, in fact, training accuracy and out-of-sample precision. Training accuracy is the level of right forecasts that the model makes when utilizing the test dataset. In any case, a high training precision isn't really something to be thankful for. Imagining if we are having a high training accuracy it could take us to an over-fit situation. This implies the model is excessively prepared to the dataset, which may catch noise and produce a non generalized model. By other side, out-of-sample accuracy is the level of right expectations that the model makes on data for which has not been prepared on. Completing a train and test on the same dataset will low out-of-sample precision because of the likelihood of being over-fit. It's very important that our models have high out-of-sample precision in light of the fact that the motivation behind our model is, obviously, to make right forecasts on unknown data.

The second approach is called train/test split. Like the term says, this splits the dataset into training and testing sets individually, which are fundamentally unrelated. After which, we use training set to train and testing set to test. This will give a progressively precise assessment on out-of-sample exactness in light of the fact that the testing dataset isn't a piece of the dataset that has been utilized to train the data, as the first example. Since this information has not been utilized to train the model, the model has no knowledge of the result. In this way, basically, it's genuinely out-of-sample testing. The issue with train/test split is that it's very reliant on the datasets on which the information was trained and tested. Train/test split normally have a superior out-of-sample prediction than training and testing on the same dataset, but it has some problems related with this dependency [Gup17].

In short, the first uses all the data to train while the second have a clearly defined set to that purpose. Even more, the first use a test set that was used for training and the train/test split used one set test not used in its training process. Is in the scope of this two approaches and their issues we want to refer another evaluation model called cross-validation.

3.11.1 Cross Validation

In this area of model validation we can find several works done out there but one has particular interest, the one that analysed variance of cross validation estimators of the generalization error[Koh95]. We can go with two of the most known techniques:

3.11.1.1 KFold

Cross-validation is a resampling strategy used to assess evaluate models in machine learning [Pap20]. This strategy only uses one parameter, k, that alludes to the quantity of gatherings that a given information test is to be part into. Each set of data in k is called fold. Imagine we have $k = 5$, in this case we say we are using $5 - fold$ cross validation.

This strategy was already used in statistics and was applied to machine learning. It is famous because of its simplicity and generally gets models with less bias in comparison with simple technique of splitting the data into train and test. As we can see in the image 3.1, we use $k - 1$ parcels to train and just one parcel to test, in the figure we use $k = 10$.

Simple methodology of the process of running $k - fold$ cross validation can be resumed in the next steps:

- Mix the elements of dataset haphazardly.
- Split the dataset into k equal sized folds and for every fold:
    - Select $k - 1$ folds for training set and other for test set
    - Fit a model on the training set and evaluate it on the test set
    - Store the score for each iteration k and dispose the model
- Condense the information of the model using the sample of model evaluation scores

The main idea is that at each iteration we never repeat the test fold across all the iterations, so each sample has the opportunity to be in the test set one time and, of course, used in training set $k - 1$ times.

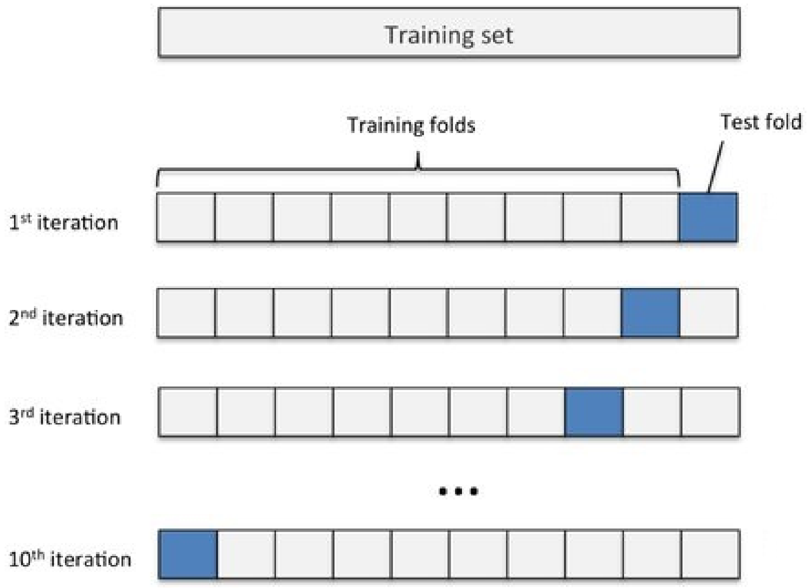


Figure 3.1: Diagram[1] $k - fold$ cross validation with $k = 10$

3.11.1.2 Leave One Out Cross Validation

This type of cross validation is a special case of $k - fold$ cross validation previously presented. Now instead of using a specific value for parameter k, in this case k has the same value as the dataset length. A very representative image can be seen in figure 3.2.

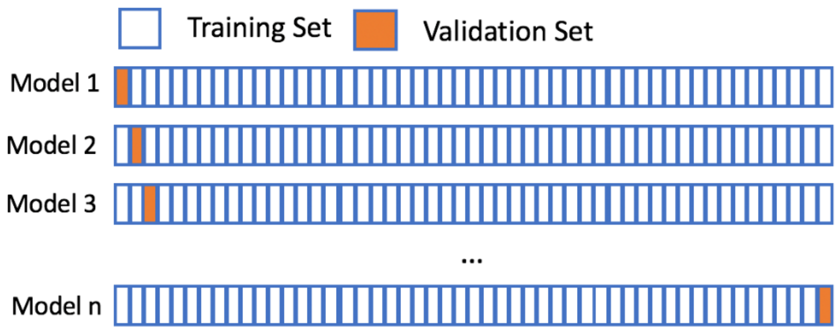


Figure 3.2: Diagram[2] leave one out cross validation with $k = n$

3.12 Conclusion

We present in the chapter 3 some important definitions and concepts, some general important concepts around regression, the dilemma between bias and variance and the advantages/disadvantages of using ensemble models.

[1] Adapted from Karl Rosaen Log http://karlrosaen.com/ml/learning-log/2016-06-20/
[2] Retrieved from https://campus.datacamp.com/courses/model-validation-in-python/cross-validation?ex=10

Our focus will be in evaluating the reduction of a given technique around bias or variance, try to understand what characteristics the models need to have great accuracy and try to find a better predictive result when mixing some techniques.

In the next chapter we will deal with some starting aspects of the implementation and how can we lead some experiments.

Chapter 4

Bias Variance Trade-off

In this chapter we intend to clarify the most recognizable trade-off between two core concepts - bias and variance.

4.1 Fragment Phenomenon

Suppose we have built three models, a linear regression model, a neural network and a regression tree for a specific problem, the order is not important.

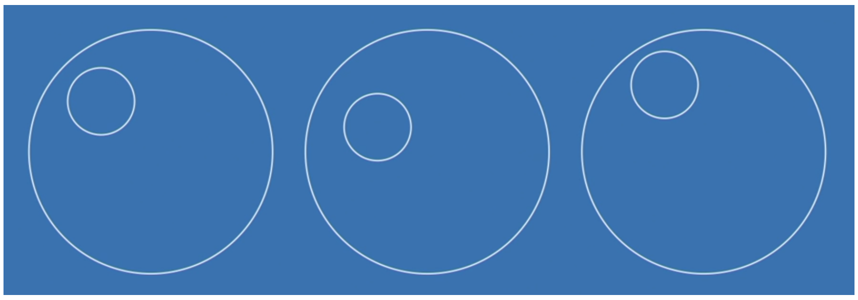


Figure 4.1: Three models

The inner-circle in each of the models refers to the error we have. If we put all the three models one on top of the others, we will get what is in the image 4.2.

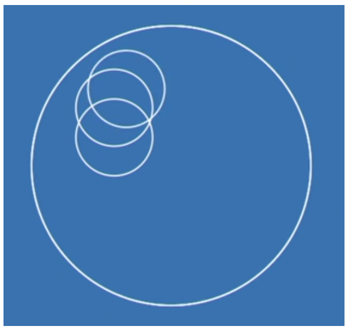


Figure 4.2: Ensemble with three models together

So, it's obvious that we have a non-empty intersection of the inner circles. This takes us to the fragment phenomenon. This situation is ineffective because we find the same errors in different models. One goal here is trying to find models that find different errors, i.e., with no overlapping errors when making the ensemble to have a more effective result. That is an important characteristic of the models to work on. Some topics related with which characteristics should the models have to be considered a good model will be explored with more detail later.

4.2 The bias-variance trade-off explained

One potential issue that arises is related with a dilemma between bias and variance, better known as trade-off. In the previous section, we take an overview of a mathematical equation, 3.2, but now, we define better the two main plots on that and how they are related.

Variance

"... refers to the amount by which (our model) would change if we estimated it using a different training data set." [GDT+13]

Bias

"... refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model." [GDT+13]

4.2.1 The trade-off

The bias could assume, generally, a low value if the methods are more flexible. This led us to a trade-off because it is described that low bias, normally implies high variance and vice versa. So, it is a core issue think how could we have a balance either with bias or with variance.

It is presented in figure 4.3 of page 23 an illustrative image of how the bias and variance are related. Assuming that red spot at the circle centre is the real value and blue dots are predictions.

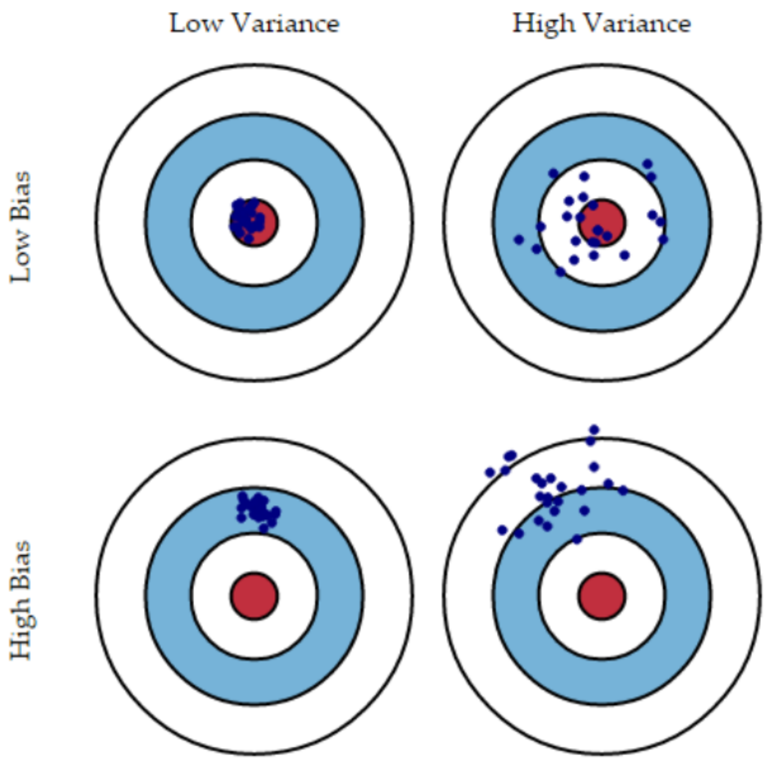


Figure 4.3: Bias/Variance [GDT^{+}13]

Typically, as we increment the unpredictability of our model, we will see a decrease in error because model has lower bias. In any case, this just occurs until a specific point. As we keep on making our model increasing complexity, we end up over-fitting, i.e., the predictor is too complex and flexible making fewer mistakes on unseen future data, and consequently, our model will begin experiencing high variance as we can see in figure 4.4, often a consequence of a model that is very simple. But the opposite can also happen, under-fitting, if the predictor is too simplistic and rigid which led to a model that does not fit well the test data.

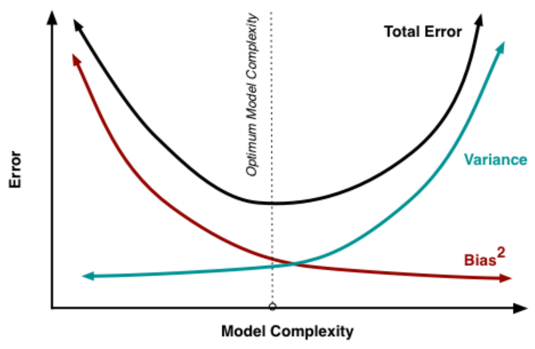


Figure 4.4: Model Complexity (Fortmann-Roe, 2012)

The previous figure present us a graphical view of the trade-off. We could say that ensemble learning mune us with a powerful tool to examine this. A winning model ought to keep up a harmony between these two sorts of components, trying to balance the final value of total error.

Next section will guide us through all the implementation plan, experiments carried out and the achieved results.

Chapter 5

Implementation

In this chapter, we start our plan of execution and will start doing some experiments. We will start by presenting some of ensemble learning methods, models and one experiment done in Python language which allowed to have the first approach to this programming language.

5.1 Data sets

Many different data sets can be used and acquired to test our experiments. Some of them can be found on Kaggle [Kag14] or UCI Repository, for example. Either one or the other are originated from true issues and were respected useful by past researchers.

Three datasets were selected:

1. Boston house prices
2. Real estate valuation
3. California housing

5.1.1 Boston house prices dataset

This data refers to the epoch of 1978 to some Boston housing. This set contains 506 entries and represents some characteristics of homes from different rural areas in Boston city, Massachusetts state. Doesn't exist missing values and the last attribute is normally the target where others are predictive features.

Next list resume all the attributes information:

- CRIM per capita crime rate by town
- ZN proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS proportion of non-retail business acres per town

- CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- NOX nitric oxides concentration (parts per 10 million)
- RM average number of rooms per dwelling
- AGE proportion of owner-occupied units built prior to 1940
- DIS weighted distances to five Boston employment centres
- RAD index of accessibility to radial highways
- TAX full-value property-tax rate per $\$10,000$
- PTRATIO pupil-teacher ratio by town
- B $1000(Bk-0.63)^2$ where Bk is the proportion of blacks by town
- LSTAT % lower status of the population
- MEDV Median value of owner-occupied homes in $1000's

retrieved from https://www.cs.toronto.edu/ delve/data/boston/bostonDetail.html

This dataset[1]was taken from the StatLib library which is maintained at Carnegie Mellon University. Looking for some regression problems we found this dataset has been used in many machine learning papers.

5.1.2 Real estate valuation dataset

Some basic characteristics of this dataset were raised: number of instances is 414, 6 features listed below, one target, the price of the house per unit area. Doesn't exist missing values.

This data was collected from Sindian Dist., New Taipei City, Taiwan and dollar as currency and ping as metric, where ping is approximately $3.3m^2$.

Next list resume all the features information:

- Transaction date - example, 2013.250 = 2013 March
- House age - in years
- Distance to the nearest MRT station - in meters
- Number of convenience stores in the living circle on foot - integer value
- Latitude - in degrees
- Longitude - in degrees

[1]Harrison, D. & Rubinfeld, D.L. (1978) Hedonic prices and the demand for clean air. J. Environ. Economics and Management 5, p.81–102

Adaptado de https://www.cs.toronto.edu/ delve/data/boston/bostonDetail.html Source `https://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set`[2]

5.1.3 California Housing dataset

This dataset represents a sample of data from the houses in California districts. Some basic characteristics of this dataset were raised: number of instances is 20640, of predictive features is 8 and does not exist missing values. Again, the target variable is the median house value.

Next list resume all the features information:

- MedInc median income in block
- HouseAge median house age in block
- AveRooms average number of rooms
- AveBedrms average number of bedrooms
- Population block population
- AveOccup average house occupancy
- Latitude house block latitude
- Longitude house block longitude

This information was retrieved from the well known StatLib repository `http://lib.stat.cmu.edu/datasets/` and derived from the 1990 U.S. census[3].

5.1.4 Datasets Aggregated Information

In the table 5.1, we summarize the information about those three datasets.

Table 5.1: Datasets Comparison

	Instances	**Features**	**Features data type**	**Target data type**
Boston	506	13	Real positive	Real, $[5,50]$
Real estate	414	6	Integer, real	Real
California	20640	8	Real	Real, $[0.15,5]$

As we can see we have two sets with approximate number of instances but with approximately with half number of features and one set distinguishes from the others due to the large number of instances. Also, the features data type are all real numbers with slightly different signs and the target is always a real value.

[2]Yeh, I. C., & Hsu, T. K. (2018). Building real estate valuation models with comparative approach through case-based reasoning. Applied Soft Computing, 65, p. 260-271

[3]Reference: Pace, R. Kelley, and Ronald Barry, "Sparse Spatial Autoregressions," Statistics and Probability Letters, Volume 33, Number 3, May 5 1997, p. 291-297.

5.2 Deep on Ensemble Learning Methods

The two most prevalent ensemble methods are bagging and boosting. The first one, bagging, trains several single models parallelly. Arbitrary, each model is created as a subset. The other method referred previously, boosting, has its main difference compared with bagging in the way that it works in a consecutive manner. This allows for one important characteristic: every single model learn from mistakes made by the precedent model [Roc19].

Another method, stacking, that regularly thinks about heterogeneous weak learners, work with them in parallel and consolidates them via training a meta-model to yield a forecast based on several weak models' predictions [Wol92]

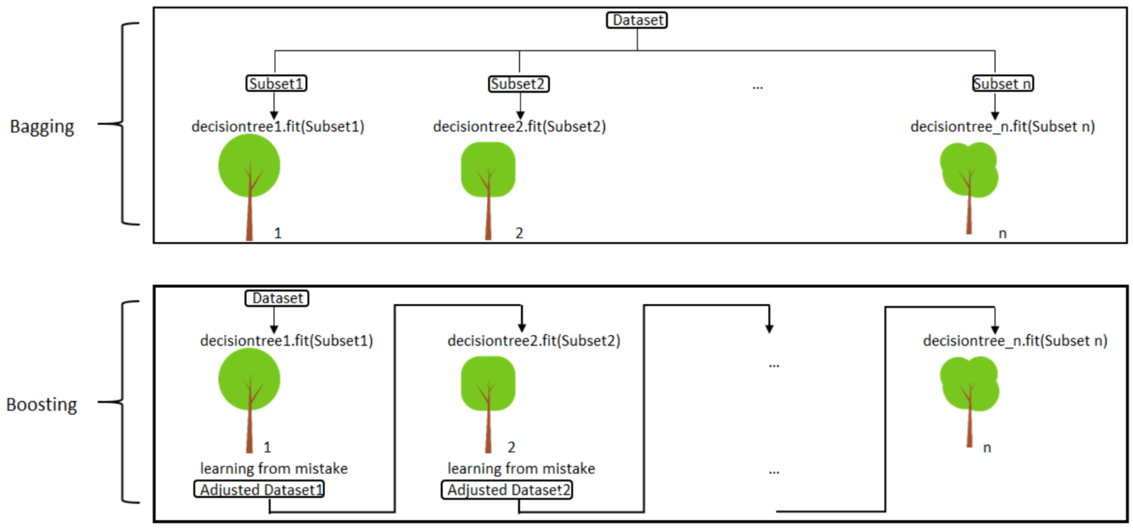

Bagging
Dataset
Subset1
Subset2
...
Subset n
decisiontree1.fit(Subset1)
decisiontree2.fit(Subset2)
decisiontree_n.fit(Subset n)
1
2
n
Boosting
Dataset
decisiontree1.fit(Subset1)
decisiontree2.fit(Subset2)
...
decisiontree_n.fit(Subset n)
1
2
n
learning from mistake
Adjusted Dataset1
learning from mistake
Adjusted Dataset2
...

Figure 5.1: Bagging and Boosting methods

5.2.1 Some algorithms based on bagging and boosting

We can enumerate a very extensive list with many and many algorithms base on those two techniques. But, for the sake of information we'll refer just three of them, judging us to be of the most used:

- Random Forest is a very well known algorithm that uses bagging as strategy and decision tree DT as the individual model;
- AdaBoost, Adaptive Boosting, like the name evince, is a boosting ensemble technique and works particularly well with DT. Boosting models are gaining from the knowledge of the previous slip-ups and AdaBoost gains from the errors by expanding the heaviness of misclassified information;
- Gradient boosting, GB, is another boosting algorithm that uses boosting technique. This technique gains from the previous errors, residual error directly, instead of updating the weights of the data.

5.3 Set up

For the overall experiments we used some concepts already defined in 3. We decided to use cross validation with k-fold as mentioned in 3.11.1.1 and the model error equation in 3.1. Besides that all the code was written[4] in Python language, version 3.7, within some well known libraries as mentioned in 2.4, namely *scikit-learn*, running a Windows 10 Pro version software in a machine with the following characteristics:

- A Intel(R) CORE(TM) i7-4710HQ CPU @2.50GHz up to 3.50GHz processor
- 16.0*GB* installed RAM
- Four is the number of cores

All the code was developed recurring to the Visual Studio Code IDE. The possibility to add a seed value was great because we can guarantee that at each iteration of the script we got always the exact numbers generated. In that way we can dismiss the uncertainty due to different partitions of the data, have always the same splits to work on and compare the results with logic.

We also used a seed value in order to have guaranteed that a similar arrangement of arbitrary numbers are created. Doing that allowed us to have, for example, the same partitions of the train and test data across all the ensembles and this way we can compare and evaluate the models execution properly.

5.4 Hyperparameter tuning for machine learning models

We think in use one tuning technique to help us decide, before we run each technique, what are the best set choice of parameters to start.

Comparing GridSearchCV and RandomizedSearchCV, both from sklearn.model_selection library, we use RandomizedSearchCV in 3 different techniques: Bagging, Random Forest and Gradient Boosting for each dataset. We just select one fixed parameter, the number of estimators, to use across all. The range was $range(100, 5001, 150)$, i.e., starting at 100 until 5000 with steps of 150.

The choose for RandomSearch is related for the velocity we have in running this against GridSearch, the first is much faster then the latter.

The configuration passed to each instance of search was:

- $n\_iter = 60$
- $scoring = r2$
- $cv = 5$
- $n\_jobs = 8$

[4]All the code developed in the scope of this work can be found in `https://github.com/nmvalente/diss`

The following tables, 5.2, 5.3 and 5.4 resume all the results obtained in the search, for each dataset.

Table 5.2: Boston House Prices Results for Randomized Search

	Bagging	Random Forest	Gradient Boosting
Best parameters	$n\_estimators = 550$	$n\_estimators = 2350$	$n\_estimators = 250$
Best CV Score	0.84063	0.84061	0.85724
R2 Score on test data	0.86216	0.86695	0.88281
Prediction	22.61007	22.59685	22.59962
Process time	$4.8min$	$5.3min$	$46.5s$

Table 5.3: Real Estate Results for Randomized Search

	Bagging	Random Forest	Gradient Boosting
Best parameters	$n\_estimators = 700$	$n\_estimators = 1150$	$n\_estimators = 100$
Best CV Score	0.704869721388346	0.70524	0.70040
R2 Score on test data	0.57842	0.57425	0.53569
Prediction	36.78823	36.85550	37.46666
Process time	$3.7min$	$3.7min$	$41s$

Table 5.4: California Price Results for Randomized Search

	Bagging	Random Forest	Gradient Boosting
Best parameters	$n\_estimators = 550$	$n\_estimators = 2350$	$n\_estimators = 250$
Best CV Score	0.84063	0.84061	0.85724
R2 Score on test data	0.86216	0.86695	0.88281
Prediction	22.61007	22.59685	22.59962
Process time	$3.7s$	$3.7s$	$3.7s$

5.5 Experiments

To run all the subsequent experiments, the script developed in python just need to indicate the filename of the dataset to run each one, which need to be by default in the *datasets/* directory accordingly to the repository structure. To run the script we only need to run it in the root folder of the project.

Next we can see some of the possible calls to the script:

- python .\script.py –file boston_house_prices.csv

- python .\script.py –file real_estate_valuation.csv
- python .\script.py –file california.csv
- python .\script.py –help and get information about how we must run it:
    1. usage -> python shared\util.py –file (or -f) <filename>
    2. usage -> python shared\util.py –path (or -p) <path>

Regarding the techniques we have been working on, we pointed out in the list 5.1 all of them.

∗ DT - Decision Tree

∗ BG - Bagging

∗ WG - Wagging

∗ RF - Random Forest

∗ GB - Gradient Boosting

∗ XGB - XGradient Boosting

∗ MB1 - Multi boosting with Wagging and GB

∗ MG2 - Multi boosting with Wagging and XGB

∗ MBB - Multi boosting with weighted Wagging and XGB

List 5.1: Resume of all techniques used

The technique Decision Tree was putted here just to compare with the others, working as a starting point. BG, RF, GB, XGB are all techniques already known and used with the values found after the Randomized Search done in tables 5.2, 5.3 and 5.4, with the respective *n_estimators* parameter value. Wagging is a technique that appears with Webb as mentioned in chaper 2.2. The idea is similar to Bagging but instead of using equally weighted samples, uses weighted averaging, following a Poisson distribution as we can see in equation 5.1 [Web00].

$$Poisson() = -log\left(\frac{Random(1...999)}{1000}\right) \tag{5.1}$$

After obtaining a random value all the values are standardized. This technique proved to have interesting results and we should consider it.

The last three techniques, MB1, MG2 and MBB were performed and new trials have been done to achieve better results. MB1 and MG2 are similar. Both use Wagging but instead of have a Decision Tree as base learner, we have a Gradient Boosting regressor for the MB1 and a XGradient Boosting regressor for the MG2. So, for both we use equal values for $n\_estimators_{WG}$ of Wagging and for GB and XGB the $n\_estimators_{GB}$ already found. The idea was to mixture techniques that reduce bias and variance in order to reduce the generalization error. With that in mind and after observing the predictive analysis of XGB in the three different datasets, we added this last one last technique. That one, MBB, uses XGB and Wagging. The idea was to separate the data, one parcel to WG and other to XGB, by a p percent value. So, for each $k-fold$ instance, we train p for WG and $(1-p)$ for XGB. We trained and predicted the values independently, taking in account the weighted.

In all subsequent experiments, with experiment order number greater than zero, we resume all the information in tables. We can see in MB1 and MG2 we did not put any percentage because we use one and other sequentially and for the MBB we always refer the percentage of each technique used.

- How we obtain those values for p?

- We ran multiple times and we used increments of 0.01 from 0.01 to 0.99 in order to find the best p value for each race.

But let us just begin with the warm up experiment realised right at the beginning of the problem.

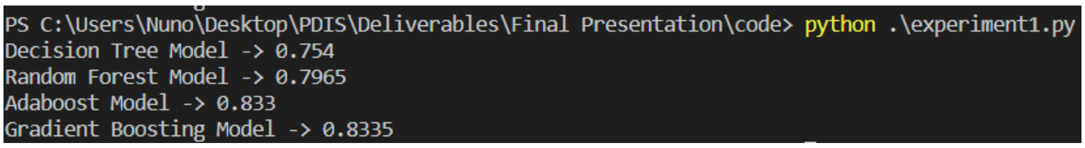
5.5.1 Experiment Zero

In the next few lines of code, listing 5.1, we try to run a first mini program in order to determine the accuracy score of each one of the algorithms referred above, in the section 5.2.1.

```
# Load Libraries
from sklearn.datasets import make_moons
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,
    GradientBoostingClassifier

# Create initial data set
X, y = make_moons(n_samples=10000, noise=.5, random_state=0)

# Splitting training test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Decision Tree model
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Decision Tree Model ->", accuracy_score(y_test, y_pred))

```

```python
# Random Forest model
clf = RandomForestClassifier(n_estimators=100, max_features="auto",random_state=0)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Random Forest Model ->", accuracy_score(y_test, y_pred))

# AdaBoost model
clf = AdaBoostClassifier(n_estimators=100)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Adaboost Model ->", accuracy_score(y_test, y_pred))

# Gradient Boosting model
clf = GradientBoostingClassifier(n_estimators=100)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print("Gradient Boosting Model ->", accuracy_score(y_test, y_pred))
```

Listing 5.1: Compare the values obtained with Random Forest, Adaboost and Gradient Boosting within a Decision Tree



```
PS C:\Users\Nuno\Desktop\PDIS\Deliverables\Final Presentation\code> python .\experiment1.py
Decision Tree Model -> 0.754
Random Forest Model -> 0.7965
Adaboost Model -> 0.833
Gradient Boosting Model -> 0.8335
```

Figure 5.2: Score output for each model

We can observe in figure 5.2, that the two last models for this experiment zero, the boosting related, are the best. They can obtain an approximately 10% of improvement related to the Decision Tree Model. That was our first approach to python language and we used classification instead of regression. After that first approach the path we defined was to focus only in regression as said in 3.11.

5.5.2 Experiment 1

In the two next tables, 5.5 and 5.6, we resume all the information regarding the decomposition of bias and variance, overall metric error evaluation - *mse*, the weight of bias and variance in *mse*, the predictive value of the ensemble, coefficient of determination R^2 score and the time taken to run each technique.

Table 5.5: Boston House Techniques Part1

	DT	**BG**	**WG**	**RF**
base learner	DT	DT	DT	DT
n_instances	1	550	550	2350
$bias^2$	0.289	0.197	0.162	0.197
variance	20.785	11.864	11.588	11.710
MSE(error)	21.074	12.061	11.750	11.906
bias weight	1.373	1.637	1.379	0.197
var weight	98.627	98.363	98.621	98.346
prediction	22.853	22.542	22.514	22.549
R2 score	0.749	0.853	0.855	0.855
process time(s)	0.024	33.847	35.520	18.123

Table 5.6: Boston House Techniques Part2

	GB	**XGB**	**MB1**	**MG2**	**MBB**
base learner	DT	DT	DT - WG/GB	DT - WG/XGB	DT - WG/XGB (3%, 97%)
n_instances	250	250	100/250	100/250	100/250
$bias^2$	0.248	0.261	0.185	0.207	0.809
variance	12.880	11.033	12.604	11.032	9.794
MSE(error)	13.128	11.294	12.789	11.239	10.603
bias weight	1.886	2.309	1.447	1.839	7.632
var weight	98.114	97.691	98.553	98.161	92.368
prediction	22.598	22.508	22.517	22.420	22.415
R2 score	0.842	0.862	0.842	0.862	0.698
process time(s)	0.733	0.349	172.022	149.164	3.396

The image 5.3 show the relationship between true values and the predicted values. This turns out to have a vision of how they relate to each other.

The best performer technique looking into *mse* error was the *MBB* technique. If we look individually to the *$bias^2$* and *variance* components, we can say WG was the best and *MBB* is the best, respectively. Of course, if we compare WG and *MBB*, the R^2 score is much different, where *MBB* loses. Notice well that *MBB* has the worse R^2 score among all the others, event DT. So, looking into all the factors we can chose XGB if time is important or $MG2$ if time is not a problem. If we are only searching by *mse* error we elected the *MBB*.

5.5.3 Experiment 2

In the two next tables, 5.7 and 5.8, we resume all the information regarding the decomposition of bias and variance, overall metric error evaluation - *mse*, the weight of bias and variance in *mse*, the predictive value of the ensemble, coefficient of determination R^2 score and the time taken to run each technique.

Table 5.7: Real Estate Techniques Part1

	DT	**BG**	**WG**	**RF**
base learner	DT	DT	DT	DT
n_instances	1	700	700	1150
bias^2	1.905	1.694	1.681	1.667
variance	88.639	57.523	56.129	57.573
MSE(error)	90.545	59.218	57.810	59.239
bias weight	2.104	2.861	2.908	2.813
var weight	97.896	97.139	97.092	97.187
prediction	38.279	38.289	38.324	38.282
R2 score	0.521	0.684	0.693	0.684
process time(s)	0.010	39.414	45.014	10.709

Table 5.8: Real Estate Techniques Part2

	GB	**XGB**	**MB1**	**MG2**	**MBB**
base learner	DT	DT	DT - WG/GB	DT - WG/XGB	DT - WG/XGB (13%, 87%)
n_instances	100	100	100/700	100/700	100/700
bias^2	1.021	1.228	0.662	1.498	1.502
variance	80.041	58.044	74.759	55.699	55.568
MSE(error)	81.062	59.272	75.421	57.197	57.071
bias weight	1.260	2.072	0.877	2.619	2.933
var weight	98.740	97.928	99.123	97.381	97.067
prediction	37.944	38.072	38.109	38.216	38.430
R2 score	0.569	0.683	0.596	0.694	0.693
process time(s)	0.2050	0.168	95.128	83.766	3.896

The image 5.4 show the relationship between true values and the predicted values. This turns out to have a vision of how they relate to each other.

The best performer technique looking into *mse* error was the *MBB* technique again. If we look individually to the *bias*^{2} and *variance* components, we can say *MB*1 was the best and *MBB* is also the best here, respectively. Comparing the values for R^2 score we can say $MG2$ and MBB are the best but the difference is so minimal, where the process time makes the decision, because $MG2$ is approximately 21 times slower. So the winner ensemble in this experiment was the MBB.

5.5.4 Experiment 3

In the two next tables, 5.9 and 5.10, we resume all the information regarding the decomposition of bias and variance, overall metric error evaluation - *mse*, the weight of bias and variance in *mse*, the predictive value of the ensemble, coefficient of determination R^2 score and the time taken to run each technique.

Table 5.9: California Housing Techniques Part1

	DT	**BG**	**WG**	**RF**
base learner	DT	DT	DT	DT
n_instances	1	550	550	2350
bias^2	0.000	0.000	0.000	0.000
variance	1.278	0.748	0.743	0.747
MSE(error)	1.278	0.748	0.743	0.747
bias weight	0.0030	0.046	0.049	0.049
var weight	99.970	99.954	99.951	99.951
prediction	1.574	1.584	1.585	1.584
R2 score	0.200	0.532	0.535	0.533
process time(s)	0.258	72.474	77.930	118.520

Table 5.10: California Housing Techniques Part2

	GB	**XGB**	**MB1**	**MG2**	**MBB**
base learner	DT	DT	DT - WG/GB	DT - WG/XGB	DT - WG/XGB (40%, 60%)
n_instances	250	250	100/250	100/250	100/250
bias^2	0.000	0.000	0.000	0.000	0.000
variance	0.818	0.686	0.816	0.689	0.629
MSE(error)	0.818	0.686	0.816	0.689	0.629
bias weight	0.047	2.309	0.033	0.064	0.067
var weight	99.953	99.942	99.967	99.936	99.933
prediction	1.572	1.572	1.572	1.576	1.570
R2 score	0.488	0.571	0.490	0.569	0.604
process time(s)	13.368	5.046	1599.769	1849.570	44.618

The image 5.5 show the relationship between true values and the predicted values. This turns out to have a vision of how they relate to each other.

In this particular dataset we didn't find any bias, only variance error. So, the *mse* is given by the value of the variance. Again, the best *mse* error appeared with *MBB*. This dataset has a number of instances very superior in comparison to the previous, so that can justify the value obtained for R^2 score. And here, *MBB* is also the best. So if time is a problem, maybe *XGB* is the right technique, otherwise *MBB*.

5.6 Conclusion

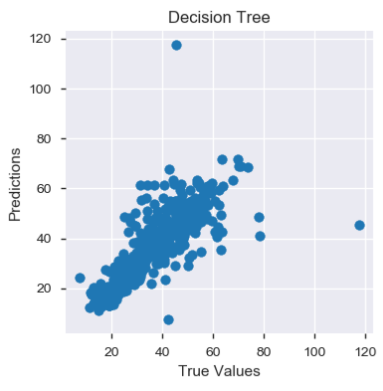
All experiments were summarized in three main packages. Of course the number of experiments here seems a little reductive because we try so many different configurations, made so many trials regarding the Randomized Search or searching for the best p percent value to use in MBB, for example.

We resumed all the information in three sub-chapters to have all relevant data condensed. That way we can compare and analyse better the techniques used and what values were obtained.

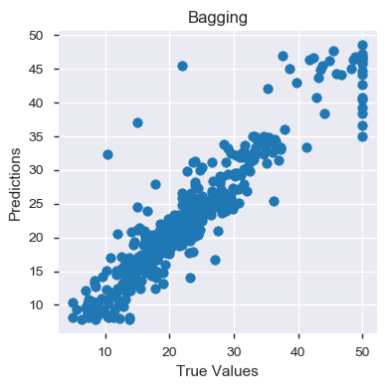
In the table 5.11, excluding the DT because of the process time(simpler), we can observe the *MBB* always have the better generalization error and it also appears as the best with R^2 score in two of three experiments. Consistency also exists when looking to the process time of each technique where, with no doubts, *XGB* wins across all the experiments. So, we can conclude *MBB* is the best for the situations we study but *XGB* is a very good option, not getting far behind but winning at race time.

Experiment #	MSE	R^2	$Time(s)$
EXP 1	MBB	XGB and $MG2$	XGB
EXP 2	MBB	WG, $MG2$ and MBB	XGB
EXP 3	MBB	MBB	XGB

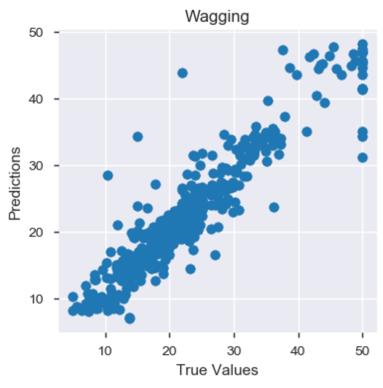
Table 5.11: Best results aggregated



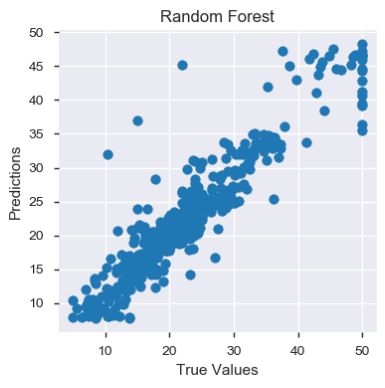
(a) DT



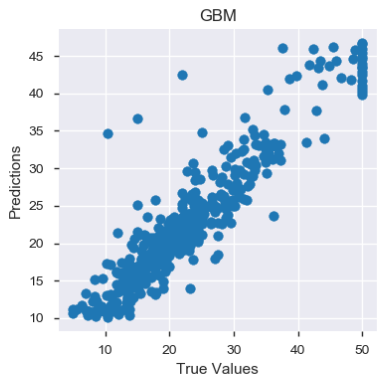
(b) BG



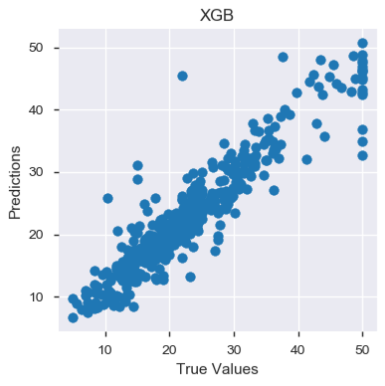
(c) WG



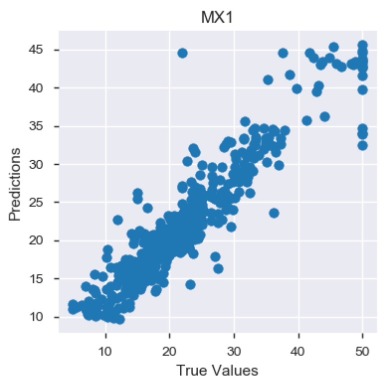
(d) RF



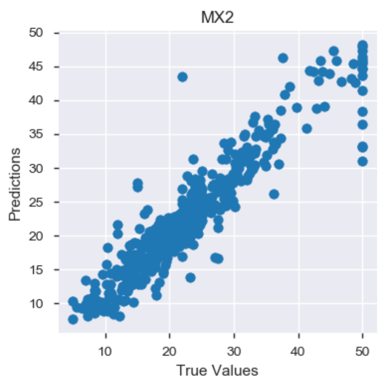
(e) GB



(f) XGB

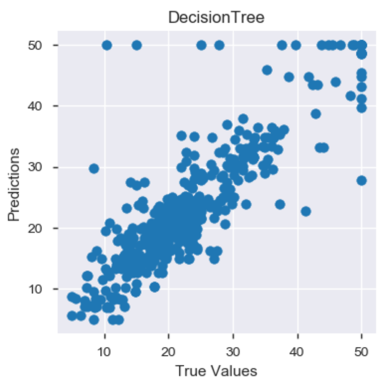


(g) MB1

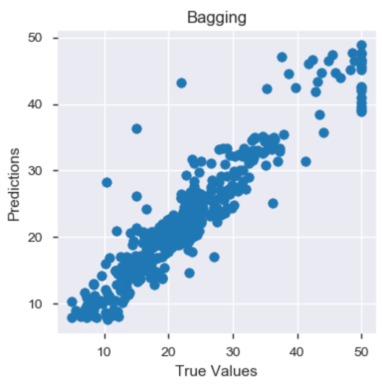


(h) MG2

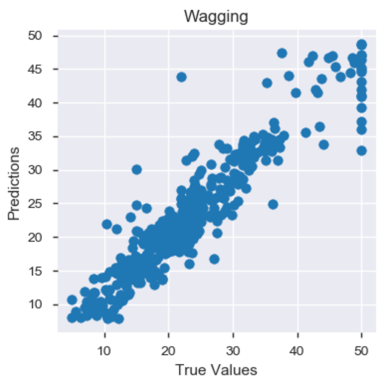
Figure 5.3: Boston - relation between true and predicted values



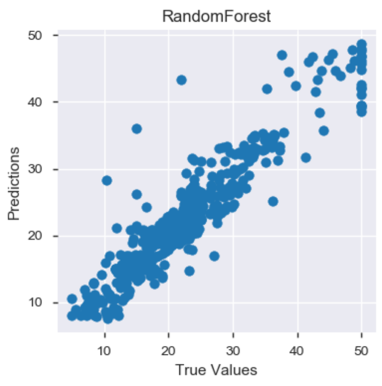
(a) DT



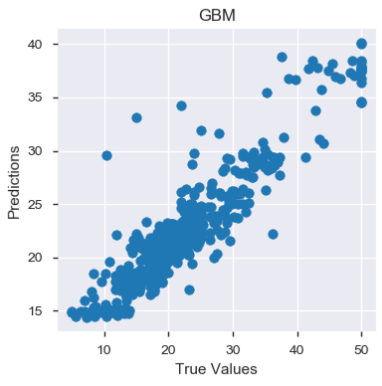
(b) BG



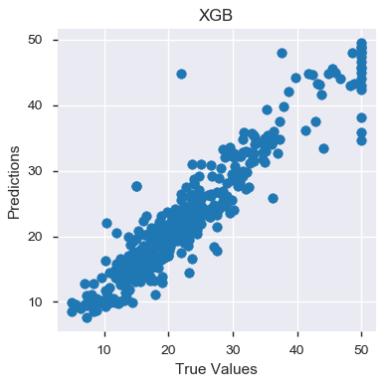
(c) WG



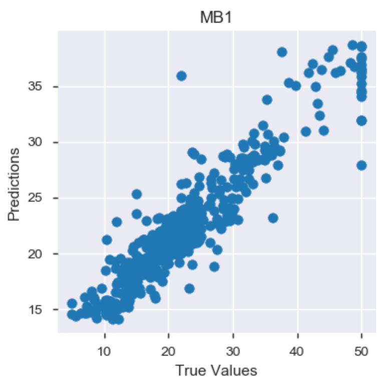
(d) RF



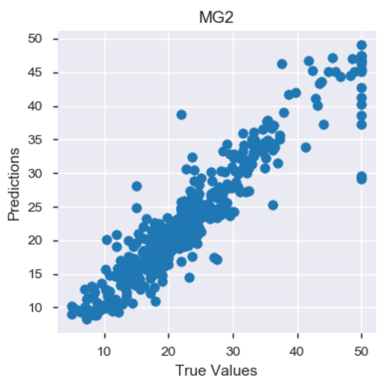
(e) GB



(f) XGB

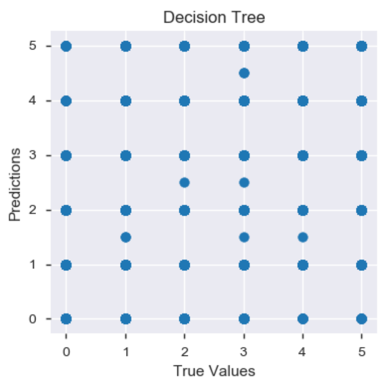


(g) MB1

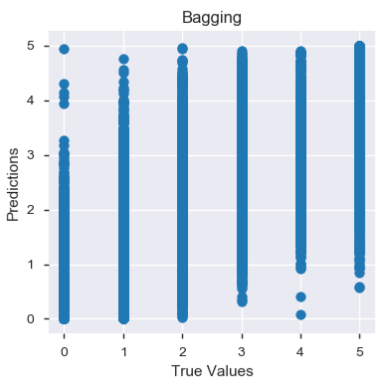


(h) MG2

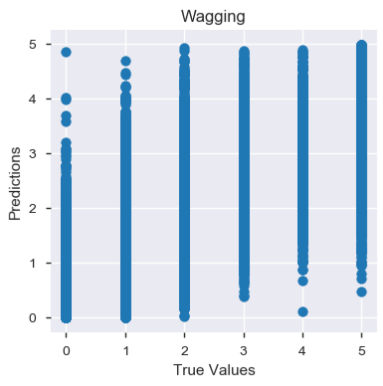
Figure 5.4: Real Estate - relation between true and predicted values



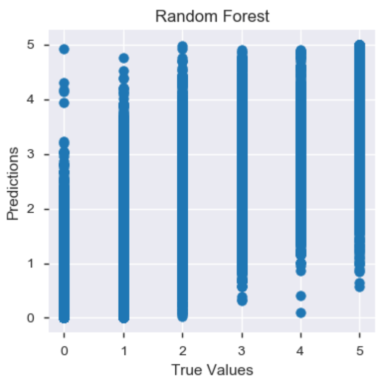
(a) DT



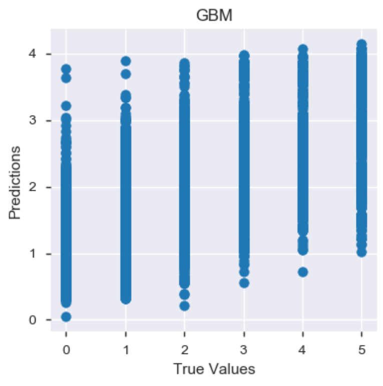
(b) BG



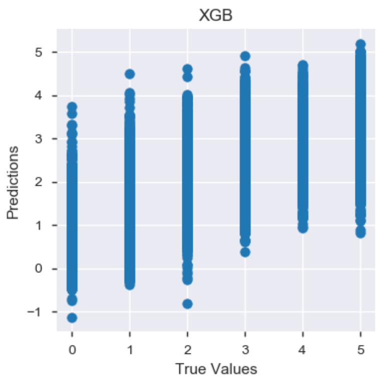
(c) WG



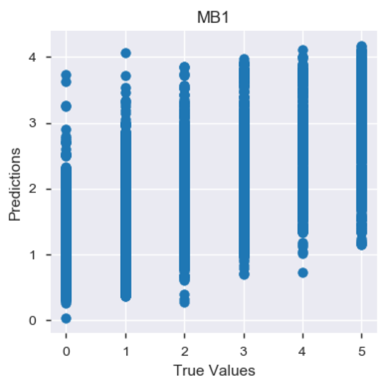
(d) RF



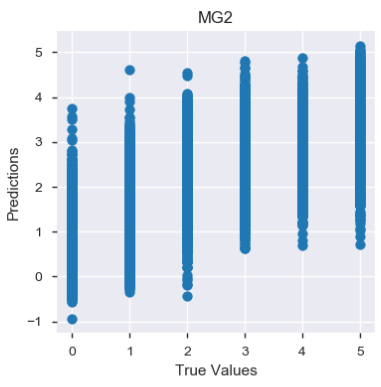
(e) GB



(f) XGB



(g) MB1



(h) MG2

Figure 5.5: Relation between true and predicted values

Chapter 6

Conclusions and Future Work

6.1 Satisfaction of Objectives

- Study and understand the bias/variance trade-off;
- Identification of which algorithms have a positive impact on the reduction of bias/variance;
- Try to obtain better predictive results working with some techniques, doing some experimentations.

Regarding the work done, and looking at our core objectives in 1.2 we think the first objective, study and understand the bias/variance trade-off was largely achieved, minutely explained and took into account at the implementation phase.

The second goal on the list, identification of which algorithms have a positive impact on the reduction of bias/variance, was also achieved. We clearly defined and measure two principal metrics on generalization error, are they the bias and variance, for each used technique.

Another thing we are satisfied is the possibility we have to already saw in action and perceive the strategies and methods in trend by different authors for a similar issue and the experiment taken in section 5.

In section 5, we made all the set up and ran different techniques, obtaining diverse results. The *MBB* technique, a hybrid technique using Wagging and XGradient Boosting, took us to a landing where we think it deserves more attention. It is in our opinion an approach to keep exploring.

6.2 Future Work

Some work needs to be done yet and this field of study proved to have different possible paths to follow. All kind of practical work like do much more simulations of results with some datasets, determine the generalization errors in each approach and compare with other ensemble learning models needs to be done. Furthermore, identify what techniques to use in order to generate better ensembles regarding which properties models need to have.

For that, we might have to implement and test the overall score of each approach and give a real tentative of mixing some other techniques to get a better predictive analysis than others already on the fly. We just have mixed Wagging with Gradient Boosting or Wagging with XGradient Boosting and maybe with other mixtures we can get further.

We are clearly convinced that within this philosophy, we can bring innovation by creating new ensembles and maybe better results.

Besides all, we also think we could explore more the idea addressed in chapter 3, namely, the Negative Correlation Learning 2.3.1 and the Stacking technique referred on 5.2 of page 25.

Another thing we can address in the future is not to limit the utilization only to decision trees and use for example also neural networks, because we only use decision trees but we could have used some other base learners in our set up process. In that case we can not only have different types of techniques combined but also different type of base learners. Evaluate those results looking at generalization error as Ueda and Nakato presented, i.e., look into the covariance component too.

Last but not least, the hyperparameter tuning could also have more attention and travel deep into that thematic.

But the main idea to absorb here is that this work was some kind abstract and we not only look into one dataset, we saw and analysed different scores, bias and variance weights at generalization error and dataset characteristics across different datasets. The dataset per se, need to have its own study and deserve analyse. Next coming soon idea is to use the approach we had in this work to submit a solution in Kaggle competition.

References

[Ana17] Clarivate Analytics. It's time to get the facts, 2017.

[Bea17] Andrew L. Beam. Deep Learning 101 - Part 1: History and Background, 2017.

[Bro09] Gavin Brown. Ensemble Learning, 2009.

[BSH$^+$17] Joshua Beemer, Kelly Spoon, Lingjun He, Juanjuan Fan, and Richard A Levine. Ensemble Learning for Estimating Individualized Treatment Effects in Student Success Studies. *International Journal of Artificial Intelligence in Education Official Journal of the International AIED Society*, 2017.

[Bur16] Jenna Burrell. How the machine 'thinks': Understanding opacity in machine learning algorithms. *Big Data & Society*, 3(1):205395171562251, jan 2016.

[BW03] Gavin Brown and Jeremy Wyatt. Negative Correlation Learning and the Ambiguity Family of Ensemble Methods. In *Multiple Classifier Systems, 4th International Workshop, MCS 2003, Guilford, UK*, pages 266–275, 2003.

[BWHY05] Gavin Brown, Jeremy Wyatt, Rachel Harris, and Xin Yao. Diversity Creation Methods: A Survey and Categorisation. Technical Report 1, University of Birmingham, School of Computer Science, 2005.

[Che16] Lujing Chen. Basic Ensemble Learning (Random Forest, AdaBoost, Gradient Boosting)- Step by Step Explained, 2016.

[CSC17] Rafael M O Cruz, Robert Sabourin, and George D C Cavalcanti. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, 41:195–216, 2017.

[CSS$^+$19] Michael Carraz, Jo Stichbury, Stijn Schuermans, Crocker Peter, Konstantinos Korakitis, and Christina Voskoglou. State of the art nation 16th edition. Technical report, Developer Economics, 2019.

[DK95] Thomas G Dietterich and Eun Bae Kong. Machine Learning Bias, Statistical Bias, and Statistical Variance of Decision Tree Algorithms. Technical report, Department of Computer Science, Oregon State University, 1995.

[DS79] B V Dasarathy and B V Sheela. A composite classifier system design: Concepts and methodology. *Proceedings of the IEEE*, 67:708–713, 1979.

[Dut09] Haimonti Dutta. Measuring diversity in regression emsembles. Technical report, 2009.

[FDCB$^+$14] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, Dinani Amorim, and Amorim Fernández-Delgado. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research*, pages 3133–3181, 2014.

[GDT$^+$13] James Gareth, Witten Daniela, Hastie Trevor, Tibshirani Rober, Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Springer Publishing Company, Incorporated ©2014, 2 edition, 2013.

[Gup17] Prashant Gupta. Balancing Bias and Variance to Control Errors in Machine Learning, 2017.

[HH18] Satoshi Hara and Kohei Hayashi. Making Tree Ensembles Interpretable: A Bayesian Model Selection Approach. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 77–85, jun 2018.

[IRC$^+$17] Fauzia Idrees, Muttukrishnan Rajarajan, Mauro Conti, Thomas M. Chen, and Yogachandran Rahulamathavan. *PIndroid: A novel Android malware detection system using ensemble learning methods*, volume 68. Elsevier Ltd, jul 2017.

[Kag14] Kaggle. Competitions - Kaggle, 2014.

[Koh95] Ron Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. Technical report, 1995.

[KS90] Lars Kai and Peter Salamon. Neural Network Ensembles. Technical report, 1990.

[KV95] Anders Krogh and Jesper Vedelsby. Neural Network Ensembles, Cross Validation, and Active Learning. Technical report, Denver, Colorado, 1995.

[LGS] Julien-Charles Lévesque, Christian Gagné, and Robert Sabourin. Bayesian Hyperparameter Optimization for Ensemble Learning. Technical report.

[LY99] Y Liu and X Yao. Ensemble learning via negative correlation. *Neural networks*, 12(10):1399–1404, 1999.

[MMSMS12] João Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa. Ensemble approaches for regression: A survey. *ACM Computing Surveys (CSUR)*, 45(1):1–40, 2012.

[MTBH05] Marianthi Markatou, Hong Tian, Shameek Biswas, and George Hripcsak. Analysis of Variance of Cross-Validation Estimators of the Generalization Error. Technical report, 2005.

[Pap20] David Paper. *Hands-on Scikit-Learn for Machine Learning Applications*. Apress, 2020.

[PGB10] Dmitry Pavlov, Alexey Gorodilov, and Cliff Brunk. BagBoo: A Scalable Hybrid Bagging-the-Boosting Model. 2010.

[PMSR14] Fábio Pinto, João Mendes Moreira, Carlos Soares, and R.J.F. Rossetti. Simulation of the ensemble generation process: The divergence between data and model similarity. In *Modelling and Simulation 2014 - European Simulation and Modelling Conference, ESM 2014*, 2014.

[Pol09] R. Polikar. Ensemble learning. *Ensemble learning*, 4(1):2776, 2009.

[PSV] Sakrapee Paisitkriangkrai, Chunhua Shen, and Anton Van Den Hengel. Strengthening the Effectiveness of Pedestrian Detection with Spatially Pooled Features *. Technical report.

[Roc19] Joseph Rocca. Ensemble methods: bagging, boosting and stacking – Towards Data Science, 2019.

[Sin18a] Seema Singh. Cousins of Artificial Intelligence - Towards Data Science, 2018.

[Sin18b] Seema Singh. Understanding the Bias-Variance Tradeoff – Towards Data Science, 2018.

[SR18] Omer Sagi and Lior Rokach. Ensemble learning: A survey, 2018.

[Stu92] René Doursat Stuart Geman, Elie Bienenstock. Neural Networks and the Bias/-Variance Dilemma. Technical report, MIT, 1992.

[UN96] Naonori Ueda and Ryohei Nakano. Generalization error of ensemble estimators. *Proceedings of International Conference on Neural Networks (ICNN'96)*, 1:90–95, 1996.

[Web00] Geoffrey I. Webb. MultiBoosting: a technique for combining boosting and wagging. Technical Report 2, 2000.

[Win] CSC2515 Winter 2015 Introduction to Machine Learning - Combining Models.

[Wol92] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, jan 1992.

[XR09] Jianjun Xie and V Rojkova. A Combination of Boosting and Bagging for KDD Cup 2009-Fast Scoring on a Large Database. Technical report, 2009.

[XRC$^+$] Jianjun Xie, Viktoria Rojkova, Stephen Coggeshall, Gideon Dror, Marc Boullé, Isabelle Guyon, Vincent Lemaire, and David Vogel. A Combination of Boosting and Bagging for KDD Cup 2009-Fast Scoring on a Large Database. Technical report.

[YZS19] Zebin Yang, Aijun Zhang, and Agus Sudjianto. Enhancing Explainability of Neural Networks through Architecture Constraints. Technical report, Department of Statistics and Actuarial Science, The University of Hong Kong and Corporate Model Risk, Wells Fargo, USA, 2019.

[Zou17] James Zou. Basic Recipe for Machine Learning (C2W1L03), 2017.

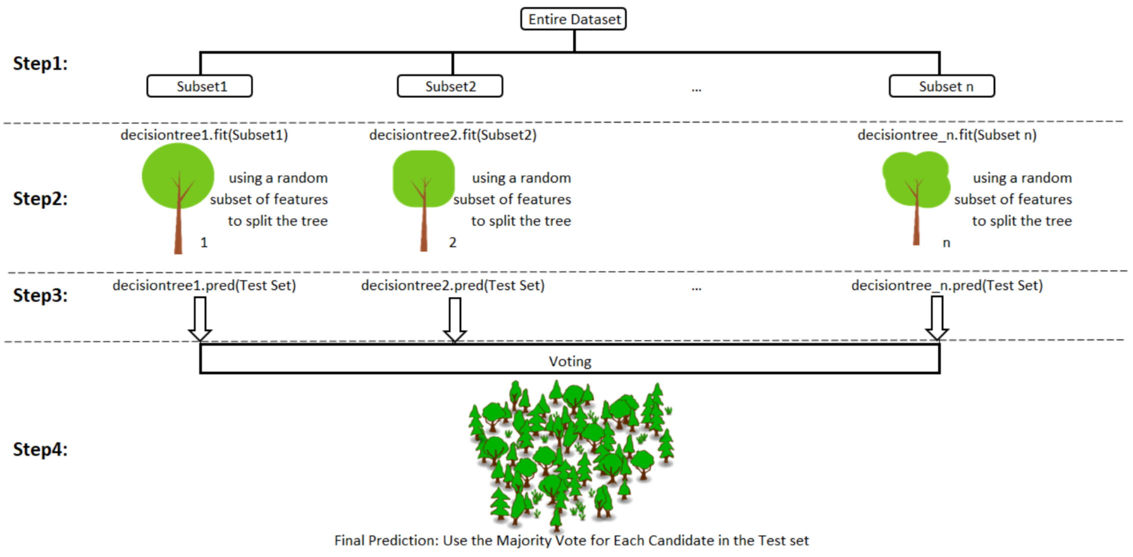
[ZW98] Z Zheng and G Webb. Multiple Boosting: A combination of Boosting and Bagging. Technical report, 1998.

REFERENCES

Appendix A

Ensemble Models Procedures [1]

A.1 Random Forest



Entire Dataset
Step1:
Subset1
Subset2
...
Subset n
Step2:
decisiontree1.fit(Subset1)
decisiontree2.fit(Subset2)
decisiontree_n.fit(Subset n)
using a random subset of features to split the tree
1
2
n
Step3:
decisiontree1.pred(Test Set)
decisiontree2.pred(Test Set)
decisiontree_n.pred(Test Set)
Voting
Step4:
Final Prediction: Use the Majority Vote for Each Candidate in the Test set

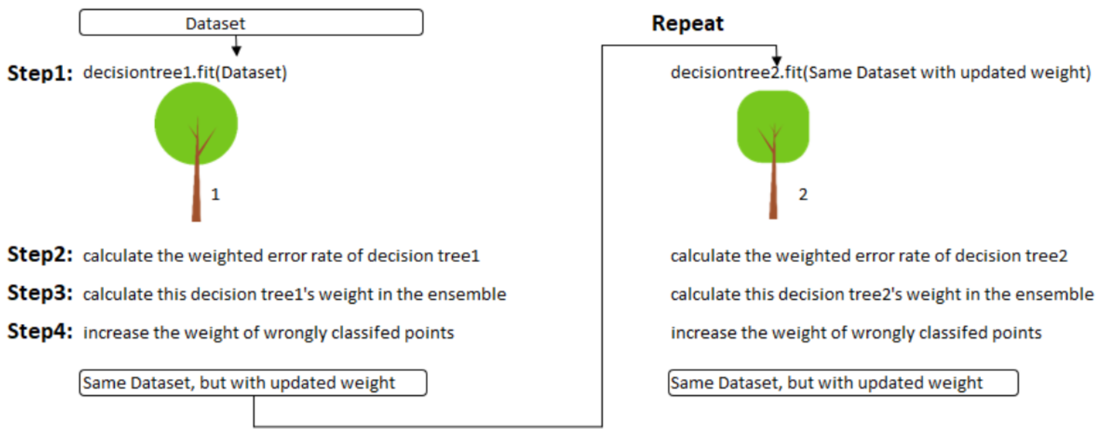
Figure A.1: Random Forest Model

1. Select a random number of subsets from the training set, let's say N.
2. After that train N decision trees.
   - one random subset is used to train one decision tree
   - the optimal splits for each decision tree are based on a random subset of features (e.g. 10 features in total, randomly select 5 out of 10 features to split)
3. Each individual tree predicts the candidates in the test set, independently.
4. Make the final prediction.

   For each candidate in the test set, Random Forest uses the class with the majority vote as this candidate's final prediction. Of course, our 1000 trees are the parliament here.

[1] Based on [Zou17]

A.2 Adaboost



Dataset
Step1: decisiontree1.fit(Dataset)
1
Step2: calculate the weighted error rate of decision tree1
Step3: calculate this decision tree1's weight in the ensemble
Step4: increase the weight of wrongly classifed points
Same Dataset, but with updated weight
Repeat
decisiontree2.fit(Same Dataset with updated weight)
2
calculate the weighted error rate of decision tree2
calculate this decision tree2's weight in the ensemble
increase the weight of wrongly classifed points
Same Dataset, but with updated weight

Figure A.2: Adaboost Model

1. Initialize the weights of each single data point. Divide uniformly by the total number of points to know the first weights.
2. Train a DT
3. Determine the error rate of the DT
4. Calculate this DT's weight in the ensemble
    - The higher weighted error rate of a tree, the less decision power the tree will be given during the later voting
    - The lower weighted error rate of a tree, the higher decision power the tree will be given during the later voting
5. Update weights of wrongly classified points. The weight of each data point is equal to:
    - If the model got this data point correct, the weight stays the same
    - If the model got this data point wrong, the new weight of this point = old weight * np.exp(weight of this tree)
6. Repeat first step unless all the trees were trained
7. Make the final prediction

Adaboost technique combines tree weights in order to obtain a new prediction.

Clearly, the impact of each tree is related with the weight it has. The more weight the greater impact it will have. AdaBoost makes another forecast looking to each weight and multiplying by the expected value for that tree as we can see in step 5 above.

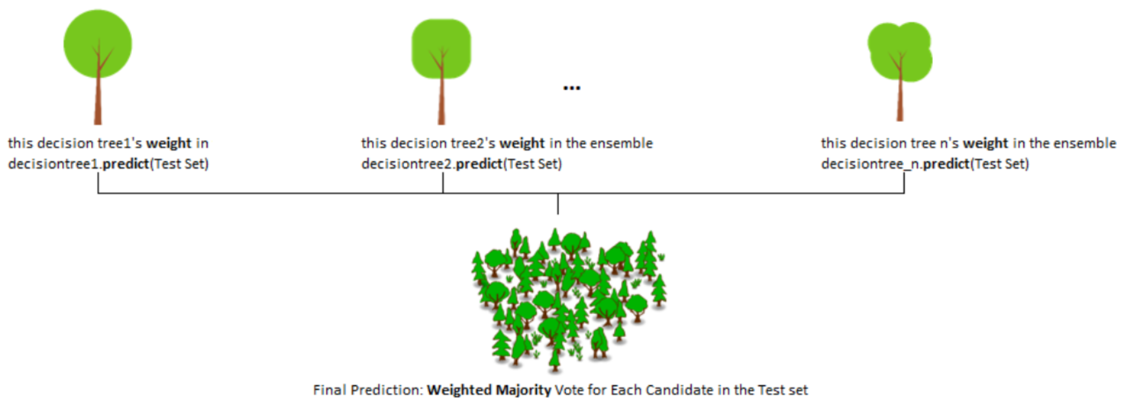

this decision tree1's weight in
decisiontree1.predict(Test Set)
this decision tree2's weight in the ensemble
decisiontree2.predict(Test Set)
...
this decision tree n's weight in the ensemble
decisiontree_n.predict(Test Set)
Final Prediction: Weighted Majority Vote for Each Candidate in the Test set

Figure A.3: Adaboost Model Final Prediction

A.3 Gradient Boosting

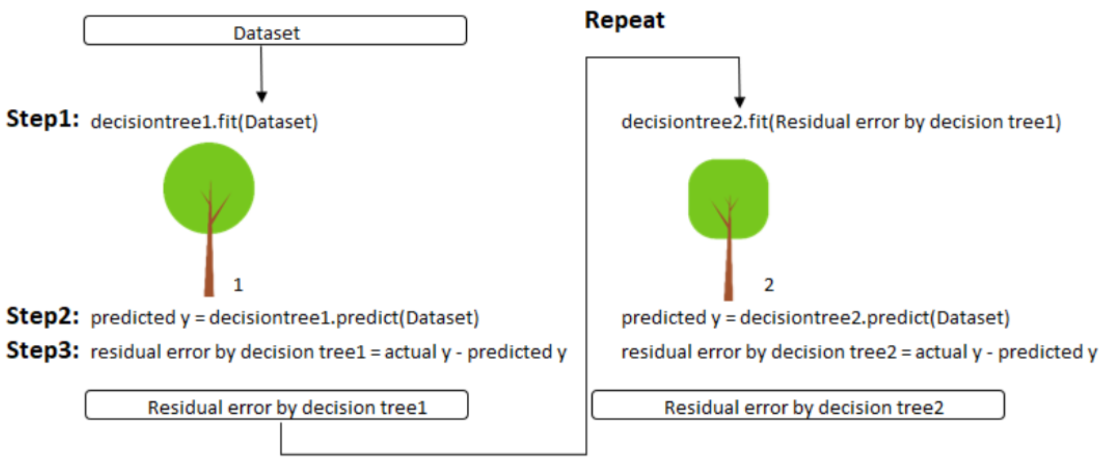

Dataset
Repeat
Step1: decisiontree1.fit(Dataset)
decisiontree2.fit(Residual error by decision tree1)
1
2
Step2: predicted y = decisiontree1.predict(Dataset)
Step3: residual error by decision tree1 = actual y - predicted y
predicted y = decisiontree2.predict(Dataset)
residual error by decision tree2 = actual y - predicted y
Residual error by decision tree1
Residual error by decision tree2

Figure A.4: Gradient Boosting Model

1. Train a DT
2. Apply the trained DT to prediction
3. Determine the residual of this DT
4. Repeat the first step, unless we train already all the trees
5. Make the final prediction

The Gradient Boosting makes a new prediction by simply adding up the predictions of all trees.

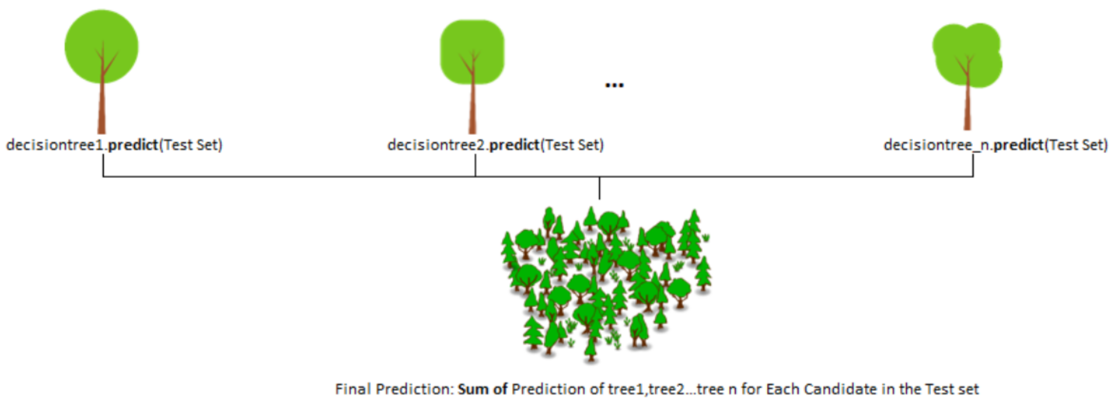
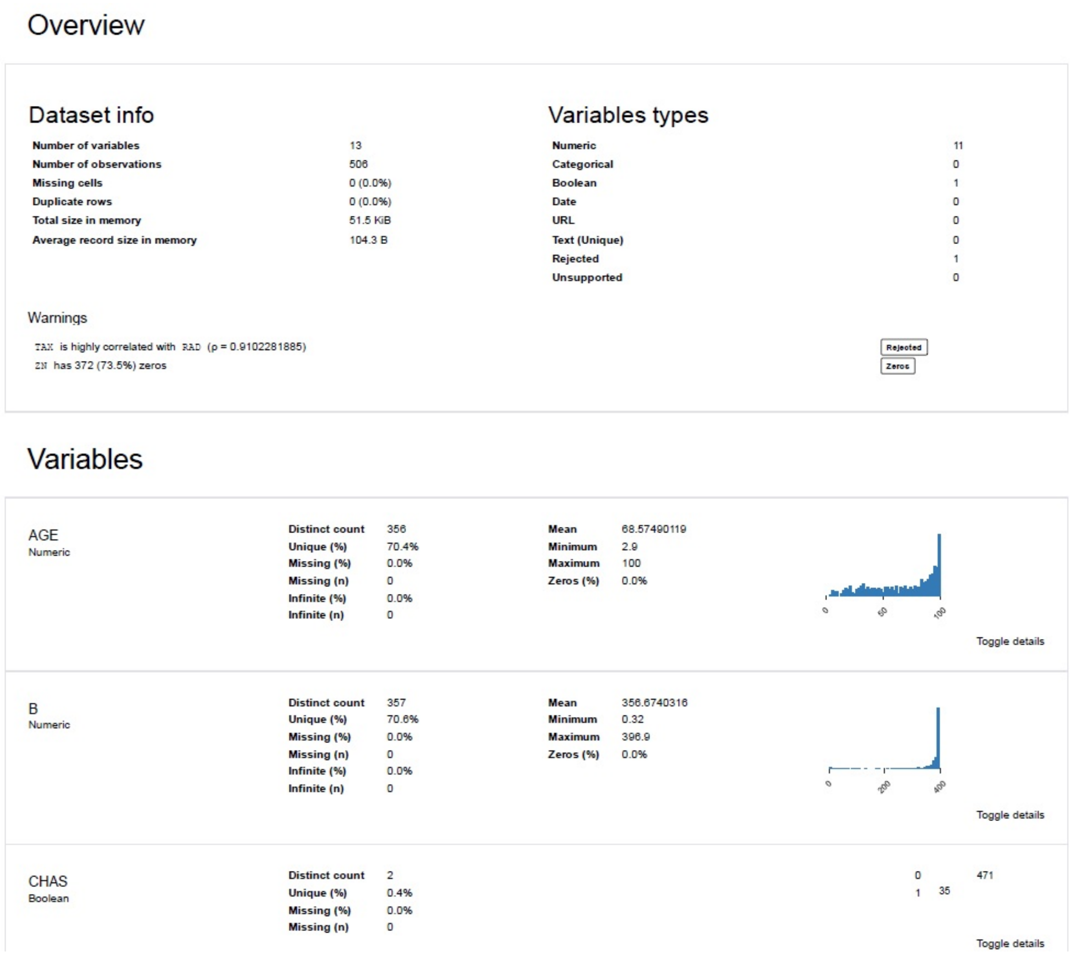

decisiontree1.predict(Test Set)
decisiontree2.predict(Test Set)
...
decisiontree_n.predict(Test Set)
Final Prediction: Sum of Prediction of tree1,tree2...tree n for Each Candidate in the Test set

Figure A.5: Gradient Boosting Model Final Prediction

Appendix B

Explored Datasets [1]

B.1 Boston house prices



Overview

Dataset info

Number of variables	13
Number of observations	506
Missing cells	0 (0.0%)
Duplicate rows	0 (0.0%)
Total size in memory	51.5 KiB
Average record size in memory	104.3 B

Variables types

Numeric	11
Categorical	0
Boolean	1
Date	0
URL	0
Text (Unique)	0
Rejected	1
Unsupported	0

Warnings

TAX is highly correlated with RAD (ρ = 0.9102281885) Rejected
ZN has 372 (73.5%) zeros Zeros

Variables

AGE
Numeric

Distinct count	356	Mean	68.57490119
Unique (%)	70.4%	Minimum	2.9
Missing (%)	0.0%	Maximum	100
Missing (n)	0	Zeros (%)	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

B
Numeric

Distinct count	357	Mean	356.6740316
Unique (%)	70.6%	Minimum	0.32
Missing (%)	0.0%	Maximum	396.9
Missing (n)	0	Zeros (%)	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

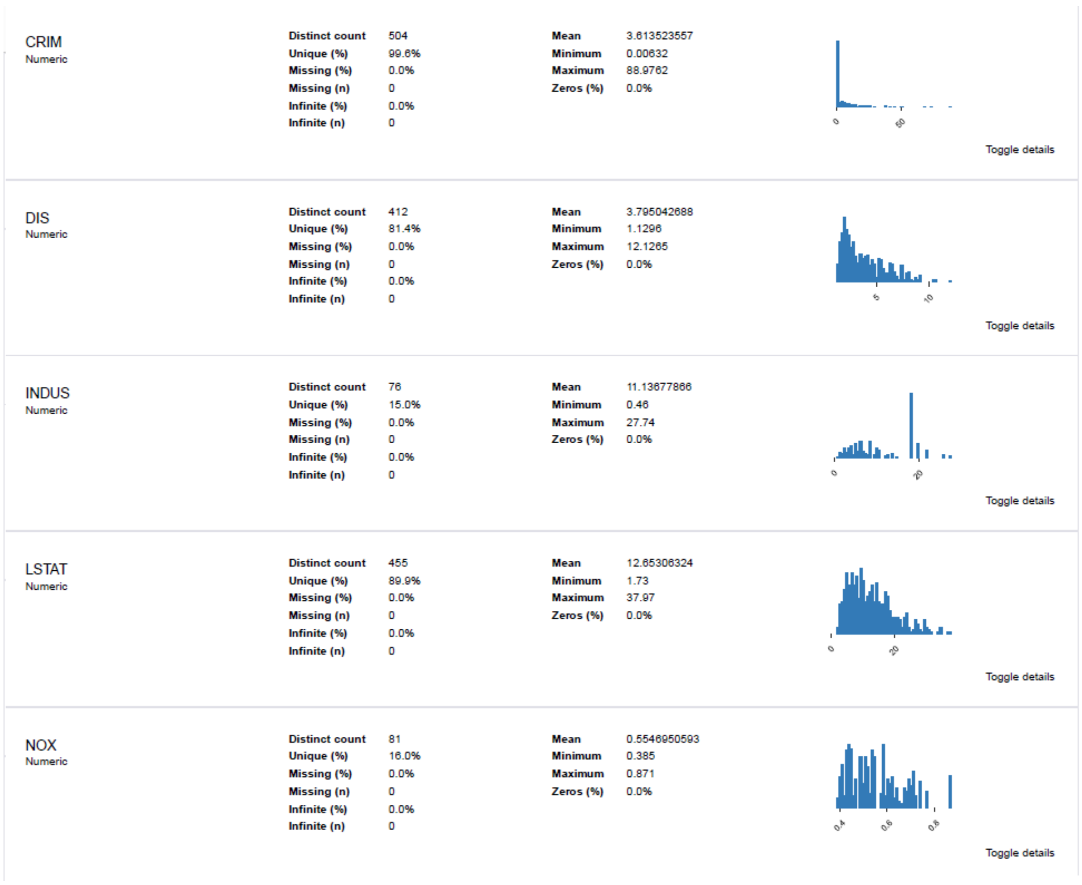
CHAS
Boolean

Distinct count	2
Unique (%)	0.4%
Missing (%)	0.0%
Missing (n)	0

0	471
1	35

Toggle details

[1] Reports generated with python library pandas-profiling



CRIM
Numeric

Distinct count	504	**Mean**	3.613523557
Unique (%)	99.6%	**Minimum**	0.00632
Missing (%)	0.0%	**Maximum**	88.9762
Missing (n)	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

DIS
Numeric

Distinct count	412	**Mean**	3.795042688
Unique (%)	81.4%	**Minimum**	1.1296
Missing (%)	0.0%	**Maximum**	12.1265
Missing (n)	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

INDUS
Numeric

Distinct count	76	**Mean**	11.13677866
Unique (%)	15.0%	**Minimum**	0.46
Missing (%)	0.0%	**Maximum**	27.74
Missing (n)	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

LSTAT
Numeric

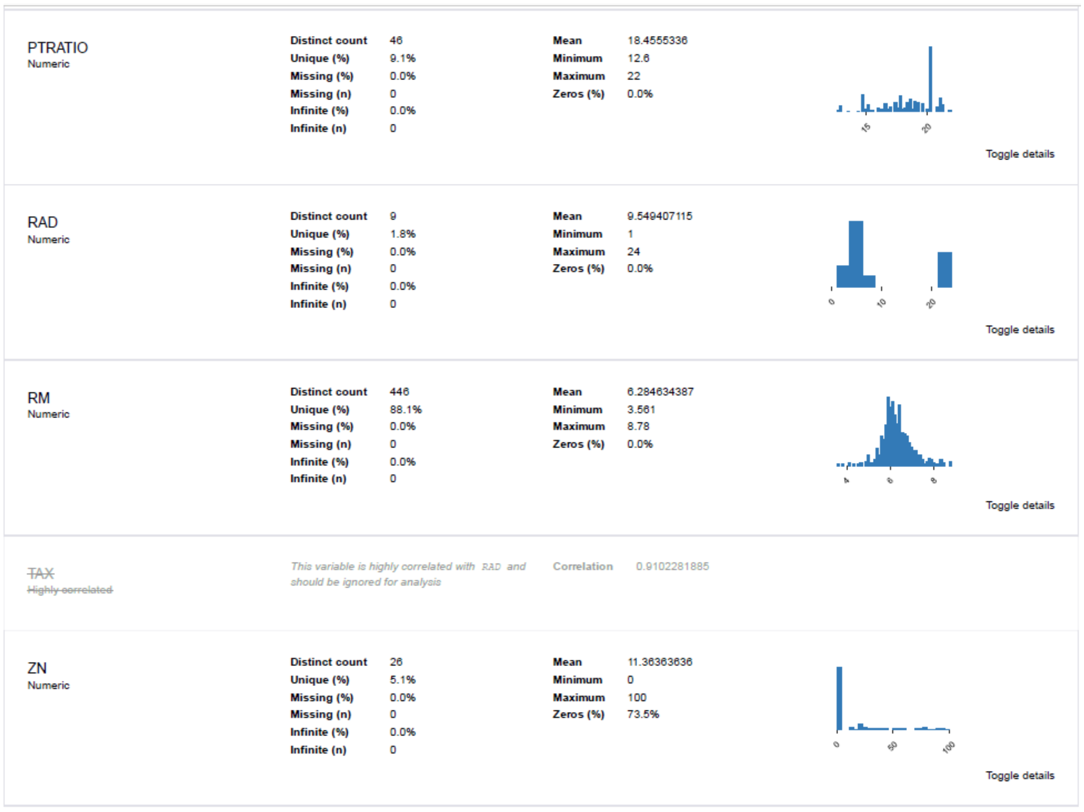
Distinct count	455	**Mean**	12.65306324
Unique (%)	89.9%	**Minimum**	1.73
Missing (%)	0.0%	**Maximum**	37.97
Missing (n)	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

NOX
Numeric

Distinct count	81	**Mean**	0.5546950593
Unique (%)	16.0%	**Minimum**	0.385
Missing (%)	0.0%	**Maximum**	0.871
Missing (n)	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details



PTRATIO
Numeric

Distinct count	46	**Mean**	18.4555336
Unique (%)	9.1%	**Minimum**	12.6
Missing (%)	0.0%	**Maximum**	22
Missing (n)	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

RAD
Numeric

Distinct count	9	**Mean**	9.549407115
Unique (%)	1.8%	**Minimum**	1
Missing (%)	0.0%	**Maximum**	24
Missing (n)	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

RM
Numeric

Distinct count	446	**Mean**	6.284634387
Unique (%)	88.1%	**Minimum**	3.561
Missing (%)	0.0%	**Maximum**	8.78
Missing (n)	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

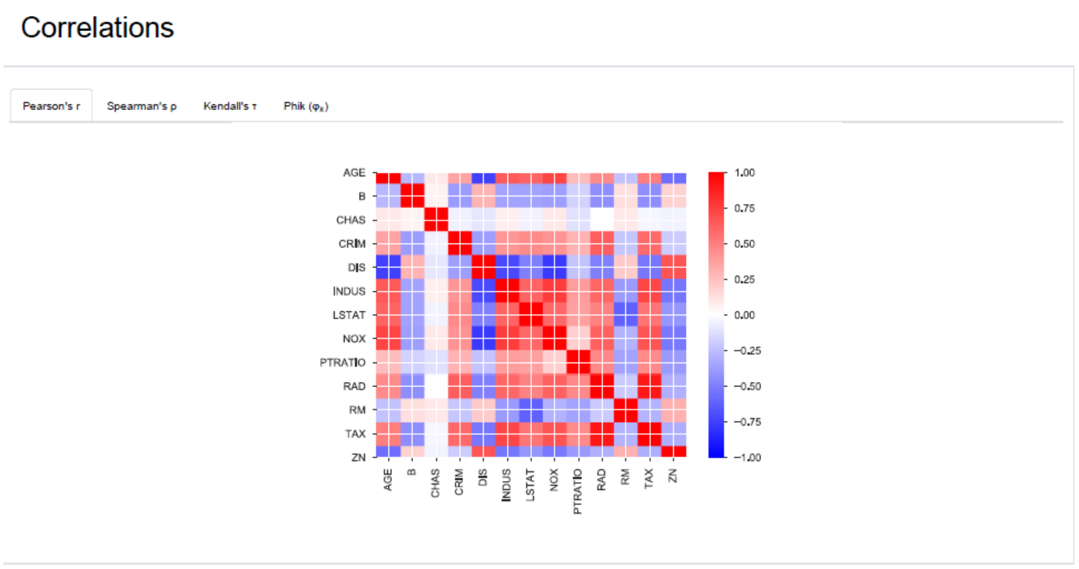
~~TAX~~
~~Highly correlated~~

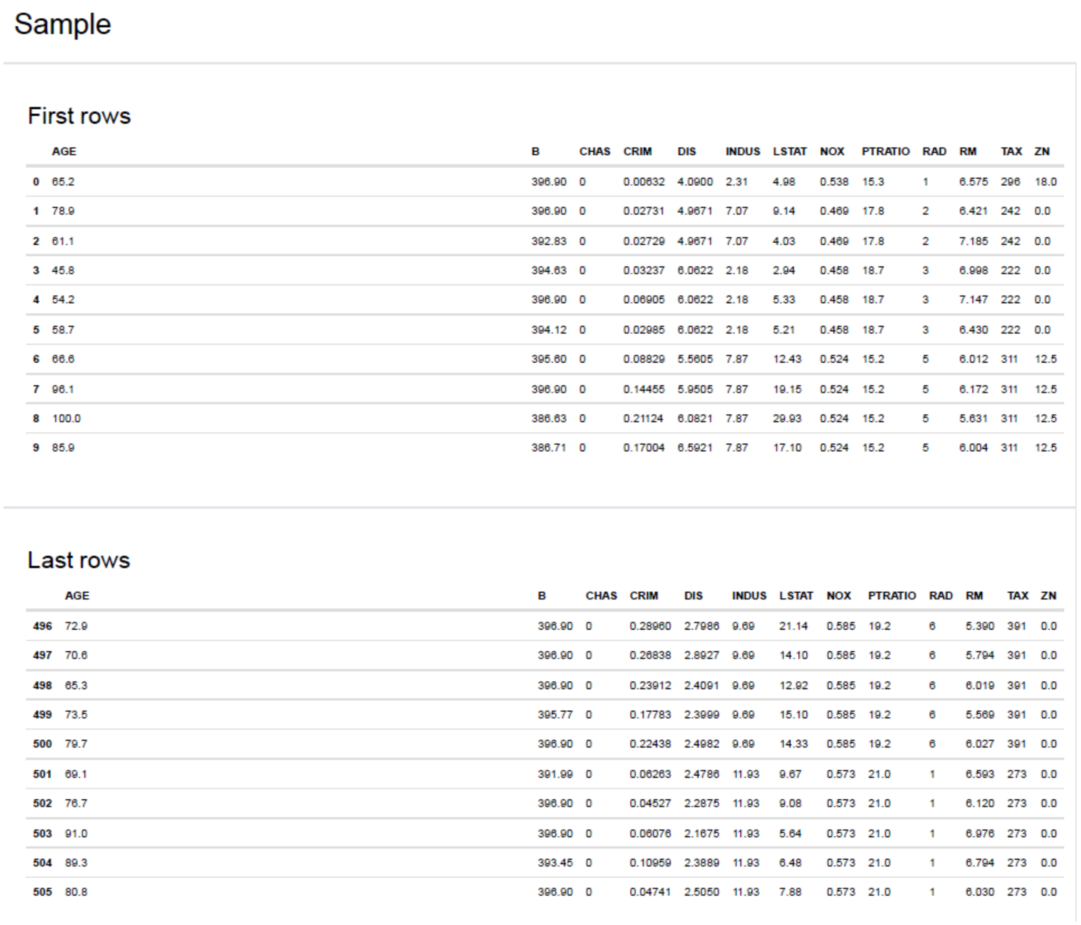
This variable is highly correlated with `RAD` *and should be ignored for analysis* Correlation 0.9102281885

ZN
Numeric

Distinct count	26	**Mean**	11.36363636
Unique (%)	5.1%	**Minimum**	0
Missing (%)	0.0%	**Maximum**	100
Missing (n)	0	**Zeros (%)**	73.5%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details





Sample

First rows

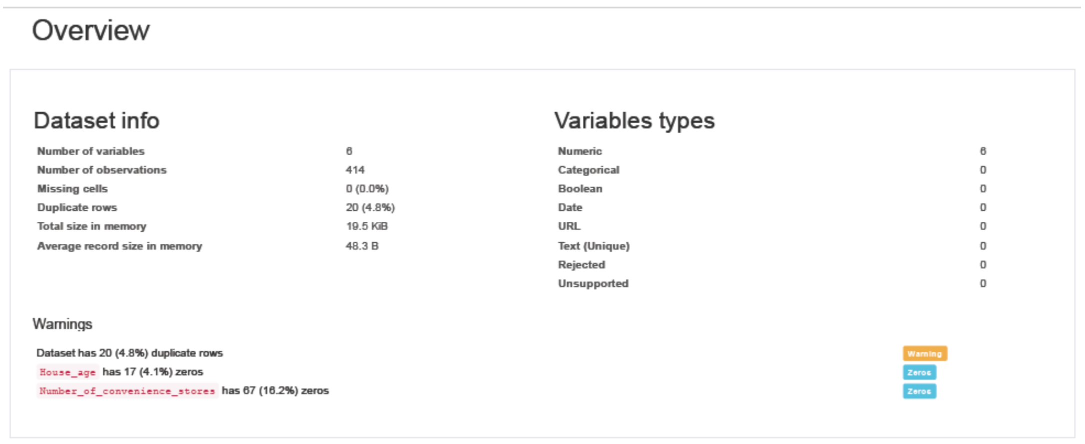
	AGE	B	CHAS	CRIM	DIS	INDUS	LSTAT	NOX	PTRATIO	RAD	RM	TAX	ZN
0	65.2	396.90	0	0.00632	4.0900	2.31	4.98	0.538	15.3	1	6.575	296	18.0
1	78.9	396.90	0	0.02731	4.9671	7.07	9.14	0.469	17.8	2	6.421	242	0.0
2	61.1	392.83	0	0.02729	4.9671	7.07	4.03	0.469	17.8	2	7.185	242	0.0
3	45.8	394.63	0	0.03237	6.0622	2.18	2.94	0.458	18.7	3	6.998	222	0.0
4	54.2	396.90	0	0.06905	6.0622	2.18	5.33	0.458	18.7	3	7.147	222	0.0
5	58.7	394.12	0	0.02985	6.0622	2.18	5.21	0.458	18.7	3	6.430	222	0.0
6	66.6	395.60	0	0.08829	5.5605	7.87	12.43	0.524	15.2	5	6.012	311	12.5
7	96.1	396.90	0	0.14455	5.9505	7.87	19.15	0.524	15.2	5	6.172	311	12.5
8	100.0	386.63	0	0.21124	6.0821	7.87	29.93	0.524	15.2	5	5.631	311	12.5
9	85.9	386.71	0	0.17004	6.5921	7.87	17.10	0.524	15.2	5	6.004	311	12.5

Last rows

	AGE	B	CHAS	CRIM	DIS	INDUS	LSTAT	NOX	PTRATIO	RAD	RM	TAX	ZN
496	72.9	396.90	0	0.28960	2.7986	9.69	21.14	0.585	19.2	6	5.390	391	0.0
497	70.6	396.90	0	0.26838	2.8927	9.69	14.10	0.585	19.2	6	5.794	391	0.0
498	65.3	396.90	0	0.23912	2.4091	9.69	12.92	0.585	19.2	6	6.019	391	0.0
499	73.5	395.77	0	0.17783	2.3999	9.69	15.10	0.585	19.2	6	5.569	391	0.0
500	79.7	396.90	0	0.22438	2.4982	9.69	14.33	0.585	19.2	6	6.027	391	0.0
501	69.1	391.99	0	0.06263	2.4786	11.93	9.67	0.573	21.0	1	6.593	273	0.0
502	76.7	396.90	0	0.04527	2.2875	11.93	9.08	0.573	21.0	1	6.120	273	0.0
503	91.0	396.90	0	0.06076	2.1675	11.93	5.64	0.573	21.0	1	6.976	273	0.0
504	89.3	393.45	0	0.10959	2.3889	11.93	6.48	0.573	21.0	1	6.794	273	0.0
505	80.8	396.90	0	0.04741	2.5050	11.93	7.88	0.573	21.0	1	6.030	273	0.0

You can download all profiling measure made in the next url `https://github.com/nmvalente/diss/blob/master/boston_profiling.html` and navigate better.

B.2 Real estate



Overview

Dataset info

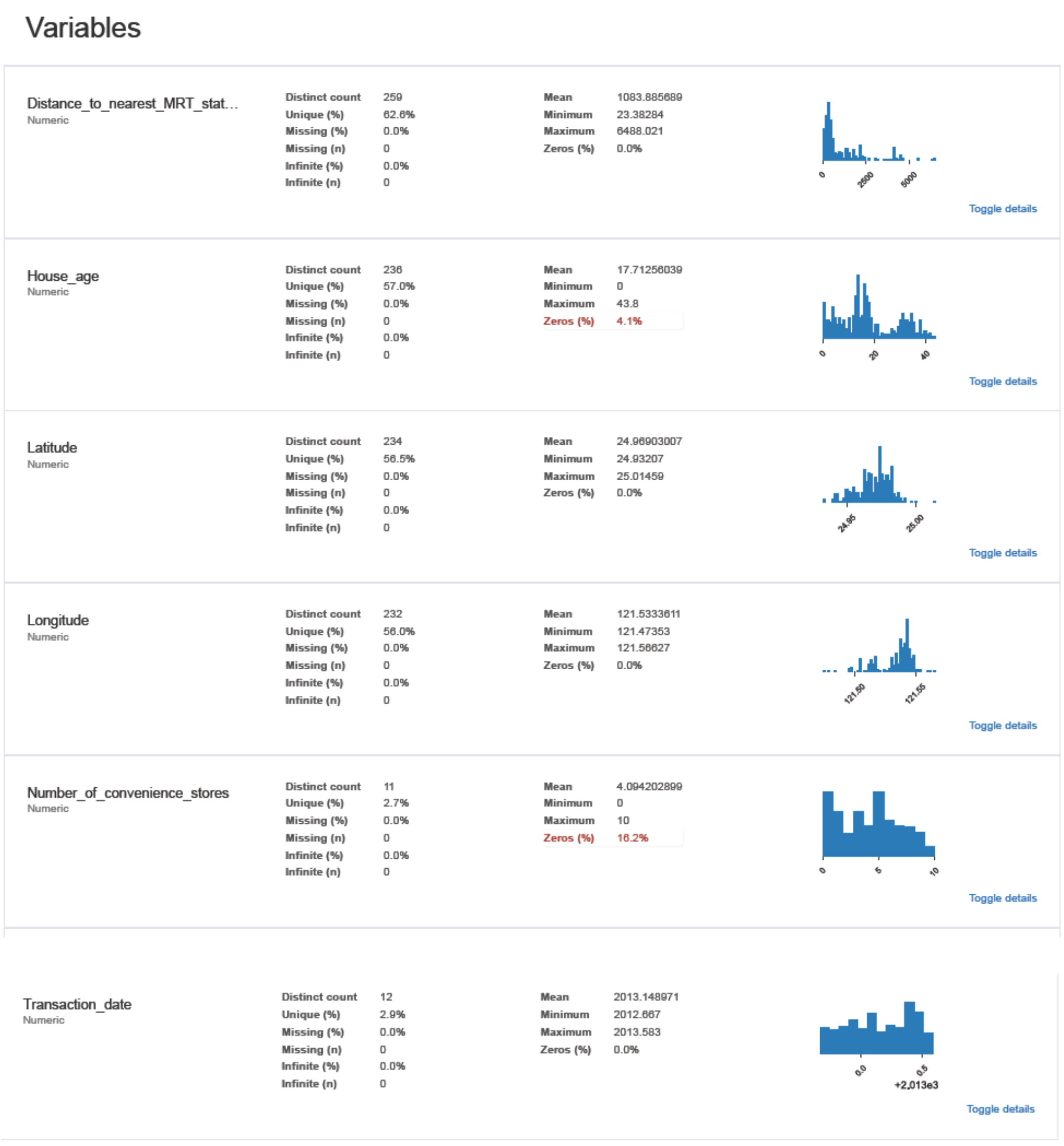
Number of variables	6
Number of observations	414
Missing cells	0 (0.0%)
Duplicate rows	20 (4.8%)
Total size in memory	19.5 KiB
Average record size in memory	48.3 B

Variables types

Numeric	6
Categorical	0
Boolean	0
Date	0
URL	0
Text (Unique)	0
Rejected	0
Unsupported	0

Warnings

Dataset has 20 (4.8%) duplicate rows	Warning
`House_age` has 17 (4.1%) zeros	Zeros
`Number_of_convenience_stores` has 67 (16.2%) zeros	Zeros



Variables

Distance_to_nearest_MRT_stat…
Numeric

Distinct count	259	Mean	1083.885689
Unique (%)	62.6%	Minimum	23.38284
Missing (%)	0.0%	Maximum	6488.021
Missing (n)	0	Zeros (%)	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

House_age
Numeric

Distinct count	236	Mean	17.71256039
Unique (%)	57.0%	Minimum	0
Missing (%)	0.0%	Maximum	43.8
Missing (n)	0	Zeros (%)	4.1%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

Latitude
Numeric

Distinct count	234	Mean	24.96903007
Unique (%)	56.5%	Minimum	24.93207
Missing (%)	0.0%	Maximum	25.01459
Missing (n)	0	Zeros (%)	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

Longitude
Numeric

Distinct count	232	Mean	121.5333611
Unique (%)	56.0%	Minimum	121.47353
Missing (%)	0.0%	Maximum	121.56627
Missing (n)	0	Zeros (%)	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

Number_of_convenience_stores
Numeric

Distinct count	11	Mean	4.094202899
Unique (%)	2.7%	Minimum	0
Missing (%)	0.0%	Maximum	10
Missing (n)	0	Zeros (%)	16.2%
Infinite (%)	0.0%		
Infinite (n)	0		

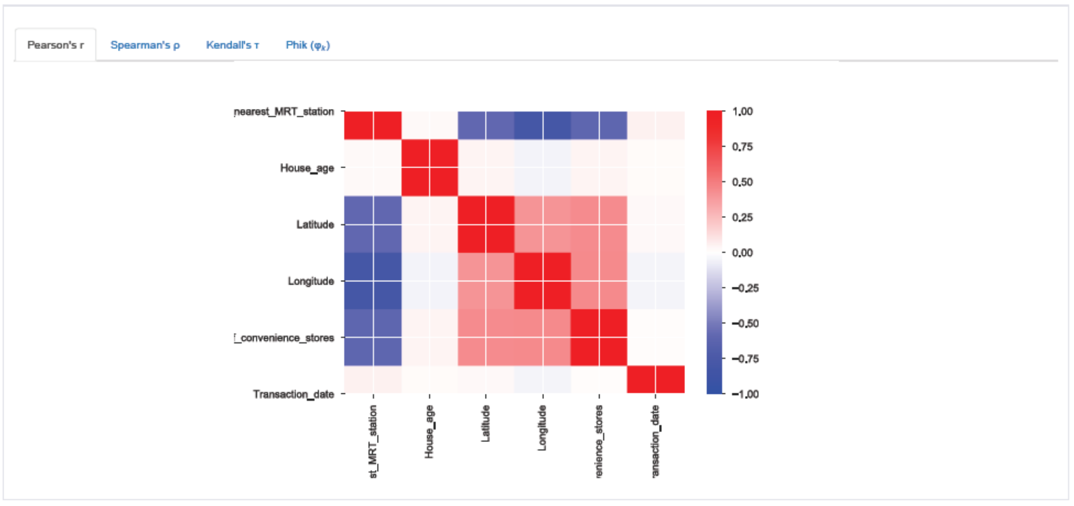
Toggle details

Transaction_date
Numeric

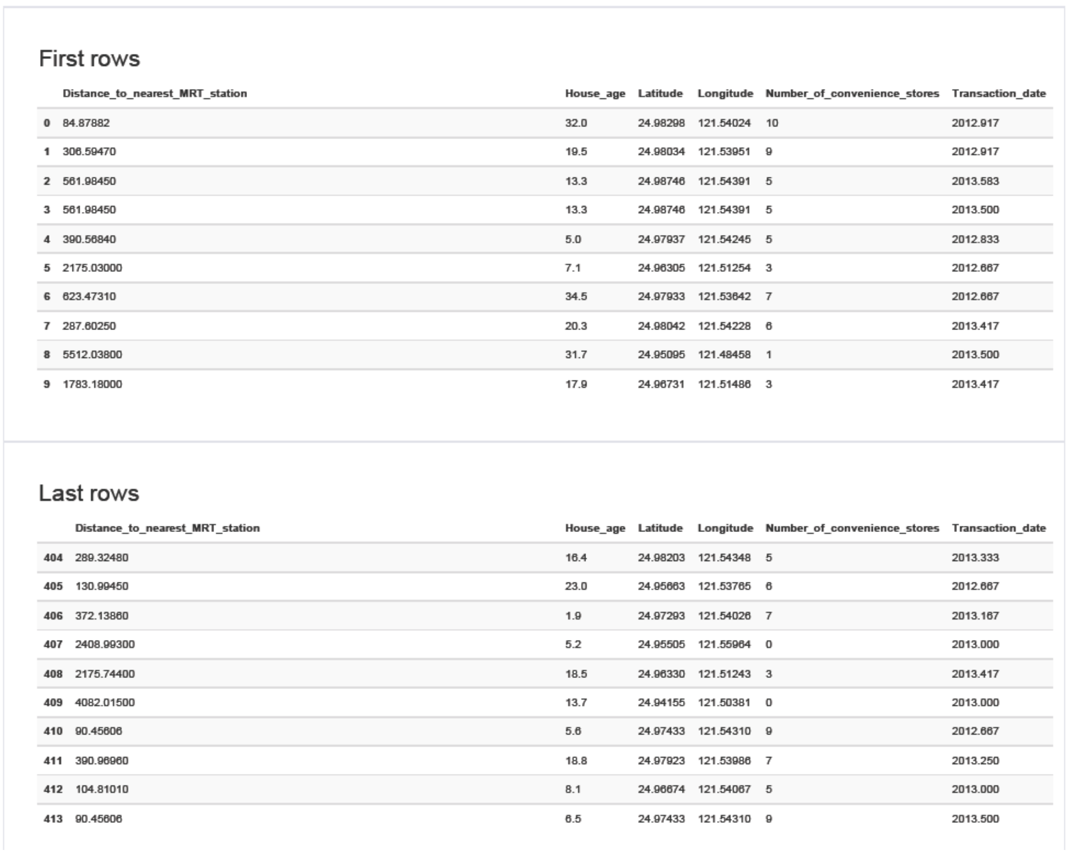
Distinct count	12	Mean	2013.148971
Unique (%)	2.9%	Minimum	2012.667
Missing (%)	0.0%	Maximum	2013.583
Missing (n)	0	Zeros (%)	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Toggle details

Correlations



Sample



First rows

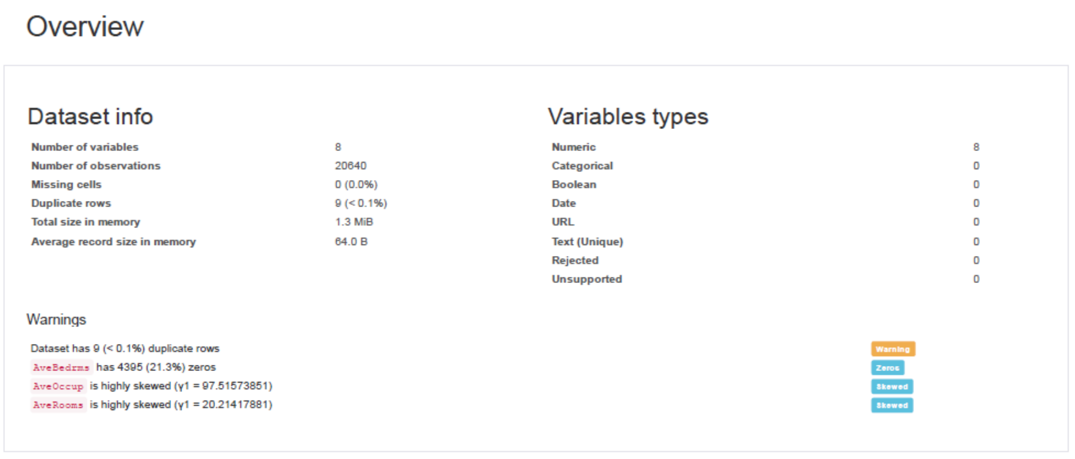
	Distance_to_nearest_MRT_station	House_age	Latitude	Longitude	Number_of_convenience_stores	Transaction_date
0	84.87882	32.0	24.98298	121.54024	10	2012.917
1	306.59470	19.5	24.98034	121.53951	9	2012.917
2	561.98450	13.3	24.98746	121.54391	5	2013.583
3	561.98450	13.3	24.98746	121.54391	5	2013.500
4	390.56840	5.0	24.97937	121.54245	5	2012.833
5	2175.03000	7.1	24.96305	121.51254	3	2012.667
6	623.47310	34.5	24.97933	121.53642	7	2012.667
7	287.60250	20.3	24.98042	121.54228	6	2013.417
8	5512.03800	31.7	24.95095	121.48458	1	2013.500
9	1783.18000	17.9	24.96731	121.51486	3	2013.417

Last rows

	Distance_to_nearest_MRT_station	House_age	Latitude	Longitude	Number_of_convenience_stores	Transaction_date
404	289.32480	16.4	24.98203	121.54348	5	2013.333
405	130.99450	23.0	24.95663	121.53765	6	2012.667
406	372.13860	1.9	24.97293	121.54026	7	2013.167
407	2408.99300	5.2	24.95505	121.55964	0	2013.000
408	2175.74400	18.5	24.96330	121.51243	3	2013.417
409	4082.01500	13.7	24.94155	121.50381	0	2013.000
410	90.45606	5.6	24.97433	121.54310	9	2012.667
411	390.96960	18.8	24.97923	121.53986	7	2013.250
412	104.81010	8.1	24.96674	121.54067	5	2013.000
413	90.45606	6.5	24.97433	121.54310	9	2013.500

You can download all profiling measure made in the next url `https://github.com/nmvalente/diss/blob/master/real_estate_valuation_profiling.html` and navigate better.

B.3 California



Overview

Dataset info

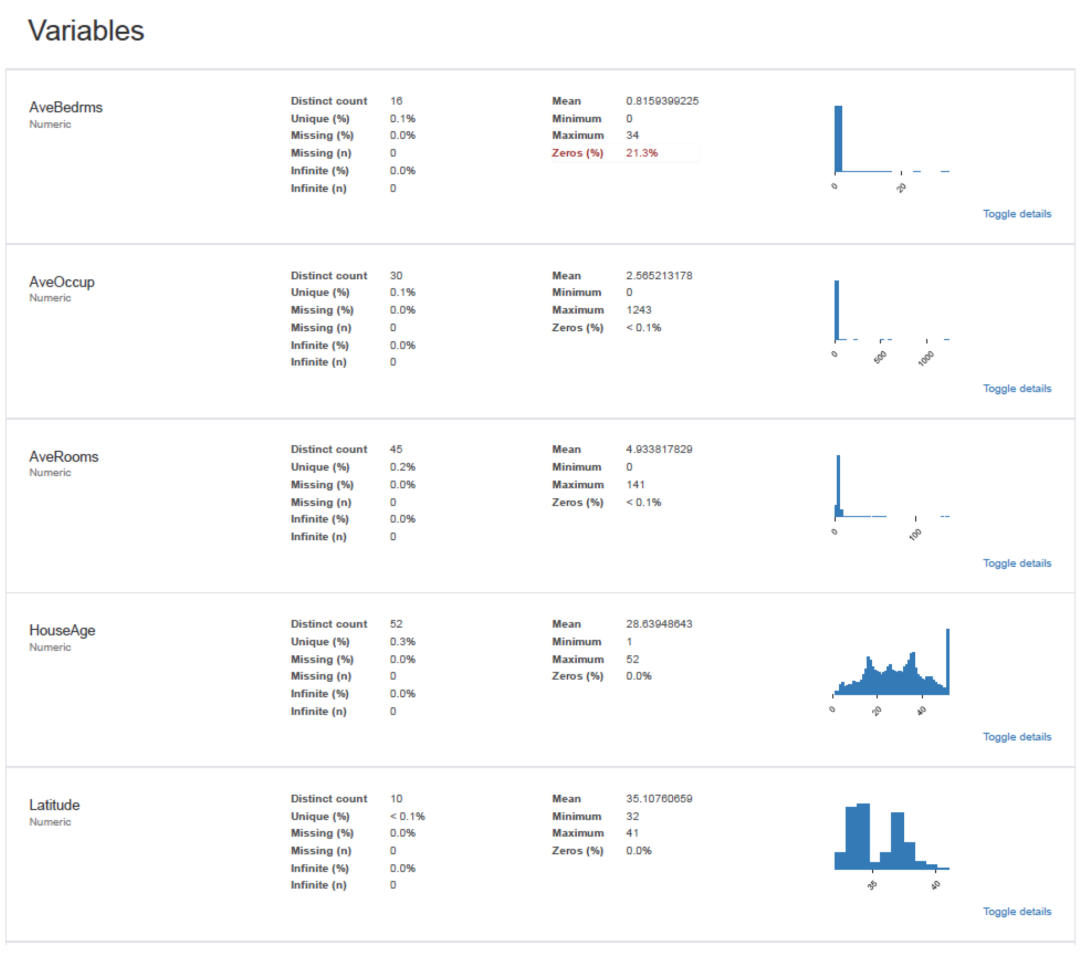
Number of variables	8
Number of observations	20640
Missing cells	0 (0.0%)
Duplicate rows	9 (< 0.1%)
Total size in memory	1.3 MiB
Average record size in memory	64.0 B

Variables types

Numeric	8
Categorical	0
Boolean	0
Date	0
URL	0
Text (Unique)	0
Rejected	0
Unsupported	0

Warnings

Dataset has 9 (< 0.1%) duplicate rows	Warning
`AveBedrms` has 4395 (21.3%) zeros	Zeros
`AveOccup` is highly skewed (γ1 = 97.51573851)	Skewed
`AveRooms` is highly skewed (γ1 = 20.21417881)	Skewed



Variables

AveBedrms
Numeric

Distinct count	16	**Mean**	0.8159399225
Unique (%)	0.1%	**Minimum**	0
Missing (%)	0.0%	**Maximum**	34
Missing (n)	0	**Zeros (%)**	21.3%
Infinite (%)	0.0%		
Infinite (n)	0		

AveOccup
Numeric

Distinct count	30	**Mean**	2.565213178
Unique (%)	0.1%	**Minimum**	0
Missing (%)	0.0%	**Maximum**	1243
Missing (n)	0	**Zeros (%)**	< 0.1%
Infinite (%)	0.0%		
Infinite (n)	0		

AveRooms
Numeric

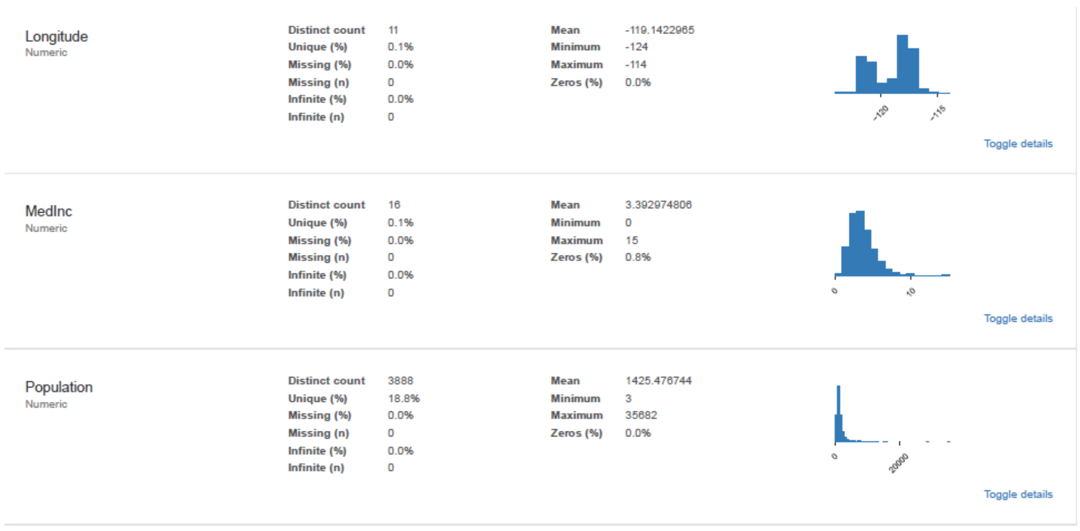
Distinct count	45	**Mean**	4.933817829
Unique (%)	0.2%	**Minimum**	0
Missing (%)	0.0%	**Maximum**	141
Missing (n)	0	**Zeros (%)**	< 0.1%
Infinite (%)	0.0%		
Infinite (n)	0		

HouseAge
Numeric

Distinct count	52	**Mean**	28.63948643
Unique (%)	0.3%	**Minimum**	1
Missing (%)	0.0%	**Maximum**	52
Missing (n)	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

Latitude
Numeric

Distinct count	10	**Mean**	35.10760659
Unique (%)	< 0.1%	**Minimum**	32
Missing (%)	0.0%	**Maximum**	41
Missing (n)	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		



Longitude
Numeric

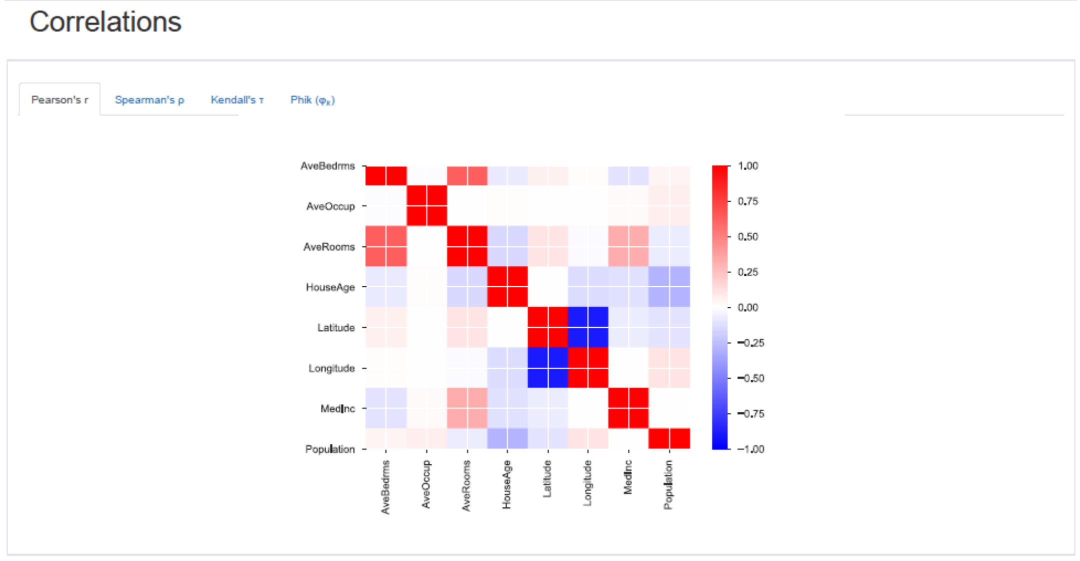
Distinct count	11	Mean	-119.1422965
Unique (%)	0.1%	Minimum	-124
Missing (%)	0.0%	Maximum	-114
Missing (n)	0	Zeros (%)	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		

MedInc
Numeric

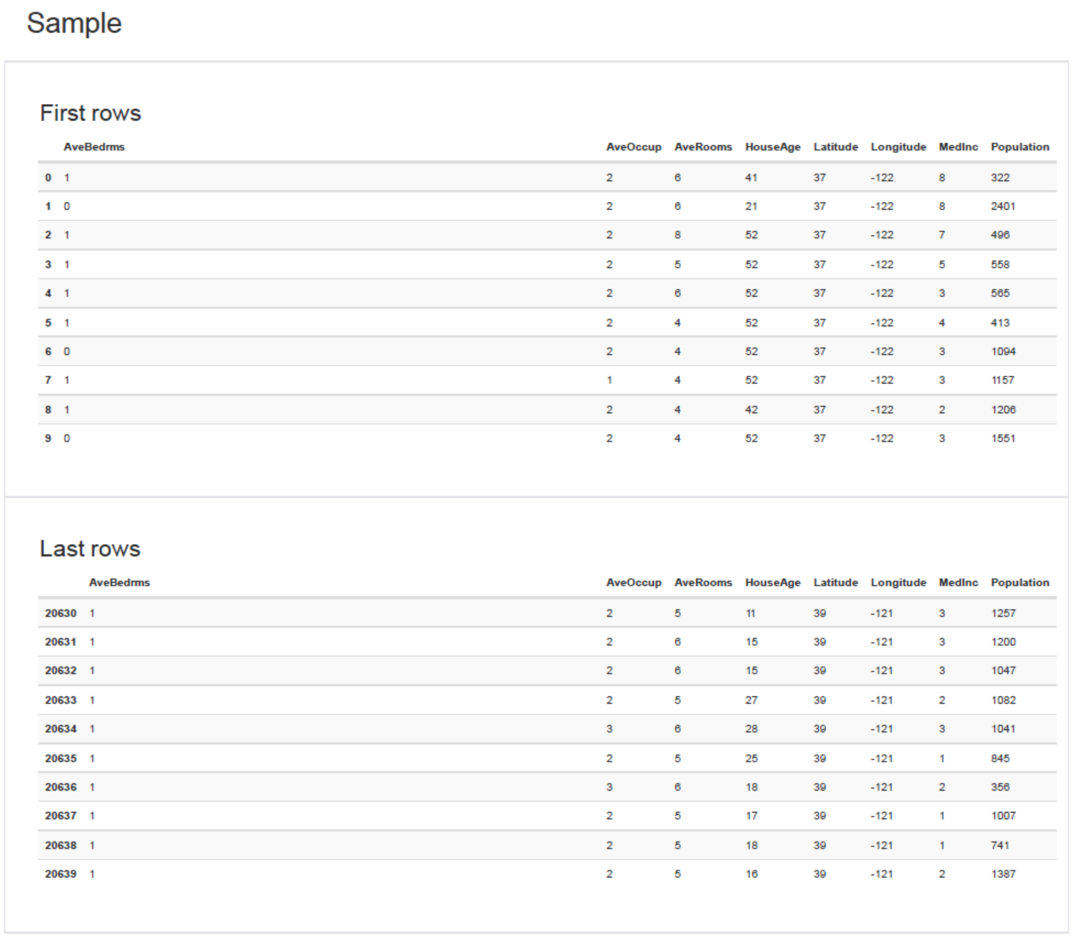
Distinct count	16	Mean	3.392974806
Unique (%)	0.1%	Minimum	0
Missing (%)	0.0%	Maximum	15
Missing (n)	0	Zeros (%)	0.8%
Infinite (%)	0.0%		
Infinite (n)	0		

Population
Numeric

Distinct count	3888	Mean	1425.476744
Unique (%)	18.8%	Minimum	3
Missing (%)	0.0%	Maximum	35682
Missing (n)	0	Zeros (%)	0.0%
Infinite (%)	0.0%		
Infinite (n)	0		



Correlations



Sample

First rows

	AveBedrms	AveOccup	AveRooms	HouseAge	Latitude	Longitude	MedInc	Population
0	1	2	6	41	37	-122	8	322
1	0	2	6	21	37	-122	8	2401
2	1	2	8	52	37	-122	7	496
3	1	2	5	52	37	-122	5	558
4	1	2	6	52	37	-122	3	565
5	1	2	4	52	37	-122	4	413
6	0	2	4	52	37	-122	3	1094
7	1	1	4	52	37	-122	3	1157
8	1	2	4	42	37	-122	2	1206
9	0	2	4	52	37	-122	3	1551

Last rows

	AveBedrms	AveOccup	AveRooms	HouseAge	Latitude	Longitude	MedInc	Population
20630	1	2	5	11	39	-121	3	1257
20631	1	2	6	15	39	-121	3	1200
20632	1	2	6	15	39	-121	3	1047
20633	1	2	5	27	39	-121	2	1082
20634	1	3	6	28	39	-121	3	1041
20635	1	2	5	25	39	-121	1	845
20636	1	3	6	18	39	-121	2	356
20637	1	2	5	17	39	-121	1	1007
20638	1	2	5	18	39	-121	1	741
20639	1	2	5	16	39	-121	2	1387

You can download all profiling measure made in the next url `https://github.com/nmvalente/diss/blob/master/california_profiling.html` and navigate better.