FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Time series vs regression, which is the best approach?

Nuno Miguel de Barbosa Ferreira

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: João Pedro Mendes Moreira

Second Supervisor: José Diogo Seca

April 04, 2021

Time series vs regression, which is the best approach?

Nuno Miguel de Barbosa Ferreira

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Approved in oral examination by the committee:

Chair: António Pedro Rodrigues Aguiar
External Examiner: Paulo Alexandre Ribeiro Cortez
Supervisor: João Pedro Mendes Moreira
Second Supervisor: José Diogo Seca

April 04, 2021

Resumo

Hoje em dia os dados assumem um papel importantíssimo para os diferentes negócios onde a previsão surge como a melhor ferramenta de interpretação e análise desses mesmos dados. Um ramo desse desenvolvimento é a análise e previsão de séries temporais que tem sido alvo de estudos nos últimos anos e pode ser aplicável a áreas como o mercado finaceiro, a meteorologia ou o consumo energético.

Para problemas de séries temporais a norma é imposta por métodos como o ARIMA ou o Exponential Smoothing, no entanto pesqiusas recentes bem como um aumento na popularidade de utilização de métodos de machine learning tornam-nos interessantes de serem aplicados a problemas deste tipo. Um dos maiores obstáculos é a estrutura típica imposta pelos problemas de séries temporais, dificultando a adaptação a métodos de machine learning. Esta dissertação propõe uma conversão automática da framework de um problema de séries temporais para a framework de um problema de regressão.

Esta conversão é feita através da análise de séries temporais e extraindo features que possam ser utilizadas pelos algoritmos de mahcine learning. Os métodos estudados são o K Nearest Neighbors, Linear Regression, Neural Networks, Random Forests e Support Vector Machines. No fim uma comparação destes métodos com os métodos clássicos ARIMA e Exponential Smoothing é realizada em três datasets diferentes. Para a obtenção de melhores resultados é também realizada uma optimização de hyperparâmetros e uma leve análise das features.

A conversão automática proposta obteve resultados competitivos e nalguns casos até superiores aos métodos clássicos, comprovando a existência de vantagens em utlilizar frameworks de regressão em problemas de séries temporais. Ao tentar pavimentar o camimho para a utlização de métodos de regressão este trabalho incentiva uma reflexão sobre as duas abordagens. Para essa reflexão o trabalho realizado na presente dissertação realça a flexibilidade dos modelos de machine learning e a fácil interpretabilidade das features criadas como argumentos a favor do uso deste tipo de abordagem na previsão de problemas de séries temporais.

Abstract

Nowadays data assumes an extremely important role for businesses where forecasting rises as the best tool to better interpret and analyze such data. One branch of this development is the time series analysis and forecast which has been a major research focus since years ago and can find application in fields such as the stock market, weather or electricity demand.

For time series problems the standard is set by classical methods like the ARIMA or Exponential Smoothing, however recent developments and a gain in popularity of machine learning methods make them tentative to use in such problems. One of the main obstacles is the imposed time series problems structures making an adaption to machine learning approaches difficult. This dissertation aims to build an automatic conversion of time series problem framework into a regression one.

This conversion is done by analyzing the time series and performing feature extraction for posterior use of machine learning methods. The methods selected to be studied are the K Nearest Neighbors, Linear Regression, Neural Networks, Random Forest and Support Vector Machines. In the end a comparison with the classic methods ARIMA and Exponential Smoothing is made across three different datasets. For better results an optimization of hyperparameters is performed and a slight feature analysis is given.

The proposed framework conversion achieved competitive results and even outperforms the classic methods in some instances proving that there are advantages in solving time series problems with machine learning methods. Furthermore, by paving the way for the regression framework a discussion is incentivized since the flexibility of such methods and the easy interpretation of the features here created come as big advantages for the forecaster to use when solving this type of problems.

Acknowledgments

The first people I'd like to thank are my parents, the best ones I could ever ask for and who have supported me all this time. I strive to make you proud.

I'd also like to thank my supervisor Professor João Moreira who took me under his wing and expertly taught me and guided me through the forecasting path even thought I had next to zero practice in this subject, but much curiosity. On a similar note a thank you to the co-supervisor Diogo Seca who also guided me in this project as well and answered all my pythonic questions. I think the three of us make a good team and you made Monday afternoons fun! Having the proper guidance for this type of work is very important and I couldn't be more satisfied with the help you've given me.

Lastly, a thank you note to the close friends who've supported me and showed their belief in my success and to my brother who has absolutely no idea what I talk about when I speak to him about engineering, but still gives me space to vent and supportive words.

Nuno Miguel de Barbosa Ferreira

"What we know is a drop, what we don't know is an ocean."

Isaac Newton

Contents

List of Figures

List of Tables

Abbreviations

ADF	Augmented Dickey-Fuller
AR	AutoRegression
AIMRA	AutoRegressive Integrated Moving Average
EMA	Exponential Moving Average
ETS	Exponential Smoothing
GW	Growing window
KNN	K Nearest Neighbors
LR	Linear Regression
MA	Moving Average
MAPE	Mean Absolute Percentage Error
MSE	Mean Squared Error
NN	Neural Networks
RMSE	Root Mean Squared Error
RF	Random Forest
sMAPE	Symmetric Absolute Percentage Error
SMA	Simple Moving Average
SVM	Support Vector Machine
SW	Sliding Window

Chapter 1

Introduction

1.1 Context

In today's world technology and data are evolving at a very fast pace leaving companies with the desire to master such tools for their own growth and benefit. Synchronously, machine learning and forecasting accompany such developments and try to recognize some importance patterns in the data, either to provide an understanding or to try and predict the future so that the companies' desires can be fulfilled.

Forecasting is a tool that aims to provide aid in decision making processes and efficient planning, whether that means learning from past mistakes, decreasing costs, better resource management amongst many other uses. When the data that's being analyzed is ordered over time then it's considered a time series and its forecasting will try to estimate how that sequence of observations will behave in the future. As such, time series forecasting has an added importance when applied in fields like stock market exchange, weather, electricity demand or cost and usage of products such as fuels and electricity, where a good prediction improve a business.

Time series forecasting is not a new problem, but work and research continue to be made in order to better tune the machine learning models and try to improve the accuracy of the predictions, whether that means using a machine learning model or a classical time series one.

1.2 Motivation

Time series forecasting has a long history where statistically based methods such as ARIMA or Exponential Time Smoothing constitute the classical time series approach. However, the development and popularity of computer intelligence methods make a case to be applied in time series forecasting but, despite some good results, it still lacks the consistency and maturity to outperform the classical statistical methods.

These more modern machine learning algorithms gain in diversity, parameter tuning and complexity making them very tempting to try as opposed to the classical time series methods that are considered more rigid. But the restrictions that time series structure impose seem to be one of the

main problems when applying the modern machine learning approaches, so the question of could there be a way to convert a time series problem into a regression one begs to be answered.

1.3 Goals

The main goal of this dissertation is to provide an automatic conversion of a time series problem into a regression framework that would enable the use of state-of-the-art machine learning regression methods. This conversion aims to extract features from a time series problem into a regression one trying to escape the restrictions of the typical time series structure, more specifically the seasonality property.

The second goal of this dissertation is to test some machine learning algorithms in this new framework and juxtapose them with the classical time series methods to try and assess the feasibility of this approach.

1.4 Outline

The remaining dissertation's outline is organized as follows. Chapter 2 presents a literature review over the main topics related to the development of this thesis. Chapter 3 describes the implementation of the framework conversion. Chapter 4 presents the results and discusses them. Chapter 5 summarizes and concludes the whole work and suggests some future related work.

Chapter 2

Theoretical Background

This chapter aims to provide some background on the topics covered by this dissertation including the theory behind the subjects as well as the relevant state of the art work. First an overview of time series and the classical statistical methods is made and later in the chapter a similar overivew is done for the machine learning methods.

2.1 Time Series

A time series is defined as a series of observations x_t recorded at time $t = 0, 1, 2 ...$ As mentioned in chapter 1 time series can cover a wide range of fields and its forecasting is of great importance:

> "The need for forecasting is increasing as management attempts to decrease its dependence on chance and becomes more scientific in dealing with its environment."[1]

As for the structure of time series two important properties emerge:

- Trend - "A trend exists when there is a long-term increase or decrease in the data. It does not have to be linear." [2]
- Seasonality - "A seasonal pattern occurs when a time series is affected by seasonal factors such as the time of the year or the day of the week. Seasonality is always of a fixed and known frequency." [2]

These two properties are easily demonstrated on figure 2.1.

2.1.1 ARIMA

The abbreviation ARIMA stands for AutoRegressive Integrated Moving Average and is one of the most commonly used forecasting methods for time series and tries to model it by the autocorrelation due to the dependence of y_t on former values. A good introduction to this method can be found in book [1].

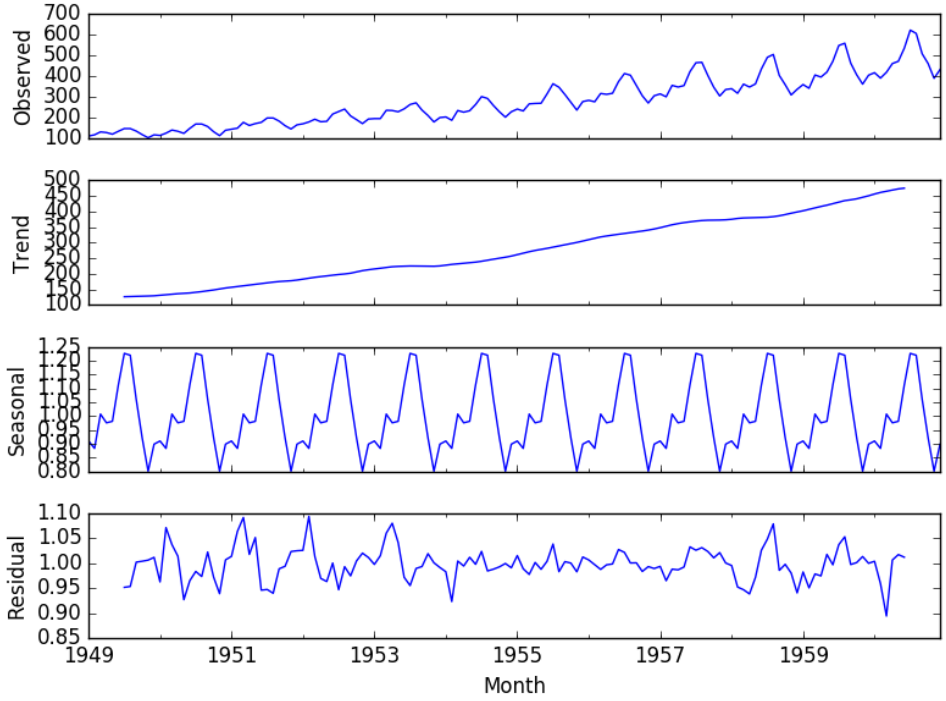

Figure 2.1: Example of trend and seasonal components in a time series decomposition

Source: https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/

The AutoRegression (AR) part of ARIMA shows that the time series is regressed on its own past data, so, for example, an autoregressive process of order p, named AR(p), models the current mean-centered target value and can be given by the following equation:

$$Y_t = \varphi Y_{t-1} + \varphi Y_{t-2} + ... + \varphi Y_{t-p} + w_t \tag{2.1}$$

where w_t is denotes a white noise process. The Moving Average (MA) part of the ARIMA model instead showcases the dependency between an observation and a residual error from a moving average model applied to lagged observations. It can generally be expressed through the form MA(q) as a moving average of order q:

$$Y_t = \theta w_{t-1} + \theta w_{t-2} + ... + \theta w_{t-p} + w_{t-q} \tag{2.2}$$

A good notion to also have is the one of stationarity, where data fluctuates "around a constant mean, independent of time, and the variance of the fluctuation remains essentially constant over time." [1]. For the ARIMA method, this relates to the Integration (I) where there's a use of differencing of observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary. Tests for this definition can me made and will be mentioned in chapter 3. Generally an ARIMA process can be described as ARIMA(p,d,q)

where each of the parameters describes the order of each part of the process previously described. The ARIMA method can be easily adapted to seasonality through the SARIMA$(p,d,q)(P,D,Q)_s$ where the second set of the parameters model the seasonality and s represents the number of periods per season. The ARIMA family can provide a whole array of options such as SARIMAX for example, however since the scope of this dissertation is to compare other frameworks against the classical time series methods, a classical SARIMA implementation was chosen since it has proven its consistency in time series forecasting as can be seen on [3].

2.1.2 Exponential Smoothing

Exponential Time Smoothing (ETS) methods separate themselves from the ARIMA family since they shift their focus from the correlations to the modeling of both trend and seasonality. As with the ARIMA a great introduction and explanation of exponential smoothing methods can be found on the book [1].

On a foundational level they note that a future value can be modeled by the weighted aggregation of past values with an exponential decaying (smoothing) attributed to older observations' weights:

$$\hat{Y}_{t+1|t} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \alpha(1-\alpha)^3 y_{t-3} + ... \tag{2.3}$$

with 0 <= α <= 1.

This original stance however lacks the complexity needed to deal with trend and seasonality. As a result the Holt Winter's method was created that was able to deal with such properties. Once again book [1] is able to provide good insights on this method. The Holt-Winter's approach models the trend and seasonality and merges everything together in two different variations: additive and multiplicative, two strategies to deal with the seasonal component of the time series depending if the seasonal component is constant over time or if its magnitude is time dependent. The equations to perform a one method forecast in the additive form are as follows:

$$\hat{y}_{t+1|t} = l_t + hb_t + s_{t-m+1} \tag{2.4}$$
$$l_t = \alpha(y_t + s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{2.5}$$
$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{2.6}$$
$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \tag{2.7}$$

The equations for the multiplicative variation are as follows:

$$\hat{y}_{t+1|t} = (l_t + hb_t)s_{t-m+1} \tag{2.8}$$
$$l_t = \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{2.9}$$
$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{2.10}$$
$$s_t = \gamma\frac{y_t}{(t-1-b_{t-1})} + (1-\gamma)s_{t-m} \tag{2.11}$$

Similarly to the ARIMA, a more classical approach was chosen with the Holt-Winters method for later results comparison since they also show consistent performance on time series problems [3] and are still a topic of research like on work [4] where this approach is taken with some tweaking to deal with anomalies.

2.2 Machine Learning

To try and comprehend the environment that surrounds the ever growing fast expansion of data, machine learning grew to popularity by trying to discover hidden patterns in such data. To better understand machine learning [5] tries to provide a definition of this discipline:

> "[...] we say that a machine learns with respect to a particular task T, performance metric P, and type of experience E if the system reliably improves its performance P at task T, following experience E. Depending on how we specify T, P, and E, the learning task might also be called by names such as data mining, autonomous discovery, database updating, programming by example, etc."

There are some important characteristics of machine learning worth noticing:

- Supervised: where there is a clear definition of input and output data for the machine to use;
  - Regression: regression problems try to predict a continuous value;
  - Classification: classification problems try to predict a discrete class label;
- Unsupervised - where the machine is forced to learn patterns from unlabelled data.

This dissertation falls into the supervised category and constitutes a regression problem.

2.2.1 Naïve method

The Naïve method is a very simple one where the forecast is estimated to be equal to the last observation, as the following equation explains:

$$Y_t = Y_{t-1} \tag{2.12}$$

Although more versions of this method exist (like the seasonal naive) the simplest implementation as described on equation 2.12 will be used. This approach is used since it provides a nice baseline comparison against the other methods and allows for a preview on the forecastability data itself as [3] noted.

2.2.2 Linear Regression

Linear Regression (LR) is one of the oldest methods available and its simple nature allows for an easy understanding and application. At its core, Linear Regression establishes the relationship between two variables using a straight line as seen on an example figure 2.2.

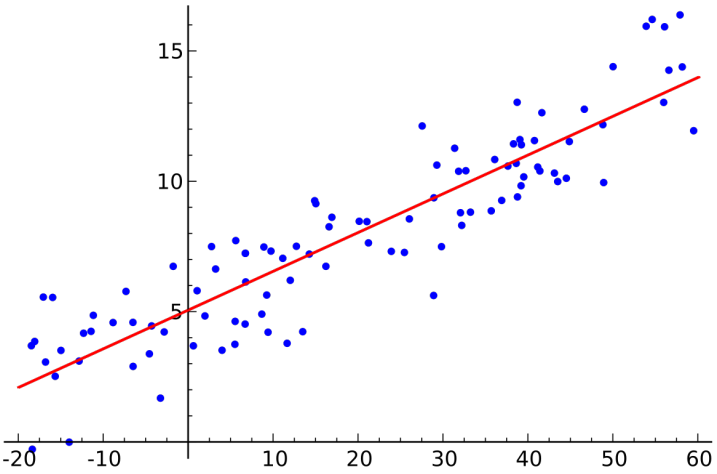

Figure 2.2: Simple Linear regression example

Source: https://en.wikipedia.org/wiki/Linear_regression

The drawn line is the one that comes closest to the data by finding the slope and intercept that define the line and minimize the regression errors. Linear regression can handle one input variable or multiple variables, leading to two types of LR: simple and multiple linear regression. Since feature extraction is going to be performed this dissertation falls into the latter category. The basic multiple linear regression model is given by the following equation:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + ... + \beta_p X_{ip} + \varepsilon_i \tag{2.13}$$

Where:

Y_i is the target value at the i^{th} observation

β_0 is the intercept

$\beta_1, \beta_2, ...$ are the regression coeficcients for the explanatory variables X

$X_1, X_2, ...$ are the explanatory variables

Linear regression has been used for time series forecasting before due to its simplicity over other algorithms, like when [6] proposed an enhanced variation of linear regression that tries to optimize the quadratic mean of the loss function. However the linear regression approaches don't usually extract features from the time series and the focus has been shifting towards other machine learning methods.

2.2.3 K Nearest Neighbors

A good introduction to this method can be found on the book [7]. K nearest neighbors (KNN) is also one of the most simple approaches, since the forecast of the new observation is made using the k-closest samples from the training set, like on figure 2.3. The distance to find the closest records is often calculated using the euclidean distance. As the name hints k is a very important parameter in this method and deserves investigation. KNN is also referred as a lazy algorithm since it memorizes the training set instead of learning a function that models the data.

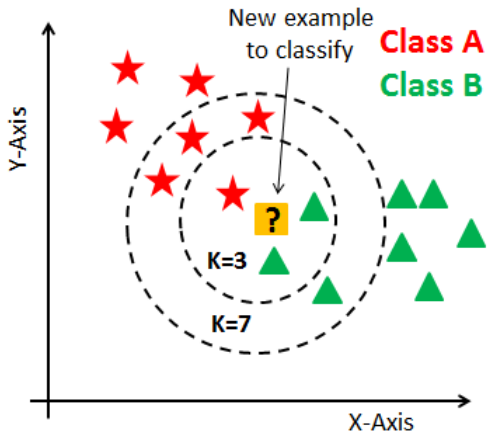

Figure 2.3: Simple KNN illustration in a classification problem

Source: https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

Despite its simplicity K nearest neighbors hasn't seen much investigation in time series forecast. On [8] an overview is made on some forecasting techniques used in the NN3 competition data, trying to analyze the performance of artificial intelligence techniques against classical models after the modest result of the only artificial neural network contender in the M3 competition [9]. On [8] a good result is achieved using K Nearest neighbors but there isn't much evidence on how the method was applied. On [10] a KNN approach is used to predict electrity load using as input a variable consisting of the load of the previous 24 hours. In [11] an approach in proximity to the goal of this dissertation was used, trying to set up an automatic method for k nearest neighbors forecasting for time series. In order to do so an outlier treatment is applied as described in [12] which will be examined later. To deal with trend 3 strategies are tested: no strategy at all, differencing and STL decomposition [13]. STL decomposition, as the name indicates, decomposes the time series in its components, mainly trend and seasonality and an example can be found on figure 2.1. To deal with seasonality also 3 strategies are tested: no strategy at all, differencing or applying the strategy described in [12] which also will appear later in this chapter. An extra test was also performed trying to use Box-Cox transformations, a method that is also used by ARIMA methods

when trying to stabilize time series with variance in the seasonality. Box-cox tranformation is given by the following equation:

$$y_t^{\lambda} = \begin{cases} \log(y_t) & \text{if } \lambda = 0 \\ \frac{(y_t^{\lambda}-1)}{\lambda} & \text{otherwise} \end{cases} \tag{2.14}$$

Where y_t is the time series to be transformed and λ the exponent value that indicates the power to which all data should be raised. An introduction to this transformation can be found on [2] and is also included in the experiments of [3]. After all these tests the conclusion was that no transformation at all was the best approach since the results didn't vary so much and the extra computational effort wasn't justified. As far input variables, the main takeaway is the use of lagged observations of the time series.

Since the K Nearest Neighbor is a simple and classical machine learning method it was decided to be included in this dissertation, since the input variables will be different than the ones used in [11] and the work of [3] doesn't include this method.

2.2.4 Neural Networks

Neural networks has gained an increased interest due to its good performance on an array of different areas such as time series forecasting, pattern recognition and classification namely due to their ability on discover underlying structures in data and a very good modeling of non-linear relationships, like the overview on [14] shows.

Neural Networks (NN) got their name due to the comparison made to the nervous system of living beings where the neurons communicate with other neurons. The network consists of different layers of these neurons, or nodes, that after performing some sort of calculations pass the information to others nodes in another layer. The perceptron is one of the most frequently used neurons in neural networks and a simple example of what the perceptron does and the calculations involved can be seen on figure 2.4. Essentially the perceptron receives the inputs, multiplies them by the corresponding weight and sums it all up including an offset (bias). If the result is above a certain threshold the neuron is fired through an activation function.

There are three different activation functions worth noticing:

- Identity function - given by the straightforward linear function $y = x$

- Sigmoid - the sigmoid activation function takes the form:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.15}$$

  The sigmoid returns a number between 0 and 1, is monotonic but its derivative is not and is not zero-centered.

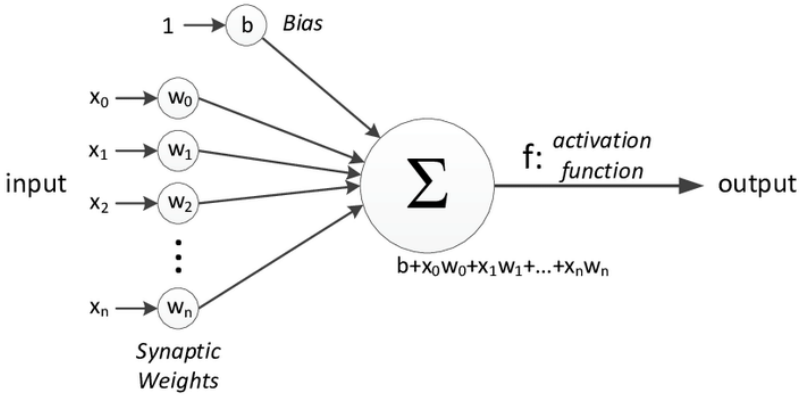

Figure 2.4: The Perceptron neuron

Source: [15]

- Tanh - The hyperbolic tangent activation, or tahn, assumes the form:

$$tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})} \tag{2.16}$$

  The Tanh function returns a number between -1 and 1, is also monotonic but its derivative is not and it's zero centered.

- Relu - The rectified linear unit (Relu) is described by the equation:

$$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases} \tag{2.17}$$

  This activation function returns numbers between 0 and $+\infty$, its function and derivative are both monotonic and transforms any negative value into zero immediately.

For this subject there was no need for an in depth study and hyperparameter optimization of NN was performed trying to find the best function as will be described in chapter 3.

Nowadays the use of many perceptron neurons in different layers is what comprises a neural network. A common example is the Multilayer Perceptron (MLP) and can be seen on figure 2.5. Training this neural network is done in three different steps: 1) receiving the inputs in the input layer and passing them after computations through the hidden layers all the way to the output (feed-forward) which, in the scope of this dissertation, will consist in a time series prediction; 2) comparing the previously made prediction with the target and reflecting it on an error; 3) back-propagating the error from the output layer to the input layer and updating the weighs of every neutron accordingly, so that the error can be minimized.

There is some research done with Neural Networks and time series, reinforcing their popularity as [16] demonstrates that this method is the predominant technique in financial time series forecasting. On the NN3 competition [8] a neural network solution placed second. On [17] a

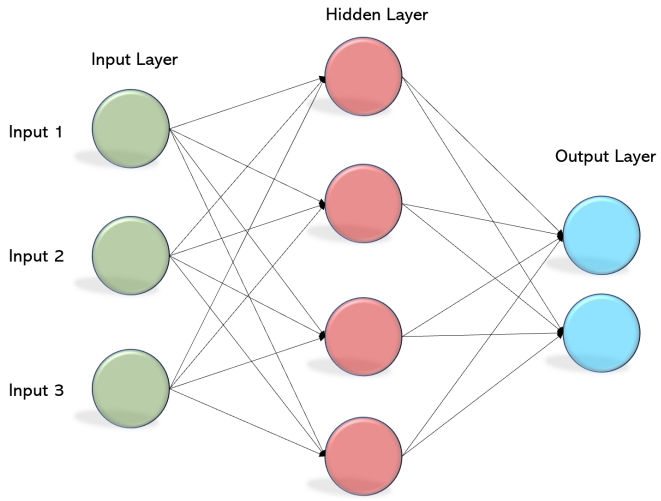

Figure 2.5: An example of a Multilayer Perceptron (MLP) neural network

Source: https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

MLP neural network obtains good results in forecasting the temperature of a building and is compared with standard regression models. A time series with both seasonal and trend comnponents is studied for a neural network model in [18] showing some good results. Interestingly, no feature extraction was performed but the results of the NN model were good when detrending and deseasonalizimg were applied. Coming closer to the goal of this dissertation an automatic model of Neural Networks for time series was proposed in [12]. In it some interesting ideas must be retained by the implementation. First is the outlier treatment also used in the previously mentioned [11] where an outlier denotes a point whose absolute value is four times greater than the absolute medians of the three consecutive points before and after, respectively, of the point. When an observation is deemed to be an outlier its value is simply replaced with the average value of the two points that are immediately before and after the outlier. For trend, a detrending technique was applied subtracting the mean value of the seasonal segment and for seasonality also a deseasonalizimg technique was applied by subtracting the mean seasonal average. In regards to features no feature extraction was provided and essentially a lagged input ends up selected as the input of the model. The work of [3], although not towards an automatic approach, shows some interesting points where a simulation study was conducted testing different algorithms and introducing some interesting input variables. Those variables were a numeric value representing the point inside a trend, a numeric value to represent the point inside a season, seasonal dummies and lagged observations. In the end, seasonal dummies and lagged variables showed best results and were chosen to the final experiments. It's worth noticing this work is of interest because it touches neural networks and random forests even thought the work is more focused towards bagging and boosting. Overall NN didn't perform so good in this work but since they are very popular and this

dissertation proposes some new features as inputs they were included in this work.

2.2.5 Random Forests

Random Forests (RF) are a large number of individual decision trees that operate as an ensemble. A very simple decision tree can be found on figure 2.6 and an ensemble of decision trees making a random forest can be seen on figure 2.7. Essentially they operate by building several trees during training and outputting the mean prediction of the trees. These trees run in paralel and without interacting between them. The process works like this: 1) pick k random observations from the training set; 2) construct a decision tree for those k observations; 3) repeat the previous steps for the number of trees existing in the ensemble; 4) for a new observation have every tree predict the value for it and then assign the mean average of all the predictions to that new observation.

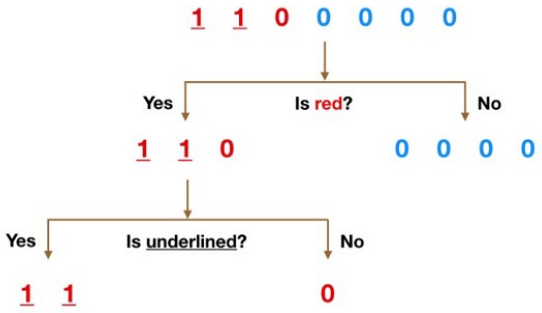

Figure 2.6: Simple decision tree for a classification problem

Source: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

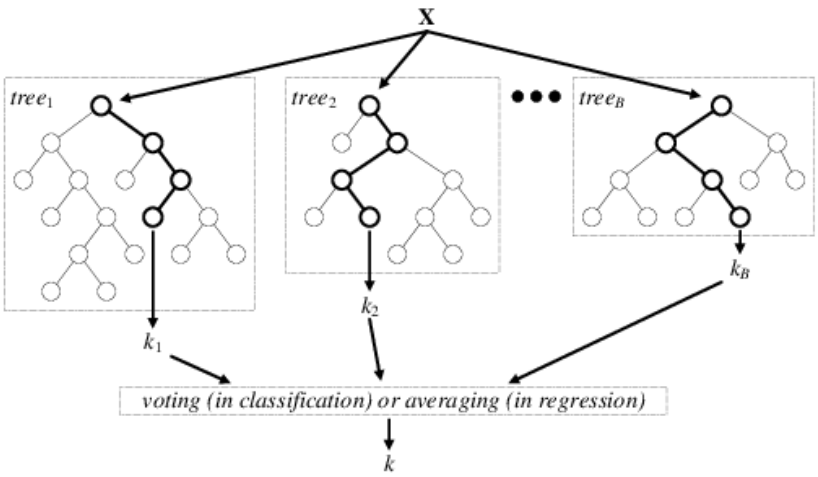

Figure 2.7: Random Forest example

Source: https://github.com/hvantil/RandomForestTutorial/blob/master/RandomForestTutorial.ipynb

Random Forests are popular since it's easy in its implementation and can be applied to complex data structures, making it applicable to different areas. For time series it has seen application on medicine for example [19] where a random forest plus an ARIMA approach is used to predict avian influenza or environmental science [20]. On [21] an interesting study is made in variable selection for a random forest model predicting temperatures and, although not automatic, it concludes that the use of lagged variables is beneficial to this approach when forecasting a time series. Going back to the work of [3] random forests denoted incapability in forecasting series that exhibit a trend, hence some detrending technique prior to the forecast is advised. Random Forests were selected for this dissertation since they're of easy implementation, versatile and can be a nice introduction to the tree ensemble family of methods.

2.2.6 Support Vector Machine

The idea behind the Support Vector Machine (SVM) [22] is to try and find a sprat line (hyperspace) between data points of different target labels while aiming at the maximization of the predictor space. A good example can be seen on figure 2.8. Assuming a linear interpretation then the hyperplane can be given by the equation $y = wx + b$ with the constraints:

- y_i - wx_i - $b \leq \varepsilon$
- $w_i + b$ - $y_i \leq \varepsilon$

which can be easily be interpreted as the support vectors for the hyperplane. The objective then is to minimize the equation:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{2.18}$$

with the constraints:

$$y_i - wx_i - b \leq \varepsilon + \xi_I \tag{2.19}$$

$$w_i + b - y_i \leq \varepsilon + \xi_i^* \tag{2.20}$$

$$\xi_i, \xi_i^* \geq 0 \tag{2.21}$$

The constant C > 0 determines the trade-off between the flatness of the hyperplane function and the amount up to which deviations larger than ε are tolerated. Non-linear adaptations for the SVM can be made through kernels like the polynomial, the radial basis function or the sigmoid.

Some work can be found on time series and SVM applications like the works of [23] or [24] where the SVM model using lagged target variables outperforms an ARIMA model. Support Vector Machine then finds a place of study in this dissertation as a representative of the kernel family of models.

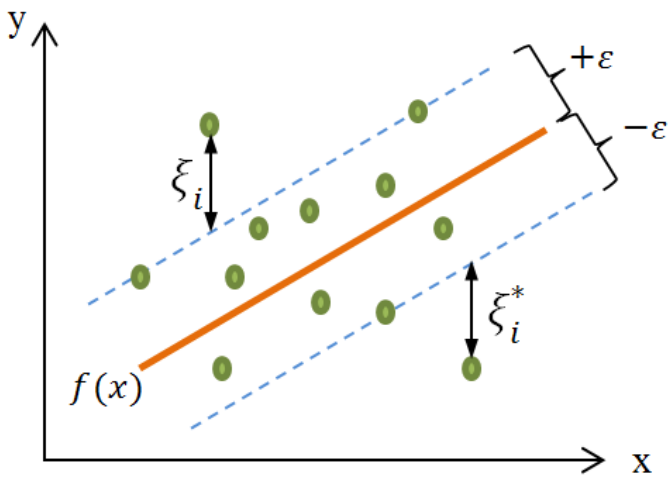

Figure 2.8: Support Vector Machine example

Source: https://www.researchgate.net/figure/Example-of-linear-support-vector-regression$_fig1_3$23588842

Chapter 3

Methodology

This chapter explains the suggested method for the automatic conversion of a time series framework to a regression one. As stated before and accordingly to the goal of this work, feature extracting is going to be done in order to provide the methods with the input variables and that will be explained in this chapter alongside model parameterization, error measures and a presentation of the datasets. This dissertation was developed with the Python 3 language using the PyCharm IDE.

3.1 Datasets

This methodology of this dissertation was tested against 3 different datasets trying to model seasonality but also seeing how it would fare with a trend as well. Before feeding the datasets to the process some aspects have to be highlighted:

- size - the dataset must be long and have, at least, 7 well defined periods of seasonily. The reasons will become clearer later in the chapter;
- organization - the univariate time series to study must follow the strucutre of 3.1 where the first column is the time index and the next column is the target value;
- seasonality- the main target of this dissertation is to model problems with seasonality, making this property a requirement.

3.1.1 Temperature Dataset

The temperature dataset consists of 240 monthly data points of temperature (Deg. F) in Nottingham Castle from 1920 to 1939. This dataset doesn't have a trend and exhibits a seasonality.A sample plot is found on figure 3.1

Month	Temperature
1920-01-01	40.6
1920-02-01	40.8
1920-03-01	44.4
1920-04-01	46.7
1920-05-01	54.1

Table 3.1: Example structure of the desired data

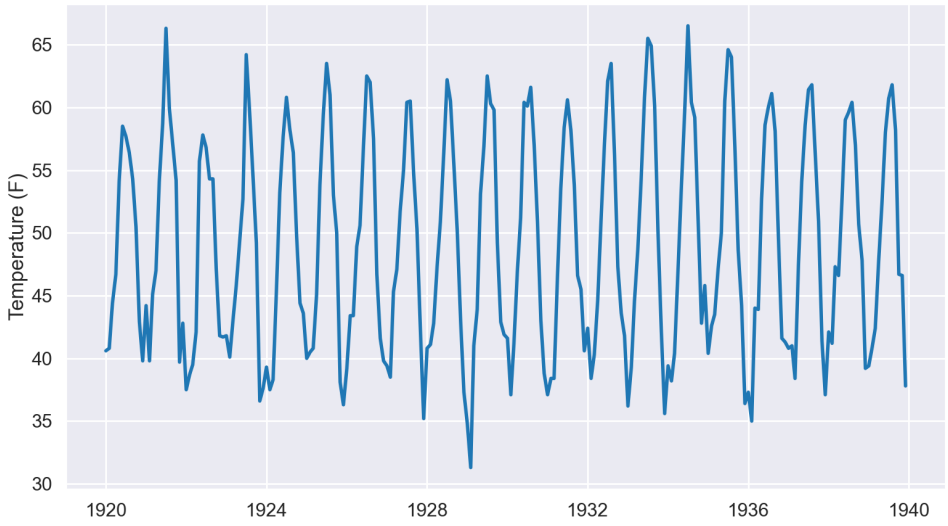

Figure 3.1: The temperature dataset

3.1.2 Champagne Sales Dataset

The champagne sales dataset consists of 105 monthly observations of the sales of the Perrin Freres champagne between 1964 an 1972. It doesn't have a trend but has a well defined seasonality, although not as regular as the one in the temperatures dataset. A sample plot can be seen on figure 3.2

3.1.3 Airline Dataset

The airline dataset comprises 144 monthly data points regarding the number of passengers in flights between 1948 and 1960. It exhibits both trend and seasonality. A sample plot can be found on figure 3.3

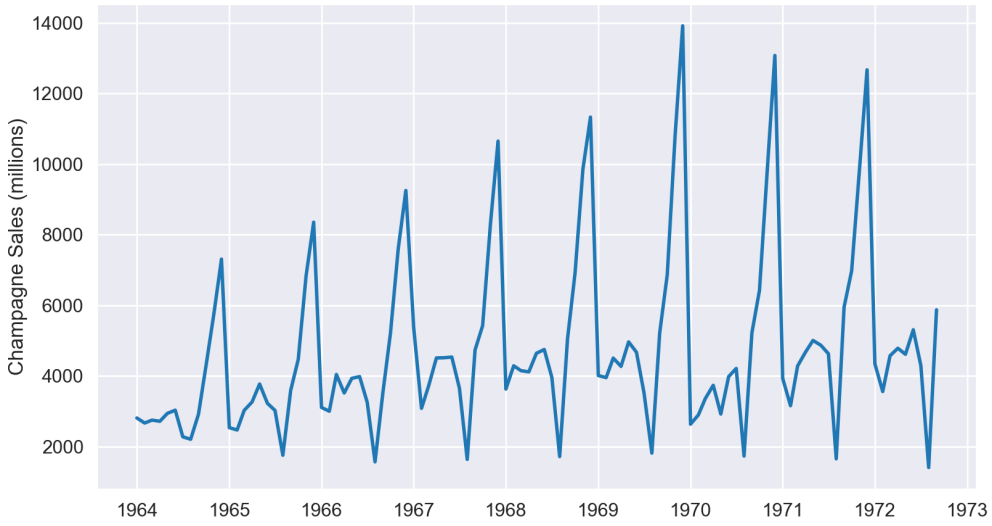

Figure 3.2: The Champagne Sales dataset

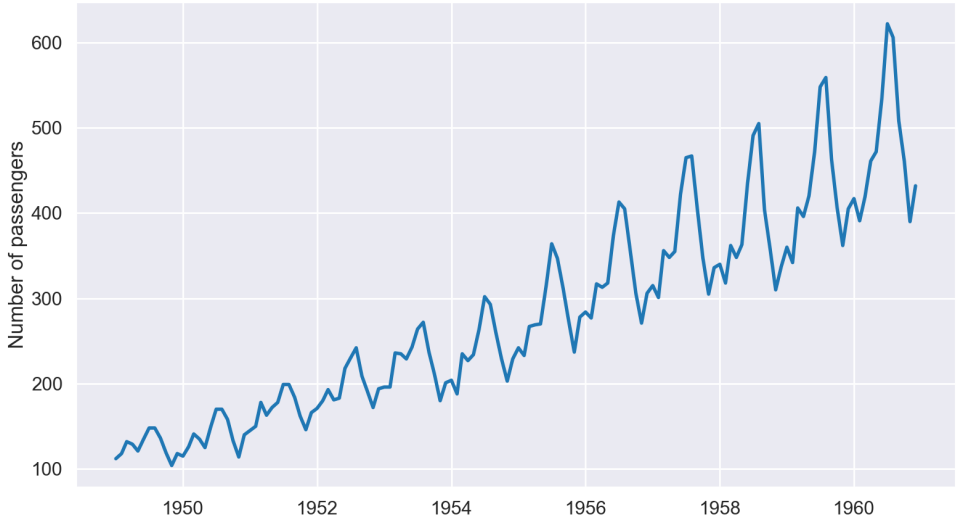

Figure 3.3: The airline dataset

3.2 Features

This section will provide an overview on the features created. First the recognition of seasonality is performed and then an additional set of features is created after that.

3.2.1 Seasonality Detection

To deal with seasonality the proposed method tries to identify the length m of the seasonal period and then assigns a numeric value (1, 2, ... m) to identify in which part of the seasonal cycle the observation is in. For detecting the seasonality we first take a look at the frequency of the data and then assign a value m that makes sense. To assess the frequency of the data a simple difference between two dates of consecutive observations is performed. In the temperature dataset for example this actions identifies the difference as being 31 days which means that the proposed framework is dealing with monthly data, resulting in $m = 12$. Next we compare that value against the critical values of the autocorrelation function plot (ACF) of the dataset. If the m, $2 \times m$ and $3 \times m$ lagged values in the autocorrelation plot are bigger than the critical value then we know we are witnessing a time series with seasonality. A more clear observation of this analysis is clearer on figure 3.4.

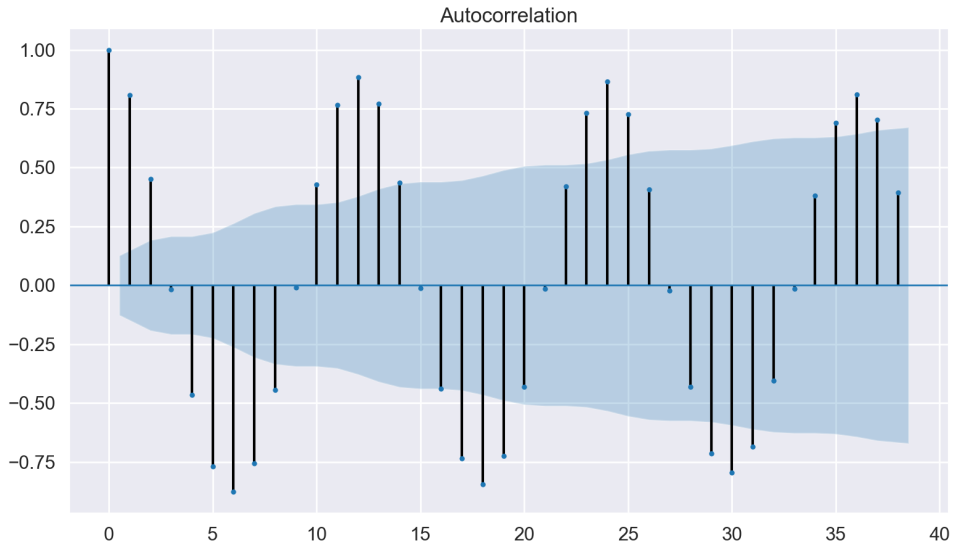

Figure 3.4: The autocorrelation plot (ACF) for the temprature dataset

To validate the previous steps a statistical test is conducted on the differenced time series: the Augmented Dickey-Fuller (ADF) test. First we apply differencing to the time series by doing: differencing = time series(t) - time series($t - m$). The ADF then tests the null hypotheses that a unit root is present in the time series. If the null hypthessis fails to be rejected, then a unit root is present and the series is not stationary. Otherwise, rejecting the null hypothesis means no unit root is present and the time series is stationary. This result is interpreted through the p value of the ADF test where:

- $p > 0.05$ - null hypothesis **not** rejected, the time series is not stationary;
- $p <= 0.05$ - null hypothesis rejected, time series is stationary.

Month	Temp	SI	Order	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
1920-01-01	40.6	1	1	1	0	0	0	0	0	0	0	0	0	0	0
1920-02-01	40.8	2	2	0	1	0	0	0	0	0	0	0	0	0	0
1920-03-01	44.4	3	3	0	0	1	0	0	0	0	0	0	0	0	0
1920-04-01	46.7	4	4	0	0	0	1	0	0	0	0	0	0	0	0
1920-05-01	54.1	5	5	0	0	0	0	1	0	0	0	0	0	0	0
1920-06-01	58.5	6	6	0	0	0	0	0	1	0	0	0	0	0	0
1920-07-01	57.7	7	7	0	0	0	0	0	0	1	0	0	0	0	0
1920-08-01	56.4	8	8	0	0	0	0	0	0	0	1	0	0	0	0
1920-09-01	54.3	9	9	0	0	0	0	0	0	0	0	1	0	0	0
1920-10-01	50.5	10	10	0	0	0	0	0	0	0	0	0	1	0	0
1920-11-01	42.9	11	11	0	0	0	0	0	0	0	0	0	0	1	0
1920-12-01	39.8	12	12	0	0	0	0	0	0	0	0	0	0	0	1
1921-01-01	44.2	1	13	1	0	0	0	0	0	0	0	0	0	0	0

Table 3.2: Example structure of the temperature dataset after the automated seasinality feature extraction

The ADF test is given by equation 3.1:

$$\delta y_t = \alpha + \beta t + \gamma y_{t-1} + \sigma \delta y_{t-1} + ... + \sigma_{p-1} \delta y_{t-p+1} + \varepsilon_t \tag{3.1}$$

In short, if the differenced time series using the value m passes the ADF test then the process that created the value m is validated and the "Seasonal Indices" feature can be created. The seasonal indices are a numeric value (1, 2, ... m) assigned to each observation to identify which part of the seasonal cycle they're in. After determining the seasonal indices another similar features were created: seasonal encoding (One Hot Encoding) features (S_1, S_2, ..., S_m) consisting of a bit value that assumes the value 1 if it corresponds to the respective seasonal period. This feature was included since different algorithms are being tested meaning one method might find one approach more suited than the other. It's also worth noticing that some feature importance analysis exists so in the end some light can be shed regarding to the efficiency of including such features. Also at this stage an order number was introduced as a feature, consisting of a simple numeric value starting from 1 to n where n is the length of the data. Data should then look like the example given in 3.2

3.2.2 Additional Features

After the previous actions are completed, several features were created to try and describe how the target variable has been behaving and, especially, how it has been changing in specific cycles of seasonality taking full advantage of the features previously created.

The Simple Moving Average (SMA) is the unweighted mean of the previous M data points, where the M parameter results in a sliding window more or less smoother depending on the value. The simple moving average is given by the equation 3.2:

$$SMA_t = \frac{x_1 + x_{t-1} + x_{t-2} + ...x_{M-(t-1)}}{M} \tag{3.2}$$

The Exponential Moving Average (EMA) assigns wights to each observation that will decrease progressively over time, meaning the exponential moving average gives greater weight to recent data points. The exponential moving average can be given by the equation 3.3:

$$EMA_t \begin{cases} 0 & \text{for } t = 0 \\ \alpha x_t + (1-\alpha) EMA_{t-1} & \text{for } t > 0 \end{cases} \tag{3.3}$$

The Exponential Moving Average is, naturally, a more reactive approach than the simple moving one. With this approach the features were expected to contribute nicely to predict the next observation and and perhaps could also be of use when a trend is exhibited by the dataset.

Not only the mean was considered but similarly some standard deviation features were also included as well as a minimum and a maximum detected in specific windows. These features were created with the following parameters:

- SMA1 - computed with a sliding window of size equal to the seasonality *m* detected previously;
- SMA2 - computed with a sliding window of size equal to 2 times the seasonality *m* detected previously;
- SMA3 - computed with a sliding window of size equal to 3 times the seasonality *m* detected previously;
- EMA1 - computed assigning α a weight of 0.1 (less reactive);
- EMA2 - computed assigning α a weight of 0.45;
- EMA3 - computed assigning α a weight of 0.9 (more reactive);
- SMA_diff = SMA1 - SMA3;
- EMA_diff = EMA3 - EMA1;
- std1 - the standard deviation computed with a sliding window of size equal to the seasonality *m* detected previously;
- std3 - the standard deviation computed with a sliding window of size equal to 3 times the seasonality *m* detected previously;
- Last_min - the minimum detected in a sliding window with size equal to 3 times the seasonality *m* detected previously;
- Last_max - the maximum detected in a sliding window with size equal to 3 times the seasonality *m* detected previously.

To take full advantage of the previously detected seasonality some seasonal "trend" features were introduced with the aid of the 'Seasonal Indices'. The idea behind them was similar to try and

model how the mean has behaved in the last s seasons. For instance, in the monthly temperature dataset the feature would answer the question 'What was the mean of the last two Januaries?" and easily adapting for each month. The features created were:

- Seasonal_avg_2 - the mean of the last 2 seasonal periods for period m, where m is equal to each index in Seasonal Indices feature;
- Seasonal_avg_3 - mean of the last 3 seasonal periods for period m;
- Seasonal_min - minimum detected in the last 3 seasonal periods for period m;
- Seasonal_max - minimum detected in the last 3 seasonal periods for period m;;
- Seasonal_std_2 - the standard of the last 2 seasonal periods for period m;
- Seasonal_std_3 - the standard of the last 3 seasonal periods for period m;
- Seasonal_avg_diff = Seasonal_avg_2 - Seasonal_avg_3;
- Seasonal_std_diff = Seasonal_std_2 - Seasonal_std_3.

With these the model could potentially analyze the behaviour of the target value for the last seasonal periods respectively.

To finish an additional 3 features were also introduced:

- Y_lag_1 - the last observation;
- Y_lag_m - m lagged observation where m is equal to each index in Seasonal Indices feature;
- Y_diff = Y_lag_s - Y_lag_1;

These were introduced as a nod to the frequent use of lagged observations in the literature and to hopefully provide some context regarding all the other features created. It is worth noticing that all these features are created upon the identification of the seasonality m and even though some features may not make sense for certain algorithms all features were considered since this work proposes some level of feature importance and analysis. In the end features might end up discarded for better predictions meaning that a correlation between certain features and machine learning models can be deduced making this work a hybrid approach.

3.3 Resampling methods

To train the data and validate the model two approaches were taken: growing window and sliding window. These approaches take place on a time series problem instead of, for example, the commonly used cross-validation or train-test splits [25] because the sequential order of the time series requires it. Being a sequence of observations ordered in time it makes sense to preserve that order during the training of the models and not use a method like the cross-validation one

where the model would use future data to predict historical data. The literature doesn't provide a clear answer on which one is the best approach, sometimes the sliding window is used [26] and sometimes the growing window is used [27].

This dissertation experimented with the two methods and some results are shown in chapter 4.

3.3.1 Growing Window

The Growing Window (GW), or expanding window, method starts with a window size for the training and makes the prediction to the following observation (or observations depending on the prediction horizon). After that the window then expands to accommodate the collection of more observations, trains the new (bigger) window and performs the test on the next observation. This method is illustrated on figure 3.5. For the starting window size, a window size of 3 times the seasonal period previously detected was used. For example, in the temperature dataset that would mean a window size of 36.

Taking into account that the previously moving averages produced some 'NaN' values for 3 cycles of seasonality and now for training 3 more cycles are needed to start the training the requirement of having at least 7 periods of seasonality is now justified.

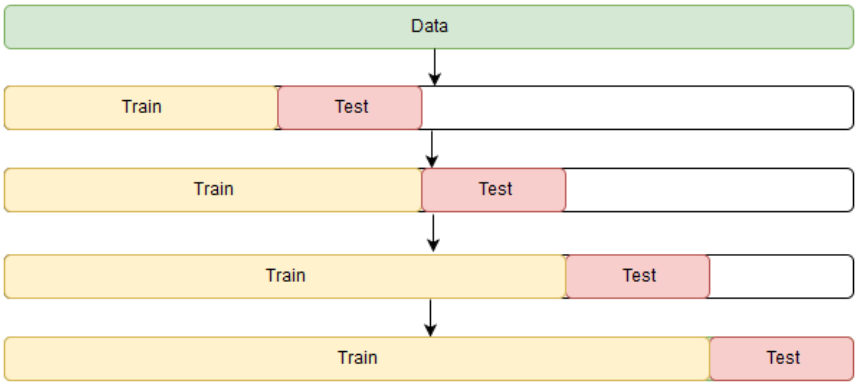

Figure 3.5: Simple illustration of the growing window process of training and testing data

Source: https://community.dataquest.io/t/how-to-split-time-series-data-into-training-and-test-set/4116/2

3.3.2 Sliding Window

The Sliding Window (SW), or rolling window, method differs from the growing window because the window size stays the same throughout the process which means that instead of growing to collect more observations the window simply slides to collect the next observation while dropping the first one. This method is demonstrated on figure 3.6. For the window size the same approach as the growing window was taken.

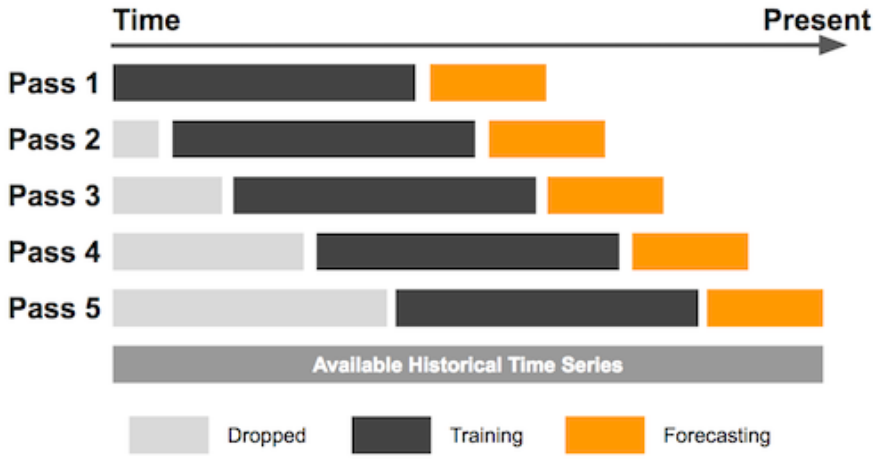

Figure 3.6: Simple illustration of the sliding window process of training and testing data

Source: https://www.kaggle.com/cworsnup/backtesting-cross-validation-for-timeseries/notebook

3.4 Performance Metrics

Essentially the performance will be evaluated through errors. Forecasting errors do not necessarily mean the prediction was bad, but rather a reflection of the unpredictability of the data. A vast array of different metrics is available to use in forecasting problems, all with with particular advantages and disadvantages and different depending factors, which could even mean that for certain models one approach might be more suited than another one. A good discussion of different metrics is found in [28], which takes a look at the metrics used in the M3 competition [9]. Generally the error can be defined by the equation 3.4:

$$e_t = Y_t - F_t \tag{3.4}$$

where Y_t is the observation at time t and F_t the forecast of Y_t.

The scope of this dissertation doesn't really include a study on what is the most suited metric but instead uses 4 different ones to try and evaluate the performance of this work and to later try and see if anything is gained by optimizing hyperparameters to minimize one of these metrics.

3.4.1 Mean Squared Error

The Mean Squared Error (MSE) is part of the family of scale-dependent measures: those whose accuracy scale depends on the scale of the data. This type of error is then suited when applying different machine learning methods to the same data. The mean squared error can be defined by the equation 3.5:

$$MSE = mean(e_t)^2 \tag{3.5}$$

The MSE is always non-negative and values closer to zero are better.

3.4.2 Root Mean Squared Error

The Root Mean Squared Error (RMSE) is given by the equation 3.6:

$$RMSE = \sqrt{MSE} \tag{3.6}$$

The RMSE is also a scalar dependent measure, is also non-zero and the closer to zero the better the fit. Historically it has been preferred to the MSE and the larger the error the bigger the RMSE which makes it sensitive to outliers.

3.4.3 Mean Absolute Percentage Error

The Mean Absolute Percentage Error (MAPE) is part of the faimly of measures based on percentage errors. Having the percentage error be defined by the equation 3.7:

$$p_t = \frac{100e_t}{Y_t} \tag{3.7}$$

Then MAPE is given by the equation 3.8:

$$MAPE = mean(|p_t|) \tag{3.8}$$

However this approachg has its drawbacks like cases where zero values appear (take demanding stock for example) and it also puts a heavier penalty on positive errors than on negative ones, leading to the creation of the next metric: sMAPE

3.4.4 Symmetric Mean Absolute Percentage Error

The Symmetric Mean Absolute Percentage Error (sMAPE) is defined by the equation 3.9:

$$sMAPE = mean\left(\frac{200|Y_t - F_t|}{(Y_t + F_t)}\right) \tag{3.9}$$

3.5 Hyperparameter Optimization

Hyperparameter Optimization has a long history of importance in machine learning. For this dissertation an optimization approach was chosen making use of Optuna [29], a state of the art Pyhton module for optimization. The idea behind this method is, for each machine learning algorithm, select a few parameters to tune and assign different values or range of values that they can take. Optuna then performs a series of trials (also defined in number) testing different sets of values for the parameters and calculating the score. In the end the best set of parameters is chosen to be in our model. The score metric decided to be used in this optimization process was the sMAPE since it is a very popular metric in time series problems right now and provides a good result when taken into account the metrics that are being calculated in this work. In the end, finding the specific best optimal hyperparameters for each model is beyond the scope of this dissertation since that

requires an in-depth study of each machine learning method and the parameters themselves and might even require a different score metric depending on the algorithm. The main objective here was to prove that something of value will be attained by doing this optimization and to try and achieve competitive results.

For the optimization process initially both the growing window and sliding window were tested to also try and give a bit of insight if there's a clear advantage in taking one approach over the other. This optimization process was used in LR, KNN, NN, RF and SVM and tested across all datasets. It is also worth noticing that all the machine learning models were implemented thourght scikit-learn [30] a state of the art Python module that provides tools for simple and efficient tools for predictive data analysis.

3.5.1 Linear regression

For Linear Regression not a whole lot of investigating needs to be done since the scikit-learn linear regressor admitted only four boolean parameters. There is no need for an in depth study of these parameters as they mainly control the intercept calculation, the use or not of regularization and the computation or not of normalization. Upon this a simple loop of optimization using 50 repetitions experimented with all the different parameters. Results for each dataset are shown in chapter 4.

3.5.2 K Nearest Neighbors

For K Nearest Neighbors the defining parameter seems to be the k value. For [11] a preset of 3 values (3, 5 and 7) was tested and the best result was the one participating in the model. For this dissertation an empirical approach will again be implemented, trying different values for k: 1 through 12; leaf_size (a parameter that affects the algorithm speed): between 1 and 50; and p: two values (1 and 2) for either the euclidean or the manhattan distance. The loop ran for 80 repetitions and results are shown in chapter 4.

3.5.3 Neural networks

Neural networks are on of the most customizable models in the field and its parameters impact not only the way the neural network is built but also the performance. One of the most important and studied parameters is the number of neurons in the hidden layer, since generally the the fewer the nodes the better the performance however a fewer number of nodes can translate to not having the power needed the model more complex data. In the literature not a universally accepted theoretical basis for this parameter exists, however some suggestions can be found. The paper [31] suggests a grid search approach to find such value and [32] studies the impact of this parameter trying different values. This dissertation will perform a experimental approach, trying different values during optimization and using the best value in the end.

For the activation function no clear best case is universally accepted since the performance can vary with properties such as the length of the time series or even the area of application. It still is an important parameter to tune deserving the attention of some studies [33]. Once again an

empirical approach will be used, trying the four different activation functions mentioned in chapter 2: identity, sigmoid, tahn and relu.

The last parameters were the batch size and number of iterations, two parameters that play a factor in the propagation of the samples through the network and its speed. Once again the empirical approach was taken here, experimenting with different values and choosing the best ones in the end. For this method a number of 80 trials was chosen.

3.5.4 Random Forests

For random forests one of the main parameters that is also tied with performance is the number of trees, where using at least 1000 trees is suggested by [7], on [34] a number between 64 and 128 is suggested based on experiments, [35] references 100 trees as the highest performance (which is actually the default number for the scikit-learn module implementation) and [21] uses 500 in its work. Regarding this parameter final models will use 1000 trees, but for performance issues during optimization it will use 100. Another parameter to tune is the minimum number samples required to split an internal node and the the minimum number of samples required to be at a leaf node. These parameters also deserve research in the literature and setting these values to 5 is a good example as seen in the satisfactory results of [36], [37] and [21]. During the optimization the empirical approach is once again taken as they'll take values ranging up to 50. The last parameter to tune was the the number of features to consider when looking for the best split, a parameter that was tested using the values 'auto', 'log2' and 'sqrt'.

Random Forest optimization loops ran for a total of 50 times.

3.5.5 Support Vector Machine

For Support Vector Machine few parameters were tuned and all followed the empirical path:

- kernel - could be linear, radial basis function, polynomial or sigmoid;
- gamma - a parameter that defines how far the influence of a single training example reaches (only the linear kernel doesn't use this parameter) adn that could assume 'scale' or 'auto' as values;
- C - the regularization parameter that was tested using values from 0.01 up to 100.

Support Vector Machine trials ran for a total of 50 times.

3.6 Feature Importance

Although not in depth this dissertation attempted to assess the importance of the features created for the model. This was achieved utilizing the permutation feature importance [38] which was implemented through the previously mentioned scikit-learn Python module.

The permutation feature importance approach is applied as described in [39] and can be summarized as follows:

1. Estimate the original model error e^{orig} = L(y, f(X))
2. For each feature j = 1,...,p do:
   (a) Generate feature matrix X^{perm} by permuting feature j in the data X. This breaks the association between feature j and true outcome y.
   (b) Estimate error e^{perm} = L(Y,f(Xperm)) based on the predictions of the permuted data.
   (c) Calculate permutation feature importance $FI^j = e^{perm}$ e^{orig}. Alternatively, the difference can be used: $FI^j = e^{perm}$ - e^{orig}
3. Sort features by descending FI.

The permutation feature importance can be interpreted as the increase in prediction error after shuffling the feature's values, giving great insight to what features the model depends the most when making a prediction. It's easy to analyze, compressed and also takes into account the relation with other features since breaking the order in a time series will also break the various feature's interactions. The results of this method will be discussed in chapter 4.

3.7 Classical Methods

The final part of this dissertation's method is to compare the results obtained in the new proposed framework with results obtained with the classical statistical methods.

3.7.1 ARIMA

For the ARIMA implementation the pmdarima [40] was used, a state of the art Python implementation for ARIMA models. This package allowed for some parameter tuning and performed an automated search for the best set of parameters to use in the ARIMA model that would best fit the data. Those parameters are the ones defined in chapter 2.

3.7.2 ETS

As mentioned in chapter 2 the Holt-Winter's method was used with the aid of [41], a state of the art Python module that provides classes and functions for the estimation of many different statistical models. For the Holt-Winter's parameters multiple scenarios were tested:

- additive trend;
- multiplicative trend;
- additive seasonality;
- multiplicative seasonality;
- with a Box-Cox transformation;

- without a Box-Cox transformation.

In the end the fit with the best sMAPE was chosen to be compared with the other machine learning models.

Chapter 4

Results

In this chapter the results of all simulations will be presented as well as the parameters used in order to obtain them, the different error values obtained including a comparison of the two training window approaches and also the feature importance analysis. As the results will show the growing window approach revealed better performance so for the permutation feature importance no experiments were done using the sliding window. All experiments were conducted for one step forecasts and the window size in the resampling methods is set to 36 (3 times the detected seasonality).

4.1 K Nearest Neighbors

4.1.1 Temperature dataset

For the growing window approach the best parameters obtained after optimization were:

- k neighbors: 7
- leaf_size: 10
- p: 1

The results using these parameters can be seen on fig 4.1 and produce a sMAPE of 4.5987.

For the sliding window approach the best parameters were:

- k neighbors: 7
- leaf_size: 3
- p: 1

The results using these parameters can be seen on fig 4.2 and produce a sMAPE of 4.6681.

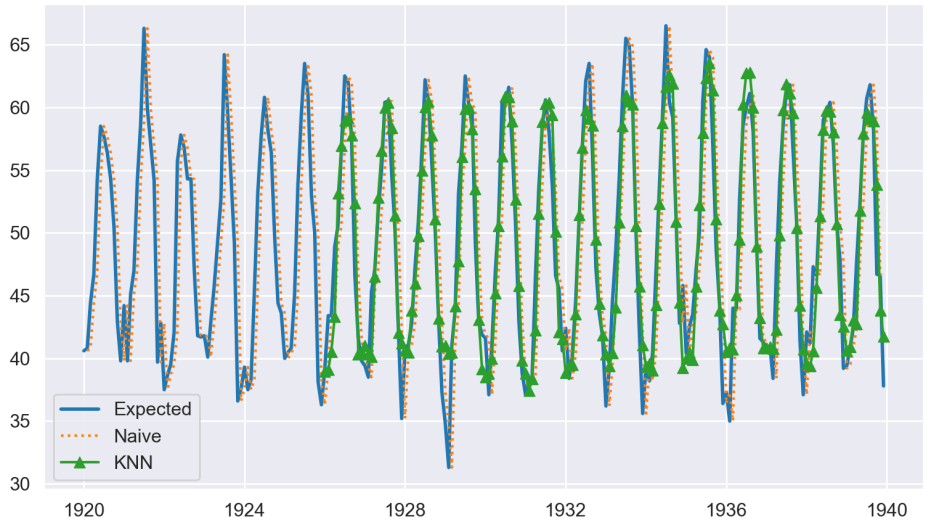

Figure 4.1: GW KNN, 1 step forecast for the temperature dataset

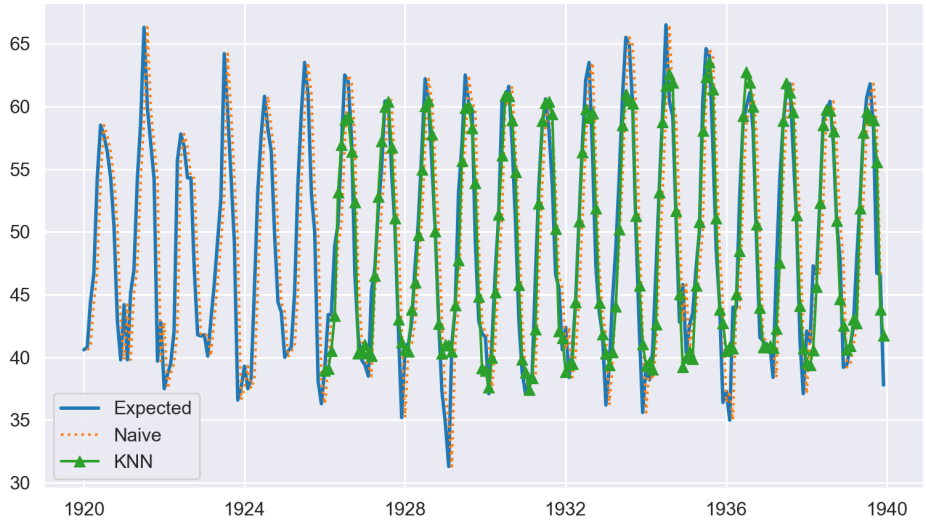

Figure 4.2: SW KNN, 1 step forecast for the temperature dataset

4.1.2 Champagne dataset

For the growing window approach the best parameters obtained after optimization were:

- k neighbors: 1
- leaf_size: 44

- p: 2

The results using these parameters can be seen on fig 4.3 and produce a sMAPE of 12.7012.

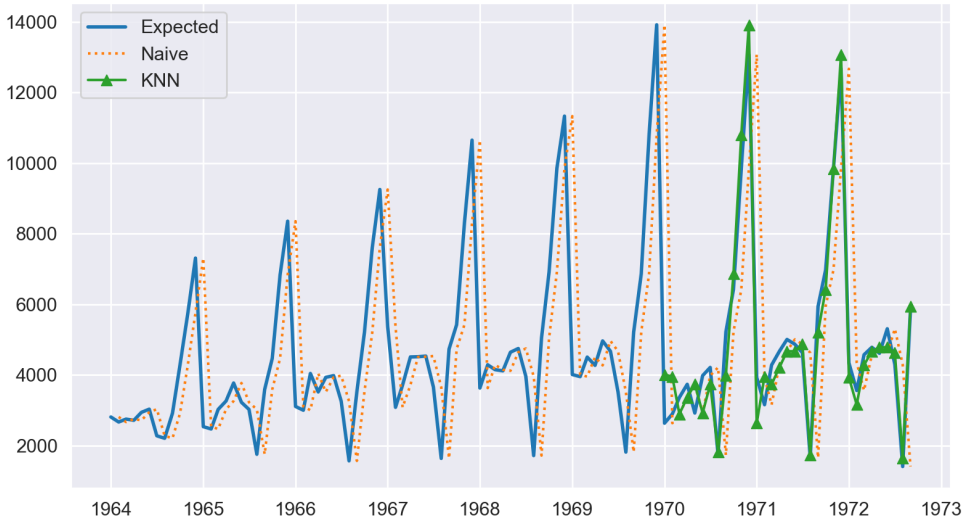

Figure 4.3: GW KNN, 1 step forecast for the champagne dataset

For the sliding window approach the best parameters were:

- k neighbors: 1
- leaf_size: 2
- p: 2

The results using these parameters can be seen on fig 4.4 and produce a sMAPE of 12.7018.

4.1.3 Ariline dataset

For the growing window approach the best parameters obtained after optimization were:

- k neighbors: 3
- leaf_size: 1
- p: 2

The results using these parameters can be seen on fig 4.5 and produce a sMAPE of 6.1785.

For the sliding window approach the best parameters were:

- k neighbors: 3

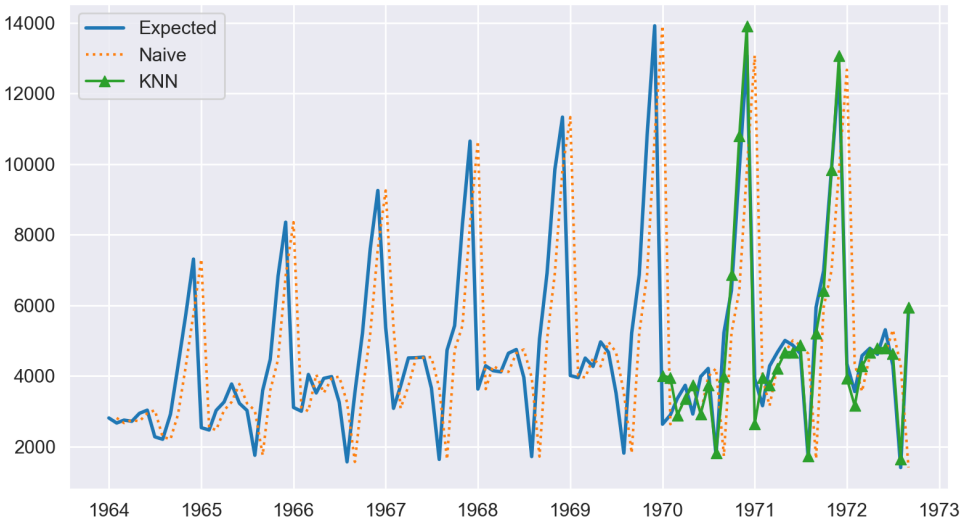

Figure 4.4: SW KNN, 1 step forecast for the champagne dataset

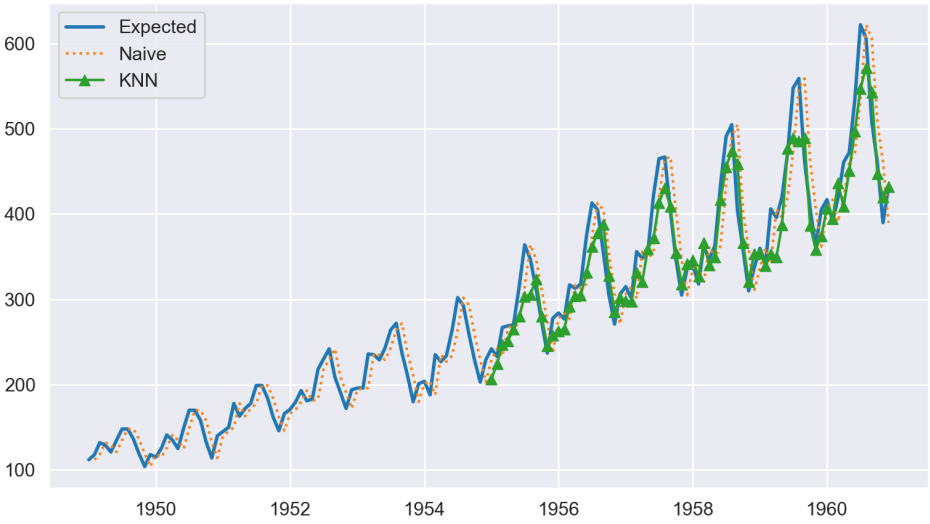

Figure 4.5: GW KNN, 1 step forecast for the airline dataset

- leaf_size: 2
- p: 2

The results using these parameters can be seen on fig 4.6 and produce a sMAPE of 6.1785.

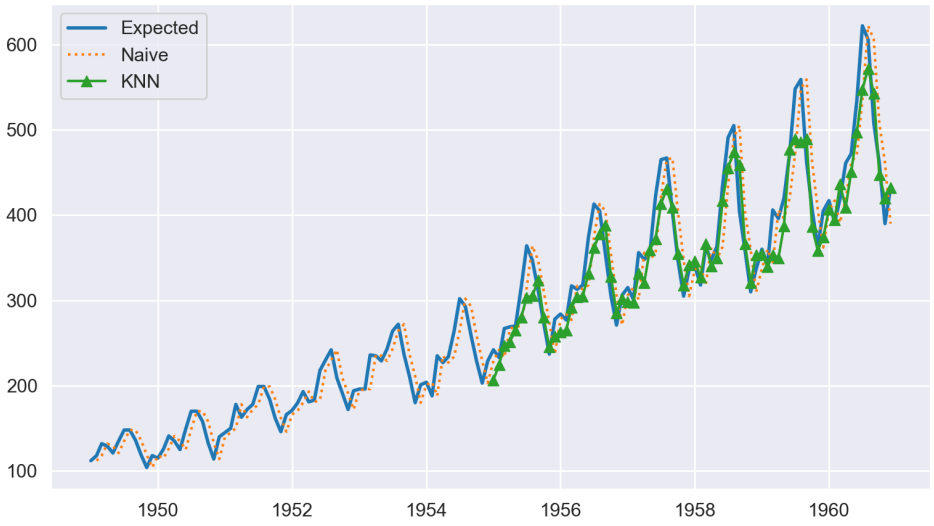

Figure 4.6: SW KNN, 1 step forecast for the airline dataset

4.2 Linear Regression

4.2.1 Temperature dataset

For the growing window approach the best parameters obtained after optimization were:

- fit_intercept: True
- normalize: False
- copy_X: False
- positive: True

The results using these parameters can be seen on fig 4.7 and produce a sMAPE of 4.0315.

For the sliding window approach the best parameters obtained after optimization were:

- fit_intercept: False
- normalize: False
- copy_X: True
- positive: True

The results using these parameters can be seen on fig 4.8 and produce a sMAPE of 4.6101.

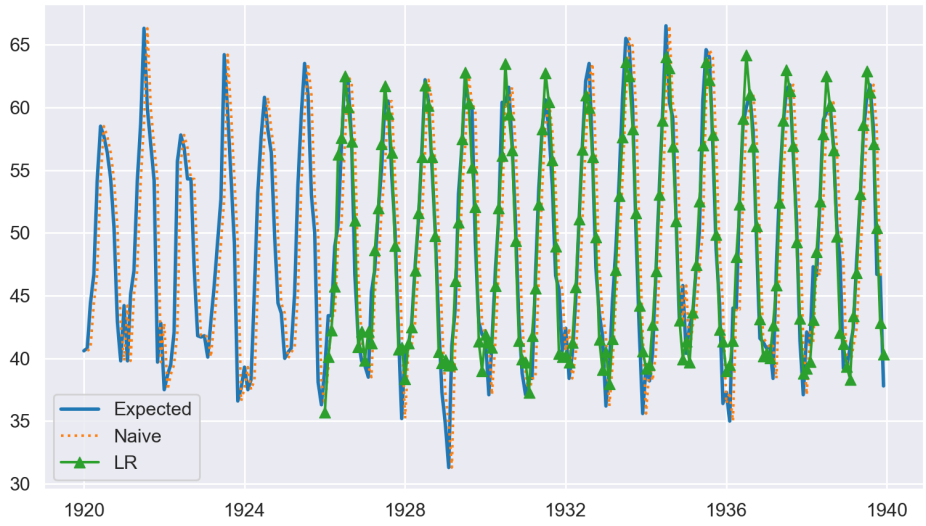

Figure 4.7: GW LR, 1 step forecast for the temperature dataset

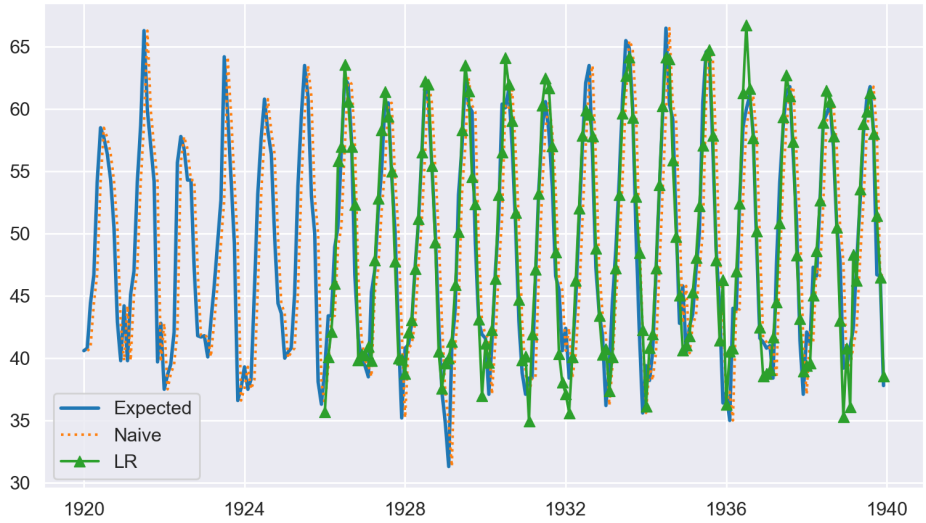

Figure 4.8: SW LR, 1 step forecast for the temperature dataset

4.2.2 Champagne dataset

For the growing window approach the best parameters obtained after optimization were:

- fit_intercept: False
- normalize: False

- copy_X: False
- positive: True

The results using these parameters can be seen on fig 4.9 and produce a sMAPE of 15.1476.

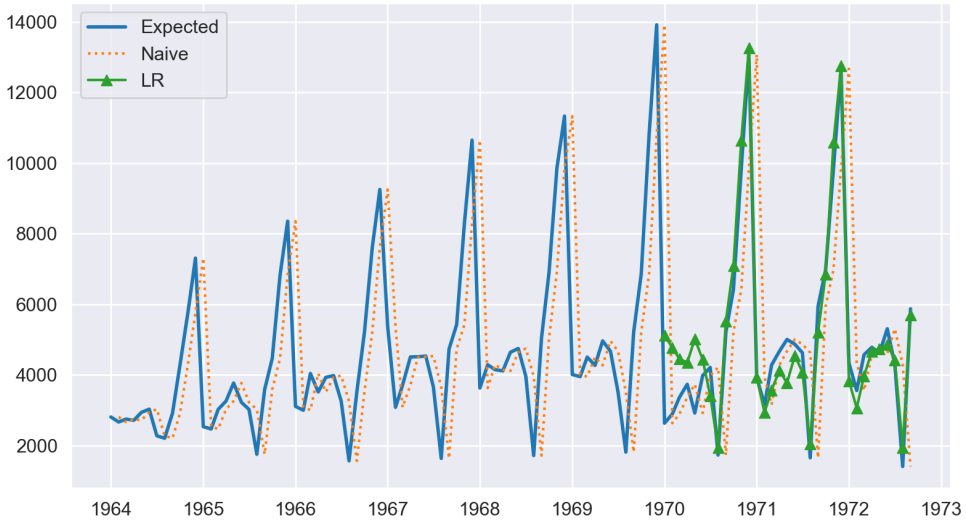

Figure 4.9: GW LR, 1 step forecast for the champagne dataset

For the sliding window approach the best parameters obtained after optimization were:

- fit_intercept: False
- normalize: False
- copy_X: True
- positive: True

The results using these parameters can be seen on fig 4.10 and produce a sMAPE of 14.1902.

4.2.3 Airline dataset

For the growing window approach the best parameters obtained after optimization were:

- fit_intercept: False
- normalize: False
- copy_X: True
- positive: True

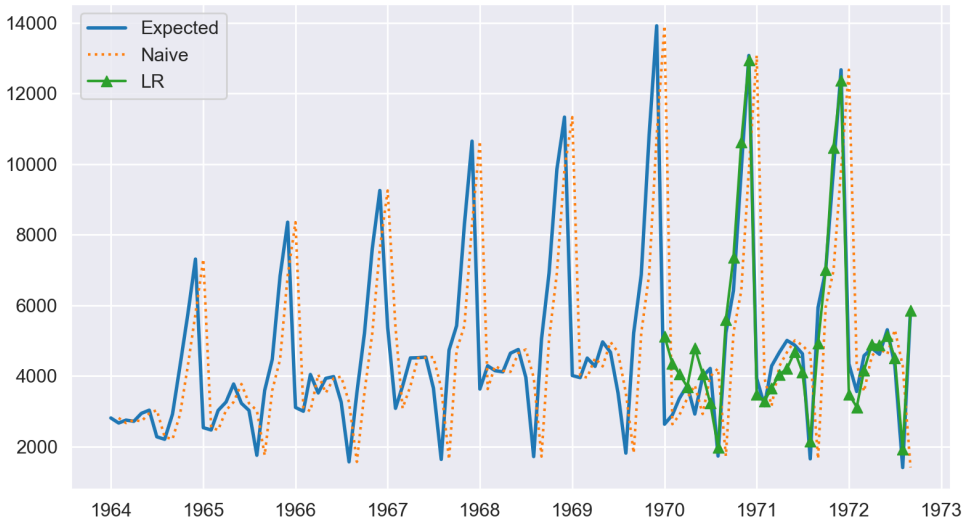

Figure 4.10: SW LR, 1 step forecast for the champagne dataset

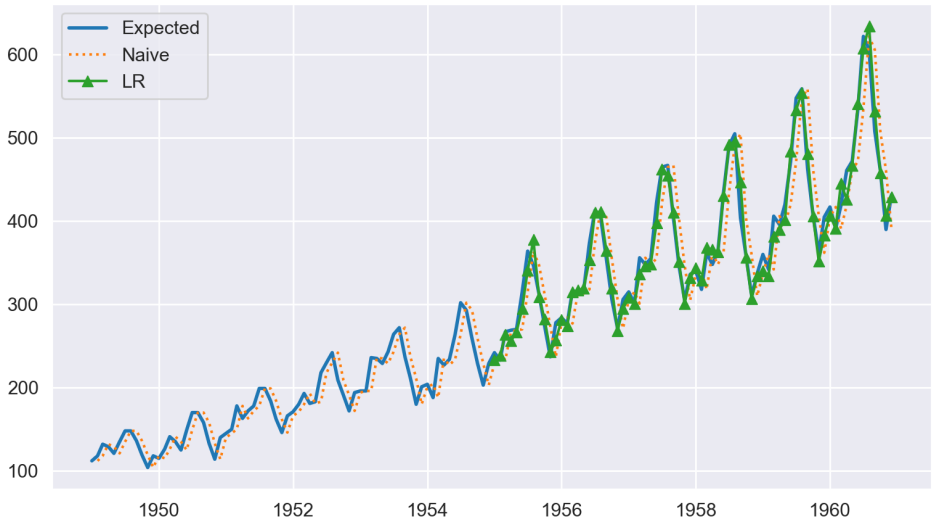

Figure 4.11: GW LR, 1 step forecast for the airline dataset

The results using these parameters can be seen on fig 4.11 and produce a sMAPE of 2.7637.

For the sliding window approach the best parameters obtained after optimization were:

- fit_intercept: False
- normalize: False

- copy_X: True
- positive: True

The results using these parameters can be seen on fig 4.12 and produce a sMAPE of 2.9595.

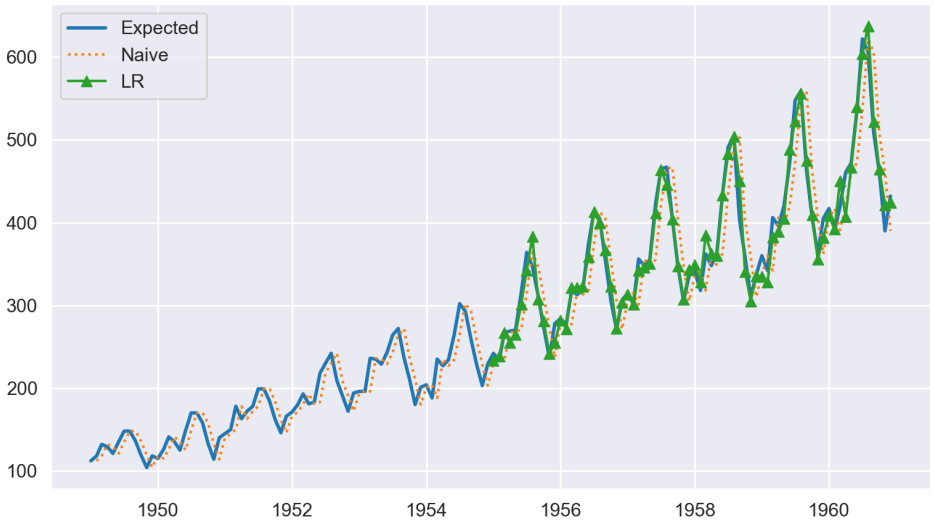

Figure 4.12: SW LR, 1 step forecast for the airline dataset

4.3 Neural Networks

4.3.1 Temperature dataset

For the growing window approach the best parameters obtained after optimization were:

- number of hidden layers: 5
- number of units per hidden layer: (4, 1, 1, 1, 30)
- activation: identity
- solver: lbgfs
- max_iter: 118
- batch_size: 159

The results using these parameters can be seen on fig 4.13 and produce a sMAPE of 4.0889.

For the sliding window approach the best parameters obtained after optimization were:

- number of hidden layers: 5

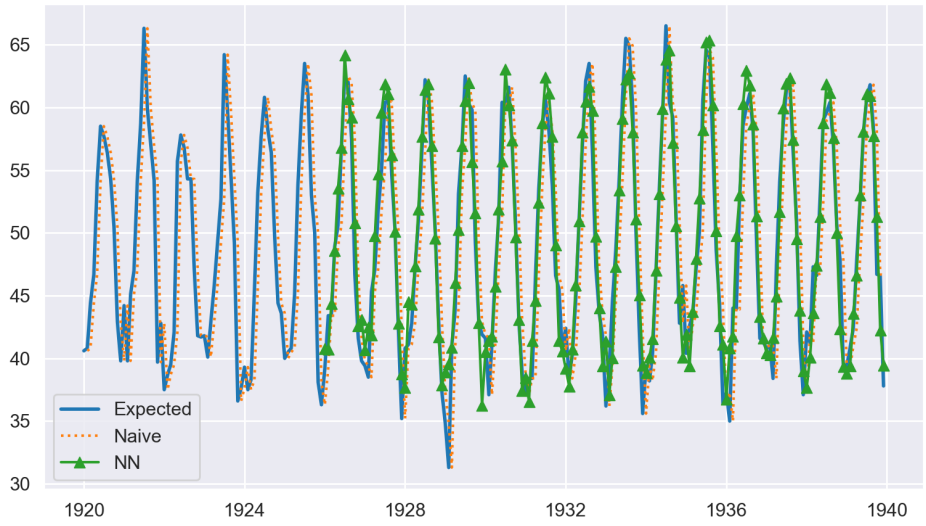

Figure 4.13: GW NN, 1 step forecast for the temperature dataset

- number of units per hidden layer: (4, 1, 1, 2, 17)
- activation: identity
- solver: lbgfs
- max_iter: 103
- batch_size: 383

The results using these parameters can be seen on fig 4.14 and produce a sMAPE of 4.5386.

4.3.2 Champagne dataset

For the growing window approach the best parameters obtained after optimization were:

- number of hidden layers: 5
- number of units per hidden layer: (4, 1, 1, 7, 42)
- activation: identity
- solver: lbgfs
- max_iter: 326
- batch_size: 456

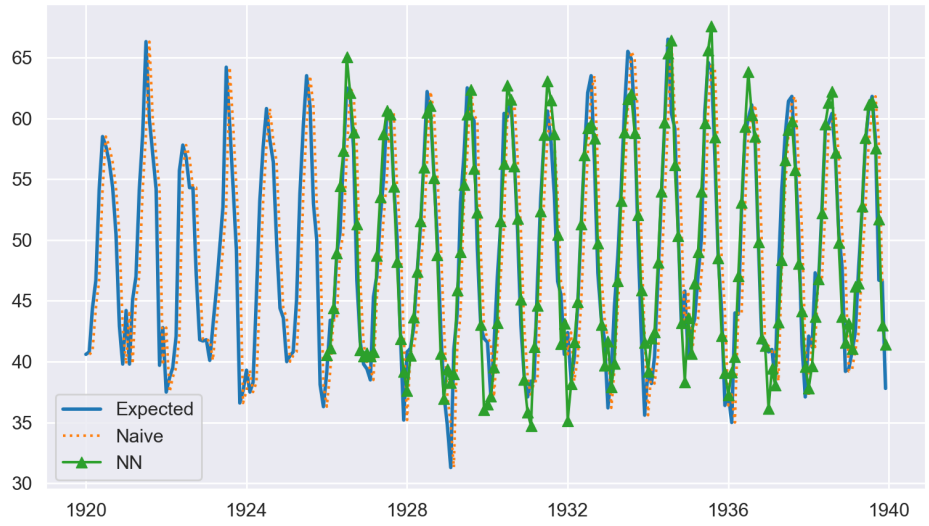

Figure 4.14: SW NN, 1 step forecast for the temperature dataset

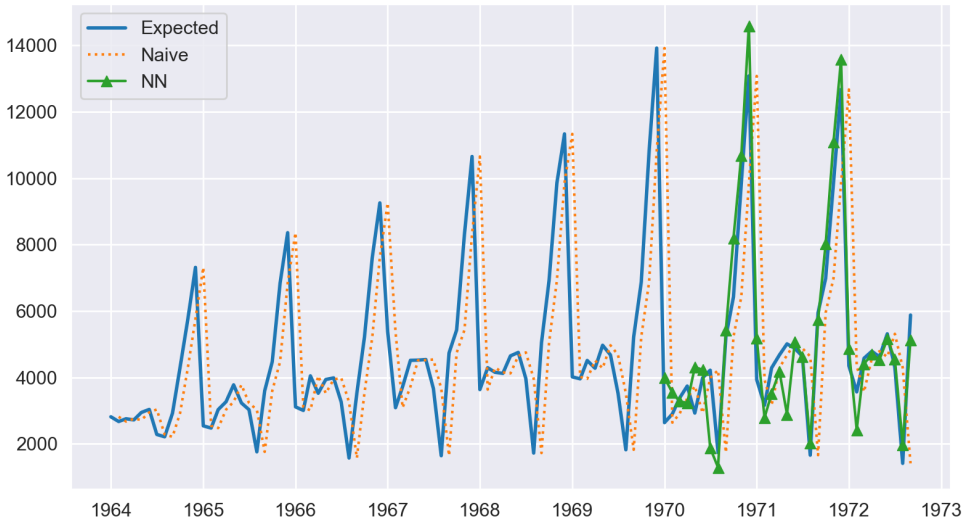

Figure 4.15: GW NN, 1 step forecast for the champagne dataset

The results using these parameters can be seen on fig 4.15 and produce a sMAPE of 17.3726.

For the sliding window approach the best parameters obtained after optimization were:

- number of hidden layers: 1
- number of units per hidden layer: 1

- activation: identity
- solver: lbgfs
- max_iter: 321
- batch_size: 273

The results using these parameters can be seen on fig 4.16 and produce a sMAPE of 18.5388.

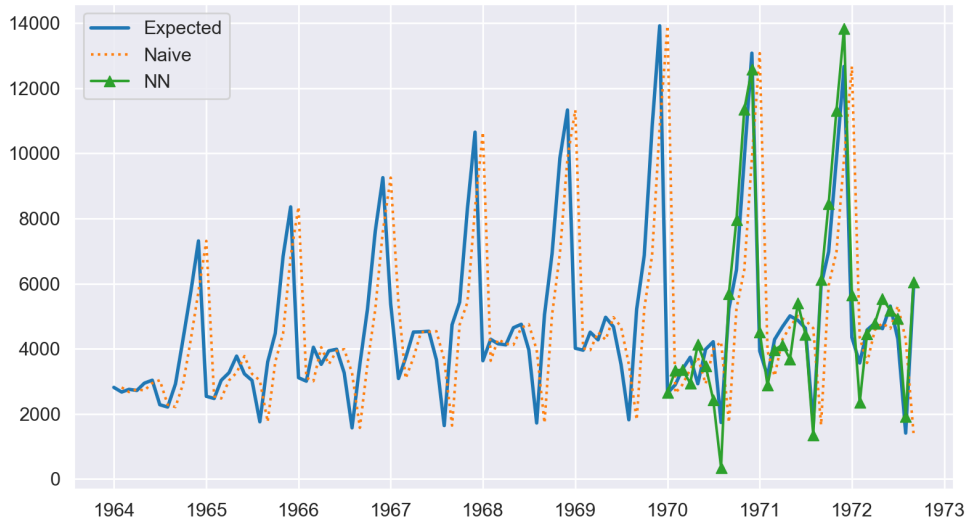

Figure 4.16: SW NN, 1 step forecast for the champagne dataset

4.3.3 Airline dataset

For the growing window approach the best parameters obtained after optimization were:

- number of hidden layers: 4
- number of units per hidden layer: (3, 56, 1, 16)
- activation: identity
- solver: lbgfs
- max_iter: 114
- batch_size: 421

The results using these parameters can be seen on fig 4.17 and produce a sMAPE of 3.0400.

For the sliding window approach the best parameters obtained after optimization were:

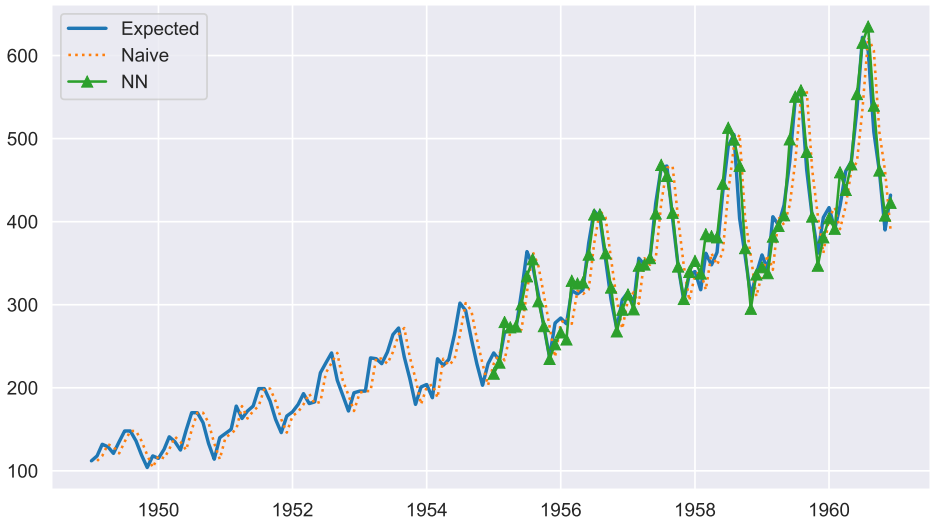

Figure 4.17: GW NN, 1 step forecast for the airline dataset

- number of hidden layers: 3
- number of units per hidden layer: (2, 1, 11)
- activation: identity
- solver: lbgfs
- max_iter: 184
- batch_size: 482

The results using these parameters can be seen on fig 4.18 and produce a sMAPE of 3.6686.

4.4 Random Forests

4.4.1 Temperature dataset

For the growing window approach the best parameters obtained after optimization were:

- number of estimators 1000
- max_features: log2
- min_samples_leaf: 2
- min_samples_split: 2

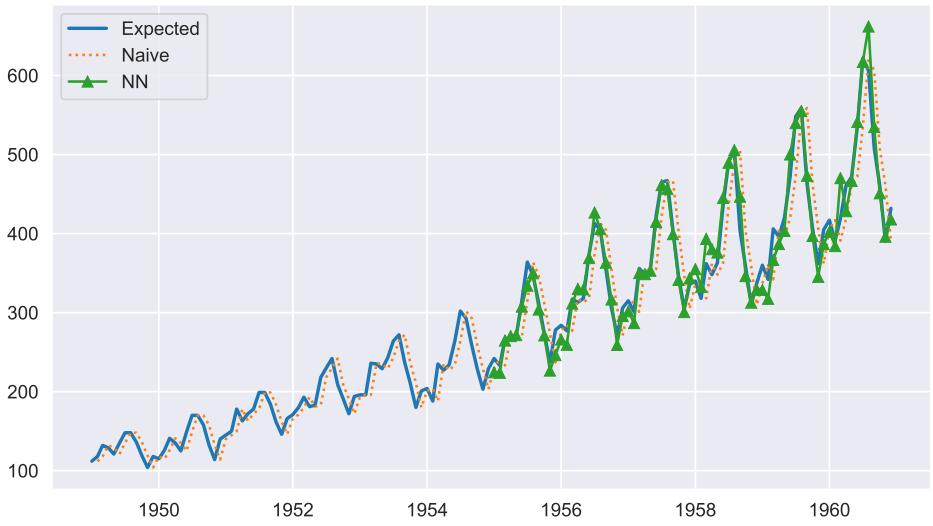

Figure 4.18: SW NN, 1 step forecast for the airline dataset

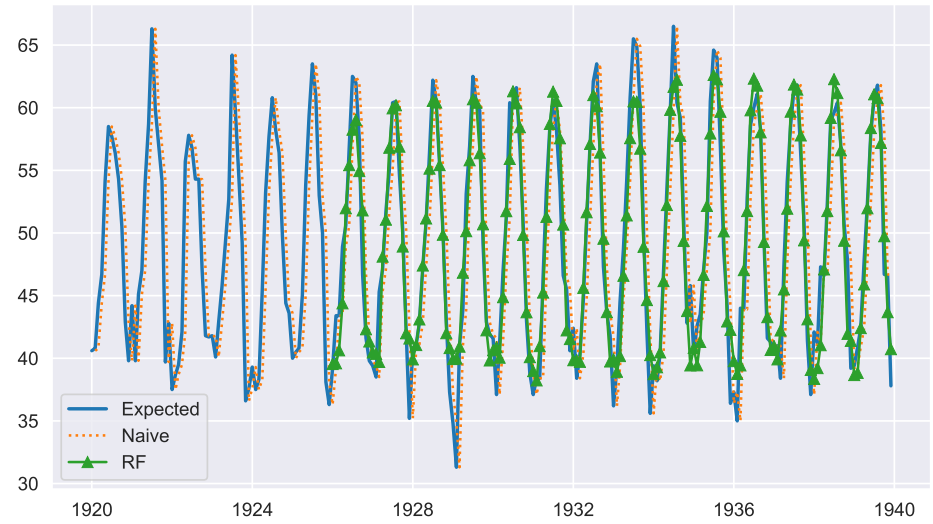

Figure 4.19: GW RF, 1 step forecast for the temperature dataset

The results using these parameters can be seen on fig 4.19 and produce a sMAPE of 4.1154.

For the sliding window approach the best parameters obtained after optimization were:

- number of estimators 1000
- max_features: auto

- min_samples_leaf: 1
- min_samples_split: 4

The results using these parameters can be seen on fig 4.20 and produce a sMAPE of 4.4935.

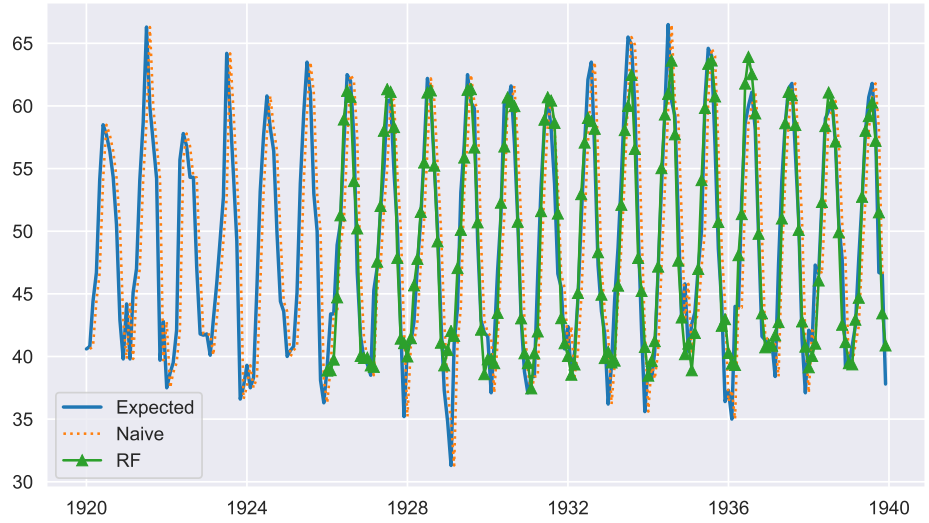

Figure 4.20: SW RF, 1 step forecast for the temperature dataset

4.4.2 Champagne dataset

For the growing window approach the best parameters obtained after optimization were:

- number of estimators 1000
- max_features: log2
- min_samples_leaf: 1
- min_samples_split: 2

The results using these parameters can be seen on fig 4.21 and produce a sMAPE of 15.4078.

For the sliding window approach the best parameters obtained after optimization were:

- number of estimators 1000
- max_features: sqrt
- min_samples_leaf: 1
- min_samples_split: 2

The results using these parameters can be seen on fig 4.22 and produce a sMAPE of 16.5232.

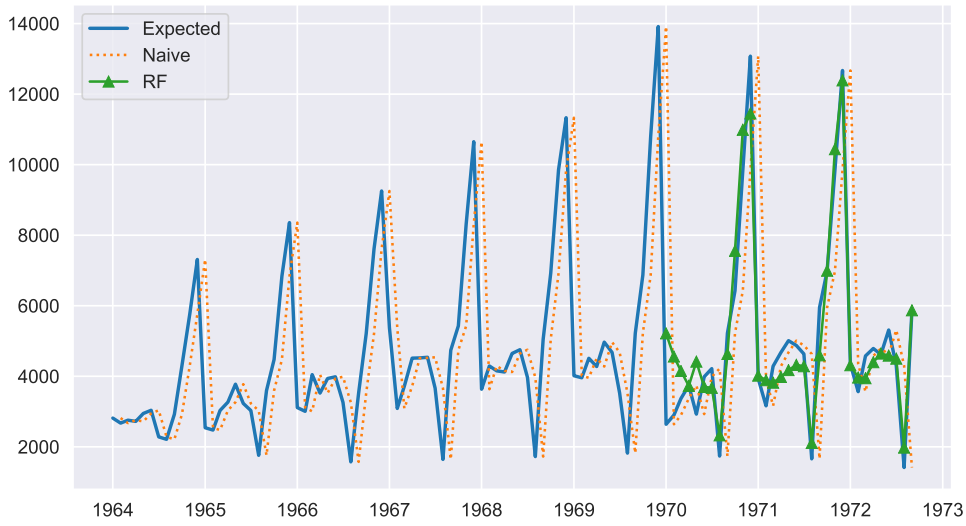

Figure 4.21: GW RF, 1 step forecast for the champagne dataset

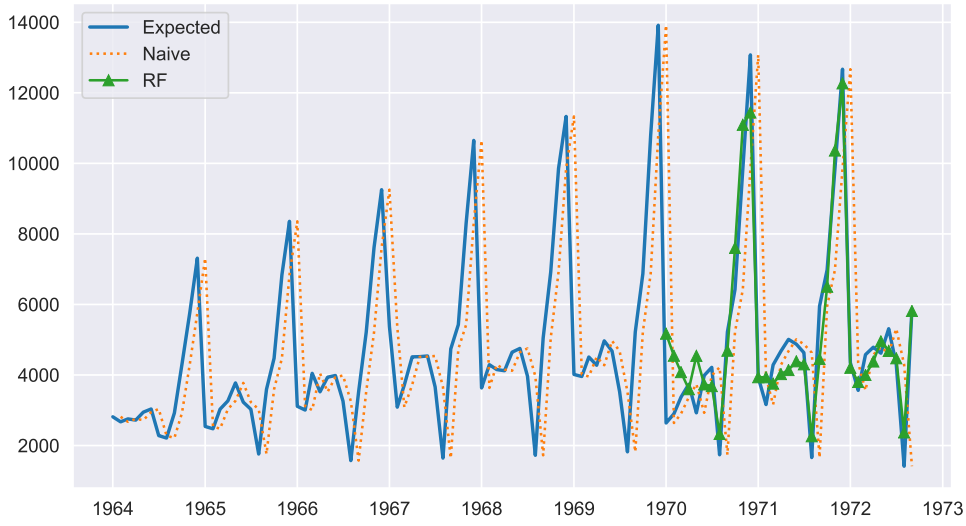

Figure 4.22: SW RF, 1 step forecast for the champagne dataset

4.4.3 Airline dataset

For the growing window approach the best parameters obtained after optimization were:

- number of estimators 1000
- max_features: auto

- min_samples_leaf: 1
- min_samples_split: 2

The results using these parameters can be seen on fig 4.23 and produce a sMAPE of 5.3726.

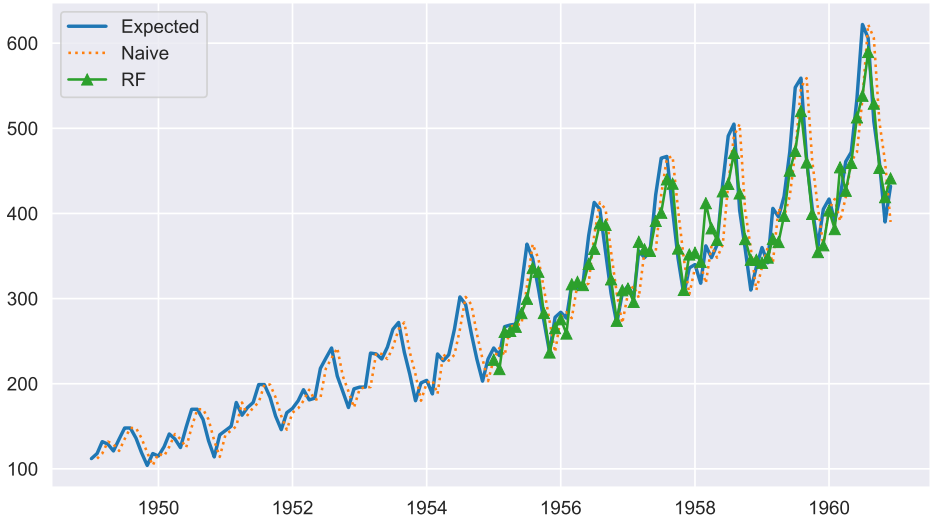

Figure 4.23: GW RF, 1 step forecast for the airline dataset

For the sliding window approach the best parameters obtained after optimization were:

- number of estimators 1000
- max_features: sqrt
- min_samples_leaf: 1
- min_samples_split: 2

The results using these parameters can be seen on fig 4.24 and produce a sMAPE of 5.7938.

4.5 Support Vector Machine

For the Support Vector Machine the parameter 'gamma' was ommited since all the simulations achieved best results using the linear kernel which doesn't make use of such parameter.

4.5.1 Temperature dataset

For the growing window approach the best parameters obtained after optimization were:

- kernel: linear

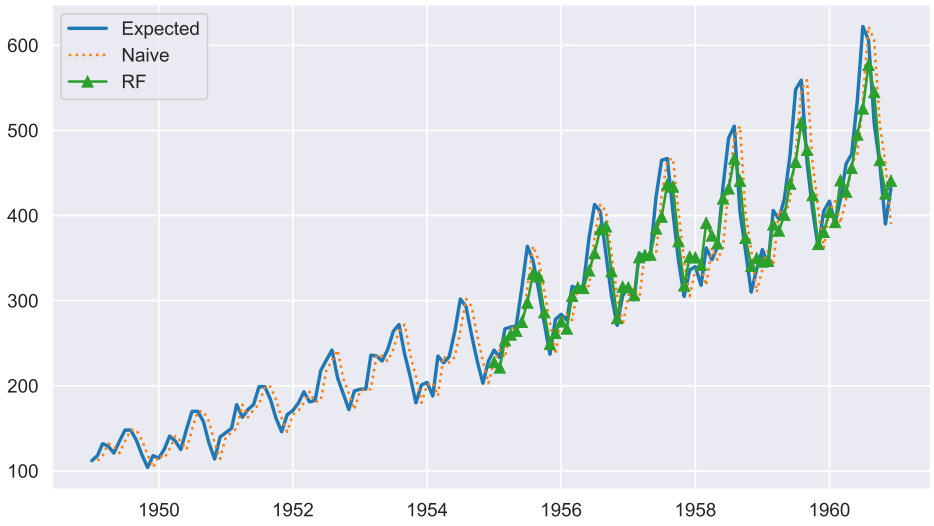

Figure 4.24: SW RF, 1 step forecast for the airline dataset

- C: 0.02025

The results using these parameters can be seen on fig 4.25 and produce a sMAPE of 4.0073.

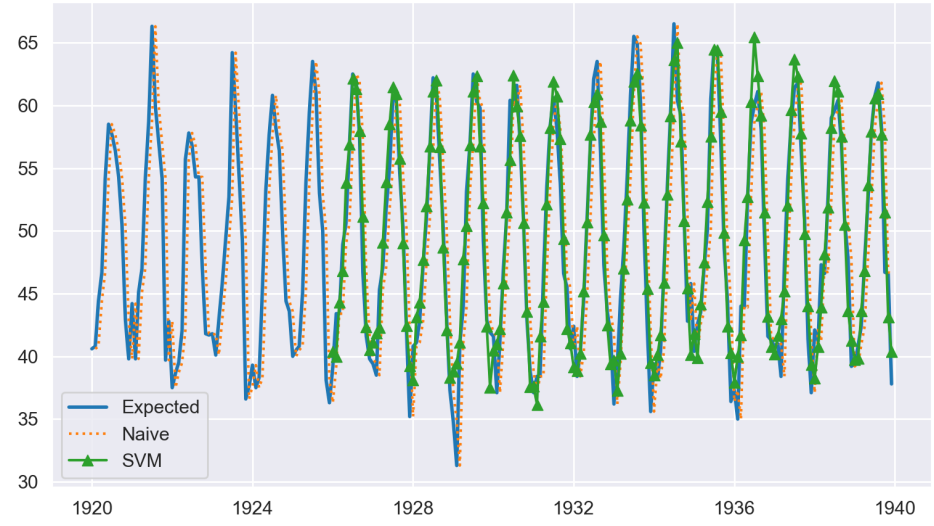

Figure 4.25: GW SVM, 1 step forecast for the temperature dataset

For the sliding window approach the best parameters obtained after optimization were:

- kernel: linear

- C: 0.0074845

The results using these parameters can be seen on fig 4.26 and produce a sMAPE of 4.2852.

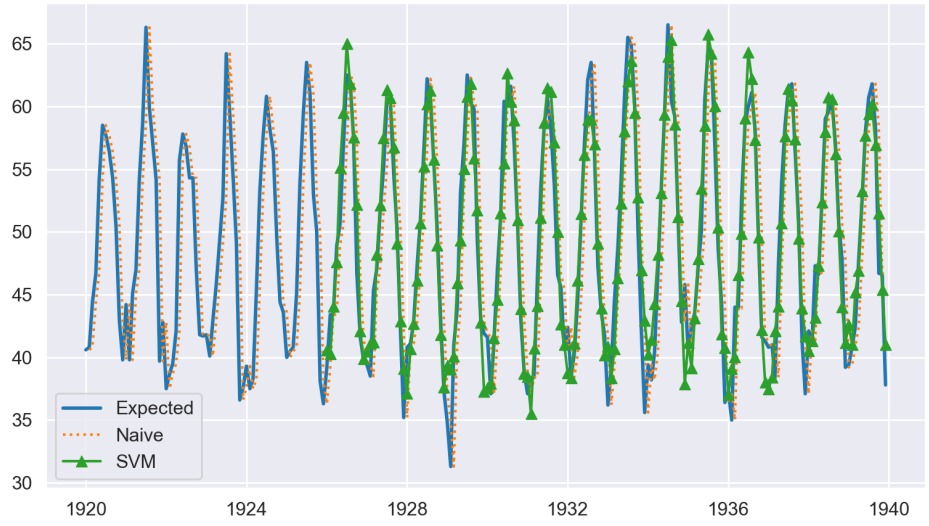

Figure 4.26: SW SVM, 1 step forecast for the temperature dataset

4.5.2 Champagne dataset

For the growing window approach the best parameters obtained after optimization were:

- kernel: linear
- C: 0.01596

The results using these parameters can be seen on fig 4.27 and produce a sMAPE of 17.9094.

For the sliding window approach the best parameters obtained after optimization were:

- kernel: linear
- C: 0.207246

The results using these parameters can be seen on fig 4.28 and produce a sMAPE of 19.2189.

4.5.3 Airline dataset

For the growing window approach the best parameters obtained after optimization were:

- kernel: linear
- C: 0.009307

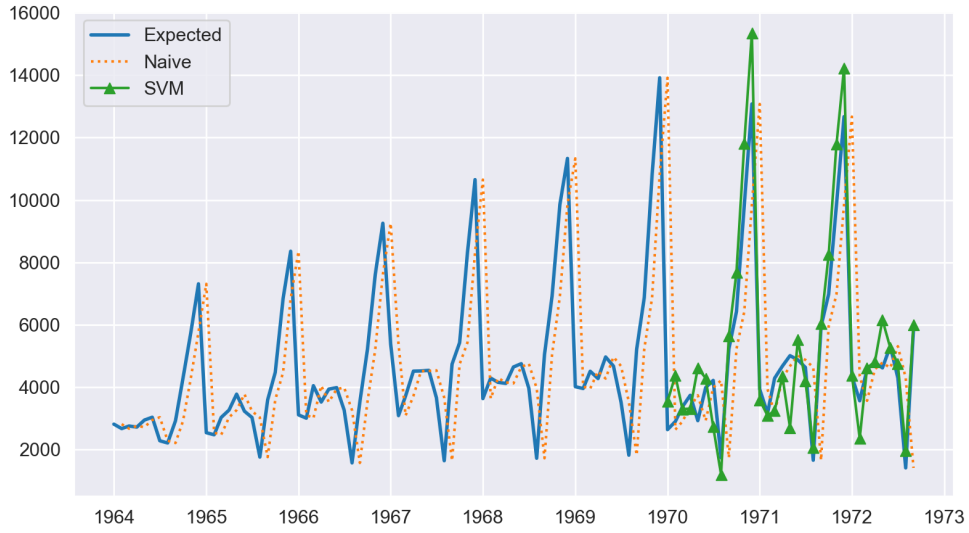

Figure 4.27: GW SVM, 1 step forecast for the champagne dataset

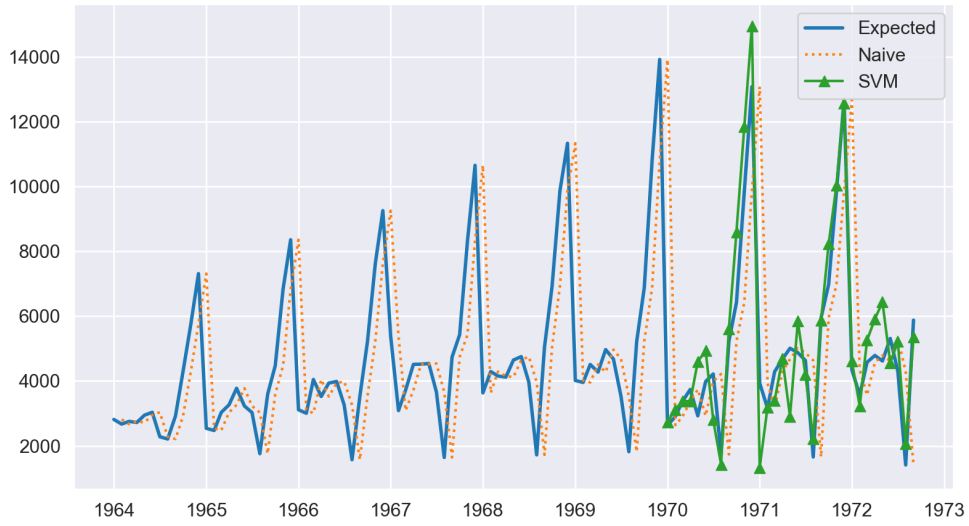

Figure 4.28: SW SVM, 1 step forecast for the champagne dataset

The results using these parameters can be seen on fig 4.29 and produce a sMAPE of 3.4597.

For the sliding window approach the best parameters obtained after optimization were:

- kernel: linear
- C: 0.00101

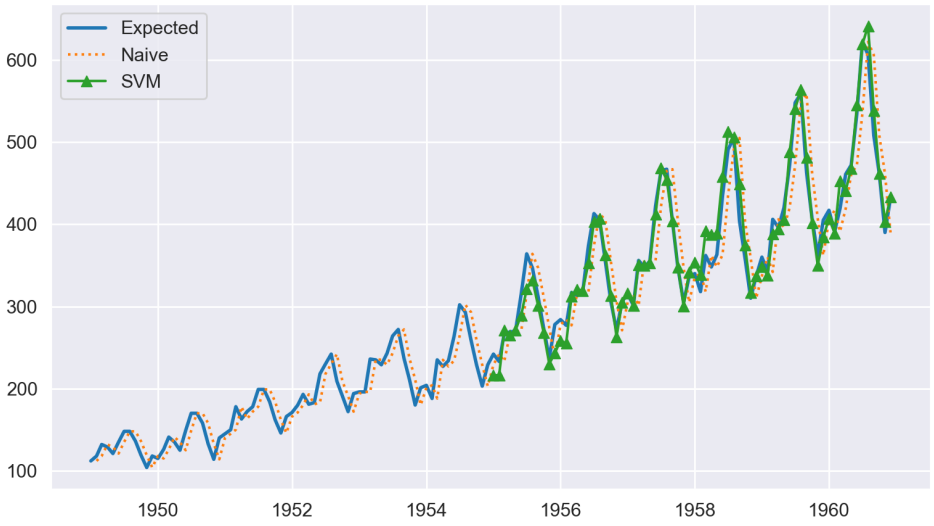

Figure 4.29: GW SVM, 1 step forecast for the airline dataset

The results using these parameters can be seen on fig 4.30 and produce a sMAPE of 4.1020.

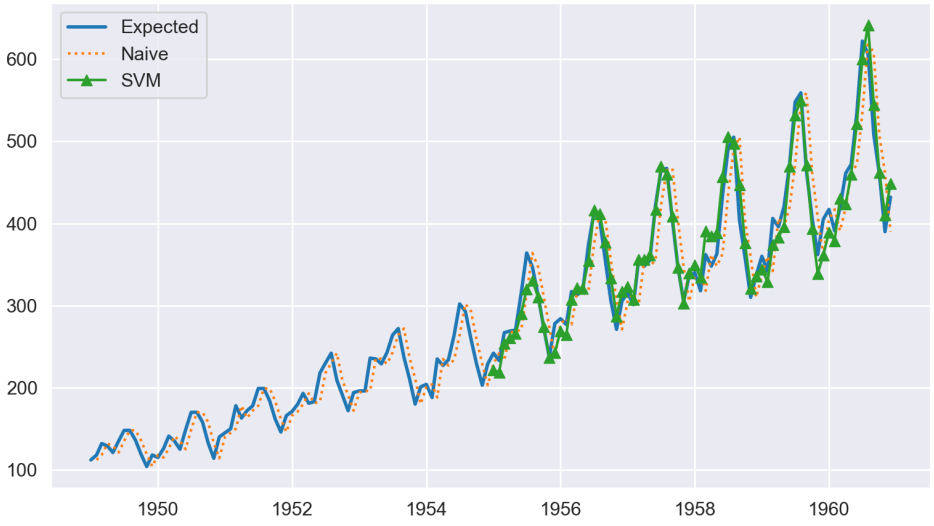

Figure 4.30: SW SVM, 1 step forecast for the airline dataset

4.6 ARIMA

After looking at the worse performance metrics values for the sliding window approach and to save computational time only the growing window approach was applied for the ARIMA method.

4.6.1 Temperature dataset

For the growing window approach the best ARIMA model obtained after optimization was SARIMA$(1,0,0)(2,0,1)_{12}$. The results using these parameters can be seen on fig 4.31 and produce a sMAPE of 4.0353.

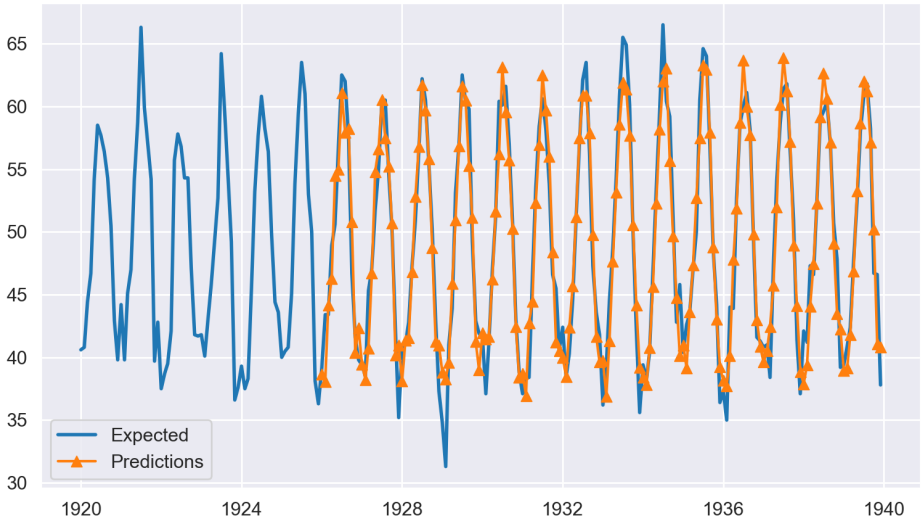

Figure 4.31: GW ARIMA, 1 step forecast for the temperature dataset

4.6.2 Champagne dataset

For the growing window approach the best ARIMA model obtained after optimization was SARIMA$(0,0,0)(0,1,0)_{12}$. The results using these parameters can be seen on fig 4.32 and produce a sMAPE of 16.2952.

4.6.3 Airline dataset

For the growing window approach the best ARIMA model obtained after optimization was SARIMA$(1,0,0)(1,1,0)_{12}$. The results using these parameters can be seen on fig 4.33 and produce a sMAPE of 2.6559.

4.7 ETS

After looking at the worse performance metrics values for the sliding window approach and to save computational time only the growing window approach was applied for the ETS method.

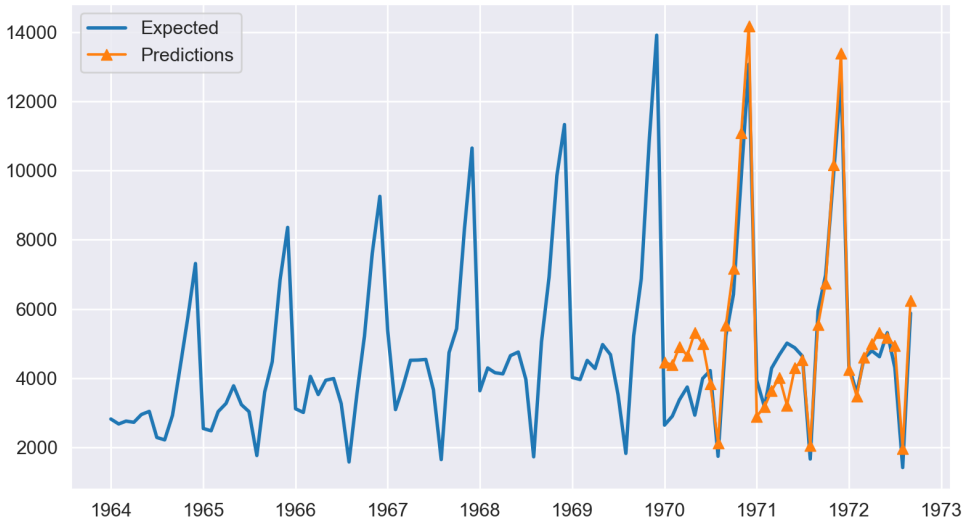

Figure 4.32: GW ARIMA, 1 step forecast for the champagne dataset

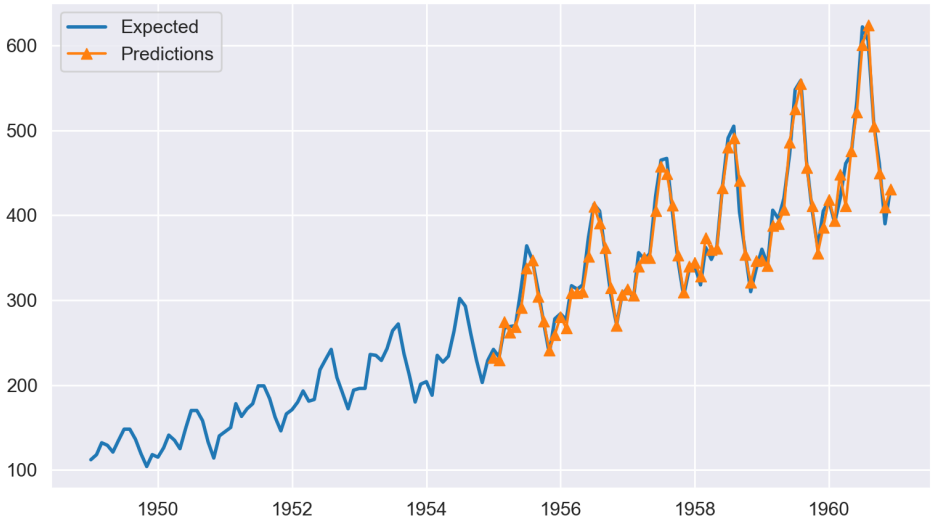

Figure 4.33: GW ARIMA, 1 step forecast for the airline dataset

4.7.1 Temperature dataset

For the growing window approach the best parameters obtained after optimization were:

- seasonal_periods: 12
- trend: add

- seasonal: mul
- use_boxcox: True
- initialization_method: estimated

The results using these parameters can be seen on fig 4.34 and produce a sMAPE of 4.0833.

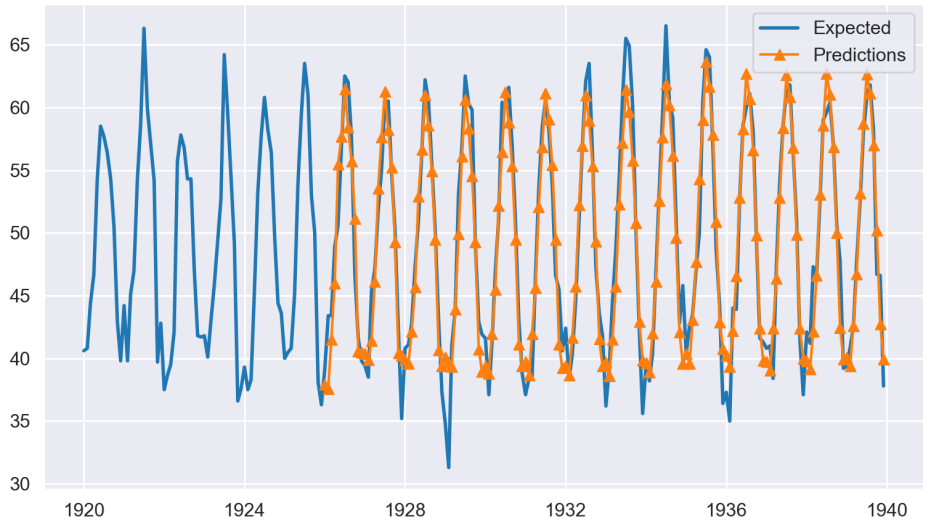

Figure 4.34: GW ETS, 1 step forecast for the temperature dataset

4.7.2 Champagne dataset

For the growing window approach the best parameters obtained after optimization were:

- seasonal_periods: 12
- trend: add
- seasonal: add
- use_boxcox: True
- initialization_method: estimated

The results using these parameters can be seen on fig 4.35 and produce a sMAPE of 13.2952.

4.7.3 Airline dataset

For the growing window approach the best parameters obtained after optimization were:

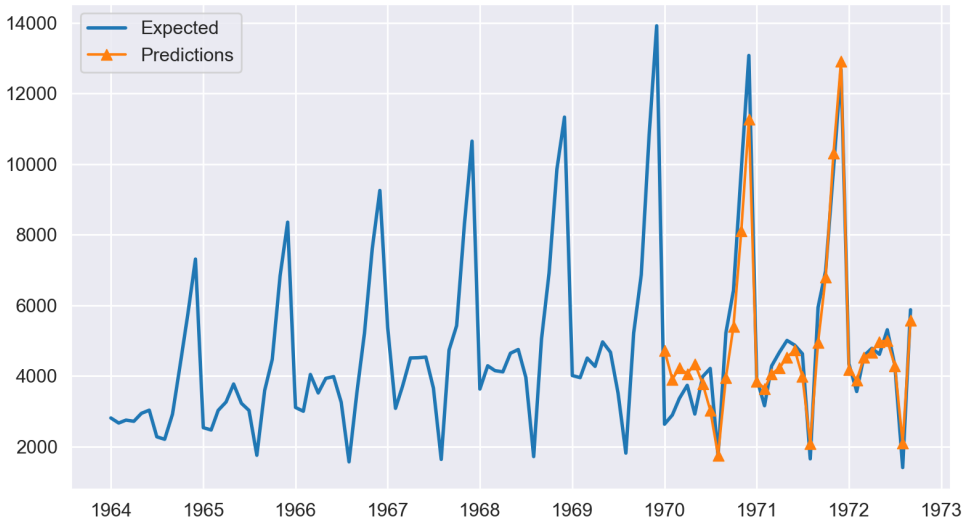

Figure 4.35: GW ETS, 1 step forecast for the champagne dataset

- seasonal_periods: 12
- trend: add
- seasonal: mul
- use_boxcox: False
- initialization_method: estimated

The results using these parameters can be seen on fig 4.36 and produce a sMAPE of 2.3873.

4.8 Performance Metrics

This section will make an overall comparison of the machine learning methods in the different resampling techniques and a comparison of the sMAPE values for all the methods using a growing window approach to demonstrate how competitve the machine learning models are against the classic methods.

4.8.1 Comparison of windows

The table 4.1 compares all the different performance metrics for the machine learning models using the two different resampling techniques for the temperature dataset.

The table 4.2 compares all the different performance metrics for the machine learning models using the two different resampling techniques for the champagne dataset.

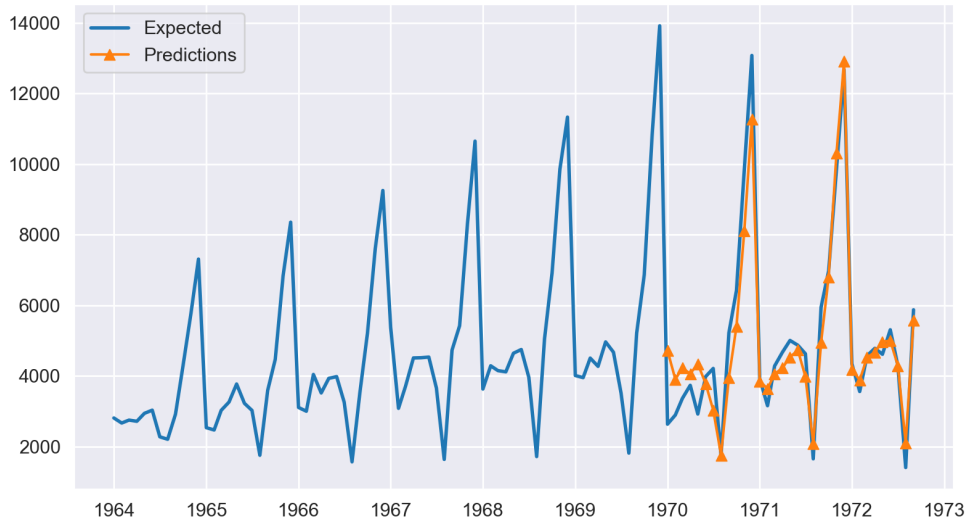

Figure 4.36: GW ETS, 1 step forecast for the airline dataset

	Growing Window				Sliding Window			
Method	MSE	RMSE	MAPE	sMAPE	MSE	RMSE	MAPE	sMAPE
KNN	7.9693	2.8230	4.6338	4.5987	8.5235	2.9195	4.7089	4.6681
LR	8.6711	2.3814	4.0707	4.0316	7.6270	2.7617	4.6621	4.6101
SVM	5.5252	2.3506	4.0551	4.0074	6.5755	2.5643	4.3206	4.2852
NN	5.9164	2.4324	4.1344	4.0890	7.3244	2.7063	4.5476	4.5386
RF	6.2290	2.4958	4.1323	4.1154	7.5123	2.7409	4.5290	4.4935

Table 4.1: Comparison of different performance metrics for the machine learning methods in the different training windows for the temperature dataset

	Growing Window				Sliding Window			
Method	MSE	RMSE	MAPE	sMAPE	MSE	RMSE	MAPE	sMAPE
KNN	415123.18	644.3005	12.8530	12.7018	415123.18	644.3005	12.8230	12.7018
LR	705934.13	840.1989	17.2500	15.1476	614066.23	783.6238	15.9283	14.1902
SVM	1183988.54	1088.11274	18.3341	17.9095	1233892.72	1110.8072	19.0645	19.2189
NN	895685.96	946.4069	17.2025	17.3726	793933.02	891.0291	16.9639	18.5388
RF	756303.04	869.6569	17.3892	15.4078	805036.63	897.2383	18.9366	16.5632

Table 4.2: Comparison of different performance metrics for the machine learning methods in the different training window for the champagne dataset

The table 4.3 compares all the different performance metrics for the machine learning models using the two different resampling techniques for the airline dataset.

	Growing Window				Sliding Window			
Method	MSE	RMSE	MAPE	sMAPE	MSE	RMSE	MAPE	sMAPE
KNN	882.02	29.6988	5.9903	6.1785	882.02	29.9688	5.9903	6.1785
LR	195.58	13.9849	2.7549	2.7637	250.04	15.8126	2.9555	2.9595
SVM	285.87	16.9078	3.4422	3.4597	359.9762	18.9730	4.0754	4.1020
NN	233.1554	14.9384	3.0526	3.0401	303.4408	17.4196	3.6392	3.6686
RF	767.7971	27.7092	5.2775	5.3727	883.5875	29.7251	5.6839	5.7938

Table 4.3: Comparison of different performance metrics for the machine learning methods in the different training window for the airline dataset

4.8.2 Comparison of methods

The table 4.4 compares the sMAPE values for all methods across the 3 different datasets. These results don't have any sort of feature importance experimentation done on them and apply only to the growing window approach, since the sliding window showed poorer results as evidenced by the previous tables 4.1, 4.2 and 4.3. Values in bold represent the best predictions for the classical methods and, if applicable, the values where the machine learning methods outperformed the classic methods.

	Temperature	Champagne	Airline
Method	sMAPE	sMAPE	sMAPE
KNN	4.5987	**12.7018**	6.1785
LR	**4.0315**	15.1476	2.7637
SVM	**4.0074**	17.9095	3.4597
NN	4.0890	17.2025	3.0401
RF	4.1154	15.4078	5.3727
ARIMA	**4.0353**	16.2952	2.6559
ETS	4.0833	**13.8774**	**2.3873**

Table 4.4: Comparison of sMAPE values for all the methods on the 3 datasets using a growing window

4.9 Feature Importance

This section will present the permutation feature importance for the machine learning methods using only the growing window approach.An additional set of experiments with different features obtained through the permutation feature importance plots in an attempt to try and improve the sMAPE errors using the fewest features possible will also be presented.

These experiments would hopefully showcase some interesting insights regarding the features created and how they could be used in combination with other features, what features contributed the most and what features worked best with specific methods. While doing the experiments

different sets of features were tested instead of specific ones, for instance, if the performance of the exponential moving averages were to be analyzed than all the features of this sort would be included in the experiment instead of just 'EMA1' for example. The objective was to describe how that set of features would behave rather than a specific one inside that type so that some conclusions could be taken from the features created and its use by the models. A special note for the random forest algorithm since it's capable of natively exhibit feature importance which will also be presented. This section is organized by the different methods and at the end a table with the updated sMAPE values will be presented. All the experiments were done using the models and the parameters described previously in the growing window approaches.

On the tables containing these experiments values that beat the original model sMAPE will be highlighted. Some useful notation for an easy read of the experiments:

- SI - Seasonal Indices
- Encoding - the whole set of seasonal enconding features ($S_1, S_2, ...S_m$);
- Seasonal_avgs - Seasonal_avg_2 and Seasonal_avg_3;
- seasonal min/max - seasonal_min and seasonal_max;
- SMA's - the whole set of simple moving average features (SMA1, SMA2, SMA3 and SMA_diff);
- EMA's - the whole set of exponential moving average features (EMA1, EMA2, EMA3 and EMA_diff);
- Y_lags = the whole ste of lagged target features (Y_lag_1, Y_lag_12, Y_diff);
- seasonal_stds = Seasonal_std_2 + Seasonal_std_3 + Seasonal_std_diff.

4.9.1 K Nearest Neighbors

The K Nearest Neighbors will not have any experiments done for the airline dataset due to its poor ability to model the target when a trend is present. This inability is highlighted by the plots in figures 4.5 and 4.6.

The permutation feature importance for KNN model used in the experiments of the temperature dataset using a growing window is shown on figure 4.37 After looking at the most important features in 4.37 a series of experiments was done to see if the results could be improved and what features might contribute the most. These experiments done in the temperature dataset can be seen on table 4.5.

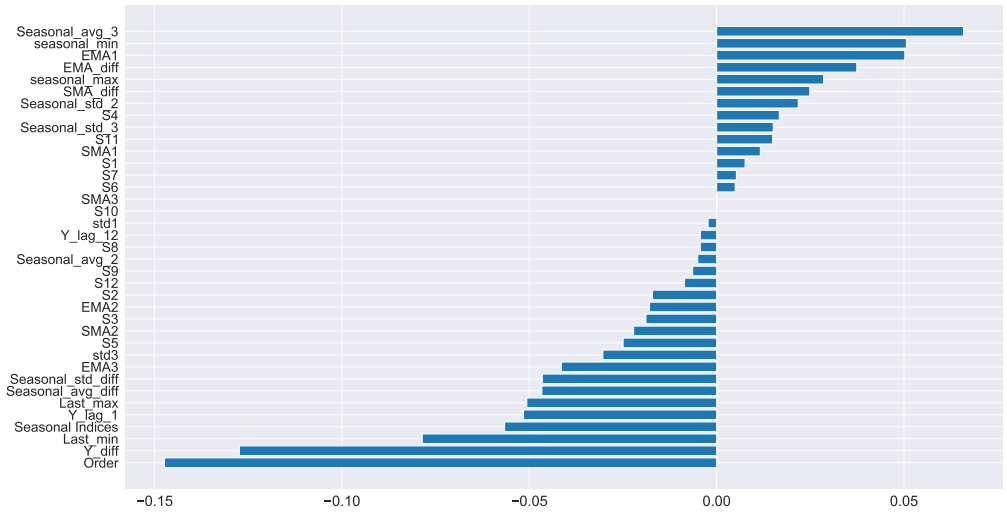

Figure 4.37: KNN permutation feature importance for the temperature dataset

Features	sMAPE
SI	**4.1543**
Encoders	5.2266
Si + Seasonal_avgs	**4.2659**
Seasonal_avgs	**4.2053**
seasonal min/max + Seasonal_avgs	**4.3323**
SI + seasonal min/max + Seasonal_avgs	**4.2638**
Encoding + seasonal min/max + Seasonal_avgs	**4.1387**
Encoding + seasonal min/max + Seasonal_avgs + EMA's	**4.1603**

Table 4.5: Summary of the experiments done with different features for KNN on the temperature dataset

The permutation feature importance for the KNN model used in the experiments of the champagne dataset using a growing window is shown on figure 4.38

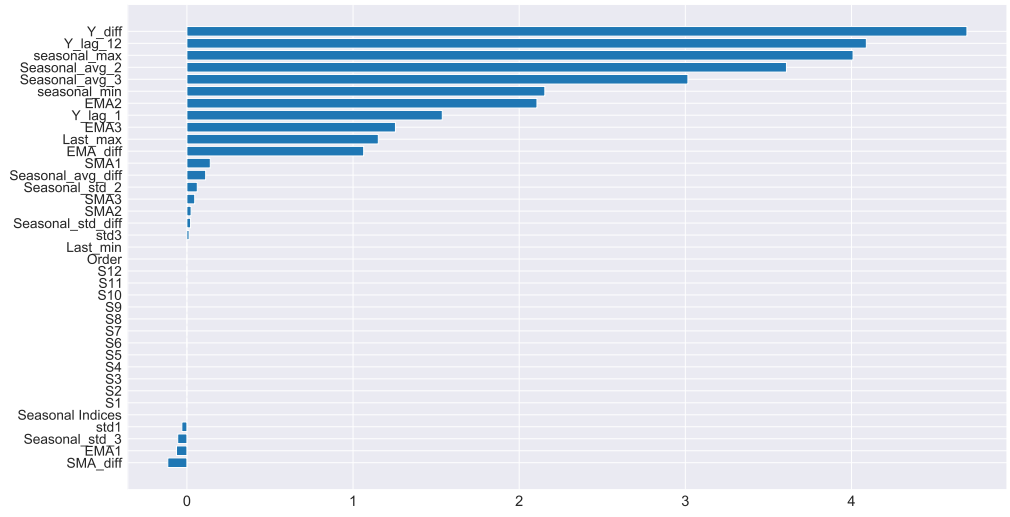

Figure 4.38: KNN permutation feature importance for the champagne dataset

After looking at the most important features in 4.38 a series of experiments was done to see if the results could be improved and what features might contribute the most. These experiments done in the champagne dataset can be seen on table 4.6.

Features	sMAPE
SI + Seasonal_avgs	15.9306
Encoding + seasonal_avgs	15.9305
seasonal min/max + Seasonal_avgs	15.9298
Y_lags	15.3390
Y_lags + Seasonal_avgs	**14.3611**
Y_lags + Seasonal_avgs + seasonal min/max	**13.4130**
Y_lags + Seasonal_avgs + seasonal min/max + EMA's	**12.6377**
Y_lags + Seasonal_avgs + seasonal min/max + EMA's + SI	**12.6377**
Y_lags + Seasonal_avgs + seasonal min/max + EMA's + Encoding	**12.6377**

Table 4.6: Summary of the experiments done with different features for KNN on the champagne dataset

The permutation feature importance for the KNN model used in the experiments of the champagne dataset using a growing window is shown on figure 4.39

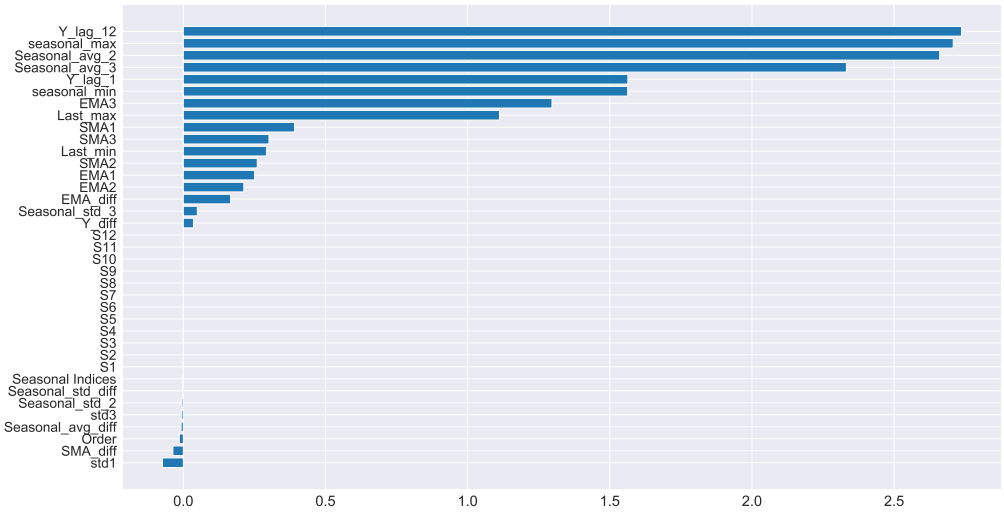

Figure 4.39: KNN permutation feature importance for the airline dataset

4.9.2 Linear Regression

The permutation feature importance for the LR model used in the experiments of the temperature dataset using a growing window is shown on figure 4.40.

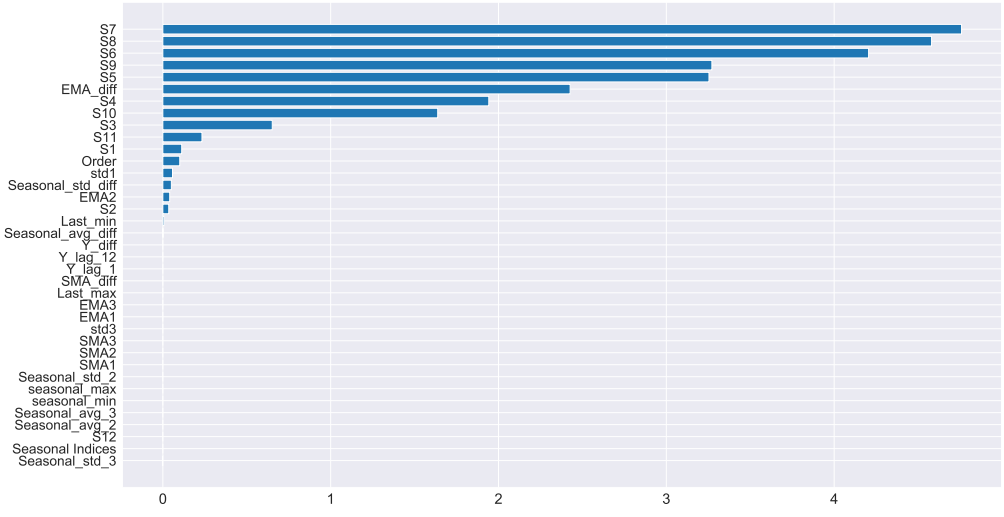

Figure 4.40: LR permutation feature importance for the temperature dataset

After looking at the most important features in 4.40 a series of experiments was done to see if the results could be improved and what features might contribute the most. These experiments done in the temperature dataset can be seen on table 4.7.

Features	sMAPE
SI	14.0448
Encoding	4.0448
Encoding + EMA's	4.0606
Encoding + Seasonal_std_diff	**3.8894**
Encoding + seasonal_stds	**3.9616**
SI + seasonal_stds	15.0774

Table 4.7: Summary of the experiments done with different features for LR on the temperature dataset

The permutation feature importance for the LR model used in the experiments of the champagne dataset using a growing window is shown on figure 4.41.

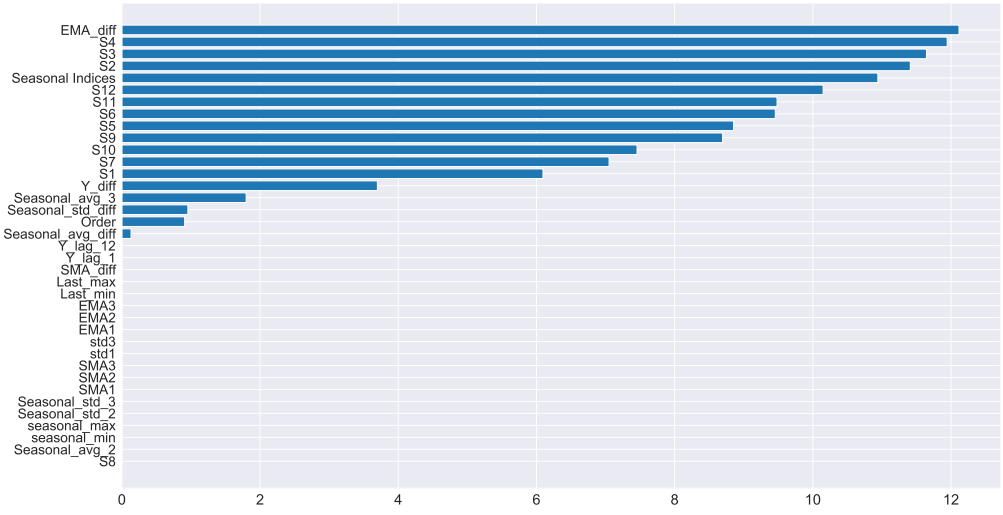

Figure 4.41: LR permutation feature importance for the champagne dataset

After looking at the most important features in 4.41 a series of experiments was done to see if the results could be improved and what features might contribute the most. These experiments done in the champagne dataset can be seen on table 4.8.

Features	sMAPE
SI	53.3560
Encoding	20.7797
Encoding + Seasonal_avgs	**13.8617**
Encoding + seasonal_stds	**11.7206**

Table 4.8: Summary of the experiments done with different features for LR on the champagne dataset

The permutation feature importance for the LR model used in the experiments of the airline dataset using a growing window is shown on figure 4.42.

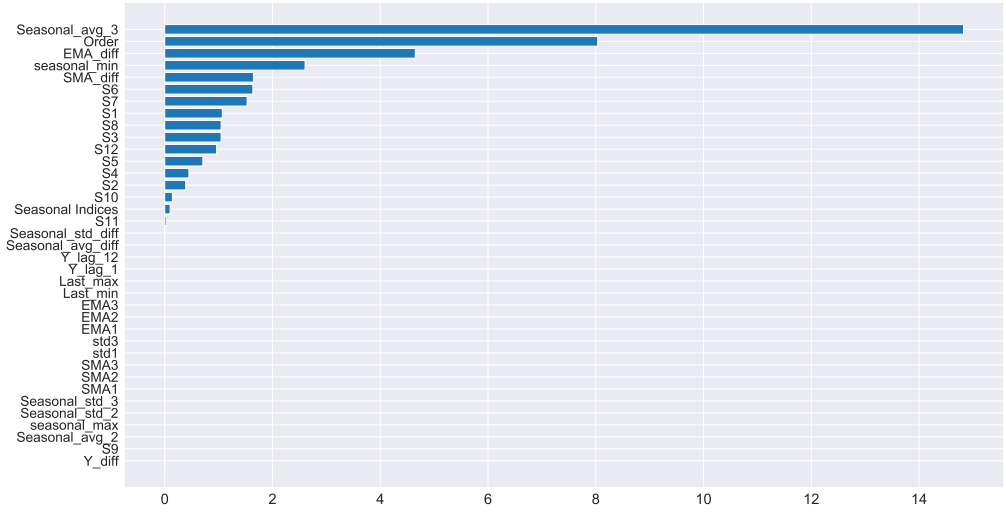

Figure 4.42: LR permutation feature importance for the airline dataset

After looking at the most important features in 4.42 a series of experiments was done to see if the results could be improved and what features might contribute the most. These experiments done in the airline dataset can be seen on table 4.9.

Features	sMAPE
Encoding + Seasonal_avgs	4.3436
Encoding + Seasonal_avgs + Order	4.2550
Encoding + Order + seasonal min/max	4.0014
Encoding + Order + seasonal min/max + EMA's	3.0703
Encoding + Order + seasonal min/max + EMA's + Seasonal_avgs	3.0914
Encoding + Order + seasonal min/max + EMA's + seasonal_stds	3.0838
Encoding + Order + seasonal min/max + EMA's + SMA's + seasonal_stds	**2.6688**
Encoding + Order + seasonal min/max + EMA's + SMA's	**2.6681**
Encoding + Order + seasonal min/max + SMA's	3.1994

Table 4.9: Summary of the experiments done with different features for LR on the airline dataset

4.9.3 Neural Networks

The permutation feature importance for the NN model used in the experiments of the temperature dataset using a growing window is shown on figure 4.43.

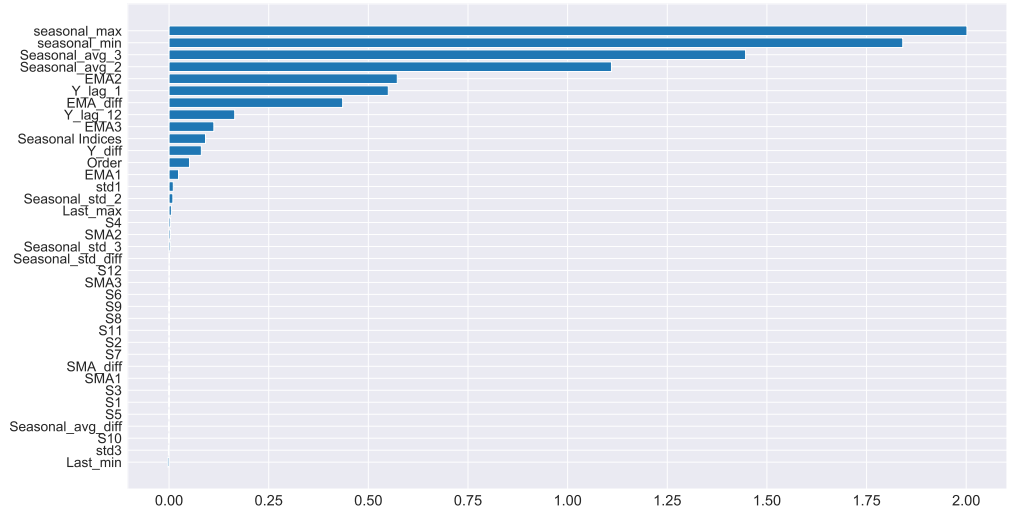

Figure 4.43: NN permutation feature importance for the temperature dataset

After looking at the most important features in 4.43 a series of experiments was done to see if the results could be improved and what features might contribute the most. These experiments done in the temperature dataset can be seen on table 4.10.

Features	sMAPE
SI	14.9603
Encoding	**4.0532**
Encoding + seasonal min/max	4.4015
seasonal min/max	4.1750
seasonal min/max + Seasonal_avgs	4.2156
seasonal min/max + Y_lags	4.2467
seasonal min/max + Y_lags + Seasonal_avgs + EMA's	**4.0336**
Y_lags + Seasonal_avgs + EMA's	**4.0110**
Y_lags + Seasonal_avgs + EMA's + Encoding	**3.9998**

Table 4.10: Summary of the experiments done with different features for NN on the temperature dataset

The permutation feature importance for the NN model used in the experiments of the champagne dataset using a growing window is shown on figure 4.44.

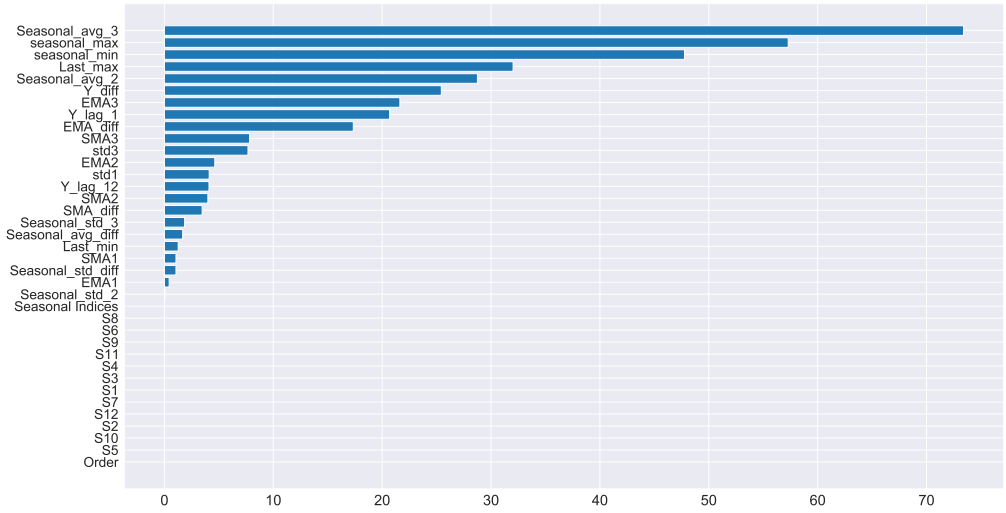

Figure 4.44: NN permutation feature importance for the champagne dataset

After looking at the most important features in 4.44 a series of experiments was done to see if the results could be improved and what features might contribute the most. These experiments done in the champagne dataset can be seen on table 4.11.

Features	sMAPE
Seasonal_avgs + seasonal min/max	17.3741
Encoding + Seasonal_avgs + seasonal min/max	17.8585
SI + Seasonal_avgs + seasonal min/max	18.7393
seasonal min/max + Y_lags + Seasonal_avgs + EMA's	20.1191
seasonal min/max + Y_lags + Seasonal_avgs + SMA's	25.5255
seasonal min/max + Y_lags + Seasonal_avgs + seasonal_stds	**16.7990**
seasonal min/max + Y_lags + Seasonal_avgs + seasonal_stds + Y_lags	20.3099
seasonal min/max + Y_lags + Seasonal_avgs + seasonal_stds + Y_lags + encoders	19.8389
seasonal min/max + Y_lags + Seasonal_avgs + seasonal_stds + Y_lags + Last_min + Last_max	**17.1097**

Table 4.11: Summary of the experiments done with different features for NN on the champagne dataset

The permutation feature importance for the NN model used in the experiments of the airline dataset using a growing window is shown on figure 4.45.

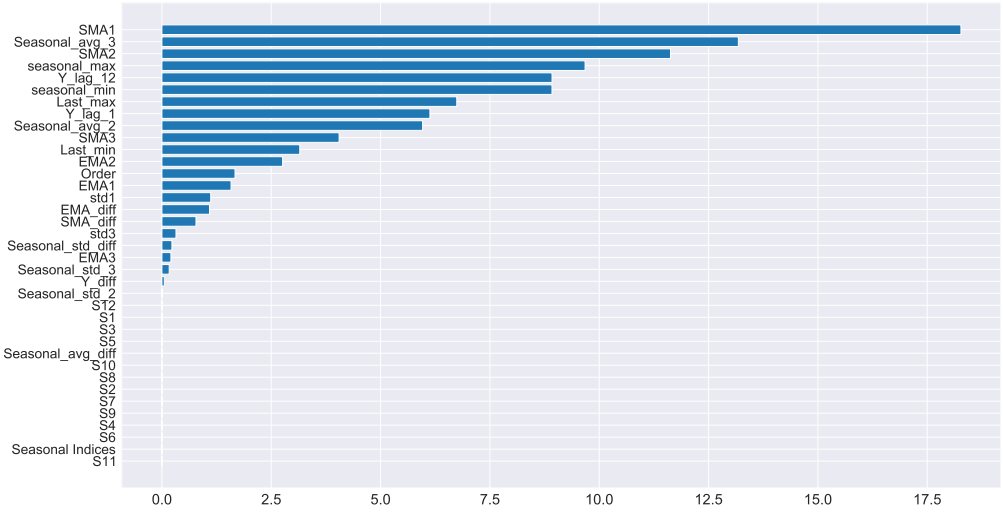

Figure 4.45: NN permutation feature importance for the airline dataset

After looking at the most important features in 4.45 a series of experiments was done to see if the results could be improved and what features might contribute the most. These experiments done in the airline dataset can be seen on table 4.12.

Features	sMAPE
SMA's	13.3718
SMA's + Seasonal_avgs	3.6224
Seasonal_avgs	4.4015
seasonal min/max + Seasonal_avgs	3.5956
seasonal min/max + Seasonal_avgs + SMA's	**3.0225**
Seasonal_avgs + SMA's + Order	3.0616
Seasonal_avgs + SMA's + Y_lags	**2.7846**

Table 4.12: Summary of the experiments done with different features for NN on the airline dataset

4.9.4 Random Forest

For the Random Forest models no experiments were done in the airline dataset due to its inability to correctly model the target when tere is a trend, as evidenced by figures 4.23 and 4.24. This is also noted in the work [3].

The permutation feature importance for the RF model used in the experiments of the temperature dataset using a growing window is shown on figure 4.46.

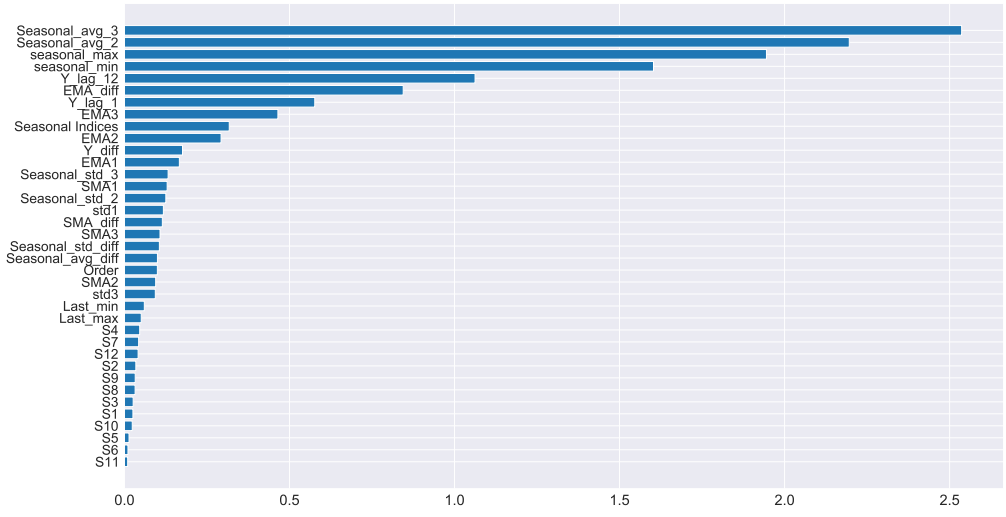

Figure 4.46: RF permutation feature importance for the temperature dataset

The native important features given by the Random Forest for the RF model used in the experiments of the temperature dataset using a growing window can be seen in figure 4.47.

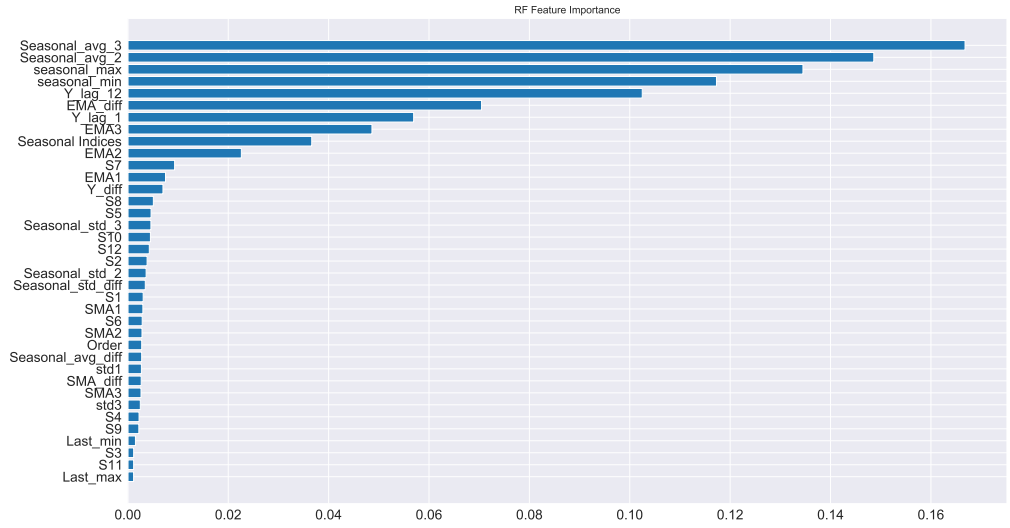

Figure 4.47: RF feature importance for the temperature dataset

After looking at the most important features in 4.46 and 4.47 a series of experiments was done to see if the results could be improved and what features might contribute the most. These experiments done in the temperature dataset can be seen on table 4.13.

Features	sMAPE
SI + Seasonal_avgs + EMA's	**4.07**
Encoding + Seasonal_avgs + seasonal min/max	**4.01**
Encoding + Seasonal_avgs + Seasonal_avg_diff seasonal min/max + Y_lags	**3.97**

Table 4.13: Summary of the experiments done with different features for RF on the temperature dataset

The permutation feature importance for the RF model used in the experiments of the champagne dataset using a growing window is shown on figure 4.48.

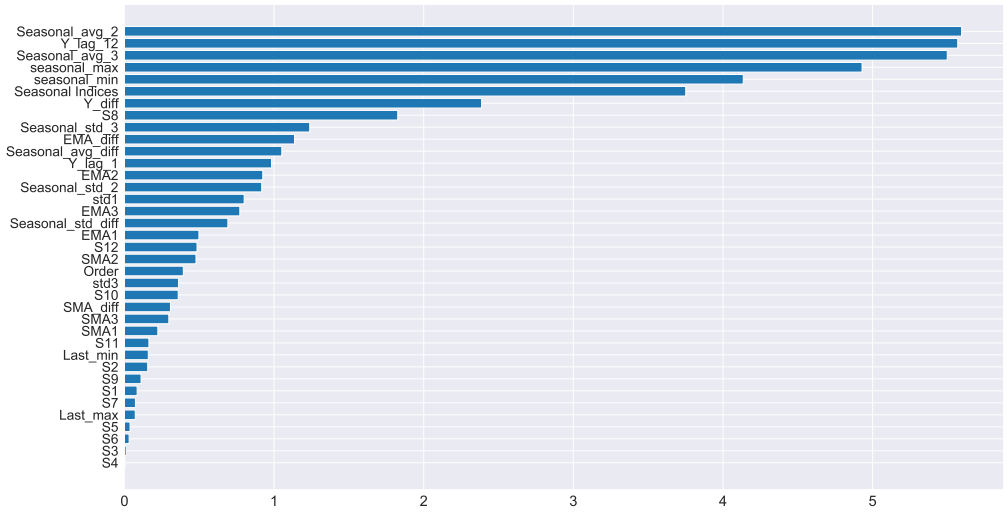

Figure 4.48: RF permutation feature importance for the champagne dataset

The native important features given by the Random Forest for the RF model used in the experiments of the champagne dataset using a growing window can be seen in figure 4.49.

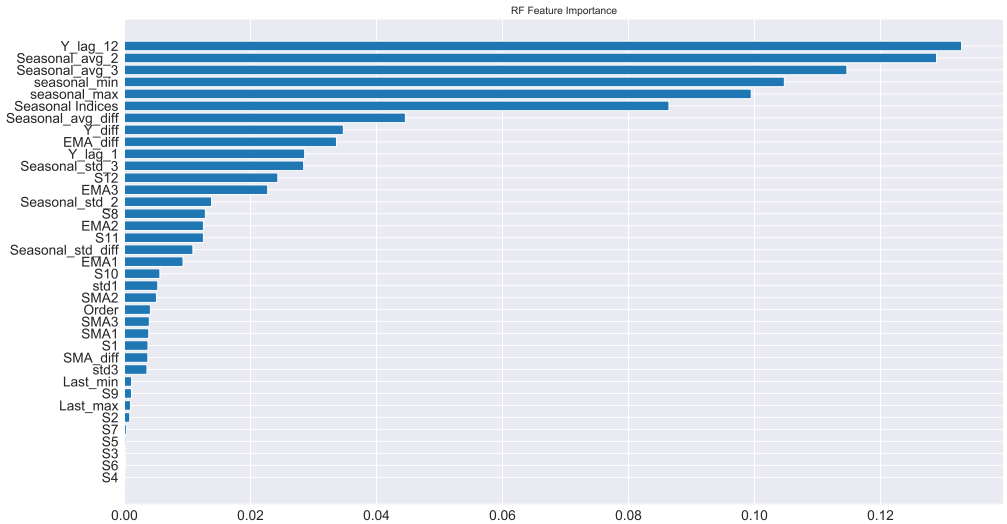

Figure 4.49: RF feature importance for the champagne dataset

After looking at the most important features in 4.48 and 4.49 a series of experiments was done to see if the results could be improved and what features might contribute the most. These experiments done in the champagne dataset can be seen on table 4.14.

Features	sMAPE
SI + Seasonal_avgs	**12.25**
Encoding + Seasonal_avgs	**12.44**
Encoding + Seasonal_avgs + seasonal min/max	**13.50**
SI + Seasonal_avgs + seasonal min/max	**13.38**
SI + Y_lags	**12.21**
Encoding + Y_lags	**13.10**
Y_lags	**15.26**
SI + Y_lags + Seasonal_avgs	**13.58**
Encoding + Y_lags + Seasonal_avgs	**14.03**

Table 4.14: Summary of the experiments done with different features for RF on the champagne dataset

The permutation feature importance for the RF model used in the experiments of the airline dataset using a growing window is shown on figure 4.50.

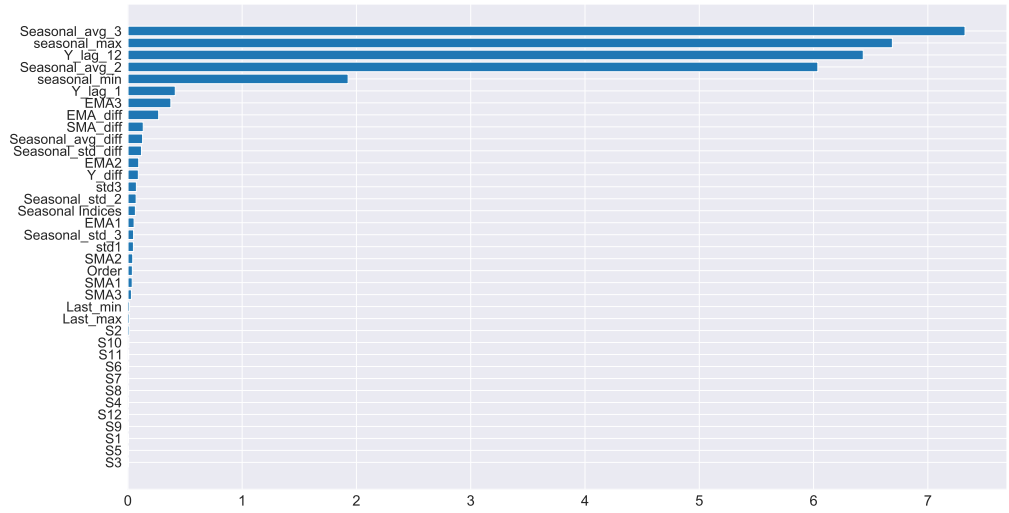

Figure 4.50: RF permutation feature importance for the airline dataset

The native important features given by the Random Forest for the RF model used in the experiments of the airline dataset using a growing window can be seen in figure 4.51.

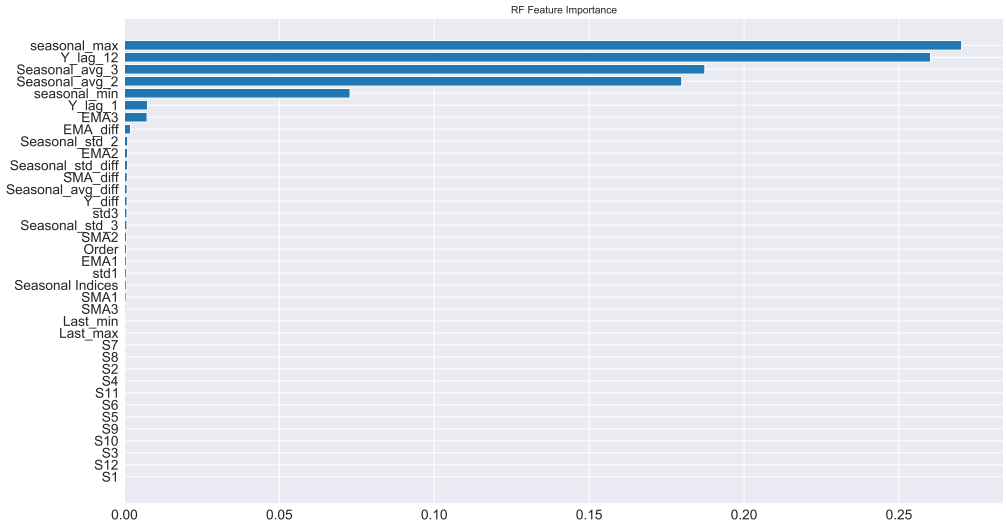

Figure 4.51: RF feature importance for the airline dataset

4.9.5 Support Vector Machine

The permutation feature importance for the SVM model used in the experiments of the temperature dataset using a growing window is shown on figure 4.52.

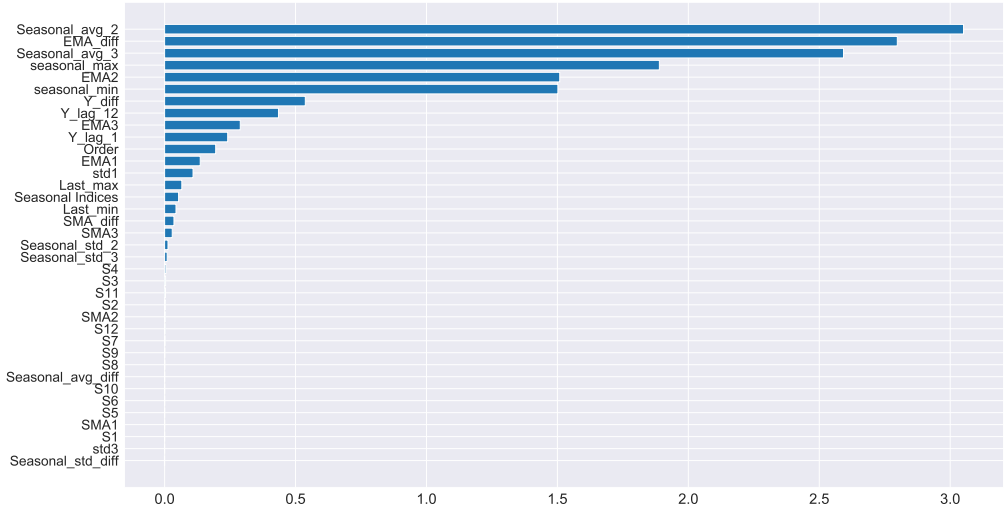

Figure 4.52: SVM permutation feature importance for the temperature dataset

After looking at the most important features in 4.52 a series of experiments was done to see if the results could be improved and what features might contribute the most. These experiments done in the temperature dataset can be seen on table 4.15.

Features	sMAPE
SI	15.40
Encoding	15.60
Seasonal_avgs	4.081
seasonal min/max	4.098
Seasonal_avgs + seasonal min/max	4.052
Y_lags	5.35
Seasonal_avgs + seasonal min/max + Y_lags	4.083
Seasonal_avgs + seasonal min/max + EMA's	**3.935**
EMA' + seasonal min/max	**3.917**
EMA' + Seasonal_avgs	**3.946**
SMA' + seasonal min/maxs	4.096
SMA' + Seasonal_avgs	4.072
SMA' + Seasonal_avgs + seasonal min/max + Y_lags	**3.917**

Table 4.15: Summary of the experiments done with different features for SVM on the temperature dataset

The permutation feature importance for the SVM model used in the experiments of the champagne dataset using a growing window is shown on figure 4.53.

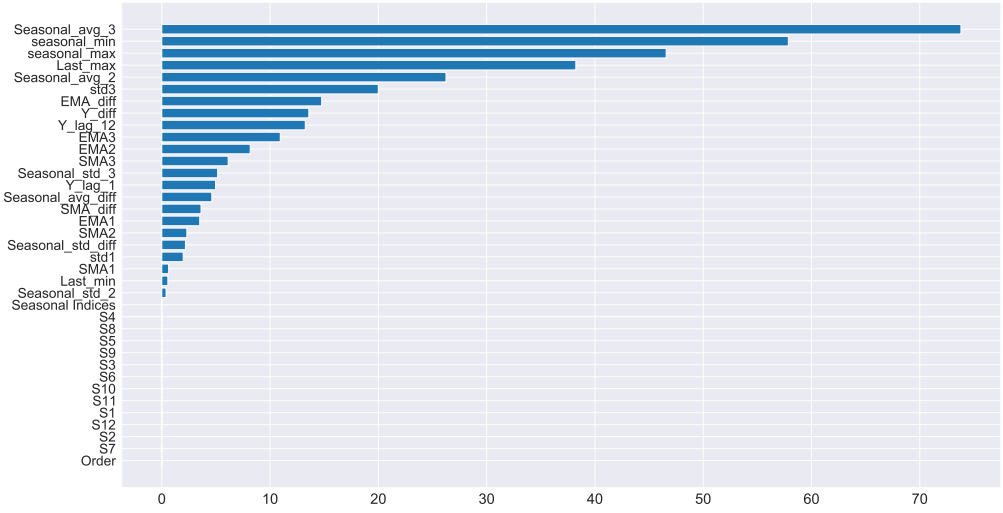

Figure 4.53: SVM permutation feature importance for the champagne dataset

After looking at the most important features in 4.53 a series of experiments was done to see if the results could be improved and what features might contribute the most. These experiments done in the champagne dataset can be seen on table 4.16.

Features	sMAPE
Seasonal_avgs + seasonal min/max	18.5422
Seasonal_avgs + seasonal min/max + EMA's	18.5161
Seasonal_avgs + seasonal min/max + EMA's + Y_lags	20.4093
Seasonal_avgs + seasonal min/max + Y_lags	18.9483
Seasonal_avgs + seasonal min/max + Y_lags + Last_min + Last_max	20.9784
Seasonal_avgs + seasonal min/max + Y_lags + seasonal_stds	18.0574
Seasonal_avgs + seasonal min/max + Y_lags + seasonal_stds + EMA's	18.3030
ALL features except SI and Encoding	**17.8987**

Table 4.16: Summary of the experiments done with different features for SVM on the champagne dataset

The permutation feature importance for the SVM model used in the experiments of the airline dataset using a growing window is shown on figure 4.54.

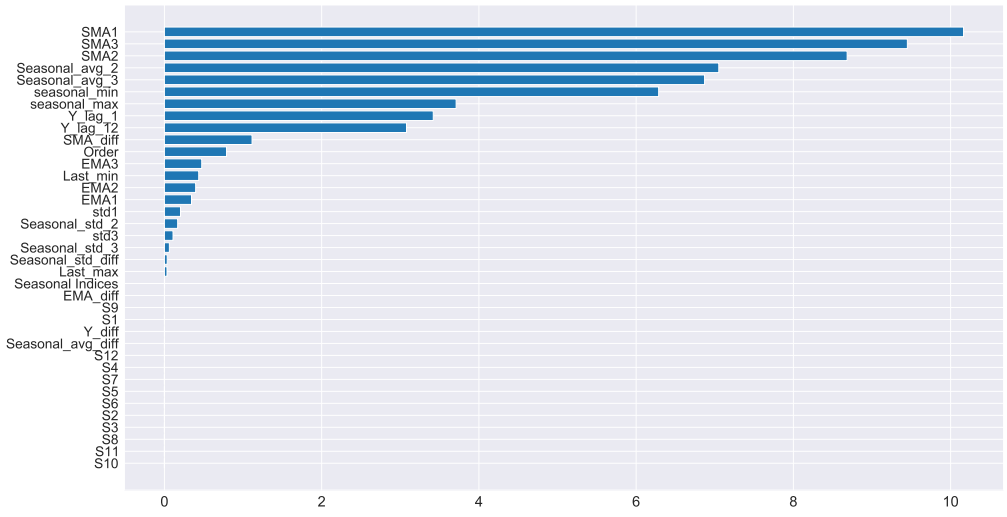

Figure 4.54: SVM permutation feature importance for the airline dataset

After looking at the most important features in 4.54 a series of experiments was done to see if the results could be improved and what features might contribute the most. These experiments done in the airline dataset can be seen on table 4.17.

Features	sMAPE
SMA's	12.5937
SMA's + Seasonal_avgs	**3.2582**
SMA's + Seasonal_avgs + seasonal min/max	**3.1165**
SMA's + Seasonal_avgs + seasonal min/max + Y_lags	**3.1185**
EMA's + Seasonal_avgs + seasonal min/max	3.8866
SMA's + Seasonal_avgs + seasonal min/max + seasonal_stds	**3.2420**
SMA's + Seasonal_avgs + seasonal min/max + Order	**3.1070**

Table 4.17: Summary of the experiments done with different features for SVM on the airline dataset

4.9.6 Updated sMAPE comparison

The following table 4.18 contains a comparison of the machine learning models against the classical methods with updated sMAPE values after feature analysis. Values that outperform the classic methods are highlighted in bold.

Method	Temperature	Champagne	Airline
KNN	4.1387	**12.6377**	6.1785
LR	**3.8894**	**11.7206**	2.6681
NN	**3.9998**	16.7990	2.7846
RF	**3.97**	**12.2106**	5.3484
SVM	**3.9171**	17.8987	3.1070
ARIMA	4.0353	16.2952	2.6559
ETS	4.0833	13.8774	2.3873

Table 4.18: Comparison of all the methods for all datasets with the updated sMAPE values

4.10 Discussion

This section will interpret and discuss the results presented in the previous sections.

4.10.1 Resampling Method

The interpretations of the resampling method are quite straight forward. A simple look at tables 4.1, 4.2 and 4.3 reveals best results for the majority of the growing window values. Hence, feature analysis was only performed using this approach. It is also interesting to highlight that using the growing window method allows the use of more observations by the models. The literature doesn't offer a general answer to what is the best method so the findings in this dissertation, although limited, showed interesting results for the growing window approach.

4.10.2 K Nearest Neighbors

Overall the K Nearest Neighbors performed well. The only instance where this model fails is on the airline dataset where it can't beat the naive forecast, as seen on figures 4.5 and 4.6. For this reason no feature analysis was done in the airline dataset since the method showed the inability to model the trend. For the other two datasets the forecasts are good as figures 4.1- 4.4 evidence and KNN outperforms the classical methods on the champagne dataset, a rather irregular one, as proved by the table 4.18.

For the hyperparameters no general conclusions can me made. The optimization proved its worth since the resulting KNN models have competitive forecasts in the temperature and champagne datasets. A note can be taken, however, on the preference (4 out of 6 times) for the manhattan distance.

Regarding feature analysis the main takeaway is that the K Nearest Neighbors algorithm performed well when using just the most important variables and overall there is a good number of experiments where the original sMAPE value were reduced. Even though the strategies defined by the encoding features and the seasonal indices can be utilized they are not the differencing factor. The features that worked best with this method were: 'Seasonal_avg_2', 'Seasonal_avg_3', 'seasonal_min' and 'seasonal_max' since they were included in the best sMAPE values in both the temperature and champagne datasets experiments.

The KNN then proved its worth and showed that it can be a viable option for the forecaster, especially considering that it is a simple method and worked with few features. For trend modeling the proposed framework didn't work with this algorithm hinting at a possible transformation (like a Box-Cox for example) for better results.

4.10.3 Linear Regression

Linear Regression is one of the oldest and simplest methods available and surprisingly it has the overall best results in this work. In every figure from 4.7 up to 4.12 this method consistently beats the naive forecasts and presents good estimations and outperforms the classical methods on the temprature and champagne dataset as proved by table 4.18.

For the hyperparameters the clearest conclusion is that all models performed best when using positive coefficients and it was also the method with the easiest parameter tuning.

Regarding feature analysis Linear Regression shows a clear preference for the encoding features and it is also the only method that achieved good results with some of standard deviation features, most notably the 'Seasonal_std_2' and 'Seasonal_std_3'. Linear Regression also showed good and reduced values for sMAPE when using fewer features.

Like the KNN, Linear Regression proved that despite being a simple method it can very useful and deserves to be included in future studies. In the present dissertation it performed the best (outperformed the classic methods in 2 of the 3 datasets) and was one of the few methods that could model the trend (even though it doesn't beat ETS the sMAPE is competitive).

4.10.3.1 Neural Networks

Neural Networks are one of the most popular methods right now and in this work it also achieved good results. On the plots 4.13 through 4.18 this method consistently shows good forecasts beating the naive forecasts. This also evidenced by the table 4.4 where competitive results are achieved and also on table 4.18 after the feature analysis, outperforming the classical methods in the temperature dataset and offering competitive forecasts for the other two datasets.

For the hyperparameters it is worth noticing the best results were achieved using the optimizer in the family of quasi-Newton methods 'lbgfs' for the solver and the dominance of the 'identity' in the activation function.

Feature analysis and tuning proved to be more challenging for this method as evidenced by the tables 4.10, 4.11 and 4.12 where not many values in bold are seen. The strategies 'Seasonal Indices' and the seasonal encoding features don't seem to be of much use for this algorithm and the features that worked the best were: 'Seasonal_avg_2', 'Seasonal_avg_3', 'seasonal_min', 'seasonal_max' and the lagged target features. Neural networks also showed good performance even when increasing the number of features, contrary to LR and KNN who performed better using fewer features. Neural networks also performed good when using moving averages features and using the Simple Moving Average features for the best prediction in the airline dataset hints at the possibility of using such features in the modeling of a trend.

4.10.4 Random Forest

Similarly to the K Nearest Neighbors method, Random Forest showed competitive forecasts for the temperature and champagne datasets, but failing to model the trend as figures 4.23 and 4.24 show where the naive forecast outperforms this method. For this reason no feature analysis was done in the airline dataset.

Regarding hyperparameter optimization Random Forests offer no general conclusion except for the good use of smaller 'min_samples_leaf' and 'min_samples_split'.

Random Forests offer native feature importance and in all datasets the resuls matched with the ones from the permutation feature importance method. The results for the experiments are also interesting since most of the experiments using fewer features improved the initial forecasts. Random Forests showed preference for the lagged target features and for the features 'Seasonal_avg_2', 'Seasonal_avg_3', 'seasonal_min', 'seasonal_max'.

This method is on par with Linear Regression in the sense that they are the only methods that outperform the ARIMA and the ETS in 2 of the 3 datasets, failing only for the airline dataset where a trend is present, which makes this method attractive to use. For trend modeling, as suggested in the KNN discussion, some detrending technique could be of use to improve this method.

4.10.5 Support Vector Machine

Support Vector Machine offered good predictions as seen on figures 4.25 - 4.30 however it only outperformed the classic methods on the temperature dataset, leaving SVM in the last place of the

studied methods.

For the hyperparameters Support Vector Machines showed preference for the linear kernel and very low regularization values for the parameter C.

Regarding feature analysis Support Vector Machine proved to be the most difficult to work with as demonstrated by the values in tables 4.15, 4.16 and 4.17, where the sMAPE improvment in comparison to the original values was little. Support Vector Machines also seem to favour a larger amount of features as highlighted by the best result in table 4.16 where all features but for 3 were necessary.

This method then showed promise to be utilized in this type of problems however more investigation will need to be done when comparing with the other methods that seem to more ready to be used by this framework as is.

Chapter 5

Conclusions

The main goal of this dissertation was to build an automated framework conversion from a time series problem to a regression one. To achieve that the extraction of features was performed and different methods were tested against the classical ones. On the temprature dataset 4 out of 5 methods outperform ARIMA and ETS and on the champagne dataset 3 out of 5 methods outperform ARIMA and ETS. The results of the proposed framework only fail to outperform the classic methods in the airline dataset (although competitive values were achieved) a problem that exhibits a trend. Since the main goal of this work was to tackle the seasonality the objective is reached.

As far as the resampling method used no definite answer is provided in the literature. The work presented in this dissertation tried to shine some light on that subject by testing the two approaches and the results seem to favour the growing window method.

As far as the machine learning methods tested using the proposed framework and its features the overall result is satisfactory since the sMAPE values are competitive with the classic time series methods and in some instances they even outperform such methods. The sMAPE metric was used due to its popularity in the literature but must not be taken as a definite rule in this work but rather a tool for hyperparamter optimization and performance comparison. Even though at the end of optimization the methods were already competitive enough, some feature analysis and tweaking further improved the results indicating that the framework and its proposition has even more space to evolve.

The K Nearest Neighbors algorithm proved its worth despite its simplicity, outperforming the classic methods in the champagne dataset, a rather irregular one, and coming up with a competitive forecast for the temperature dataset. The only instance where it is not competitive is in the airline dataset showing the incapability to model a trend. For the features the KNN showed a preference in combining the seasonal features with other ones. In one case the seasonal encoding features were used paired with seasonal averages and on another the pair of seasonal indices, seasonal max/min and exponential moving averages seemed to be the best approach.

Linear regression turned out to be the best method since it outperformed the classic methods in two datasets and, while failing to beat the airline dataset, it has the best sMAPE value amongst

all the other machine learning algorithms for that dataset showing that it can also be used for problems that have a trend, For linear regression the main takeaway is the superior performance using the encoding variables and it is also the only algorithm that consistently seem to find use for any of the standard deviation features.

Neural networks were able to outperform the classic methods in the temperature dataset and it offers the second best forecast for the airline dataset, where a trend is present.. Neural networks don't find necessity to pair either the 'Seasonal Indices' or the seasonal encoding features, letting the other variables model the seasonality. Also of notice is a good combination with the lagged target features, the seasonal averages and the seasonal min/max ones.

Random Forests also performed well and similarly to the KNN they are not capable to model a trend, as also noted in [3]. They are on par with Linear Rgeression in the sense that they outperform the classical methods in two of the three datasets. This method seem to favor the lagged target variables but also showcased better performance when pairing features with either the seasonal indices or the encoding features to better model the seasonality of the data.

Support Vector Machines may have worse results than other methods but in the end they also present competitive forecasts and even outperform the classic methods on the temperature dataset. They were the most difficult to work with in feature analysis and didn't offer much improvement of sMAPE values.

Overall the features extracted worked well and as expected different methods will prefer different features, making generalizations a bit difficult to achieve. The highlights are the seasonal indices and the seasonal encoding features that proved to be important for the methods and also to be paired with other features, specially 'Seasonal_avg_2', 'Seasonal_avg_3', 'seasonal_min' and 'seasonal_max' since they were consistently marked as important. Meanwhile the features 'Order', any of the standard deviation ones, 'Last_min' and 'Last_max' were not of much use.

Analyzing these results it seems clear that the proposed framework found success and the features contributed well to solve time series problems were the seasonality property is present which raises a very interesting discussion: if feature extraction of this sort has good results this could be a weight that tips the scale in favour of regression frameworks for time series problems. Regression frameworks have complexities and intricacies that the classic time series methods do not possess, making them very attractive to use. This desire is even amplified by the popularity of such methods, their easy use and when taking into account features like the ones proposed in this dissertation and its easy interpretability. Furthermore, the seasonality problem was tackled and solved through the use of the features created and without the need of any transformation technique like the literature suggests.

The machine learning methods selected to be in this work seem to justify their presence as demonstrated by the results obtained. The answer to this dissertation's initial question was provided with the aid of such methods however they only represent a small fraction of the machine learning family. In that sense the hyperparameter optimization that was performed also leaves the feeling of accomplishment but the complexity of the methods allow for even more tuning.

5.1 Future Work

This dissertation tackled a very complex and nuanced problem. The main goal was achieved but small steps through this whole process deserve more thorough investigation of their own. Taking performance metrics for example the sMAPE error was chosen as the main performance metric but other options in the field exist and might be better depending on the method applied. By opening the field play in solving time series problems with regression frameworks the opportunities for this type of research increase. Although the results were good this framework is not without its limitations. The main takeaway is that it begs for more tests with different datasets. Since the proposed framework seems to be paving the way, more tests are needed in more complex datasets, in situations where the data is not monthly or even has multiple seasonalities. It would also be interesting to test with forecast horizons bigger than one.

This dissertation made use of a diverse family of machine learning methods but this is only a gentle brush in what is a fast expanding area with a vast array of different options, making a case for it to be tested using more methods. The present work only tuned general parameters leaving space for hyperparameter optimization to operate even further. In a similar vein, feature importance was also performed but a case for a more in depth study of these features and especially how well they work with what methods can be made.

The main objective was to model seasonality and that goal was achieved making trend modeling the next logical step to take. Despite not being the primary objective of the framework some methods seem to perform well even in the presence of a trend hinting at a possibility of using some features in this work for trend problems like 'Seasonal_avg_2', 'Seasonal_avg_3' and the moving average features. It would be also interesting to see how the framework would behave if a detrending operation was performed like the Box-Cox method mentioned in chapter 2.

References

[1] Steven Wheelwright, Spyros Makridakis, and Rob J Hyndman. *Forecasting: methods and applications*. John Wiley & Sons, 1998.

[2] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[3] Uwe Pritzsche. *Benchmarking of classical and machine-learning algorithms (with special emphasis on bagging and boosting approaches) for time series forecasting*. PhD thesis, 2015.

[4] Adam Aboode. Anomaly detection in time series data based on holt-winters method, 2018.

[5] Tom Michael Mitchell. *The discipline of machine learning*, volume 9. 2006.

[6] Goce Ristanoski, Wei Liu, and James Bailey. Time series forecasting using distribution enhanced linear regression. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 484–495. Springer, 2013.

[7] Max Kuhn, Kjell Johnson, et al. *Applied predictive modeling*, volume 26. Springer, 2013.

[8] Sven F Crone, Michele Hibon, and Konstantinos Nikolopoulos. Advances in forecasting with neural networks? empirical evidence from the nn3 competition on time series prediction. *International Journal of forecasting*, 27(3):635–660, 2011.

[9] Spyros Makridakis and Michele Hibon. The m3-competition: results, conclusions and implications. *International journal of forecasting*, 16(4):451–476, 2000.

[10] Alicia Troncoso Lora, José C Riquelme, José Luís Martínez Ramos, Jesús M Riquelme Santos, and Antonio Gómez Expósito. Influence of knn-based load forecasting errors on optimal energy production. In *Portuguese Conference on Artificial Intelligence*, pages 189–203. Springer, 2003.

[11] Francisco Martínez, María Pilar Frías, María Dolores Pérez, and Antonio Jesús Rivera. A methodology for applying k-nearest neighbor to time series forecasting. *Artificial Intelligence Review*, 52(3), 2019.

[12] Weizhong Yan. Toward automatic time-series forecasting using neural networks. *IEEE transactions on neural networks and learning systems*, 23(7):1028–1039, 2012.

[13] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. Stl: A seasonal-trend decomposition. *Journal of official statistics*, 6(1):3–73, 1990.

[14] Guoqiang Zhang, B Eddy Patuwo, and Michael Y Hu. Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1):35–62, 1998.

[15] Zafeirios Fountas et al. Spiking neural networks for human-like avatar control in a simulated environment. *Computing Science of Imperial College London*, 2011.

[16] Bjoern Krollner, Bruce J Vanstone, and Gavin R Finnie. Financial time series forecasting with machine learning techniques: a survey. In *ESANN*, 2010.

[17] Fernando Mateo, Juan José Carrasco, Abderrahim Sellami, Mónica Millán-Giraldo, Manuel Domínguez, and Emilio Soria-Olivas. Machine learning methods to forecast temperature in buildings. *Expert Systems with Applications*, 40(4):1061–1068, 2013.

[18] G Peter Zhang and Min Qi. Neural network forecasting for seasonal and trend time series. *European journal of operational research*, 160(2):501–514, 2005.

[19] Michael J Kane, Natalie Price, Matthew Scotch, and Peter Rabinowitz. Comparison of arima and random forest time series models for prediction of avian influenza h5n1 outbreaks. *BMC bioinformatics*, 15(1):1–9, 2014.

[20] Junfei Chen, Ming Li, and Weiguang Wang. Statistical uncertainty estimation using random forests and its application to drought forecast. *Mathematical Problems in Engineering*, 2012, 2012.

[21] Hristos Tyralis and Georgia Papacharalampous. Variable selection in time series forecasting using random forests. *Algorithms*, 10(4):114, 2017.

[22] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression, 2004.

[23] UVBR Thissen, R Van Brakel, AP De Weijer, WJ Melssen, and LMC Buydens. Using support vector machines for time series prediction. *Chemometrics and intelligent laboratory systems*, 69(1-2):35–49, 2003.

[24] Karin Kandananond. A comparison of various forecasting methods for autocorrelated time series. *International Journal of Engineering Business Management*, 4:4, 2012.

[25] Tomislav Horvat, Ladislav Havaš, and Dunja Srpak. The impact of selecting a validation method in machine learning on predicting basketball game outcomes. *Symmetry*, 12(3):431, 2020.

[26] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021.

[27] Noah Aggeborn Leander. Forcasting the daily air temperature in uppsala using univariate time series, 2020.

[28] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

[29] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[31] Wilpen L Gorr, Daniel Nagin, and Janusz Szczypula. Comparative study of artificial neural network and statistical models for predicting student grade point averages. *International Journal of Forecasting*, 10(1):17–34, 1994.

[32] Zaiyong Tang and Paul A Fishwick. Feedforward neural nets as models for time series forecasting. *ORSA journal on computing*, 5(4):374–385, 1993.

[33] Gecynalda S da S Gomes, Teresa B Ludermir, and Leyla MMR Lima. Comparison of new activation functions in neural network for forecasting financial time series. *Neural Computing and Applications*, 20(3):417–439, 2011.

[34] Thais Mayumi Oshiro, Pedro Santoro Perez, and José Augusto Baranauskas. How many trees in a random forest? In *International workshop on machine learning and data mining in pattern recognition*, pages 154–168. Springer, 2012.

[35] Philipp Probst and Anne-Laure Boulesteix. To tune or not to tune the number of trees in random forest. *J. Mach. Learn. Res.*, 18(1):6673–6690, 2017.

[36] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016.

[37] Ramón Díaz-Uriarte and Sara Alvarez De Andres. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):1–13, 2006.

[38] Christoph Molnar. *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/.

[39] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20(177):1–81, 2019.

[40] Taylor G. Smith et al. pmdarima: Arima estimators for Python, 2017–. [Online; accessed <today>]. URL: http://www.alkaline-ml.com/pmdarima.

[41] Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.