A COMPARATIVE STUDY OF CLASSICAL AND MACHINE LEARNING APPROACHES FOR TIME SERIES FORECASTING: AN EMPIRICAL ANALYSIS ON EXPORTS IN TURKEY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EDA GÜNEL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
STATISTICS

JULY 2020

Approval of the thesis:

A COMPARATIVE STUDY OF CLASSICAL AND MACHINE LEARNING APPROACHES FOR TIME SERIES FORECASTING: AN EMPIRICAL ANALYSIS ON EXPORTS IN TURKEY

submitted by **EDA GÜNEL** in partial fulfillment of the requirements for the degree of **Master of Science in Statistics , Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Ayşen Dener Akkaya
Head of Department, **Statistics** _____

Assoc. Prof. Dr. Ceylan Yozgatlıgil
Supervisor, **Statistics, METU** _____

Examining Committee Members:

Prof. Dr. Özlem İlk Dağ
Statistics, METU _____

Assoc. Prof. Dr. Ceylan Yozgatlıgil
Statistics, METU _____

Prof. Dr. Seher Nur Sülkü
Econometrics, Hacı Bayram Veli University _____

Date: 16.07.2020

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Eda Günel

Signature :

ABSTRACT

A COMPARATIVE STUDY OF CLASSICAL AND MACHINE LEARNING APPROACHES FOR TIME SERIES FORECASTING: AN EMPIRICAL ANALYSIS ON EXPORTS IN TURKEY

Günel, Eda
M.S., Department of Statistics
Supervisor: Assoc. Prof. Dr. Ceylan Yozgatlıgil

July 2020, 74 pages

Exports has become one of the main economic indicator for countries. Accordingly, an accurate forecasting for exports is an important step for decision making and finding the most appropriate forecasting model constitutes the main subject of many studies. By taking the popularity and success of the machine learning (ML) methods on time series forecasting tasks into consideration, they are utilized also in this study to observe their predictive performances on Turkish exports. In this respect, Long Short Term Memory (LSTM), Support Vector Machines (SVM) and Random Forest (RF) are applied and the results are compared with the most commonly used classical time series models such as Autoregressive Integrated Moving Average (ARIMA) and Exponential Smoothing (ETS) models. The analysis is conducted on Turkish monthly exports data taken from Turkish Statistical Institute (TURKSTAT) within the time interval of January 1997 – September 2019 and the main steps of the analysis are anomaly detection and cleaning, data preprocessing, model development, hyperparameter tuning and model selection and model comparison. The main findings can be summarized as follows; the anomaly detection and cleaning process improves the forecasting ability of the models, ETS is the best forecasting model and SVM model

is the most promising among the ML models and the most competitive with the leading one. Besides, ARIMA has the poorest generalization ability among the others.

Keywords: machine learning, time series, random forest, long short term memory, support vector machine, exports forecasts

ÖZ

ZAMAN SERİLERİNİN TAHMİNLENMESİNDE KLASİK VE MAKİNE ÖĞRENMESİ YAKLAŞIMLARINA YÖNELİK KARŞILAŞTIRMALI BİR ÇALIŞMA:TÜRKİYE'NİN İHRACATI ÜZERİNE DENEYSEL BİR ANALİZ

Günel, Eda
Yüksek Lisans, İstatistik Bölümü
Tez Yöneticisi: Doç. Dr. Ceylan Yozgatlıgil

Temmuz 2020 , 74 sayfa

İhracat, ülkeler için temel ekonomik göstergelerden biri haline gelmiştir. Bu doğrultuda, ihracatın doğru tahmin edilmesi karar alma süreçlerinde oldukça önemlidir ve en uygun tahmin modelinin bulunması birçok çalışmanın konusunu oluşturmaktadır. Bu çalışmada, zaman serilerinin tahminindeki popülerliği ve başarısı dikkate alınarak makine öğrenmesi yöntemlerinin Türkiye ihracatı üzerindeki tahmin performansları analiz edilmiştir. Bu kapsamda, uzun kısa süreli bellek (LSTM), destek vektör makineleri (SVM) ve rassal orman (RF) yöntemlerinden yararlanılmış ve bulgular klasik zaman serisi modelleri olan bütünleşik otoregresif hareketli ortalama (ARIMA) ve üstel düzeltme (ETS) modelleriyle karşılaştırılmıştır. Analizde, Türkiye İstatistik Kurumundan (TÜİK) alınan Ocak 1997 – Eylül 2019 tarihleri arasında gerçekleşen aylık ihracat verileri kullanılmıştır. Analizin temel aşamaları, anomalilerin belirlenmesi ve temizlenmesi, veri üzerinde yapılan ön işlemler, model gelişimi, parametrelerin ayarlanması ile model seçimi ve seçilen modellerin karşılaştırılmasıdır. Çalışma sonucunda elde edilen bulgular şu şekilde özetlenebilir: İhracatın tahminlenmesinde en

iyi tahmin modeli ETS olurken; SVM, makine öğrenmesi modelleri arasında gelecek çalışmalar için en umut verici model olmuştur. Bunun yanında, ARIMA modeli ihracat verisinin tahmininde diğer modellere göre zayıf performans göstermiştir.

Anahtar Kelimeler: makine öğrenmesi, zaman serileri, rassal orman, uzun kısa süreli bellek, destek vektör makineleri, ihracat tahmini

To my beloved daughter

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my advisor Assoc. Prof. Dr. Ceylan Yozgatlıgil for her guidance and patience during this thesis. In every steps, she was very kind and understanding. It was a grate chance to study with her.

I owe my sincere thanks to the committee members, Prof. Dr. Özlem İlk Dağ and Prof. Dr. Seher Nur Sülkü for sparing their valuable time to examine this thesis.

I would like to thank my friend Tuğcan Adem Özdemir for his guidance and assist. His knowledges and experiences on thesis writing were pretty much beneficial.

I would also like to thank my chair at Ministry of Trade, Aysun Tekin for her patience, support and kind attitude.

I would also like to express my deepest and special gratitude to my husband Aykut Yücel Günel for his great support and encouragement.

I would also like to thank my family members; Nazım Karabacak, Sevil Karabacak, Kamber Günel, Sevgi Günel and Damla Günel for their support and sacrifice during this thesis.

Finally, I would like to state that I dedicate this thesis to my lovely 2 years old daughter, Ece Günel. She showed great patience within this period which deprived her mother of spend more time with her.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	
ADAM	Adaptive Moment Estimation
AIC	Akaike's Information Criterion
AICc	Akaike's Information Criterion Correction
ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average Model
ARMA	Autoregressive Moving Average Model
ARMAX	Autoregressive Moving Average Cause-Effect Model
BIC	Bayesian Information Criterion
BRP	Binary Recursive Partitioning Algorithm
CART	Classification and Regression Trees
ERNN	Elman Recurrent Neural Network
ETS	Exponential Smoothing based on Error, Trend, Seasonal Components
GDP	Gross Domestic Product
GESD	Generalized Extreme Studentized Deviate Test
KPSS	The Kwiatkowski–Phillips–Schmidt–Shin Test
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MLP	Multilayer Perceptron
MSE	Mean Squared Error
OCSB	Osborn, Chui, Smith, and Birchenhall
OLS	Ordinary Least Square

PPML	Poisson Pseudo Maximum Likelihood
RF	Random Forest
RMSE	Root Mean Squared Error
RMSPRrop	Root Mean Square Propagation
RNN	Recurrent Neural Network
SES	Simple Exponential Smoothing
SSE	Sum of Squared Error
STL	Seasonal Trend Decomposition by Loess
SVM	Support Vector Machine
TURKSTAT	Turkish Statistical Institute
TÜİK	Türkiye İstatistik Kurumu
WTO	World Trade Organization

CHAPTER 1

INTRODUCTION

In the globalized world, international borders have been removed in economic aspects with an increasing trade and financial relations among countries. Along this line, a steady increase in exports has been the main road for economic development and a sustainable economic growth [8]. In this context, with the purpose of more powerful exports structure, developing countries make a grate effort to be more competitive and integrated in the global value chains. Turkey is one of those countries in which the export is deemed as one of the main economic indicator. The growth and industrialization policies in Turkish economy were based on the import substitution industrialization strategies before 1980, especially in 1960s and 1970s. As a result of this, the resources could not be integrated in the competitive sectors and Turkish economy became dependent more on the external sources. After all, it was unable to compete with the rest of the world with low quality and high prices [9].

In 1980, a structural adjustment program called "24th January Decisions" was introduced. With this new approach, Turkey has adopted export-oriented industrialization policy instead of the import substitution industrialization strategies; thereupon the opinion of exports being the driving force of the economic growth has gained importance [10].

The economic value of exports has triggered analysts and researchers to conduct studies on the issue such as estimating the relationship between trade flows and economic growth, finding optimal model for export structure and foreseeing the future values. In the same way, many international institutions and economic research bodies have produced forecasts on economic indicators such as trade flows, GDP (gross domestic product) growth in order for developing an overview on a country's economic outlook

and decision making on the economic policies beforehand.

Most of the traditional forecasting models are based on fitting data by pre-determining the relationship between input and output variables and assuming a specific functional/stochastic process [11]. Despite of the fact that these models are generally accurate for particular cases, they are not capable of working well in non-linear cases. With the proliferation of computing technology and urgent solution for this gap, the machine learning models, which do not necessitate a specific assumption for the structure of the data, has come to the fore [12]. In line with this, the researches on machine learning (ML) performance on time series data and comparison studies with the traditional models have risen. However, which model outperforms is still an open question. The results basically vary depending on the research field, the structure of the data, forecast horizon, multivariate or univariate cases etc.

In this thesis, we applied popular machine learning methods and traditional time series methods on one of Turkey's main economic indicator, exports data. The main purpose of this study is not to find the best model and make a forecast for the future, but to realize which method performs better on the data and more promising for the future studies. In this respect, we applied "Random Forests (RF)", "Support Vector Machines (SVM)", "Long Short Term Memory (LSTM)" in the machine learning side and ARIMA and ETS for the benchmark. To the best our knowledge, there is no such a study evaluating the performance of SVM, RF and LSTM with the classical time series models in forecasting Turkish exports. In this regard, this study will contribute to the literature being a base for the future studies in this field.

We planned this master thesis as follows. In Chapter 2, we introduced the methodology used while conducting the analysis. We started with the anomaly detection and cleaning procedure and then continued with the classical time series models and machine learning methods. We finished up this chapter with the methodology for the model evaluation and comparison process. In Chapter 3, we presented the results of each model and compared them by checking the root mean squared error (RMSE) and mean absolute error (MAPE). In Chapter 4, we concluded our thesis summarizing the main findings and making suggestions for the possible studies in the future.

1.1 Literature Review of Time Series Forecasting Models for Turkish Exports

Turkish exports as an important economic indicator has been widely studied from various aspects. While some researchers have tried to reveal the relationship between the exports and other macroeconomic indicators in Turkey, some others have tried to find the best forecasting model out. In addition to these two fields, another group has placed a special emphasis on finding determinants of Turkish exports. In this thesis, we studied on time series forecasting area and accordingly divided this section for providing further insights on the studies conducted before within the scope of forecasting models on Turkish Exports. Some of the studies are summarized as follows.

Ersen et al. [13] studied on determining the most accurate time series model for exports and imports of the paper and paper product sector applying Box-Jenkins methodology. They used monthly data from January 2003 to December 2013 to conduct the models and monthly data from January 2014 to December 2014 to test the forecasting performances of the models. They used sum of squared errors (SSE) and mean squared error (MSE) to select the model and mean absolute percentage error (MAPE), mean absolute error (MAE) and root mean squared error (RMSE) to evaluate their forecasting performances. According to the results, they found the best model for the paper and paper products exports prediction as ARIMA(2,1,0)$(0,0,1)_{12}$ with a MAPE value of 18.4 and import prediction as ARIMA(3,1,2)$(1,0,1)_{12}$ with a MAPE value of 8.05.

Başer et al. [14] researched on the best model for chestnut production and export amounts. By using the Box-Jenkins method, they obtained ARIMA(1,1,1) as the best-fitted model for production and ARIMA(1,2,1) for export amounts. The model was fitted on the data from 1961 to 2016. Besides that, they made forecasts for the period of 2017-2021 using the selected models.

Demir et al. [15] aimed at estimating the loss of Turkey's exports to Iraq between 1988-2003 applying, Exponential Smoothing, Winters' Additive and Winters' Multiplicative, and SARIMA models. Then, he compared the model fitting and test performances using MAPE, MAE, and RMSE. According to the results, winters' additive

model was the best in both fitting training set and test estimation performance with lower error levels. The MAPE value for the test data was 16 for the best model and 38 for the SARIMA model.

Akal [16] estimated hazelnut export of Turkey by using simple econometric cause-effect and autoregressive moving average cause-effect (ARMAX) methods for the 1998-2001 period. For the model development, he used exchange rate, hazelnut export quantity and hazelnut export revenue as the explanatory variables. Then, he compared two models' out-of sample forecasting performances for the 2002-2006 period based on MAPE criterion. They found that ARMAX technique outperformed the simple cause-effect technique on average based on the MAPE results which was found as 61.2 on average.

Uysal and Karabat [17] studied on the raisin export of Turkey. By using the annual data of raisin export quantity on 1982-2015, they fitted a model using double exponential smoothing approach. Then, they used MAPE values for the model evaluation. The MAPE value was found 10 indicating high accuracy for the forecasting purpose and they made forecasts for the following 5 years concluding that export of raisin will decrease.

Karahan [18] compared feedforward neural network model in dried apricot export volume forecasting with ARIMA model. He used 5 independent variables which are time, exchange rate, dried apricot prices, number of exporting markets and percentage damage caused by the seasonal effects as inputs. For the model development of neural network, 5 input layers, 1 hidden nodes with 5 nodes and one output layer were used. Data for the 5 variables was taken between January 2004 and December 2010. He used 70 data rows for training and 14 for the model testing. According to the MAPE values, he found that neural network model having lower MAPE value (15.1) was more consistent and accurate in predicting dried apricots compared with ARIMA model (MAPE: 23.3).

Ozbek and Akalın [19] predicted Turkey's denim trousers export to Germany with multilayer perceptron (MLP) and Elman Recurrent Neural Network (ERNN) models. They used 29 monthly input variables such as Turkey's denim trousers import, minimum wage in Turkey, Export Credit etc. for the 1995-2008 period. For the model

development, they used 2 hidden layers. In addition to this, they used R^2 and RMSE for the evaluation purpose and concluded that Elman Network with less RMSE values has better prediction performance than MLP.

1.2 Literature Review of Time Series Forecasting Models for Exports

In this section we searched on the global studies for time series forecasting models on exports some of which are summarized as below.

Farooqi [20] studied on annually exports and imports of Pakistan from 1947 to 2013. In the study, he aimed at finding the most appropriate forecasting model and built ARIMA models using Box-Jenkins approach through R statistical software. He chose the models ARIMA (2,2,2) and ARIMA(1,2,2) for imports and exports respectively, by observing the model selection criterion, AIC for each models. Then, he evaluated these tentative models by analysing their residuals for the existence of lack of fit using Box-Pierce and Ljung-Box tests. After the model evaluation, he obtained the forecasts for the following 5 years.

Alam [21] investigated forecasting models for annual exports and imports of the Kingdom of Saudi Arabia using Artificial Neural Network (ANN) and ARIMA models. He fitted models on a yearly data from 1968 to 2017 through the statistical software XLSTAT and made a forecast for the following 3 years. According to the results, he concluded that ARIMA (1,1,2) showed better predictive performance than ARIMA(1,0,0) and ARIMA (0,1,1) models for total exports. Furthermore, he pointed out that ANN outperformed the proposed ARIMA (1,1,2) model based on the RMSE values.

Ghosh [22] utilized ARIMA models to forecast cotton exports in India by using the monthly data from April 2010 to June 2015 through statistical software Gretl and Stata. He followed the Box-Jenkins methodology and proposed ARIMA(1,1,0) model having the minimum AIC value. After the model diagnostics were checked, he concluded that the proposed model can be used for the future forecasting purpose and he obtained forecasts for the following 5 months. Additionally, he used Simple Exponential Smoothing (SES) and Holt's Exponential Smoothing (HES) models as the

benchmark and concluded that ARIMA had better predictive performance according to the RMSE values.

Wohl and Kennedy [23] analysed neural networks' generalization performance on international trade comparing with Ordinary Least Square (OLS) and Poisson Pseudo Maximum Likelihood (PPML) estimators. They used aggregate bilateral manufacturing trade data as the dependent variable and GDP, distance, border, language, colonial ties, and trade agreements as the explanatory variables. The dataset consisted of 68 partners' trade data from the year 1986 to 2006. They used 70% of the dataset for training and developing the models and the rest of them for testing. After the model development, they obtained predictions for trade between United States and its major trading partners for the following 10 years and measured out-of-sample prediction accuracy based on RMSE. According to the results, multi-layered perceptron model with one hidden layer having 20 nodes and the sigmoid function had better generalization ability.

It has been identified that the classical time series models such as ARIMA, ARMAX, Exponential Smoothing etc. have dominated the studies conducted with the purpose of examining export structure and making forecasts and in a small group of those, neural network algorithms on the machine learning side have begun to take stage. It is of significant to note that not only the number of the studies conducted by using machine learning algorithms, but also number of variety of the methods are insufficient in terms of export forecasting researches. Along this line, this study, in which SVM, LSTM, RF were utilized, will be the very first step to bridge the gap and deemed as a fine resource in this field. Another contribution of the study is that the computation time of the models were calculated which provide an insight for the researchers and bring a new dimension for model selection. Besides, in this study anomaly detection and cleaning process as opposed to the others were applied and put forward the truth that the models with cleaned data yield more precise forecasts. In this regard, this process presents an alternative approach to improve the model forecasting results.

CHAPTER 2

METHODOLOGY

In this chapter, we briefly introduced the methodology used in the anomaly detection and cleaning process, then we presented each models applied in this thesis. Moreover, we explained the model selection process for both classical and machine learning sides. Finally, we defined the model evaluation and comparison steps.

2.1 Anomaly/Outlier Detection and Data Cleaning

Anomalies and outliers are widely used interchangeably within the context of anomaly detection [24]. Anomaly/outlier detection is the process of capturing patterns that shine amongst others in the data and do not conform to expected or normal behaviour [25]. This process is one of the most important part in time series forecasting problems since anomalies in data can cause to get faulty results. Two main steps of anomaly detection and data cleaning process are listed as follows:

- Time Series Decomposition
- Detecting Anomalies/Outliers

2.1.1 Time Series Decomposition

A time series data is comprised of three components: seasonal, trend-cycle and remainder components. Seasonal components exhibit regular patterns during the same months or quarters every year. Trend is the long-term pattern while cycle is any pattern showing up and down movements around the trend. These two terms are gen-

erally combined into a single trend-cycle component and called as trend for simplicity [26]. Remainder component is the remaining part of a time series when seasonal and trend components are removed. It demonstrates a random increase or decrease in a specific time period [27]. A time series model is assumed to be either additive or multiplicative [28]. An additive model and a multiplicative model can be described as follows:

- Additive model of Y at time t:

$$\mathbf{Y}_t = \mathbf{S}_t + \mathbf{T}_t + \mathbf{R}_t \tag{2.1}$$

- Multiplicative model of Y at time t:

$$\mathbf{Y}_t = \mathbf{S}_t * \mathbf{T}_t * \mathbf{R}_t \tag{2.2}$$

where Y is the data, S is the seasonal component, T is the trend-cycle component and R is the remainder at time t.

If the variation in the seasonal fluctuations, or the variation around the trend-cycle change proportional to the level of the time series, then a multiplicative decomposition is more suitable to the model. Multiplicative models are commonly seen on economic data [26]. Alternatively, a multiplicative model can be converted to an additive model by using a log transformation and then additive decomposition can be applied. The transformed model can be described as follows:

- Additive model of logY at time t equivalent to the Equation (2.2):

$$\mathbf{log}\mathbf{Y}_t = \mathbf{log}\mathbf{S}_t + \mathbf{log}\mathbf{T}_t + \mathbf{log}\mathbf{R}_t. \tag{2.3}$$

The main purpose of time series decomposition in the context of anomaly detection is to obtain the remainder component since the anomaly detection process can cause faulty results with the existing of seasonal and trend term.

2.1.1.1 Seasonal Trend Decomposition By Loess (STL)

Seasonal trend decomposition by loess [29] is a robust decomposition method for obtaining seasonal, trend and remainder components. It performs additive decomposition of the data through a sequence of applications of the loess smoother while loess is one of the non-parametric regression technique and stands for *locally weighted scatterplot smoothing* [29]. A loess model is denoted by

$$\mathbf{y} = \mathbf{g}(x) + \epsilon \tag{2.4}$$

where g is the locally-fitted polynomial function at x and ϵ is an error term. The idea is that, at any point *x* whose response is being estimated, a polynomial is fitted to a subset of the data consisting of the data points near *x*. The weighted least squares is used in the fitting procedure giving more weight to the points closer to *x* and decreasing the weights based on the neighbours' distance from *x*. This procedure is applied for each *n* data point and the loess fit is completed.

Then, the loess fit at *x* with degree *p* is

$$\hat{\mathbf{g}}(x) = \hat{\beta}_0 + \hat{\beta}_1 x + \cdots + \hat{\beta}_p x^p. \tag{2.5}$$

STL method has several advantages:

- Any kind of seasonality not limited to monthly or quarterly periods can be tackled.
- The seasonal component is not obliged to be a constant and it can vary and the rate can be controlled by the user.
- It provides the ability to control the smoothness of the trend-cycle by the user
- The effects of outliers can be handled.

On the contrary, poor performance with the trading day or calendar effects and providing only additive decomposition are two disadvantages of STL decomposition method. However, a log transformation to the multiplicative model can be a solution to the second drawback.

2.1.2 Detecting Anomalies/Outliers

After obtaining the decomposed time series an anomaly detection method can be applied on the remainder component. Then, determined outliers are removed and replaced with trend and seasonal terms.

2.1.2.1 Generalized Extreme Studentized Deviate Test (GESD)

The generalized extreme studentized deviate test (GESD)[30] is one of the methods to identify one or more outliers in a time series. This method only takes an upper bound for the number of outliers that is desired to be observed. This number can be determined at most 49.9% of the total number of data points according to [31]. Given the upper bound, *k*, GESD performs *k* separate test that is, a test for one outlier, another test for two outliers, and so on. The hypothesis for the test:

$\mathbf{H}_0$: There is no outlier found in the data set

$\mathbf{H}_1$: There are up to *k* outliers found in the data set

Test statistic (T) for the k most extreme values in the data set is defined as follows:

$$\mathbf{T}_i = \frac{\max_i \mid x_i - \overline{x} \mid}{s} \tag{2.6}$$

where $\overline{x}$ and s are the mean and standard deviation respectively.

The corresponding critical value is defined as follows:

$$\lambda_i = \frac{(n-i)t_{p,n-i-1}}{\sqrt{(n-i-1+t^2_{p,n-i-1})(n-i+1)}} \tag{2.7}$$

where n is the number of observation in a data set i=1,2,...k and $t_{p,v}$ denotes t distribution with v degrees of freedom.

The observations are tested by comparing the test statistic with the critical value to determine anomalies. If the tested value is an outlier then it is removed from the data set and the critical value is recalculated with *n-1* observation. This process is repeated *k* times and *k* test statistics and critical values are obtained in total. For the largest value of k, let call *r*, such that $T_{\mathrm{r}} > \lambda_{\mathrm{r}}$, it is concluded that there are r outliers in the data set.

2.2 Classical Time Series Models

In this section, we introduced the classical time series models applied in the study which are Autoregressive Integrated Moving Average Model(ARIMA) and Exponential Smoothing Model (ETS) and the criteria for the model selection.

2.2.1 Autoregressive Integrated Moving Average Model(ARIMA)

The ARIMA model is one of the most widely used classical approach for univariate time series forecasting. It was generalized and popularized by statisticians George Box and Gwilym Jenkins [32]. In this respect, the ARIMA models can also be mentioned as Box-Jenkins models. ARIMA model is generalized form of autoregressive moving average (ARMA) model.

An ARIMA model combines Autoregressive (AR) and Moving Average (MA) processes as well as the integration (I) part and it is stated as ARIMA(p, d, q)[33] where,

- p is the number of autoregressive term
- d is the number of non-seasonal differences necessary for stationarity
- q is the number of lagged prediction errors

The autoregressive process, AR(p): Autoregressive model is a regression model which assumes that Y_t is a linear function of its lagged values and has the form of Equation (2.8)

$$\mathbf{Y}_t = \sum_{i=1}^{p} \phi_i Y_{t-i} + \varepsilon_t \tag{2.8}$$

where Y_t is a stationary time series, the ϕ_i terms are autocorrelation coefficient at lags, from 1 up to p, and the residuals, ε_t, are white noise terms with $mean = 0$ and $variance = \sigma_\varepsilon^2$.

The moving average process, MA(q): A moving average model is a regression model which uses the past forecast errors as a predictor instead of the past values

of the forecast variable. It can be written as

$$\mathbf{Y}_t = \sum_{i=0}^{q} \theta_i \varepsilon_{t-i} \tag{2.9}$$

where θ_i terms are the moving average coefficients relating Y_t to the current and past white noise terms, ε_t with $mean = 0$ and $variance = \sigma_\varepsilon^2$ and $\theta_0 = 1$.

Then, the combined two models can form an ARMA(p, q) model:

$$\mathbf{Y}_t = \sum_{i=1}^{p} \phi_i Y_{t-i} + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}. \tag{2.10}$$

This model requires stationary time series. A stationary time series is the one whose statistical characteristics do not depend on the time, that is to say mean and the autocovariance function are constant over time. In order to meet this requirement, variance-stabilizing transformation and differencing are applied [34]. A log transformation can stabilize the variance while differencing can make the mean of a time series constant eliminating the trend and seasonality [35]. Within the context of mean stabilizing, differencing, ARMA model necessitates an **integrated part** denoted as I, in which a non-stationary time series is converted to stationary by taking the non-seasonal difference of the consecutive time series values. Then, ARMA model is turned into ARIMA model denoted with the backshift notation:

$$(1 - \phi_1 B - \cdots - \phi_p B^p)(1 - B)^d Y_t = (1 + \theta_1 B + \cdots + \theta_q B^q)\varepsilon_t \tag{2.11}$$

where B is the backshift operator used for the simplicity.

Seasonal ARIMA model: ARIMA models can also handle with seasonal data by including seasonal terms to the model and are denoted as: ARIMA$(p, d, q)(P, D, Q)_m$ while (p, d, q) represents the non-seasonal part, (P, D, Q) shows the seasonal part in which P is the seasonal AR order, D is the number of seasonal difference, Q is the seasonal MA order and m stands for the number of periods in each season [36].

The Box-Jenkins methodology: This method is three-stages guideline for ARIMA time series models including model identification, model parameter estimation and model checking [37]. These steps are extended with the addition of data preparation and model application phases [38]. The steps are summarized as follows:

- **Data preparation:** This step includes transformation and differencing processes to obtain a stationary data. Firstly, a proper transformation is applied

such as Box-Cox transformation [39] to achieve stationarity. If it is not met, differencing should be applied based on d and D which are identified through non-seasonal unit root tests such as PP [40] and KPSS [41] test and seasonal unit root tests such as HEGY [42] and OCSB [43] test.

- **Model identification:** After identifying d and D and obtaining the stationary series, both seasonal and non-seasonal AR or MA orders are decided by using autocorrelation (ACF) and partial autocorrelation (PACF) plots of stationary series.

- **Parameter estimation and model selection:** The parameters of the identified model are estimated by using the method of maximum likelihood. After estimation, the model selection tools such as Akaike's Information Criterion (AIC), Akaike's Information Criterion correction (AIC_c) and Bayesian Information Criterion (BIC) are used to select the best model among the proposed models.

- **Statistical model checking:** The goodness of the fitted model is checked by using some plots and formal tests on the residuals. If the model residuals resemble a white noise (zero mean, constant variance, uncorrelated process and normally distributed [44]), the model is appropriate. The adequacy of the model is formally tested by a portmanteau test such as Ljung-Box test [34] having the null hypothesis that the residuals do not exhibit serial correlation and a normality test such as Shapiro-Wilk test [45] having the null hypothesis that the residuals belong to a normal distribution. Besides, the significance of the proposed model parameters are tested by student-t test and the modelling procedure is completed.

- **Forecasting:** Finally, the forecasts can be computed once the model is selected, estimated and checked.

ARIMA models follow these 5 steps-methodology stated above and R has a function called $auto.arima$ under $forecast$ package which conducts all steps automatically except model checking and forecasting [35] and it was used in this thesis to find the best ARIMA model.

To add more on $auto.arima$, we used default tests defined in the function. It applies KPSS test for the existing of non-seasonal unit root and a seasonal strength test for seasonal unit root which is computed by STL decomposition [46]. Indeed, we applied OCSB test for the existence of the seasonal unit root to be sure, which provided the same result with default.

2.2.2 Exponential Smoothing Model (ETS)

ETS, which is an acronym for Error, Trend and Seasonal components, is another popular forecasting method for a univariate time series. ETS models are based on state space approaches underlying exponential smoothing methods. They focuses on modelling seasonal and trend components [47].

The classification for exponential smoothing methods is first introduced by Pegels (1969), enlarged by Gardner(1985), modified by Hyndman et al. (2002) and widened by Taylor (2003) as stated in [48]. The table for this classification is as follows.

Table 2.1: Classification of Exponential Smoothing Methods

	Seasonal Component		
Trend Component	N (None)	A (Additive)	M (Multiplicative)
N(None)	(N,N)	(N,A)	(N,M)
A (Additive)	(A,N)	(A,A)	(A,M)
A_d(Additive damped)	(A_d,N)	(A_d,A)	(A_d,M)
M (Multiplicative)	(M,N)	(M,A)	(M,M)
M_d(Multiplicative damped)	(M_d,N)	(M_d,A)	(M_d,M)

Each ETS(E,T,S) model has either additive or multiplicative error, which mainly distinguishes the methods from models [49]. Therefore, for each method there are two options and 30 different models come in sights in total.

It is possible to describe each model by using ETS(E,T,S) notation [50]. A simple model can be ETS(A,N,N) representing the simple exponential smoothing (SES) with additive errors, no trend and no seasonality, where N stands for "none". In the simple exponential smoothing method, one-step ahead forecast is obtained by weighted

averages of the past observations, where the weights decrease exponentially through the past values and the forecast equation can be shown as;

$$\widehat{y}_{t+1|t} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \cdots \tag{2.12}$$

where α is the smoothing parameter such that $0 \leq \alpha \leq 1$. Alternatively the component form can be used to represent the exponential smoothing. In this respect, the component form of Equation (2.12);

$$\widehat{y}_{t+1|t} = l_t \tag{2.13}$$

$$l_t = \alpha y_t + (1-\alpha)l_{t-1} \tag{2.14}$$

where l_t is the level of the series at time t as stated in Equation (2.14) and (2.13) represents the forecast equation.

The state space model for the corresponding method above, SES, is obtained by using the forecast error $e_t = y_t - l_{t-1} = y_t - \widehat{y}_{t|t-1}$ at time t, which is thought as the model noise ε_t distributed as $NID(0, \sigma^2)$ [47].

The state space model:

$$y_t = l_{t-1} + \varepsilon_t \tag{2.15}$$

$$l_t = l_{t-1} + \alpha\varepsilon_t \tag{2.16}$$

where Equation (2.15) is called as *measurement equation* and (2.16) is named as *state equation*. While the measurement equation represents the relationship between the observations and the unobserved states, the state equation shows changes in the state over time [49].

This model has a simple form; however, it can be extended according to the series in concern such that adding a trend term and/or seasonal term will create a new model such as ETS(A,A_d,A), ETS(A,A,M). The state space equation for ETS(A,A_d,A) can

be shown as follows:

$$\begin{aligned} y_t &= l_{t-1} + \phi\, b_{t-1} + s_{t-m} + \varepsilon_t \\ l_t &= l_{t-1} + \phi\, b_{t-1} + \alpha\varepsilon_t \\ b_t &= \phi\, b_{t-1} + \beta\varepsilon_t \\ s_t &= s_{t-m} + \gamma\varepsilon_t \end{aligned} \tag{2.17}$$

where b_t stands for the slope at time t, s_t represents the seasonal component at time t and m is the number of seasons.

In the model estimation step, the smoothing parameters α, β, γ and ϕ on Equation (2.17) and the initial states of l, b and s are estimated by maximizing the likelihood and the one among the candidate models is chosen based on the information criteria such as AIC, AIC_c and BIC [49]. The restriction for the coefficients is that they can take the values between 0 and 1. There is an R function called as ets which conducts the modelling steps automatically like $auto.arima$ function. It checks all the combinations of trend and seasonal components and choose the model with minimum AIC_c. After the model estimation, the residual analysis is required for the model adequacy.

2.2.3 Model Selection Criteria

In this subsection, we presented model selection criteria used in classical approaches for choosing the best model among the candidates. AIC_c and AIC used by $auto.arima$ and ets functions as defaults respectively.

- **Akaike's Information Criterion (AIC):**

$$AIC = 2k - 2log(L(\widehat{\theta} \mid y)) \tag{2.18}$$

  where *k* is the number of estimated parameters in the model and L is the log likelihood at its maximum value of the estimated model. The model is chosen regarding the one which maximizes the likelihood and at the same time minimizes the AIC.

- **Akaike's Information Criterion correction (AIC_c):**

$$AIC_c = AIC + \frac{2k(k+1)}{n-k-1} \tag{2.19}$$

where *n* is the sample size. If *n* is large then both criteria will be equal to each other. AIC_c is more general and can be used instead of AIC [51]. The idea is the same; the model with the lowest AIC_c is chosen.

- **Bayesian Information Criterion (BIC):**

$$BIC = klog(n) - 2log(L(\widehat{\theta} \mid y) \tag{2.20}$$

Both AIC and BIC penalize the likelihood when more parameters are added to the model to prevent the overfitting; however, BIC has larger penalty term than AIC.

2.3 Machine Learning Approaches (ML)

This section consists of three machine learning methods; the first one is LSTM, the second one is RF and the third one is SVM. After the model definitions, we introduced cross validation techniques used in the model selection phase and the methodology for the model evaluation.

2.3.1 Long Short Term Memory (LSTM)

LSTM is a type of Recurrent Neural Network (RNN). Before going deep into the LSTM networks, first we introduced Artificial Neural Networks (ANNs) and RNNs structure.

Artificial Neural Networks (ANNs): Artificial neural networks are composed of artificial neurons or nodes grouped in layers. Figure 2.1 illustrates the working structure of an artificial neuron. The neuron takes the inputs and adds them up by giving weights to each of them according to the importance of the inputs. After the summation, neuron applies an activation function to obtain the output value.

The simplest version of ANNs is the perceptron models comprising of a single artificial neuron and they form the basis of various ANN models. They work in the binary

classification problems mapping the real valued inputs to a single binary value. The output is either 1 or 0 which is determined by observing the weighted sum whether it is less than or greater than a threshold value. This perceptron algorithm works with linearly separable data sets and uses step function as the activation function.

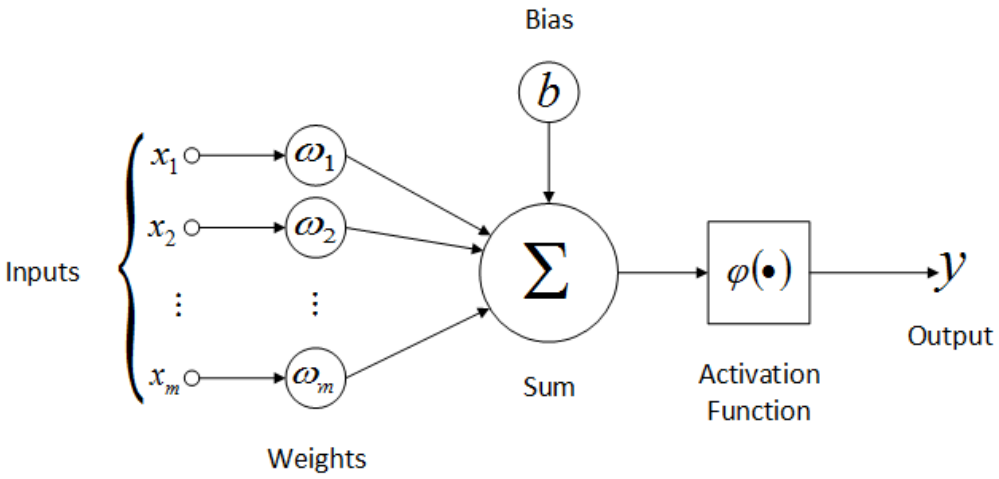


Figure 2.1: A flowchart of an artificial neuron structure [1]

ANNs are structured by three layers called input layer, hidden layer and output layer. Nodes in the input layer are connected to the nodes in hidden layer with links giving weights to each node in the input layer. The weights are crucial in deciding which input can pass through. The hidden layer is the learning part of the network and it filters inputs based on the weights. Nodes in hidden layers use an activation function on the weighted sum of inputs to transform inputs to outputs [33]. The output layer produces candidate output values and chooses the one giving minimum error. However, this may not be the optimal model. To achieve the optimal model the weights are re-adjusted based on the output errors. This process starts from back, the output layer, goes through the input layer and is called as *back-propagation*. The back-propagation is repeated to achieve the optimal model, called training. Training stops when the error between predicted and actual outputs does not decrease anymore and the optimal model is achieved.

Recurrent Neural Networks (RNNs): RNNs have a structure of predicting the next step by learning a sequence of observations and taking the previous steps in the mem-

ory [33]. This type of neural network is *recurrent* since the network uses the previous outputs as inputs for calculating the next outputs and it performs the same process for each observation in the sequence [52]. This recurrence process provides the neural network a memory property remembering the previous stages and storing them in network's internal state [53]. An unfolded RNN structure can be shown as follows.

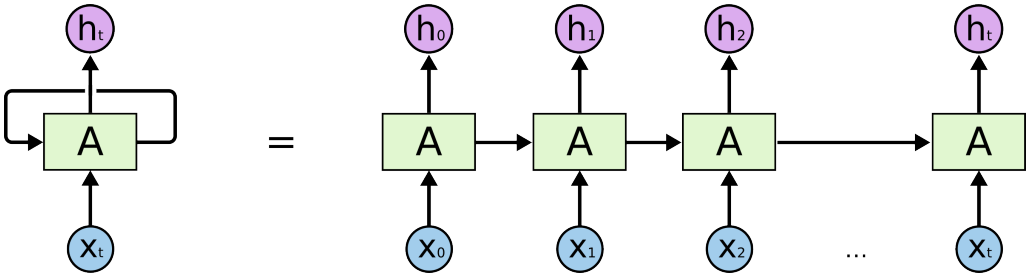


Figure 2.2: An unfolded RNN structure with an input layer, a single hidden layer and an output layer [2]

In Figure 2.2, the network takes the input x_t and creates the output h_t by using both current information and the sequence of information coming from the previous stages and also *A* illustrates the hidden layer. However, RNNs are not convenient for longer sequences because they can remember only a few earlier steps [33]. The solution to this drawback is the Long Short Term Memory (LSTM) networks.

Long Short Term Memory (LSTM): LSTM network has basically the same structure with RNN. The main difference in LSTM is that the summation units in the RNN's hidden layer, seen on Figure 2.2, are replaced by *memory blocks* [53], which provides the capability of learning long-term dependencies to the network. The internal structure of an LSTM, the memory block, can be seen as follows.

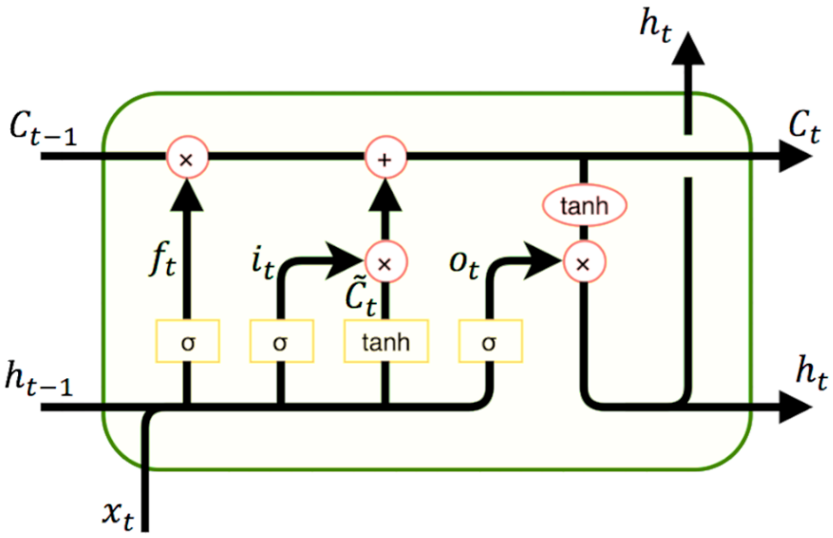


Figure 2.3: The internal structure of an LSTM [3]

The memory blocks recurrently connected to each other constitute LSTM networks. The main players in a LSTM block are the inputs; x_t, h_{t-1}, and c_{t-1} namely *input vector* at time t, *hidden state*, and *cell state* at $t-1$ and three gates for controlling the state of each cell are *forget gate*, *input gate* and *output gate*. The working process of an LSTM is as follows.

- **Step 1.** The forget gate f_t decides what information will be kept and thrown away from the cell state through a sigmoid activation function (σ). The forget gate takes x_t and h_{t-1} and calculates number between 0 and 1. 0 means "completely throw away" and 1 means "completely keep". The formulation for the forget gate is;

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f). \tag{2.21}$$

- **Step 2.** There are two phases. Firstly, the input gate i_t decides which values will be updated. Secondly, the new candidate values, $\widetilde{C}_t$ are obtained by using hyperbolic tangent activation function ($tanh$). The formulation of the input gate and new candidate values are;

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i), \tag{2.22}$$

$$\widetilde{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c). \tag{2.23}$$

- **Step 3.** The input gate and the new candidates are multiplied. Therefore, the sigmoid output of the input gate determines which new candidate values will be added to the cell state.

- **Step 4.** The old cell state is updated by multiplying old cell state with the forget gate and adding the multiplication of input gate and new candidate values, as in step 3. Updated cell state can be formalized as follows;

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t. \tag{2.24}$$

- **Step 5.** The output is obtained based on the cell state updated in previous step. This step has also two phases. First, the output gate o_t determines which parts of the old cell state will be produced as output through the sigmoid function. Second, the updated cell passed through hyperbolic tangent function is multiplied by the output gate resulting in updated hidden state h_t which will be used for predictions at time $t+1$. Output gate and updated hidden state is formalized as follows,

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o), \tag{2.25}$$

$$h_t = o_t * tanh(C_t). \tag{2.26}$$

  The W's and b's are weights and bias terms respectively and these are readjusted while minimizing the error between LSTM outputs and actual values.

In addition to sigmoid and hyperbolic tangent, there are some other activation functions such as softmax and relu, however; the most popular activation functions adopted in LSTM structures are sigmoid and hyperbolic tangent. Others can perform successfully in other type of neural networks. The formulation for sigmoid and hyperbolic tangent functions are as follows:

$$Sigmoid(x) = \frac{1}{1 + e^{-x}}, \tag{2.27}$$

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{2.28}$$

Moreover, LSTM models have a complex architecture and use many hyperparameters for the model building. We introduced them and described briefly as follows.

- **Learning Rate:** It is one of the most important hyperparameter for a neural network. It refers to the amount of change in the weights after each iteration [89]. If it is too small, it will take much time to reach the optimal point and requires more iterations. On the other hand, if it is too large, it will reach to a some point quickly other than the optimal [90]. Although it should be tuned to find the best performing one, 0.001 is suggested as a reasonable selection in [90].

- **Epoch:** The number of times that the model is trained on the entire dataset. After each epoch, the model is re-trained and the weights are updated until the prediction error continues to decrease [33]. The more epochs means the more training and the more correction on the weights. Therefore, higher epochs are desirable without causing overfitting problem.

- **Batch Size:** This is the number of training examples to pass through one epoch.

- **Number of hidden layer:** This is the number of memory blocks. The RNN models are deeper and more complex with more hidden layers which is desirable for the model development [91]. However, increasing the number of hidden layers makes the training harder. Likewise, the model is more likely to overfit the data when it is getting more deeper and one or two hidden layers are enough to get good results [92].

- **Number of hidden units:** This can be described as the dimensionality of the output state and hidden state. It is another hyperparameter that affects the learning capacity and possibly leads to overfitting with much larger values [90].

- **Optimizer:** This is the algorithm to be used for optimizing the weights to reduce the error. In the LSTM structure we applied two popular algorithms; Adaptive Moment Estimation (Adam) and Root Mean Square Propagation (RMSProp) optimizers and then compared the results. They are basically modified versions of a gradient descent algorithm which computes the coefficients of a function minimizing the cost.

  Briefly, RMSProp works similar with the gradient descent algorithm with momentum which is used for accelerating the algorithm. It restrains the oscillation in the vertical direction while going through the global minimum point

and allow faster convergence in the horizontal direction [93]. It chooses different learning rates for each parameter (coefficient) and adjusts them during the learning process [94]. It utilizes the moving average of the squared gradients for each parameter. The working structure of ADAM is similar with RMSProp but it adds momentum to RMSProp keeping also moving average of the past gradients. Generally, ADAM outperforms the RMSProp but it can change in different tasks and as a hyperparameter they should be tuned and the best performed should be used.

2.3.2 Support Vector Machine (SVM)

The main task in SVM is to separate the data into two classes using a line where the data points belonging to first class are on one side of the line and other data points belonging to the second class are on the counter part of the line [52]. Although its basic function is based on the classification, SVMs are used for regression problems, as well and called as Support Vector Regression (SVR) in the literature [54].

The regression problem can be linear or non-linear and the core idea behind the rise in SVMs is their success for non-linear cases [55]. The working structure of SVMs in the regression problems is similar to the classification problem with some particular features. The SVR uses linear functions defined in a high dimensional space to solve the non-linear regression problems and they approximate the function as shown in the following equation:

$$f(x) = w^T \phi(x) + b \tag{2.29}$$

where $\phi(x)$ is the high dimensional feature space non-linearly mapped from the data x [56]. The coefficients w and b are weights and the constant terms respectively and w is estimated by minimizing the function shown in Equation (2.30).

$$J(w) = \frac{1}{2} w^T w \tag{2.30}$$

subject to the residuals less than ε shown as follows:

$$w^T \phi(x) + b - y_i \leq \varepsilon$$
$$y_i - w^T \phi(x) - b \leq \varepsilon$$

where y_i is the actual response values. The objective is to find the function $f(x)$ with at most ε deviation from the training data (y_i) which is as flat as possible [57]. To exactly fulfil the condition under Equation (2.30) is not possible for all data points. Therefore, slack variables are used to make the condition more flexible and the function becomes as Vapnik proposed [58]:
Minimize

$$J(w, \xi_i^+, \xi_i^-) = \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\xi_i^+ + \xi_i^-) \tag{2.31}$$

subject to the constraints:

$$\begin{aligned} y_i - (w^T \phi(x) + b) &\leq \varepsilon + \xi_i^+ \\ (w^T \phi(x) + b) - y_i &\leq \varepsilon + \xi_i^- \\ \xi_i^+, \xi_i^- &\geq 0, \text{ for } i = 1, ..., n \end{aligned}$$

where n is the total number of data points. ξ_i^+, ξ_i^- are the slack variables. C is the regularization constant (cost) which determines the trade-off between generalization ability and accuracy in the training data, and ε is the degree of tolerance to errors and called as tube size of SVR [59]. SVR can be illustrated with the following scheme:

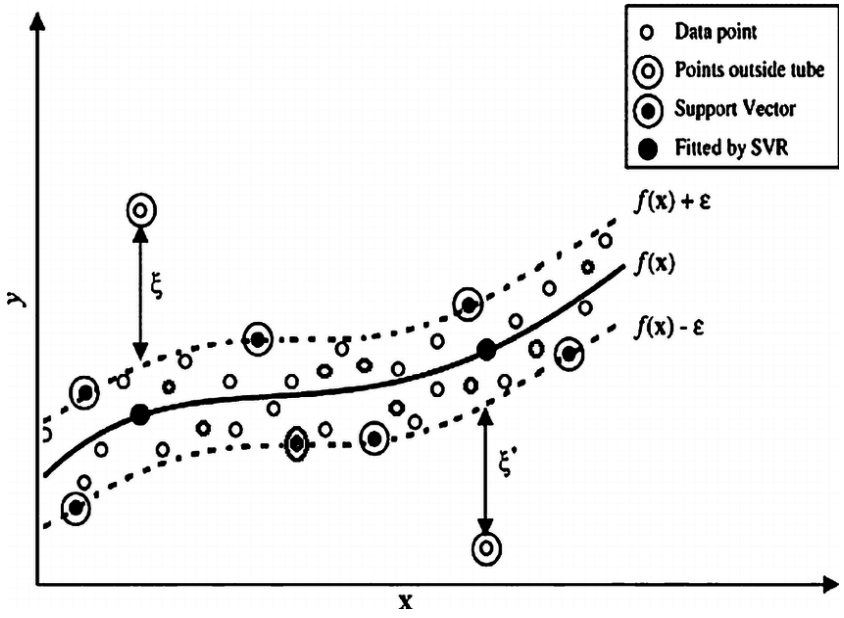


Figure 2.4: A schematic diagram of SVR [4]

By using the Lagrange multipliers, the support vector regression function $f(x)$ in Equation (2.29) can be expressed in the following form [60]:

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x, x_i) + b \tag{2.32}$$

where α_i, α_i^* are the Lagrange multipliers and $K(x_i, x_j)$ is the kernel function which calculates the inner product of two vectors in the feature space and can be shown as $K(x_i, x_j) = \phi(x_i) * \phi(x_j)$ [56].

Although there are many choices for the kernel functions, to catch the non-linear structure within the data, the radial basis function (RBF), also known as Gaussian kernel function, is the one most widely used and defined as follows [55]:

$$K(x_i, x_j) = exp(\frac{-\|x_i - x_j\|^2}{2\sigma^2}). \tag{2.33}$$

Lagrange multipliers in Equation (2.32), α_i and α_i^*, are obtained by maximizing the dual function of Equation (2.31) given as follows [56]:

Maximize

$$\begin{aligned} J(\alpha_i, \alpha_i^*) = &-\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i, x_j) \\ &- \varepsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) \end{aligned} \tag{2.34}$$

subject to

$$\begin{aligned} \textstyle\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0, \\ 0 \leq \alpha_i, \alpha_i^* \leq C. \end{aligned}$$

The constant term b given in Equation (2.29) and (2.32) is computed by the following equation:

$$b = average_i\{\delta_i + y_i - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x, x_i)\} \tag{2.35}$$

where δ_i is exact value of the prediction error [60] and shown as $\delta_i = \varepsilon sign(\alpha_i - \alpha_i^*)$.

Training performance of the SVR mainly depends on three hyperparameters such as regularized constant C, tube size ε and the Kernel parameter which is σ^2 in this study. These are user-determined parameters and needed to be considered in parameter tuning process to obtain better results. Grid search or optimization methods are used to achieve the best combination of hyperparameters.

Tay and Cao [56] investigated the sensitivity of the SVM models to the hyperparameters and showed that the performance of SVMs is insensitive to the ε when it is selected properly. On the other hand, improper selection of C and σ^2 can result in overfitting or underfitting problems. Therefore, the selection of C and σ^2 is important for the SVMs performances. While 0.001 is considered as reasonable for ε, the range of 1-100 for σ^2 and 10-100 for C are found appropriate [56].

2.3.3 Random Forest (RF)

Random Forest (RF) proposed by [61] is another machine learning method used for both classification and regression problems. It is an ensemble learning algorithm based on CART (Classification and Regression Trees) [62]. In this thesis, we focused on the regression part of RF for our forecasting purpose. To understand RF structure, it is important to grasp a better understanding for regression tree which is the backbone of the structure.

Regression Trees : Regression trees is a term describing the decision tree algorithms used for the tasks with continuous response variable. The objective is to predict the response values in regression problems and to classify the observations in classification problems.

Regression trees used in RF is constructed by binary recursive partitioning (BRP) method [63] in which partitioning represents dividing the data into sub-groups on the basis of predictor variables. Binary stands for describing the partition or split of the data into two daughter sub-groups and recursive describes the continuity of the split process on each sub-groups or nodes until a stopping criterion is reached [64].

Binary Recursive Partitioning Algorithm (BRP) [63] : Let $x_1, ..., x_N$ and $y_1, ..., y_N$ be the training data with $x_i = (x_{i,1}, ..., x_{i,p})^T$.

1. Start with a node comprising all data points (x_i, y_i) called root node,
2. Find the best split among all possible splits on all p predictor variables.
3. Split the node into two new nodes based on the best split found in step 2.
4. Repeat steps 2 and 3 recursively on each unsplit node until reaching the stopping criterion.
5. To predict the response variable at x, go down through the tree until landing in the final node, called terminal node, which will not be split anymore based on the stopping rule. Let z be the terminal node and let $y_{z_1}, ..., y_{z_n}$ denote n response values of the training data in node z. Predicted values of the response variable is given as follows:

$$\widehat{g}(x) = \overline{y}_z = \frac{1}{n}\sum_{i=1}^{n} y_{z_i}. \tag{2.36}$$

In the above algorithm, step 2 and step 4 have some critical processes needed to be explained further. Finding the best split for a node mentioned in step 2 is based on the splitting criterion which is the mean squared error at that node and computed as follows [63]:

$$R = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2 \tag{2.37}$$

where $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ is the predicted value at that node. The splitting criterion, R, shows how well the data is fitted namely the goodness of fit. In this context, large values of R indicate poor regression.

Let C be the split point. Based on C values decided through the splitting criterion, training data is divided into two nodes, one is on the left side containing the predictor values less than C and the other is on the right side containing the predictor values greater than C.

The split point C is chosen by minimizing the following equation:

$$R_{split} = n_L R_L + n_R R_R \tag{2.38}$$

where R_L and R_R are splitting criteria for the two new nodes with their sample sizes n_L and n_R respectively.

Best split is decided by ordering predictor values and calculating R_{split} for all possible values of C. This calculation is applied on each predictor variables and the one with the minimum R_{split} value is chosen.

At this stage, the question is when the splitting process will stop. As it is stated in step 4, splitting continues recursively until the pre-defined stopping criterion is met. One of the widely used stopping rule to control the tree growing process is the *minimum node size* which represents the minimum number of observation in the node. If the unsplit node size is less than the user-defined minimum node size, that node will not be split. Within this scope, when all unsplit nodes meet this condition the process will stop and unsplit nodes become terminal nodes [63].

As mentioned earlier, RF is a tree-based ensemble learning method which combines the predictions from multiple regression tree algorithms for the regression problems. In the constructing process of a RF, an ensemble learner bagging (bootstrap aggregation) gets involved.

Bagging (Bootstrap Aggregation) : Bagging stands for "*bootstrap and aggregation*" as it uses bootstrap samples from the original data and takes the average of the predictions obtained from the samples while fitting a regression model [65].

Bootstrap refers to resampling randomly from the data with replacement and the samples are called bootstrap samples. Bagging fits the models (regression tree in our RF) on each bootstrap sample and prediction functions called as "base learners" $\widehat{g_1}(x), ..., \widehat{g_j}(x)$ are obtained [63]. After that, it combines the base learners to construct an ensemble prediction function $\widehat{f}(x)$ computed as follows:

$$\widehat{f}(x) = \frac{1}{J} \sum_{j=1}^{J} \widehat{g_j}(x). \tag{2.39}$$

As a result of this, combining the predictions from multiple models yields lower variance compared to a single model prediction [66].

Random Forest Algorithm : RF is obtained by growing a forest of many trees with the same logic of bagging algorithm of trees stated above. However, RF has an additional feature compared to the bagging scheme. In the node splitting process, RF uses m randomly selected variables out of p predictors in total independently at each node which provides randomness to the regression trees as bagging does. Therefore, RF gains randomness in two ways; one with the bootstrap sampling and one with randomly selected predictors. Additionally, this makes the trees less correlated and more diversified [67].

The RF algorithm can be itemized as follows:
Let $x_1, ..., x_N$ and $y_1, ..., y_N$ be the training data with $x_i = (x_{i,1}, ..., x_{i,p})^T$. For $j = 1, ..., J$;

1. Take a bootstrap sample D_j with a size of N from the training data.

2. Fit a regression tree on the sample D_j using BRP algorithm modified as follows:

   (a) Start with a root node including all observations.

   (b) For each unsplit node, repeat the following 3 steps until reaching the stopping rule;

      i. Select m predictors randomly from p predictors.

      ii. Find the best split among all possible splits on m predictors chosen previously.

      iii. Split the node in two daughter nodes based on the best split.

To predict response variable at a new point x, the results are combined as shown in Equation (2.39) in bagging algorithm. RF prediction structure can be illustrated as in the following figure:

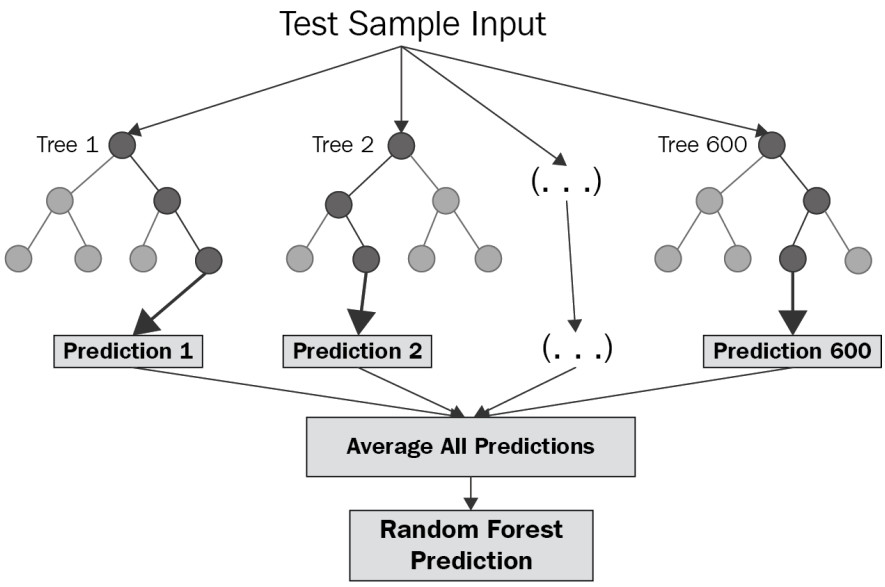


Figure 2.5: A schematic diagram of RF prediction [5]

As it can be seen in the RF algorithm, there are some hyperparameters which may influence the RF performance and may be tuned for accurate results. Probst et al. [99] investigated the tunability of the parameters of six common machine learning methods including RF. They concluded that RF works well with default values and needs parameter tuning less than others.

However, if one needs to tune the parameters, the biggest improvement can be provided by the number of parameters randomly chosen (m) called as $mtry$. The other parameter that should be considered is the number of trees called as $ntree$ which is not a tunable parameter but suggested to be sufficiently large [100].

2.3.4 Hyperparameter Tuning and Model Selection

In this subsection, we introduced the hyperparameter tuning and cross validation techniques used in this thesis to achieve the optimal model.

Hyperparameters are user defined parameters which may change the accuracy of the model to a large extent [68]. Therefore, they should be treated and determined carefully before training the model. The procedure is based on trial and error approach,

i.e., the model is trained for different hyperparameters, then the model performance is measured by a validation error which is typically mean squared error [69] computed as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 \tag{2.40}$$

where y_i is the actual observations and $\widehat{y}_i$ is the predicted values for n observation.

In tuning step, we applied manual search and grid search approaches. In manual search, the hyperparameters and their candidate values are determined based on some knowledge or experience without an automated selection, then the model is trained with manually changing parameters. Trial continues until a satisfying result is obtained. This approach allows to learn the behaviour of the hyperparameters [70] and it does not yield the best model but results in a better model.

In grid search approach, for all possible hyperparameters, a model is trained and the validation error is obtained for each. It gives the best combination of hyperparameters and the optimal model is achieved. This approach can be computationally heavy in the case of having many hyperparameters [68].

The validation error is used as a sign of model performance on unseen data and shows the generalizability of the algorithm. In this respect, validation methods are widely used [71]. In other words, validation allows to tune hyperparameters of a model. At the same time, it allows to observe if the model overfits or underfits the data.

When a learner tries to capture the underlying patterns in the data called as *signal*, overfitting and underfitting problems have come to the fore. Overfitting is a big challenge in model estimation. According to Nate Silver [72] *"overfitting is the most important scientific problem you've never heard of"* and occurs when the model starts to fit the noise instead of the signal [73]. The overfitting is tricky since the model fits the training data quite well but poorly performs on a new set of data and it may not be noticed at the first glance. On the other hand, if the model is unable to learn as much signals as possible, then the model underfits the data.

Well tuned hyperparameters prevent the model from overfitting and underfitting prob-

lems [74] and results in the optimal model. In this context, validation methods are the key paths of choosing the best-performing model among the others produced in the hyperparameter tuning stage, called as model selection [75]. In this thesis, hold-out and k-fold cross-validation methods were applied for the model selection and introduced as follows.

Hold-Out Method : In this method, data set is divided into three parts as training, validation and test as illustrated in Figure 2.6.

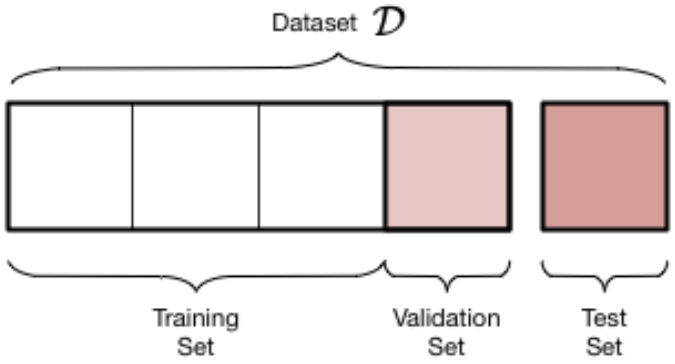


Figure 2.6: The hold-out split diagram [6]

This method allows to learn the data and fit models on training set with different hyperparameters and see how the models perform on the validation set. Based on the model accuracies the best-performing model is chosen. In this context, hold-out method allows to estimate the generalization ability of a model and increase its predictive performance [75]. Finally, the selected model is evaluated on the test set which will be introduced in the next section (2.4).

K-Fold Cross-Validation Method: This method is one of the most widely applied techniques for model selection [75]. In this method, the data is divided into two parts as training and test to evaluate at the end. The cross-validation is applied on the training part which will be randomly partitioned in k almost equal sized subsets namely k-folds. The model is trained on k-1 combined subsets and 1 fold is left for the validation purpose. Each k-fold is used as validation once and the model is trained k times on different combination of remaining subsets. For each k model, the model performance is obtained and all performances are averaged at the end. The final

averaged performance is the cross-validated performance [76]. As in the hold-out method, k-fold cross-validation allows to tune hyperparameters and select the best-performed model based on the cross-validated performance. There is no such a strict rule regarding what *k* should be; however, 5 or 10-fold is often used in practice [77]. The k-fold partitioning can be illustrated as in the following figure.

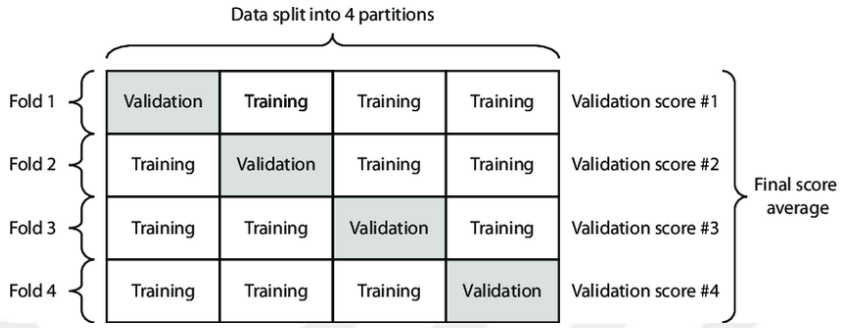


Figure 2.7: The k-fold partitioning scheme (k=4) [7]

This method reduces the bias by fitting and testing the model many times on different sets and repeating the cross validation process reduces the variance in performance measures [76] and increases the precision of the estimates [77]; however, due to its expensive computational cost, the hold-out method can be preferred in the case of having many hyperparameters.

On the other hand, using cross-validation methods on a time series data is a controversial issue in the literature because of the serial correlation in the data; however, Bergmeir and his colleagues [71] has shown that k-fold cross validation can be applied when purely autoregressive models are used.

2.4 Model Evaluation and Comparison

Model evaluation stands for assessing how well the selected model performs on unseen data and the model performances are compared with each other based on the model evaluation metrics. To evaluate the selected model, we used hold-out method by splitting data into two parts as training and test. For a time series data, the last part of the data is reserved for testing. As we mentioned in model selection part,

training data is used for hyperparameter tuning and model selection while test set is for the evaluation. For the evaluation process, we applied the estimated model on test data and predicted the responses of the test data. This can be referred to as one-step forecasts on test data bearing in mind that the test data is not used in the parameter estimation and it gives us a fair forecast [78]. Then, we compared them with the actual responses of the test set.

There is no such a strict criterion for the proportion of the partitions. It depends on the data size and may be defined by some trials as in the tuning step. The common training/test splitting ratios are 60/40, 70/30, 80/20 and 90/10 [75].

Model Evaluation Metrics: In the evaluation process, analysts generally use more than one performance metric to obtain more reliable evaluation and accurate comparison among the models as each metric has unique properties, advantages and disadvantages [79]. In line with this, we considered two common model performance metrics; root mean squared error (RMSE) and mean absolute percentage error (MAPE) calculated as follows:

$$RMSE = \sqrt{MSE}, \tag{2.41}$$

$$MAPE = mean(|100 * \frac{e_t}{y_t}|) \tag{2.42}$$

where e_t is the forecast error and y_t is actual response value at time t.

RMSE is scale-dependent and can only be used for comparing the models on the same data sets. On the other hand, MAPE is independent from the scale of the data and can evaluate the models fitted on different data sets [80]. Furthermore, MAPE shows the accuracy as a percentage. Smaller values of both accuracy metrics (RMSE MAPE) indicate a better model performance.

CHAPTER 3

EMPIRICAL ANALYSIS

In this chapter, we introduced the empirical analysis on Turkish monthly exports data. Firstly, we described the data and introduced the anomaly detection and data cleaning process. Secondly, we presented the classical and machine learning model results. Finally, we finished up this section by comparing the models.

To conduct the analysis, we used R programming language and benefited from its packages and functions allowing automated computation to get fast results.

3.1 Data Description and Anomaly Detection

3.1.1 Data Description

We used Turkish monthly exports data from January 1997 to September 2019 having 273 observations in total. We decided on the starting point by taking the policy changes related with foreign trade into consideration. As stated in the introduction part, Turkey's exports structure has significantly changed with 24^{th} January Decisions in 1980 making Turkish Economy integrated in the global world. Another important pace influencing Turkish economic integration and trade structure is the Customs Unions Agreement with European Union which came into effect in 1996 [81]. While Turkish volume of industrial production and its quality were increasing with this agreement, Turkey became more competitive across the World [82]. In addition to the policy changes, we regarded that Turkish Statistical Institute, compiling foreign trade statistics with Republic of Turkey - Ministry of Trade, has published seasonally adjusted exports statistics starting from 1997. Taking into account all of

these we decided to study within this time interval.

3.1.2 General Overview of the Time Series of Exports

At the first step, we examined the original data to get a general overview and understand the nature of the data. As it can be seen on Figure 3.1, exports data shows an increasing trend especially after 2001 which might be affected by China's participation in the World Trade Organization (WTO) in 2001. Moreover, seasonal fluctuations in the series seems to increase proportionally through time that indicates a multiplicative model as seen in most economic data. Furthermore, the mean and the variance of the series are not constant which necessitates some transformations on the series.

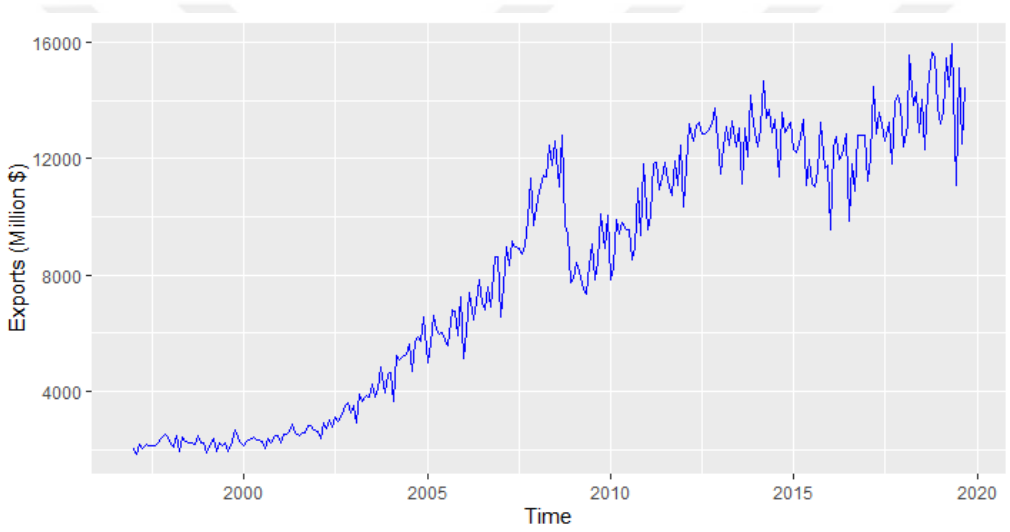


Figure 3.1: Time series plot of Turkish monthly exports from January 1997 to September 2019; Source: TURKSTAT, http://www.turkstat.gov.tr/

Another issue observed in the data is that it has some anomalies which may affect the overall results unless they are handled. Within the time interval that we examined, 2 financial crisis occurred in 2001 and 2008-2009 which can be considered as a potential anomaly points, particularly there was a sharp decrease in export level around 2009. Beside on the visual analysis, we followed the anomaly detection process to obtain the anomalous points and clean them after all.

3.1.3 Anomaly Detection and Cleaning Process

In the anomaly detection and cleaning process, we used *anomalize* package and what we applied is summarized step by step as follows:

- **Step 1:** We took the natural logarithm of the data to transform its multiplicative structure to additive.
- **Step 2:** We applied stl decomposition through $time\_decomposition()$ function to obtain the remainder component of the series.
- **Step 3:** We applied gesd outlier detection method on the remainder component via $anomalize()$ function and we determined the maximum proportion for anomalous points in the data as 20%.
- **Step 4:** We used $clean\_anomalies()$ function to remove the anomalous points from the series. This function replaced them with seasonal and trend components.
- **Step 5:** At the end, we got the cleaned time series data and we back-transformed it to the original series.

Following the first 3 steps, we obtain the anomaly points which is demonstrated in the following figure.

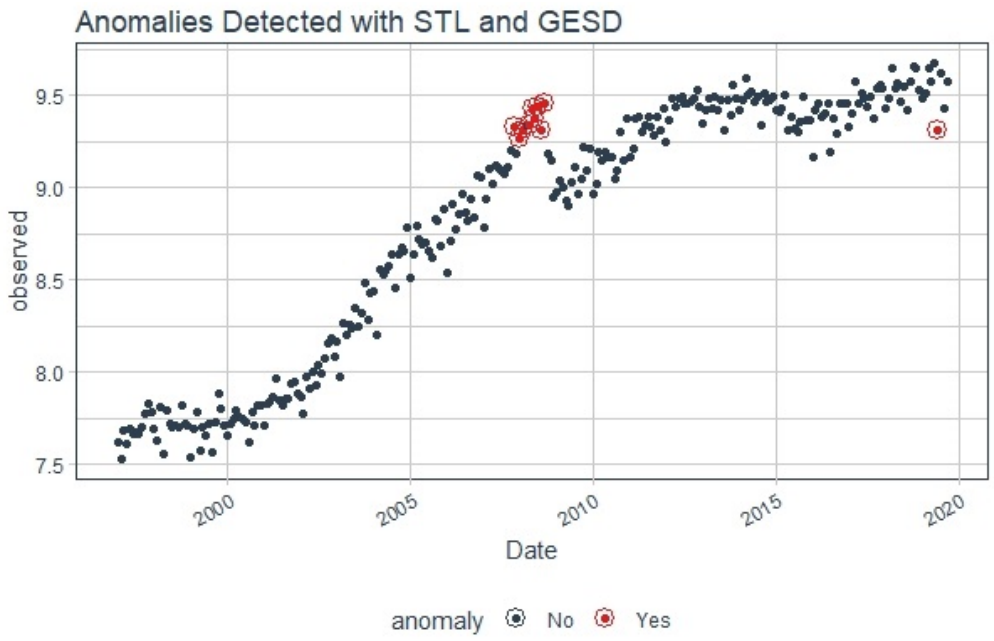


Figure 3.2: Anomaly points in the series

The red labelled points refer to anomalies or outliers, most of which are captured around 2008 in which a global financial crisis affecting many developed and emerging countries came about. Turkey is one of those countries feeling the crisis deeply. The effects started to be seen on the export flows in the second half of 2008 such that the volume of exports decreased and continued to fall with decreasing demand in European countries [83]. 9 anomaly points observed around 2008 were caused where the structural change occurred in the data after the global crisis. On the other hand, there is another outlier detected in June, 2019 in which there was a sharp fall caused by 9 off days due to Ramadan Bairam.

Following the 5 steps above, we obtained cleaned data shown in Figure 3.3 below.

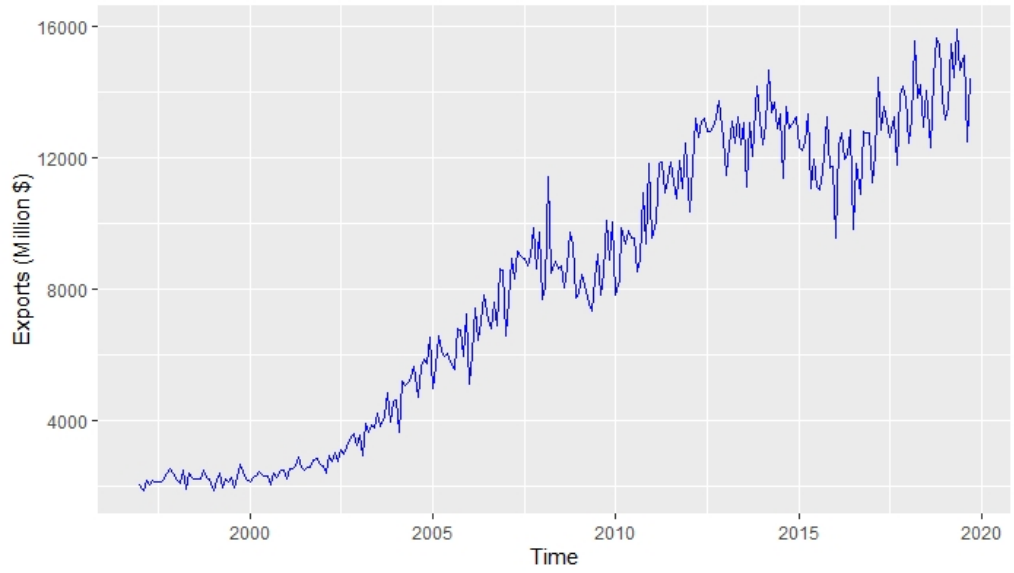


Figure 3.3: Time series plot after cleaning the anomalies

In order to see how this process improved the accuracy, we applied ARIMA and ETS models on both cleaned and the original data. The performance results are discussed under ARIMA and ETS model sections. After all, we continued to the model fitting with cleaned data.

3.2 Classical Time Series Model Results

3.2.1 ARIMA Model Results

At the first step, we divided the data in two as training and test sets with 90/10 splitting ratio and we used the first 90% of the data for the model fitting and last 10% of the data for evaluation purpose.

First, we presented the model with cleaned data. In the data preparation step, we applied variance-stabilizing transformation based on the Box-Cox transformation. First, we got the parameter lambda (λ) using $BoxCox.lambda$ function as -0.0363 which is close to 0 and we took the natural logarithm of the series determining the lambda value as 0. At this stage, $auto.arima$ offered ARIMA(0,1,2)$(0,0,2)_{12}$ with drift by minimizing the AIC_c which is -524.05. The estimated coefficients which can be seen

on the following table are all significantly different than zero. Besides, $\widehat{\sigma}^2 = 0.0066$.

Table 3.1: Model Coefficients

Coefficients	**ma1**	**ma2**	**sma1**	**sma2**	**drift**
	-0.7917	0.1242	0.4781	0.2179	0.0078
s.e.	0.0664	0.0624	0.0650	0.0559	0.0028

As stated before, the model selection process is completed by checking the model residuals whether the proposed model fits the data well. In this context, we used the residual plots to check the model adequacy. According to the plots seen on Figure 3.4, residuals have approximately zero mean and by looking at the histogram it can be said that the residuals seem to meet the normality assumption. Additionally, in order to ensure the normality of the residuals, we applied Shapiro-Wilk test. P-value for the test statistic is 0.5778 which is greater than significance level of alpha (0.05) indicating the model residuals are normally distributed.

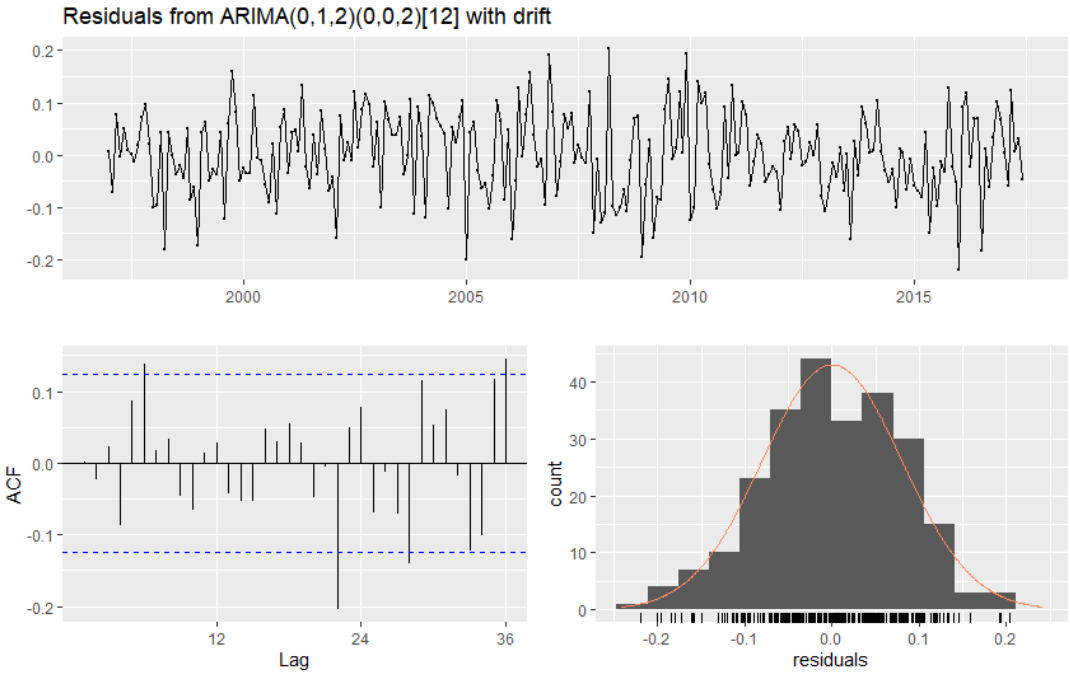


Figure 3.4: The residual plots for the ARIMA model

Besides, while most spikes in ACF plot seem to be within the white noise band, some are above the band; however, we neglected them and applied the Ljung-Box test to see formally whether the model is adequate. According to the test results, the p value is 0.0605. It is greater than significance level of alpha (0.05) indicating that the model does not exhibit lack of fit and we can use it to obtain future values. The plot of actual and predicted values obtained from this model is as follows:

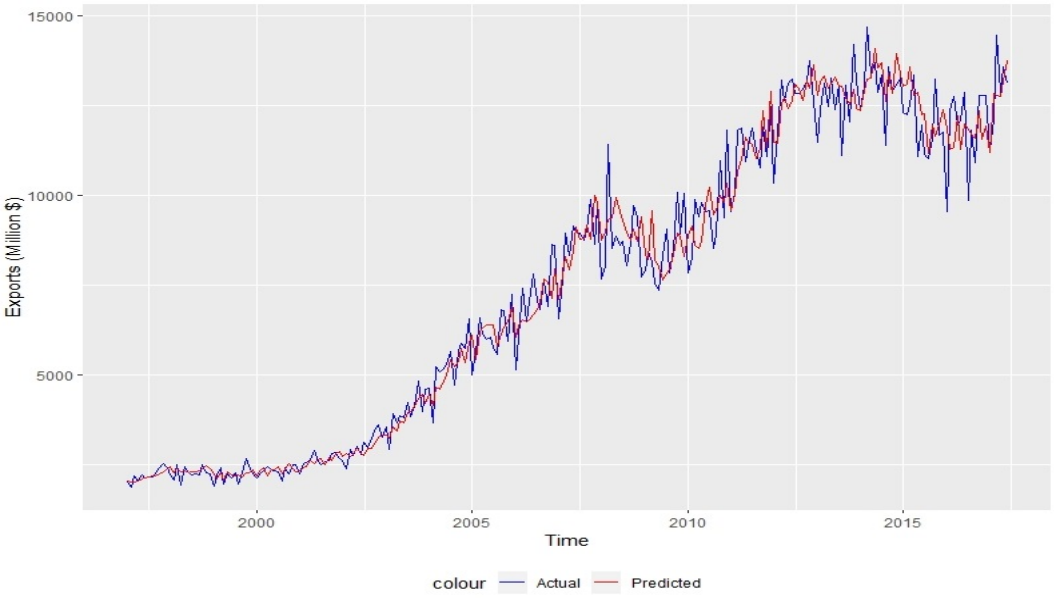


Figure 3.5: Time plot of the actual and predicted values by ARIMA model

Up to here, we used the cleaned data; however, the results below belong to the original data. We followed the same process; applied log transformation (λ is -0.188 close to 0) and the same model was proposed, ARIMA(0,1,2)(0,0,2)$_{12}$ with drift by minimizing the AIC$_c$ which is -491.17. For this model, Shapiro-Wilk test shows that the residuals are not normally distributed (p-value is 0.033<0.05) and the Ljung-Box test indicates a lack of fit (p value is 0.021<0.05). Therefore, we can say that cleaning procedure has led to an improved model. Additionally, we compared their performances on training and test data as shown on Table 3.2. It is clearly seen that the model fitted on cleaned data has lower error metrics indicating a better performance on both training and test data.

Table 3.2: Model Performances for ARIMA Models

Original Data	**RMSE**	**MAPE**
Training Set	728.0385	6.7630
Test Set	1220.5388	7.0409
Cleaned Data	**RMSE**	**MAPE**
Training Set	663.1871	6.4851
Test Set	985.7293	5.7633

Furthermore, we estimated ARIMA model by adding a dummy variable for the observations detected as anomaly instead of cleaning the data to see the difference in forecasting ability of the model. At the end, we found that the model with cleaned data had slightly better performance on the test set.

3.2.2 ETS Model Results

The model fitted on the cleaned data by the ets function is as follows:

ETS(M,Ad,M)

Smoothing parameters:
alpha = 0.2598
beta = 0.0395
gamma = 1e-04
phi = 0.9677
Initial states:
l = 2134.8599
b = 6.9103
s = 1.0408 1.03 1.067 0.994 0.9288 0.9995 1.0014 1.0194 0.9895 1.0781 0.9431 0.9082
sigma: 0.0741

The estimated model is Holt-Winters' multiplicative method with additive damped

trend and multiplicative error. It is chosen minimizing the AIC_c value which is 4380.22 for this model. Damped trend in the model indicates that trend is reduced over future time steps by the rate of damping coefficient phi(ϕ). It is 0.9677 for this model and very close to 1 showing a persistent linear trend. Besides, beta(β) and especially gamma(γ) controlling the rate of change in slope and seasonal components respectively, take the values very close to 0 indicating that they hardly change over time.

For the model evaluation, we performed the residual analysis. From the residual plots on Figure 3.6, the residuals seem to have zero mean, constant variance and be normally distributed. However, ACF plot shows significant correlation between residuals.

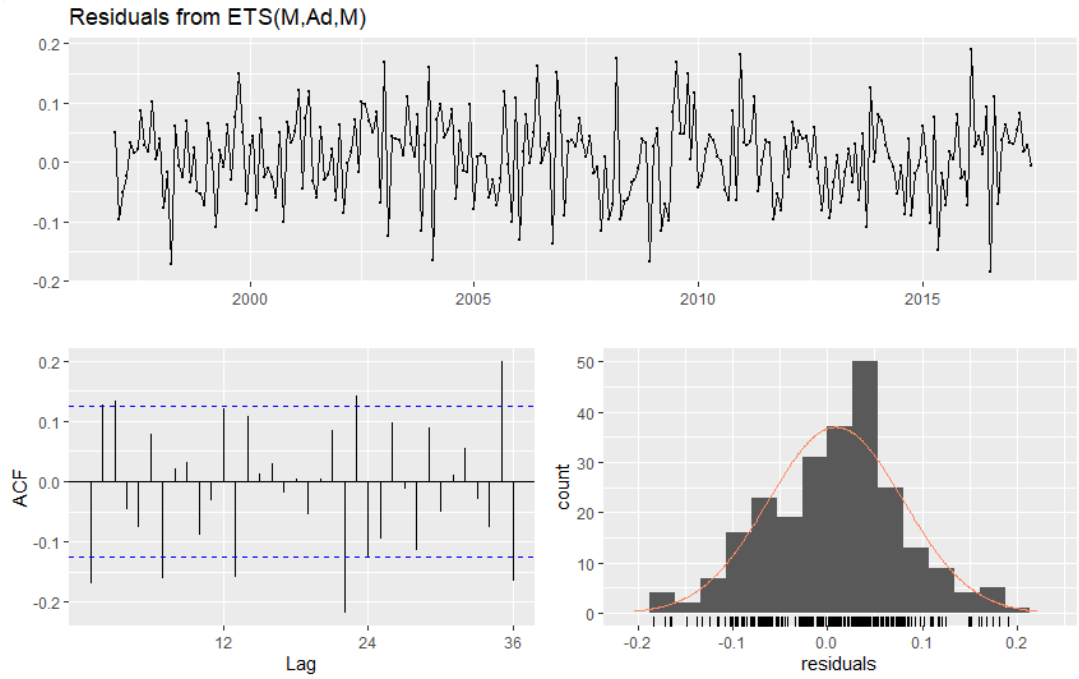


Figure 3.6: The residual plots for the ETS model

In order to check the normality and autocorrelation, we applied Shapiro-Wilk and Ljung-Box tests. The p-value calculated from the normality test is 0.2182 greater than the significance level of 0.05 and the p value obtained from the Ljung-Box test is 3.983e-12 smaller than 0.05. The normality test result shows that the residuals are normally distributed. However, the Ljung-Box test result indicates that they are

correlated. Although, uncorrelated residuals are required for the model adequacy, the forecasts obtained with correlated residuals are still unbiased and not wrong, but it may result in a larger prediction intervals [84]. Time plot of the actual and predicted values obtained from this ETS model is as follows:

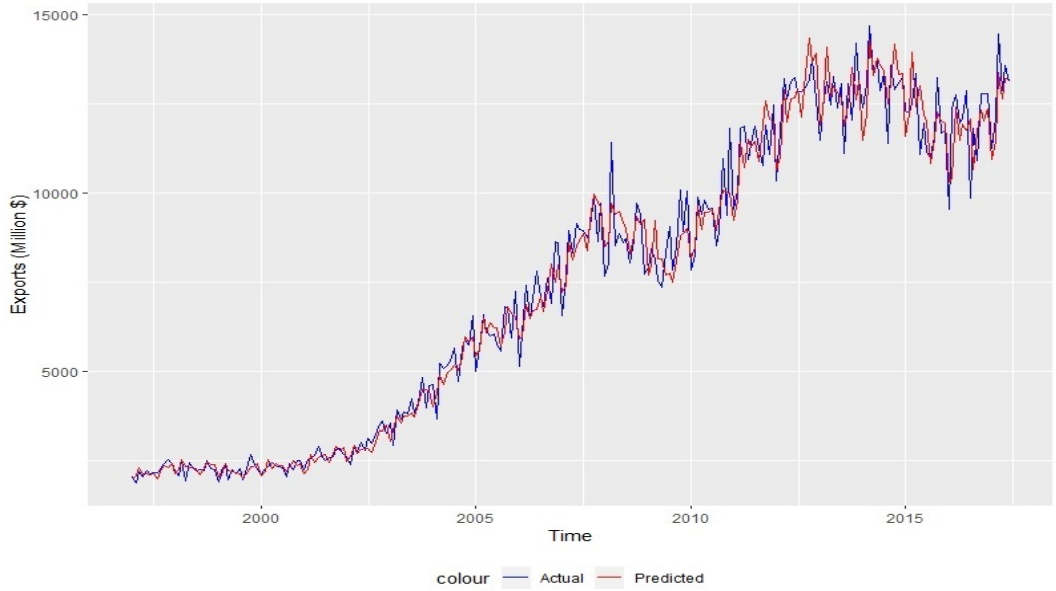


Figure 3.7: Time plot of the actual and predicted values by ETS model

As in the ARIMA modelling, we compared the model performances on the original data and cleaned data through error metrics. The best model for the original series chosen based on the AIC_c which is 4425.786 is ETS(M,A,M) and the comparison is as follows:

Table 3.3: Model Performances for ETS Models

Original Data	**RMSE**	**MAPE**
Training Set	688.5348	6.1272
Test Set	707.0847	4.2348
Cleaned Data	**RMSE**	**MAPE**
Training Set	587.6243	5.7283
Test Set	591.6126	3.4089

The RMSE and MAPE values for the cleaned data is lower than the original one

indicating the anomaly cleaning process improves the model performances as in the ARIMA case.

3.3 Machine Learning Results

In this section, the modelling process of each machine learning method was introduced, then the model results were discussed. Before conducting model training, the data is prepared for the process which may differ in ML techniques. Firstly, we detected anomalies and cleaned them to prevent faulty results and to improve the model performances by following the 5 steps in section 3.1.3. Beside on this, the improvement in the model performance was tested in classical models.

Time Delay Embedding: An important step of data preparation is time delay embedding. It is used for transforming the time series data into a regression task which ML algorithms handle. In this process, our time series is converted to a matrix as follows:

$$A = \begin{bmatrix} y_1 & y_2 & \cdots & y_n & y_{n+1} \\ y_2 & y_3 & \cdots & y_{n+1} & y_{n+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{N-n} & y_{N-n+1} & \cdots & y_{N-1} & y_N \end{bmatrix} \tag{3.1}$$

$A_{(N-n)x(n+1)}$ is the data matrix in which the last column is the response, the remaining columns are inputs or features defined by the lag order and y_i represents the observations. This matrix corresponds to one-step forecasting task that we performed in this thesis. As we mentioned before, in the classical models we kept the last 10% (27 observations) for testing procedure. Likewise, for the ML models we kept the last 27 rows of $A_{(N-n)x(n+1)}$ matrix as test set and the remaining rows as training.

In the following subsections, before discussing the model results, we introduced the preprocessing we applied on the data for each model. Some studies show that without suitable preprocessing, the ML methods may be unstable [85] and it is considered as a vital step to improve the performance and the accuracy of the model [86]. Some of them are data cleansing, normalization, detrending, deseasonalization and log transformation. After the forecasts are obtained, they are transformed back to the original

scale to make a proper comparison with the actual series.

After the data preparation, we continued with the hyperparameter tuning and the model selection part. In the last subsection, we compared each models' forecasting performances.

3.3.1 Long Short Term Memory (LSTM)

Data Preprocessing

LSTM is a type of neural network and Zhang and Qi [87] has found that neural networks have difficulties in capturing the trend and seasonal variations precisely without preprocessing the data and they concluded in their study that detrending or deseasonalization has a big effect on decreasing forecasting errors. In this regard, we also applied differencing and removed the trend from the series.

Another preprocessing step is data scaling recommended for deep learning neural networks [88] such as LSTM models as they are sensitive to the data scales. The problem may be much more difficult to be modelled due to the differences in the scales [88]. Within this scope, we applied normalization to transform the data scales to the activation function's scale which is $tanh$ whose range is [-1,1].

Model Development

Before the hyperparameter tuning, we introduced what LSTM needs for model development. LSTM requires the inputs in 3 dimensions as samples, time steps and features. Sample is the number of sequences that the model uses and it can also be expressed as the number of rows of the data matrix. The sample size for the training data is 233 and accordingly 27 for the evaluation. Time steps is the lag order which we selected as 12 for our monthly seasonal time series and the feature represents the number of variables which is 1 for our univariate case.

For the model development in R, we used *keras* and *tensorflow* libraries which are written in Phyton and integrated into R for deep learning studies.

Hyperparameter Tuning and Model Selection

One of the main obstacle in tuning the LSTM models is the computational cost re-

quiring many iterations for many hyperparameters. Therefore, we applied hold-out method, i.e., we trained models using hyperparameters mentioned in methodology section and tested on a validation set which is the last 10% of the training set. In this stage, we tried to make the model complex as much as possible considering the cost and controlling the overfitting problem. The hyperparameters that we tuned are the number of hidden units, number of epoch and the optimizer. Besides, we set hidden layer as 1, batch size as 1 and the learning rate as 0.001.

We started to the trials with a simple model. By observing the training and validation errors, we changed the hyperparameters and obtained the errors again. For the starting point, we set the number of units as 1, epochs as 10 and the optimizer as ADAM. With these settings, the model did not capture as much signals as possible and underfitted data which can be seen clearly in Figure 3.8 - plot (a) that training and validation loss did not converge to each other and the error value did not decrease as much as needed. Then, we increased epochs to 20 and 30 in which the underfitting problem continued. It can be understood from the results, training with just 1 unit prevented the model to learn the data and the training and validation losses did not drop much (Table 3.4).

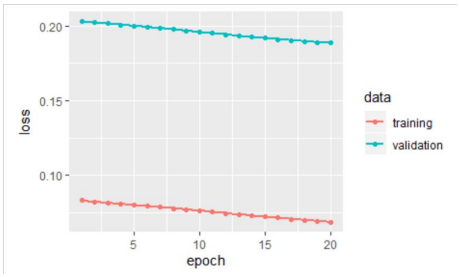
At the second stage, we continued with 20 epochs giving less validation loss and we changed the number of hidden units from 1 to 5 to increase the learning capacity. As we thought, we obtained lower error values compared to the 20 epochs and 1 unit case. However, increasing it to 10 caused to a rise in validation loss and we kept unit as 5. We tried to increase epochs to 30 with 5 units; however, it resulted in increased validation loss and we continued with 20 epochs and 5 units.

At the third step, we changed the ADAM optimizer to RMSProp and trained the model with some combinations of units (5, 10, 15) and epochs (20,30,40,50) by controlling the training and validation loss behaviours. At the end, we achieved the least validation loss with the settings of 5 units, 40 epochs and RMSProp optimizer without underfitting and overfitting the data as shown in Table 3.4 and Figure 3.8 - plot(e).

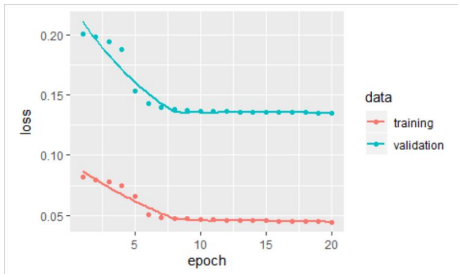
Table 3.4: Hyperparameter Tuning for LSTM Model

Optimizer	**Epochs**	**Units**	**Training Loss**	**Validation Loss**	**Situation**
ADAM	10	1	0.0745	0.1919	Underfitting
ADAM	20	1	0.0623	0.1793	Underfitting
ADAM	30	1	0.0612	0.1826	Underfitting
ADAM	20	5	0.0417	0.1304	Decreasing Loss
ADAM	30	5	0.0426	0.1323	Overfitting
ADAM	20	10	0.0402	0.1328	Overfitting
RMSProp	20	5	0.0437	0.1363	Decreasing Loss
RMSProp	30	5	0.0406	0.1299	Decreasing Loss
RMSProp	**40**	**5**	**0.0402**	**0.1235**	**Decreasing Loss**
RMSProp	50	5	0.0373	0.1251	Overfitting
RMSProp	40	10	0.0351	0.1273	Overfitting

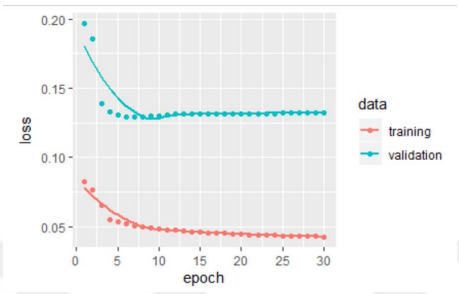
The overfitting situations in Table 3.4 indicates that the model is re-trained as much as the total number of epochs defined beforehand. For example, for 20 epochs, the weights are updated 20 times. In this situations, the validation loss starts to increase after some iterations while the training loss continues to decrease. We eliminated these situations and tried to improve the remaining which are in the situation of decreasing loss seen in Table 3.4. It means that the validation loss decreases until the training process is accomplished. Some of the plots for training and validation loss in Table 3.4. are shown in the following figure.



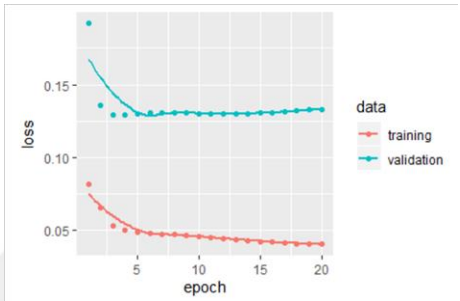
(a) 1 unit - 20 epochs - ADAM



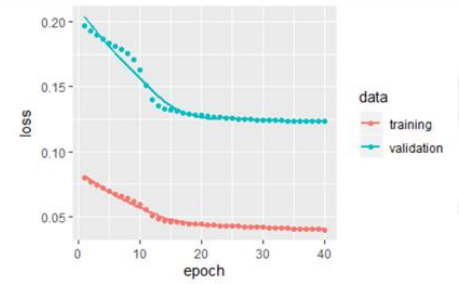
(b) 5 units - 20 epochs - ADAM



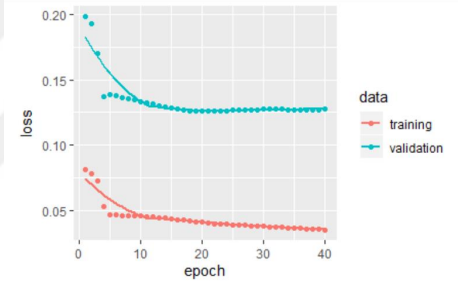
(c) 5 units - 30 epochs - ADAM



(d) 10 units - 20 epochs - ADAM



(e) 5 units - 40 epochs - RMSProp



(f) 10 units - 40 epochs - RMSProp

Figure 3.8: Plots of training and validation loss for different hyperparameters in which loss is the mean squared error

After all, we trained the model with optimal hyperparameters on the entire train set. As we did not apply a grid search we do not say that it is the best one. However, we found a better performing LSTM model on our series and we evaluated the model on the test data.

The model error metrics and the plot for actual and predicted values on the training set are shown on Table 3.5 and Figure 3.9 respectively. The plot for the one-step forecasts on test data was introduced and compared with others at the model comparison part.

Table 3.5: Model Performance for LSTM Model

Cleaned Data	**RMSE**	**MAPE**
Training Set	721.7579	7.2047
Test Set	886.9615	5.3165

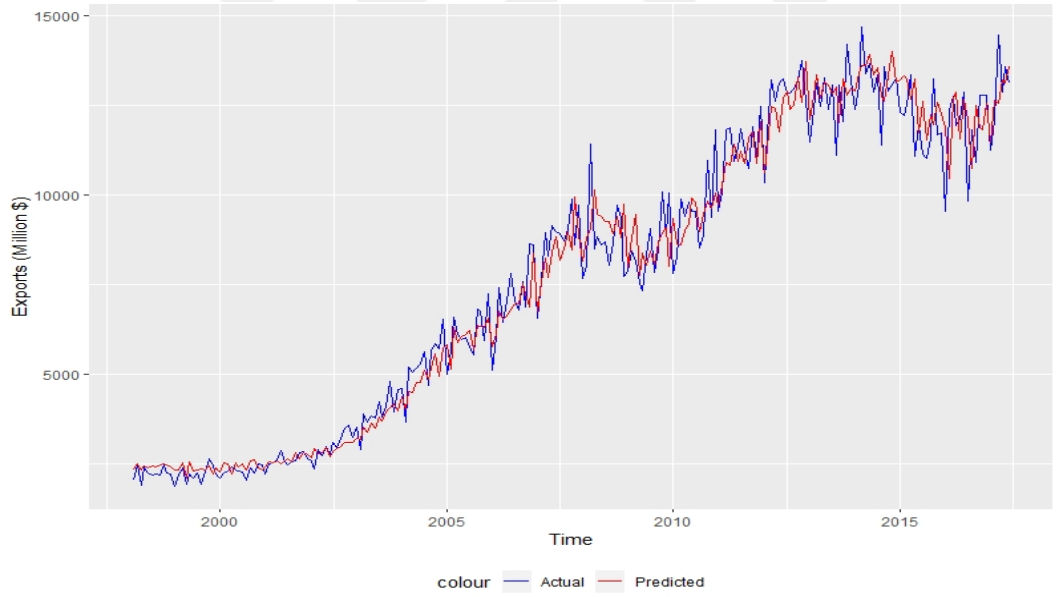


Figure 3.9: Time plot of the actual and predicted values by LSTM model

3.3.2 Support Vector Machines (SVM)

Data Preprocessing

Before the model development, data needs to be processed to improve the model results as in the neural network models. In this context, we applied log-transformation and took the first difference of the series to achieve stationarity in the mean and variance as in the study of Okasha [95] for fitting SVM to the financial time series data.

Model Development

At the model development stage, the hyperparameters, regularized constant C, tube size ε and the Kernel parameter σ^2, are defined to the model. Beside on the hyperparameters, the model also requires time steps, namely lag order. We set time steps as 12 to make the learner capture the seasonal structure of the data and did not change it in the model development process. Indeed, this is considered as another hyperparameter in the literature and can be tuned for the model improvement.

Hyperparameter Tuning and Model Selection

In this thesis, we took the study of Tay and Cao [56] as a reference, as other SVR researchers [96] , [97] for SVR tuning. We used $tune()$ function from $e1071$ package. With the help of this function, we set ε as 0.001 and applied a grid search for the range of 0.01-1 in steps of 0.01 for γ which is defined in $tune()$ function as $1/\sigma^2$ and the range of 10-100 in steps of 10 for C. For the model evaluation in tuning step, we used 10-fold cross validation method repeated 3 times. The best performing hyperparameters were chosen based on minimizing the mean squared error of the validation set. Then, we changed the ε to 0.01 and 0.1 respectively, and repeated the tuning process with the pre-determined ranges for γ and C. According to the results, best performing parameter combinations were obtained as follows:

Table 3.6: SVR Best Performing Hyperparameters

C	ε	γ	**MSE**
10	0.1	0.01	0.006829
10	0.01	0.03	0.007125
10	0.001	0.02	0.007065

Table 3.6 shows the model performance errors on validation sets for three different settings of ε and the least error was achieved when ε was 0.1, γ was 0.01 and C was 10. Then, the model with these hyperparameter values were trained on the entire training set and the predictions on training and test sets were obtained. The model performance metrics and time plot for the predictions and the actual values of the training set are as follows:

Table 3.7: Model Performance for SVR Model

Cleaned Data	**RMSE**	**MAPE**
Training Set	579.4433	5.1158
Test Set	785.2213	4.6774

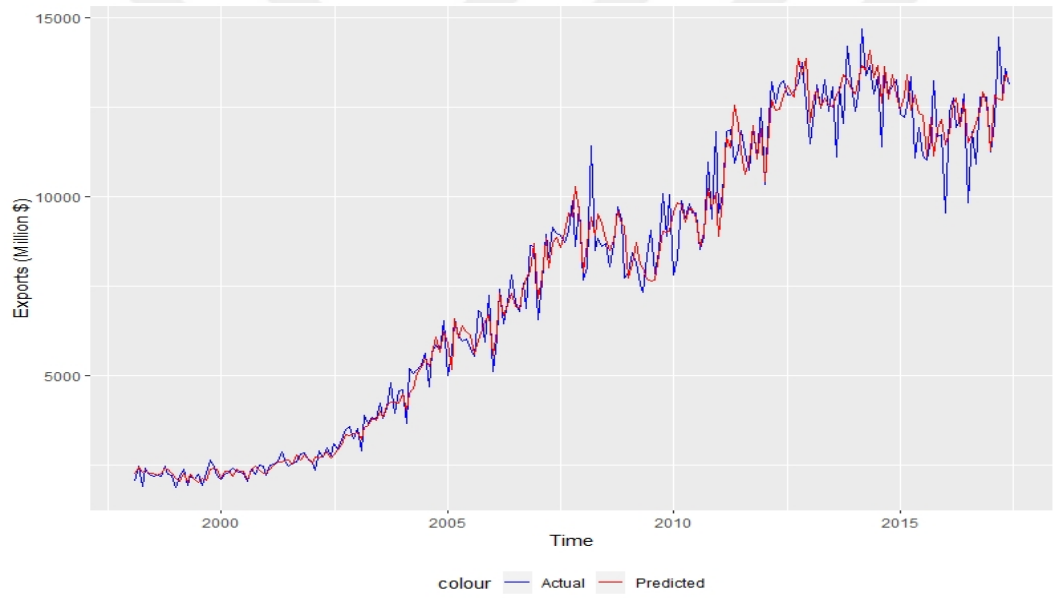


Figure 3.10: Time plot of the actual and predicted values by SVR model

3.3.3 Random Forest (RF)

Data Preprocessing

As in the SVR fitting, we applied log-transformation and took the first difference of the series before the model development. The main reason applying these preprocessing is the poor extrapolating ability of RF models in a time series with trend. RF,

constituting decision trees based on the training set and averaging the results to make a prediction, can make false predictions when it works on a new set which contains greater or lower observations than the training set [98]. Thus, we took the difference of the data to remove the trend.

Model Development

In the model development process, RF requires model hyperparameters. Except the values of $mtry$ and $ntree$, the model requires a pre-defined stopping rule which is chosen as the minimum node size. When it is lower, deeper trees are obtained. In many software packages, the minimum node size is 5 as default and it gives good results according to Díaz-Uriarte and De Andres [101]. Thus, we defined the minimum node size as 5.

Moreover, as in the before mentioned models we set the lag order as 12. Therefore, the model took the values of 12 previous months as inputs and predicted one step ahead. To develop an RF model, we used the $randomForest()$ function written in the $randomForest$ package to fit the model.

Hyperparameter Tuning and Model Selection

The default value of $ntree$ is set 500 by $randomForest()$ function and according to Liaw and Wiener [102], it is sufficient for model development. Another study conducted by Probst and Boulesteix [103] depicts that RF has the highest performance with 100 trees.

In line with these studies, we tried to find the $mtry$ and $ntree$ values which provide the optimal model by using 10-fold cross validation technique repeating 3 times. We tested $mtry$ values from 1 to 12 which is the lag order showing total number of predictor variables and $ntree$ values from 100 to 500 in steps of 100.

In order to tune the RF model, we used $train()$ function which trains the model for various $mtry$ values and calculates the validation errors based on RMSE through cross validation. We repeated this process for different $ntree$ values and the function provided the best possible number of parameters for each number of trees which is shown on the following table.

Table 3.8: RF Best Performing Hyperparameters

ntree	mtry	RMSE
100	9	0.08765
200	7	0.08761
300	7	0.08746
400	7	0.08751
500	7	0.08733

The results on Table 3.8 indicates that the model with 500 trees and 7 randomly chosen predictors among 12 is the optimal one having the least validation error (0.08733). However, considering that error value of the model with 300 trees and 7 predictors is very close to those with 500 trees, the computational times for model construction may take an important role on model selection. The model with 300 trees and 7 predictors requires 1 minute and 5 seconds while the model with 500 trees and 7 predictors is conducted in 1 minute and 51 seconds. As the computational time for the model with 500 trees is not quite higher than the other, we selected the model with the lowest error (500 trees and 7 predictors).

The graph of the RMSE values for the various $mtry$ and 500 trees is demonstrated in Figure 3.11.

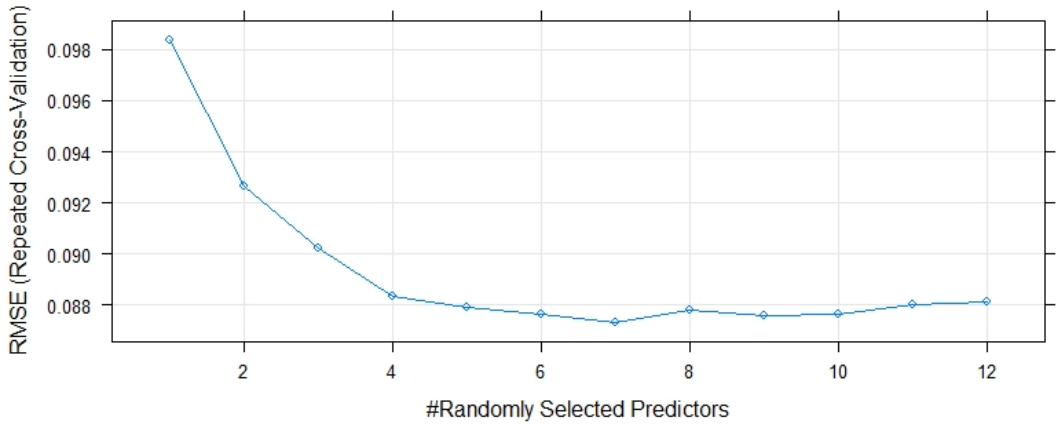


Figure 3.11: RMSE for randomly selected predictors with 500 trees

After the tuning step, we re-trained the model on the entire train set with the optimal hyperparameters and the learning curve of the RF with 500 trees was obtained as shown on Figure 3.12. The curve indicated that the model learned the data with less and less error when adding more trees up to 500.

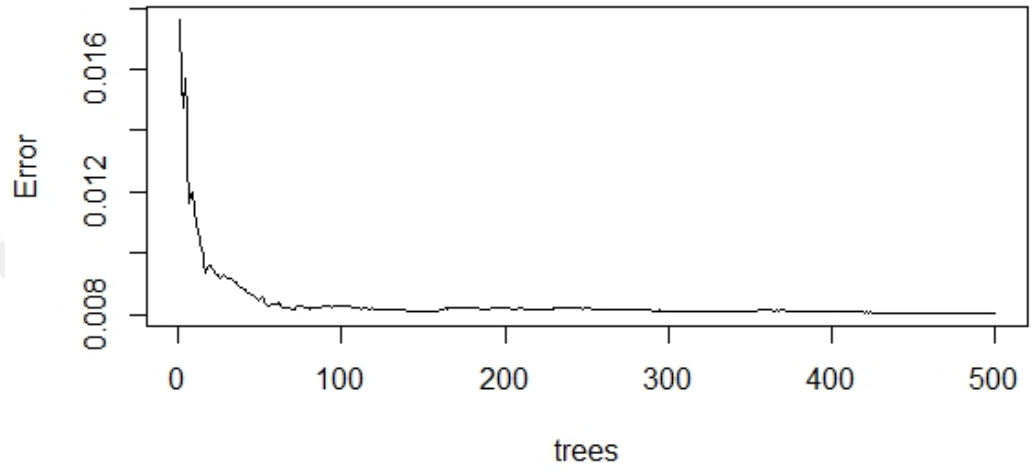


Figure 3.12: The learning curve for RF with 500 trees

After predicting the training and test observations with the last model, we transformed the predicted values back to the original data scale. Time plot for the predictions and the actual values of the training set and the model accuracy metrics are as follows:

Table 3.9: Model Performance for RF Model

Cleaned Data	**RMSE**	**MAPE**
Training Set	308.5109	2.8485
Test Set	841.2527	5.1740

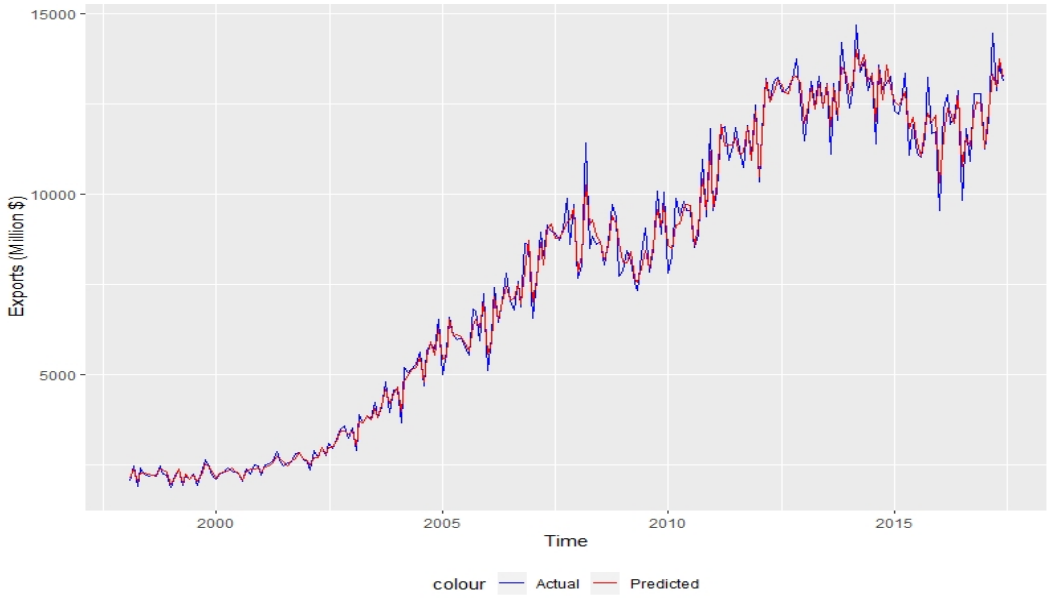


Figure 3.13: Time plot of the actual and predicted values by RF model

3.3.4 Model Comparison

We introduced the model development and selection process for each applied models before this section. In this section, we compared the models' accuracies on the test set, namely the out-of-sample set. Considering the main goal of time series analysis in many fields is to predict the future which is essential in decision making, the models are evaluated on an unseen set to see their generalization ability and forecasting performances. The model training performance is also very important to see how the model captures the structure of series; however, any model which fits the data may not demonstrate a good performance on a new set and cannot be accurate for further predictions. Therefore, we tested the models on a new set.

As we stated before, we obtained one step forecasts using the test observations and computed the accuracy metrics through $accuracy()$ function under the $forecast$ package. The resulting table is as follows:

Table 3.10: Generalization Performances of the Models

Test Set	**RMSE**	**MAPE**	**Computational Time (seconds)**
ETS	591.6126	3.4089	0.98
SVM	785.2213	4.6774	0.02
RF	841.2527	5.1740	0.35
LSTM	886.9615	5.3165	295.85
ARIMA	985.7293	5.7633	0.45

In Table 3.10, models were lined up with respect to their performance metrics. Based on both RMSE and MAPE results, the model having the least generalization error is ETS model even it has autocorrelated model residuals. Bearing in mind that the forecasts produced from a model with autocorrelated residuals are not wrong and the ETS is one of the proven and most widely used method in time series modelling and forecasting, being the best model is not a surprising result. Indeed, one of the broad study on a time series in the M3 Competition which compared the 8 ML models including LSTM, SVR with the statistical ones in one step forecasting, ETS outperformed all the applied models [85]. On the other hand, in this research some of the machine learning approaches are promising and have been studied recently in many time series analysis. For our univariate monthly exports series, it was observed that SVM model is the most promising as a forecasting tool and the most competitive with the classical ETS model.

In Table 3.11 and Figure 3.14 , one step forecasts on the test data and the actual series are shown to compare the model generalization performances.

Table 3.11: Actual Observations and One Step Forecasts (closests are bold) of the Models

Date	**Actual**	**ARIMA**	**ETS**	**LSTM**	**SVM**	**RF**
1.07.2017	12.612	**12.518**	13.182	11.827	11.730	12.188
1.08.2017	13.248	12.875	11.986	13.602	**13.468**	14.325
1.09.2017	11.810	12.825	13.081	12.804	**12.667**	12.838
1.10.2017	13.913	12.963	14.236	13.034	**14.074**	13.320
1.11.2017	14.188	13.167	**14.146**	13.604	13.727	13.194
1.12.2017	13.846	13.547	13.177	**13.726**	13.432	13.690
1.01.2018	12.434	13.659	**12.342**	12.940	12.582	13.215
1.02.2018	13.148	13.610	12.899	13.510	**13.023**	13.364
1.03.2018	15.553	13.710	**15.090**	14.168	14.408	14.395
1.04.2018	13.847	13.925	**13.853**	13.803	13.747	14.172
1.05.2018	14.257	**14.274**	14.838	14.381	14.717	14.758
1.06.2018	12.924	14.211	**13.360**	14.027	13.937	13.871
1.07.2018	14.049	**14.061**	13.775	13.460	13.722	13.858
1.08.2018	12.332	14.204	**12.729**	14.393	14.093	14.099
1.09.2018	14.398	13.141	13.341	12.836	13.184	**13.476**
1.10.2018	15.677	14.265	15.223	15.164	15.139	**15.879**
1.11.2018	15.492	14.909	**15.386**	15.061	14.574	14.891
1.12.2018	13.810	15.091	**14.395**	14.780	14.591	15.169
1.01.2019	13.180	14.232	**13.164**	13.884	13.824	14.044
1.02.2019	13.572	14.271	**13.730**	14.222	14.141	13.740
1.03.2019	15.463	**15.170**	15.910	14.641	14.905	14.408
1.04.2019	14.463	14.414	14.332	14.012	**14.449**	14.750
1.05.2019	15.944	14.721	**15.341**	14.812	14.685	14.592
1.06.2019	14.670	14.374	14.096	**14.603**	14.351	14.763
1.07.2019	15.137	**15.227**	14.852	15.278	15.017	15.879
1.08.2019	12.503	14.348	13.755	14.247	13.862	**13.710**
1.09.2019	14.436	15.154	14.193	13.904	15.467	**14.263**

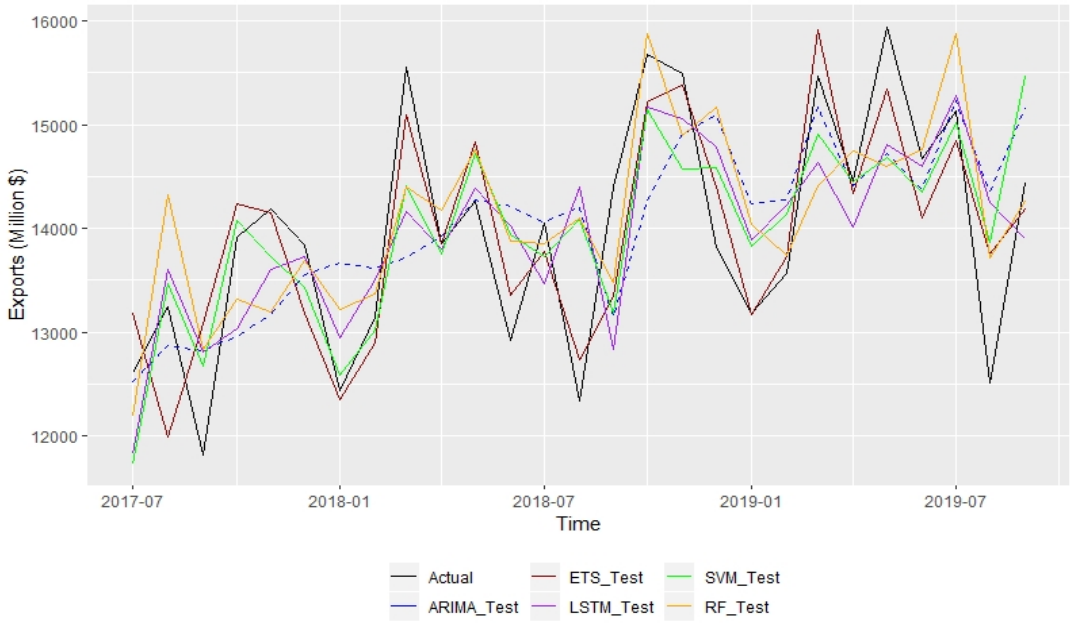


Figure 3.14: Time plot of the actual and one-step forecasts on test set by applied models

Although ARIMA models are another widely used forecasting models, in our case ARIMA is the worst performing model. As it can be indicated from the time plot of the models in Figure 3.14, the blue line representing the ARIMA model cannot capture the fluctuations clearly in the test series. This is the same in its training performance as seen in Figure 3.5 under Section 3.2.1. The proposed ARIMA model captures the trend in the actual series but it seems to perform poorly in capturing the seasonal structure. Even if the formal test did not show a significant existence of the lack of fit, the autocorrelation in residuals is significant at some lags which were neglected. Besides, because of the linearity restriction, ARIMA models may not be that accurate for a real-world data. Thus, these might be resulted in a poor performance and other modelling procedures can be applied such as fitting ARIMA model on a seasonally adjusted data through STL decomposition or a hybrid model with a model good at non-linear data.

Despite of the fact that ARIMA model fitted the data better than LSTM model, the out-of-sample error of the LSTM is less than the ARIMA model indicating better generalization ability. Finally, the RF model accuracy is observed as an average model

such that its out-of-sample performance is neither as good as SVM nor as poor as the LSTM model. RF models in time series forecasting is one of the rising stars due to its being simple, having fewer hyperparameters to tune, dealing with small sized samples and handling with complex structures [104]. In our series, RF performed best in fitting the data as it can be seen on the Table 3.9 under Section 3.3.3; however, its poor generalization ability took it away from being selected as the best model.

In addition to the error metrics, we recorded computational time of models which can be seen on Table 3.10. According to the results, SVM requires least time, just 0.02 seconds, to construct the model. With this regard, having the least computational time besides the best forecasting performance among ML models makes SVM models much more attractive in forecasting. Furthermore, LSTM has the highest computational time, 295.85 seconds, because of having much more complex structure than the others. In this context, except being the worst performing model among ML models, its high computational time puts LSTM much more behind the others in model selection.

CHAPTER 4

CONCLUSION AND FUTURE WORKS

In this study, we aimed at finding the most promising model for Turkey's monthly exports data by comparing the most widely accepted classical models such as ARIMA and ETS with recently the most popular machine learning models such as LSTM, SVR and RF. The main purpose is not to make a forecast for the future export values but to constitute a base for the future studies which is the main contribution of this thesis.

At the first step, we studied on the anomalies in the data and cleaned them before the model development stages. We compared the cleaned and original data results via ETS and ARIMA models and concluded that the cleaning process improved the model performance on both in-sample and out-of-sample sets. Then, we applied the following methods on the cleaned series. We used automatized ETS and ARIMA model tools in R and took their results as the benchmark for ML methods.

Recently, it has been an undeniable fact that the ML algorithms have been utilized in many time series researches owing to their good performances in complex structures and not being assumption based. In this context, we applied LSTM, SVR and RF models. If the main struggle in classical approaches is to meet the assumptions, it is the parameter tuning in ML approaches. Along this line, LSTM is the main challenging model in this study, considering of its tuning parameters and high potential to overfit the data with the memory block structures. Due to these issues and the high computational time, we considered a simple LSTM model which could fit the data without an overfitting problem through manual search. Therefore, the resulting model may not be the best but qualified as competitive and promising for the future study and it may be improved further with a more complex structure allocating much

more time and attention.

Furthermore, it should be pointed out that the ML models are developed based on the trial and error process. Accordingly, the model which provides better training and generalization performance is chosen for the forecasting purpose. However, the model performances and the selected model may vary according to various settings such as different preprocessing methods, different lag orders, different ratios for the train and hold-out set splitting, different cross-validation techniques, different forecasting techniques, horizons and different data structures etc. All these factors acting as hyperparameters in the ML methods should be considered carefully and applied on the exports data with various combinations.

To sum up the results of our research, ETS outperformed all other models we applied. While SVR was the most accurate among the ML models, LSTM was the worst performing model; however, it was observed that it outperformed the ARIMA model. As being one of the complex neural network model, LSTM's better forecasting performance over ARIMA is the thing that one would expect and this should pave the way for further analysis on LSTM models. Besides, RNN models are stronger in trend forecasting and the seasonality might be handled constructing a hybrid model with seasonal ARIMA [105] or seasonal component might be removed by STL decomposing and deseasonalizing the series like in the study of Makridakis et $al.$ [85].

Another interesting result reported from the RF model. RF depicted a better model fitting than other models, even ETS. However, it is also a fact that it has worse forecasting performance than ETS and SVR. To improve the generalization ability of RF, different preprocessing methods such as data scaling or normalization can be applied. Moreover, a hybrid model comprised of LSTM for trend forecasting and RF for the remaining part can be constructed to obtain a better prediction results.

As summarized in the introduction, while some studies were conducted on the univariate exports data, some of them included independent variables to the models. In line with this, adding the exogenous variables of Turkish exports to our models may be another future research for the model improvement. ML methods can also be utilized in the feature selection process while determining the factors affecting exports in Turkey.

Additionally, the studies in the literature, mentioned in Section 1.1, depict that ARIMA and exponential smoothing models are widely used for exports data and compared with some machine learning models such as ANN, ERNN, MLP. Despite the fact that data is not same for each study, MAPE results of the models applied on Turkish exports data provide a general evaluation for our study. The MAPE values obtained from the studies are generally higher (above 10) compared to our model's performances (below 6). One possible reason is the forecast horizon considering that some studies used the estimated models for longer forecasting purposes. Although some of them used the selected models for test estimation which produces one-step forecasts as in our study, the test accuracy results show that our models have again better forecasting ability. One reason of having less error metrics compared to the other studies can be the anomaly/outlier cleaning process which improves the models' forecasting performance.

Besides, a linear model, which is successful in a linear time series, may not perform well in a non-linear structure and a model good at non-linear cases may not show a good performance on linear component [106]. In our study, while non-linear models showed better forecasting performances on our data, ARIMA as a linear-model performed poorly. Considering this issue, constructing a hybrid model by combining a non-linear model to capture the non-linear structure and a linear model like ARIMA to model linear structure in our data may be another future research.

Further researches following this study can focus on the ETS model which has autocorrelated residuals. Although forecasts from this model are acceptable, autocorrelation between the residuals indicates that there is still something to do with this model to improve and to obtain better forecasts. Fixing autocorrelation problem is challenging; however, a solution may be applying ARIMA model to the errors to capture the information left in the errors then obtaining new errors assumed to be white noise.

REFERENCES

[1] J. B. Ahire, "The Artificial Neural Networks Handbook: Part 4." Online, 2018. https://medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e, (Accessed on 05/04/2020).

[2] C. Olah, "Understanding LSTM Networks." Online, Aug. 2015. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, (Accessed on 09/05/2020).

[3] S. Bouktif, A. Fiaz, A. Ouni, and M. A. Serhani, "Single and multi-sequence deep learning models for short and medium term electric load forecasting," *Energies*, vol. 12, jan 2019.

[4] S. K. Lahiri and K. C. Ghanta, "The support vector regression with the parameter tuning assisted by a differential evolution technique: Study of the critical velocity of a slurry flow in a pipeline," *Chem. Ind. Chem. Eng. Q.*, vol. 14, no. 3, pp. 191–203, 2008.

[5] A. Chakure, "Random Forest Regression." Online, June 2019. https://towardsdatascience.com/random-forest-and-its-implementation-71824ced454f, (Accessed on 13/05/2020).

[6] D. Ziganto, "Model Tuning (Part 2 - Validation Cross-Validation)." Online. https://dziganto.github.io/cross-validation/datascience/machinelearning/modeltuning/python/Model-Tuning-with-Validation-and-Cross-Validation/, (Accessed on 21/04/2020).

[7] J. Toutouh, J. Arellano, and E. Alba, "BiPred: A Bilevel Evolutionary Algorithm for Prediction in Smart Mobility," *Sensors (Basel).*, vol. 18, nov 2018.

[8] E. Tokucu and A. Yüce, "Türkiye'nin İhracat Performansının 1980 Sonrası

Dönemde Gelişimi ve İhracatın Artırılmasında Uluslararası Pazarlama İnovasyonunun Rolü [development of turkey's export performance after 1980 and the role of international marketing innovation on increasing exports]," *Trak. Univ. E-Journal*, vol. 2, no. 1, pp. 47–75, 2003.

[9] S. Azgun, *Dış Ticaret ve Rekabet Gücü [Foreign Trade and Competitiveness]*. Ekin, 2017.

[10] C. Hepaktan, "Türkiye'nin Dönüşüm Sürecinde Dış Ticaret Politikaları [foreign trade policy in turkey's transformation process]," tech. rep., 2008.

[11] J.-K. Jung, M. Patnam, and A. Ter-Martirosyan, "An Algorithmic Crystal Ball: Forecasts-based on Machine Learning." 2018.

[12] G. P. Zhang, "Business Forecasting with Artificial Neural Networks," *Neural Networks Bus. Forecast.*, pp. 1–22, 2004.

[13] N. Ersen, l. Akyuz, and B. C. Bayram, "The forecasting of the exports and imports of paper and paper products of Turkey using Box-Jenkins method," *Eurasian J. For. Sci.*, vol. 7, pp. 54–65, feb 2019.

[14] U. Başer, M. Bozoğlu, N. Alhas Eroğlu, and B. Kilic Topuz, "Forecasting Chestnut Production and Export of Turkey Using ARIMA Model," *Turkish J. Forecast.*, vol. 2, no. 2, pp. 27–33, 2018.

[15] A. Demir, O. Ozmen, and A. Rashid, "An Estimation of Turkey's Export Loss to Iraq," *Procedia - Soc. Behav. Sci.*, vol. 150, pp. 1240–1247, 2014.

[16] M. AKAL, "Estimation of Hazelnut Export Of Turkey And Forecast Accuracies," *ZKÜ Social Science*, vol. 5, no. 10, pp. 77–96, 2009.

[17] H. Uysal and S. Karabat, "Forecasting and Evaluation for Raisin Export in Turkey," *BIO Web Conf. 9*, vol. 9, 2017.

[18] M. Karahan, "Yapay Sinir Ağları İle İhracat Miktarlarının Tahmini: ARIMA ve YSA Metodunun Karşılaştırmalı Analizi," *Ege Acad. Rev.*, vol. 15, no. 2, pp. 165–172, 2015.

[19] A. Ozbek and M. Akalin, "The Prediction of Turkey's Denim Trousers Export to Germany with Ann Models," vol. 21, pp. 313–322, 2011.

[20] A. Farooqi, "ARIMA model building and forecasting on imports and exports of Pakistan," *Pakistan J. Stat. Oper. Res.*, vol. 10, no. 2, pp. 157–168, 2014.

[21] T. Alam, "Forecasting exports and imports through artificial neural network and autoregressive integrated moving average," *Decis. Sci. Lett.*, vol. 8, no. 3, pp. 249–260, 2019.

[22] S. Ghosh, "Forecasting Cotton Exports in India using the ARIMA model," *Amity J. Econ.*, vol. 2, no. 2, pp. 36–52, 2017.

[23] I. Wohl and J. Kennedy, "Neural Network Analysis of International Trade." 2018.

[24] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection," *ACM Comput. Surv.*, vol. 41, pp. 1–22, 2009.

[25] V. Kotu and B. Deshpande, "Anomaly Detection," in *Data Sci.*, ch. 13, pp. 447–465, Elsevier, 2019.

[26] R. J. Hyndman and G. Athanasopoulos, "Time Series Decomposition," in *Forecast. Princ. Pract.*, ch. 6, OTexts, 2018.

[27] D. Gerbing, "Time Series Components," tech. rep., Portland State University, 2016.

[28] H. Gweon and A. I. Mcleod, *Seasonal Decomposition for Geographical Time Series using Nonparametric Regression*. PhD thesis, The University of Western Ontario, 2013.

[29] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "STL: A Seasonal-Trend Decomposition Procedure Based on Loess," *J. Off. Stat.*, vol. 6, no. 1, pp. 3–73, 1990.

[30] B. Rosner, "Percentage Points for a Generalized ESD Many-Outlier Procedure," *Technometrics*, vol. 25, no. 2, pp. 165–172, 1983.

[31] J. Hochenbaum, O. S. Vallis, and A. Kejariwal, "Automatic Anomaly Detection in the Cloud Via Statistical Learning," 2017.

[32] W. contributors, "Box-Jenkins Method," 2020. (Accessed on 13/05/2020).

[33] S. Siami Namin and A. Siami Namin, "Forecasting Economic and Financial Time Series: ARIMA Vs. LSTM," tech. rep., 2018.

[34] T. J. Fisher, "Testing Adequacy of ARMA Models using a Weighted Portmanteau Test on the Residual Autocorrelations." 2011.

[35] R. J. Hyndman and G. Athanasopoulos, "ARIMA modelling in R," in *Forecast. Princ. Pract.*, ch. 8, OTexts, 2018.

[36] W. contributors, "Autoregressive integrated moving average," 2020.

[37] NCSS Statistical Software, "The Box-Jenkins Method." Online. `https://www.ncss.com/`, (Accessed on 25/04/2020).

[38] R. J. Hyndman, "Box-Jenkins modelling," tech. rep., 2001.

[39] G. E. P. Box and D. Cox, "An Analysis of Transformations," *J. R. Stat. Soc.*, vol. 26, no. 2, pp. 211–252, 1964.

[40] P. C. Phillips and P. Perron, "Testing for a unit root in time series regression," *Biometrika*, vol. 75, no. 2, pp. 335–346, 1988.

[41] D. Kwiatkowski, P. C. Phillips, P. Schmidt, and Y. Shin, "Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?," *J. Econom.*, vol. 54, no. 1-3, pp. 159–178, 1992.

[42] S.Hylleberg, R.F.Engle, C.W.J.Granger, and B.S.Yoo, "Seasonal Integration and Cointegration," *J. Econom.*, vol. 44, no. 1-2, pp. 215–238, 1990.

[43] D. R. Osborn, A. P. Chui, J. P. Smith, and C. R. Birchenhall, "Seasonality and the Order of Integration for Consumption," *Oxf. Bull. Econ. Stat.*, vol. 50, no. 4, pp. 361–377, 1988.

[44] A. Maleki, S. Nasseri, M. S. Aminabad, and M. Hadi, "Comparison of ARIMA and NNAR Models for Forecasting Water Treatment Plant's Influent Characteristics," *KSCE J. Civ. Eng.*, vol. 22, pp. 3233–3245, sep 2018.

[45] A. S. S. Shapiro and M. B. Wilk, "Biometrika Trust An Analysis of Variance Test for Normality," *Oxford Journals*, vol. 52, no. 3, pp. 591–611, 1965.

[46] X. Wang, K. Smith, and R. Hyndman, "Characteristic-Based Clustering for Time Series Data," *Data Min. Knowl. Discov.*, vol. 13, no. 3, pp. 335–364, 2006.

[47] U. Pritzsche, *Benchmarking of Classical and Machine-Learning Algorithms (with special emphasis on Bagging and Boosting Approaches) for Time Series Forecasting*. PhD thesis, 2014.

[48] S. Panigrahi and H. S. Behera, "A Hybrid ETS–ANN Model for Time Series Forecasting," *Eng. Appl. Artif. Intell.*, vol. 66, no. November, pp. 49–59, 2017.

[49] R. J. Hyndman and G. Athanasopoulos, "Innovations State Space Models for Exponential Smoothing," in *Forecast. Princ. Pract.*, ch. 7, OTexts, 2018.

[50] E. Romano, R. Revetria, G. Guizzi, and C. Silvestri, "A Comparison of Forecast Models to Predict Weather Parameters New Quantitative Management Models To Increase Healthcare Efficiency View Project A Comparison Of Forecast Models To Predict Weather Parameters," 2015.

[51] M. Snipes and D. C. Taylor, "Model selection and Akaike Information Criteria: An example from wine ratings and prices," *Wine Econ. Policy*, vol. 3, no. 1, pp. 3–9, 2014.

[52] F. Hansson and J. Rostami, *Time Series Forecasting of House Prices: An Evaluation of a Support Vector Machine and a Recurrent Neural Network with LSTM cells*. PhD thesis, Uppsala University, 2019.

[53] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, Berlin, Heidelberg, 2012.

[54] R. UNLU, "A Comparative Study of Machine Learning and Deep Learning for Time Series Forecasting: A Case Study of Choosing the Best Prediction Model for Turkey Electricity Production," *Süleyman Demirel Üniversitesi Fen Bilim. Enstitüsü Derg.*, vol. 23, pp. 359–370, august 2019.

[55] C. J. Lu, T. S. Lee, and C. C. Chiu, "Financial Time Series Forecasting Using Independent Component Analysis and Support Vector Regression," *Decis. Support Syst.*, vol. 47, pp. 115–125, may 2009.

[56] F. E. H. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," *Omega*, vol. 29, pp. 309–317, 2001.

[57] W. He, Z. Wang, and H. Jiang, "Model Optimizing and Feature Selecting for Support Vector Regression in Time Series Forecasting," *Neurocomputing*, vol. 72, pp. 600–611, dec 2008.

[58] C. Cortes and V. Vapnik, "Support-Vector Networks," tech. rep., 1995.

[59] J. Guajardo, R. Weber, and J. Miranda, "A Forecasting Methodology Using Support Vector Regression and Dynamic Feature Selection," *J. Inf. Knowl. Manag.*, vol. 5, no. 4, 2006.

[60] K.-R. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik, "Using Support Vector Machines for Time Series Prediction," in *Adv. kernel methods Support vector Learn.*, DE GRUYTER, 1999.

[61] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, pp. 5–32, 2001.

[62] J. Nilsson and M. Kalmár, *The art of forecasting-an analysis of predictive precision of machine learning models*. PhD thesis, Uppsala University, 2016.

[63] A. Cutler, D. R. Cutler, and J. R. Stevens, "Random Forests," in *Ensemble Mach. Learn. Methods Appl.*, pp. 157–175, Boston, MA: Springer US, 2012.

[64] E. C. Merkle and V. A. Shaffer, "Binary recursive partitioning: Background, methods, and application to psychology," *Br. J. Math. Stat. Psychol.*, vol. 64, pp. 161–181, feb 2011.

[65] A. Cutler, "Random Forests for Regression and Classification," tech. rep., 2010.

[66] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. Springer, 2017.

[67] T. Yiu, "Understanding Random Forest." Online, June 2019. `https://towardsdatascience.com/understanding-random-forest-58381e0602d2`, (Accessed on 20/04/2020).

[68] V. Jatana, "Hyperparameter Tuning," tech. rep., 2019.

[69] M. Claesen, K. U. Leuven, and B. L. R. De Moor, "Hyperparameter Search in Machine Learning Quantum entanglement distillation View project Physical modeling synthesis View project," 2015.

[70] M. Dei, "Hyperparameter Tuning Explained — Tuning Phases, Tuning Methods, Bayesian Optimization, and Sample Code." Online, Dec. 2019. `https://towardsdatascience.com/hyperparameter-tuning-explained-d0ebb2ba1d35`, (Accessed on 04/05/2020).

[71] C. Bergmeir, R. J. Hyndman, and B. Koo, "A Note on the Validity of Cross-Validation for Evaluating Time Series Prediction." 2015.

[72] N. Silver, *The Signal and the Noise: Why So Many Predictions Fail – But Some Don't*. New York: Penguin, 2012.

[73] C. B. Smart, "The Signal and the Noise in Cost Estimating," in *Int. Train. Symp.*, 2016.

[74] X. Ying, "An Overview of Overfitting and its Solutions," *J. Phys. Conf. Ser.*, vol. 1168, no. 2, 2019.

[75] S. Raschka, "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning," nov 2018.

[76] D. Berrar, "Cross-validation," in *Ref. Modul. Life Sci.*, vol. 1-3, pp. 542–545, Elsevier, jan 2018.

[77] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. Springer, New York, NY, 2013.

[78] R. J. Hyndman and G. Athanasopoulos, "Forecasting on Training and Test Sets," in *Forecast. Princ. Pract.*, ch. 12, OTexts, 2018.

[79] R. Adhikari and R. K. Agrawal, *An Introductory Study on Time Series Modeling and Forecasting*. LAP Lambert Academic Publishing, 2013.

[80] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *Int. J. Forecast.*, vol. 22, no. 4, 2005.

[81] S. Keskin, "Impacts of Customs Union With The European Union to the Turkish Economy," *Int. J. Econ. Financ.*, vol. 1, no. 2, 2009.

[82] S. D. Vesterbye and M. S. Akman, "A Modernized EU-Turkey Customs Union," tech. rep., TEPAV, 2017.

[83] I. Enser, *Global Financial Crisis and Its Effects on Turkey*. PhD thesis, Dokuz Eylül University, 2013.

[84] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*. 2018.

[85] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and Machine Learning forecasting methods: Concerns and ways forward," *PLoS One*, vol. 13, mar 2018.

[86] S. Bouktif, A. Fiaz, A. Ouni, and M. A. Serhani, "Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches," *Energies*, vol. 11, no. 7, 2018.

[87] G. P. Zhang and M. Qi, "Neural network forecasting for seasonal and trend time series," *Eur. J. Oper. Res.*, vol. 160, no. 2, pp. 501–514, 2005.

[88] J. Brownlee, "Understand the Impact of Learning Rate on Neural Network Performance." Online, Jan. 2019. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/, (Accessed on 19/05/2020).

[89] J. Brownlee, "How to Use Data Scaling Improve Deep Learning Model Stability and Performance." Online, Feb. 2019. https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/, (Accessed on 15/05/2020).

[90] J. Leonel, "Hyperparameters in Machine / Deep Learning." Online, Apr. 2019. https://medium.com/@jorgesleonel/hyperparameters-

in-machine-deep-learning-ca69ad10b981 (Accessed on 04/03/2020).

[91] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[92] I. Kaastra and M. Boyd, "Designing a neural network for forecasting financial and economic time series," *Elsevier*, vol. 10, no. 3, pp. 215–236, 1996.

[93] R. Gandhi, "A Look at Gradient Descent and RMSprop Optimizers." Online, 2018. https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b, (Accessed on 01/06/2020).

[94] A. Kathuria, "Introduction to optimization in deep learning: Momentum, RMSProp and Adam." Online, 2018. https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/(Accessed on 01/06/2020).

[95] M. K. Okasha, "Using Support Vector Machines in Financial Time Series Forecasting," *Int. J. Stat. Appl.*, vol. 4, no. 1, pp. 28–39, 2014.

[96] B. T. Ojemakinde, *Support Vector Regression for Non-Stationary Time Series*. PhD thesis, University of Tennessee - Knoxville, 2006.

[97] K. J. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, no. 1-2, pp. 307–319, 2003.

[98] M. Tilgner, "Time series forecasting with random forest." Online, 2019. https://www.r-bloggers.com/time-series-forecasting-with-random-forest/, (Accessed on 20/04/2020).

[99] P. Probst, B. Bischl, and A.-L. Boulesteix, "Tunability: Importance of Hyperparameters of Machine Learning Algorithms," feb 2018.

[100] P. Probst, M. Wright, and A.-L. Boulesteix, "Hyperparameters and Tuning Strategies for Random Forest," apr 2018.

[101] R. Díaz-Uriarte and S. Alvarez de Andrés, "Gene selection and classification of microarray data using random forest," *BMC Bioinformatics*, vol. 7, pp. 1–13, 2006.

[102] A. Liaw and M. Wiener, "Classification and Regression by randomForest," *R News*, vol. 3, no. December 2002, pp. 18–22, 2003.

[103] P. Probst and A. L. Boulesteix, "To tune or not to tune the number of trees in random forest," *J. Mach. Learn. Res.*, vol. 18, no. 2001, pp. 1–8, 2018.

[104] H. Tyralis and G. Papacharalampous, "Variable selection in time series forecasting using random forests," *Algorithms*, vol. 10, dec 2017.

[105] O. Fathi, "Time series forecasting using a hybrid ARIMA and LSTM model," tech. rep.

[106] K. H. Topal, S. Kizilarslan, H. Bulbul, and E. Ç. Akay, "Forecasting of Turkish housing price index: ARIMA, random forest, ARIMA-random forest," *Pressacademia*, vol. 10, pp. 7–11, dec 2019.