PERFORMANCE COMPARISON OF MACHINE LEARNING METHODS AND TRADITIONAL TIME SERIES METHODS FOR FORECASTING

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

BY

OZANCAN ÖZDEMIR

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

AUGUST 2020

Approval of the thesis:

PERFORMANCE COMPARISON OF MACHINE LEARNING METHODS AND TRADITIONAL TIME SERIES METHODS FOR FORECASTING

submitted by **OZANCAN ÖZDEMIR** in partial fulfillment of the requirements for the degree of **Master of Science in Statistics Department, 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
Department of Statistics, METU _____

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

Prof. Dr. Esin Firuzan
Department of Statistics, Dokuz Eylül University _____

Date:

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: Ozancan Özdemir

Signature :

ABSTRACT

PERFORMANCE COMPARISON OF MACHINE LEARNING METHODS AND TRADITIONAL TIME SERIES METHODS FOR FORECASTING

Özdemir, Ozancan
M.S., Department of Statistics
Supervisor: Assoc. Prof. Dr. Ceylan Yozgatlıgil

August 2020, 147 pages

One of the main objectives of the time series analysis is forecasting, and for this purpose both Machine Learning methods and statistical methods have been proposed in the literature [1]. In this study, we use and compare some of these approaches in time series modelling and forecasting. In addition to traditional forecasting methods for time series data set which are namely Naive Method, Seasonal Naive Method, ARIMA, SARIMA, Exponential Smoothing, TBATS, Bayesian Exponential Smoothing Models with Trend Modifications and STL Decomposition, the forecasts are also obtained by using seven different machine learning methods. These methods are Random Forest, Support Vector Regression, XGBoosting, Bayesian Neural Network, Recurrent Neural Network, Long Short Term Memory Neural Network and Feed Forward Neural Network. It is also known that time series generally contain both linear and nonlinear patterns. In order to deal with this mixture data structure, a hybrid methodology which combines linear and nonlinear components was proposed by Zhang. According to him, predicted values of a time series can be obtained by summing both linear and nonlinear components [2]. In this study, hybrid models are

constructed by using machine learning methods for nonlinear pattern and statistical methods for linear pattern. Therefore, the forecasts are also obtained using hybrid models. The data set selected proportionally from different time frequencies in M4 Competition is used in this study. After observing the results of studies, the performance and impact of all methods are discussed. At the end of this discussion, most of the best models are mainly selected from machine learning methods for forecasting in this study. It is also seen that the forecasting performance of the model depends on both time frequency and forecast horizon. Lastly, the study proves that the hybrid approach is not always the best forecasting model for time series.

Keywords: Time series analysis, Forecast, Hybrid method, Machine Learning, M4 Competition

ÖZ

GELENEKSEL ZAMAN SERİSİ YÖNTEMLERİ VE MAKİNE ÖĞRENMESİ YÖNTEMLERİNİN ÖNGÖRÜ PERFORMANS KARŞILAŞTIRMASI

Özdemir, Ozancan
Yüksek Lisans, İstatistik Bölümü
Tez Yöneticisi: Doç. Dr. Ceylan Yozgatlıgil

Ağustos 2020 , 147 sayfa

Zaman serisi analizin en temel amaçlarından biri tahminleme yapmaktır. Literatürde tahminleme için hem istatistiksel hem de makine öğrenmesi modellerinin kullanıldığı ve önerildiği görülmektedir [1]. Bu çalışma, bu yaklaşımlardan bazılarının zaman serilerinin modellenmesi ve tahminlenmesi yönünden karşılaştırmalarını içermektedir. Naive, mevsimel naive, otoregresif tamamlanmış hareketli ortalama, mevsimsel otoregresif tamamlanmış hareketli ortalama,üstel düzleştirme, Bayesçi üstel düzleştirme, TBATS ve STL modelleri gibi istatistiksel zaman serisi modellerine ek olarak, rassal ormanlar, destek vektör makinesi, ekstrem gradyan arttırma, Bayesçi yapay sinir ağları, tekrarlayan yapay sinir ağları, uzun kısa süreli bellek ve ileri beslemeli sinir ağları modelleri de kullanılarak çalışmadaki zaman serileri tahminlenmiştir. Zaman serileri genel yapılarında hem doğrusal hem doğrusal olmayan içerikleri barındırmaktadır. Bu yapıyla ilgilenmek için, Zhang bir hibrit yaklaşım önermiştir [2]. Bu yaklaşıma göre, tahmin değerleri doğrusal ve doğrusal olmayan içeriklerin toplamından elde edilecektir. Yukarıda sıralanan modellere ek olarak, doğrusal içerikler için is-

tatistiksel ,doğrusal olmayan içerikler için de makine öğrenmesi modellerinin kullanıldığı hibrit modeller de tahminleme için kullanılmıştır. Bu çalışmada kullanılan veri seti M4 Yarışmasında kullanılan veri setinde yer alan farklı zaman frekanslarına sahip serilerden sayılarına orantılı bir şekilde seçilerek oluşturulmuştur. Modellerin uygulanmasından sonra, modellerin tahmin performansları karşılaştırılmıştır. Bu karşılaştırma sonucunda, bu çalışmadaki en iyi tahmin performansına sahip modellerin çoğunlukla makine öğrenmesi metotları arasından seçildiği görülmüştür. Aynı zamanda modellerin tahmin performansının tahminlenen serinin zaman frekansına ve tahminin uzunluğuna bağlı olduğu görülmüştür. Son olarak, bu çalışma ile hibrit yaklaşımın zaman serileri için her zaman en iyi tahminleme metodu olmadığı kantılanmıştır.

Anahtar Kelimeler: Zaman Serisi Analizi, Tahmin, Hibrit Method, Makine Öğrenmesi, M4 Yarışması

To my family and health workers

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my advisor Assoc. Prof. Ceylan Talu Yozgatlıgil for her never ending guidance, encouragement, patience and support during this study. It was an honour for me to work with her. I am really lucky to have an opportunity of utilizing her experiences and knowledge. I strongly believe that this thesis could not be ended without her stimulating advices and motivational energy.

I would like to thank the examining committee members; Prof. Özlem İlk Dağ and Prof. Esin Firuzan for their valuable suggestions and comments.

I am really thankful to my great co-workers Petek Aydemir being also my lovely roommate, Berna Tuncer, Duygu Varol, Serenay Çakar, Sevilay Doğan, Orçun Oltulu and Onur Çamlı for helphing me whenever I need. Besides, I am really appreciated to my lovely friends Büşra Taş, Buket Coşkun, Ecem Ünal, Haydar Haluk Ceylan, Koray Akalın, Doğa Furkan Güneş, Onur Güler and Ayşe Türkkan for their precious supports and advices during this thesis study.

I would like to express my thanks to health workers who fight against COVID-19 at the cost of their own life. Also, I am really grateful to people who satisfy our needs during the pandemic period.

Finally, I would like to express my special thanks to my dear mother Sabriye Özdemir and my dear father Hasan Özdemir for their everlasting love and support during my life. Their love is the best motivational source for me.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xviii

LIST OF FIGURES xxii

LIST OF ABBREVIATIONS xxvi

CHAPTERS

1 INTRODUCTION 1

2 LITERATURE REVIEW 5

2.1 M-Competitions 5

2.1.1 M1-Competition 5

2.1.2 M2-Competition 6

2.1.3 M3 Competition 7

2.1.4 M4 Competition 7

2.1.5 M5 Competition 8

2.2 Related Studies 8

2.2.1 Macroeconomics and Microeconomics 8

2.2.2 Industry 9
2.2.3 Finance 10
2.2.4 Demography 11
3 METHODOLOGY 13
3.1 Simple Forecasting Methods 13
3.1.1 Naive Method 13
3.1.2 Seasonal Naive Method 14
3.2 Autoregressive Integrated Moving Average Model (ARIMA) 14
3.3 Seasonal Autoregressive Integrated Moving Average Model (SARIMA) 17
3.4 Exponential Smoothing Method (ETS) 17
3.4.1 Simple Exponential Smoothing 18
3.4.2 Holt's Exponential Smoothing 18
3.4.3 Holt Winters Exponential Smoothing 19
3.4.3.1 Holt Winters Additive Method 19
3.4.3.2 Holt Winters Multiplicative Method 20
3.5 Bayesian Exponential Smoothing Models with Trend Modifications . 21
3.5.1 LGT 22
3.5.2 SGT 23
3.5.3 S2GT 24
3.6 TBATS 25
3.7 STL Decomposition 27
3.8 Support Vector Machine 29
3.9 Random Forests 32

3.10 Extreme Gradient Boosting (XGBoost) 34
3.11 Artificial Neural Network 37
3.11.1 Feed-Forward Neural Network 40
3.11.2 Recurrent Neural Network 42
3.11.3 Long Short Term Memory 44
3.11.4 Bayesian Regularized Neural Network 45
3.12 Hybrid Methods 47
3.13 Performance Measures 48
3.13.1 Forecast Accuracy Measures 48
3.13.1.1 Symmetric Mean Absolute Performance Error (sMAPE) 48
3.13.1.2 Mean Absolute Scaled Error (MASE) 49
3.13.1.3 The Arithmetic Mean of sMAPE and MASE 49
3.13.2 Computational Time 50
4 ANALYSIS 51
4.1 Data Introduction 51
4.2 Data Preprocessing 53
4.3 Model Implementation 54
4.3.1 Statistical Models 55
4.3.1.1 ARIMA and SARIMA 55
4.3.1.2 ETS 56
4.3.1.3 TBATS 56
4.3.1.4 LGT and SGT 56
4.3.1.5 STL Decomposition 57

4.3.2 Machine Learning Models 58
4.3.2.1 Support Vector Machine 58
4.3.2.2 Random Forest 59
4.3.2.3 XGBoost 59
4.3.2.4 Feed-Forward Neural Network 59
4.3.2.5 Recurrent Neural Network 59
4.3.2.6 Long Short Term Memory 60
4.3.2.7 Bayesian Neural Network 60
4.3.3 Hybrid Models 61
4.4 Empirical Analysis 61
4.4.1 Yearly Series Analysis 62
4.4.1.1 Original Yearly Series Analysis 63
Statistical Models: 63
Machine Learning Models: 64
Hybrid Models: 66
Result: 66
4.4.1.2 Transformed Yearly Series Analysis 68
Statistical Models: 68
Machine Learning Models: 69
Hybrid Models: 70
Result: 71
4.4.2 Quarterly Series Analysis 73
4.4.2.1 Original Quarterly Series Analysis 74

Statistical Models: 75
Machine Learning Models: 76
Hybrid Models: 77
Result: 78
4.4.2.2 Transformed Quarterly Series Analysis 79
Statistical Models: 80
Machine Learning Models: 81
Hybrid Models: 82
Result: 83
4.4.3 Monthly Series Analysis 85
4.4.3.1 Original Monthly Series Analysis 86
Statistical Models: 86
Machine Learning Models: 87
Hybrid Models: 88
Result: 89
4.4.3.2 Transformed Monthly Series Analysis 91
Statistical Models: 91
Machine Learning Models: 92
Hybrid Models: 94
Result: 95
4.4.4 Weekly Series Analysis 97
4.4.4.1 Original Weekly Series Analysis 98
Statistical Models: 98

Machine Learning Models: 99
Hybrid Models: 100
Result: 101
4.4.4.2 Transformed Weekly Series Analysis 103
Statistical Models: 103
Machine Learning Models: 104
Hybrid Models: 106
Result: 107
4.4.5 Daily Series Analysis 109
4.4.5.1 Original Daily Series Analysis 110
Statistical Models: 110
Machine Learning Models: 111
Hybrid Models: 112
Result: 113
4.4.5.2 Transformed Daily Series Analysis 115
Statistical Models: 115
Machine Learning Models: 116
Hybrid Models: 117
Result: 118
4.4.6 Hourly Series Analysis 120
4.4.6.1 Original Hourly Series Analysis 121
Statistical Models: 121
Machine Learning Models: 122

Hybrid Models: 124

Result: 125

4.4.6.2 Transformed Hourly Series Analysis 126

Statistical Models: 127

Machine Learning Models: 128

Hybrid Models: 129

Result: 130

5 CONCLUSION AND FUTURE STUDIES 133

REFERENCES 139

LIST OF TABLES

TABLES

Table 4.1 Number of M4 series for each frequency and area 52

Table 4.2 Number of series used in the study with frequencies and forecast horizons 53

Table 4.3 The Forecasting Performance of Statistical Methods for Original Yearly Series 64

Table 4.4 The Forecasting Performance of Machine Learning Methods for Original Yearly Series 65

Table 4.5 The Forecasting Performance of Hybrid Methods for Original Yearly Series 66

Table 4.6 The Forecasting Performance of Statistical Methods for Transformed Yearly Series 69

Table 4.7 The Forecasting Performance of Machine Learning Methods for Transformed Yearly Series 70

Table 4.8 The Forecasting Performance of Hybrid Methods for Transformed Yearly Series 71

Table 4.9 The Forecasting Performance of Statistical Methods for Original Quarterly Series 75

Table 4.10 The Forecasting Performance of Machine Learning Methods for Original Quarterly Series 76

Table 4.11 The Forecasting Performance of Hybrid Methods for Original Quarterly Series 77

Table 4.12 The Forecasting Performance of Statistical Methods for Transformed Quarterly Series 80

Table 4.13 The Forecasting Performance of Machine Learning Methods for Transformed Quarterly Series 81

Table 4.14 The Forecasting Performance of Hybrid Methods for Transformed Quarterly Series 82

Table 4.15 The Forecasting Performance of Statistical Methods for Original Monthly Series 86

Table 4.16 The Forecasting Performance of Machine Learning Methods for Original Monthly Series 87

Table 4.17 The Forecasting Performance of Hybrid Methods for Original Monthly Series 88

Table 4.18 The Forecasting Performance of Statistical Methods for Transformed Monthly Series 92

Table 4.19 The Forecasting Performance of Machine Learning Methods for Transformed Monthly Series 93

Table 4.20 The Forecasting Performance of Hybrid Methods for Transformed Monthly Series 94

Table 4.21 The Forecasting Performance of Statistical Methods for Original Weekly Series 98

Table 4.22 The Forecasting Performance of Machine Learning Methods for Original Weekly Series 99

Table 4.23 The Forecasting Performance of Hybrid Methods for Original Weekly Series 101

Table 4.24 The Forecasting Performance of Statistical Methods for Transformed Weekly Series 104

Table 4.25 The Forecasting Performance of Machine Learning Methods for Transformed Weekly Series 105

Table 4.26 The Forecasting Performance of Hybrid Methods for Transformed Weekly Series 106

Table 4.27 The Forecasting Performance of Statistical Methods for Original Daily Series 110

Table 4.28 The Forecasting Performance of Machine Learning Methods for Original Daily Series 111

Table 4.29 The Forecasting Performance of Hybrid Methods for Original Daily Series 112

Table 4.30 The Forecasting Performance of Statistical Methods for Transformed Daily Series 115

Table 4.31 The Forecasting Performance of Machine Learning Methods for Transformed Daily Series 116

Table 4.32 The Forecasting Performance of Hybrid Methods for Transformed Daily Series 117

Table 4.33 The Forecasting Performance of Statistical Methods for Original Hourly Series 121

Table 4.34 The Forecasting Performance of Machine Learning Methods for Original Hourly Series 123

Table 4.35 The Forecasting Performance of Hybrid Methods for Original Hourly Series 124

Table 4.36 The Forecasting Performance of Statistical Methods for Transformed Hourly Series 127

Table 4.37 The Forecasting Performance of Machine Learning Methods for Transformed Hourly Series 128

Table 4.38 The Forecasting Performance of Hybrid Methods for Transformed Hourly Series 129

Table 5.1 The Average of Performance Measures of Both Statistical and Machine Learning Methods for Original and Transformed Series 137

Table 5.2 The Average of Performance Measures of Hybrid Methods for Original and Transformed Series 138

LIST OF FIGURES

FIGURES

Figure 3.1 The General Structure of ANN Models 38

Figure 3.2 The Working Structure of ANN Model 38

Figure 3.3 The information flows of Feed-forward Neural Network and Recurrent Neural Network 40

Figure 3.4 The Network Architecture of Single Layer Neural Network . . . 41

Figure 3.5 The Network Architecture of Multi-Layer Neural Network . . . 42

Figure 3.6 The Network Architecture of Recurrent Neural Network 43

Figure 3.7 The Working Architecture of LSTM 45

Figure 4.1 The Time Series Plots of Subset of Yearly Series 63

Figure 4.2 sMAPE Performance of All Methods for Original Yearly Series . 67

Figure 4.3 MASE Performance of All Methods for Original Yearly Series . 67

Figure 4.4 The Average of sMAPE and MASE of All Methods for Original Yearly Series 68

Figure 4.5 sMAPE Performance of All Models for Transformed Yearly Series 72

Figure 4.6 MASE Performance of All Models for Transformed Yearly Series 72

Figure 4.7 The Average Performances of All Models for Transformed Yearly Series 73

Figure 4.8 The Time Series Plots of Subset of Quarterly Series 74

Figure 4.9 sMAPE Performance of All Models for Original Quarterly Series 78

Figure 4.10 MASE Performance of All Models for Original Quarterly Series 79

Figure 4.11 The Average Performances of All Models for Original Quarterly Series 79

Figure 4.12 sMAPE Performances of All Models for Transformed Quarterly Series 83

Figure 4.13 MASE Performances of All Models for Transformed Quarterly Series 84

Figure 4.14 The Average Performances of All Models for Transformed Quarterly Series 84

Figure 4.15 The Time Series Plots of Subset of Monthly Series 85

Figure 4.16 sMAPE Performances of All Models for Original Monthly Series 90

Figure 4.17 MASE Performances of All Models for Original Monthly Series 90

Figure 4.18 The Average Performances of All Models for Original Monthly Series 91

Figure 4.19 sMAPE Performances of All Models for Transformed Monthly Series 95

Figure 4.20 MASE Performances of All Models for Transformed Monthly Series 96

Figure 4.21 The Average Performances of All Models for Transformed Monthly Series 96

Figure 4.22 The Time Series Plots of Subset of Weekly Series 97

Figure 4.23 sMAPE Performances of All Models for Original Weekly Series 102

Figure 4.24 MASE Performances of All Models for Original Weekly Series . 102

Figure 4.25 The Average Performances of All Models for Original Weekly Series 103

Figure 4.26 sMAPE Performances of All Models for Transformed Weekly Series 107

Figure 4.27 MASE Performances of All Models for Transformed Weekly Series 108

Figure 4.28 The Average Performances of All Models for Transformed Weekly Series 108

Figure 4.29 The Time Series Plots of Subset of Daily Series 109

Figure 4.30 sMAPE Performances of All Models for Original Daily Series . 113

Figure 4.31 MASE Performances of All Models for Original Daily Series . . 114

Figure 4.32 The Average Performances of All Models for Original Daily Series 114

Figure 4.33 sMAPE Performances of All Models for Transformed Daily Series 119

Figure 4.34 MASE Performances of All Models for Transformed Daily Series 119

Figure 4.35 The Average Performances of All Models for Transformed Daily Series 119

Figure 4.36 The Time Series Plots of Subset of Hourly Series 120

Figure 4.37 sMAPE Performances of All Models for Original Hourly Series 125

Figure 4.38 MASE Performances of All Models for Original Hourly Series . 126

Figure 4.39 The Average Performances of All Models for Original Hourly Series 126

Figure 4.40 sMAPE Performances of All Models for Transformed Hourly Series 130

Figure 4.41 MASE Performances of All Models for Transformed Hourly Series 131

Figure 4.42 The Average Performances of All Models for Transformed Hourly Series 131

LIST OF ABBREVIATIONS

ABBREVIATIONS	
AAE	Average Absolute Error
AIC	Akaike Information Criteria
AICc	Small-Sample Corrected Akaike Information Criterion
ANN	Artificial Neural Network
AR	Autoregressive-Average Ranking
ARCH	Autoregressive Conditional Heteroscedastic
ARMA	Autoregressive Moving Average
ARIMA	Autoregressive Integrated Moving Average
BATS	Box-Cox Transformation, ARMA residuals, Trend, Seasonality
BIC	Bayesian Information Criteria
BNN	Bayesian Neural Network
BP	Back Propagation
BVAR	Bayesian Vector Autoregressive Model
CART	Classification And Regression tree
CD	Correct Down
CP	Correct Up
CPI	Consumer Price Index
COVID-19	Coronavirus Disease
DS	Directional Symmetry
DJ	Dow Jones
EP	Electricity Production
ETS	Exponential Smoothing Method

GBM	Gradient Boosting Method
GSPC	SP 500 Commodity Price Index
HIS	Hang Seng Index
HMC	Hamiltonian Monte Carlo
HW	Holt Winters Exponential Smoothing
IP	Industrial Production
IXIC	NASDAQ Composite Index
LGT	Local and Global Trend
LSTM	Long Short Term Memory
M-Competition	Makridakis Competition
MA	Moving Average
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
Md	Medians of Absolute Percentage Error
ML	Machine Learning
MLP	Multi-Layer Perception
MSE	Mean Square Error
MPSE	Mean Percent Square Error
MRAE	Median Relative Absolute Error
MSAPE	Median Symetric Absolute Percentage Error
N225	Nikkei 225 Index
NBS	National Bureau of Statistics
NMSE	Normalized Mean Square Error
NN	Neural Network
NNETAR	Feed-Forward Neural Network
NUTS	No-U-Turn Sample
NTUA	National Technical University of Athens

OWA	Overall Weighted Average
PB	Percentage Better
PPI	Producer Price Index
RMSE	Root Mean Square Errors
R	Correlation Coefficient
RF	Random Forests
RNN	Recurrent Neural Network
RR	Ridge Regression
SARIMA	Seasonal Autoregressive Integrated Moving Average
SGT	Seasonal and Global Trend
S2GT	Double Seasonal and Global Trend
SMSE	Standardized Mean Squared Error
sMAPE	Symmetric Mean Absolute Performance Error
sNAIVE	Seasonal Naive
SS	State Space
SSE	Sum of Square Error
STL	Seasonal Trend Decomposition Procedure Based On Loess
SVC	Support Vector Classification
SVM	Support Vector Machine
SVR	Support Vector Regression
TAR	Treshold Autoregressive
TBATS	Trigonometric Seasonal, Box-Cox Transformation, ARMA residuals, Trend, Seasonality
XGBoost	Extreme Gradient Boosting

CHAPTER 1

INTRODUCTION

The prediction of the future has been very popular and attractive subject since the major civilizations in the ancient world. In those times, people used divinations generated by the diviners to have an idea about the future. For example, the weather and drought were predicted by diviners in both Eastern Han and Qing in early China [3]. In addition to this, there are some studies showing that there was a competition between diviners and physicians to produce the most accurate prediction about the future in ancient Greek [3]. However, the ways of predicting future has been changed and become more scientific in the time, especially after introduction of the notion of a time series by Yule. In 1927, Yule stated the notion that a time series is a realization of stochastic process [4]. Since then, many methods that use the past observations to identify the underlying relationship have been developed, and thereby time series forecasting and also forecasting accuracy naturally has gained popularity in the area of forecasting.

The first nontrivial study about time series forecasting accuracy was conducted by David Reid during his PhD at the University of Nottingham [5]. After this study, Box and Jenkins introduced autoregressive integrated moving average (ARIMA) model in 1970, and accelerated the time series analysis and forecasting [6].

In 1975, the study of Newbold and Granger inspired by the study of Reid opened a new door into the time series forecasting by starting the discussion about finding the most accurate model for time series from different frequencies [5]. This study inspired and encouraged many researchers to carry on studies which involve finding the most accurate forecasting model for time series under different scenarios by using several accuracy criteria to make a comparison between the models in the study.

In the lights of this kind of studies, it was explored that the most of the statistical time series models such as ARIMA suffer from producing inaccurate forecast values for the series including nonlinear patterns since they were constructed based on the linear assumptions. In this regard, several time series models for modelling nonlinear patterns in the data were developed such as treshold autoregressive (TAR) model or autoregressive conditional heteroscedastic (ARCH) model. However, it was seen that these models could outperform only for some special nonlinear problems. They could not give a sufficient forecasting performance for general nonlinear problems [7]. Due to this problem, machine learning models have been suggested and started to be used frequently in the field of time series forecasting since the introduction of back propagation for neural network models because they are designed with ability of modelling complex structures with the data [1].

Since then, there have been many studies that utilize machine learning models to improve the accuracy of time series forecasting and compare the performance of models with statistical models in terms of accuracy. However, it is seen that there are some cases where both statistical and machine learning models are insufficient to produce accurate future values for time series [2].

Zhang [2] says that it may be difficult to find the right forecasting technique for time series since it is hard to know whether the series under the study comes from linear or nonlinear process. Besides, he claims there are a few case where the series under the study include only linear or nonlinear pattern. Lastly, he states the notion accepted universally that there is no method which predicts the future accurately for every situation.

Because of problems listed above, he suggested a hybrid approach which combines both statistical and machine learning methods with the aim of capturing both linear and nonlinear patterns in the series. After his study, the hybrid approach has taken part in the comparison of being a model having high forecasting accuracy.

The increasing number of forecasting approaches and forecasters who aim to find the most accurate model for time series has caused the organization of forecasting competitions which enables forecasters to conduct large scale study and compare the newly proposed model against the existing model for forecasting [5].

One of the most famous competitions is Makridakis Competition, also known as the M Competition. This competition has been organized by Spyros Makridakis who is a professor at the University of Nicosia UNIC and his team since 1979. The most recent M Competition whose results were announced is M4 Competition including more series and methods compared to previous ones. Although there are 61 forecasting methods in this competition, most of the methods are developed by the participants. In the competition, both statistical and machine learning models whose success in accuracy were proven before the competition, such as Bayesian exponential smoothing with trend modification, recurrent neural network (RNN), long short term memory (LSTM), were not considered. Besides, there was only one hybrid model being winner in the competition, so it can be said that the competition also suffered from lack of number of hybrid approaches. Lastly, the methods were applied to the original time series. The forecasters did not consider the effect of transformation on the forecasting accuracy.

The main motivation behind this study is to make up the shortages in M4 Competition explained above by using the same data. With this study, we aim to demonstrate the functionality of different forecasting approaches that include statistical, machine learning and hybrid approaches on the univariate time series from different time frequencies. in terms of time series forecasting and forecasting accuracy, and contribute the main purpose of M Competitions by stating empirical evidence utilizing to improve the forecasting accuracy.

In this study, we consider eight statistical models including Naive Method, Seasonal Naive Method, ARIMA, SARIMA, Exponential Smoothing, TBATS, Bayesian Exponential Smoothing Models with Trend Modifications and STL Decomposition, seven machine learning covering Support Vector Regression, Random Forest, XgBoost, Feed Forward Neural Network, Bayesian Neural Network, Recurrent Neural Network and Long Short Term Memory Neural Network and hybrid models which changes based on individual performances of both statistical and machine learning models on the series. The data set used in the study are obtained from 1000 series which are selected out of 100000 time series in M4 Competition proportionally from each time class, and forecasts are obtained from both original and transformed series by Box-Cox transformation. The forecasting performance of the models are compared by

accuracy measures which are symmetric mean absolute performance error (sMAPE), mean absolute scaled error (MASE) and the arithmetic average of both measures. In addition to this, the models in the study are also compared in terms of demanding computational time in hours for forecasting. All of the analysis are conducted using R Studio with version 1.3.959.

The study consists of five chapters. The following chapter has two sections. The first section will give details about M Competitions, and the second section will review the similar studies. In Chapter 3, the models used in the study will be explained theoretically in three sections. In addition to this, both accuracy and performance measures will be explained in Chapter 3. Chapter 4 shows the result of empirical analysis. In this chapter, more details about the data set will be provided. Furthermore, the data preprocessing techniques will be given in this chapter. Then, the results obtained from both original and transformed series will be shown in six sections where each sections correspond to each time class. The study ends with conclusions and suggestions for the future work.

CHAPTER 2

LITERATURE REVIEW

There are many studies carried out by researchers to improve the forecasting accuracy and find the most accurate forecasting approach. In this part of the study, some of these studies will be summarized in two sections. First of all, M-Competitions being the inspiration to this study will be explained. Then, some of the studies involving the forecasting for the several areas included by M4-Competition dataset will be mentioned.

2.1 M-Competitions

M-Competitions being short for Makridakis Competitions are the competitions which has been held by Sypros Makridakis and his team with the aim of finding the way to improve the forecasting accuracy by making a comparison between the forecasting methods for years. There has been organized five M-competitions so far, and four of them which are resulted will be explained chronologically.

2.1.1 M1-Competition

The first M-Competition was organized in 1982 as an extension of the study of Makridakis and Hibbon in 1979 [8], which can be considered as ancestor of M-Competitions [9]. The competitions had 1001 time series consisting of yearly, quarterly and monthly series from the areas of macro, micro, industry and demographic. The forecasters produced 6 steps, 8 steps and 18 steps ahead forecasts for yearly, quarterly and monthly series, respectively via several methods including not only in-

dividual of statistical models such as naive model, versions of exponential smoothing model, Bayesian forecasting model, Box-Jenkins Methodology, linear regression but also both combination of the models and sophisticated models developed by forecasters. The forecast results were compared with respect to five accuracy measures which are Mean Absolute Percentage Error (MAPE), Mean Square Error (MSE), Average Ranking (AR), Medians of Absolute Percentage Error (Md) and Percentage Better (PB). At the end of this competition, the four main results were concluded. First of all, statistically sophisticated models did not provide significantly better results compared to simpler models. Also, it was observed that the ranking of the models varied based on the accuracy measure considered. Then, it was seen that the combination of the individual models were more accurate forecasting models when compared to individual ones. Lastly, it was stated that the accuracy of the models changed with respect to forecast horizons.

2.1.2 M2-Competition

The second M-Competition was organized in 1993 in cooperation with four companies [10]. The main reason of organizing this competition was to stop the main critique of M-Competition which was not allowing forecasters to include the additional information in the forecasting process that is done in real life forecasting. In this competition, 29 monthly series including 23 series from the companies and 6 series from the macroeconomics were used by 5 forecasters. They used both simple and well-known time series models including naive model, exponential smoothing, ARMA and ARIMA models and models developed by themselves for 15 steps ahead forecasts of the 29 series. The forecast results coming from the models were compared in terms of MAPE. At the end, the competition showed that Dampen and Single exponential smoothing models were surprisingly the most accurate models compared to other models including more complex models. Another result of the competition is that the combination of the exponential smoothing models showed a considerable result and could be used when the forecaster had no idea for the appropriate smoothing technique for the series. Also, it was stated that sophisticated or forecaster's own method seldomly produced better forecasts in comparison with simple models or combination of the models. In addition to this results, the results obtained in the

previous competition was also valid for M2-Competition.

2.1.3 M3 Competition

The third Makridakis Competition was held in 2000 with the aim of replicating and extending previous M-Competitions [11]. The extension was done by increasing number of series and including more methods and forecasters. The replication was done by testing whether the four main conclusions obtained in the previous two competition were still valid or not. The competition dataset included 3003 series consisting of yearly, quarterly and monthly series mostly. In this study, not only statistical models involving naive model, exponential smoothing model, theta model but also Machine Learning model which is Artificial Neural Network for the first time. Naive 2 model was determined as benchmark, and the forecast results were compared with respect to five measures which are Symmetric Mean Absolute Percentage Error (sMAPE), Average Ranking (AR), Percentage Better (PB), Median Symetric Absolute Percentage Error (MSAPE) , Median Relative Absolute Error (MRAE). Therefore, it was seen that Theta method which does not depend on strong statistical theory and has easy implementation majorly outperformed the other models. In addition to this final conclusion, the four main conclusions drawn from the previous competitions were proven again.

2.1.4 M4 Competition

The M4 Competition, announced in November 2017, was started in January 2018 and ended in May 2018 [12]. The main reason of organizing this competition was same as the previous M-Competitions. In this competition, the extension of M-Competitions was completed in three ways. Firstly, the number of series were increased, then ML models were included. Lastly, prediction interval was estimated in addition to point forecast. The competition presented 100000 series including even daily and hourly series to forecasters and 61 models having a broad range from the simplest model to hybrid model were applied on these series. The simple arithmetic average of Simple, Holt and Damped exponential smoothing models were set as benchmark for point

forecast. On the other hand, Naive 1 model was considered as benchmark for the prediction interval. The performance of the models were compared sMAPE, MASE and Overall Weighted Average (OWA). At the end of the competition, the hybrid approach developed by Slavek Smyl extremly outperformed the other models [12]. The second best model was the combination of seven statistical methods and one ML method. Both of these models also had the great performance for the prediction intervals. In addition to this, it was found that the first seventeen accurate model included the twelve models which were the combinations of mainly statistical models. Lastly, it was stated that pure ML models had very poor performance. Most of pure ML models in the competition were worser than the benchmarks.

2.1.5 M5 Competition

The fifth of Makridakis Competition was opened in March 2020 and ended in June 2020. The competition was run on Kaggle and presented a dataset having almost 100000 hierarchical daily series. The results of the competition has not been published yet.

2.2 Related Studies

In this part, the several forecasting studies based on topics covered by M4-Competition data set will be reviewed.

2.2.1 Macroeconomics and Microeconomics

Heaton et al. [13] carry on a study to make a decision whether models used to forecast Western macroeconomics are appropriate to forecast Chinese macroeconomics. The study involves 19 forecasting models including both the simplest time series models such as Naive Model and complex time series models such as Bayesian Vector Autoregressive Model (BVAR) . They apply the models on the data set taken from China's National Bureau of Statistics (NBS) including monthly revision of Consumer Price Index (CPI) month-on-month inflation rate, the Producer Price Index (PPI)

year-on-year inflation rate, the year-on-year growth rate of industrial production (IP), and the month-on-month growth rate of the production of electricity (EP), and compare the results based on Standardized Mean Squared Error (SMSE). They state AR, ARMA, VAR and BVAR models are superior to other models in the study.

Collin and Kies [14] propose Random Forest model for the forecasting of the univariate daily microeconomic data. In the study, they collect daily store deposits from 1990 stores and applies large number of Machine Learning models including Elastic Net, Partial Least Squares, Generalized Additive Model, Random Forest and Gradient Boosting with the optimal parameters for each series. The model performances are compared by using Root Mean Square Errors (RMSE). As a result, they conclude Random Forest is superior to other models for forecasting of microeconomic data.

Pavlov [15] suggests that Neural Networks (NN) and Support Vector Machines (SVM) are appropriate models to forecast monthly inflation which is an important macroeconomic variable. In his study, he uses monthly inflation rate in Russia and applies four models which are NN, SVM, Autoregression (AR) and Ridge Regression (RR). Among the four models, AR and RR are used as benchmarks. At the end of his study, he concludes that both ML models outperforms the benchmark models with respect to Root Mean Square Error (RMSE).

2.2.2 Industry

Centeno and Marquex [16] conduct a study to estimate the negative effect of COVID-19 on the earnings loss of the tourism industry. That's why they use the data set about monthly earnings loss of the tourism industry in Philippines taken from the Webpage of Department of Tourism. The reason of selecting Philippines is that the significant importance of tourism industry in the economy of Philippines. They apply several Seasonal Autoregressive Integrated Moving Average (SARIMA) models on the series and compare the results based on Akaike Information Criteria (AIC) and Root Mean Square Error (RMSE). Then, they decide SARIMA $(1,1,1)(1,0,1)_{12}$ is the best model to forecast the monthly earning loss in the tourism industry because of COVID-19.

Azadeh et al. [17] applies an Artificial Neural Network (ANN) model based on multilayer perception (MLP) for forecasting of annual electricity consumption belongs to high energy consumption industrial sectors including chemicals, basic metals and non-metals minerals industries in Iran from 1979 to 2003. The forecast results obtained by ANN is compared with the results produced by conventional regression models with respect to Mean Absolute Error (MAE), Mean Square Error (MSE) and Mean Absolute Percentage Error (MAPE).Therefore, they state that ANN model based on MLP is superior to non-linear regression models for annual forecasting of electricity consumption.

Chen and Wang [18] propose a hybrid SARIMA and SVM approach for the forecasting of machinery industry. They consider the monthly series including the production values for Taiwanese machinery industry in the study. First, they consider the individual performances of SVM and SARIMA models on the data set. Then, they include the hybrid approach of both models where SARIMA is used for forecasting of the linear part of the series, and SVM is used for forecasting of the non-linear part of the series. The models are evaluated in terms of Normalized Mean Square Error (NMSE), Mean Absolute Percetage Error (MAPE) and Correlation Coefficient (R). The experimental results of the study reveals that the hybrid SVM and SARIMA model clearly is more accurate model for forecasting of monthly machine industry in Taiwan.

2.2.3 Finance

Namin and Namin [19] compare the forecasting performance of Long Short Term Memory (LSTM) and Autoregressive Integrated Moving Average (ARIMA) models on the monthly data set taken from Yahoo finance Website. The data set includes Nikkei 225 index (N225), NASDAQ composite index (IXIC), Hang Seng Index (HIS), SP 500 commodity price index (GSPC), and Dow Jones industrial average index (DJ) values. The subsets are divided into two sets where 70% of them are used as training data and 30% of them are used as test data. The future values predicted by both LSTM and ARIMA are compared with respect to Rooted Mean Square Error (RMSE). Therefore, they state that LSTM clearly outperforms ARIMA in the forecasting of monthly financial time series data sets.

Merh et al. [20] focus on developing and comparing two hybrid approaches including ARIMA and ANN for the forecasting of Indian Stock Market dataset which is an important topic in the field of finance. First of all, they develop an approach called ARIMA-ANN by predicting future values using ARIMA, then modeling the residuals coming from the previous model using ANN. Then, they sum the results obtained from both models. After this, they change the usage order of the models. The future values are predicted by ANN, and residuals are modelled by ARIMA. At the end, this approach is called ANN-ARIMA. The performance of both models are compared using Average Absolute Error (AAE), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), Mean Percent Square Error (MPSE). They show that the hybrid approach of ANN-ARIMA mainly performs better than the approach of ARIMA-ANN.

Cao and Tay [21] carry on a study involving financial forecasting using several Machine Learning models. In this study, they use SP 500 daily price index data set and predict the future observations by Support Vector Machine (SVM) and Multilayer Perception (MLP) with Back Propagation (BP) . The accuracy of the models are compared based on Normalised Mean Square Error (NMSE), Mean Absolute Error (MAE), Directional Symmetry (DS), Correct Up (CP) trend and Correct Down (CD) trend. At the end of the study, they conclude that SVM performs better than MLP, although it requires less number of parameters and demands shorter time to train the model.

2.2.4 Demography

Sulaiman and Shukur [22] show that Recurrent Neural Network (RNN) and NN approaches can be an efficient way for forecasting the size and growth of the population which is the demographic variable. They decide to forecast the annual size and growth of the world population using Poisson Regression, Logistic Regression and RNN in this study. Then, they compare the results of the models based on MAPE and conclude that RNN extremely outperform the other models.

Bravo and Coelho [23] carry on a study to determine the appropriate model for forecasting monthly birth and death rate. They consider SARIMA, Holt Winter's

Seasonal Model and State Space (SS) Model since the demographic series generally include strong seasonality. The data under the study includes monthly birth and death rate at both local and regional and provided by Statistics Portugal. The performance of the models are evaluated and decided with the usage of MAPE. Therefore, it is showed that SS outperforms other models in the forecasting of birth rate and SARIMA is superior to other models for death rate forecasting.

CHAPTER 3

METHODOLOGY

Forecasting is a statistical concept used in many areas to have an idea about the future. Time series forecasting is a necessary part of the forecasting concept where the past observations is considered to produce the future observations. Various statistical methods have been developed for this purpose and contributed to the forecasting literature. Also, machine learning methods and their combinations with statistical methods have been suggested for forecasting since the first implementation of Neural Network models [1].

In this section, the forecasting methods used in this study are introduced. The statistical methods are Naive Method, Seasonal Naive Method, ARIMA, SARIMA, Exponential Smoothing, TBATS, Bayesian Exponential Smoothing Models with Trend Modifications and STL Decomposition, respectively. The machine learning methods are Support Vector Regression, Random Forest, XGBoost, Feed Forward Neural Network, Bayesian Neural Network, Recurrent Neural Network and Long Short Term Memory Neural Network. Lastly, the combination of statistical methods with machine learning methods are used for forecasting.

3.1 Simple Forecasting Methods

3.1.1 Naive Method

This method is one of the simple forecasting method in the literature. In this literature, the last observation of the data is considered as the forecast values. The forecast

equation of the model is given below.

$$\hat{y}_{t+h|t} = y_t, \tag{3.1}$$

where y_t is the value of series at time t and h is the forecast horizon. This method is also called random walk forecast and implemented by `naive()` function in R.

3.1.2 Seasonal Naive Method

This technique is similar to naive method. In this method, the forecast values are the last value from the same season. That is,

$$\hat{y}_{t+h|t} = y_{t+h-m(k+1)}, \tag{3.2}$$

where m denotes the seasonal period and k is the integer part of $(h-1)/m$ [24]. This model is implemented by using `snaive()` function in R.

3.2 Autoregressive Integrated Moving Average Model (ARIMA)

Making a statistical inference about time series data is not easy since each observation in the series is considered as a random variable. In such case, an assumption is needed to simplify the analysis, and this assumption is stationarity. If the time series has constant mean, variance and autocorrelation over time, it is called stationary time series.

The stationary time series y_t can be expressed as a weighted linear combination of current and past white noise terms e_t with zero mean and constant variance as follows

$$y_t = e_t + \psi_1 e_{t-1} + \psi_2 e_{t-2} + \cdots \tag{3.3}$$

If there are finite number of weights having nonzero values, then the series can be

reexpressed by changing notation as follows

$$y_t = e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} - \cdots - \theta_q e_{t-q} \tag{3.4}$$

This expression is known as moving average representation with order q denoted by $MA(q)$ and predicts the future values by using the linear combination of white noise terms. This model was firstly used by Slutsky (1927) and Wold (1938) [25].

The MA models suffer from being nonuniqueness which results in irrelevant questions in the following steps [25]. To overcome this problem, the invertibility, another important statistical concept for forecasting, was introduced by Granger and Andersen in 1978 [26]. This concept provides the uniqueness of the model, and makes the model ready to use for forecasting. If the series y_t has invertibility property, it can be respresented as follows

$$y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p y_{t-p} + e_t \tag{3.5}$$

where y_t for $i = 1, \ldots, p$ are the observations in the series until time p. This representation is known as the autoregressive representation of order p denoted by $AR(p)$ and uses the past observations to predict the future values. The first study for this model was conducted by Yule in 1928[25].

The series y_t that can be expressed as a combination of moving average (MA) and autoregressive (AR) representations is known as the Autoregressive Moving Average Model with orders p and q denoted by $ARMA(p, q)$ proposed by Box and Jenkins on the strength of studies of Yule and Wold explained above [6]. The mathematical expression of $ARMA(p, q)$ model is given below

$$\dot{y}_t = \phi_1 \dot{y}_{t-1} + \phi_2 \dot{y}_{t-2} + \cdots + \phi_p \dot{y}_{t-p} + e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} \tag{3.6}$$

where $\dot{y}_t = (y_t - \mu)$ is the series, e_t is independent and identically distributed white noise term with zero mean and constant variance σ^2. ϕ and θ are the autoregressive and moving average parameters, respectively.

Although this model is used many times for forecasting, the applicability of the model is limited to only stationary data set. However, most of the time series in practice are non stationary[27]. In this situation, Autoregressive Integrated Moving Average Model with orders p, d, and q $ARIMA(p, d, q)$ is proposed by Box and Jenkins[6]. In this model, I stands for integration which is the data preprocessing process used for making series stationary. The equation of this model is given below.

$$\phi(B)(1 - B)^d \dot{Y}_t = \theta(B)\epsilon_t \tag{3.7}$$

where B is backshift operator which equals to $B^d \dot{Y}_t = \dot{Y}_{t-d}$, d is dth difference operator showing the number of difference taken to make the data stationary, ϵ_t is the independent and identically distributed white noise term with zero mean and constant variance σ^2, $\phi(B)$ and $\theta(B)$ are the autoregressive and moving average polynomial terms, respectively.

This approach has three steps which are model identification, parameter estimation and diagnostic checking [2].

In model identification step, the data preprocessing operators including differencing and transformations is applied to make the series stationary and stailized in variance. Then, the autocorrelation and the partial autocorrelation structure of the series is used to identify the appropriate ARIMA model.

After this step, nonlinear optimization techniques is used to estimate the model parameters by minimizing the error function.

In the last step, diagnostic checking, the assumptions about the model errors are checked by using both visual tools and statistical tests. If any violations are detected, the model is concluded as not adequate for the data. In this case, the process returns to the first step and continues until obtaining the model that satisfied the diagnostics.

In order to construct ARIMA model in R, the steps above are conducted manually or `auto.arima` function that selects the best model via AIC, AICc, or BIC is used [28].

3.3 Seasonal Autoregressive Integrated Moving Average Model (SARIMA)

Many time series show some form of cycling or seasonality which is the repeated behaviors occurring over time period [28]. If the data has such a pattern, ARIMA model tends to be failed. For this case, Seasonal Autoregressive Integrated Moving Average Model SARIMA developed from ARIMA was proposed by Box and Jenkins [6]. The model denoted by $SARIMA(p,d,q)(P,D,Q)_s$ is given below.

$$\phi(B)\Phi\left(B^s\right)(1-B)^d\left(1-B^s\right)^D Y_t = \theta(B)\Theta\left(B^s\right)\epsilon_t \tag{3.8}$$

where s denotes the number of seasonal period, B and B^s are backshift operators, D and d are difference operators showing the number of seasonal and nonseasonal differences taken to make the data stationary, respectively. ϵ_t is the independent and identically distributed white noise term with zero mean and constant variance σ^2, $\phi(B)$ and $\theta(B)$ are the ordinary autoregressive and moving average polynomial terms, respectively. Lastly, $\Phi\left(B^s\right)$ and $\Theta\left(B^s\right)$ are the seasonal autoregressive and moving average polynomial terms, respectively.

Since $SARIMA(p,d,q)(P,D,Q)_s$ model is developed from ARIMA, it consists of three steps explained above. In order to fit SARIMA model in R, the steps are conducted manually or `auto.arima` which is the automated algorithm based on minimizing the selected information criteria is used.

3.4 Exponential Smoothing Method (ETS)

Exponential Smoothing is a forecasting technique introduced by Brown [29], Holt [30] and Winters [31] in the late 1950s. The forecasts by this method are obtained from the weighted average of the past observations where the weights downward exponentially over the observations getting older. While ARIMA models try to identify the autocorrelation in the data, this method relies on the trend and seasonality in the data [24]. This method is widely used in the time series forecasting due to its simplicity, computational efficiency and accurate forecast performance [32].

Types of the exponential smoothing method from the simplest to complicated are detailed below, respectively.

3.4.1 Simple Exponential Smoothing

The method was proposed by Brown [29]. It is generally preferred for the short term forecasting of the data with no trend and seasonality [33].

The forecast equation of the model is given below.

$$\hat{y}_{t+1} = \alpha y_t + (1 - \alpha)\hat{y}_t \tag{3.9}$$

where $\hat{y}_{t+1}$ is the forecast at time t+1, also known as smoothed value at time t, y_t is the current value of the series and $\hat{y}_t$ the smoothed value at the previous time. α is the smoothing parameter which takes on value between 0 and 1.

The equation 3.9 clearly shows that the forecast provided by SES is the weighted average of the current value and the previous smoothed value.

3.4.2 Holt's Exponential Smoothing

The method was designed for the forecasting of the data having trend component by Holt [30]. The method is based on two equations, level and trend.

$$\hat{y}_{t+h} = \ell_t + hb_t \tag{3.10}$$

$$\ell_t = \alpha y_t + (1 - \alpha)\left(\ell_{t-1} + b_{t-1}\right) \tag{3.11}$$

$$b_t = \beta^*\left(\ell_t - \ell_{t-1}\right) + (1 - \beta^*)\, b_{t-1} \tag{3.12}$$

where h is the forecast horizon, $\hat{y}_{t+h}$ is the h-step forecast values, ℓ_t is the estimated level term at time t, b_t is the estimated trend term at time t. α and β^* are the smoothing parameters whose values are between 0 and 1.

The equation of l_t shows that the level term is considered as weighted average of the value at time t and and the one-step-ahead training forecast for time t. b_t is the weighted average of difference between level term at time t and $t-1$ and the estimated trend term at the previous time [24]. Thus, the forecasts given by this method is the sum of level term and trend term multiplied by forecast horizon.

3.4.3 Holt Winters Exponential Smoothing

This exponential smoothing method was introduced by Holt[30] and Winters[31] as an extension of Holt's Exponential Smoothing method for the use of forecasting of the data having both trend and seasonality. Holt Winters method depends on three equations which are level, trend and seasonality.

The model has two different versions due to the two types of seasonality pattern which are additive and multiplicative. The additive HW model is used if the series has additive seasonal pattern, and the multiplicative HW model is used if the series exhibits multiplicative seasonal pattern.

3.4.3.1 Holt Winters Additive Method

As highlighted above, the method is used when the data under the study displays additive seasonal behaviour. The equations related to the model are given below.

$$\hat{y}_{t+h} = l_t + b_t h + s_{t-m+(k+1)} \tag{3.13}$$

where
Level: $l_t = \alpha\left(y_t - s_{t-m}\right) + (1-\alpha)\left(l_{t-1} + b_{t-1}\right)$
Trend: $b_t = \beta^*\left(l_t - l_{t-1}\right) + (1-\beta^*)b_{t-1}$
Seasonal: $s_t = \gamma\left(y_t - l_{t-1} - b_{t-1}\right) + (1-\gamma)s_{t-m}$

m is the seasonal period, h is the forecast horizon, $\hat{y}_{t+h}$ is the h-step forecast values, ℓ_t is the estimated level term at time t, b_t is the estimated trend term at time t, s_t is the seasonal component, k is an integer confirming the seasonal component m come

from the last year of the sample [24]. α, β^* and γ are the smoothing parameters whose values are between 0 and 1.

3.4.3.2 Holt Winters Multiplicative Method

As highlighted above, the method is used when the data under the study displays multiplicative seasonal behaviour. The equations related to the model are given below.

$$\hat{y}_{t+h} = (\ell_t + hb_t)s_{t+h-m(k+1)} \tag{3.14}$$

where
Level: $\ell_t = \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1})$
Trend: $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$
Seasonal: $s_t = \gamma\frac{y_t}{(\ell_{t-1}+b_{t-1})} + (1-\gamma)s_{t-m}$

m is the seasonal period, h is the forecast horizon, $\hat{y}_{t+h}$ is the h-step forecast values, ℓ_t is the estimated level term at time t, b_t is the estimated trend term at time t, s_t is the seasonal component, k is an integer confirming the seasonal component m come from the last year of the sample [24]. α, β^* and γ are the smoothing parameters whose values are between 0 and 1. It is seen that the seasonal component in this method is different than the additive one. The difference is that the seasonal components are multiplied or divided rather than addition or subtraction [34].

In 2003, Taylor [35] adapted this method for the series having multiple seasonality such as hourly data and introduced the following Double Seasonal Holt Winters Exponential Smoothing methods.

$$y_t = l_{t-1} + b_{t-1} + s_t^{(1)} + s_t^{(2)} + d_t \tag{3.15}$$

where
Level: $l_t = l_{t-1} + b_{t-1} + \alpha d_t$
Trend: $b_t = b_{t-1} + \beta^* d_t$
First Seasonal Component:$s_t^{(1)} = s_{t-m_1}^{(1)} + \gamma_1 d_t$

Second Seasonal Component:$s_t^{(2)} = s_{t-m_2}^{(2)} + \gamma_2 d_t$

l_t and b_t display the level and trend components of the series at time t, $s_t^{(1)}$ and $s_t^{(2)}$ are the equations corresponding to the first and second seasonal components of the series, respectively. m_1 and m_2 are the seasonal periods, d_t is the error term known as white noise having normal distribution, and α, β^*, γ_1 and γ_2 are the smoothing parameters which are between 0 and 1.

`ets` function in `forecast` package developed by Hyndman et al. [36] is used to implement the exponential smoothing methods. The function has three alternatives, which are N stands for none, A stands for additive and M stands for multiplicative, for error, trend and seasonal components. The parameter selection and model identification of the final model are made by the automated algorithm in the function considering information criteria such as AIC, AICc, BIC [28].

3.5 Bayesian Exponential Smoothing Models with Trend Modifications

This method is proposed by Syml et al. [37] as a Bayesian extension of Exponential Smoothing methods for the time series forecasting. In this extended version, the additional concepts are considering nonlinear trend including damped, linear and exponential patterns, assumption of the series having Student-t distribution, constructing a model for the error components which makes the model heteroscedastic and having an opportunity to use additional regressors being related to series under the study in the model. As standing the name, the models in the method are Bayesian models, and No-U-Turn sample (NUTS) being a version of Hamiltonian Monte Carlo (HMC) algorithm is used to fit them [38]. Because of the used algorithm, the expected computational time of the method for a time series varies between two and twenty minutes. If the process takes longer time, it can be considered as lack of fit of the model.

The method performs well especially for the forecasting of the short series that has to have positive values for implementation.

The performance of the model was tested on M3-Competition data set including 3003 time series with different frequencies and stated that the model has the most accurate forecast values compared to other techniques [37].

The model under this approach changes depending on the type of the series used.

3.5.1 LGT

This method can be defined as Bayesian extension of Holt's Linear Method explained in the previous part. LGT is an acronym for Local and Global Trend model and used for the forecasting of the series with no seasonality. At the beginning of the forecasting procedure, the time series observations are assumed Student-t distributions with the parameters ν, $\hat{y}_{t+1}$ and σ_{t+1}

$$y_{t+1} \sim Student\left(\nu, \hat{y}_{t+1}, \sigma_{t+1}\right) \tag{3.16}$$

where ν is the degrees of freedom, $\hat{y}_{t+1}$ is the forecast value at time $t+1$ given that all information in the data until time t, and σ_{t+1} is the error term at time $t+1$.

After this assumption, the forecasting process of LGT model works by using the following equations.

$$\hat{y}_{t+1} = l_t + \gamma l_t^{\rho} + \lambda b_t \tag{3.17}$$

where

$l_{t+1} = \alpha y_{t+1} + (1-\alpha) l_t$
$b_{t+1} = \beta^* \left(l_{t+1} - l_t\right) + (1-\beta^*) b_t$

$\hat{\sigma}_{t+1} = \sigma \hat{y}_{t+1}^{\tau} + \xi$
l_t and b_t are the level and local trend of the series at time t. $\hat{\sigma}_{t+1}$ denotes the expected error value at time $t+1$. γ is the global trend coefficient. ρ and λ are the power coefficient being between -0.5 and 1 and damping coefficient being between 0 and 1 of the global trend components, respectively. α and β^* are the smoothing parameters that take on values between 0 and 1. σ is the error coefficient. τ is the power coefficient of the error components with values between 0 and 1. Lastly, ξ denotes the minimum value of error.

The equations show that the forecasts in this approach are obtained via the summation of the local level at time t denoted l_t and global nonlinear trend γl_t^{ρ} and damped local linear trend λb_t [38].

3.5.2 SGT

The SGT stands for Seasonal and Global Trend and used for the forecasting of the series having seasonality [37]. It is considered as a Bayesian extension of Holt-Winters Exponential Smoothing mentioned in the previous section. As same as LGT, the series is assumed Student-t distribution with the parameters ν, $\hat{y}_{t+1}$ and σ_{t+1} in SGT.

$$y_{t+1} \sim Student\left(\nu, \hat{y}_{t+1}, \sigma_{t+1}\right) \tag{3.18}$$

where ν is the degrees of freedom, $\hat{y}_{t+1}$ is the forecast value at time $t+1$ given that all information in the data until time t, and σ_{t+1} is the error term at time $t+1$.

Then, we have

$$\hat{y}_{t+1} = \left(l_t + \gamma l_t^{\rho}\right) s_{t+1} \tag{3.19}$$

where

$l_{t+1} = \alpha \frac{y_{t+1}}{s_{t+1}} + (1-\alpha) l_t$
$s_{t+m+1} = \zeta \frac{y_{t+1}}{l_{t+1}} + (1-\zeta) s_{t+1}$

$\widehat{\sigma}_{t+1} = \sigma \hat{y}_{t+1}^{\tau} + \xi$
l_t and s_t are the level and seasonal trend of the series at time t. $\hat{\sigma}_{t+1}$ denotes the expected error value at time $t+1$. γ is the global trend coefficient. ρ is the power coefficient being between -0.5 and 1. m is the seasonal period. α and ζ are the smoothing parameters that take on values between 0 and 1 for level and seasonal terms, respectively. σ is the error coefficient. τ is the power coefficient of the error components with values between 0 and 1. Lastly, ξ denotes the minimum value of error.

As seen, the mathematical concept in both LGT and SGT are similar to each other. However, seasonal component s_t is included and used in the forecast equation as

multiplication term in SGT. Also, the local trend term denoted by b_t is dropped in this approach.

3.5.3 S2GT

This method is designed for the series having double seasonality and can be defined as a Bayesian extension of Taylor's Double Seasonal Exponential Smoothing methods defined in the previous section [38]. Like LGT and SGT, this method starts with an assumption of observations having Student-t distribution with the parameters ν, $\hat{y}_{t+1}$ and σ_{t+1}.

$$y_{t+1} \sim Student\left(\nu, \hat{y}_{t+1}, \sigma_{t+1}\right) \tag{3.20}$$

where ν is the degrees of freedom, $\hat{y}_{t+1}$ is the forecast value at time $t+1$ given that all information in the data until time t, and σ_{t+1} is the error term at time $t+1$.

In this method compared to SGT, the second seasonal component w_t is considered and the equations are obtained.

$$\hat{y}_{t+1} = \left(l_t + \gamma l_t^{\rho}\right) s_{t+1} w_{t+1} \tag{3.21}$$

where

$l_{t+1} = \alpha \frac{y_{t+1}}{s_{t+1} w_{t+1}} + (1-\alpha) l_t$
$s_{t+m+1} = \zeta \frac{y_{t+1}}{l_{t+1} w_{t+1}} + (1-\zeta) s_{t+1}$
$w_{t+d+1} = \delta \frac{y_{t+1}}{l_{t+1} s_{t+1}} + (1-\delta)\, w_{t+1} \quad \widehat{\sigma}_{t+1} = \sigma \hat{y}_{t+1}^{\tau} + \xi$

Most of the parameters and terms have same explanations as in SGT. The additional terms are d denoting the seasonal period for the second seasonal component and δ denoting the smoothing parameter of the second seasonal component.

The algorithms for this model given above is executed on R using `rlgt` function in `Rlgt` package [37].

3.6 TBATS

The well-known time series methods including ARIMA, ETS and State Space Models usually cannot handle the data having complex seasonal pattern such as multiple seasonality, high seasonal frequency or non-integer seasonality for modelling and forecasting easily.

To avoid such a problem, Livera et al. [39] developed and introduced a model which can be defined as the generalization of exponential smoothing model for the data with high frequency in 2011 [40]. The model is called $\mathrm{BATS}\,(\omega, \phi, p, q, m_1, m_2, \ldots, m_V)$. The BATS model is an acronym for Box-Cox transformation, ARMA residuals, trend and seasonal components [39]. Moreover, ω is Box-Cox parameter, ϕ is damping parameter, p and q are ARMA orders. Lastly, $m_1, m_2, \ldots, m_V$ denote the seasonal periods.

The BATS model applies Box-Cox transformation to the series y_t to avoid nonstationarity in the variance. This implementation restricts the applicability of the model only for the series with positive observations, but most of the time series in practice contain positive observations [39].

The series with Box-Cox transformation is described by

$$y_t^{(\omega)} = \begin{cases} \frac{y_t^{\omega}-1}{\omega}, & \omega \neq 0 \\ \log y_t, & \omega = 0 \end{cases} \tag{3.22}$$

where ω is Box-Cox parameter as mentioned above. Then, the transformed series $y_t^{(\omega)}$ can be extended as follows

$$\begin{aligned} y_t^{(\omega)} &= \ell_{t-1} + \phi b_{t-1} + \sum_{i=1}^{T} s_{t-m_i}^{(i)} + d_t \\ \ell_t &= \ell_{t-1} + \phi b_{t-1} + \alpha d_t \\ b_t &= (1-\phi)b + \phi b_{t-1} + \beta d_t \\ s_t^{(i)} &= s_{t-m_i}^{(i)} + \gamma_i d_t \\ d_t &= \sum_{i=1}^{p} \varphi_i d_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t \end{aligned} \tag{3.23}$$

where l_t denotes the local level in period t, b and b_t represent long-run and short-run trend, respectively. Furthermore, d_t exhibits an ARMA(p,q) with the gaussian white noise term ϵ_i having zero mean and constant variance σ^2, and $s_t^{(i)}$ demonstrates i^{th} seasonal component at time t. Lastly, the values of α, β and γ_i for $i = 1, \ldots, T$ are the smoothing parameters in the concept of BATS.

Although the BATS model can handle multiple seasonality, it is unable to handle non-integer seasonality [39]. For this reason, Livera at al. developed the trigonometric representation of i^{th} seasonal component s_t^i based on Fourier series and introduced $\mathrm{TBATS}\left(\omega, \phi, p, q, \left\{m_1, k_1\right\}, \left\{m_2, k_2\right\}, \ldots, \left\{m_V, k_V\right\}\right)$ where T stands for trigonometric representation of the seasonal component and remaining letters have the same meaning as in the BATS model. Moreover, the parameters of the TBATS model are the same as in the BATS model. The one additional parameter is k_i representing the number of harmonics for the seasonal component $s_t^{(i)}$.

The trigonometric representation of the seasonal component of $s_t^{(i)}$ in the concept of the TBATS is expressed by the following equations.

$$\begin{aligned} s_t^{(i)} &= \textstyle\sum_{j=1}^{k_i} s_{j,t}^{(i)} \\ s_{j,t}^{(i)} &= s_{j,t-1}^{(i)} \cos \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \sin \lambda_j^{(i)} + \gamma_1^{(i)} d_t \\ s_{j,t}^{*(i)} &= -s_{j,t-1} \sin \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \cos \lambda_j^{(i)} + \gamma_2^{(i)} d_t \end{aligned} \tag{3.24}$$

where $\gamma_1^{(i)}$ and $\gamma_2^{(i)}$ are the smoothing parameters and $\lambda_j^{(i)} = 2\pi j / m_i$. Also, $s_{j,t}^{(i)}$ and $s_{j,t}^{*(i)}$ define the stochastic level of the i^{th} seasonal component, and stochastic growth of i^{th} seasonal component, respectively.

In the concept of TBATS model, the best parameter values are selected among the values obtained from the decomposition of the time series by Akaike information criteria (AIC) [40].

TBATS model deals with typical nonlinear features occuring in the real time series frequently and takes into account the autocorrelation structure of residuals [39]. The most important advantage of this technique is that it can be used for the data having high or non-integer seasonal frequency [40]. On the other hand, the most important disadvantage of this model is working slowly when modelling long time series. In

order to implement this methods in R, `tbats()` function from `forecast` package is used.

3.7 STL Decomposition

Time series data display various behaviours which are trend, seasonality and cycles. It is generally useful to decompose a time series into its components that represents those patterns individually.

For this purpose, a variety of decomposition techniques have been developed since the first decomposition technique by Person (1919) [41].

The seasonal trend decomposition procedure based on loess (STL), [42] is one of the decomposition techniques used for additive decomposition for the time series [41].

STL procedure decomposes time series of y_t into three additive components of trend T_t, seasonal S_t and remainder R_t.

Among these three components, trend component shows the variation at the low frequency of the data. Then, seasonal component represents the variation at or near the seasonal frequency of the data. Lastly, remainder component exhibits the remaining variation in the data after removing both trend and seasonal component [42].

Therefore, every time series can be written as

$$y_t = T_t + S_t + R_t \quad t = 1, \ldots, N. \tag{3.25}$$

This decomposition procedure works by applying loess smoother that builds the locally weighted polynomial regression at each data point through the data sequentially [41]. While doing this, it uses the parameters obtained from the eigenvalues and frequency response analysis of the series [43].

STL decomposition is formed by two recursive procedures, inner and outer loops. In each inner loop, seasonal component and trend component are updated by seasonal smoothing and trend smoothing, respectively. At end of each inner loop, the remain-

der component is obtained by using both seasonal and trend components estimated in the inner loop [44].

The details for inner loop are given below.

Suppose that seasonal and trend components at k^{th} iteration are denoted by $S_t^{(k)}$ and $T_t^{(k)}$, respectively. Also, the inital value for $T_t^{(k)}$ is zero.

Step 1: *Detrending*
At $(k+1)^{th}$ iteration, the detrended series is obtained by removing the trend component $T_t^{(k)}$ from the series y_t, i.e

$$y_t - T_t^{(k)} \tag{3.26}$$

Step 2: *Sub-Cycle Smoothing*
At this step, the subseries obtained at the previous step are smoothed by applying loess smoother to obtain cycle-subseries, $C_t^{(k+1)}$ which is transitional seasonal series.
Step 3: *Low Pass Filtering of Smoothed Cycle Subseries*
First, low pass filtering and then loess regression is implemented to $C_t^{(k+1)}$ to determine ant remaining trend, $L_t^{(k+1)}$.
Step 4: *Detrending of Smoothed Cycle Subseries*
The seasonal component $S_t^{(k+1)}$ is estimated by removing $L_t^{(k+1)}$ obtained at the previous step from the cycle-subseries $C_t^{(k+1)}$. That is,

$$S_t^{(k+1)} = C_t^{(k+1)} - L_t^{(k+1)} \tag{3.27}$$

Step 5: *Deseasonalizing*
The seasonally adjusted series is obtained by substracting $S_t^{(k+1)}$ computed at the fourth step from the original data, i.e

$$y_t - S_t^{(k+1)} \tag{3.28}$$

Step 6: *Trend Smoothing*
At the last step, a loess smoother is applied to the seasonally adjusted series coming from the previous step to get the trend component $T_t^{(k+1)}$.
Therefore, as given above, it is seen seasonal smoothing part of the inner loop is step

2, 3 and 4. Trend smoothing part is step 6 [42].
At the end of the each inner loops, the remainder component $R_t^{(k+1)}$ is calculated by using the following formula.

$$R_t^{(k+1)} = y_t - S_t^{(k+1)} - T_t^{(k+1)} \tag{3.29}$$

If any R_t values are considerably large, the corresponding weights are calculated. Then, those weights are included in the next inner loop to reduce the effect of outlier which is the result of the outer loop of the previous iteration [44].

STL procedure is frequently used for time series decomposition compared to other methods since it has several advantages. As highlighted above, this procedure is based on pure numerical methods and does not depend on any mathematical formula. That's why it is easy to apply this method to large number of the series [41]. Also, it has capability of handling data with outliers and any type of seasonality addition to quarterly and monthly data [24].

In order to carry out STL procedure in R, `stl()` function under `stats` package is used.

3.8 Support Vector Machine

Support Vector Machine was proposed by Boser based on his study with Vapnik and Guyon in the area of statistical learning where the aim is to minimize the expected risk with respect to cost function in 1992 [45], [46].

The main work of SVM is to divide data into classes by finding a hyper plane with the greatest margin which is the distance between hyper plane and the closest data point on the each side. As understood from this definition, Support Vector Machine was originally designed for the classification tasks. However, the algorithm of the SVM was adapted to regression problems [47]. Since then, the SVM has been called Support Vector Classification denoted SVC if the interested problem involves the classification and called Support Vector Regression denoted SVR if the interested problem involves regression [48].

Time series forecasting can be considered as autoregressive in time. If so, SVR is used for the forecasting task [49]. Thus, the concept of SVR is explained in this section.

Support Vector Regression builds the model based on the given data and forecasts the future observations using this trained model. The general form of the SVR is given below [46].

$$f(x) = (w \cdot \Phi(x)) + b \tag{3.30}$$

where x shows the input vector, w is the weight vector for the input vector, Φ is the kernel function which will be explained later and b is the bias term.

Assume that a linear problem is discussed. If so, the SVR equation becomes

$$f(x) = (w \cdot x) + b. \tag{3.31}$$

In order to find the best hyper plane under this circumstance, the following cost function denoted by Q should be minimized.

$$\begin{aligned} Q &= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} L^{\varepsilon}\left(x_i, y_i, f\right) \\ &\text{subject to } \begin{cases} y_i - wx_i - b \leq \varepsilon + \xi_i \\ wx_i + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \end{aligned} \tag{3.32}$$

This cost function can be explained in three parts. In the first part of it, we have a term that arranges the weight size. It also reduces the large weight which may give rise to large variance. In the second part, the penalty function C is introduced. This function imposes a penalty on the error terms being larger than $\pm\varepsilon$ via ε-insenstive loss functon L^{ε} for each training observations. These large error terms are called slack variables and denoted by ξ and ξ^* for the errors greater than ε and less than ε, respectively. The last part of the equation shows the constraints determined by the error term [49].

Minimizing the cost function Q given above is an optimization problem. This problem is solved by Quadratic Programming optimization depends on Lagrangian theory.

During this optimization process, the model parameters w and b are also estimated based on the given data. The weight vector w is obtained from the derivation of the following equation

$$w = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \, x_i \tag{3.33}$$

where α and α^* are the Lagrange multipliers corresponding to some training observations. Therefore, the problem is solved by using this formula in the equation (3.30). Then, we have the following equation as a solution for unknown data point.

$$f(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot \langle x_i, x \rangle + b. \tag{3.34}$$

In this solution, not all of the training points are used. Some of them have zero Lagrange multipliers. In other words, the same solutions are obtained even if these observations are dropped. The observations in the training set having nonzero Lagrange multipliers are called the support vector and decide the shape of the solution [49]. If the observations have considerably large multipliers values, the process demands more computation to find the final solution, and if they have considerably smaller values, the process demands less computation to find the final solution.

As stated above, the SVR approach for the linear problem has been described up to this point. For the problem which requires nonlinear optimization problem such as time series prediction, a function that maps the data from input space into high dimensional space, also called the feature space, making the relationship between output and input variables linear is considered. Then, the following solution function is obtained for the unknown data point in the nonlinear problem.

$$f(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot K \left(x_i, x \right) + b \tag{3.35}$$

where K denotes the kernel function Φ given in the equation (3.30). There are several kernel functions used for this mapping process such as Linear, Sigmoid, Radial Basis and Polynomial. The choice of Kernel function is one of the most important factor

affecting accuracy of SVR [50]. Rüping stated [51] that Radial Basis Function shows the best performance in the time series forecasting compared to other functions. The formulation of Radial Basis Function is given below.

$$\Phi(x) = K(x_i, x_j) = \exp\left(-\gamma \left\|x_i - x_j\right\|^2\right), \quad \gamma > 0 \tag{3.36}$$

where x_i and x_j are the data points and γ is the kernel parameter.

In this concept, the type of both penalize and kernel functions, the value of the parameter of kernel function, and ε should be set by the user, except the Lagrange multipliers before the implementation. In R programming, the process is applied by using `svm()` function in `e1071` package. The parameter tuning is also done by using `tune()` function in the same package [52].

3.9 Random Forests

Random forests, also called a random decision forest [53] is one of the most popular and used ensemble learning algorithm in the field of data mining. The method was proposed by Breiman in 1984 for the use in the classification or regression problem like SVM [54].

It has also been used in the field of time series forecasting for various purposes including weather forecasting, solar radiation forecasting, biostatistics etc. [55]. The random forest algorithm was developed on the basis of two concepts, bootstrap aggregating, also known as bagging introduced by Breiman, and randomly selection of features in the data under study.

For a given data set with n observation and m features, the random subset with n' is created by technique of sampling with replacement, known as bootstrapping, to construct an individual decision tree at first. This is so-called bootstrapped sample. The number of n' is equal to n being the size of the original data, and one-thirds of the size n' is formed by the replicated observations [56]. During the resampling, the observations that are not taken into the bootstrapped sample is called out-of-bag (OOB) observations which has importance in terminating the algorithm [55].

Then, p features out of m features are selected randomly at the each node of the tree to make it decorrelated.The value of p is generally taken as $\sqrt{m}$ for the classification problem [57] and $m/3$ for the regression problem [58]. However, the choice of p does not affect the performance of the method entirely [55].

Among the selected p features, the RF algorithm allows for using only one feature which is selected via an impurity measure at each split in the tree. During this selection, although they are known as strong predictors, they are not taken into account by $(m-p)/m$ of the splits on the average. By this way, the other predictors are given a chance to ensure the decorrelation [57].

After selection of the bootstrapped sample with size n' and p features, Classification and Regression tree (CART) without pruning is used to construct the individual tree on this sample [56]. Then, this procedure is repeated B times, and the grown tree is added to ensemble each time to make it grown. Then, the final decision is obtained by aggregating the results of B trees. To do so, mean or median of the outputs is calculated in the regression problem, or the majority rule is used for the classification problem.

The approach that RF used not only ensures the stability of the model, but also improves the accuracy. Moreover, it reduces variance and protects the model from overfitting problem [55].

The RF procedure explained above can be summarized as follows

1. For $b = 1 \cdots B$
   - Draw a bootstrap sample with size n' from the training data.
   - Build a random forest tree T_b on the bootstrapped sample by replicating the following steps until the tree T_b is grown.
     - Select p predictors among m predictors randomly.
     - Find the predictor that best split the node into two nodes.
2. To find the final decision, aggregate the outputs of the ensemble of the trees T_b where $b = 1 \cdots B$. i.e,
   For regression problem

$f(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$

For classification problem

$f(x) =$ majority vote$(T_b(x))_1^B$

where $f(x)$ is the predicted value, $T_b(x)$ is the output obtained from tree T_b, B is the number of trees. m and p are the total number of features and randomly selected features, respectively.

In R programming, the process is applied by using `randomForest()` function in `randomForest` package. The parameter tuning is also done by using `tuneRF()` function in the same package [59].

3.10 Extreme Gradient Boosting (XGBoost)

Boosting is an ensemble learning algorithm proposed by Schapire in 1990 to produce accurate predictions using decision trees [60]. This technique aims to generate a strong learner by combining a set of weak learners which are decision trees. In this technique, the trees are grown sequentially and slowly [57]. In other words, information obtained from the current tree is used to grow the next tree. Therefore, the trees are not independent unlike bagging defined in the previous method, which is Random Forests. Boosting algorithm has been used as a solution for both regression and classification tasks since its first implementation in 1990.

Several learning algorithm based on the idea of boosting have been suggested in the literature. Gradient Boosting, proposed by Friedman [61], is one of them. After this technique, the boosting algorithm has started to be considered as a general modelling algorithm regarded as an optimization algorithm using gradient descent method. As stated, the algorithm builds models in the negative sense of partial derivative of loss function using gradient descent approach [62]. This algorithm can also be seen as a numerical optimization approach whose aim is to form an additive model minimizing the loss function [63].

In GBM, the process starts with fitting a tree to the data, initially. Based on this initial tree, predictions are procured and the residuals are obtained. Afterward, a new tree

is fitted using the residuals and it is added into the previous fitted function. Then, the updated residuals are created.

This procedure is running until the convergence is satisfied and the final decision is made by averaging for the regression tasks and using majority rule for the classification tasks.

In Gradient Boosting, the main focus is to repair the previous model by assigning weights to the training observations. If the training observation has the largest error, the more weight is assigned to this observation [64]. By this way, the algorithm is both prevented from over fitting problem and produces more accurate results.

XGBoosting, which is short for Extreme Gradient Boosting, is another learning algorithm introduced by Chen and Guestrin in 2016 [65]. The method involves some modification on the Friedman's Gradient Boosting algorithm. Due to these modifications, this approach can also be interpreted as an efficient and scalable application of Friedman's method [66].

According to Chen and Guestrin [65], XGBoosting can be described as an ensemble of Classification and Regression Trees (CART). Thus, it is so-called Tree Boosting [64]. The reason is that a single tree may not be adequate to obtain good results, so it is probable to get good results by considering multiple trees. The final results are obtained by summing each tree's result. Therefore, the model can be written as

$$\hat{y}_i = \sum_{k=1}^{K} f_k\left(x_i\right), f_k \in \mathcal{F} \tag{3.37}$$

where K is the number of trees and f_k denotes the function in the functional space $\mathcal{F}$.

In order to find out the set of functions used in the model, the following objective function consisting of two components, loss and regularization, is defined as follows [67]

$$Obj = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i\right) + \sum_{k=1}^{K} \Omega\left(f_k\right), f_k \in \mathcal{F} \tag{3.38}$$

where l denotes the differentiable convex loss function measuring the difference between prediction $\hat{y}_i$ and the real value y_i, and Ω is the regularization term penalizing the complexity of the model to prevent over fitting.

This objective function cannot be optimized using traditional optimization methods because the functions in the model behaves like parameters [67]. To minimize the objective function, the addition of training functions are considered, then we have the following equation as an objective function

$$\begin{aligned} Obj &= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{k=1}^{K} \Omega\left(f_k\right) \\ &= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t\left(x_i\right)\right) + \Sigma_{k=1}^{K} \Omega\left(f_k\right). \end{aligned} \tag{3.39}$$

In order to simplify the optimization of the objective function, the two order of Taylor polynomial is used in the previously defined function. Then, the optimization function at t^{th} iteration is defined

$$Obj^{(t)} \approx \sum_{i=1}^{n} \left[l\left(y_i, \hat{y_i}^{(t-1)}\right) + g_i f_t\left(x_i\right) + \frac{1}{2} h_i f_t^2\left(x_i\right) \right] + \sum_{k=1}^{K} \Omega\left(f_k\right) \tag{3.40}$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y_i}^{(t-1)}\right)$ is the gradient and $h_i = \partial^2_{y^{(t-1)}} l\left(y_i, \hat{y_i}^{(t-1)}\right)$ is the hessian. Afterward, the objective function can be also rewritten by removing the constant terms

$$Obj^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t\left(x_i\right) + \frac{1}{2} h_i f_t^2\left(x_i\right) \right] + \Omega\left(f_t\right). \tag{3.41}$$

After this, the mapping $\mathrm{q} : R^d \rightarrow \{1, 2, \ldots, T\}$ is defined to map the input to the index of the region. In addition to this, a vector ω, which represents the score of each region, is defined. The function of ω is given below

$$\begin{aligned} f_t(x) &= \omega_{q(x)} \\ \omega \in R^T, q &: R^d \rightarrow \{1, 2, \ldots, T\} \end{aligned} \tag{3.42}$$

To calculate the regularization term in the objective function, the following equation is concerned.

$$\Omega(f_t) = \frac{1}{2} \lambda \sum_{j=1}^{T} \omega_j^2 + \gamma T \tag{3.43}$$

where λ and γ are the regularization parameters.

After applying some mathematical process on the objective function, the following gain function used to score a leaf node during splitting is formed [68].

$$Gain = \frac{1}{2} \left[\frac{\left(\sum_{i \in I_L} g_i\right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i\right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i\right)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma \tag{3.44}$$

where $I = I_L U I_R$. In this equation, the first three terms denotes the score on left, right and original leaf, respectively [68] and γ is a regularization parameter as stated previously.

Unlike Friedman's Gradient Boosting, XGBoosting builds tree in a parallel way that uses all of the CPU's of the machine and this makes the algorithm faster [64]. Thus, in addition to theoretical developments, this improvement is one of the key factor behind the good performance of XGBoosting [68].

In R programming, the process is applied by using `xgbar()` function in the package of `forecastxgb` [69].

3.11 Artificial Neural Network

The scientists have started to carry on some studies about the learning processing in the human brain that can be described as a complex, nonlinear and parallel computer since the early 1950s [43]. In the light of these studies, ANN was developed as a mathematical model simulating the human neural biology to solve the nonlinear problems. Since its first implementation, it has been used in the several areas and gaining in popularity due to its characteristic properties.

The one of its significant properties is containing non-linearity in its structure. Thereby, the ANN models are able to capture and model complex nonlinear problems. Also, the ANN models can be described as universal approximators that approximate to large class of target functions accurately [2]. They don't require any modelling assumptions. Instead, they are specified by the data under the study. On the other hand, ANN models suffer from being time consuming since they require high level of modelling complexity [43]. The general structure of ANN models is illustrated in Figure 3.1.

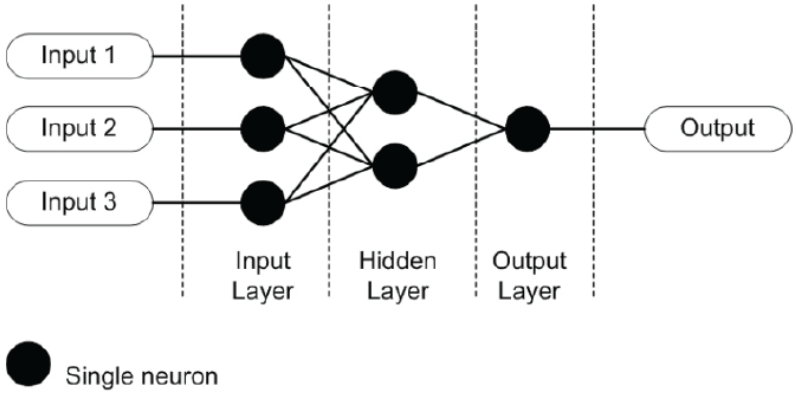

Figure 3.1: The General Structure of ANN Models [70]

The ANN consists of three layers, input layer, hidden layer and output layer. The input layer is the one where data is introduced to the network. In the hidden layer, the information in the data are proceed. Lastly, the output layer denotes the predicted value based on the given inputs [71].

As illustrated in Figure 3.1, the units called neuron in the layers are connected to each other. These connection parameters are called weights used to capture the complex structure in the data [27].

The working structure of ANN model is illustrated in Figure 3.2.

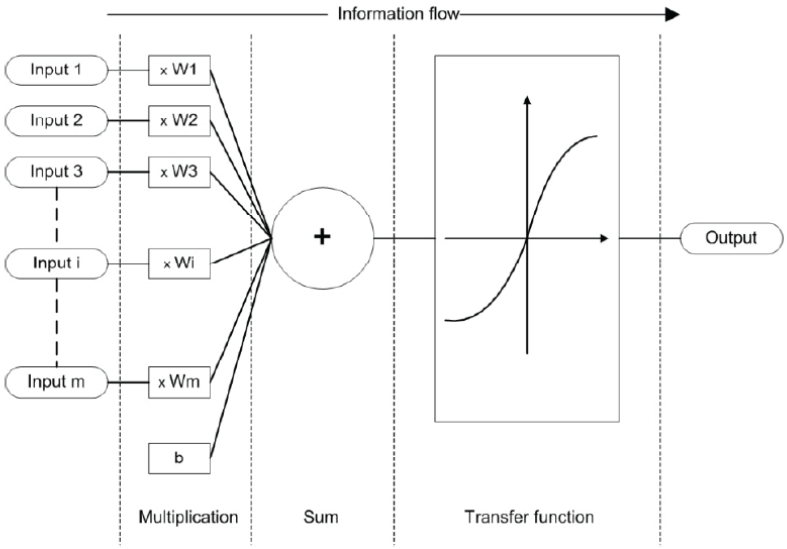

Figure 3.2: The Working Structure of ANN Model [70]

The working structure of ANN has three steps which are multiplication, summation and activation as shown in Figure 3.2. At the first step, the given inputs are multiplied by the weights. Next, these weighted inputs are summed. Then, a bias term is added to adjust the threshold of the transfer function known as an activation function. At the final step, the summation of the weighted inputs and bias are transformed into the final result by the activation function.

The mathematical formulation of this process is given below.

$$y(t) = F\left(\sum_{i=0}^{m} w_i(t) \cdot x_i(t) + b\right) \tag{3.45}$$

where $x_i(t)$ is the input in discrete time t where i goes from 0 to m, $w_i(t)$ is the weight value in discrete time t, b is bias, F is an activation function, and y_t is the output value in discrete time t.

In this formulation, the most essential component for the ANN is the activation function denoted by F. It will characterize the mathematical form of the ANN and bring nonlinearity into the network. The type of the activaiton funciton determines the success of the model, and depends on the type of the problem. It is selected from the following set of functions: set function, linear function, nonlinear function (sigmoid, hyperbolic, relu, etc.) [70].

Among these functions, the sigmoid function that produces output in the range between 0 and 1 is a commonly used activation function in practice [72].

$$sig(x) = \frac{1}{1 + exp(-x)}. \tag{3.46}$$

In addition to the activation function, training is another essential factor for the success of the model. [71]. Back-propagation (BP) algorithm is the most commonly used algorithm for the training process among several algorithms for this purpose. In this algorithm, the weights are updated by the Gradient Descent Algorithm based on the error values calculated by the difference between actual value and predicted value. This process is repeated until convergence is achieved. In other words, the algorithm runs until obtaining set of weights that minimizes the cost function such as

SSE, MSE.

The ANNs can be categorized into two major class, dynamic and static. If the output values are produced from the given input directly, the ANN is called a static network such as Feed-forward Neural Network. If the output of the network are produced from the current and previous input, output and hidden states, it is called a dynamic network such as Recurrent Neural Network [73].

The information flows of Feed-forward Neural Network and Recurrent Neural Network are shown in Figure 3.3.

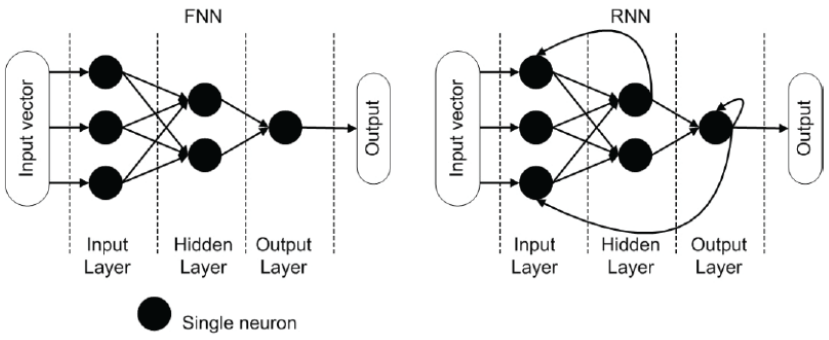

Figure 3.3: The information flows of Feed-forward Neural Network and Recurrent Neural Network [70]

In this study, both types of NNs are considered because capturing nonlinearity in time series forecasting is an essential process to improve the accuracy. In time series forecasting, the lagged values of the series is given to the network as inputs [71].

3.11.1 Feed-Forward Neural Network

As defined in the previous section, the network architecture is called the feed-forward neural network if the outputs are produced by the inputs without feedback. In other words, the inputs do not take any feedbacks coming from the outputs through the networks, and the information in the data is transmitted in only one direction.

In feed-forward neural network, the number of layers, the choice of activation function and the number of connections between neurons are not restricted [70].

The feed-forward neural networks can be categorized into two groups based on the number of layers in their structure. These groups are single layer feed forward neural networks and multi-layer feed forward neural networks.

The network architecture of single layer feed-forward neural network is illustrated in Figure 3.4.

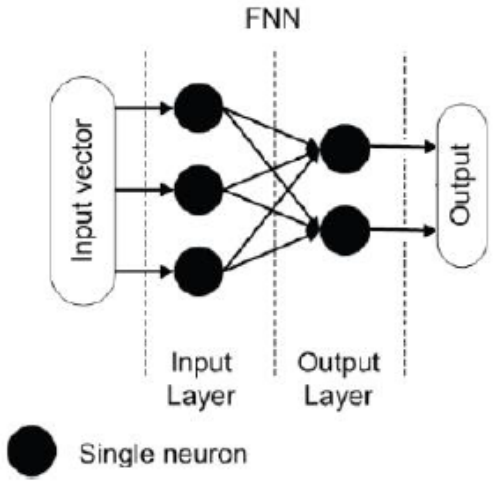

Figure 3.4: The Network Architecture of Single Layer Neural Network [70]

Although single layer feed forward neural network has two layers which are input and output layers, the input layer is not taken into account because the computation process does not occur in that layer. In this NNs, inputs are carried forward by the weights, and output is produced by the neurons in the output layer [74]. This type of NNs can solve only linear problems and it is very similar to AR models [70].

If a hidden layer is inserted between input layer and output layer, then the feed-forward neural networks is able to capture the nonlinearity in the data. In this case, the feed-forward neural networks is called the multi-layer feed forward neural networks. The architecure of multi-layer feed forward neural networks is represented in Figure 3.5.

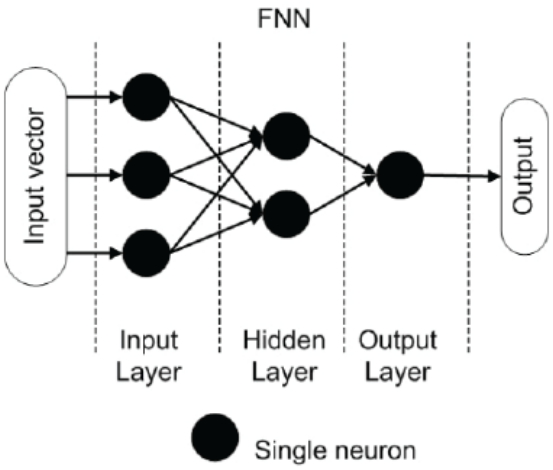

Figure 3.5: The Network Architecture of Multi-Layer Neural Network [70]

Both types of NNs are generally used back-propagation algorithm to train the network. However, the single hidden layer feed-forward neural networks is the most widely used model for time series and forecasting [2].

The structures in Figure 3.4 and Figure 3.5 are called a fully connected neural network because all of the neurons in the layer are connected to neurons in the next layer. If some of connections between neurons are not established, then it is called a partially connected neural networks.[74].

In R programming, the process is applied by using `nnetar()` function in `forecast` package [36].

3.11.2 Recurrent Neural Network

Two properties of the feed-forward neural networks make them useless in the sequence learning. The first one is that the feed-forward neural networks require an independence assumption between training and test data. If the points in the data depends on each other such as time series data, this assumption is violated naturally [75]. The second one is that the feed-forward neural networks try to make both input and outputs a fixed length vector. Therefore, it becomes an inappropriate model for the prediction in time series data [76].

RNNs were developed particularly to process the information in the sequential data or time series data because of these reasons. The network architecture of RNN is illustrated in Figure 3.6.

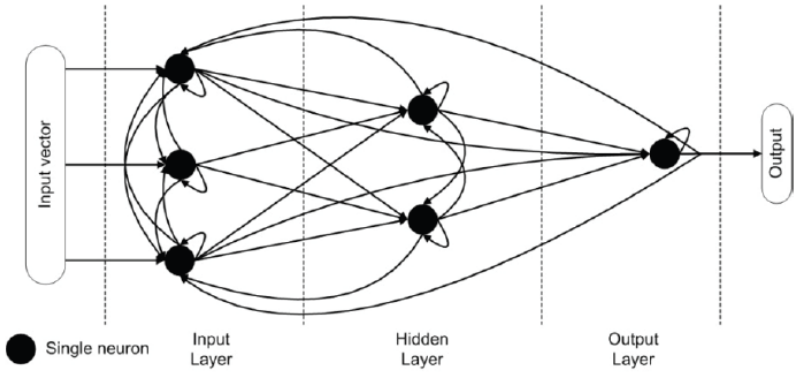

Figure 3.6: The Network Architecture of Multi-Layer Neural Network [70]

The working structure of RNN is similar to the feed-forward neural network, but there is one important difference which is the feedback loop. In feed-forward neural network, the signals are transmitted forward from input to output. However, the signals are transmitted both forward and backward in RNN structure [75]. This working process creates internal connections between the hidden units. In addition to this, the loops are introduced. These internal connections improve the capability of the usage of the previous observations while making predictions. Each layer in an RNN has a repetitive link with a delay associated with it.

Thus, the previous outputs are said to be stored in some form of "memory" and the portion of the RNN that maintains a state at a given temporal interval is said to be the "memory cell". These memory cells can also be modified to form other, more complex cell types [76]. As opposed to its advantages in the sequential data, RNNs suffer from vanishing of gradient when long term dependence exists.[75].

The method is implemented in R by using `train()` function with an argument `rnn` from `rnn` package. [77]

3.11.3 Long Short Term Memory

The standard recurrent neural networks suffer from the problem of failing to converge to the optimum minima called vanishing of gradients when the problem in the study involves the long term dependencies, as stated in the previous section. Long short term memory was developed as a special type of RNN by Hochreiter and Schmidhuber in 1997 with the motivation of solving this problem [78]. Since its first application, the method has been improved by many people [79].

The standard RNNs have a structure consisting of series of repeated simple hidden layers. As opposed to it, the hidden layers in LSTM are more complex. The recurrent hidden layer in LSTM includes units called memory blocks that are able to remember the value recorded in any time point in the data [80]. Also, each memory block contains units called a gate which defines the working structure of LSTM. The gates in the memory block are an input gate, an output gate and a forget gate. The memory blocks also have a self-connected memory cells. The input gate controls activation item entering the memory cell. The output gate knows which cell to activate for filtering and passing to the next network. The forget gate helps the network to forget past inputs and reset the storage unit. In addition, multiplier gates were carefully used to allow memory cells to access and store information for long periods of time [75].

By this working architecture, LSTM can reduce the effect of problem of vanishing gradients and becomes an appropriate method for the problems involving long term dependencies. The working architecture of LSTM is illustrated in Figure 3.7.

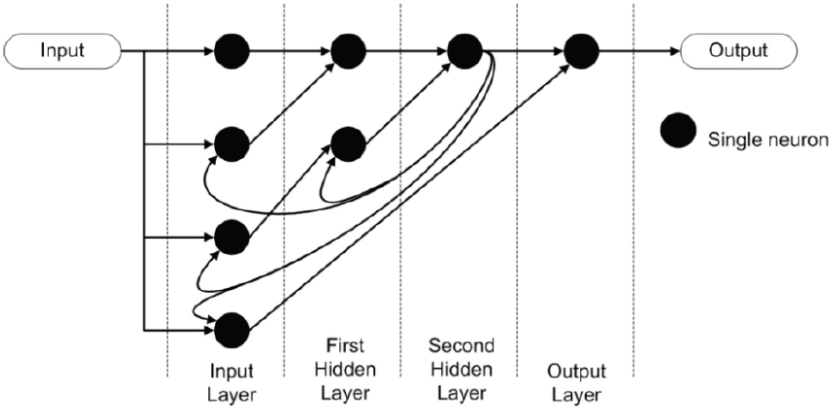

Figure 3.7: The Working Architecture of LSTM [70]

The method is implemented in R by using `train()` function with an argument `lstm` from `rnn` package [77].

3.11.4 Bayesian Regularized Neural Network

A Bayesian neural network was developed as a type of neural network applying Bayesian approach in 1992 [81], [82]. Since then, the model has been applied in many areas. The structure of BNN is similar to multi-layer feed-forward neural network, but it uses Bayesian and regularization approaches in the networks [83].

In the model, the connection parameters of the networks called weights are considered as random variables and optimized according to Bayesian concept. This means that a prior distribution is assigned to each weight. Thereby, the model is able to produce smooth fits. Then, the posterior distributions of the weights are evaluated by introducing the data to the network which helps the model in producing accurate fit. Therefore, the BNNs can produce smooth and accurate fits due to usage of prior and posterior distributions. The details about this working structure is detailed below.

The model minimizes the following objective function.

$$O = \alpha E_D + (1 - \alpha) E_W \tag{3.47}$$

where E_D is the sum of the square errors in the network outputs, E_W is the sum of

the squares of the weights, and α is the regularization parameter.

The general choice of prior distribution in BNN is the following normal density which puts more weights on the smaller parameter values.

$$p(w) = \left(\frac{1-\alpha}{\pi}\right)^{\frac{L}{2}} e^{-(1-\alpha)E_W} \tag{3.48}$$

where L denotes the number of weights. Then, the posterior distribution is calculated as follows.

$$p(w|D,\alpha) = \frac{p(D|w,\alpha)p(w|\alpha)}{p(D|\alpha)} \tag{3.49}$$

where D denotes the observed data. The probability density function of D can be derived as follows by assuming normally distributed errors.

$$p(D|w,\alpha) = \left(\frac{\alpha}{\pi}\right)^{\frac{M}{2}} e^{-\alpha E_D}. \tag{3.50}$$

If we put the prior distribution of the weights and probability density function of the observed data in the posterior distribution equation, the posterior distribution of the weights can be formulated as follows.

$$p(w|D,\alpha) = c\exp(-O) \tag{3.51}$$

where O is objective function and c is normalizing constant.

In addition to weights, the regularization parameter denoted by α is also optimized according to Bayesian approach in BNN via following function.

$$p(\alpha|D) = \frac{p(D|\alpha)p(\alpha)}{p(D)}. \tag{3.52}$$

In the optimization process of both weights and regularization parameter, The Nguyen and Widrow algorithm is considered. This algorithm assigns initial weights to parameters and optimize them using Gauss-Newton algorithm [84].

The method is implemented in R by using `brnn()` function from `brnn` package [85].

3.12 Hybrid Methods

Most of the time in practice, the time series data contains complex structure like many problems in the real-world. It means that the time series data usually exhibits linear and nonlinear patterns. They occasionally consist of pure linear or pure nonlinear structure. Because of this feature of the data, it is hard to understand the characteristic of the data under study. This case causes that most of the unique time series forecasting methods suggested as the best model cannot be universal because they suffer from capturing data structure. For example, many popular statistical time series models including ARIMA and ETS produce the forecast values by using the linear combination of past observations and this results in lack of modelling nonlinear component in the data. On the other hand, machine learning methods such as Neural Network perform poorly for the linear problem. Therefore, using an hybrid approach by combining both statistical and machine learning methods generates a model which is able to model both linear and nonlinear structures in the data and becomes a good and efficient alternative for forecasting.

According to Zhang[2], the time series consists of linear correlation structure and nonlinear component.

$$y_t = L_t + N_t \tag{3.53}$$

where L_t and N_t indicate linear and nonlinear components, respectively. Both of these components are supposed to be estimated from the studied series. First, the statistical model such as ARIMA, ETS or TBATS models the linear component. Then, the model residuals including nonlinear pattern of the data is calculated by using the following equation.

$$e_t = y_t - \hat{L}_t \tag{3.54}$$

where e_t is the model residuals at time t, y_t is the actual value of the series at time t, and $\hat{L}_t$ is the estimated forecast value at time t. At this step, the linear model may not be adequate, although no violence is seen in the residual analysis which

cannot identify the nonlinear pattern in the data. To be able to catch the nonlinear relationship, the model residuals are modelled by machine learning methods such as Neural Networks.

$$e_t = f(e_{t-1} \ldots, e_{t-n}) + \epsilon_t \tag{3.55}$$

where f is the nonlinear function used by machine learning model and ϵ_t is the random error.

At the end, the final forecast values are obtained by combining forecasts form the linear model and forecasts of the residuals.

$$\hat{y}_t = \hat{L}_t + \hat{N}_t. \tag{3.56}$$

In conclusion, this methodology has two steps. In the first step, the linear part of the data is modelled by linear models like ARIMA, ETS. Then, machine learning methods like NN is used to model the residuals of the linear model.

3.13 Performance Measures

3.13.1 Forecast Accuracy Measures

In the literature, there are several measures for comparing the forecasting performance of the model. Among these measures, three accuracy measures are used in this study.

3.13.1.1 Symmetric Mean Absolute Performance Error (sMAPE)

The symmetric mean absolute percentage error (sMAPE), developed by Makridakis in 1993 [86], is the modified version of MAPE where the divisor is half of the sum of the actual values and forecast values [87].

sMAPE is defined as follows:

$$sMAPE = \frac{2}{h} \sum_{t=n+1}^{n+h} \frac{|y_t - \hat{y}_t|}{|y_t| + |\hat{y}_t|} * 100(\%) \tag{3.57}$$

where y_t is the value of time series at time t, $\hat{y}_t$ is the forecast values, h is forecasting horizons and n is the number of data points in the sample.

The model having smallest sMAPE is considered a better forecasting technique in the comparison.

3.13.1.2 Mean Absolute Scaled Error (MASE)

In 2006, Hyndman et al. proposed a new measures for forecasting accuracy called MASE [88].

This measure provides mean absolute error of forecast values which is divided by in sample mean absolute error from naïve forecast method.

MASE is described as follows:

$$MASE = \frac{1}{h} \frac{\sum_{t=n+1}^{n+h} |y_t - \widehat{y}_t|}{\frac{1}{n-m} \sum_{t=m+1}^{n} |y_t - y_{t-m}|} \tag{3.58}$$

where y_t is the value of time series at time t, $\hat{y}_t$ is the forecast values, h is forecasting horizons,n is the number of data points in the sample and m is the time interval between successive observations.

If MASE value is less than 1, it indicates the method results in smaller error than naïve forecast method. If it is greater than 1, the reverse result can be concluded.

The aim of preference of this measure in M-Competition is to make a correction on the problems arising from sMAPE and procure an alternative [12].

3.13.1.3 The Arithmetic Mean of sMAPE and MASE

In addition to two popular accuracy measures defined above, their arithmetic mean is also considered as accuracy measure since it is hard to decide for best model when one

model has the smallest sMAPE, but higher MASE compared to the other methods. In order to calculate this suggested measure, we will use the following formula

The average of both measures is described as follows:

$$Avg = \frac{(sMAPE/100) + (MASE)}{2} \tag{3.59}$$

3.13.2 Computational Time

The computational time required by each method expressed above is considered in this study addition to three accuracy measures. To do so in R, `sys.time()` function is used and times for each method is recorded in terms of hour.

CHAPTER 4

ANALYSIS

In the analysis section, the selected series from the M4 competition data set containing series with different time frequencies are forecasted via the models explained in the Chapter 3. Then, the models are compared in terms of their forecast performance via some performance measures detailed in the previous chapter. R Studio with version 1.3.959 is used for the implementation and comparison of the models.Therefore, the analysis process are explained according to the following context in this part.

Firstly, the data sets used in the study is introduced. Then, data preprocessing techniques applied is explained. After this, selected subsets of the series from each time frequency are introduced using both visual and numerical ways. Then, the results of empirical analysis for the series from each time frequency are represented in both numerical and visual ways, seperately.

4.1 Data Introduction

In this study, the M4 competition data set, created by Makridakis on December 28th, 2017, is used. The main motivation behind this data set is to learn the ways of improving forecasting accuracy and using these ways in the theory and practice of the forecasting concept.The M4 competition data set has 100000 series consisting of yearly, quarterly, monthly, weekly, daily and hourly series from several areas including micro, finance, macro, industry, demographic and other. The number of 100000 series are taken from the database of the National Technical University of Athens (NTUA) called ForeDeCK which includes 900000 time series randomly [12].

The classification of number of the series in the data set according to their time interval and domain is given in Table 4.1.

Table 4.1: Number of M4 series for each frequency and area

Time\Domain	**Micro**	**Industry**	**Macro**	**Finance**	**DG**	**Other**	**Total**
Yearly	6538	3716	3903	6519	1088	1236	23000
Quarterly	6020	4637	5315	5305	1858	865	24000
Monthly	10975	10017	10016	10987	5728	277	48000
Weekly	112	6	41	164	24	12	359
Daily	1476	422	127	1559	10	633	4227
Hourly	0	0	0	0	0	414	414
Total	25121	18798	19402	24534	8708	3437	100000

As seen in Table 4.1, almost half of 100000 series are monthly series. The number of monthly series is followed by quarterly series, yearly series, daily series, hourly and weekly series, respectively. On the other hand, the domain of micro has the maximum number of series among 100000 series, and it is followed by finance, macro, industry, demographic and other in terms of number of series, respectively.

During the determination of the number of series in terms of time interval between successive observations and domains for M4 Competition, their usage frequencies and importance are considered. For example, monthly series are used more frequently than yearly or quarterly series in the field of business that includes micro, industry, macro and finance areas. In the same way, micro and financial series are more important than demographic series for making the decision [12].

The data set is obtained from the GitHub repository of the competition [89]. It can be also obtained from the web page of the competition and R package related to it.

In this study, since analysis of 100000 series requires a high computational demand and takes a long time, 1000 series are randomly selected in terms of time interval with respect to their corresponding proportion of the series. In this selection, the domain of the series are not considered because the data set file used in the study did not include the domain names.

Also, time frequency and forecast horizon for each series in this study are same as M4 Competition. For example, 6 forecast values are produced for yearly series whose time frequency is equal to 1. The number of series from each time frequency with their corresponding time frequency and forecast horizons are given in Table 4.2.

Table 4.2: Number of series used in the study with frequencies and forecast horizons

Time Interval	*Yearly*	*Quarterly*	*Monthly*	*Weekly*	*Daily*	*Hourly*	*Total*
Number	230	240	480	4	42	4	1000
Frequency (f)	1	4	12	52	7	24	
Horizon (h)	6	8	18	13	14	48	

4.2 Data Preprocessing

There is an argument about the effect of the pre-processing on the forecast accuracy in the literature of forecasting especially for machine learning models. Some of them claims that the stationary must be achieved in both mean and variance while rest of them claims that machine learning models are capable of modelling any pattern in the data [1].

The pre-processing on the time series data can be applied in different ways such as Box-Cox transformation or power transformation, removing trend and deseasonalization. Among these ways, only Box-Cox transformation used to achive stationarity in variance is considered in this study because both statistical and machine learning methods and also hybrid approach, except ARIMA and SARIMA, are designed with the capability of dealing with non-stationarity. Also, `auto.arima` function, which is used to fit both ARIMA and SARIMA models, removes the trend in the data without requiring any pre-processing. Therefore, the following alternatives of the series are used in this study.

- **Original Data**: None of the pre-processing methods is applied on the series.
- **Box-Cox Transformation**: Box-Cox transformation is applied on the series to assess the stationarity in variance. To do so, `BoxCox` function is used [36].

In addition to this pre-processing stage, two additional techniques explained below are used in this study.

- **Min-Max Scaling**: The series are scaled between 0 and 1 manually before the application of RNN and LSTM since the network structure has non-linear activation function. Morever, the computational time becomes shorter, and speed of the network learning becomes faster with this way [90]. On the other hand, R functions for BNN and feed-forward neural network scale the data without needing a manual scaling. The formulation of min-max scaling is given below.

$$y'_t = \frac{y_t - y_{min}}{y_{max} - y_{min}} \tag{4.1}$$

  where y'_t is the scaled observation at time t, y_t is the actual observation at time t, y_{min} and y_{max} are the minimum and maximum observations of the series, respectively.

- **Time Delay Embedding**: One of the main characteristic structure of the time series is to have the autocorrelation structure among the observations. In other words, the observations at time t depends on its past values. Although the statistical methods evaluate this property, most of the machine learning methods such as SVM, RF cannot. In this case, the inclusion of past observations in the model is the best solution to overcome this problem. The inclusion of the historical information of the series in the analysis is called a Time Delay Embedding [91]. The lag of the included past observations is determined based on the idea aiming that capturing of the second seasonal lag. For example, the lag of two years are considered in the yearly series as input data.

4.3 Model Implementation

Since 35080 models are constructed in this study, it is hard to explain the models numerically such as giving the estimated parameter values or number of neurons in the hidden layers. Instead, the general working principle of the functions used to fit both statistical and ML methods will be expressed. The specific details about

the models will be given in the following section. Besides, the techniques applied manually for the model implementation will be mentioned.

4.3.1 Statistical Models

Six statistical methods, which are NAIVE or sNAIVE, ARIMA, ETS, TBATS, LGT or SGT and STL, are fitted to produce h steps ahead forecasts for the series. Among these models, sNAIVE and STL are not fitted for the yearly series since these models are not applicable for the yearly series which does not contain any seasonal components. When the models are performed, only necessary arguments for the functions used to fit the models are tuned. In other words, most of the models used in the study are considered with their default versions.

4.3.1.1 ARIMA and SARIMA

ARIMA models are fitted for the yearly series by using `auto.arima` function from `forecast` package as stated in the previous chapter. This function applies differencing technique used to remove the stochastic trend if it exists, and makes the series appropriate for the model suggestion. Then, it determines the order of AR and MA components by the minimization of Bayesian Information Criterion which is used to compare the quality of the models. The mathematical form of BIC is given below.

$$\mathrm{BIC} = \log(n)k - 2\log(\hat{L}) \tag{4.2}$$

where n is the length of the series, k is the number of parameters to be estimated and $\hat{L}$ is the maximum value of the likelihood function in which the observed data is used. Then, it decides the best model and estimates its parameters via Maximum Likelihood Estimation which is a statistical estimation technique based on minimizing the likelihood that is the probability density calculated from the observed data.

SARIMA models are also fitted by `auto.arima` function. This function also applies the unit root tests for seasonal series and removes the seasonal unit root by taking seasonal differencing automatically, if it exists. Like determining regular AR and

MA orders, seasonal AR and MA orders are determined by minimizing BIC. Then, the best model whose parameters are estimated by Maximum Likelihood Estimation is selected via BIC.

4.3.1.2 ETS

ETS model is constructed via `ets` function as stated in the previous chapter. The algorithm of the function is able to decide type of the ETS model, whether it is multiplicative or additive, automatically. Then, it decides the best model based on BIC whose formulation given above, and estimates the parameters of the model using Maximum Likelihood Estimation.

4.3.1.3 TBATS

TBATS model is fitted by `tbats` function. In this model, the orders of ARIMA model are selected via Akaike Information Criteria (AIC). The function also uses AIC to decide the best model. The mathematical form of AIC is given in the following equation.

$$\text{AIC} = -2\log(\hat{L}) + 2k, \tag{4.3}$$

where k is the number of parameters to be estimated and $\hat{L}$ is the maximum value of the likelihood function in which the observed data is used.

4.3.1.4 LGT and SGT

LGT that stands for Local and Global Trend model is used to fit the forecast model for the yearly series instead of SGT since yearly series do not contain any seasonal components. The model is considered Bayesian extension of Holts Linear Method. Therefore, the model parameters are estimated according to Bayesian approach. This means that the prior distribution should be assigned to each parameter to be estimated. The parameters can be seen in the mathematical form of the model given in Equation

3.17. Therefore, the list of the default prior distributions of the parameters can be summarized as follows.

- σ, γ and ξ: Cauchy distribution with 0 location value and the scale parameter equals to $\frac{y_{max}}{200}$, where y_{max} is the maximum value in the series.
- b: Normal distribution with mean 0 and standard deviation $\frac{y_{max}}{200}$, where y_{max} is the maximum value in the series.
- ϕ : Uniform distribution with bounds -1 and 1, respectively.
- α and β: Uniform distribution with bounds 0 and 1, respectively.
- ρ: Uniform distribution with bounds -0.5 and 1, respectively.
- ν: Uniform distribution with bounds 2 and 20, respectively.
- τ: Beta distribution with shape parameters α =1 and β =1.

The parameters are estimated via Monte Carlo Markov Chain simulations with 4 chains each with 5000 iterations, 2500 warm-ups that identifies the early stage of the simulations where the sequences get closer to the mass of the distribution [92] and 1 thin.

SGT, which stands for Seasonal and Global Trend, is used for the series including seasonal components. This model is a Bayesian extension of Holt-Winters Exponential Smoothing as stated in the previous chapter. The working structure of the model is similar to LGT, but more parameters are considered to be estimated compared to LGT. The prior distributions of the additional parameters are listed below.

- ζ: Uniform distribution with bounds 0 and 1, respectively.
- s_t for $t \leq m$: Normal distribution with mean 1 and standard deviation 0.3.

4.3.1.5 STL Decomposition

The STL decomposition is applied onto the series containing seasonal components. The model is consturucted via `stl` function. In the model, the given series are de-

composed into trend, seasonal and remainder components. Then, the seasonally adjusted series are created. After this, forecasts are obtained after the application of ARIMA or ETS on the seasonally adjusted series and deseasonalizing using the seasonal components belongs to the last year. In this study, only ETS which is the default model in the function is applied on the seasonally adjusted series.

After constructing the models, the forecast values related to models are produced by `forecast` function from `forecast` package. Then, sMAPE and MASE are calculated using forecast values and test values for each series, and the average of each performance metrics are calculated and presented as the final results.

4.3.2 Machine Learning Models

In this part, seven machine learning models which are SVM, RF, XGBoost, Feed-forward neural network, RNN, LSTM and BNN are fit to produce h steps ahead forecasts for the series in the study. As stated in the implementation of the statistical models, models cannot be expressed numerically since the usage of large number of series. Instead, the general application principles of them will be explained.

4.3.2.1 Support Vector Machine

SVM model is built using `svm` function from `e1071` library. The radial basis function is used as kernel parameter for all series. Also, the type of SVM is determined as eps-regression for all of the series. However, the cost parameter which can be defined as a penalty term in SVM, and gamma parameter controlling the distance of influence of a single training point are tuned for each series separately by using `tune.svm`. The lag of the past values given as input to the model varies in accordance with the frequency of the series. All of the past observations until capturing the second seasonal lag for all type of series are given to the model as input variables. For example, last two previous time points are used in the yearly series or last eight previous time points are used in the quarterly series as input data. SVM model fits the model by minimizing MSE for the given data.

4.3.2.2 Random Forest

Random forest is fit via `randomForest` function. As detailed in the previous subsection, the given inputs data changes according to the time frequency of the series under the study. However, the model makes a selection and takes the necesseary inputs among given inputs. This selection is done for all of the series by `tuneRF` function.

4.3.2.3 XGBoost

XGBoost model is developed using `xgbar` function. The function applies the row-wise cross validation to identify the best number of rounds of the iterations for the boosting algorithm by avoiding overfitting. Just as other machine learning methods, the time delay embedding matrix is created and given to the XGBoost model as input data.

4.3.2.4 Feed-Forward Neural Network

Feed-forward neural network model is conducted using `nnetar` function. The function fits a feed-forward neural network with single hidden layer. The number of nodes in the hidden layer and the number of weights are determined by the function for each series separately. The all previous lags including the second seasonal lag are introduced to the network as inputs, and the network is trained for the given data by BP.

4.3.2.5 Recurrent Neural Network

Recurrent neural network is built via usage of `rnn` argument in the `trainr` function. The past observations are given to the network as the other machine learning models. Before building the model, both input and output series are scaled to have the values between 0 and 1. In addition to this data preprocessing step, the function needs three arguments which are number of hidden layer, learning rate and number of epoch. All of these arguments are determined manually for each categories of the series by

cross validation, respectively. The learning rate of the series are selected among the sample containing 0.01, 0.05, and 0.1 which are the most commonly used weights. The random sample is taken from each series, and then the model is trained for each learning rate values. Then, the learning rate producing minimum error is used for the analysis of the series. The same procedure is repeated for the selection of epoch number and hidden layers. The interval for epoch number is from 1 to 3000, and the interval for the hidden layers is from 1 to 30 for all series. After the determination of the arguments, the model is trained by BP using the scaled inputs. Then, produced arguments is transformed to original values by applying inverse min-max scaling.

4.3.2.6 Long Short Term Memory

Long Short Term Memory is built via usage of `lstm` argument in the `trainr` function. The model has same building process as Recurrent Neural Network which was described in the previous section. The only difference is the use of `lstm` argument as said before.

4.3.2.7 Bayesian Neural Network

BNN is constructed via `brnn` function. The function fits a two layer neural networks. The historical observations whose number varies according to type of the series are introduced to the network as inputs. The function scales the inputs and outputs automatically while constructing the network. It assigns a normal distribution to weight as prior distribution and optimize them via the Gauss-Newton algorithm. The model trains the network using BP and use 1000 epoch numbers to train as default.

After fitting the models, the forecast values related to models are produced by using `predict` function in R. Then, sMAPE and MASE are calculated in the way same as statistical models.

4.3.3 Hybrid Models

In this study, the following steps are followed to build the hybrid models. Firstly, all of the statistical models except naive or seasonal naive are used in the modelling of the linear component of the series. Secondly, machine learning models are performed, and the corresponding accuracy and performance measures for each model are calculated. Then, the model outperforming others is selected to model the non-linear part of the series. However, there are some cases in this study in which the second best model that demands shorter time for forecasting compared the best one is selected, if the best model requires more time for forecasting such as 14 days. After determining the type of machine learning model used as the non-linear component of the hybrid approach, it is used to construct the hybrid models by combining statistical models.

The hybrid models start with construction of the statistical model on the series. Then, the residuals are calculated by taking the difference between actual value and the fitted value. After this, selected ML model takes the residuals and models them. At the end, both statistical model and ML model produce forecast values, and both of the values are summed.

4.4 Empirical Analysis

In this part, the model performances for both original and transformed series are represented and compared in terms of sMAPE, MASE, the average of these two metrics and their computational time. In addition to this, introductory level summary information will be given for each series via both numerical and graphical ways.

As written before, any data pre-processing methods are considered in the analysis of original series. However, the series are transformed via `BoxCox` function using λ values produced by `BoxCox.lambda` that uses the Guerrero's method where λ is produced to minimize the coefficient of the variation [93] before applying the models in the analysis of transformed series. Then, the inverse Box-Cox transformation is applied on the forecast values obtained from transformed series via `InvBoxCox`, and the final forecast values are produced.

In the analysis of both original and transformed series, the forecast values for each statistical model are produced by `forecast` function from `forecast` package. In addition to `forecast` function, the forecast values of machine learning models are obtained via `predict` based on the model. Since hybrid models contain both statistical and machine learning models, the forecast values for each hybrid model are obtained by using both `forecast` and `predict` functions used for statistical and machine learning models, respectively.

After obtaining forecast values for the models in the study, sMAPE and MASE are calculated using both forecast values and test values for each series. In addition to them, the average of both sMAPE and MASE are calculated. Lastly, the computational time in hour used for forecasting by each model is recorded and represented as written above.

4.4.1 Yearly Series Analysis

As mentioned in the introduction part of this chapter, 230 yearly series are randomly selected among 23000 yearly series being part of the M4 competition data set. The time interval between successive observations known as time frequency is considered as 1 for yearly series and 6 steps ahead forecast values are produced for all of the 230 series. In Figure 4.1, the time series plot of four series randomly selected among 230 series are drawn.

Figure 4.1 shows that all of the series are not stationary in mean. The first two series contain some ups and downs, but last two series show an increasing trend.

The maximum and the minimum lengths of the yearly series used in the study are 13 and 222, respectively. The corresponding frequencies for the maximum and minimum lengths are 3 and 1, respectively. Moreover, the mode of the length of the yearly series in the study is 40 appearing 37 times. Besides, there are 12 series whose length frequency is 1.

The analysis of the yearly series are divided into original series and transformed series. For both cases, the time frequency is considered as 1, and 6 steps ahead forecast values are produced as given in the previous section.

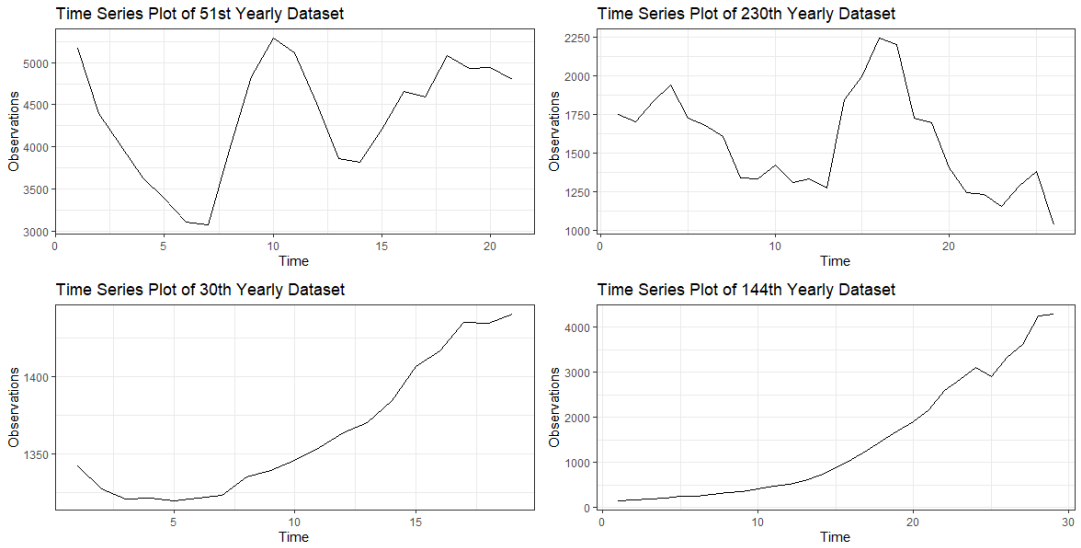

Figure 4.1: The Time Series Plots of Subset of Yearly Series

4.4.1.1 Original Yearly Series Analysis

In this section, 6 steps ahead forecasting performance of the models on the original yearly series will be presented with the corresponding accuracy and performance measures. Besides, some specific details about the models will be given.

Statistical Models: Five statistical methods which are naive, ARIMA, ETS, TBATS and LGT are fitted to produce 6 steps ahead forecasts for the yearly series. STL decomposition is not fitted since the concept of the model is not applicable for the yearly series which does not contain any seasonal components. The principles which are followed in the construction of the models are given in the previous chapter.

The performance of the models calculated in the way expressed above are represented in Table 4.3.

Table 4.3: The Forecasting Performance of Statistical Methods for Original Yearly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
NAIVE	16.499	3.802	1.983	**0.001**
ARIMA	16.298	3.467	1.819	0.004
ETS	16.468	3.611	1.888	0.006
TBATS	16.057	3.501	1.830	0.033
LGT	**15.131**	**3.169**	**1.660**	3.303

Table 4.3 shows that the LGT outperforms the other statistical methods in terms of sMAPE and MASE. In terms of sMAPE, TBATS is the second successful model for the six steps ahead forecast of the yearly series compared to others. However, ARIMA is the second successful model compared to others in terms of MASE. That's why it is better to compare the model by considering average of these two measures to solve this contradiction. According to average of both measures, LGT outperforms the other models, and it is followed by ARIMA, TBATS, ETS and NAIVE. NAIVE has the lowest forecasting accuracy in terms of all accuracy measures.

As opposed to its best performance, LGT needs more computation time for forecasting compared to other models. On the other hand, naive model having the most inaccurate forecast values demands the shortest time among all of the statistical models.

Machine Learning Models: In this part, seven machine learning models which are SVM, RF, XGBoost, feed-forward neural network, RNN, LSTM and BNN are fit to produce six steps ahead forecasts for the yearly series. As stated before, the lag of last two years observations are used as input variables in the training process of the models for yearly series. Moreover, the learning rate of the selected rates for RNN and LSTM are 0.05 and 0.1, respectively. Both of the models have 17 hidden layers, and use 1000 epoch for training. The performance of the models are represented in Table 4.4.

Table 4.4: The Forecasting Performance of Machine Learning Methods for Original Yearly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SVM	15.018	3.271	1.710	0.030
RF	19.229	4.516	2.354	0.003
XGBoost	17.905	3.937	2.058	0.026
NNETAR	22.997	5.192	2.711	0.003
RNN	**5.482**	**1.159**	**0.607**	0.826
LSTM	8.008	1.885	0.982	1.520
BNN	12.089	2.495	1.308	**0.002**

Table 4.4 shows that the RNN clearly outperforms the other ML models in terms of all accuracy measures. Additionaly, LSTM can be stated as the second superior model for the forecasting of original yearly series based on all accuracy measures. On the other hand, feed-forward neural network called NNETAR shows the worst forecasting performance compared to the other methods in terms of all accuracy measures. In addition to NNETAR, RF model gives insufficient forecasting accuracy among all machine learning for the six steps ahead forecasting of the original yearly series.

In addition to comparison of the models in the accuracy measures, the models are compared based on their computational time used for forecasting. LSTM takes the longest time in hour to produce the forecast values. After this, RNN being the superior model needs the second longest time to predict the future values for yearly series. On the other hand, the shortest time in hour belongs to the BNN. Besides, RF considered as one of the insufficient forecasting model produces the forecast values in shorter time compared to other machine learning models except BNN.

Therefore, RNN is the best ML models to produce the six steps ahead forecasts compared to other ML models in the study, selected to model the residuals coming from the statistical models in the hybrid models.

Hybrid Models: In this part, all statistical models except naive model are combined with RNN with learning rate 0.05, 1000 epoch and 17 hidden numbers to construct the hybrid models. The six steps ahead forecasts performance of the hybrid models are given in Table 4.5.

Table 4.5: The Forecasting Performance of Hybrid Methods for Original Yearly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
ARIMA-RNN	8.465	1.799	0.942	0.650
ETS-RNN	8.227	1.848	0.965	**0.649**
TBATS-RNN	8.441	1.785	0.935	0.756
LGT-RNN	**6.476**	**1.363**	**0.741**	3.939

As shown in Table 4.5, a hybrid LGT and RNN clearly outperforms the other hybrid models in terms of all accuracy measures. Besides, a hybrid ETS and RNN is the second successful model for the six steps ahead forecast of the yearly series compared to others in terms of sMAPE. However, a hybrid TBATS and RNN can be seen as the second successful model for the six steps ahead forecast of the yearly series compared to others based on MASE. A hybrid TBATS and RNN is also superior to both a hybrid ETS and RNN and a hybrid ARIMA and RNN with respect to the average of both sMAPE and MASE.

In the comparison of the models in terms of computational time, a hybrid LGT and RNN giving the most accurate forecast values among all hybrid model takes the longest time in hour to produce the forecast values while the shortest time in hour belongs to the ETS-RNN being superior model in terms of sMAPE

Therefore, LGT-RNN is the best hybrid approach to produce the six steps ahead forecasts compared to other hybrid models used for the yearly series. However, the model cannot be concluded as the better than best machine learning model.

Result: It can be said that the RNN shows the best performance for the six steps ahead forecasts of the original yearly series. The second forecasting model having

high forecasting accuracy is the hybrid of LGT and RNN. On the other hand, feed-forward neural network and RF models produce the most inaccurate forecasting values for original yearly series. It can be also stated that the averages of the performance measures of the all hybrid models are smaller than the averages of both statistical and machine learning models, although any of the hybrid models show the best forecasting performance.

In addition to this comparison, the accuracy performance of the models are compared visually. The following Figure 4.2, Figure 4.3 and Figure 4.4 represent box plot of each measure drawn by using the error values obtained from each model for the each yearly series in the analysis.

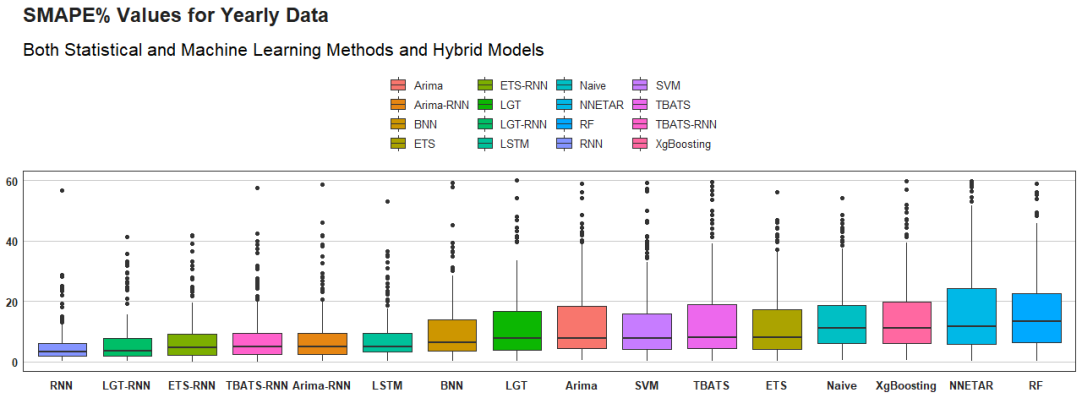

Figure 4.2: sMAPE Performance of All Methods for Original Yearly Series

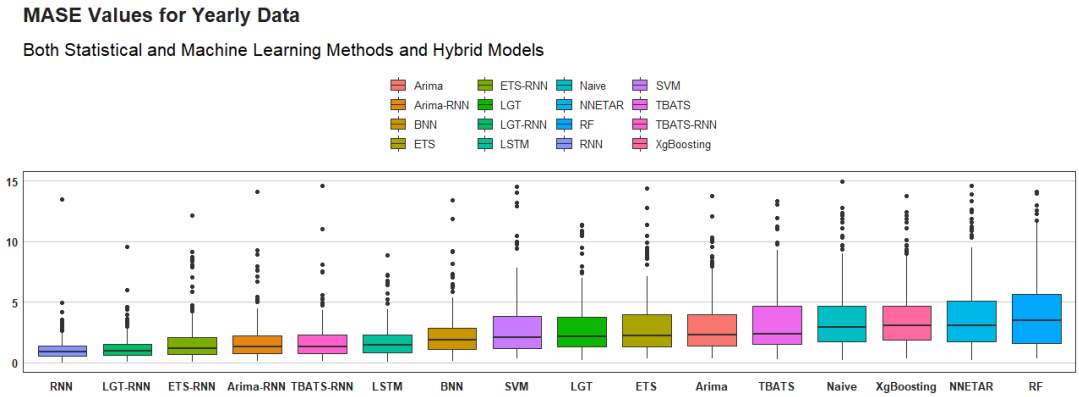

Figure 4.3: MASE Performance of All Methods for Original Yearly Series

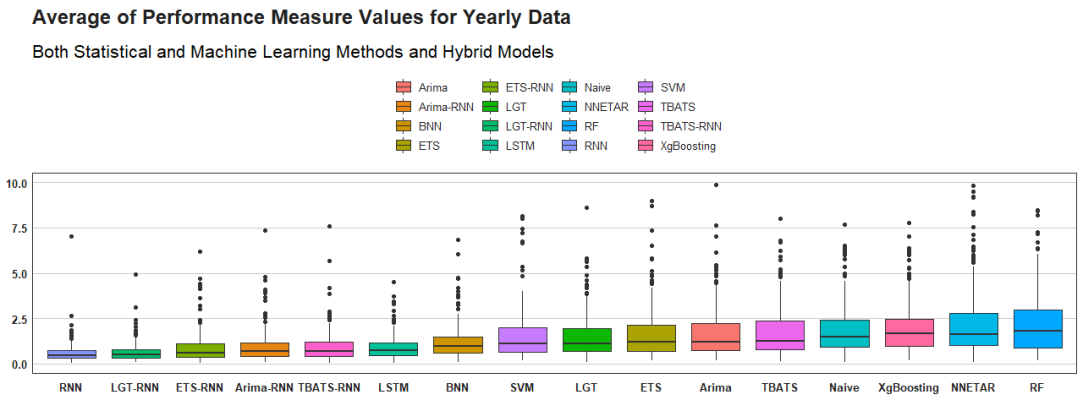

Figure 4.4: The Average of sMAPE and MASE of All Methods for Original Yearly Series

Figure 4.2, 4.3 and 4.4 show that feed-forward neural network, RF and naive model have more variations and higher error values compared to other models. On the other hand, RNN and the hybrid models being more accurate than the other models have less variation and lower error values on the average.Lastly, it can be said that some models have error values that can be investigated as outlier observations that violates the model performances.

4.4.1.2 Transformed Yearly Series Analysis

In this section, the six steps ahead forecasting performance of the models on the transformed series will be presented with the corresponding accuracy and performance measures. Besides, the details about the models will be given. The application of the models on the transformed series are same as the original ones. The transformation process of the series is explained in Section 4.4.

Statistical Models: Five statistical methods which are naive, ARIMA, ETS, TBATS and LGT are fitted to produce 6 steps ahead forecasts for the yearly series. STL is not fitted since the model is not applicable for the yearly series which does not contain any seasonal components as stated in the previous section. The values of forecasting accuracy measures of the models for transformed yearly series are represented in Table 4.6.

Table 4.6: The Forecasting Performance of Statistical Methods for Transformed Yearly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
NAIVE	32.900	**3.802**	**2.065**	**0.001**
ARIMA	17.826	4.428	2.303	0.004
ETS	17.232	4.286	2.229	0.001
TBATS	**17.155**	4.877	2.524	0.025
LGT	20.578	7.831	4.018	3.291

Table 4.6 shows that TBATS outperforms the other statistical models models for the six steps ahead forecast of the yearly series compared to others after applying Box-Cox transformation according to sMAPE. On the other hand, naive model has the best MASE value compared to other models interestingly. Naive model can be also considered as the model showing the best forecasting performance in terms of arithmetic average of sMAPE and MASE. As opposed to its successful performance on the original series, LGT shows the worst forecasting performance if it is applied on the transformed yearly series.

LGT reporting the poorest forecasting accuracy based on two criteria needs the longest time to predict the future values of yearly series with Box-Cox transformation among all statistical methods. In addition to best performance, naive method requires the shortest time in hour to produce the forecasts.

Machine Learning Models: In this part, seven machine learning models which are SVM, RF, XGBoost, Feed-forward neural network, RNN, LSTM and BNN are fit to produce six steps ahead forecasts for the yearly series The learning rate of the selected rates for RNN and LSTM are 0.05 and 0.1, respectively. Both of the models has 17 hidden layers and uses 1000 epoch for training. The accuracy measures for the models are shown in Table 4.7.

Table 4.7: The Forecasting Performance of Machine Learning Methods for Transformed Yearly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SVM	13.651	2.856	1.496	0.028
RF	19.711	4.629	2.413	0.003
XGBoost	19.809	4.477	2.338	0.0185
NNETAR	23.860	7.686	3.962	0.0031
RNN	**5.575**	**1.196**	**0.626**	0.7517
LSTM	7.629	1.836	0.956	1.515
BNN	11.751	2.359	1.238	**0.002**

As seen in Table 4.7 shows that the RNN clearly outperforms the other ML models in terms of all accuracy measures. LSTM can be stated as the second successful forecasting model model for the six steps ahead forecast of the transformed yearly series compared to others based on all criteria. On the other hand, feed-forward neural network shows the worst forecasting performance compared to the other methods in terms of all accuracy measures. After this model, XGBoost and RF models produce the most inaccurate forecast values for transformed yearly series.

The comparison of models in terms of computational time shows that LSTM takes the longest time in hour to produce the forecast values while the shortest time in hour belongs to the BNN.

Therefore, RNN is the best ML models to produce the six steps ahead forecasts compared to other ML models in the study, selected to model the residuals coming from the statistical models in the hybrid models.

Hybrid Models: In this part, all statistical models except naive model are combined with RNN with learning rate 0.05, 1000 epoch and 17 hidden numbers to construct the hybrid models. The six steps ahead forecasts performance of the hybrid models applied on the transformed are given in Table 4.8.

Table 4.8: The Forecasting Performance of Hybrid Methods for Transformed Yearly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
ARIMA-RNN	9.256	1.928	1.010	**0.590**
ETS-RNN	**8.676**	**1.824**	**0.955**	0.657
TBATS-RNN	9.208	1.960	1.026	0.654
LGT-RNN	9.256	2.036	1.065	4.521

Table 4.8 shows that a hybrid ETS and RNN clearly outperforms the other hybrid models in terms of all accuracy measures. In terms of sMAPE, a hybrid TBATS and RNN is the second successful model for the six steps ahead forecast of the yearly series compared to others. However, a hybrid ARIMA and RNN is the second successful model for the six steps ahead forecast of the yearly series compared to others in terms of not only MASE but also the average of both sMAPE and MASE. On the other hand, it is seen that a hybrid LGT and RNN reports the worst accuracy values based on all criterias. Besides, the hybrid ARIMA and RNN reported as the second superior model based on MASE and the average can be considered as unsuccessful forecasting model together with the hybrid LGT and RNN with respect to sMAPE.

In the comparison of compuational time, the hybrid LGT and RNN demands the longest time in hour to produce the forecast values while the shortest time in hour belongs to the hybrid ARIMA and RNN.

Therefore, ETS-RNN is the best hybrid approach to produce the six steps ahead forecasts compared to other hybrid models used for the transformed yearly series. However, the model cannot be concluded as the best forecasting model for transformed yearly series.

Result: It is seen that RNN gives best forecasting accuracy among all models for the six steps ahead forecasts. After RNN, LSTM can be stated as the second forecasting model producing accurate forecasts. On the other hand, naive and feed- forward neural network models give the lowest forecasting accuracy values, respectively. It is

also shown that the averages of the performance measures of the all hybrid models are smaller than the averages of both statistical and machine learning models, although any of the hybrid models show the best forecasting performance.

In addition to this comparison based on numerical values, the accuracy performance of the models are compared visually. The box plot of accuracy values drawn using error values produced by each model are represented in the following Figure 4.5, Figure 4.6 and Figure 4.7, respectively.

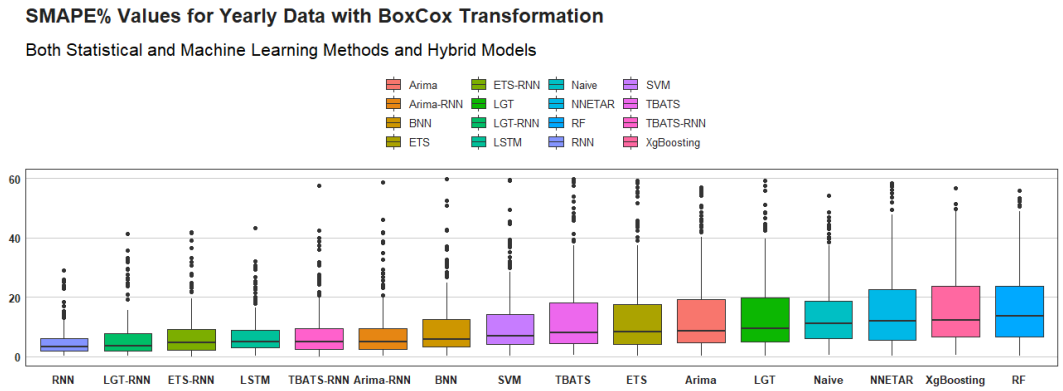

Figure 4.5: sMAPE Performance of All Models for Transformed Yearly Series

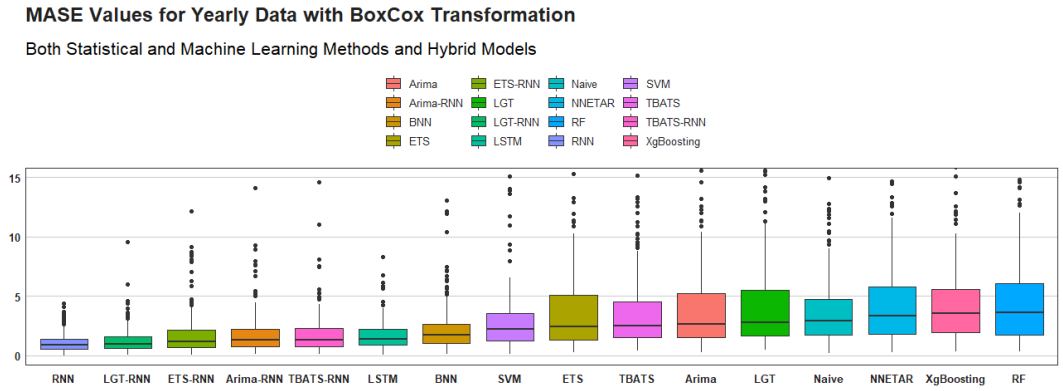

Figure 4.6: MASE Performance of All Models for Transformed Yearly Series

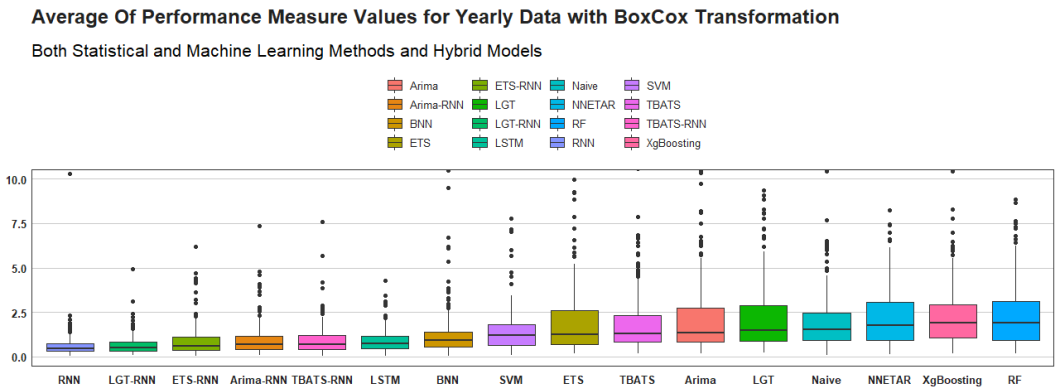

Figure 4.7: The Average Performances of All Models for Transformed Yearly Series

As displayed in Figure 4.5, 4.6 and 4.7, the most inaccurate models such as feed-forward neural network, naive model, RF and LGT have more variations and higher error values on the average compared to other models. On the other hand, RNN, LSTM and the hybrid models which can be classified as more accurate models have lower variations and error values on the average in terms of accuracy measures. Lastly, the figures represents that all models have error values that can be detected as outlier observations which violates the model performances.

In addition to this, the transformation results in increasing in the error measures for statistical models and hybrid models. However, it does not have a considerable effect on the performance of ML models.

4.4.2 Quarterly Series Analysis

240 quarterly series selected from M4 competition data set are used to compare the forecasting performance of the models in the study. The time interval between successive observations known as time frequency is considered as 4 for quarterly series and 8 steps ahead forecast values are produced for all of the 240 series. In Figure 4.8, the time series plot of four series randomly selected among 240 series are drawn.

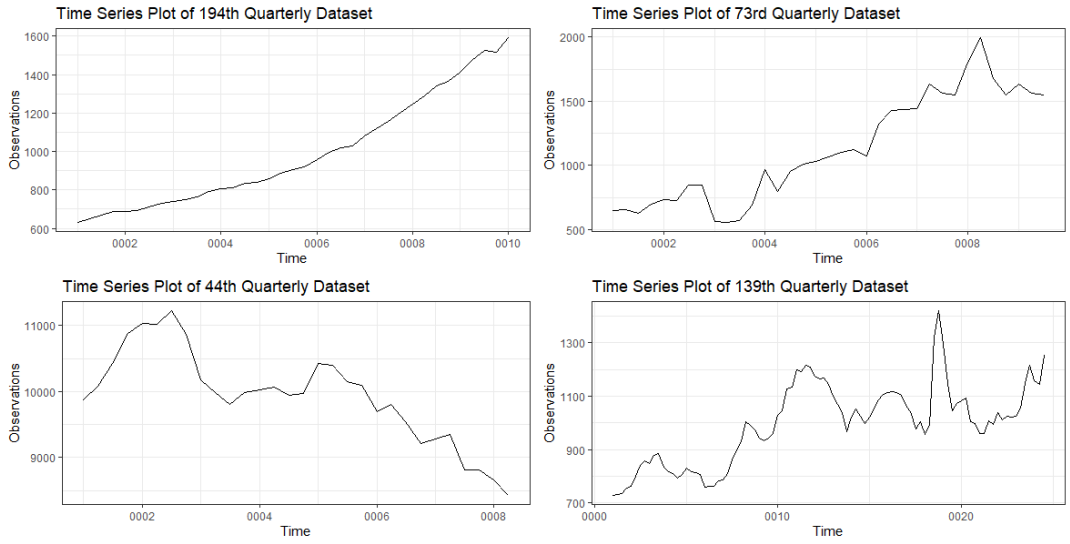

Figure 4.8: The Time Series Plots of Subset of Quarterly Series

As seen in Figure 4.8, all series are nonstationary. Also, all series show the increasing pattern except for the third one. They give us some clues about suffering unit root problems in their time series plots. Moreover, seasonal patterns can be observed from the graph, especially in the plot of the last series.

The maximum and the minimum length of the quarterly series used in the study are 21 and 279, respectively. The series having minimum and maximum length appears in the study once, separately. Moreover, the mode of the length of the quarterly series in the study is 114 appearing 12 times.

The quarterly series are analyzed in two different sections, original series and transformed series. For both cases, the time frequency is used as 4, and 8 steps ahead forecast values are produced as given in the previous section.

4.4.2.1 Original Quarterly Series Analysis

In this section, 8 steps ahead forecasting performance of the models on the quarterly original series are presented with the corresponding accuracy and performance measures. Besides, some specific details about the models are given.

Statistical Models: Six statistical methods which are sNAIVE, SARIMA, ETS, TBATS, SGT and STL are fitted to produce eight steps ahead forecast values for the quarterly series. Since the series have the seasonal cycles in their structure, STL is included in the study of quarterly series differently from the analysis of yearly series. The general details for the construction of the models are given in the section of model implementation.

The forecasting performance of the statistical models for original quarterly series are shown in Table 4.9

Table 4.9: The Forecasting Performance of Statistical Methods for Original Quarterly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
sNAIVE	13.753	1.659	0.898	**0.001**
SARIMA	10.612	1.093	0.599	0.017
ETS	**9.834**	1.065	0.581	0.011
TBATS	10.053	1.058	**0.579**	0.113
SGT	10.341	1.070	0.587	16.435
STL	13.845	**1.053**	0.596	0.004

Table 4.9 shows that the performance of the most of the models are close to each other and that's why it is hard to suggest a model being successful in forecasting. Exponential smoothing model known as ETS outperforms the other statistical methods in terms of sMAPE. On the other side, STL performs the best eight steps ahead forecasts performance compared to other methods according to MASE. However, the average of both sMAPE and MASE suggests TBATS has the best performance rather than STL and ETS on the average. The similar situation is also valid for determining the model producing inaccurate forecast values. Although STL gives the lowest forecasting accuracy in terms of sMAPE, seasonal naive model has the worst eight steps ahead forecasts performance among all statistical methods according to both MASE and the average of MASE and sMAPE.

In terms of computational time demanding to predict the forecasts for 240 quarterly series, SGT has the longest time period in hours since it uses MCMC algorithm to es-

timate the model parameters. Seasonal naive is the model that produces the forecasts in shorter time period when compared to other statistical models.

Machine Learning Models: In this part, seven machine learning models which are SVM, RF, XGBoost, feed-forward neural network, RNN, LSTM and BNN are fit to produce eight steps ahead forecasts for the quarterly series. The past observations until the second seasonal lag which is eight are used as input data to train the models for the quarterly series. Since the general structure of the models are given above, only specific details about RNN and LSTM will be given in this part. Both of the models train their network structure with leaning rate 0.05, 2000 epoch and 28 hidden layers. The eight steps ahead forecasts performance of the machine learning models are described in Table 4.10

Table 4.10: The Forecasting Performance of Machine Learning Methods for Original Quarterly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SVM	11.054	1.438	0.774	0.058
RF	11.434	1.476	0.795	0.008
XGBoost	13.391	1.568	0.851	0.053
NNETAR	14.458	1.686	0.915	**0.001**
RNN	**7.066**	**0.823**	**0.447**	11.302
LSTM	7.846	0.843	0.461	49.208
BNN	9.207	0.918	0.505	0.002

Table 4.10 shows that the RNN clearly outperforms the other ML models in terms of all accuracy measures. In addition, it is easily seen that LSTM is the second successful model for the eight steps ahead forecasts of the quarterly series with respect to all measures. On the other hand, feed-forward neural network shows the worst forecasting performance compared to the other methods in terms of all accuracy measures. XGBoost model also give lowest accuracy values for original quarterly series right after NNETAR.

In addition to comparision in the accuracy values, the computational time of the models shows that LSTM needs more than two days to produce the forecast values for the quarterly series. However, BNN produces the forecasts for 240 series in the shortest time period compared to other models.

Therefore, RNN is regarded as the machine learning model showing the best eight steps ahead forecasts performance for the quarterly series among all of the ML models in the study. Due to its performance, RNN with related model parameters are used as non-linear component of the hybrid models that will be explained in the following section.

Hybrid Models: In this part, all statistical models except seasonal naive model are combined with RNN with learning rate 0.05, 2000 epoch and 28 hidden numbers to construct the hybrid models. The performance of eight steps ahead forecasts performance of the hybrid models are given in Table 4.11. The models have 17 hidden layers and use 1000 epoch for training.

Table 4.11: The Forecasting Performance of Hybrid Methods for Original Quarterly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SARIMA-RNN	8.377	0.817	0.450	18.469
ETS-RNN	7.075	0.708	0.389	18.805
TBATS-RNN	7.442	0.737	0.406	19.261
SGT-RNN	7.443	0.679	0.377	38.112
STL-RNN	**6.476**	**0.637**	**0.351**	**16.289**

Table 4.11 shows that hybrid approach using STL and RNN clearly outperforms the other hybrid models on the basis of all of the accuracy measures. The performance of this model is followed by TBATS-RNN hybrid model in terms of sMAPE and SGT-RNN in terms of MASE and the average of both measures. The worst performance among the hybrid approaches belongs to SARIMA-RNN approach on the basis of three accuracy measures.

In the comparison of the model in terms of their computational requirements, the hybrid SGT and RNN demands more computational time in hours in comparison with other models, while the hybrid STL and RNN needs the shortest time period in hours to produce the eight steps ahead forecasts for the quarterly series in addition to its best performance.

In conclusion, a hybrid STL and RNN model shows the best performance among the all hybrid approaches applied on the quarterly series according to accuracy measures and computational time. Different than yearly series, the one of the hybrid model which is the hybrid STL decomposition and RNN is concluded as the best forecasting model in the analysis of original quarterly series.

Result: It can be said that the hybrid approach STL and RNN has the lowest error values compared to all models in the study. The second model having highest forecast accuracy is also a hybrid model which is the hybrid model ETS and RNN in this analysis. On the other hand, feed- forward neural network and XGBoost models produce the most inaccurate forecasting values for original quarterly series. Lastly, we can state that the hybrid models show the best forecasting performance, and they outperform the both statistical and machine learning models on the average.

The visual comparison of the models are illustrated in Figures 4.9, 4.10 and 4.11. These figures show box plot for each model drawn by error values calculated using forecast values and test data set.

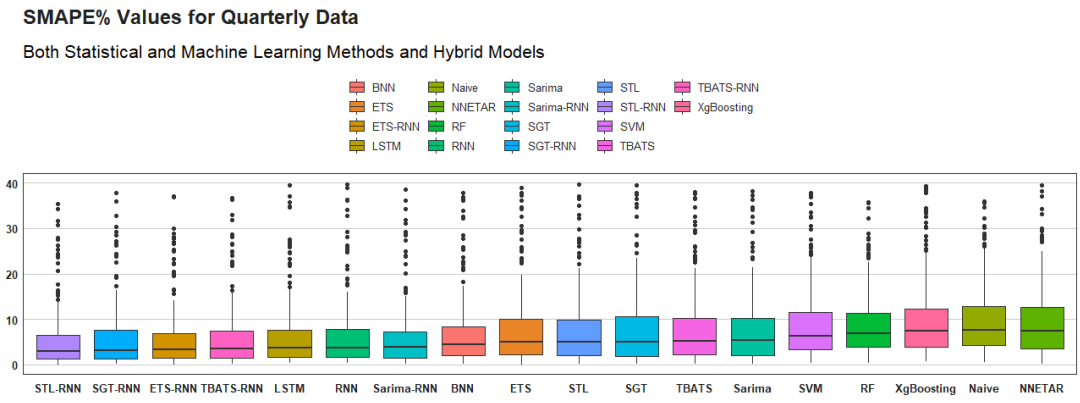

Figure 4.9: sMAPE Performance of All Models for Original Quarterly Series

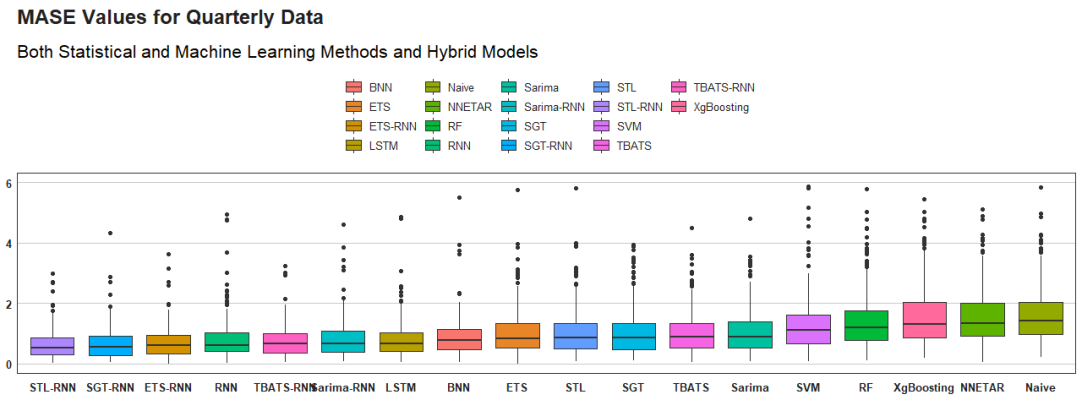

Figure 4.10: MASE Performance of All Models for Original Quarterly Series

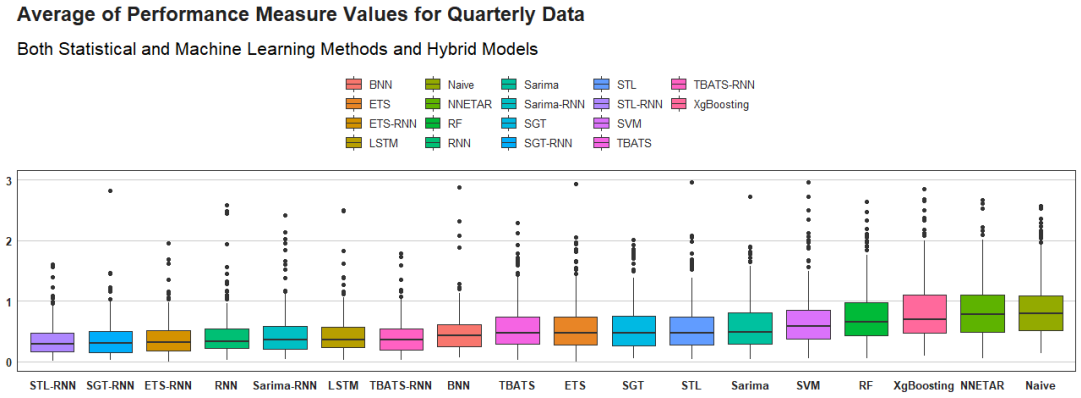

Figure 4.11: The Average Performances of All Models for Original Quarterly Series

Figures 4.9, 4.10 and 4.11 show that feed-forward neural network, XGBoost and naive models, which are the model having lower forecasting accuracy compared to other models, have more variations and higher error values different than the other models in the study. On the other hand, hybrid models particularly the hybrid STL decomposition and RNN have less variations and lower error values on the average compared to other models in the study. Finally, all models have outlier error values which have a negative effect on the model performances.

4.4.2.2 Transformed Quarterly Series Analysis

In this section, the eight steps ahead forecasting performance of the models on the transformed quarterly series will be presented with their corresponding accuracy and

performance measure values. Besides, the details about the models will be given. The application of the models on the transformed series are same as the original ones. The transformation process of the series is explained in Section 4.4.

Statistical Models: Six statistical methods which are sNAIVE, SARIMA, ETS, TBATS, SGT and STL are fitted to produce eight steps ahead forecasts for the quarterly series. The forecasting performances of the statistical methods on the transformed quarterly series is summarized in Table 4.12

Table 4.12: The Forecasting Performance of Statistical Methods for Transformed Quarterly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
sNAIVE	13.753	1.659	0.666	**0.001**
SARIMA	10.956	1.222	0.643	0.003
ETS	10.853	1.177	0.612	0.002
TBATS	**10.348**	**1.119**	**0.611**	0.173
SGT	13.963	2.115	0.898	34.416
STL	13.834	1.180	0.659	**0.001**

Table 4.12 states that TBATS outperforms the other statistical models for eight steps ahead forecast of the quarterly series transformed by Box-Cox in regards to all accuracy measures. The performance of TBATS is followed by exponential smoothing method on the basis of all accuracy measures. On the other hand, the worst forecasting performances in the statistical models arises from SGT because of having convergence problem raised by usage of default model. In addition to SGT, seasonal naive model can be reported as a model producing inaccurate forecast values based on MASE and the arithmetic average of both sMAPE and MASE.

In terms of computational requirements, SGT takes the longest time compared to other models. The shortest time period to obtain the forecasts for the series is needed by seasonal naive model.

Machine Learning Models: In this part, seven machine learning models which are SVM, RF, XGBoost, feed-forward neural network, RNN, LSTM and BNN are fit to produce eight steps ahead forecasts using the transformed quarterly series. The parameters of RNN and LSTM are same as the ones applied on the original quarterly series. The forecasting performance of the ML methods are represented in Table 4.13.

Table 4.13: The Forecasting Performance of Machine Learning Methods for Transformed Quarterly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SVM	10.721	1.434	0.771	0.055
RF	11.382	1.482	0.798	0.007
XGBoost	16.035	2.148	1.154	0.024
NNETAR	14.631	1.669	0.908	0.009
RNN	**7.029**	0.831	0.451	14.862
LSTM	7.356	**0.820**	**0.446**	40.269
BNN	8.233	0.873	0.478	**0.002**

Table 4.13 shows that the RNN clearly outperforms the other ML models in terms of both sMAPE, and the model is followed by LSTM. However, LSTM takes the place of RNN on the basis of MASE and the arithmetic average of sMAPE and MASE, and followed by RNN. On other hand, XGBoost shows the worst forecasting performance compared to the other methods in terms of all accuracy measures. After XGBoost model, NNETAR can be concluded as the model having insufficient forecast performance compared to other ML models.

According to the computational time for forecasting, LSTM needs the longest time period in hours to produce the forecast values for the quarterly series, while BNN produces the forecasts for 240 series in the shortest time period in hours. Although LSTM has the best forecasting performance on the average, RNN is used in the construction of the hybrid approach because of longest time of LSTM. The reason of selecting RNN is to become the second successful model and have accuracy measure values being very close to LSTM. It also requires shorter time period compared to

LSTM.

Therefore, LSTM is regarded as the machine learning model showing the best eight steps ahead forecasts performance for the quarterly series among all of the ML models in the study. However, RNN with related model parameters are used as non-linear component of the hybrid models that will be explained in the following section because of high computational requirements of LSTM.

Hybrid Models: In this part, all statistical models except naive model are combined with RNN with learning rate 0.05, 2000 epoch and 28 hidden numbers to construct the hybrid models. The forecasting performance of hybrid methods for the transformed quarterly series are represented in Table 4.14.

Table 4.14: The Forecasting Performance of Hybrid Methods for Transformed Quarterly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SARIMA-RNN	7.998	0.834	0.457	**18.815**
ETS-RNN	7.447	0.746	0.410	18.987
TBATS-RNN	9.506	1.011	0.553	19.886
SGT-RNN	11.572	1.564	0.840	105.432
STL-RNN	**6.774**	**0.649**	**0.359**	18.947

As seen in Table 4.14, a hybrid model STL decomposition and RNN clearly outperforms the other hybrid models on the basis of all of the accuracy measures. The performance of this model is followed by the hybrid model ETS and RNN hybrid model in terms of all measures. The worst performance among the hybrid approaches belongs to the hybrid model SGT and RNN according to sMAPE, MASE and their equally weighted average. A hybrid model TBATS and RNN can be also concluded as the forecasting model with lowest forecasting accuracy among all hybrid models right after a hybrid model SGT and RNN.

In the comparison of the models in regards to their computational requirements, the hybrid model SGT and RNN demands more computational time in hours in compari-

son with other models, the shortest time period to produce eight steps ahead forecast values belongs to the hybrid model SARIMA and RNN. However, the hybrid model STL decomposition and RNN showing the best forecasting performance has a computational time which is very close to the shortest one.

In conclusion, a hybrid STL and RNN model shows the best performance among the all hybrid approaches applied on the transformed quarterly series according to forecast performance measures. Different than yearly series, the one of the hybrid models which is the hybrid STL decomposition and RNN is concluded as the best forecasting model in the analysis of transformed quarterly series.

Result: It can be said that the hybrid approach STL and RNN has the lowest error values compared to all models in the study. The second model having highest forecast accuracy is also a hybrid model which is the hybrid model ETS and RNN in this analysis. On the other hand, feed-forward neural network and XGBoost models produce the most inaccurate forecasting values for transformed quarterly series. Lastly, we can state that the hybrid models show the best forecasting performance, and they outperform the both statistical and machine learning models on the average.

The distribution of the accuracy measure regarding to models in the study are illustrated via box plot in Figures 4.12, 4.13 and 4.14.

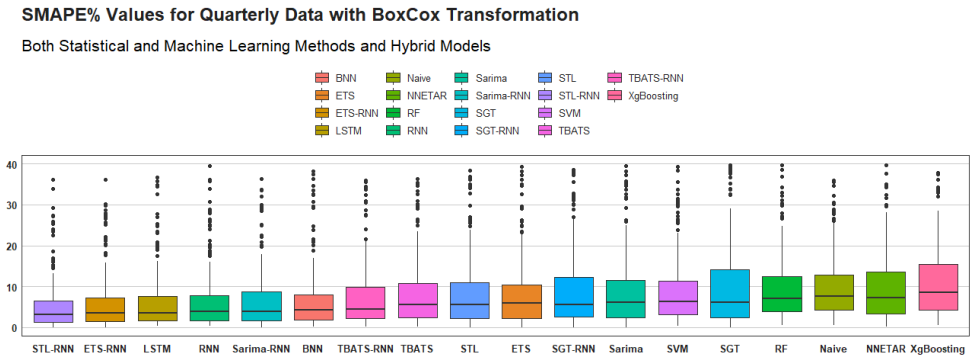

Figure 4.12: sMAPE Performances of All Models for Transformed Quarterly Series

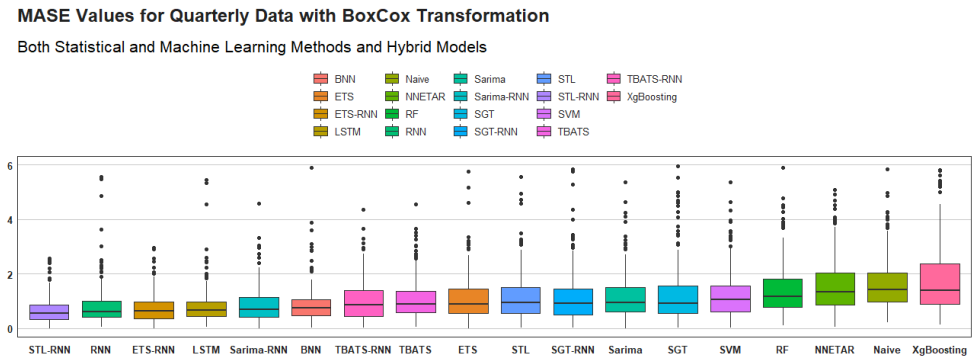

Figure 4.13: MASE Performances of All Models for Transformed Quarterly Series

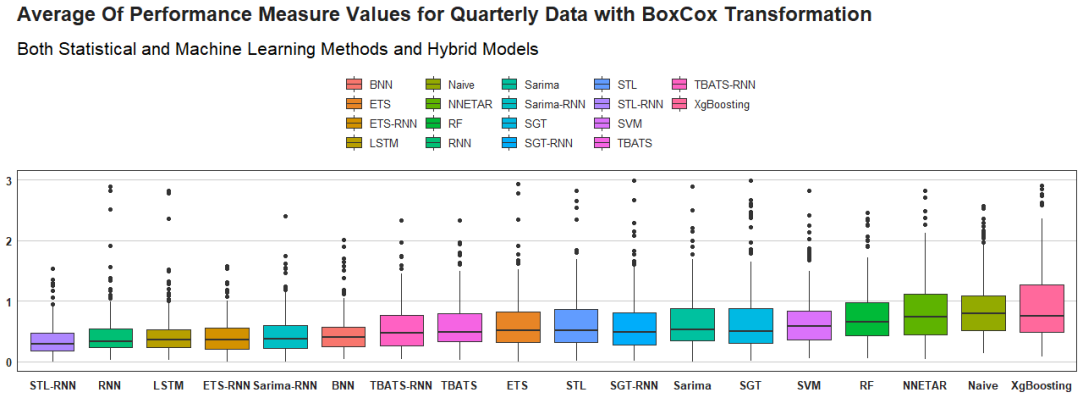

Figure 4.14: The Average Performances of All Models for Transformed Quarterly Series

Figures 4.12, 4.13 and 4.14 display that the models having lowest forecasting accuracy such as SGT, XGBoost and NNETAR represents the higher variability and error values on the average compared to other models. On the other hand, hybrid models particularly the hybrid STL decomposition and RNN have less variations and lower error values on the average compared to other models in the study. Finally, all models have outlier error values which have a negative effect on the model performances.

Lastly, it is also observed that the error values for statistical models and hybrid models increase after Box-Cox transformation. However, the similar conclusion cannot be made for the ML models.

4.4.3 Monthly Series Analysis

In this study, 480 monthly series selected from M4 competition data set are used and the forecast performance of the models included in the study are compared using these selected series. The time interval between successive observations known as time frequency is considered as 12 for monthly series and 18 steps ahead forecast values are produced for all of the 480 series. In Figure 4.15, the time series plot of four series randomly selected among 480 series are drawn.

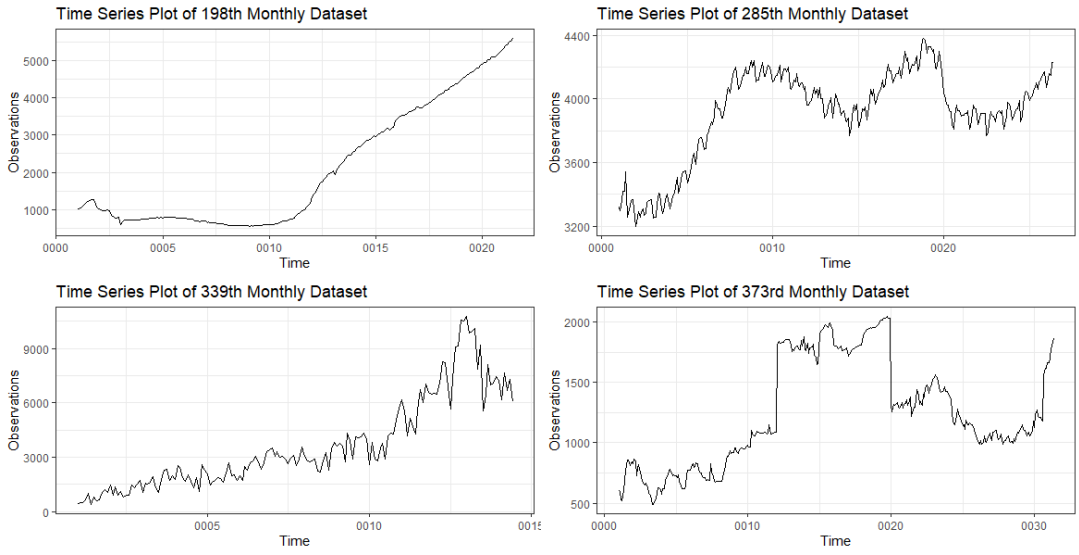

Figure 4.15: The Time Series Plots of Subset of Monthly Series

As seen in Figure 4.15, all series can be interpreted as nonstationary. They may have unit root problems. Also, all series show the increasing pattern with seasonal cycles. Lastly, it can be said that there is a time shift in the plot of the last series.

The maximum and the minimum lengths of the monthly series used in the study are 43 and 918, respectively. The frequencies of both minimum and maximum length of the series are 1 and 2, respectively. Moreover, the mode of the length of the quarterly series in the study is 306 whose appearance in the study is 61.

The analysis of the monthly series are divided into two groups, original series and transformed series. For both cases, the time frequency is used as 12, and 18 steps ahead forecast values are produced as given in the previous section.

4.4.3.1 Original Monthly Series Analysis

In this section, 18 steps ahead forecasting performance of the models on the monthly original series will be presented with the corresponding accuracy and performance measures. Besides, the details about the models will be given.

Statistical Models: Six statistical methods which are sNAIVE, SARIMA, ETS, TBATS, SGT and STL decomposition are fitted to produce six steps ahead forecasts for the original monthly series. Since the series have the seasonal cycles in their structure, STL is included in the study of quarterly series differently from the analysis of yearly series. The general details for the construction of the models are given in the section of model implementation.

The forecasting performance of the statistical methods for the original monthly series are represented in Table 4.15

Table 4.15: The Forecasting Performance of Statistical Methods for Original Monthly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
sNAIVE	15.581	1.241	0.690	0.007
SARIMA	13.221	0.912	0.528	0.222
ETS	12.851	0.900	0.517	0.153
TBATS	12.758	0.890	0.512	1.026
SGT	**12.132**	**0.872**	**0.496**	65.937
STL	15.586	0.912	0.533	**0.004**

As shown in Table 4.15, Bayesian exponential smoothing model called SGT has the best eighteen steps ahead forecasts performance shows according to all accuracy measures when compared to other statistical models. The second successful model in forecasting of the monthly series is TBATS in terms of all accuracy measures. On the other hand, the worst eighteen steps ahead forecasts performance is shown by seasonal naive model on the basis of all accuracy measures. STL decomposition can be also considered as one of the statistical model producing inadequate forecast values

based on all accuracy values.

According to the computational time producing the forecasts for 480 monthly series, SGT has the longest time period in hours since it uses MCMC algorithm to estimate the model parameters. Seasonal naive is the model that produces the forecasts in shorter time period when compared to other statistical models.

Machine Learning Models: In this part, seven machine learning models which are SVM, RF, XGBoost, feed-forward neural network, RNN, LSTM and BNN are fit to produce eighteen steps ahead forecasts for the monthly series. The past observations until the second seasonal lag which is twenty-four are included to the model as input data in the training process of the ML models. Since the general structure of the models are expressed before, only specific details about RNN and LSTM will be given in this part. RNN uses learning rate 0.05, 1956 epoch and 30 hidden layers, while LSTM uses the same learning rate with 1398 epoch and 28 hidden layers to train the network structure. The eighteen steps ahead forecasts performance of the machine learning methods are summarized in Table 4.16

Table 4.16: The Forecasting Performance of Machine Learning Methods for Original Monthly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SVM	13.374	1.526	0.829	0.554
RF	11.398	1.227	0.671	0.356
XGBoost	15.322	1.677	0.915	0.113
NNETAR	17.983	2.359	1.269	0.265
RNN	**9.208**	**0.845**	**0.468**	231.360
LSTM	10.968	1.037	0.573	73.600
BNN	10.987	0.909	0.509	**0.018**

Table 4.16 shows that the RNN clearly outperforms the other ML models in terms of all accuracy measures. LSTM performs the second best performance in reference to sMAPE, but BNN takes the place of LSTM according to MASE and the arithmetic

average of sMAPE and MASE. On the other hand, feed-forward neural network and XGBoost models can be reported as the models having the worst forecasting performance compared to the other methods in terms of all accuracy measures, respectively.

In accordance with computational time, RNN requires almost ten days to produce eighteen steps ahead forecast values for monthly series, although it produces the most accurate forecast values compared to other models. The shortest time period to produce the forecast values is required by BNN. In conclusion, RNN is considered as the machine learning model producing the most accurate forecast values and demanding the longest time. On the other hand, BNN can be evaluated as the model that produces very accurate forecast values in a very short time period compared to other models in the study. Due to performing a considerable forecasting performance in very short time, BNN is preffered as the nonlinear component of hybrid approaches instead of RNN.

Hybrid Models: In this part, all statistical models except seasonal naive model are combined with BNN as defined in the previous section. The performance of eighteen steps ahead forecasts performance of the hybrid methods are given in Table 4.17.

Table 4.17: The Forecasting Performance of Hybrid Methods for Original Monthly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SARIMA-BNN	13.988	0.967	0.553	0.038
ETS-BNN	**12.989**	0.907	0.518	0.139
TBATS-BNN	13.205	0.918	0.525	0.474
SGT-BNN	14.500	0.999	0.572	61.800
STL-BNN	13.067	**0.872**	**0.501**	**0.025**

Table 4.17 shows that performance of the hybrid approaches used in forecasting of the monthly series are very close to each other and hence the name of the best model varies based on the type of the accuracy measures. According to sMAPE, a hybrid model ETS and BNN shows the best forecasting performance. However, a hybrid

model STL decomposition and BNN can be considered the model producing the most accurate forecast values for the monthly series based on MASE and the arithmetic average of sMAPE and MASE compared to other hybrid approaches. On the other hand, the worst forecast results among the hybrid models is performed by the model uses Bayesian exponential smoothing called SGT and BNN. In addition to the hybrid model SGT and BNN, the hybrid model SARIMA and BNN performs insufficient forecasting performance compared to other hybrid models in the study.

In the comparison of the models in terms of their computational requirements, SGT-BNN demands more computational time in hours in comparison with other models in addition to its worst performance. The hybrid model demanding the shortest time period in the forecasting of monthly series is the hybrid model STL decomposition and BNN which is the best hybrid approach according to two measures.

In conclusion, a hybrid model using STL and BNN shows the best performance among the all hybrid approaches applied on 480 monthly series according to accuracy measures and computational time. However, the model cannot be concluded as the best forecasting model for original monthly series.

Result: In the analysis of monthly series without transformation, it said that the best forecasting performance is shown by RNN model that uses 0.05 learning rate, 1956 epoch and 30 hidden layers. However, the model uses the longest time for the forecasting compared to all models used for the monthly series. The hybrid model STL decomposition and BNN can be considered as the second model giving the best forecasting accuracy in the study and the model demands one of the shortest time period to produce the future values of the monthly series. On the other hand, feed-forward neural network and XGBoost have the worst eighteen steps ahead forecasting performances compared to other models including the seasonal navie which is the simplest model in the study. As seen in the previous analysis, the hybrid approaches outperform the both statistical and machine learning models on the average. The best hybrid model using STL and BNN shows the very close forecasting performance to the best model.

In addition to this, the accuracy performance of the models are compared visually.

The following Figure 4.16, Figure 4.17 and Figure 4.18 represent box plot of the each measure drawn by using the error values obtained from each model for the each monthly series in the analysis.

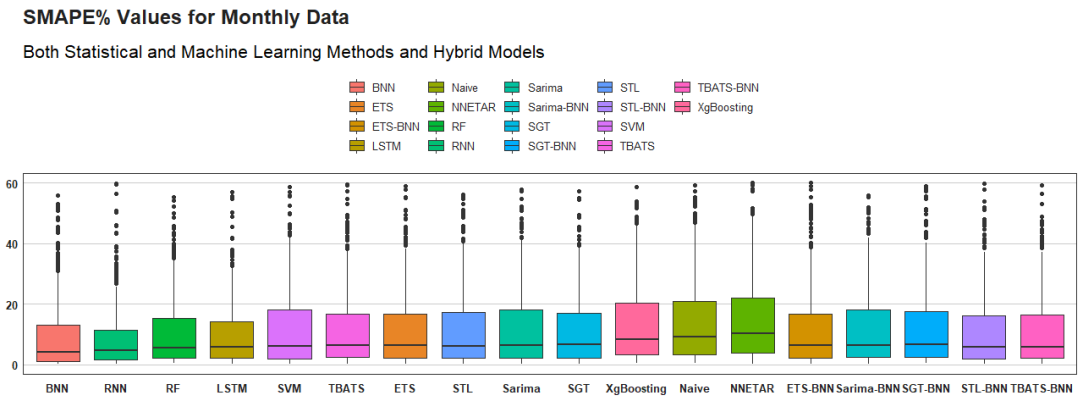

Figure 4.16: sMAPE Performances of All Models for Original Monthly Series

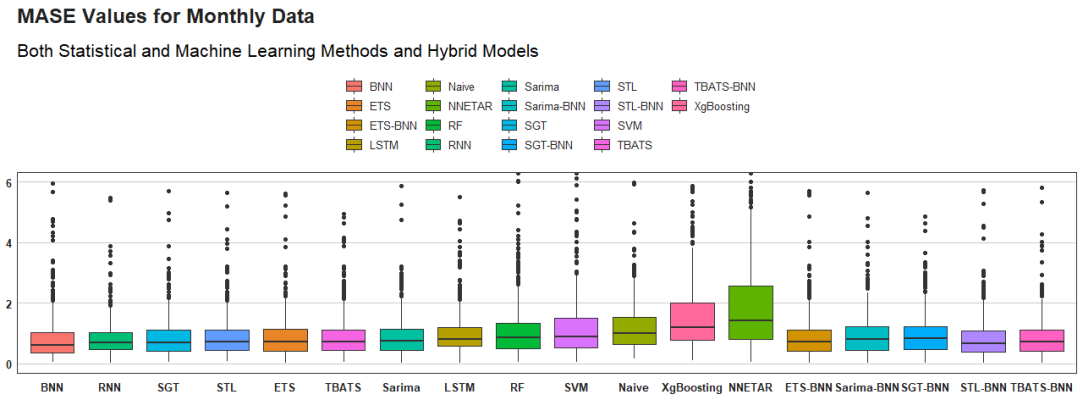

Figure 4.17: MASE Performances of All Models for Original Monthly Series

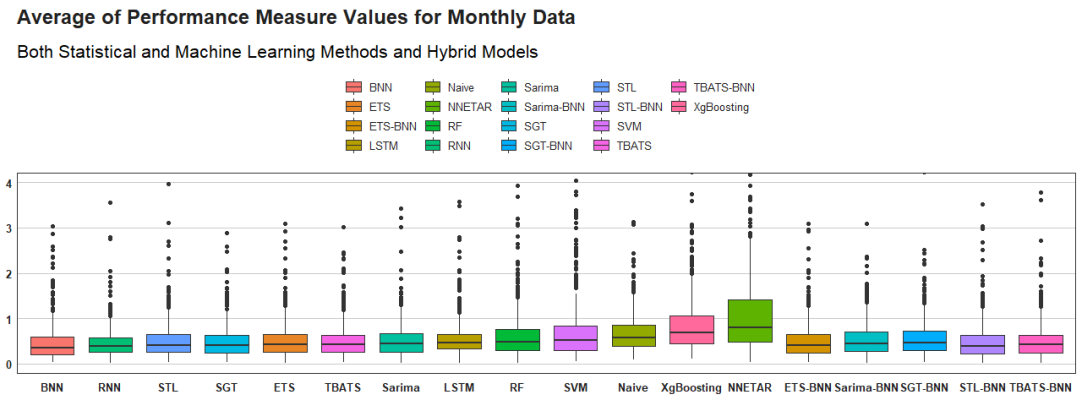

Figure 4.18: The Average Performances of All Models for Original Monthly Series

Figures 4.16, 4.17 and 4.18 represent feed-forward and XGBoost which are the worst forecasting models have more variability and higher error values on the average. On the other hand, the less variability and lower error values are performed by models including best ML models and hybrid models. Lastly, it can be said that all models produces error values that can be investigated as outlier observations that violates the model performances.

4.4.3.2 Transformed Monthly Series Analysis

In this section, the eighteen steps ahead forecasting performance of the models on the transformed monthly series will be presented with their corresponding accuracy and performance measure values. Besides, the details about the models will be given. The application of the models on the transformed series are same as the original ones. The transformation process of the series is explained in Section 4.4.

Statistical Models: Six statistical methods which are sNAIVE, SARIMA, ETS, TBATS SGT and STL are fitted to produce eighteen steps ahead forecasts for monthly series. The forecasts performances of the statistical models on the transformed monthly series is summarized in Table 4.18.

Table 4.18: The Forecasting Performance of Statistical Methods for Transformed Monthly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
sNAIVE	15.558	1.241	0.698	**0.005**
SARIMA	13.338	1.209	0.671	0.150
ETS	13.245	1.057	0.595	0.125
TBATS	**12.950**	**0.941**	**0.535**	0.471
SGT	13.716	1.749	0.943	201.130
STL	15.588	0.998	0.577	0.011

As summarized in Table 4.18, TBATS is identified as the statistical model that shows the best eighteen steps ahead forecasting performance compared to other models. The model showing the second best forecasting performance changes based on the accuracy measures. According to sMAPE, exponential smoothing model abbreviated as ETS takes the second rank in the success rating of the models, while STL decomposion is placed on the second rank according to MASE and the average of sMAPE and MASE. On the other hand, the worst eighteen steps ahead forecast performance is shown by Bayesian exponential smoothing model called SGT in terms of all measures. After this model, seasonal naive model reports insufficient forecast values for transformed monthly series.

In the comparison of the models with respect to their computational time, SGT showing the worst forecasting performance among statistical models demands the longest time period. The shortest time period for forecasting of the monthly series is demanded by seasonal naive model which can be described as the simplest forecasting model in the study.

Machine Learning Models: In this part, seven machine learning models which are SVM, RF, XGBoost, Feed-forward neural network, RNN, LSTM and BNN are fit to produce eight steps ahead forecasts using the transformed quarterly series. The parameters of RNN and LSTM are same as the ones applied on the original monthly series. The eighteen steps ahead forecasting performance of the ML models are rep-

resented in Table 4.19.

Table 4.19: The Forecasting Performance of Machine Learning Methods for Transformed Monthly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SVM	13.397	1.000	0.569	0.473
RF	11.474	0.861	0.488	0.089
XGBoost	16.537	1.497	0.831	0.155
NNETAR	17.967	1.449	0.815	0.227
RNN	**9.111**	0.682	0.386	231.549
LSTM	11.000	0.829	0.469	499.920
BNN	10.786	**0.647**	**0.378**	**0.017**

Table 4.19 shows that the RNN clearly has the best forecasting performance compared to other ML models in terms of sMAPE. However, both MASE and the average of measures indicates that BNN is the superior forecasting model among all ML models for transformed monthly series. On the other hand, the worst performance of the eighteen steps ahead forecasts belongs to feed-forward neural network. XGBoost model is also considered as ML models producing inaccurate forecast values.

According to the computational time, LSTM needs the longest time period in hours to produce the forecast values for the monthly series with Box-Cox transformation. On the other hand BNN being the superior model based on two criteria produces the forecast values for monthly series in the shortest time period.

Therefore, BNN produces the most accurate forecast values according to two criteria and needs the shortest time period for forecasting compared to other models. Hence it can be considered as the best ML model for transformed monthly series forecasting. Due to having a considerable forecasting performance in very short time, BNN is preffered as the nonlinear component of hybrid approaches for the forecasting of transformed monthly series.

Hybrid Models: In this part, all statistical models except seasonal naive model are combined with BNN as defined in the previous section to make eighteen steps ahead forecasts for the monthly series. The performance of the hybrid models for transformed monthly are given in Table 4.20.

Table 4.20: The Forecasting Performance of Hybrid Methods for Transformed Monthly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SARIMA-BNN	14.319	1.169	0.656	0.039
ETS-BNN	13.553	2.338	1.237	0.167
TBATS-BNN	**13.014**	**0.927**	**0.529**	0.904
SGT-BNN	14.806	1.743	0.946	203.028
STL-BNN	13.029	0.951	0.541	**0.027**

As displayed in Table 4.20, the performance of the hybrid approaches used in forecasting of the monthly series with Box-Cox transformation are very close to each other. However, a hybrid model TBATS and BNN shows the best forecasting performance compared to other hybrid models used for the monthly series. Also, a hybrid model STL decomposition and BNN has the second best forecasting performance based on all accuracy measures among all hybrid models used for the monthly series. On the other hand, the model name showing the worst forecasting performances changes based on the accuracy measure. According to sMAPE, a hybrid model constructed with SGT and BNN has the worst performance, but a hybrid model with ETS and BNN shows the poorest forecasting performance regarding to MASE and average.

In the comparison of the model in terms of their computational requirements, SGT-BNN demands more computational time in hours in comparison with other models in addition to showing one of the worst performance. The hybrid model demanding the shortest time period in the forecasting of monthly series is STL-BNN which is the second best hybrid approach according to accuacy measures.

In conclusion, a hybrid model using TBATS and BNN shows the best performance among the all hybrid approaches applied on 480 transformed monthly series accord-

ing to accuracy measures. However, the model cannot be concluded as the best forecasting model for transformed monthly series.

Result: In the analysis of monthly series with transformation, it said that the best forecasting performance is shown by BNN demanding the shortest time period to produce the forecasts compared to most of the models. RNN also gives the second highest forecasting accuracy, but the model needs one of the longest time period to make a forecast. On the other hand, feed-forward neural network and XGBoost have the worst eighteen steps ahead forecasting performances compared to other models including the seasonal naive which is the simplest model in the study. As opposed to previous studies, the hybrid approach doest not outperform the both statistical and machine learning models on the average. The ML methods can be stated as the most accurate forecasting model for transformed monthly series on the average compared to both statistical and hybrid models.

In addition to this, the accuracy performance of the models are compared visually. The following Figure 4.19, Figure 4.20 and Figure 4.21 represent box plot of the each measure drawn by using the error values obtained from each model for the each monthly series in the analysis.

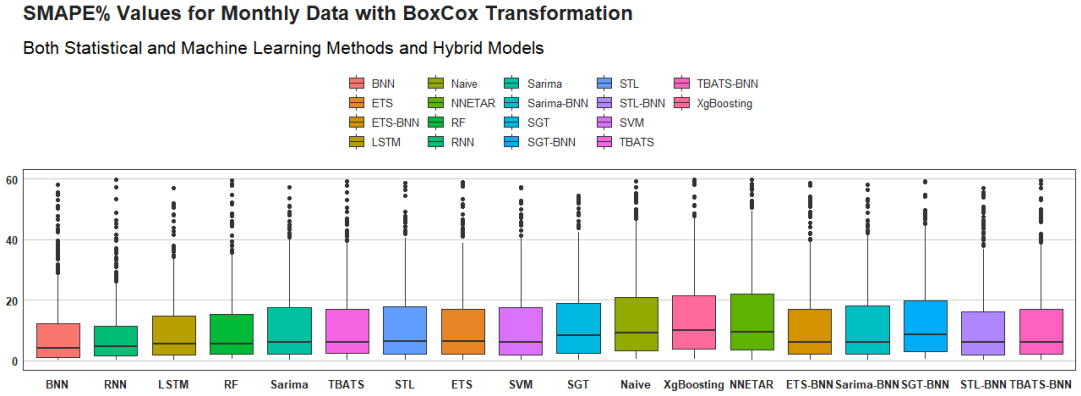

Figure 4.19: sMAPE Performances of All Models for Transformed Monthly Series

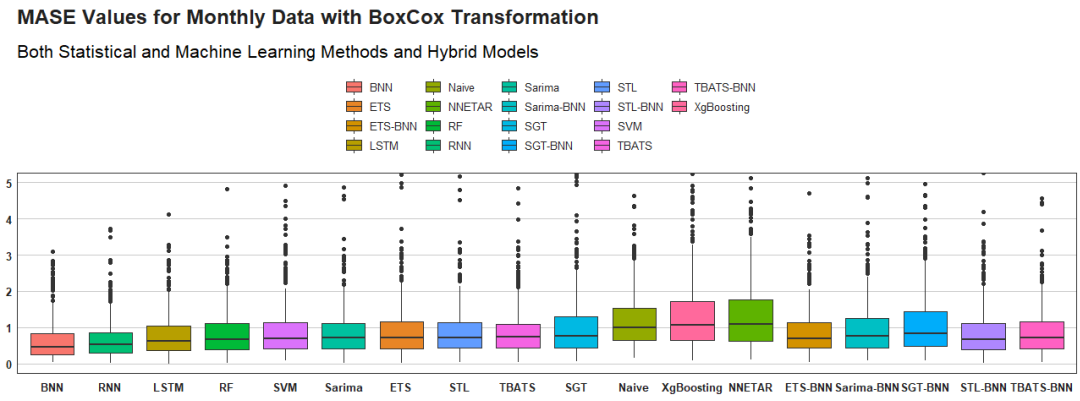

Figure 4.20: MASE Performances of All Models for Transformed Monthly Series

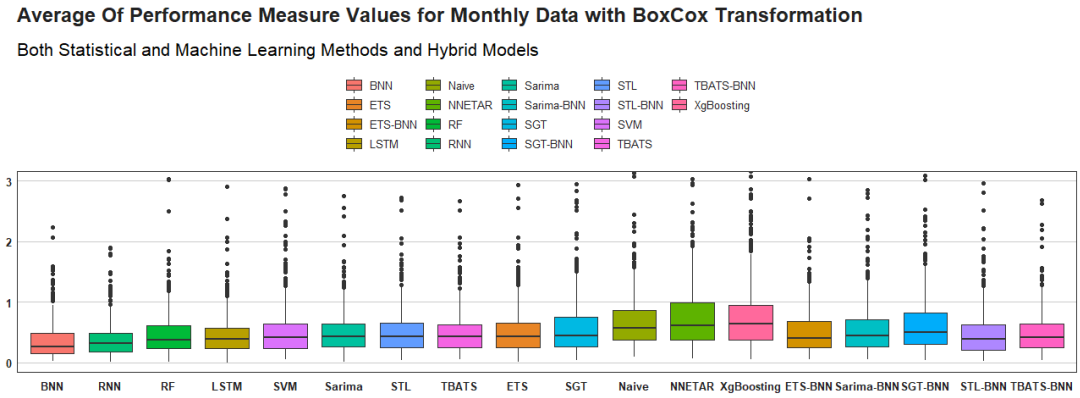

Figure 4.21: The Average Performances of All Models for Transformed Monthly Series

Figures 4.19, 4.20 and 4.21 represent feed-forward and XGBoost which are the worst forecasting models have more variability and higher error values on the average. On the other hand, the less variability and lower error values are performed by models including best ML models and hybrid models. Moreover, it can be said that all models produces error values that can be investigated as outlier observations that violates the model performances.

Lastly, it cannot be observed a remarkable effect of Box-Cox transformation on the performance of the all methods.

4.4.4 Weekly Series Analysis

In this study, 4 weekly series selected from M4 competition data set are used and the forecast performance of the models included in the study are compared using these selected series. The time interval between successive observations known as time frequency is considered as 52 for weekly series and 13 steps ahead forecast values are produced for all of the 4 series. In Figure 4.22, the time series plots of the four weekly series used in the study are represented.

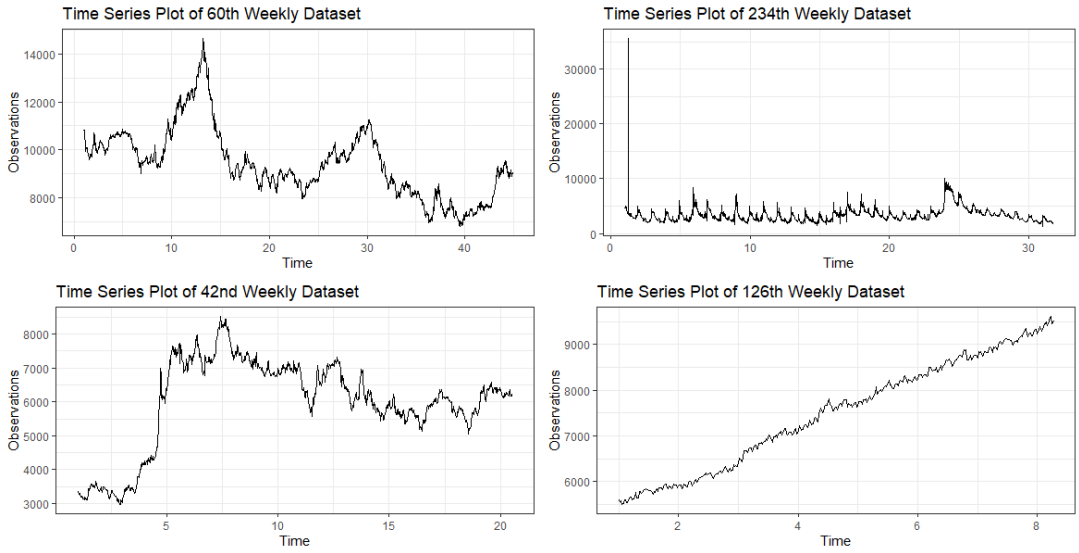

Figure 4.22: The Time Series Plots of Subset of Weekly Series

As seen in Figure 4.22, all series can be interpreted as nonstationary. Also, all series show the decreasing pattern with seasonal cycles except for the last one. Lastly, it can be said that there is a time shift in the plot of the second series.

The length of the four weekly series analyzed in this study are 379, 1017, 1602, 2284. Therefore, the minimum and maximum lengths of the weekly series used in the study are 379 and 2284, respectively.

The analysis of the weekly series are divided into two groups, original series and transformed series. For both cases, the time frequency is used as 52, and 13 steps ahead forecast values are produced as stated above.

4.4.4.1 Original Weekly Series Analysis

In this section, 13 steps ahead forecasting performance of the models on the weekly original series are presented with the corresponding accuracy and performance measures. Besides, some specific details about the models are given.

Statistical Models: Six statistical methods which are sNAIVE, SARIMA, ETS, TBATS, SGT and STL are fitted to produce six steps ahead forecasts for the quarterly series. The forecasting performance of the statistical methods for the original weekly series are represented in Table 4.21

Table 4.21: The Forecasting Performance of Statistical Methods for Original Weekly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
sNAIVE	6.684	0.593	0.330	**0.001**
SARIMA	**3.534**	**0.285**	**0.160**	0.481
ETS	8.418	0.441	0.263	**0.001**
TBATS	4.232	0.302	0.172	0.011
SGT	4.181	0.323	0.182	3.987
STL	6.827	0.514	0.291	**0.001**

As summarized in Table 4.21, SARIMA has the best thirteen steps ahead forecasts performance shows according to all accuracy measures when compared to other statistical models. The second best model in forecasting of the weekly series changes based on the accuracy measures. According to sMAPE, Bayesian exponential smoothing model called SGT is the second best statistical model for the forecasting of the weekly series. However, TBATS can be considered as the second best statistical model for the forecast of weekly series with respect to both MAPE and the arithmetic average of both sMAPE and MASE. On the other hand, the worst thirteen steps ahead forecasts performance is shown by seasonal naive model on the basis of all accuracy measures. After this model, ETS gives inaccurate forecast values for original weekly series since the function of the model cannot deal with the seasonality which is greater

than 24.

According to the computational time, SGT, which is the second best model in terms sMAPE, has the longest time period in hours since it uses MCMC algorithm to estimate the model parameters. Seasonal naive model proven as the worst statistical model demands the shortest time period to produce the forecasts when compared to other statistical models.

Machine Learning Models: In this part, seven machine learning models which are SVM, RF, XGBoost, Feed-forward neural network, RNN, LSTM and BNN are fit to produce thirteen steps ahead forecasts for the weekly series. The past observations until the second seasonal lag which is a hundered and four are included to the model as input data in the training process of the ML models. Since the general structure of the models are expressed before, only specific details about RNN and LSTM will be given in this part. RNN uses learning rate 0.01, 2998 epoch and 18 hidden layers, while LSTM uses learning rate 0.05, 1680 epoch and 13 hidden layers to train the network structure. The thirteen steps ahead forecasts performance of the machine learning models for the weekly series are summarized in Table 4.22

Table 4.22: The Forecasting Performance of Machine Learning Methods for Original Weekly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SVM	4.069	0.259	0.150	0.160
RF	**2.198**	0.178	**0.100**	0.070
XGBoost	3.179	0.270	0.151	0.037
NNETAR	5.598	0.545	0.301	0.948
RNN	6.184	0.327	0.194	52.776
LSTM	3.506	0.229	0.132	185.292
BNN	2.56	**0.176**	0.101	**0.003**

Table 4.22 reveals that Random Forest and BNN has very close forecasting performance and outperform the other models. According to sMAPE, RF has the best performance, but BNN produces the most accurate forecast values rather than other models with respect to MASE. However, RF can be considered as the best forecasting model on the basis of the arithmetic average of sMAPE and MASE. On the other hand, the similar situation is valid for the model having the worst forecasting performance. RNN can be evaluated as the model having the poorest forecast values in terms of sMAPE, but feed-forward neural network has the worst thirteen steps ahead forecasting performance in terms of not only MASE, but also the average of sMAPE and MASE.

With reference to computational time, it can be stated that LSTM demands the longest time period to produce the forecast values for the weekly series in the study. The shortest time period to produce the forecast values is required by BNN.

Consequently, RF is considered as the machine learning model producing the most accurate forecast values for the weekly series in this study. In addition to its best performance, since it takes shorter time to make a forecasting compared to most of the ML models in the study, it is preferred to be used as nonlinear component of the hybrid approach.

Hybrid Models: In this part, all statistical models except seasonal naive model are combined with RF as defined in the previous section. The performance of thirteen steps ahead forecasts performance of the hybrid models for weekly series are given in Table 4.23.

Table 4.23: The Forecasting Performance of Hybrid Methods for Original Weekly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SARIMA-RF	**3.476**	**0.293**	**0.169**	0.682
ETS-RF	8.416	0.441	0.262	0.050
TBATS-RF	4.082	0.309	0.175	**0.041**
SGT-RF	10.305	0.549	0.326	3.313
STL-RF	7.188	0.382	0.227	0.060

Table 4.23 represents a hybrid model SARIMA and RF which are the best statistical and ML models outperforms the other hybrid model for weekly series forecasting. Additionally, a hybrid model TBATS and RF shows a forecasting performance which is close to best model in terms of all accuracy measures. On the other hand, a hybrid model SGT and RF performs the poorest forecasts among both hybrid and other models in the study because of suffering from achieving convergence of default model. Also, a hybrid model STL decomposition and RF produces inaccurate forecast values with respect to all criteria.

In the comparison of the models with respect to their computational requirements, SGT-RF demands more computational time in hours in comparison with other models in addition to its worst performance. The hybrid model demanding the shortest time period in the forecasting of weekly series is TBATS-RF proven as the second best hybrid approach for weekly series in this study.

In conclusion, a hybrid model using SARIMA and RF shows the best performance among the all hybrid approaches for weekly series according to accuracy measures. However, the model cannot be concluded as the better than best machine learning model.

Result: In the analysis of weekly series without transformation, it can be said that the best forecasting performance is shown by RF model producing the future values in the shorter time period unlike the most of the models. In addition to RF, it is made

an inference that BNN has the second best forecasting performance for weekly series. On the other hand, seasonal naive model and the hybrid approach of SGT and RF have the worst thirteen steps ahead forecasting performances compared to other models in the study. Unlike the other types of series expressed above, the best forecasting performance is displayed by ML models compared to hybrid approach and statistical models. In fact, the best hybrid approach gives lower forecasting performance than most of the ML models.

In addition to this comparison, the accuracy performance of the models are compared visually. The following Figure 4.23, Figure 4.24 and Figure 4.25 represent box plot of the each measure drawn by using the error values obtained from each model for the each weekly series in the analysis.

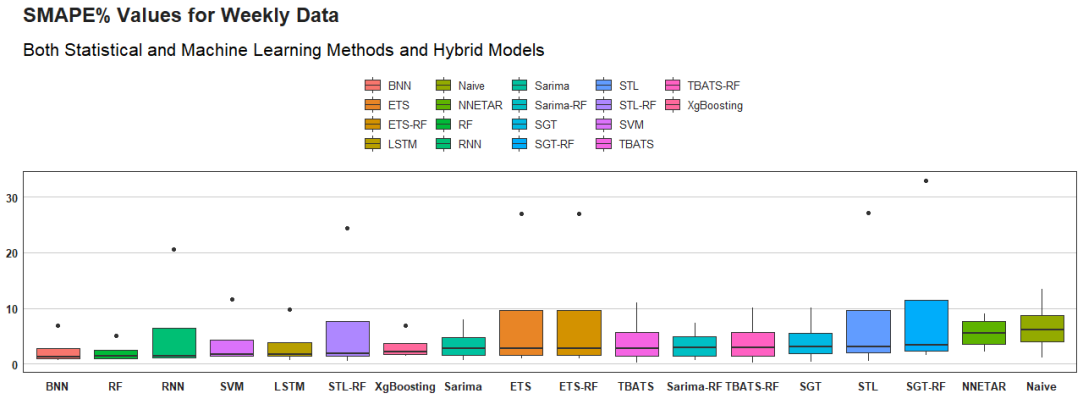

Figure 4.23: sMAPE Performances of All Models for Original Weekly Series

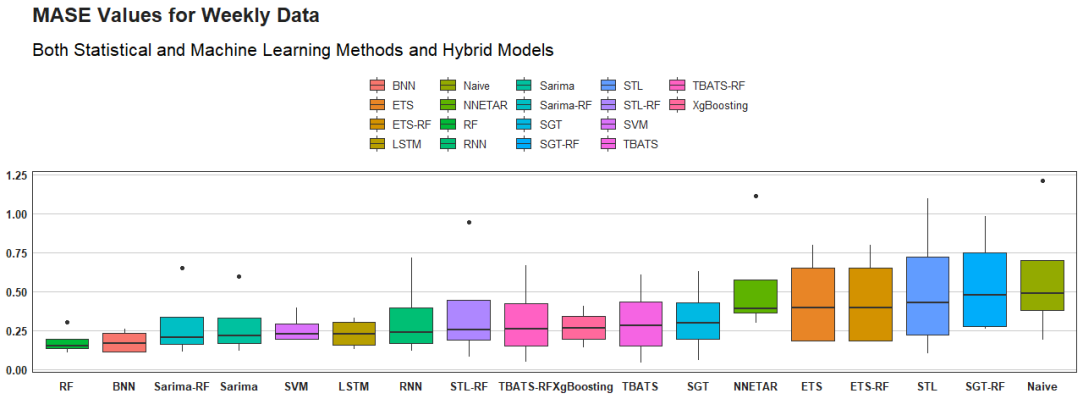

Figure 4.24: MASE Performances of All Models for Original Weekly Series

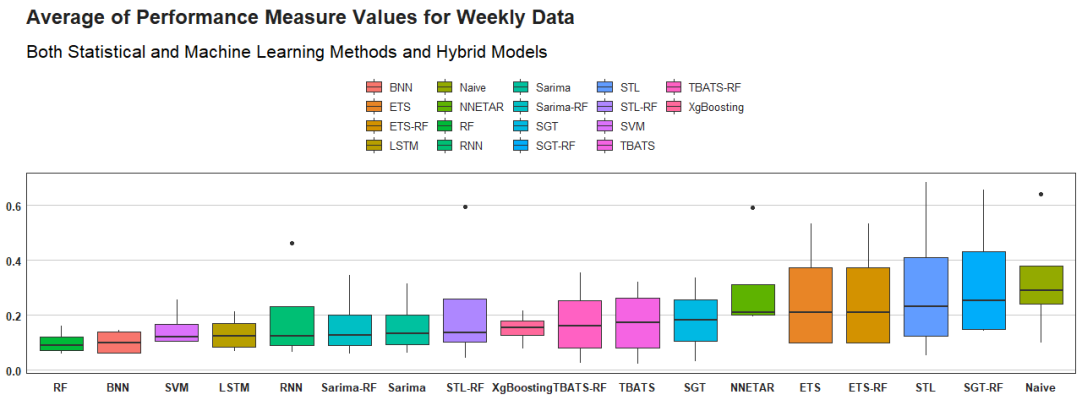

Figure 4.25: The Average Performances of All Models for Original Weekly Series

Figure 4.23, 4.24 and 4.25 display the models having the poor forecasting performance such as the hybrid SGT and RF, ETS and sNAIVE have more variability and higher error values compared to other values on the average. On the other hand, it is seen that the best forecasting models for weekly series such as RF and BNN have lower variability and error values compared to other models on the average. Lastly, it can be said that some models have error values that can be investigated as outlier observations that violates the model performances.

4.4.4.2 Transformed Weekly Series Analysis

In this section, the thirteen steps ahead forecasting performance of the models on the transformed weekly series will be presented with their corresponding accuracy and performance measure values. Besides, the details about the models will be given. The application of the models on the transformed series are same as the original ones. The transformation process of the series is explained in Section 4.4.

Statistical Models: Six statistical methods which are sNAIVE, SARIMA, ETS, TBATS SGT and STL are fitted to produce thirteen steps ahead forecasts for weekly series. The forecasts performances of the statistical models on the transformed weekly series is summarized in Table 4.24.

Table 4.24: The Forecasting Performance of Statistical Methods for Transformed Weekly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
sNAIVE	6.684	0.593	0.330	0.002
SARIMA	**3.419**	**0.282**	**0.158**	0.599
ETS	5.392	0.361	0.207	**0.001**
TBATS	4.259	0.333	0.188	0.253
SGT	4.863	0.349	0.199	16.476
STL	6.804	0.327	0.198	0.007

As summarized in Table 4.24, SARIMA has the best thirteen steps ahead forecasts performance shows according to all accuracy measures when compared to other statistical models. The second best model in forecasting of the weekly series changes based on the accuracy measures. According to sMAPE, TBATS is the second best statistical model for the forecasting of the weekly series with transformation, while STL can be considered as the second best statistical model for the forecast of weekly series with respect to MAPE. However, TBATS can also be considered as the second best forecasting model according to the arithmetic average of both sMAPE and MASE. On the other hand, the name of the model producing the poorest results depends on the type of the performance measure. The values of sMAPE show that STL decomposition has the poorest perfomance inspite of its success in MASE value, but seasonal naive model has the worst thirteen steps forecasting performance according to MASE and the arithmetic average of both sMAPE and MASE. Seasonal naive model is also the second worst model in terms of sMAPE.

According to the computational time, SGT has the longest time period in hours since it uses MCMC algorithm to estimate the model parameters. ETS considered as the one of the insufficient statistical model for weekly series demands the shortest time period to produce the forecasts when compared to other statistical models.

Machine Learning Models: In this part, seven machine learning models which are SVM, RF, XGBoost, feed-forward neural network, RNN, LSTM and BNN are fit to

produce eight steps ahead forecasts using the transformed quarterly series. The parameters of RNN and LSTM are same as the ones applied on the original weekly series. The thirteen steps ahead forecasting performance of the ML models are represented in Table 4.25.

Table 4.25: The Forecasting Performance of Machine Learning Methods for Transformed Weekly Series

Method	**sMAPE (%)**	**MASE**	**Avg.**	**Computation Time (Hour)**
SVM	4.025	0.268	0.154	0.203
RF	**2.832**	**0.200**	**0.114**	0.031
XGBoost	6.116	0.405	0.233	0.032
NNETAR	6.845	0.650	0.359	1.078
RNN	3.846	0.244	0.141	134.904
LSTM	4.615	0.264	0.155	146.904
BNN	3.660	0.213	0.125	**0.004**

Table 4.25 reveals that Random Forest clearly outperforms other ML models in the study. Also, it is seen that BNN is the another method displaying excellent forecasting performance which is close to best one according to all accuracy measures. On the other hand, it is directly said the feed-forward neural network and XGBoost models perform the worst thirteen steps ahead forecasting performance in terms of all accuracy measures.

According to the computational time, it can be stated that LSTM demands the longest time period to produce the forecast values for the weekly series in the study. The shortest time period to produce the forecast values is required by BNN which is the second best forecasting model for weekly series among the ML models.

Consequently, RF is considered as the machine learning model producing the most accurate forecast values for the weekly series in this study. In addition to its best performance, since it takes shorter time to make a forecasting compared to most of the ML models in the study, it is preferred to be used as nonlinear component of the hybrid approach.

Hybrid Models: In this part, all statistical models except seasonal naive model are combined with RF as defined in the previous section to make thirteen steps ahead forecasts for the weekly series. The performance of the hybrid models are given in Table 4.26.

Table 4.26: The Forecasting Performance of Hybrid Methods for Transformed Weekly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SARIMA-RF	4.3656	0.316	0.179	0.032
ETS-RF	4.3724	0.324	0.184	**0.023**
TBATS-RF	**4.0853**	0.329	0.185	0.044
SGT-RF	5.4183	0.408	0.231	17.874
STL-RF	4.5601	**0.295**	**0.170**	0.037

Table 4.26 shows that performance of the hybrid approaches used in forecasting of the weekly series with Box-Cox transformation are very close to each other. That's why the best model changes with respect to accuracy measures. The values of sMAPE show that a hybrid model TBATS and RF has the lowest sMAPE value which indicates that the model is the best, while MASE and the average of both measures show that a hybrid model including STL decomposition and RF outperforms the other hybrid models for weekly series in the study. However, this situation is not valid for detecting the model displaying the worst thirteen steps ahead forecast performance for the weekly series. It is obvious that a hybrid model SGT and RF is the worst hybrid approach for weekly series with transformation. In addition to this, the hybrid model TBATS and RF gives insufficient forecasting accuracy with respect to both MASE and the average of sMAPE and MASE in spite of its success in terms of sMAPE.

In the comparison of the models in terms of their computational requirements, the hybrid model SGT and RF demands more computational time in hours in comparison with other models in addition to its worst performance. The hybrid model demanding the shortest time period in the forecasting of weekly series is the hybrid model ETS and RF in this study.

Therefore, it is said that the hybrid approach of STL and RF is the best hybrid model for the forecasting of weekly series with transformation among all hybrid models considered in the study. However, the model cannot be concluded as the better than best machine learning model.

Result: In the analysis of weekly series with Box-Cox transformation, it said that the best forecasting performance is shown by RF model producing the future values in the shorter time period unlike the most of the models. In addition to RF, it is made an inference that BNN has the second best forecasting performance for the transformed weekly series. On the other hand, seasonal naive model and feed-forward neural network have the worst thirteen steps ahead forecasting performances compared to other models in the study. Like original weekly series, the best forecasting performance is displayed by ML models compared to hybrid approach and statistical models for transformed weekly series. In fact, the best hybrid approach has lower forecasting accuracy than most of the ML models.

In addition to this comparison, the accuracy performance of the models are compared visually. The following Figure 4.26, Figure 4.27 and Figure 4.28 represent box plot of the each measure drawn by using the error values obtained from each model for the each weekly series in the analysis.

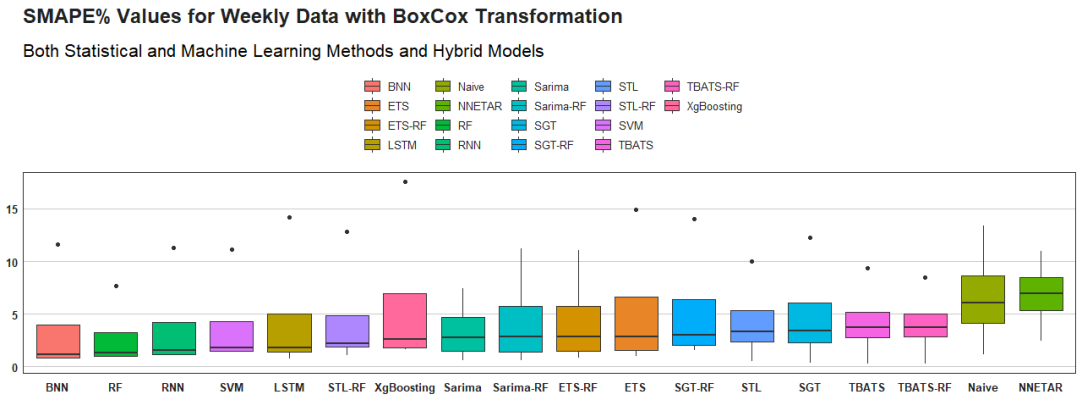

Figure 4.26: sMAPE Performances of All Models for Transformed Weekly Series

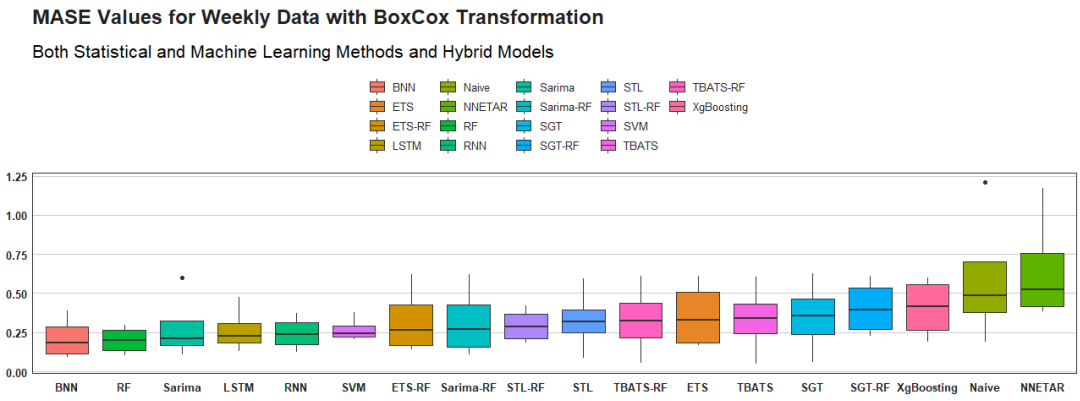

Figure 4.27: MASE Performances of All Models for Transformed Weekly Series

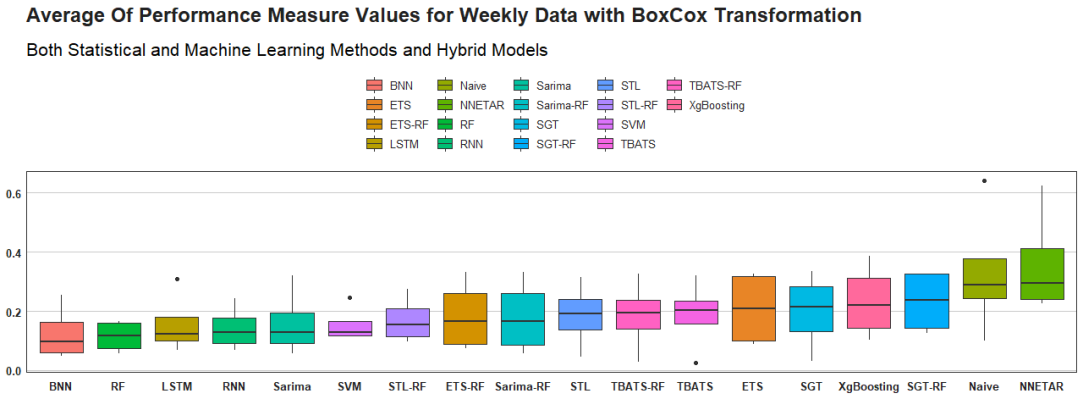

Figure 4.28: The Average Performances of All Models for Transformed Weekly Series

Figures 4.26, Figure 4.27 and Figure 4.28 display the models having the poor forecasting performance such as feed-forward neural network called NNETAR and sNAIVE have more variability and error values compared to other values on the average. On the other hand, it is seen that the best forecasting models for weekly series such as RF and BNN have lower variability and error values compared to other models on the average. Lastly, it can be said that some models have error values that can be investigated as outlier observations that violates the model performances.

In addition to this, it can be said that Box-Cox transformation results in better forecasting performance for some models such as a hybrid model SGT and RNN, but it has a negative effect on the forecasting performance of the some models such as XGBoost.

4.4.5 Daily Series Analysis

In this section, the analysis of 42 daily series randomly selected from M4 Competition will be explained. The time interval between successive observations known as time frequency is considered as 7 for daily series as determined in M4 Competition, and 14 steps ahead forecast values are produced for all series. In Figure 4.29, the time series plots of the four randomly selected daily series used in the study are represented.

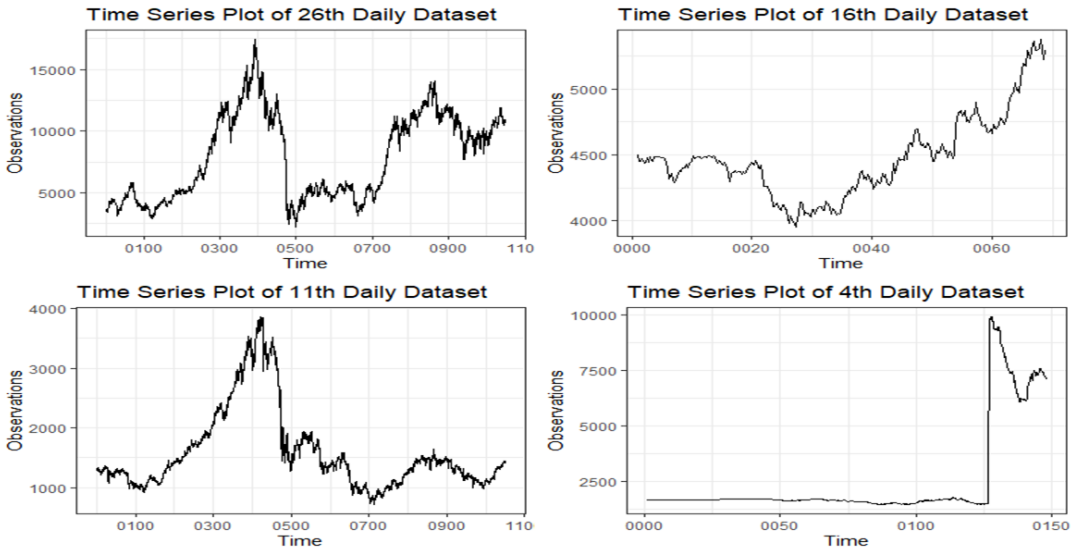

Figure 4.29: The Time Series Plots of Subset of Daily Series

As seen in Figure 4.29, all selected series have seasonal cycles. They also show both increasing and decreasing trend over time. Besides, they represent time shift in the plots.

In this study, the length of daily series vary between 175 and 4197 being the minimum and maximum lengths of the series. Besides, the maximum length of the series is also mode of length of the all series.

The analysis of the daily series are divided into two groups, original series and transformed series. For both cases, the time frequency is used as 7, and 14 steps ahead forecast values are produced as given above.

4.4.5.1 Original Daily Series Analysis

In this section, 14 steps ahead forecasting performance of the models on the daily original series are presented with the corresponding accuracy and performance measures. Besides, some specific details about the models are given.

Statistical Models: Six statistical methods which are sNAIVE, SARIMA, ETS, TBATS, SGT and STL are fitted to produce fourteen steps ahead forecasts for the daily series. The forecasting performance of the models for the original hourly series are represented in Table 4.27.

Table 4.27: The Forecasting Performance of Statistical Methods for Original Daily Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
sNAIVE	3.321	1.434	0.734	**0.003**
SARIMA	**2.605**	**1.020**	**0.523**	0.005
ETS	2.598	1.023	0.525	0.041
TBATS	2.631	1.046	0.536	0.126
SGT	2.609	1.044	0.535	36.094
STL	3.399	1.035	0.534	0.023

In Table 4.27, it can be seen that ETS model is superior to other on the basis of sMAPE, but SARIMA model produces the best forecast values according to MASE. The superior performance of SARIMA is also summarized by the arithmetic average of sMAPE and MASE. The statistical model that produces the poorest forecast also changes based on the accuracy measure. STL decompostion can be seen as the statistical model having the worst forecasting performance by sMAPE. However, seasonal naive model clearly produces the insufficient forecasts for original daily series according to MASE and the arithmetic average of sMAPE and MASE. The naive model is shown as the second worst statistical model for forecasting by sMAPE among six statistical models.

As expected because of its previous performances, the statistical model that demands

the longest time to make forecasts for daily series is SGT, while the shortest time is used by seasonal naive model having the insufficient accuracy values.

Machine Learning Models: In this part, seven machine learning models which are SVM, RF, XGBoost, Feed-forward neural network, RNN, LSTM and BNN are considered in forecasting of the original daily series. The past observations until the second seasonal lag which is fourteen are included to the model as input data in the training process of the ML models. Since the general structure of the models are expressed before, only specific details about RNN and LSTM will be given in this part. RNN uses learning rate 0.1, 1979 epoch and 21 hidden layers, while LSTM uses 0.05 learning rate with 848 epoch and 30 hidden layers to train the network structure. The forecasting performance of the machine learning models for the daily series are represented in Table 4.28

Table 4.28: The Forecasting Performance of Machine Learning Methods for Original Daily Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SVM	1.165	0.549	0.280	2.440
RF	1.248	0.632	0.322	0.172
XGBoost	2.937	1.286	0.658	0.121
NNETAR	3.148	1.439	0.735	0.099
RNN	0.930	0.435	0.222	442.632
LSTM	1.397	0.599	0.306	386.016
BNN	**0.897**	**0.403**	**0.206**	**0.007**

Table 4.28 shows that BNN is clearly the best ML model for forecasting of original daily series in terms of all accuracy measures. The second superior ML models for original daily series can be stated as RNN based on all accuracy measures. On the other hand, the worst forecast values in terms of accuracy measures is performed by feed-forward neural network written as NNETAR. After this model, XGBoost has the second insufficient forecasting performance among ML models.

Although RNN has the second superior ML models, it needs the longest time period

to predict the future values for the daily series. However, BNN produces the forecast values in the shortest time period, and has the best performance.

Therefore, BNN is used in the construction of hybrid models due to being the most successful model in terms of all accuracy measures and computational time.

Hybrid Models: In this part, all statistical models except seasonal naive model are combined with BNN as defined in the previous section in the construction of hybrid models. The forecasting performance of the hybrid models for the original daily series are summarized in Table 4.29.

Table 4.29: The Forecasting Performance of Hybrid Methods for Original Daily Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SARIMA-BNN	2.171	0.945	0.483	0.008
ETS-BNN	2.167	0.945	0.483	0.048
TBATS-BNN	2.282	1.009	0.516	0.126
SGT-BNN	2.171	0.952	0.487	18.301
STL-BNN	**1.669**	**0.786**	**0.402**	**0.017**

As represented in Table 4.29, a hybrid STL and BNN clearly outperforms the other hybrid models in forecasting of daily series in terms of all accuracy measures. Also, it can be stated that a hybrid ETS and BNN having a slightly better performance than a hybrid SARIMA and BNN is the second superior model. On the other hand, the most inaccurate forecast values among all hybrid models are performed by a hybrid TBATS and BNN. The second worst hybrid approach can be considered as a hybrid SGT and BNN in terms of all accuracy measures.

In the comparison of the model in terms of their computational requirements, the hybrid SGT and BNN demands more computational time in hours in comparison with other models as expected. The hybrid model demanding the shortest time period in the forecasting of the daily series is SARIMA and BNN.

Therefore, it is seen that a hybrid STL and BNN model is the most superior hybrid models compared to other hybrid models in addition to demanding the second shortest time for forecasting.

Result: The analysis of original daily series shows that the most accurate forecast values are produced by BNN which demands shorter time period compared to most of the models in the study. After BNN, it is stated that RNN is superior to the remaning models for forecasting of original daily series. On the other hand, sNAIVE and feed-forward neural network give the worst predicting accuracy, respectively. The XGBoost is another model having the poor forecast accuracy for daily original series. If the forecasting performances of the models are analyzed in terms of the class of the models, ML models give better predicting accuracy compared to other models on the average.

In addition to this numerical comparison, the accuracy performances of the models are compared visually. The following Figure 4.30, Figure 4.31 and Figure 4.32 represent box plot of the each measure drawn by using the error values obtained from each model for the each daily series in the analysis.

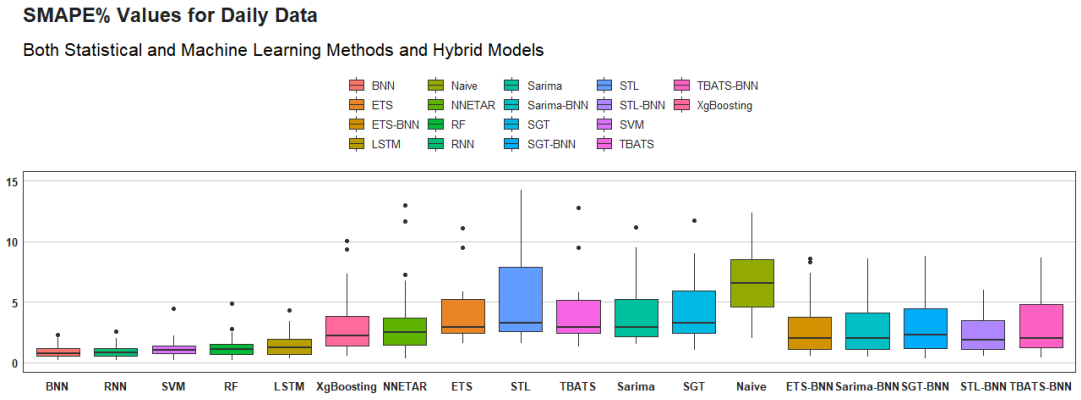

Figure 4.30: sMAPE Performances of All Models for Original Daily Series

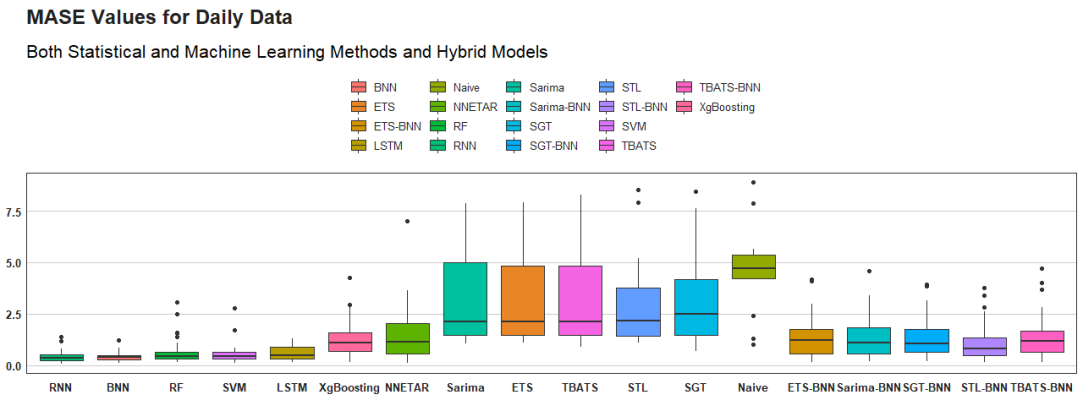

Figure 4.31: MASE Performances of All Models for Original Daily Series

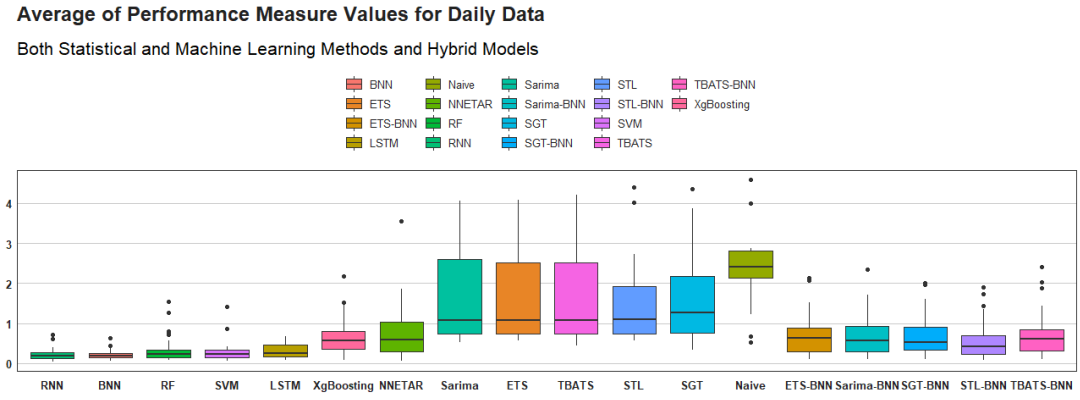

Figure 4.32: The Average Performances of All Models for Original Daily Series

As seen in Figure 4.30, Figure 4.31 and Figure 4.32, statistical models, which are the class of model having the poorest forecasting performance, have more variability than both ML and hybrid models. The figures show that naive model has the highest error value among all models. Besides, the success of ML models in forecasting of daily series can be observed from the figures above. It can be also seen that the hybrid models described as combination of statistical models with BNN being the most successful model have less variability compared to statistical models due to success of BNN in the forecasting. Lastly, it can be said that some models have error values that can be investigated as outlier observations that violates the model performances.

4.4.5.2 Transformed Daily Series Analysis

In this section, the fourteen steps ahead forecasting performance of the models on the transformed daily series will be presented with their corresponding accuracy and performance measure values. Besides, the details about the models are given. The application of the models on the transformed series are same as the original ones. The transformation process of the series is explained in Section 4.4.

Statistical Models: Six statistical methods which are sNAIVE, SARIMA, ETS, TBATS, SGT and STL are fitted to produce six steps ahead forecasts for the hourly series.

After constructing the models, the forecast values related to models are produced by `forecast` function. Then, sMAPE and MASE are calculated using forecast values and test values for each series, and then the arithmetic average of each performance measures are calculated and presented as the final results. The forecasting performance of the models for the original daily series are represented in Table 4.30.

Table 4.30: The Forecasting Performance of Statistical Methods for Transformed Daily Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
sNAIVE	2.849	1.301	0.665	0.003
SARIMA	**2.314**	1.042	**0.533**	**0.006**
ETS	3.760	1.436	0.737	0.058
TBATS	2.470	1.097	0.561	0.155
SGT	2.477	1.094	0.559	145.320
STL	2.883	**1.037**	**0.533**	0.033

According to results in Table 4.30, it is hard to suggest a single model that is clearly outperforms the other statistical model for transformed daily series forecasting in terms of all accuracy measures. The sMAPE values show SARIMA gives the best forecasting accuracy among all statistical models. However, STL decomposition is superior to other models based on MASE. Interestingly, the both of the models have

the best forecasting accuracy for transformed daily series with respect to the average of both sMAPE and MASE. On the other hand, the similar situtation is not valid for the model having the lowest forecasting accuracy values. All metrics clearly indicates that ETS gives the lowest forecasting accuracy among all statistical models. After ETS, seasonal naive model can be considered as the forecasting model having insufficient forecasting accuracy.

In the comparison of the models in terms of computational time, SGT model needs the longest time as we expected. The shortest time for the forecasting is used by seasonal naive model. Additionaly, SARIMA being most successful model in terms of sMAPE has the second shortest time period among all statistical models.

Machine Learning Models: In this part, seven machine learning models which are SVM, RF, XGBoost, Feed-forward neural network, RNN, LSTM and BNN are fit to produce forty eight steps ahead forecasts using the transformed daily series. The parameters of RNN and LSTM are same as the ones applied on the original series. The forecasting performance of the ML models are represented in Table 4.31.

Table 4.31: The Forecasting Performance of Machine Learning Methods for Transformed Daily Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SVM	1.294	0.617	0.315	0.037
RF	1.237	0.627	0.319	0.189
XGBoost	3.325	1.724	0.879	0.026
NNETAR	3.145	1.493	0.762	0.099
RNN	0.943	0.449	0.229	450.532
LSTM	1.588	0.652	0.334	388.012
BNN	**0.899**	**0.403**	**0.206**	**0.006**

Table 4.31 shows that BNN is superior to other models in terms of forecasting accuracy. Also, it can be seen RNN outperforms other ML models except BNN. On the other hand, XGBoost gives the worst forecasting accuracy for transformed daily

series among all ML models. After this model, it can be easily reported feed-forward neural network called NNETAR performed the second insufficient forecasting performance for transformed daily series.

As seen in the previous analysis, RNN demands the longest time for fourteen steps ahead forecasting. The second longest time for forecasting of transformed daily series is needed by LSTM. On the other hand, the shortest time where fourteen steps ahead forecasting performed is used by BNN being superior to all ML models in the study.

Therefore, BNN is used in building hybrid models because it is model having best forecasting accuracy and needing the shortest time period for producing forecasts.

Hybrid Models: In this part, all statistical models except seasonal naive model are combined with BNN in the construction of the hybrid models for the forecasting of Box-Cox transformed daily series. The performance of the hybrid models are reported in Table 4.32.

Table 4.32: The Forecasting Performance of Hybrid Methods for Transformed Daily Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SARIMA-BNN	2.089	0.780	0.400	**0.007**
ETS-BNN	2.180	0.802	0.412	0.061
TBATS-BNN	2.263	0.836	0.429	0.156
SGT-BNN	2.251	0.815	0.433	297.828
STL-BNN	**1.467**	**0.571**	**0.293**	0.013

As observed in Table 4.32, a hybrid STL decomposition and BNN outperforms all hybrid models for forecasting of transformed daily series. It can be also seen that the second best forecasting accuracy is achieved by a hybrid SARIMA and BNN. On the other hand, a hybrid model with TBATS and BNN shows the worst forecasting performance compared to other hybrid models for transformed daily series. After this, a hybrid model with SGT and BNN performs the second poorest forecasts among hybrid models in the study.

In the comparison of the model in terms of their computational requirements, the hybrid approach of SGT and BNN demands more computational time in hours in comparison with other models in addition to have one of the worst performances. The hybrid model demanding the shortest time period in the forecasting of daily series is STL decomposition and BNN proven as the best hybrid approach for transformed daily series in this study.

In conclusion, a hybrid model using STL decomposition and BNN shows the best performance among the all hybrid approaches for daily series according to accuracy measures.However, the model cannot be concluded as the best forecasting model for transformed yearly series.

Result: In the analysis of daily series with Box-Cox transformation, it is reported that BNN produces the best forecasting accuracy in the one of the shortest time period among all models. In addition to BNN, the accuracy measures reveal that RNN has the second best forecasting performance for the transformed daily series. On the other hand, ETS and XGBoost produce the most inaccurate forecasting values for transformed daily series. In addition to them, feed-forward neural network has considerably insufficient forecasting performance for transformed daily series. The study shows that the best forecasting performance is displayed by ML models compared to hybrid approach and statistical models. In fact, the best hybrid approach has lower forecasting accuracy than most of the ML models.

In addition to this comparison, the accuracy performance of the models are compared visually. The following Figure 4.26, Figure 4.27 and Figure 4.28 represent box plot of the each measure drawn by using the error values obtained from each model for the each daily series in the analysis.

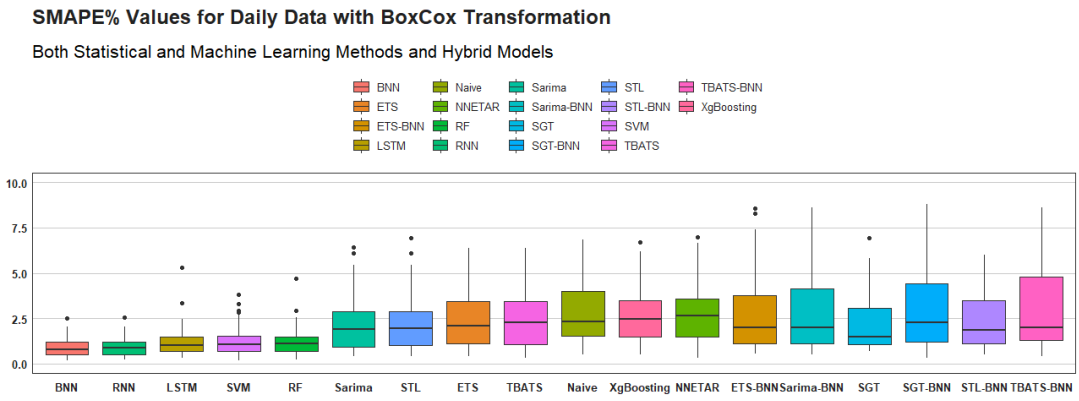

Figure 4.33: sMAPE Performances of All Models for Transformed Daily Series

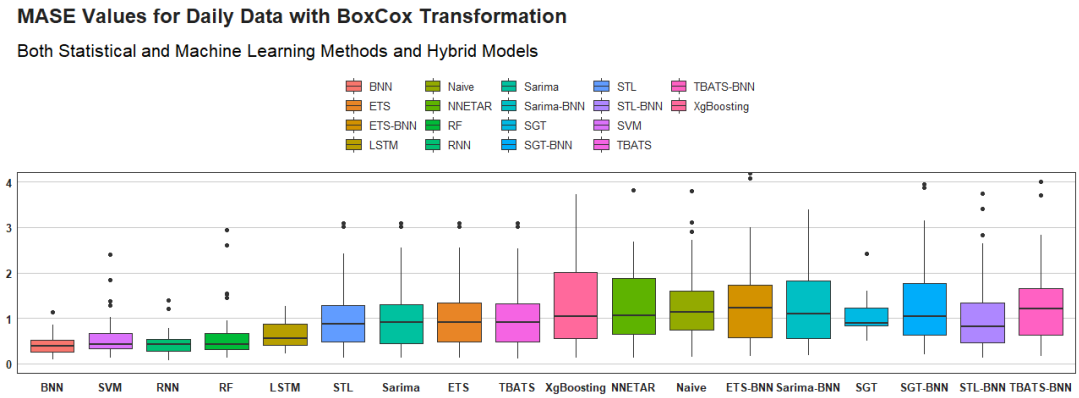

Figure 4.34: MASE Performances of All Models for Transformed Daily Series

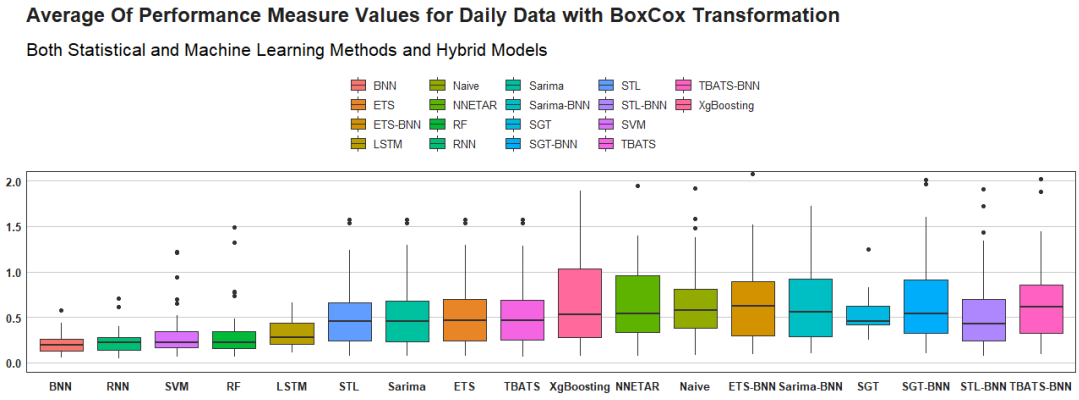

Figure 4.35: The Average Performances of All Models for Transformed Daily Series

As visualized in Figure 4.33, Figure 4.34 and Figure 4.35, the models having the poor forecasting performance such as sNAIVE, XGBoost and feed-forward neural

network called NNETAR have more variability and higher error values compared to other values on the average. On the other hand, it is seen that the best forecasting models for daily series like BNN and RNN have lower variability and error values compared to other models on the average. Lastly, it can be said that some models have error values that can be investigated as outlier observations that violates the model performances. In addition to this, it can be observed that the error values of both statistical and hybrid models decrease after Box-Cox transformation, but there is an increment in the error values for ML models after the transformation.

4.4.6 Hourly Series Analysis

In this study, 4 hourly series randomly selected from M4 competition data set are used and the forecast performance of the models included in the study are compared using these selected series. The time interval between successive observations known as time frequency is considered as 24 for hourly series and 48 steps ahead forecast values are produced for all of the 4 series. In Figure 4.36, the time series plots of the four hourly series used in the study are represented.

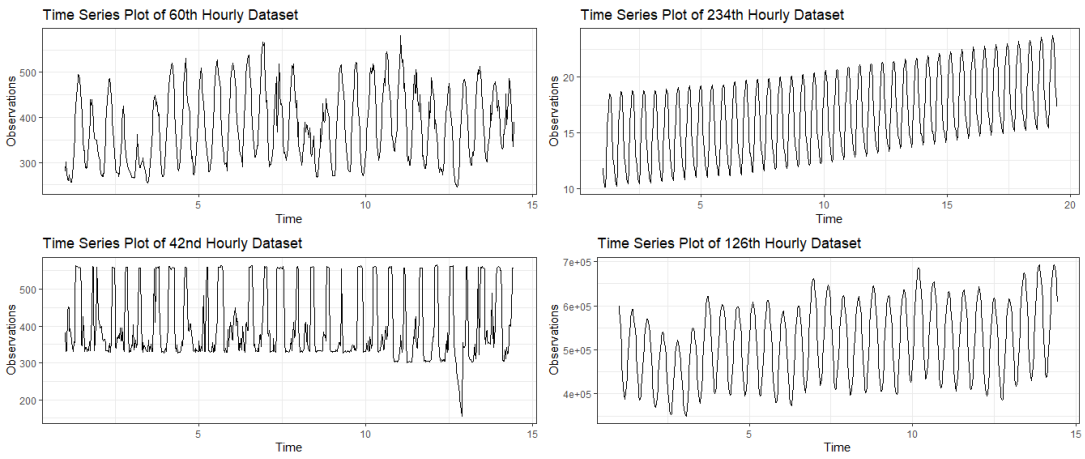

Figure 4.36: The Time Series Plots of Subset of Hourly Series

As seen in Figure 4.36, all of the hourly series have considerable seasonal cycles. Also, we have some series having increasing trend or time shift in their structure.

In this study, the length of three hourly series are 700, so they have the minimum

length in the hourly series used in the study. There is only one series having the maximum length which is 960 in the study.

The analysis of the hourly series are divided into two groups, original series and transformed series. For both cases, the time frequency is used as 24, and 48 steps ahead forecast values are produced as stated above.

4.4.6.1 Original Hourly Series Analysis

In this section, 48 steps ahead forecasting performance of the models on the hourly original series are presented with the corresponding accuracy and performance measures. Besides, the details about the models are given.

Statistical Models: Six statistical methods which are sNAIVE, SARIMA, ETS, TBATS, SGT and STL are fitted to produce six steps ahead forecasts for the hourly series. The forecasting performance of the models for the original hourly series are represented in Table 4.33.

Table 4.33: The Forecasting Performance of Statistical Methods for Original Hourly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
sNAIVE	7.639	1.219	0.648	**0.001**
SARIMA	6.097	**0.822**	**0.441**	0.092
ETS	7.146	1.111	0.591	0.006
TBATS	**5.719**	0.841	0.449	0.026
SGT	8.461	0.981	0.533	2.171
STL	8.117	1.104	0.593	0.002

As summarized in Table 4.33, the model displaying the best forty eight steps ahead forecasting performance varies based on the accuracy measure values. If the values of sMAPE is considered, it is said TBATS has the best statistical model for the forecasting of the original hourly series. However, it is seen that seasonal SARIMA model displays the best forecasting performance among all statistical models used in

the study according to both MASE and the arithmetic average of sMAPE and MASE. On the other hand, the similar situation is valid for determining the statistical model having producing the poorest forecast values. Bayesian exponential smoothing called SGT can be concluded as the worst statistical model in the forecasting of hourly series according to sMAPE. However, the other two measures show that seasonal naive model is the worst statistical model in the forecasting of the original hourly series in this study. Seasonal naive model is also the second worst statistical model according to both MASE and the average of sMAPE and MASE.

In addition to comparison in terms of accuracy measures, the models are compared with respect to their computational time as done in the previous analysis. It is clearly seen that SGT demands the longest time period in spite of displaying one of the worst performance among the statistical models. On the other hand, the shortest time period for forecasting hourly series is used by seasonal naive which can be concluded as the model with the poorest forecasting performance on the basis of MASE and the average of sMAPE and MASE.

Machine Learning Models: In this part, seven machine learning models which are SVM, RF, XGBoost, Feed-forward neural network, RNN, LSTM and BNN are fit to produce thirteen steps ahead forecasts for the hourly series. The past observations until the second seasonal lag which is forty eight are included to the model as input data in the training process of the ML models. Since the general structure of the models are expressed before, only specific details about RNN and LSTM will be given in this part. RNN uses learning rate 0.1, 942 epoch and 25 hidden layers, while LSTM uses same learning rate with 350 epoch and 14 hidden layers to train the network structure. The forty eight steps ahead forecasts performance of the machine learning models for the hourly series are summarized in Table 4.34.

Table 4.34: The Forecasting Performance of Machine Learning Methods for Original Hourly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SVM	4.710	0.761	0.404	0.041
RF	3.168	0.471	0.251	0.008
XGBoost	7.389	0.986	0.530	0.009
NNETAR	7.608	0.994	0.301	0.035
RNN	5.623	2.095	1.076	5.649
LSTM	16.295	6.663	3.413	9.949
BNN	**2.912**	**0.352**	**0.191**	**0.001**

Table 4.34 displays BNN outperforms other ML methods for the forecasting of the hourly series in this study. Also, RF can be considered as the second best ML models for hourly series forecasting on the basis of all accuracy measures. On the other hand, it is directly seen that LSTM displays the worst forecasting performance for hourly series among all ML models evaluated in the study. The main reason of this performance is raising from insufficient number of epoch used to train the network. The same reason may be valid for RNN which gives one of the lowest forecasting accuracy in spite of its perfomance in the previous analysis.

According to the computational time, it can be stated that LSTM demands the longest time period to produce the forecast values for the hourly series in the study, in addition to it has the poorest performance. The shortest time period to produce the forecast values is required by BNN which is the best ML models for forty steps ahead forecasts of the hourly series.

Therefore, BNN is considered as the machine learning model producing the most accurate forecast values for the hourly series in this study. In addition to its best performance, since it demands the shortest time to make a forecasting compared to all of the ML models in the study, it is preferred to be used as nonlinear component of the hybrid approach.

Hybrid Models: In this part, all statistical models except seasonal naive model are combined with BNN as defined in the previous section in the construction of hybrid models. The performance of forty eight steps ahead forecasts performance of the hybrid models for hourly series are given in Table 4.35.

Table 4.35: The Forecasting Performance of Hybrid Methods for Original Hourly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SARIMA-BNN	6.069	**0.803**	**0.432**	0.097
ETS-BNN	7.056	1.193	0.632	0.007
TBATS-BNN	6.085	0.967	0.514	0.024
SGT-BNN	8.355	2.866	1.475	2.048
STL-BNN	**4.609**	0.913	0.479	**0.001**

Table 4.35 represents the best hybrid approach for the forecasting of hourly series changes based on the accuracy measures. The sMAPE indicates that a hybrid model using STL decomposition and BNN has the best forty eight steps ahead forecasting performance. However, a hybrid model using seasonal SARIMA and BNN which are the best statistical and ML models outperforms the other hybrid model for hourly series forecasting according to MASE and the equally weighted average of MASE and sMAPE. On the other hand, a hybrid model including SGT and BNN performs the poorest forecasts among both hybrid and other models in the study because of suffering from achieving convergence of default model of SGT.

In the comparison of the model in terms of their computational requirements, the hybrid approach of SGT and BNN demands more computational time in hours in comparison with other models in addition to its worst performance. The hybrid model demanding the shortest time period in the forecasting of hourly series is seasonal SARIMA and BNN proven as the best hybrid approach for hourly series in this study.

In conclusion, a hybrid model using seasonal SARIMA and BNN shows the best performance among the all hybrid approaches for hourly series according to accuracy measures. However, the model cannot be concluded as the better than best machine

learning model.

Result: In the analysis of original hourly series, it can be said that the best forecasting performance is shown by BNN model demanding one of the shortest time to make a forecasting compared to the most of the models in the study. In addition to BNN, it is made an inference that RF has the second best forecasting performance for hourly series. On the other hand, LSTM and the hybrid approach of SGT and BNN have the worst forty eight steps ahead forecasting performances compared to other models in the study. Unlike the other types of series expressed above, the best forecasting performance is displayed by statistical models compared to hybrid approach and ML models on the average, although it does not contain the best model.

In addition to this comparison, the accuracy performance of the models are compared visually. The following Figure 4.37, Figure 4.38 and Figure 4.39 represent box plot of the each measure drawn by using the error values obtained from each model for the each hourly series in the analysis.

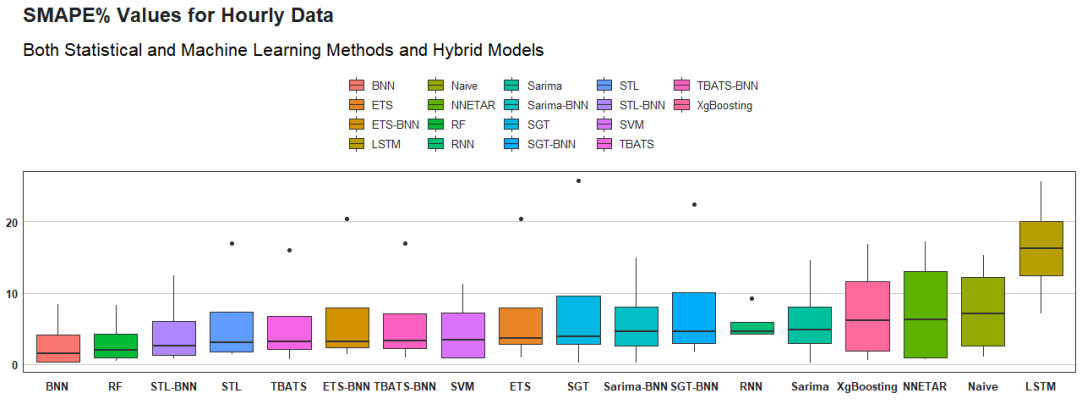

Figure 4.37: sMAPE Performances of All Models for Original Hourly Series

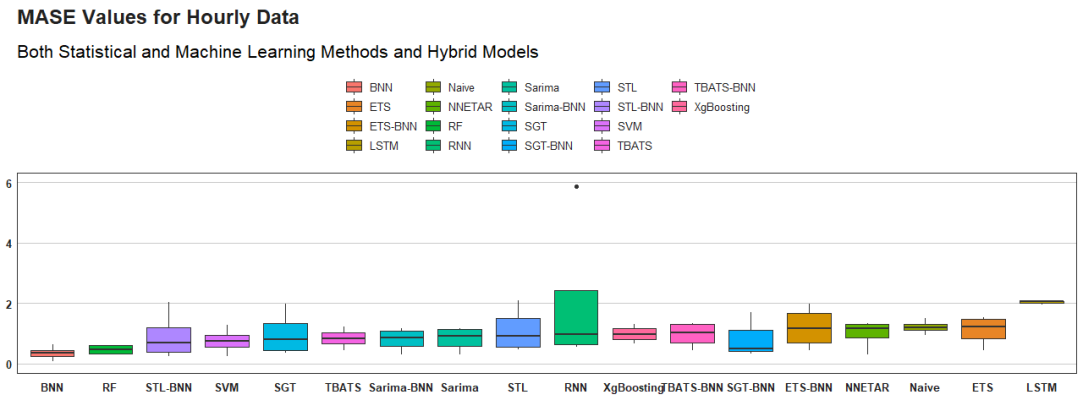

Figure 4.38: MASE Performances of All Models for Original Hourly Series

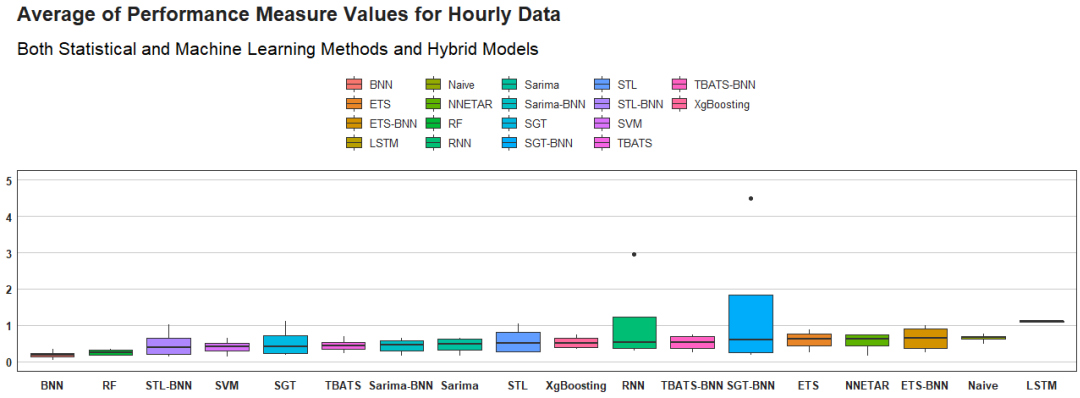

Figure 4.39: The Average Performances of All Models for Original Hourly Series

Figure 4.37, Figure 4.38 and Figure 4.39 display the models having the poor forecasting performance such as LSTM, NNETAR, SGT-BNN, XGBoost and sNAIVE have more variability and error values compared to other values on the average. On the other hand, it is seen that the best forecasting models for hourly series such as RF and BNN have lower variability and error values compared to other models on the average. Lastly, it can be said that some models have error values that can be investigated as outlier observations that violates the model performances.

4.4.6.2 Transformed Hourly Series Analysis

In this section, the forty eight steps ahead forecasting performance of the models on the transformed hourly series are presented with their corresponding accuracy and

performance measure values. Besides, the details about the models are given. The application of the models on the transformed series are same as the original ones. The transformation process of the series is explained in Section 4.4.

Statistical Models: Six statistical methods which are sNAIVE, SARIMA, ETS, TBATS SGT and STL are fitted to make a forecasting for the the hourly series. The forecasts performances of the statistical models on the transformed hourly series is displayed in Table 4.36.

Table 4.36: The Forecasting Performance of Statistical Methods for Transformed Hourly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
sNAIVE	7.639	1.219	0.648	0.002
SARIMA	5.845	0.768	0.413	0.071
ETS	9.968	1.277	0.688	0.006
TBATS	**5.598**	**0.735**	**0.395**	0.028
SGT	11.049	1.293	0.702	7.549
STL	8.384	0.985	0.535	**0.001**

As displayed in Table 4.36, the most successful statistical model for the forecasting of hourly series with Box-Cox transformation is indicated as TBATS depending on all of the accuracy measures. After TBATS, seasonal SARIMA is considered as the second best model for the forecasting of the hourly series. On the other hand, it is directly seen that Bayesian exponential smoothing known as SGT has the worst forecasting performance among all statistical models in terms of all accuracy measures. In addition to SGT, exponential smoothing model denoted by ETS is the second unsuccessful statistical model for hourly series forecasting in this study.

According to the computational time, SGT which is the worst statistical model has the longest time period in hours since it uses MCMC algorithm to estimate the model parameters. On the contrary, STL decomposition demands the shortest time period to make the forecasts for the hourly series when compared to other statistical models.

Machine Learning Models: In this part, seven machine learning models which are SVM, RF, XGBoost, Feed-forward neural network, RNN, LSTM and BNN are fit to produce forty eight steps ahead forecasts using the transformed hourly series. The parameters of RNN and LSTM are same as the ones applied on the original series. The forecasting performance of the ML models are represented in Table 4.37.

Table 4.37: The Forecasting Performance of Machine Learning Methods for Transformed Hourly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SVM	3.966	0.478	0.359	0.037
RF	3.398	0.483	0.259	0.008
XGBoost	9.342	1.408	0.751	0.006
NNETAR	7.589	0.967	0.522	0.047
RNN	5.646	2.106	1.082	7.623
LSTM	18.854	7.134	3.661	13.442
BNN	**3.402**	**0.395**	**0.215**	**0.001**

As seen in Table 4.37, the situation seen in the determination of the best statistical model for the forecasting of the transformed hourly series is seen for ML models. Random forest can be considered as the most successful ML models for forty eight steps ahead forecasts of the hourly series with Box-Cox transformation on the basis of sMAPE. However, BNN is clearly most successful ML models according to MASE and the average of sMAPE and MASE. On the other hand, it is directly said LSTM displays the worst forecasting performance for the hourly series. The second ML models producing inaccurate forecast values is XGBoost in terms of all accuracy measures.

According to the computational time, it can be stated that LSTM demands the longest time period to produce the forecast values for the hourly series in the study in addition to its worst performance. The shortest time period to produce the forecast values is required by BNN which is the best forecasting model for hourly series among the ML models. Consequently, BNN is considered as the machine learning model producing the most accurate forecast values for the transformed hourly series in this

study. In addition to its best performance, since it demands the shortest time to make a forecasting compared to all ML models in the study, it is used as nonlinear component of the hybrid approach.

Hybrid Models: In this part, all statistical models except seasonal naive model are combined with BNN in the construction of the hybrid models for the forecasting of Box-Cox transformed hourly series. The performance of the hybrid models are summarized in Table 4.38.

Table 4.38: The Forecasting Performance of Hybrid Methods for Transformed Hourly Series

Method	sMAPE (%)	MASE	Avg.	Computation Time (Hour)
SARIMA-BNN	17.716	8.155	4.166	0.002
ETS-BNN	11.217	1.266	0.689	0.006
TBATS-BNN	5.752	0.749	0.403	0.030
SGT-BNN	8.197	2.065	1.073	7.732
STL-BNN	**3.991**	**0.573**	**0.302**	**0.001**

Table 4.38 shows the best hybrid approach for the forecasting of transformed hourly series is obviously a hybrid approach using STL decomposition and BNN in terms of all accuracy measures. A hybrid model using TBATS and BNN can be considered as the second best hybrid approach for the forecasting of hourly series on the basis of all measures. On the other hand, a hybrid model with seasonal SARIMA and BNN shows the worst forecasting performance compared to other hybrid models and other models. After this, a hybrid model with SGT and BNN performs the second poorest forecasts among hybrid models in the study.

In the comparison of the model in terms of their computational requirements, the hybrid approach of SGT and BNN demands more computational time in hours in comparison with other models in addition to have one of the worst performances. The hybrid model demanding the shortest time period in the forecasting of hourly series is STL decomposition and BNN proven as the best hybrid approach for hourly

series in this study.

In conclusion, a hybrid model using STL decomposition and BNN shows the best performance among the all hybrid approaches for hourly series according to accuracy measures. However, the model cannot be concluded as the better than best machine learning model.

Result: In the analysis of hourly series with Box-Cox transformation, it said that the best forecasting performance is shown by BNN producing the most accurate future values in the shorter time period unlike the most of the models. On the other hand, a hybrid approach of seasonal SARIMA and BNN performs the worst forty eight steps ahead forecasting performances compared to other models in the study. Unlike the other types of series expressed above, the best forecasting performance is displayed by ML models compared to hybrid approach and statistical models.

In addition to this comparison, the accuracy performance of the models are compared visually. The following Figure 4.40, Figure 4.41 and Figure 4.42 represent box plot of the each measure drawn by using the error values obtained from each model for the each hourly series in the analysis.

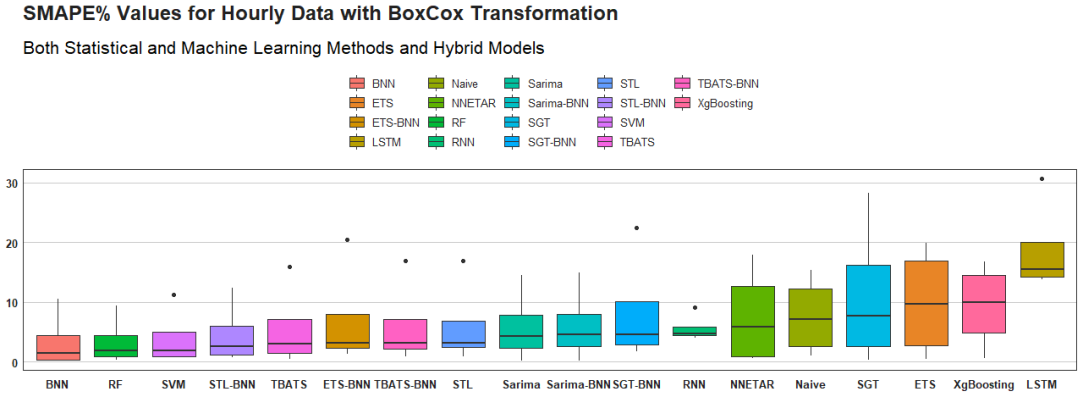

Figure 4.40: sMAPE Performances of All Models for Transformed Hourly Series

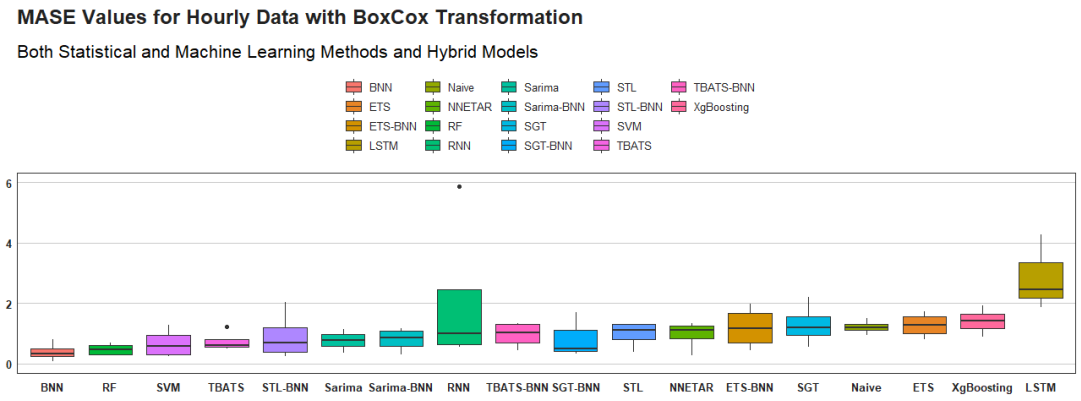

Figure 4.41: MASE Performances of All Models for Transformed Hourly Series

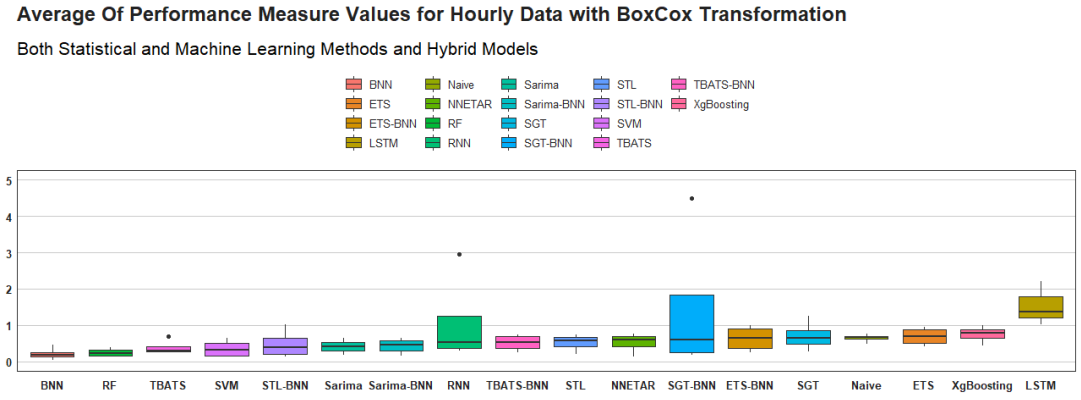

Figure 4.42: The Average Performances of All Models for Transformed Hourly Series

Figure 4.40, Figure 4.41 and Figure 4.42 display the models having the poor forecasting performance such as LSTM and the hybrid approach of seasonal SARIMA and BNN have more variability and error values compared to other values on the average. On the other hand, it is seen that the best forecasting models for hourly series such as RF and BNN have lower variability and error values compared to other models on the average. Besides, it can be said that some models have error values that can be investigated as outlier observations that violates the model performances.

Finally, it is seen that Box-Cox transformation results in increment in the error values of the most of the models including all of three methods in spite of making the error values of the some models fall.

CHAPTER 5

CONCLUSION AND FUTURE STUDIES

Forecasting is a concept that has been progressed for thousands of years due to the sense of wondering future of people. The ways of predicting future which started with divinations has evolved and earned academic respectability over time, especially after the introduction of a time series notion defining as a realization of stochastic process by Yule. After this introduction, not only time series forecasting, but also time series forecasting with high accuracy have become an active research area and developed a great number of methods aiming to achieve high forecasting accuracy such as machine learning and hybrid models in addition to statistical ones. The reason is that the accuracy of time series is a key concept for many decision process. As a consequence, there has been a large number of studies for improving the accuracy of forecasting model and suggesting new algorithms for it [2].

In this thesis, we focus on the forecasting accuracy of a wide range of forecasting models from several categories on the univariate time series having different time frequency. Besides, we try to observe the effect of Box-Cox transformation used for achieving stationarity in variance on forecast accuracy. Not only forecasting accuracy but also computational time demanding for predicting future values of the models are compared in this study.

We implement six statistical models involving naive, ARIMA, ETS, TBATS, Bayesian exponential smoothing model with trend modifitications and STL decomposition, seven machine learning models including SVM, RF, XGBoosting, feed-forward neural network, RNN, LSTM, BNN, and five hybrid models built by combination of ARIMA, ETS, TBATS and Bayesian exponential smoothing model with trend modifitications and STL decomposition models with a machine learning model having the

highest forecasting accuracy for studied time class. sMAPE, MASE and their arithmetic average are selected to compare the forecasting performances of models in the study. In addition to this, time period demanding by each model for forecasting is recorded in hours and used to compare the models. All numerical analysis in this study are done by R Studio with version 1.3.959.

In this thesis, some of similar studies are introduced at first. Then, theoretical backgrounds of the models listed above are explained in details. Finally, the models are implemented on the both original and transformed univariate time series from six time classes which are yearly, quarterly, monthly, weekly, daily and hourly. After this, the forecasting performance of the models are presented in both numerical and visual ways for each time classes, separately.

The study shows that the model having the highest forecasting accuracy varies based on series under the study. In other words, it cannot be suggested a single model being superior to other in this study. Although we cannot suggest a single model in this study, some remarkable conclusions are obtained.

First of all, it is seen that the forecasting performance of the model depends on both time frequency and forecast horizons as concluded in M Competitions. For example, Bayesian exponential smoothing model is superior to other statistical models for predicting the future values of yearly series. However, the model starts to produce worser accuracy values while time frequency is increasing. Another example that supports this conclusion is displayed by SARIMA. Although the model cannot be one of the best in quarterly or monthly series, there is an increase in the accuracy of SARIMA for series having higher time frequency.

Secondly, it is proven that some of the models whose success was announced in forecasting before such as Bayesian exponential smoothing, XGBoosting and LSTM are not as good as researchers said in forecasting. SGT and XGBoosting are proven several times as the model having one of the lowest forecasting accuracy in the previous chapter. In addition to this, feed-forward neural network proven successful in some of the previous studies shows one of the insufficient forecasting performance in the study. The main reason behind their failures is some computational problems. For example, SGT model suffers from convergence problem in some cases and hence it

produces inaccurate forecast values or LSTM may produce inaccurate forecast values for some series due to insufficient number of epoch number caused by lack of representation of selected sample for parameter tuning.

The third conclusion is about hybrid models. Although the success of hybrid approach in time series forecasting is proven by many of previous studies, the hybrid model is selected rarely as the model having the highest forecasting frequency in this study. However, it is seen that the hybrid approach makes progress in time series forecasting accuracy when compared to statistical models and some machine learning models. It also provides decreasing in the variability of the statistical models by modelling their residuals with ML models.

Furthermore, the study shows that the performance of ML model improves when the frequency of the series increases. For example, BNN gives one of the inadequate forecasting accuracies for yearly series, but the performance of the model considerably increases for the series having more time frequency.

Moreover, it is stated that spending more time forecasting does not provide more forecasting accuracy. In previous chapter, it is showed that some models which demand the longest computational time period for forecasting such as SGT, LSTM can give worser predicting accuracy. On the contrary, some models such as BNN or RF outperforms other models in spite of demanding less time period for time series forecasting.

As seen in Table 5.1 and Table 5.2, Box-Cox transformation, which makes series stationary in terms of variability, cannot reduce the error rate of the most of the models which is calculated using future values. However, it can be also said that Box-Cox transformation decreases the variability of the error measure when compared to error measures obtained from original series. Besides, it is seen that the transformation mostly does not change the name of the model whose success proven in the original series, although it extends the forecasting time of the models for most case due to requiring both transformation and back transformation process. Therefore, it can be concluded that the forecast models give better results without the transformation in the shorter time period.

Lastly, both Table 5.1 and Table 5.2 show that the model having highest forecasting

accuracy for the series under the study are mainly selected from machine learning models, and the model having the worst forecasting performance generally becomes naive model. For example, RNN has the highest forecasting accuracy for yearly series or BNN gives the best forecasting accuracy for most type of the series. As a consequence, this study can state that machine learning models especially BNN and RNN have better forecasting performance than both statistical models and hybrid models regardless of type of the time series.

For future studies, a better model identification process for ARIMA models instead of using `auto.arima` can be considered in hope to ARIMA results. Also, different models whose success in time series forecasting proven by studies such as Light Gradient Boosting, Convolutional Neural Networks can be added. Besides, hybrid approaches where ML models are combined with best statistical models can be added and hence the effect of model choice on the forecasting accuracy will be explored. In addition to real data, the forecasting performances of the series can be compared on the simulated time series. Also, the models can be implemented with parameters tuned for each series, separately and thereby some convergence or lack of training problems can be protected. Furthermore, a more efficient parameter search ways should be preferred for some models in the study such as LSTM. In addition to them, supercomputers can be used instead of normal computers to reduce the computational time for some models. Lastly, we can determine a benchmark to make a better comparison among the models.

Table 5.1: The Average of Performance Measures of Both Statistical and Machine Learning Methods for Original and Transformed Series

Method	**Original**						**Transformed**					
	Yearly	*Quarterly*	*Monthly*	*Weekly*	*Daily*	*Hourly*	*Yearly*	*Quarterly*	*Monthly*	*Weekly*	*Daily*	*Hourly*
NAIVE/sNAIVE	1.983	0.898	0.690	0.330	0.734	0.648	2.065	0.666	0.698	0.330	0.665	0.648
(S)ARIMA	1.819	0.599	0.528	0.160	0.523	0.441	2.303	0.643	0.671	0.158	0.533	0.413
ETS	1.888	0.581	0.517	0.263	0.525	0.591	2.229	0.612	0.595	0.207	0.737	0.688
TBATS	1.830	0.579	0.512	0.172	0.536	0.449	2.524	0.611	0.535	0.188	0.561	0.395
LGT/SGT	1.660	0.587	0.496	0.182	0.535	0.533	4.018	0.898	0.943	0.199	0.559	0.702
STL	-	0.596	0.533	0.291	0.534	0.593	-	0.659	0.577	0.198	0.533	0.535
SVM	1.710	0.774	0.829	0.150	0.280	0.404	1.496	0.771	0.569	0.154	0.315	0.359
RF	2.354	0.795	0.671	0.100	0.322	0.251	2.413	0.798	0.488	0.114	0.319	0.259
XGBoost	2.058	0.851	0.915	0.151	0.618	0.530	2.338	1.154	0.831	0.233	0.879	0.751
NNETAR	2.711	0.915	1.269	0.301	0.735	0.301	3.962	0.908	0.845	0.359	0.762	0.522
RNN	0.607	0.447	0.468	0.194	0.222	1.076	0.626	0.451	0.386	0.141	0.229	1.082
LSTM	0.982	0.461	0.573	0.132	0.306	3.413	0.956	0.446	0.469	0.155	0.334	3.661
BNN	1.308	0.505	0.509	0.101	0.206	0.191	1.238	0.478	0.378	0.125	0.206	0.215

Table 5.2: The Average of Performance Measures of Hybrid Methods for Original and Transformed Series

Method	**Original**						**Transformed**					
	Yearly	*Quarterly*	*Monthly*	*Weekly*	*Daily*	*Hourly*	*Yearly*	*Quarterly*	*Monthly*	*Weekly*	*Daily*	*Hourly*
(S)ARIMA-RNN	0.942	0.450	-	-	-	-	1.010	0.457	-	-	-	-
ETS-RNN	0.965	0.389	-	-	-	-	0.955	0.410	-	-	-	-
TBATS-RNN	0.934	0.406	-	-	-	-	1.026	0.553	-	-	-	-
LGT/SGT-RNN	0.741	0.377	-	-	-	-	1.065	0.840	-	-	-	-
STL-RNN	-	0.351	-	-	-	-	-	0.359	-	-	-	-
SARIMA-BNN	-	-	0.553	-	0.483	0.432	-	-	0.656	-	0.400	4.166
ETS-BNN	-	-	0.518	-	0.483	0.632	-	-	1.237	-	0.412	0.689
TBATS-BNN	-	-	0.525	-	0.516	0.514	-	-	0.529	-	0.429	0.403
SGT-BNN	-	-	0.572	-	0.487	1.475	-	-	0.946	-	0.433	1.073
STL-BNN	-	-	0.501	-	0.402	0.479	-	-	0.541	-	0.293	0.302
SARIMA-RF	-	-	-	0.169	-	-	-	-	-	0.179	-	-
ETS-RF	-	-	-	0.262	-	-	-	-	-	0.184	-	-
TBATS-RF	-	-	-	0.175	-	-	-	-	-	0.185	-	-
SGT-RF	-	-	-	0.326	-	-	-	-	-	0.231	-	-
STL-RF	-	-	-	0.227	-	-	-	-	-	0.170	-	-

REFERENCES

[1] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," *Plos One*, vol. 13, no. 3, 2018.

[2] I. M. G. Peter Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing vol.50, pp. 159-175*, 2003.

[3] L. Raphals, *Divination and prediction in early China and ancient Greece*. Cambridge University Press, 2013.

[4] G. U. Yule, "On reading a scale," *Journal of the Royal Statistical Society*, vol. 90, no. 3, p. 570, 1927.

[5] R. J. Hyndman, "A brief history of forecasting competitions," *International Journal of Forecasting*, vol. 36, no. 1, p. 714, 2020.

[6] G. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*. Holden Day, 1976.

[7] T. Fischer, C. Krauss, and A. Treichel, "Machine learning for time series forecasting - a simulation study," FAU Discussion Papers in Economics 02/2018, Friedrich-Alexander University Erlangen-Nuremberg, Institute for Economics, 2018.

[8] S. Makridakis and M. Hibon, "Accuracy of forecasting: An empirical investigation," *Journal of the Royal Statistical Society. Series A (General)*, vol. 142, 01 1979.

[9] S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. Winkler, "The accuracy of extrapolation (time series) methods: Results of a forecasting competition," *Journal of Forecasting*, vol. 1, no. 2, pp. 111–153, 1982.

[10] S. Makridakis, C. Chatfield, M. Hibon, M. Lawrence, T. Mills, K. Ord, and L. F. Simmons, "The m2-competition: A real-time judgmentally based forecasting study," *International Journal of Forecasting*, vol. 9, no. 1, p. 522, 1993.

[11] S. Makridakis and M. Hibon, "The m3-competition: results, conclusions and implications," *International Journal of Forecasting*, vol. 16, no. 4, p. 451476, 2000.

[12] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: 100,000 time series and 61 forecasting methods," *International Journal of Forecasting, 36, 54-74*, 2020.

[13] C. Heaton, N. Ponomareva, and Q. Zhang, "Forecasting models for the chinese macroeconomy: the simpler the better?," *Empirical Economics*, vol. 58, 01 2020.

[14] F. Collin and M. Kies, "Impact of near-time information for prediction on microeconomic balanced time series data using different machine learning methods," *SSRN Electronic Journal*, 2020.

[15] E. Pavlov, "Forecasting inflation in russia using neural networks," *Russian Journal of Money and Finance*, vol. 79, no. 1, p. 5773, 2020.

[16] R. S. Centeno and J. P. Marquez, "How much did the tourism industry lost? estimating earning loss of tourism in the philippines," 2020.

[17] A. Azadeh, S. Ghaderi, and S. Sohrabkhani, "Annual electricity consumption forecasting by neural network in high energy consuming industrial sectors," *Energy Conversion and Management*, vol. 49, pp. 2272–2278, 08 2008.

[18] K.-Y. Chen and C.-H. Wang, "A hybrid sarima and support vector machines in forecasting the production values of the machinery industry in taiwan," *Expert Systems with Applications*, vol. 32, pp. 254–264, 01 2007.

[19] S. Siami-Namini and A. S. Namin, "Forecasting economics and financial time series: ARIMA vs. LSTM," *CoRR*, vol. abs/1803.06386, 2018.

[20] N. Merh, V. Saxena, and K. Pardasani, "A comparison between hybrid approaches of ann and arima for indian stock trend forecasting," *Business Intelligence Journal*, vol. 3, pp. 23–44, 07 2010.

[21] L. Cao and F. E. Tay, "Financial forecasting using support vector machines," *Neural Computing Applications*, vol. 10, p. 184192, Sep 2001.

[22] M. S. Sulaiman and O. B. Shukur, "Using some classical and neural networks methods for demography predicting."

[23] J. M. V. Bravo and E. I. F. Coelho, "Forecasting small population monthly fertility and mortality data with seasonal time series methods," *As Ciências Sociais Aplicadas e a Interface com vários Saberes 2*, p. 158176, 2020.

[24] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[25] J. D. Cryer and k. Chan, *Time Series Analysis With Application in R*. Springer, 2008.

[26] C. W. J. Granger and A. Andersen, "On the invertibility of time series models," *Stochastic -Process and their Applications pp. 87-92*, 1978.

[27] B. Tas and C. Yozgatlgil, "Electricity price forecasting using hybrid time series models. master's thesis. middle east technical university," 2018.

[28] H. Liu, C. Li, Y. Shao, X. Zhang, Z. Zhai, X. Wang, X. Qi, J. Wang, Y. Hao, Q. Wu, and M. Jiao, "Forecast of the trend in incidence of acute hemorrhagic conjunctivitis in china from 20112019 using the seasonal autoregressive integrated moving average (sarima) and exponential smoothing (ets) models," *Journal of Infection and Public Health*, vol. 13, 01 2020.

[29] R. G. Brown, *Statistical forecasting for inventory control*. McGraw-Hill Book Company, 1959.

[30] C.C.Holt, "Forecasting seasonal and trends by exponentially weighted moving averages, office of naval research, research memorandum no. 52," *Reprint in International Journal of Forecasting (2004)*, vol. 1, no. 52, pp. 5–10, 1957.

[31] P. R. Winters, "Forecasting sales by exponentially weighted moving averages," *Management Science*, vol. 6, no. 3, p. 324342, 1960.

[32] E. Ostertagová and O. Ostertag, "Forecasting using simple exponential smoothing method," *Acta Electrotechnica et Informatica*, vol. 12, Jan 2012.

[33] P. S. Kalekar, "Time series forecasting using holt-winters exponential smoothing," 2004.

[34] P. Aydemir and C. Yozgatlgil, "Performance of ensemble forecasting tools for analysis turkish consumer price index. master's thesis.middle east technical university," 2018.

[35] J. W. Taylor, "Short-term electricity demand forecasting using double seasonal exponential smoothing," *Journal of the Operational Research Society*, vol. 54, no. 8, p. 799805, 2003.

[36] R. Hyndman, G. Athanasopoulos, C. Bergmeir, G. Caceres, L. Chhay, M. O'Hara-Wild, F. Petropoulos, S. Razbash, E. Wang, and F. Yasmeen, *forecast: Forecasting functions for time series and linear models*, 2019. R package version 8.9.

[37] S. Smyl, C. Bergmeir, E. Wibowo, and T. W. Ng, *Rlgt: Bayesian Exponential Smoothing Models with Trend Modifications*, 2019. R package version 0.1-3.

[38] "Introduction." `https://cran.r-project.org/web/packages/Rlgt/vignettes/GT_models.html`.

[39] A. M. D. Livera, R. J. Hyndman, and R. D. Snyder, "Forecasting time series with complex seasonal patterns using exponential smoothing," *Journal of the American Statistical Association, 106:496, 1513-1527*, 2011.

[40] M. Gosa, J. Krzyszczaka, P. Baranowskia, and I. M. Magorzata Muratb, "Combined tbats and svm model of minimum and maximum air temperatures applied to wheat yield prediction at different locations in europe," *Agricultural and Forest Meteorology*, 2020.

[41] M. Theodosiou, "Forecasting monthly and quarterly time series using stl decomposition," *International Journal of Forecasting, 27, 11781195*, 2011.

[42] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "Stl: A seasonal-trend decomposition procedure based on loess," *Journal of Official Statistics Vol. 6. No. 1, pp. 3-73*, 1990.

[43] M. Theodosiou, "Disaggregation aggregation of time series components:a hybrid forecasting approach using generalized regression neural networks and the theta method," *Neurocomputing Vol. 74. , pp. 896-905*, 2011.

[44] L. Qin, W. Li, and S. Li, "Effective passenger flow forecasting using stl and esn based on two improvement strategies," *Neurocomputing Vol. 356. , pp. 244-256*, 2019.

[45] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifier," *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, vol. 5, 08 1996.

[46] S. Xu, H. K. Chan, and T. Zhang, "Forecasting the demand of the aviation industry using hybrid time series sarima-svr approach," *Transportation Research Part E: Logistics and Transportation Review*, vol. 122, p. 169180, 2019.

[47] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, p. 199222, 2004.

[48] S. M. Basal and M. K. Okasha, "Comparison between support vector machine box-jenkins methodology for forecasting paltels stock market prices. master's thesis. al-azhar university," n.d.

[49] U. Thissen, R. V. Brakel, A. D. Weijer, W. Melssen, and L. Buydens, "Using support vector machines for time series prediction," *Chemometrics and Intelligent Laboratory Systems*, vol. 69, no. 1-2, p. 3549, 2003.

[50] M. Bouzerdoum, A. Mellit, and A. M. Pavan, "A hybrid model (sarimasvm) for short-term power forecasting of a small-scale grid-connected photovoltaic plant," *Solar Energy*, vol. 98, p. 226235, 2013.

[51] S. Rüping, "Svm kernels for time series analysis," November 2001.

[52] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch, *e1071:*

Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien, 2019. R package version 1.7-2.

[53] T. K. Ho, "The random subspace method for constructing decision forests," August 1998.

[54] Bauer, Kohavi, Breiman, T. K., Ho, Kleinberg, Freund, Bartlett, and Lee, "Random forests," *Machine Learning*, Jan 1997.

[55] W. Y. N. Naing and Z. Z. Htike, "Forecasting of monthly temperature variations using random forests," *ARPN Journal of Engineering and Applied Sciences No. 21*, vol. 10, November 2015.

[56] P.-S. Yu, T.-C. Yang, S.-Y. Chen, C.-M. Kuo, and H.-W. Tseng, "Comparison of random forests and support vector machine for real-time radar-derived rainfall forecasting," *Journal of Hydrology*, vol. 552, p. 92104, 2017.

[57] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning with applications in R*. Springer, 2017.

[58] A. Liaw and M. C. Wiener, "Classification and regression by randomforest," December 2002.

[59] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[60] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, p. 197227, 1990.

[61] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, pp. 1189–1232, 2001.

[62] M. H. D. M. Ribeiro and L. D. S. Coelho, "Ensemble approach based on bagging, boosting and stacking for short-term prediction in agribusiness time series," *Applied Soft Computing*, vol. 86, p. 105837, 2020.

[63] C. Persson, P. Bacher, T. Shiga, and H. Madsen, "Multi-site solar power forecasting using gradient boosted regression trees," *Solar Energy*, vol. 150, p. 423436, 2017.

[64] J. Nobre and R. F. Neves, "Combining principal component analysis, discrete wavelet transform and xgboost to trade in the financial markets," *Expert Systems with Applications*, vol. 125, p. 181194, 2019.

[65] Chen, Tianqi, and Carlos, "Xgboost: A scalable tree boosting system," Jun 2016.

[66] F. Climent, A. Momparler, and P. Carmona, "Anticipating bank distress in the eurozone: An extreme gradient boosting approach," *Journal of Business Research*, vol. 101, p. 885896, 2019.

[67] P. Li and J.-S. Zhang, "A new hybrid method for chinas energy supply security forecasting based on arima and xgboost," *Energies*, vol. 11, no. 7, p. 1687, 2018.

[68] W. Wang, Y. Shi, G. Lyu, and W. Deng, "Electricity consumption prediction using xgboost based on discrete wavelet transform," *DEStech Transactions on Computer Science and Engineering*, no. aiea, 2017.

[69] P. Ellis, *forecastxgb: Time Series Models and Forecasts using "xgboost"*, 2019. R package version 0.1.2.9000.

[70] A. Krenker, A. Kos, and B. Janez, *Introduction to the Artificial Neural Networks*. INTECH Open Access Publisher, 2011.

[71] R. Samsudin, A. Shabri, and P. Saad, "A comparison of time series forecasting using support vector machine and artificial neural network model," *Journal of Applied Sciences*, vol. 10, p. 950958, Jan 2010.

[72] M. Khashei and M. Bijari, "An artificial neural network (p, d, q) model for timeseries forecasting," *Expert Systems with Applications*, vol. 37, p. 479489, 2010.

[73] N. D. C. Lewis, *Deep time series forecasting with Python: an intuitive introduction to deep learning for applied time series modeling*. N.D. Lewis, 2016.

[74] S. M. H., "A brief review of feed-forward neural networks," *Communications Faculty Of Science University of Ankara*, vol. 50, no. 1, p. 1117, 2006.

[75] J. Zheng, C. Xu, Z. Zhang, and X. Li, "Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network," *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, 2017.

[76] R. K. Agrawal, F. Muchahary, and M. M. Tripathi, "Long term load forecasting with hourly predictions based on long-short-term-memory networks," *2018 IEEE Texas Power and Energy Conference (TPEC)*, 2018.

[77] B. Quast, "rnn: a recurrent neural network in r," *Working Papers*, 2016.

[78] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, p. 17351780, 1997.

[79] A. Graves, N. Jaitly, and A.-R. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013.

[80] A. G. Salman, Y. Heryadi, E. Abdurahman, and W. Suparta, "Single layer multi-layer long short-term memory (lstm) model with intermediate variables for weather forecasting," *Procedia Computer Science*, vol. 135, p. 8998, 2018.

[81] D. J. C. Mackay, "Bayesian interpolation," *Maximum Entropy and Bayesian Methods*, p. 3966, 1992.

[82] D. J. C. Mackay, "A practical bayesian framework for backpropagation networks," *Neural Computation*, vol. 4, no. 3, p. 448472, 1992.

[83] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econometric Reviews*, vol. 29, no. 5-6, p. 594621, 2010.

[84] D. Nguyen and B. Widrow, "Neural networks for self-learning control systems," *IEEE Control Systems Magazine*, vol. 10, no. 3, p. 1823, 1990.

[85] P. P. Rodriguez and D. Gianola, *brnn: Bayesian Regularization for Feed-Forward Neural Networks*, 2018. R package version 0.7.

[86] S. Makridakis, "Accuracy measures: theoretical and practical concerns," *International Journal of Forecasting, 9, 527-529*, 1993.

[87] S. Kima and H. Kimb, "A new metric of absolute percentage error for intermittent demand forecasts," *International Journal of Forecasting, 32, 669-679*, 2016.

[88] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting, 22, 679-688*, 2006.

[89] Mcompetitions, "Mcompetitions/m4-methods," Mar 2020.

[90] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:," *International Journal of Forecasting*, vol. 14, no. 1, p. 3562, 1998.

[91] S. Pan and K. Duraisamy, "On the structure of time-delay embedding in linear models of non-linear dynamical systems," 2019.

[92] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis*. Chapman and Hall/CRC, 2nd ed. ed., 2004.

[93] V. M. Guerrero, "Time-series analysis supported by power transformations," *Journal of Forecasting*, vol. 12, no. 1, p. 3748, 1993.