FORECASTING WARRANTY CLAIMS FOR MONTH IN SERVICES GROUPS IN AUTOMOTIVE SECTOR

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

BY

BEGÜM TEKÖZ

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

AUGUST 2022

Approval of the thesis:

FORECASTING WARRANTY CLAIMS FOR MONTH IN SERVICES GROUPS IN AUTOMOTIVE SECTOR

submitted by **BEGÜM TEKÖZ** in partial fulfillment of the requirements for the degree of **Master of Science** i**n Statistics, Middle East Technical University** by,

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

Prof. Dr. Özlem İlk Dağ
Head of the Department, **Statistics** ________________

Prof. Dr. Ceylan Yozgatlıgil
Supervisor, **Statistics, METU** ________________

Prof. Dr. Tuğba Taşkaya Temizel
Co-Supervisor, **Data Informatics, METU** ________________

Examining Committee Members:

Assist. Prof. Fulya Gökalp Yavuz
Department of Statistics, METU ________________

Prof. Dr. Ceylan Yozgatlıgil
Department of Statistics, METU ________________

Assist. Prof. Kamil Demirberk Ünlü
Department of Industrial Engineering, Atılım University ________________

Date: 02.09.2022

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 Last name : Begüm Teköz

Signature :

ABSTRACT

FORECASTING WARRANTY CLAIMS FOR MONTH IN SERVICES GROUPS IN AUTOMOTIVE SECTOR

Teköz, Begüm
Master of Science, Statistics
Supervisor: Prof. Dr. Ceylan Yozgatlıgil
Co-Supervisor: Prof. Dr. Tuğba Taşkaya Temizel

August 2022, 97 pages

Forecasting claim rate under warranty allows companies to optimize their production processes, reduce warranty costs and maintain customer satisfaction. In the case of a production crisis, the poor performance of the claim rate forecast negatively affects business processes. This thesis aims to improve the business processes of many departments, including production, research and development, quality, and after-sales, by forecasting the number of monthly claims in each service group. In this study, warranty data obtained from an automotive industry is used to forecast three months data for twenty-five different in-service warranty performance groups using statistical and machine learning algorithms. Specifically, statistical approaches including ARIMA, TBATS and ETS models and machine learning methods including random forest, support vector regression, XGBoosting, feed forward neural network, long short-term memory neural network, and Bayesian regularized neural network are employed. The performance of the models is compared with the Wilcoxon signed-rank test, and the results show that the best performing models are machine learning methods and the random forest model.

Keywords: Time series analysis, Machine Learning, Warranty Performance

ÖZ

OTOMOTİV SEKTÖRÜNDE HİZMET GRUPLARINDA AYLIK GARANTİ TALEPLERİNİN TAHMİNİ

Teköz, Begüm
Yüksek Lisans, İstatistik
Tez Yöneticisi: Prof. Dr. Ceylan Yozgatlıgil
Co-Supervisor: Prof. Dr. Tuğba Taşkaya Temizel

Ağustos 2022, 97 sayfa

Garanti kapsamında talep oranını tahmin etmek, şirketlerin üretim süreçlerini optimize etmelerine, garanti maliyetlerini düşürmelerine ve müşteri memnuniyetini sürdürmelerine olanak tanır. Bir üretim krizi durumunda, hasar oranı tahmininin zayıf performansı iş süreçlerini olumsuz etkiler. Bu tez, her bir hizmet grubundaki aylık hasar sayısını tahmin ederek, üretim, araştırma ve geliştirme, kalite ve satış sonrası dahil olmak üzere birçok departmanın iş süreçlerini iyileştirmeyi amaçlamaktadır. Bu çalışmada, bir otomotiv endüstrisinden elde edilen garanti verileri, istatistiksel ve makine öğrenmesi algoritmaları kullanılarak yirmi beş farklı hizmet içi garanti performans grubu için üç aylık verileri tahmin etmek için kullanılmıştır. Özellikle, otoregresif bütünleşik hareketli ortalama, TBATS ve üstel düzleştirme modellerini içeren istatistiksel yaklaşımlar ve rastgele orman, destek vektör makinesi, ekstrem gradyan arttırma, ileri beslemeli sinir ağları, uzun kısa süreli bellek ve Bayesçi yapay sinir ağını içeren makine öğrenme yöntemleri kullanılmaktadır. Modellerin performansı Wilcoxon işaretli sıra testi ile karşılaştırılmış ve sonuçlar en iyi performans gösteren modellerin makine öğrenmesi yöntemleri ve rastgele orman modeli olduğunu göstermektedir.

Anahtar Kelimeler: Zaman Serisi Analizi, Tahmin, Makine Öğrenmesi, Garanti Performansı

To my family

ACKNOWLEDGMENTS

I would like to express my gratitude and appreciation to my thesis advisor Prof. Dr. Ceylan Yozgatlıgil and co-advisor Prof. Dr. Tuğba Taşkaya Temizel for their never-ending guidance, encouragement, and patience. I feel privileged to work with them and learn from their experiences and knowledge. Without their inspiring guidance and encouraging energy, this thesis could not have been completed.

I would like to thank the examining committee members; Assist. Prof. Fulya Yavuz Gökalp, Prof. Dr. Ceylan Yozgatlıgil, Assist. Prof. Kamil Demirberk Ünlü for their suggestions and comments.

I would like to thank Ozancan Özdemir and Mutakabbir Ahmed Tayib, who helped me whenever I needed during my thesis process and did not spare their valuable support and experience. Moreover, I would like to thank my co-workers and managers who provided all kinds of support and contributed to my development during my thesis process.

I'm thankful to my lovely friends Polen Alemdar, Emre Altın, Pınar Cemre Yazıcı, İlayda Korkmaz, Buse Bakış, Rümeysa Rana Arslan for their endless support and encouragement during this thesis study.

Finally, I would like to thank my family, Elif Teköz, Mustafa Teköz, Gülay Teköz Ece, Nuray Teköz Dengiz, Hakan Dengiz, Sedat Ece, Zeynep Dengiz, Elif Dengiz and Ali Ece for their endless love and support throughout my life. I am grateful for their patience and endless love in the thesis process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	
AIC	Akaike Information Criterion
AICC	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
BIC	Bayesian Information Criteria
BRNN	Bayesian Regularized Neural Network
BP	Back Propagation
ETS	Exponential Smoothing Method
GRBF	Gaussian Radial Basis Functions
LSTM	Long Short-Term Memory
MA	Moving Average
MAPE	Mean Absolute Percentage Error
MIS	Months in Service
ML	Machine Learning
MLP	Multi-Layer Perception
MNHPP	Mixed Non-homogeneous Poisson Process

MSE	Mean Square Error
NHPP	Non-homogeneous Poisson Process
NN	Neural Network
NNETAR	Feed-Forward Neural Network
NRMSE	Normalized Root Mean Square Error
RBFN	Radial Basis Function Network
RMSE	Root Mean Square Errors
RF	Random Forests
RNN	Recurrent Neural Network
SV	Support Vector
SVM	Support Vector Machine
SVR	Support Vector Regression
TBATS	Trigonometric Seasonal, Box-Cox Transformation, ARMA residuals, Trend, Seasonality
tSVR	Twin Support Vector Regression
XGBoost	Extreme Gradient Boosting
WNHPP	Weighted Non-homogeneous Poisson Process
WMNHPP	Weighted Mixed Non-homogeneous Poisson Process
wSVR	Weighted SVR-based time series model

CHAPTER 1

INTRODUCTION

Warranty is a contract indicating that the manufacturer is responsible for resolving this situation in case the products produced fail within the scope of the warranty. The contract includes both the expected performance and the compensation available to the buyer should a failure occur (Murthy *et al.,* 2004). Automobile companies that spend a large amount of money annually to repair products that fail in line with the scope of the contract give priority to the analysis of warranty data (Rai *et al.*,2005).

A typical life cycle of defective products is as follows: Manufactured products are stored in warehouses and products are delivered to customers through distributors. Some of these product’s malfunction during the warranty period and are brought to the service. Warranty data is data collected during the repair of defective products under warranty, and there is additional data that includes production, sales, and supply information of the defective product (Wu, 2012).

Warranty data may contain different information depending on the industry, and each sector may have its own set of goals. According to the number of factors, warranty procedures coverage is categorized as one-dimensional or two-dimensional. One-dimensional warranty policies only consider a single variable, such as age (Marshall *et al.*, 2009), manufacturing characteristics of items (Kalbfleisch *et al.*, 1988), and usage (Lawless *et al.,* 1992, Hu *et al.,* 1997 and Hu *et al.,* 1998). Two-dimensional warranty policies typically take two factors such as age and usage amount or age and mileage limits. Warranty policies for high-capital products like automobiles and aircraft engines are typically two-dimensional, evaluating use and age together (Wang *et al.,* 2018). In warranty models used in some automotive industries, the coverage area is one-dimensional and only time is preferred (Yang *et al.*, 2004). The aim of both one-dimensional and two-dimensional approaches is to forecast the claim rates by using past observations.

The warranty claim prediction is used in both warranty procedures to provide manufacturers with knowledge about the product's performance and quality. Since time series have a wide application area, approaches have been developed in line with the needs. Many methods, including lifetime distributions, stochastic processes, time series analysis, and machine learning algorithms such as artificial neural networks and support vector machines, have been used to predict warranty claims in this way.

Although the follow-up of the warranty process varies according to the working principles of the companies, the focus is on the month of the warranty of a defective product. The month in services (MIS) is defined as the time the vehicle was used by the last customer. This information shows the month of service during the warranty period. Products that fail in the first month of the warranty are included in the MIS 0 service group, while the products that fail in the second month of the warranty are included in the MIS 1 service group. The warranty period in the automotive industry is generally 2 years. Accordingly, products that failed in the last month of the warranty are also in the MIS 24 service group.

The focused metrics differ, as the warranty data contains different information depending on the industry and the priorities of each sector are different. Some companies prefer to use the cumulative number of repairs in the field per 1000 products per month of service, while some companies prefer to consider the ratio of the number of requests in the month of service divided by the total number of productions during the period in which the defective product was produced. In addition, the ratio formed by dividing the number of requests in the service month by the number of products sold for the same MIS group also be applied by some companies.

When predicting the warranty data, various ratios and methods have been investigated in the literature, which will be discussed in the next section. Moreover, point estimates for the cumulative mean function of warranty claims were constructed by Lawless *et al.* (1995) based on Poisson models. Akbarov and Wu (2012) used the autoregressive mean model and Poisson methods to estimate the distribution of the claim rate, which is formed by dividing the number of claims received in the same service group by the number of products produced in the same service group. With new requirements and techniques developed over time, the prediction of total claims in warranty data has also improved. Nonparametric techniques such as neural networks were used to predict warranty claims using multilayer perceptions (MLP) and radial basis functions (Rai *et al.*, 2005) have been applied to predict the cumulative number of repairs carried out per 1000 vehicles. Moreover, forecasting the claim rate was constructed by using a support vector machine (Wu *et al.*, 2011). Details of the applied methods are shared in detail in the second part.

The global automotive industry is currently dealing with a chip shortage, which has an impact on production volume. As Wu *et al.* (2021) stated, the continuation of the chip shortage can increase the risk of breaking the industrial chain. Warranty data of a company in the automotive industry, whose name cannot be disclosed due to confidentiality, was used in this study. Due to the chip shortage, the company's production plan is constantly changing upon notification from suppliers. The company had to prefer to use different alternatives, as the use of a claim rate, which includes the number of sales or the number of productions, became difficult in business processes. Forecasting the number of claims appears to be an alternative based on the requirements of the organization and the applicability of the predicted results processes. Besides that, the business operations of the departments in the factory benefit from forecasting the total number of claims for each MIS group. For the service component, forecasting MIS groups separately aids in a more thorough analysis of failures' root causes. It is also used to oversee whether the malfunction requests accurately reflect the actions taken during the product quality development phase. In addition, the amount of budget allocated by the finance department varies according to the MIS groups to be foreseen, considering the total number of claims.

Most studies in the literature concentrate on forecasting the claim rate, but this study contributes to the literature by directly addressing the number of claims. With this contribution to the literature, it proposes methods that other companies can apply. In line with the needs of the company, the total number of claims for a total of 25 MIS groups in the service group is forecast to be 3 months consecutive. The aim of this thesis is to test whether the frequently used ML models and time series models can be used effectively in the forecasting of the number of claims. To the best of our knowledge, these methods have been applied for the first time for the company providing the warranty data, and they aim to contribute to the management of business processes within the company. In this study, three statistical models ARIMA, Exponential Smoothing, TBATS, and six different machine learning methods Support Vector Regression, Random Forest, XgBoost, Feed Forward Neural Network, Bayesian Regularized Neural Network, and Long Short-Term Memory Neural Network were used. By applying non-parametric statistical tests, the performances of the models were compared and the approach to be applied to the company was determined.

The pre-processing of the data set was chosen considering the needs of the models and is explained in detail in Chapter 4. The prediction performance error of the models is compared by considering the mean absolute percentage error (MAPE) and root means square error (RMSE) values. Except for the Long Short-Term Memory Neural Network, all other models are installed on R Studio with version 1.3.959, and Python with version 3.6. is used in the Long Short-Term Memory Neural Network model to be able to control the hyper-parameters more efficiently.

The study is divided into five main sections. In the first section, the importance of warranty data analysis and forecasting the number of claims is described. The second part covers the research in the literature about predicting the number of claims. The theoretical justification of the models used in the study and the accuracy measurement metrics utilized in the model performance comparison is presented in the third section. The data set, pre-processing techniques, applied models' parameters, and tuning techniques are all covered in Chapter 4. Additionally, the study concludes and suggestions for additional research.

CHAPTER 2

LITERATURE REVIEW

Warranty data is industry specific and comprise information regarding product quality and reliability. Forecasting the number of claims under warranty directly affects the company's production, supply, after-sales, and finance departments. The methods developed for estimating warranty claims and their performances were listed and explained below.

2.1 Stochastic Process

The Poisson process is one of the most used methods for predicting warranty claims, and it has been emphasized by researchers that it can be applied to a wide variety of environments (Veevers *et al.*, 1986 and Lawless, 1987). Stochastic approaches have been preferred by most researchers, since the claim numbers are recurrent events in the warranty data. The number of claims is predicted using the heterogeneity and random effects between incoming fault requests.

2.1.1 Non-Homogeneous Poisson Process

The Non-Homogeneous Poisson model in the stochastic process, which estimates the number of failures up to time *t*, was used by Majeske (2007) to predict automobile failures and their occurrence time using the warranty dataset (Majeske, 2007). It is similar to an ordinary Poisson process, except for the fact that the average rate of arrival can change with time. The assumption behind NHPP is that the mean and variance of the total warranty claims over any given period are equal.

Automobile manufacturers tend to attempt to predict the number of claims in the warranty dataset without taking mileage into account. When predicting the number of claims, Majeske (2007) contended that using a two-dimensional perspective would lead to better outcomes if the total number of claims and mileage were assessed. When using the NHPP approach, homogeneity was assumed, implied that all the vehicles in the population have the same density function or failure rate (Majeske, 2007). It was also allowed for the inclusion of past experiences when describing the failure process. His study revealed that warranty claims could be forecasted more precisely with the NHPP.

2.1.2 Flexible Non- Homogeneous Poisson Process

Flexible nonhomogeneous Poisson processes were proposed by Fredette *et al.* (2007) to forecast the warranty claims with random effects being used to model any potential product heterogeneity. This study incorporated the finite horizon total prediction, which allows the prediction of the population's total number of events over a given period without sacrificing generality (Fredette *et al.*, 2007). The study of Fredette *et al.* (2007) aimed to handle sizable heterogeneous populations of units, to use the age of product or time in the event process, and to provide accurate forecast intervals. In this study, the car warranty dataset from those presented by Kalbfleisch *et al.* (1991) was used. It includes 15,775 cars produced in a total of 206 days, and the total number of requests is 2620. Moreover, they estimated based on the data accumulated in the first 150 days of production. The scenarios where heterogeneity was observed between processes for different units were considered. The independent and identically distributed random variables that could not be observed were added to the model as gamma distribution every 50 days.

In the study conducted by Fredette *et al.* (2007), it was determined that even if the alpha values in the gamma distribution were not actually random, the beta components could be determined accordingly, and the reliability of the model could be ensured. In their proposed methodology, unit-level random effects were used as a gamma distribution, thus unit-to-unit heterogeneity was captured, and realistic estimation intervals were provided (Fredette *et al.*, 2007).

2.1.3 Weighted Poisson Process

Wu and Akbarov, in their 2011 study, showed that the warranty claims in recent months is more important in predicting future demands by using machine learning methods. In accordance with this study, Akbarov *et al.* (2012) predicted warranty claims and compared models with a total of six different methods including a weighted approach in Poisson processes. The implemented methods are Autoregressive Integrated Moving Average (ARIMA), Inhomogeneous Poisson Process (NHPP), Mixed NHPP (MNHPP) and Artificial Neural Network model, Weighted NHPP and Weighted MNHPP. The MHNPP model includes a useful technique for overcoming overdispersions in which the increments are not independent. ANN, on the other hand, was chosen for this study because it works well in time series models. Also, weighted maximum likelihood estimation was used for NHPP and MNHPP.

By examining the 18-month warranty data of eight different electronics industry products, forecasts were conducted with six different methods according to two different time horizons, 3 and 6 months. The dataset includes the number of items in the market at time *t* and the number of claims at month *t*. The claim rate was forecast by dividing the number of claims at the observed time *t* by the number of products in the market. The performance of the models was compared using the normalized mean squared error to measure the prediction error.

The average NRMSE value was evaluated for each utilized framework and the two time periods. According to the model performance results, it was found that the MNHPP model approach had the lowest NRMSE value. Moreover, the performance of the MNHPP model was superior to the NHPP model when weighted maximum likelihood was used. When all models were compared, it was found out that the

MNHPP approach performs better than other approaches overall. It was stated that the reason for this might be the selection of the weight function and parameters with selecting static validation data set. Through this study, Akbarov *et al.* (2012) demonstrated how adding the weighted maximum likelihood into the warranty data can bring the forecast result even closer to the actual value.

2.2 Machine Learning Algorithm

2.2.1 Multi-Layer Perceptron Neural Networks

Rai *et al.* (2005) designed a new multi-layered perception neural network to analyze the claim ratio in the warranty data of the automotive industry. In addition to forecasting future MIS values, this study also involved predicting the performance of a given month's services warranty (MIS) in a specific future. The claim rate that they predicted was the total number of claims out of every 1000 units (*R/1000*), and the average claim rate for the services group increases each month by gradually adding the subsequent months (*Rai et al.*, 2005).

Although dynamic linear models and log-log plots have been used in the literature to predict warranty data, they argued that new methodologies are required for maturing data. Rai *et al.* (2005) proposed the multiple perception approach, arguing that instead of using dynamic linear models with a radial basis, a dynamic linear model with innovation terms would be better to consider the uncertainties introduced by maturing data. There were two different types of signals flowing in the MLP networks implemented in this study: function and error signals. The function signal ran from the network's entrance to its outcome, while the error signal acted in the opposite way. The signal factor, which affects the forecasting window and the initial synaptic weight values, was the first factor in the study to have an impact on the MLP networks forecasting. The second factor was the control factor, which includes the number of neurons in each layer, the learning rate, the momentum, and the training mode. The third factor was the noise factor, which includes the design, manufacturing, assembly, service-related change, and biased warranty data.

The performances of the MLP networks were compared with the performances of the RBF networks and log-log regression models using the normalized root mean square error (NRMSE) metric. Because the network's initial value assignments were random, two different NRMSE values were calculated using the same parameters. Both the test and train datasets were chosen to contain the number of neurons and learning rate that will have the highest signal-to-noise ratio and the lowest NRMSE value (Rai *et* al, 2005). As a result of this model, claim rates were predicted correctly, but when the forecast horizon exceeded eight months, it was observed that the prediction error values increased.

2.2.2 Support Vector Machine

Wu *et al.* (2011) claimed that models that have been implemented in the literature which are log-linear Poisson models, Kalman filter, time series models, and artificial neural network, had two weaknesses. Firstly, it was stated that predicting the rates calculated by arithmetic mean processing would cause a loss of information, and secondly, the number of claims in recent months is more important in predicting future requests. To overcome all these problems, the original warranty data was taken into account, such as claim rates, and repair rates in this study, and a more flexible model structure was preferred with higher weight given to the final warranty claims (Wu *et al.*, 2011). In order to prove the veracity of these claims, the performances of numerous models were compared using two different industries' warranty datasets from automobile and electronics manufacturers. They applied five different models, radial basis function network (RBFN), MLP, SVR, tSVR, and wSVR Java programming language, and some functions from two data mining packages, Weka and LIBSVM, were borrowed.

When using the RBFN, three hyper-parameters, which were the number of Gaussian radial basis functions, the minimum standard deviation for GRBFs, and the ridge value for using the outputs of GRBFs, were considered in this study. Moreover, a single hidden layer using backpropagation was constructed to find the optimal parameters of the multilayer perception, and hyper-parameters which were learning rate, momentum, and the number of hidden nodes in the hidden layer were used. The SVR was created by optimizing the constant C, epsilon, and gamma hyper-parameters. Eventually, gamma and epsilon were utilized as hyper-parameters in weighted SVR based time series (tSVR) and weighted support vector regression (wSVR) models. Unlike SVR, adaptive *C* was used instead of the constant term *C* in the wSVR and tSVR. Within the two datasets, the data were divided into the training, validation, and test datasets, and the hyper-parameters with the lowest MSE were determined from the validation dataset. The mean square error and the weighted total square error were applied to compare the performances of the constructed models.

As a result of this study, Wu *et al.* (2011) showed that the weighted SVR-based time series model outperforms MLP, RBFN, and SVR in predicting the number of claims on warranty data from two different industries. Likewise, the weighted SVR regression model also generated superior outcomes to other methodologies. According to Wu *et al.* (2011), the main reason weighted methods had higher prediction accuracy was the fact that more weight was given to the most recent data rather than previous data.

CHAPTER 3

METHODOLOGY

In this section, forecasting methods will be introduced. Nine different methods were applied to determine which forecasting technique performs the best for the dataset. The first three of these models are well known traditional statistical methods, namely ARIMA, ETS and TBATS. Altough ARIMA and ETS were applied in the literature, TBATS was not preferred by researchers. The machine learning methods, that are chosen for forecasting, are the Support Vector Regression, Random Forest, XGBoost, Feed Forward Neural Network, Bayesian Regularized Neural Network and Long Short Term Memory Neural Network.

3.1 Autoregressive Integrated Moving Average Model (ARIMA)

Autoregressive (AR) model was introduced by Yule (1926) and Moving Average (MA) was introduced by Slutsky (1937). Box *et al.* (1970) investigated the autoregressive moving average model (ARMA) based on works of Yule (1926), Slutsky (1937) and Wold (1938). This model is the linear combination of p past time series variables and q past white noise error terms, which can be used for a large class of stationary time series and predict future values for the stationary condition (Slutsky, 1937). The general expression of ARMA models is written as ARMA (*p*, *q*). The mathematical expression is given in Equation 3.1.

$$\dot{y}_t - \phi_1 \dot{y}_{t-1} - \cdots - \phi_p \dot{y}_{t-p} = \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \ +\theta_q \epsilon_{t-q} \tag{3.1}$$

The coefficients of ϕ and θ which are the autoregressive and moving average parameters, respectively satisfy stationarity and invertibility conditions. The most important assumption in time series analysis is that stationarity is required to make statistical inference. In a stationary process, the mean, variance, and autocorrelation structure do not change over time. The uncorrelated random variable of ϵ_t is white noise, which is assumed independent and identically distributed (i.i.d) random variables, and it is distributed normally with zero mean and constant variance. The orders of autoregressive models and moving average models are p and q, respectively.

The backward shift operator is called B defined by $By_t = y_{t-1}$. The autoregressive part $(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^P)$ and moving average part $(1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q)$ are represented by $\phi(B)$ and $\theta(B)$ polynomials.

To be able to apply AR, MA and ARMA models mentioned above, the time series must maintain the assumption of stationarity. The Autoregressive Integrated Moving Average (ARIMA) model has been implemented for time series where the stationary assumption cannot be provided which is proposed by Box *et al.* (1970). The ARIMA model has three parts which are Auto Regressive (AR), Integrated (I), Moving Average (MA). The autoregressive part covers the lags of the differenced series, and it is a linear regression that relates past values of the series to the future values. The integrated part *(I)* is the number of differences to make time series stationary. Moving average terms cover the lag of errors and relate past forecast errors to future values of time series.

The general expression of ARIMA (*p, d, q*) is given as Equation 3.2.

$$\phi(B)(1-B)^d \dot{y}_t = \theta(B)\epsilon_t. \tag{3.2}$$

The d^{th}difference operator shows how many differences are required to make the series stationary. The ordinary AR and MA polynomials are represented by $\phi(B)$

and $\theta(B)$, respectively. The uncorrelated random variable of ϵ_t is independently and identically distributed with a mean of zero and a constant variance of σ^2 which is generally denoted as WN (0, σ^2). The model parameters are generally estimated by maximum likelihood estimation and the model errors should be normally distributed.

3.2 Exponential Smoothing Methods (ETS)

Exponential Smoothing is proposed by Holt in (1957), Brown (1959), and Winters (1960). Exponential smoothing forecasts are weighted averages of previous observations, with exponentially decreasing weights over time. This methodology can also be used when there is seasonality and trend in the data (Hyndman *et al.*, 2021). Exponential smoothing can be done in a variety of ways. Some of these are discussed further down.

3.2.1 Simple Exponential Smoothing

Brown (1959) proposed SES, which covers no trend and seasonality. This model is commonly used for forecasting over a short period of time (Majeske *et al.,* 1998). The forecast equation is given below.

$$\hat{y}_{t+1} = \alpha y_t + (1 - \alpha)\hat{y}_t. \tag{3.3}$$

A weight value α is assigned to the most recent observation in the series y_t, the smoothed value at the previous time to forecast y_{t+1} at time *t+1*. The SES is based on a weighted average of the previous level and the current observation, as shown in Equation 3.3.

3.2.2 Holt's Exponential Smoothing

Holt (1957) invented simple exponential smoothing with two parameters to estimate data with a trend. Holt's exponential smoothing model consists of two equations: trend and level. The method is intended for estimating data that includes a trend component. The technique of the method is given in the following equations.

$$\hat{y}_{t+h} = l_t + hb_t, \tag{3.4}$$

$$l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}), \tag{3.5}$$

$$b_t = \beta^*\left(l_t - l_{t_1}\right) + (1-\beta^*)b_{t-1}, \tag{3.6}$$

where l_t represents an estimate of the series' level at time *t*, b_t is the difference between level term at time *t* and *t-1* and an estimate of the trend (slope) of the series at time *t*. The smoothing parameters are α and β^* which are between 0 and 1. The forecast horizon is represented as h and the h-step forecast is provided in Equation 3.4 (Hyndman *et al.*, 2021). As a result, the forecast is computed by multiplying trend terms by the horizon and adding the level term.

3.2.3 Holt-Winters Exponential Smoothing

The Holt-Winters Exponential Smoothing Method is an exponential smoothing theory utilized by Holt (1957) and Winters (1960) for the series with trend and seasonality. This method contains three straightening techniques for level, trend, and seasonality. There are two Holt-Winters Exponential Smoothing methods which are Holt Winters Additive Method and Holt Winters Multiplicative Method.

If the series under the study represents additive seasonal pattern, Holt Winters Additive Method is used. On the other hand, Holt Winters Multiplicative Method is used when the series has a multiplicative seasonal pattern function (Ozdemir *et al.*, 2020).

3.2.3.1 Holt-Winters Additive Method

The Holt Winters Additive process is preferred when seasonal variations are reasonably stable throughout the series (Hyndman *et al.*, 2021). The equations related to the model are given below.

$$(\hat{y}_{t+h}) = l_t + \ b_t h + s_{t-m+(k+1)} \tag{3.7}$$

where,

Level: $l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1})$ (3.8)

Trend: $b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1}$ (3.9)

Seasonal: $s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}$. (3.10)

The seasonal period is shown as *m*, the forecast horizon is represented as *h*, the forecast values are $(\hat{y}_{t+h}), l_t$ represents an estimate of the series' level at time *t*, b_t is the estimate of the trend of the series at time *t*. The seasonal component is shown as s_t, an integer confirming the seasonal component is shown as k in the equation (Hyndman *et al.*, 2021). The smoothed parameters are denoted by *α, β, γ* and these smoothed parameters take values between 0 and 1.

3.2.3.2 Holt-Winters Multiplicative Method

The Holt Winters Multiplicative method grasps the cases where the seasonal variations in the data vary in proportion to the level of the series (Hyndman *et al.*, 2021). The equations related to the model are given below.

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

(3.11)

where,

Level: $l_t = \alpha \frac{y_t}{s_{t-m}} + (1-\alpha)(l_{t-1} + b_{t-1})$

(3.12)

Trend: $b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1}$

(3.13)

Seasonal: $s_t = \gamma \frac{y_t}{(l_{t-1}+b_{t-1})} + (1-\gamma)s_{t-m}$

(3.14)

The seasonal period is denoted by m, and the forecast horizon is denoted by h. The forecasted values are $\hat{y}_{t+h}$ for *h* period, l_t denotes a prediction of the series' level at time *t*, $b_t$is an estimate of the series' trend at time *t*, s_t is a seasonal component, *k* represents an integer that confirms the seasonal component (Hyndman *et al.*, 2021). The smoothing parameters are between 0 and 1, which are α, β^* and γ. All these parameters are incremented until the smallest MSE value is reached. All exponential smoothing models require initial values for level, trend, and seasonal components.

3.3 Trigonometric Seasonal Models (TBATS)

The ultimate goal of Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components (TBATS) is to use the multiple techniques to forecast time series data with complex seasonal patterns (De Livera *et al.*, 2011).

The TBATS was developed by De Livera *et al.* (2011) using a combination of Box-Cox transform, ARMA errors and trigonometric seasonal patterns to handle complex seasonal patterns (Gos *et al.*, 2020). The methodology used in TBATS modeling is represented.

$$y_t^{(\omega)} = \begin{cases} \dfrac{y_\omega^t - 1}{\omega} & \omega \neq 0, \\ log y_t & \omega = 0, \end{cases}$$

(3.15)

$$y_t^{(\omega)} = l_{t-1} + \emptyset b_{t-1} + \sum_{i=1}^{T} s_{t-m_i}^{(i)} + d_t$$

(3.16)

where ω is Box-Cox parameter, the original series y_t is transformed using the Box-Cox transformation in the first equation. The transformed series $y_t^{(\omega)}$can be extended by following equations.

$$l_t = l_{t-1} + \emptyset b_{t-1} + \alpha d_t,$$

(3.17)

$$b_t = (1 - \emptyset)b + \emptyset b_{t-1} + \beta d_t,$$

(3.18)

$$s_t^{(i)} = s_{t-m_i}^{(i)} + \gamma_i d_t,$$

(3.19)

$$d_t = \sum_{i=1}^{p} \emptyset_i d_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t,$$

(3.20)

where l_t is the local level in period *t*, *b* and b_t are the long-run and short-run trends, and the trend damping parameter is shown as $\emptyset$ and the smoothing parameters in the concept of BATS are *α, β* and γ_i for *i= 1, ..., T*. In addition, d_t follows an ARMA *(p, q)* process and ϵ_t is a Gaussian white noise process with mean equal to 0 and constant variance equal to σ^2 (De Livera *et al.*, 2011). The seasonally trigonometric component is depicted by the following equations.

$$s_t^{(i)} = \sum_{j=!}^{k_i} s_{j,t}^{(i)},$$

(3.21)

$$s_{j,t}^{(i)} = s_{j,t-1}^{(i)} cos\lambda_j^{(i)} + s_{j,t-1}^{*(i)} sin\lambda_j^{(i)} + \gamma_1^{(i)} d_t,$$

(3.22)

$$s_{j,t}^{*(i)} = -s_{j,t-1} sin\lambda_j^{(i)} + s_{j,t-1}^{*(i)} cos\lambda_j^{(i)} + \gamma_2^{(i)} d_t,$$

(3.23)

$$\lambda_j^{(i)} = \frac{2\pi j}{m_i}.$$

(3.24)

The above equations show a Fourier series-based trigonometric representation of seasonal components and $\gamma_1^{(i)}$, $\gamma_2^{(i)}$ and $\lambda_j^{(i)} = 2\pi j/m_i$ are the smoothing parameters. The stochastic level of i^{th} seasonal component is defined by $s_{j,t}^{(i)}$ and the stochastic growth required to describe seasonal variations in the seasonal component is defined by $s_{j,t}^{*(i)}$.

3.4 Support Vector Machine

Support vector (SV) machines are supervised learning models that examine data for classification and regression analysis which comes with associated learning algorithms. Boser *et al.* (1992) developed the SV machine in its current form at AT&T Bell Laboratories.

The following concept has been conceptually executed by the machine: The input vectors are non-linearly mapped to a feature space with a high dimension. As a result, Cortes *et al.* (1995) constructed a linear decision surface in feature space. SVM attempts to find a hyperplane to correctly divide a given training set and maximize the greatest distance on both sides of the data input between the closest illustrations to the hyperplane. Although SVM's working logic appears to be better suited to solving classification problems, the SVM algorithm which was converted to regression problems is also very powerful for solving the time series problems. The technique of SVR is based on the structured risk minimization principle and aims to minimize an upper bound of the generalization (Pai *et al.*, 2010).

In the given a set of data $(x_i, A_i)_{i=1}^{N}$, where x_i is the input vector, A_i is the actual vector and N is the total number of data patterns. The general expression of the regression function is as follows:

$$G = w\phi(x_i) + b \tag{3.25}$$

The property of the inputs is denoted by $\phi(x_i)$, and the coefficients are weights and bias. The coefficients w_i and b are obtained by minimizing the regularized risk function given in Equation 3.26.

$$P(G) = C\frac{1}{N}\sum_{i=1}^{N} L_{\varepsilon}(A_i, G_i) + \frac{1}{2}\|w\|^2$$

(3.26)

where,

$$L_{\mathcal{E}(A_i,G_i)} = \begin{cases} 0, if\ |A_i - G_i| \leq \mathcal{E} \\ |A_i - G_i| - \mathcal{E}, otherwise \end{cases}$$

(3.27)

The penalty function and the error term in the equation are the cost, C and $\mathcal{E}$ parameters, respectively. For each training observation, the penalty function applies a penalty, if the error terms are larger than ± ε via insensitive loss function (Ozdemir *et al.*, 2020). The error term illustrates the difference between actual values and values calculated by the regression function.

Minimizing the cost functions can be solved by the Lagrange theory. In the Lagrange theory, the multipliers of Lagrange satisfy the equality $\beta_i * \beta_i^* = 0$. These multipliers are determined by the regression hyperplane's optimal weight vector which is represented below.

$$W^* = \sum_{i=1}^{N} (\beta_i - \beta_i^*) K(x, x_i).$$

(3.28)

In the nonlinear problem, the regression function is obtained for the unknown data point below.

$$G(x, \beta, \beta^*) = \sum_{i=1}^{N} (\beta_i - \beta_i^*) K(x, x_i) + b,$$

(3.29)

where $K(x_i, x_j)$ is a kernel function, whose value equals the inner product of two vectors which are x_j, x_i, Φ given in the Equation 3.26, transforms the input vectors x_j, x_i into a high dimensional space. Kernel function is introduced to be able to create non-linear hyperplanes and separate complex structures in the series. There are several Kernel functions which are Gaussian Kernel Radial Basis Function (RBF), Linear, Sigmoid and Polynomial, which affect the accuracy of SVR (Bouzerdoum *et al.*, 2013).

3.5 Random Forest

Breiman (2001) proposed the Random Forest, which is a gathering of regression trees, each identified in a bootstrap sample of the raw data. It relies on an ensemble learning method for classification and regression and works by constructing multiple decision trees without replacement at training time. Random Forest has been used for various purposes such as weather forecasting, solar radiation forecasting, and biostatistics (Naing *et al., 2015).*

Random forest consists of two principles: bagging and random subspace method (RSM), which is executed for each node of the classification and regression tree (CART) (Breiman, 2001). Not only the training data, but also the input variables are arbitrarily chosen when constructing each decision tree classifier or decision tree regressor (Géron, 2019). The procedure of Random Forest is as follows: X denotes the training dataset of dimension $N \times n$, where N denotes the number of observations, and n is the number of input features, the random subset with n' is initially created using replacement sampling technique which is bootstrapping to generate individual decisions (Yu *et al.*, 2017).

The target values of the training dataset with dimension $N \times 1$ are represented by Y, and the number of trees in the model is represented by L. Each decision tree in the RF is denoted by T_i, where $i = 1, \dots, L$. The number of features chosen at random in each node of the decision tree is p features out of m (Qiu *et al.*, 2017). The value of p can be selected as $\sqrt{m}$ (James *et al.*, 2021) for the classification problem and $m/3$ (Liaw *et al.*, 2002) for the regression problem.

Firstly, in the RF algorithm, the training set is created by sampling *B* times from all observations, with T_i replacement for each decision tree. To calculate the best split criterion for T_i, *m* randomly selected features are used in each node of one decision tree and this step repeated until the decision tree is large enough.

The average of each tree's forecasts applied to the original data yields the final forecast. In a regression problem, the mean or median of the outputs is calculated, while in a classification problem, the majority rule is applied (Naing *et al.*, 2015).

For classification problem: $f(x)$= majority vote $(T_b(x))_1^B$

(3.30)

For regression problem: $f(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

(3.31)

3.6 Extreme Gradient Boosting (XGBoost)

The boosting algorithm is an ensemble learning algorithm proposed by Schapire (1990). The aim of the boosting algorithm is to construct a strong classifier from weak classifiers in the series by using the decision trees iteratively, so it is dealing with the bias-variance trade-off which has an ability to minimize the variance of the predicted parameter between samples by increasing the predicted parameter's deviation.

The boosting method has been used to develop many algorithms. The Gradient Boosting algorithm, which is among the developed methods, was also suggested by Friedman (2001). Gradient Boosting is a machine learning algorithm that combines Gradient Descent and Boosting. Gradient Boost comprises three major components: an additive model, a loss function, and a weak learner (Guelman, 2012). In order to optimize the cost function, the algorithm iteratively chooses a function (weak hypothesis) that points in the negative gradient direction and this process is iterated until convergence has been obtained and the last decision has been made (Ribeiro *et al.*, 2020). In the Gradient Boost method, prediction is obtained by averaging the regression estimates. In classification estimation, the majority rule is used for classification tasks.

XGBoost, which was proposed by Chen *et al.* (2016) is an implemented gradient boosting decision tree-based algorithm that can build boosted trees quickly and in a linear way (Li *et al.*, 2019). XGBoost supports both Classification and Regression Trees (CART) and linear classifiers as base classifiers. The loss function is expanded by XGBoost using a second-order Taylor expansion (Li *et al.*, 2019). The XGBoost model can be shown as a formula.

$$\hat{y}_i \coloneqq \sum_{k=1}^{K} f_k(x_i), \qquad f_k \epsilon\, F$$

(3.32)

where *K* is the number of trees and *F* represents all regression trees, f_k illustrates one regression tree, the predicted value is shown as $\hat{y}_i$.

The objective function is given below.

$$Obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{i=1}^{t} \Omega(f_t) + constant$$

(3.33)

where *l* is loss function which enables to measure the difference between prediction $\hat{y}_i$ and real value y_i. Ω is the regularization term, which defines the complexity of the model, and it is used for avoiding the over fitting. It is calculated as follows.

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

(3.34)

where, *T* is the number of leaf nodes and *w* is the score represented by the leaf nodes. The aim of training minimizes loss of an objective function. The loss function is enlarged utilizing Taylor expansion in XGBoost. The final objective function is represented below.

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

(3.35)

$$= \sum_{i=1}^{n} \left[g_i w_q(x_i) + \frac{1}{2} h_i w_{q(x_i)}^2 \right] + \gamma T + \lambda \frac{1}{2} \sum_{j=!}^{T} w_j^2$$

(3.36)

$$= \sum_{J=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \epsilon I_j} h_i + \lambda) w_j^2] + \gamma T$$

(3.37)

$$g_i = \partial_{\widehat{y^{(t-1)}}} l \left(y_i, \widehat{y_i^{(t-1)}} \right), \quad h_i = \partial^2_{\widehat{y^{(t-1)}}} l \left(y_i, \widehat{y_i^{(t-1)}} \right).$$

(3.38)

The first-order derivative and second-order derivative of each data point in the error function, respectively, are g_i and h_i. I_j the index set of samples on each leaf node *j* (Yu *et al.*, 2021).

The formula of calculating the optimal value is as follows below.

$$\widetilde{L^{(t)}}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{\left(\sum_{i \in I_R} g_i \right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} - \gamma \, .$$

(3.39)

3.7 Artificial Neural Network

Artificial neural networks (ANNs), also known as neural networks (NNs) are data processing systems associated with biological neural networks derived from natural brains (Hardesty, 2017). In terms of design, the connectivity of many autonomous individual processing elements in the neural network model works in a similar way to the interconnections of individual cells in the brain in some ways (Brockett *et al.*, 1994). An artificial neural network (ANN) is made up of artificial neurons, which are a collection of connected units or nodes that resemble the neurons in a biological brain. ANNs imitate the way our brains work and predict or classify the features.

In the A part of Figure 3.1, the chemical inputs from the dendrites are converted into electrical signals by the nucleus. Through the axon terminals, the signal passes on to the next neurons. Edges are the terms for the connections (Kim *et al.*, 2018). The weight of neurons and edges is typically adjusted as learning progresses. The signal strength at a connection can be changed by the weight.

The weights, biases, and activation functions transform the input values in a node and the perception's output values are passed onto the next activation functions in part B.

In the C part, the general structure of the ANN is given, it consists of three parts: input layer, hidden layers, and output layers. Data is introduced to the network through the input layer and the information in the data is processed in the hidden state. Eventually, the output layer represents the measured value based on the inputs (Samsudin *et al.*, 2010).

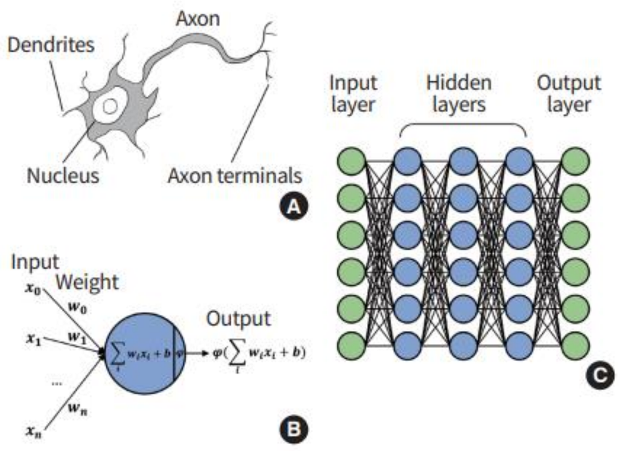

Figure 3.1. The Structure of Neural Network (Kim *et al.*, 2018)

First, ANNs have shown that, unlike traditional model-based methods, it does not have any prior assumptions of the model form required in the model building process. Second, ANNs can be widely applied in various situations. It also performs by extrapolating previous behavior patterns instances to forecast future nonlinear models and a predefined nonlinear model. Third, nonlinear methods have been used to create ANNs. It has been revealed that a network can perform nonlinear modeling without having to approximate any continuous function to any prior knowledge about feature relationships (Zhang *et al.*, 1998).

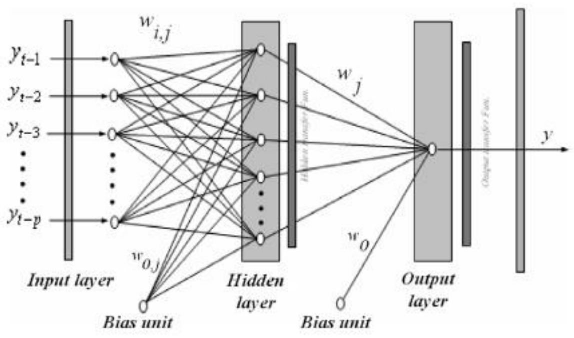

Figure 3.2. The General Concept of Artificial Neural Network (Chughtai *et al.*, 2008)

As illustrated in Figure 3.2, the working structure of ANN consists of three steps: multiplication, summation, and activation. A unit gets inputs which are multiplied by the weights and the weighted inputs are gathered together in the second step. Next, to adjust the threshold of the transfer function called as an activation function, a bias term is added. The activation function converts the sum of the weighted inputs and bias into the outcome in the final step. The status of the neuron within the network identifies the type of activation function (Zhang *et al.*, 1998). The model equation is given below.

$$y_t = w_0 + \sum_{j=1}^{q} w_j . g \left(w_{0j} + \sum_{i=1}^{p} w_{ij} . y_{t-i} \right) + \epsilon_t$$

(3.40)

where *i = 0, 1, 2, ..., p; j = 1, 2, ..., q*; y_t is the output, y_{t-1}, ..., y_{t-p}, are inputs, α_j and β_{ij} are model parameters that are connection weights; the number of input nodes is called p and q is the number of hidden nodes and the hidden layer is represented as g. The activation function will identify the empirical formula of the ANN and the network's non-linearity. There are several types of activation functions which are non-linear such as logistic or sigmoid Equation 3.42 and hyperbolic functions. Also, other activation functions which are linear, and quadratic can also be used for the different modeling applications. The sigmoid function converts the values into a range of 0 to 1.

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

(3.41)

Dynamic and static ANNs are the two major categories of ANNs. The ANN is referred to as a static network if the output signals are generated directly from the given input. It is said to be dynamic when the network's output is the input of ongoing and prospective neurons (Lewis, 2016).

The learning process is a critical component of the model's success which is called 'back propagation'. In this algorithm, the weights are updated and have been initialized either randomly or certain techniques such as Gradient Descent Algorithm. It enables minimizing by the help of the loss/errors of training any NN (Nielsen, 2015).

3.7.1 Feed-Forward Neural Network

The static neural network described in the previous section is the neural network where outputs are generated by the inputs without feedback. A feed forward neural network (FNN) which is a static neural network, is a type of neural network in which the nodes' contacts do not form a cycle (Ozdemir *et al.*, 2020).

The single layer feed forward neural network and the multi-layer feed neural network are the two parts of this neural network. The single layer feed forward neural network has two layers which are input and output layers. The multi-layer feed forward neural networks have three layers which are input layer, hidden layer and output layer. Figure 3.3 demonstrates the network design of a multi- layer feed forward neural network. The working mechanism in both models is forward: data continues to flow from the input nodes to the output nodes, passing through any hidden nodes. Single hidden layer feed forward neural networks are the most widely used model for time series and forecasting compared to multi-layer feed forward neural networks. It can be applied to forecast one step ahead values (Zhang, 2003).

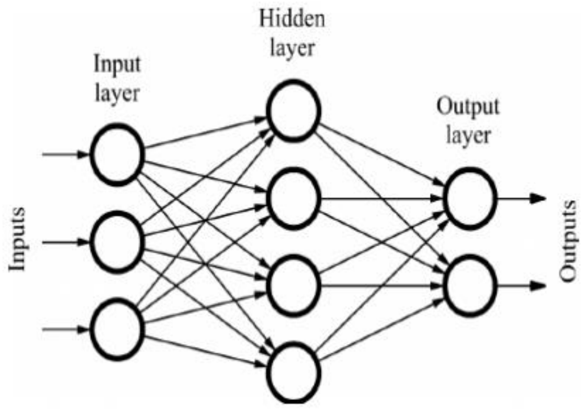

Figure 3.3. The Structure of Feed Forward Neural Network (Ozel *et al.*,2009)

3.7.2 Long Short-Term Memory

A typical feed-forward neural network may not be a good option for time series forecasting since it assumes the independence of both train and test data, and it needs fixed length input and output. Recurrent Neural Network arises as a solution for this problem by using feedback connection to account for earlier states in addition to the current input before producing the final output. In this process, a duplicate of the previous values for the layer containing the recurrent nodes is saved, and they are then used as an additional input for the following phase. This allows the network to display dynamic temporal behavior for a given time sequence. However, this structure suffers from long term dependency because the gradients approach to zero during the training. This problem is called the vanishing gradient problem.

The long short-term memory (LSTM), a variation of Recurrent Neural Network, is a solution to this gradient problem explained above. LSTM has feedback connections, so it can learn long-term dependencies, which is effective in sequence prediction and classification and has a structure of simple repeated secrets.

LSTM was established by Hochreiter and Schmidhuber in 1997 to overcome the issue of optimum gradients and non-convergence in RNN (Hochreiter *et al.*,1997). LSTM expertise in many fields such as robot control (Mayer *et al.*, 2006), time series prediction (Schmidhuber *et al.*, 2005), speech recognition (Graves *et al.*, 2005), human action recognition (Baccouche *et al.*,2011).

A classic LSTM network consists of cells, which are memory blocks. In an LSTM cell there are forget, input and output gates. The memorization process can be controlled by LSTM's gating mechanism. Gates that open and close provide for the storage, writing, and reading of data in LSTMs. The forget gate determines whether to remove existing data and the input gate determines how much new data will be added to the memory. Finally, the output gate determines whether the cell's present value relates to the output (Siami-Namini *et al.*, 2019).

The Feed Forward neural network has two assumptions which independence with train and test dataset and transformation to vector. Because of this reason, time series forecasting by using a feed forward neural network is not an appropriate approach. Recurrent Neural Network which is appropriate for time series analysis can handle the sequential dataset since its architecture of uses the previous layers and feeds the signals both forward and backward. Inputs with forward and backward feed the model by using RNN. Therefore, the working mechanism of NN models is like AR models (Krenker *et al.*, 2011). In general, in LSTM, it performs well in time series data as the behavior of historical data is kept in memory (Bilgili *et al.*, 2022).

A hidden vector, *h*, and a memory vector, *m*, are maintained in an LSTM, and they control state updates and outputs at each time step, respectively. The structure of LSTM is shown in Figure 3.4.

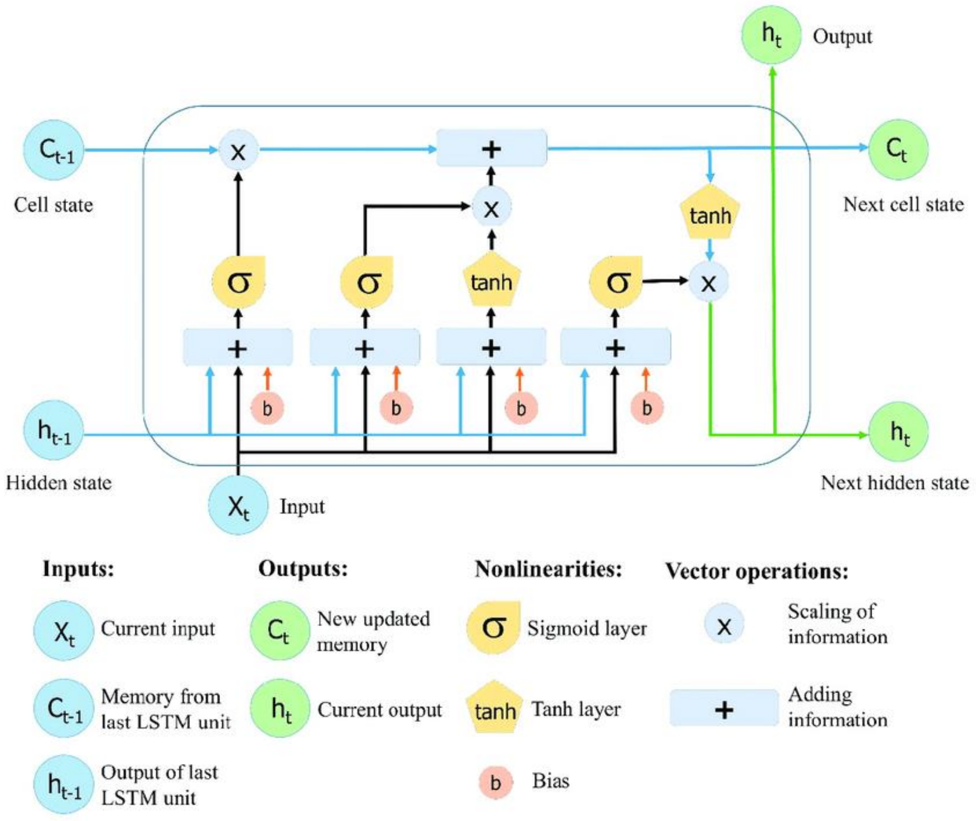

Figure 3.4. The Structure of the Long Short-Term Memory (LSTM) Neural Network (Yan, 2017)

In the forget gate, the sigmoid function is used to determine what information is required based on the values of h_{t-1} and x_t. The output of this gate is f_t and is a value between 0 and 1. Output 0 means getting rid of the learned value completely, if 1 holds values.

This result is calculated as follows:

$$f_t = \sigma(W_{fh}[h_{t-1}], W_{fx}[x_t], b_f)$$

(3.42)

where b_f is called the bias value, W_{fh} is the weight of the previous hidden state and W_{fx} is the weight of the input. In the input gate, there are two different layers which are *sigmoid* and *tanh* layer. The *sigmoid* layer determines which values will be updated and *tanh* layer generates a new value which will be stored in the memory (Siami-Namini *et al.,* 2019). These layers can be calculated as follows.

$$i_t = \sigma(W_{i_h}[h_{t-1}], W_{i_x}[x_t], b_i), \tag{3.43}$$

$$c_t = \tanh\left(W_{c_h}[h_{t-1}], W_{c_x}[x_t], b_c\right). \tag{3.44}$$

In the output gate, the *sigmoid* layer is used to understand which information in the memory contributes to the output. The values between -1 and 1 are then mapped using a non-linear tanh function, and finally is multiplied by both (Siami-Namini *et al.*, 2019). The equation of the final step is given below.

$$o_t = \sigma(W_{o_h}[h_{t-1}], W_{o_x}[x_t], b_o). \tag{3.45}$$

$$h_t = o_t * \tanh\,(c_t). \tag{3.46}$$

3.7.3 Bayesian Regularized Neural Network

While optimizing Artificial Neural Network models, it is attempted to reduce error values by changing the weights. However, this method can sometimes lead to overtraining or overfitting issues (Guelman, 2012). To reduce overfitting, the Bayesian regularization method was developed for nonlinear systems (Burden *et al.*, 2008).

The sum of squared error is between the model output and the target value. In the training part, it is expected to have a low SSE because of this reason Bayesian regularization term is added to this equation (Guelman, 2012).

$$F = \beta E_D + \alpha E_W \tag{3.47}$$

where F denotes the objective function, the sum of squared errors is shown as E_D, the sum of squares of the network weights is illustrated as E_W. The objective parameters are α and β.

The weights of networks whose density function is written in Bayes rule are contemplated as random variables, the weights of networks are considered (Dan Foresee *et al.*, 1997). The probability density function of the weight of networks is calculated with given data as follows.

$$f(w|D,\alpha,\beta,M) = \frac{f(D|w,\beta,M)f(w|\alpha,M)}{f(D|\alpha,\beta,M)} \tag{3.48}$$

where *D* denotes the observed data and *M* is a particular neural network. The prior density is P(w|α, M), which covers the knowledge of the weights. The likelihood function is P (D| w, β, M), which is the probability of the data occurring given the weights. P (D| α, β, M) is a normalization factor (Burden *et al.*, 2008). Assuming that the noise in the data is normal, the density function was determined for the weights. In this condition, given the parameters *w*, the probability of the data is given by below.

$$f(D|w,\beta,M) = \frac{\exp\left(-\beta E_D\right)}{Z_D(\beta)} \tag{3.49}$$

where $Z_D(\beta) = (\frac{2\pi}{\beta})^{N/2}, \beta = 1/\sigma^2$ .The density of prior can be shown as;

$$f(w|\alpha, M) = \frac{\exp(-\alpha E_W)}{Z_W(\alpha)} \tag{3.50}$$

where $Z_w(\alpha) = \int \exp(-\alpha E_W)$. The last two equations are combined with probability density function of the network weights. The optimal weights can maximize the posterior probability by using the following equation.

$$f(w|D, \alpha, \beta, M) = \frac{\exp(-(\beta E_D + \alpha E_W))}{Z_W(\alpha) Z_D(\beta)} \tag{3.51}$$

where $Z_W(\alpha) = \int \exp(-F)\, dw$. The optimal weights which maximize the posterior probability are obtained by Nguyen and Widrow algorithm, which starts with initial weights and optimizes them via Gauss-Newton Algorithm (Nguyen *et al.*, 1990).

3.8 Forecast Accuracy Measures

Forecast performance measures how well applied models can predict future values. It is discovered by contrasting which model approach yields superior outcomes because of performance metrics. The literature contains various types of performance metrics. This study considered computation time, mean absolute percentage error, and root mean square error.

3.8.1 Root Mean Square Error

While the RMSE measures the distance between two observed and predicted values, it also measures the magnitude of the √(1/h) error by using the Euclidean distance factor. In time series analysis, the root mean squared errors are the most chosen performance metric (RMSE). The forecasting method has high accuracy if the RMSE performance value is low. The expression of RMSE is given below.

$$RMSE = \sqrt{\frac{1}{h}\sum_{i=1}^{h}(y_t - \hat{y}_t)^2}$$

(3.52)

where h is the number of forecasted points, y_t is the actual value and the $\hat{y}_t$ is the forecast value.

3.8.2 Mean Absolute Percentage Error

The advantage of the Mean Absolute Percentage Error (MAPE) is that it minimizes the impact of positive and negative errors on one another, making it easier to compare the forecast performance of various models. It is a widely used performance metric for forecasting models and measuring the accuracy of time series; it determines the absolute percentage error for each period (Charles *et al.*, 2013). The MAPE is represented by the equation below.

$$MAPE = \frac{1}{h}\sum_{t=1}^{h}\left|\frac{y_t - \hat{y}_t}{y_t}\right|$$

(3.53)

where h is the number of forecast points, y_t is the actual value, and the $\hat{y}_t$ is the forecast value. The forecast technique has high accuracy if the MAPE value is low, same with RMSE performance.

3.8.3 Computational Time

Computational time is also considered along with forecast accuracy measures. Utilizing R's sys.time() function, the working time of the techniques mentioned for each group is calculated in seconds. LSTM algorithm, Python is used and ipython-autotime is implemented to measure the time it takes to execute each cell.

CHAPTER 4

ANALYSIS

In this chapter, the forecasting of the number of claims according to the month in service groups (MIS) is performed using the warranty data of a company in the automotive industry with the techniques described in Chapter 3. The properties of the data set, the forecasted groups used in the study, and the implemented data preprocessing techniques will be explained to present the specific details of the analysis process. The 3-month numerical results for each model will be demonstrated with visual and numerical examples and the accuracy of selected models for each group will be assessed.

4.1 Dataset

The warranty data, which is formed by keeping the records of the malfunction requests received under the warranty, illustrates the product quality and directly affects many departments in the enterprise. In this study, the warranty dataset from an automotive manufacturer covers three variables, which are the period of use of the defective products by the customer (MIS), the total number of claims and the time. The warranty contract of the product covers 2 years, so there is a total of twenty-five months in service groups within the warranty period. The dataset contains 44 monthly total number of claims for each MIS group from November 2021 to June 2022. There are a total of 1100 observations in the dataset, and these observations cover the 44-month total error number of 25 different MIS groups.

If the product breaks down within a month of the warranty period beginning, it is in MIS 0 group. If the product works fine in the first month but breaks down within the second month of the warranty period, it is in MIS1 group, and so on. In other words, for the MIS 0 group, the total number of claims from November 2021 to June 2022 is included, and the other MIS groups include the total number of failures in the same time periods. For each MIS group, 3-month periods are forecast.

4.2 Data Preprocessing

Preprocessing aids in converting original data into a format suitable for modeling. Data preparation, particularly in machine learning, enables the model to incorporate the unidentified structure underlying the issue (Brownlee, 2020).

There are many data preprocessing techniques in the literature. These techniques can be shaped according to the needs or assumptions of the models. Many preprocessing techniques, such as Box-Cox transformation, power transformation, detrends, and deseasonalization, help ensure the assumptions of some statistical time series modeling methods, whereas some machine learning models can only be used with standardized data. The preprocessing processes applied in this study were designed according to the requirements of the models used. Moreover, all preprocessing processes were performed for each MIS group themselves, and these techniques are explained in detail below.

- **Original Data:** There has been no preprocessing applied to this data set. The original dataset was used for the ARIMA model because the required preprocessing for the ARIMA model such as differencing for stationarity is already applied by the *auto.arima* function.

- **Box-Cox Transformation:** The Box-Cox transformation technique was applied with the *BoxCox* function to assess the stationarity in variance (Hyndman *et al.*, 2021). The machine learning algorithms utilized in this study can also deal with non-stationary variance. In the traditional time series method, the *tbats* and *auto.arima* function implement the Box-Cox transformation internally. Moreover, the Box-Cox transformation was applied manually for the ETS model. The *nnetar* function contains the lambda parameter which is used for Box-Cox transformation.

- **Min-Max Scaling:** In this study, scaling was applied before applying LSTM, XGBoost, RF and SVM. In addition, the Bayesian Regularized Neural Network and Feed-Forward Neural Network functions implemented in R in this study, apply scaling within themselves. The formula for Min-Max scaling is given below. y_{max} and y_{min} are the minimum and maximum values respectively, y_t is the actual value at time *t* and y'_t is the observations scaled value at time *t*.

$$y'_t = \frac{y_t - y_{min}}{y_{max} - y_{min}}. \tag{3.54}$$

- **Time Delay Embedding:** Machine learning methods cannot account for the time series' autocorrelation property the way statistical methods can. Time delay embedding, also known as sliding window, is a technique for converting a time series into a matrix of time-dependent datasets (Von Oertzen *et al.*, 2009). By converting a long data series into short time-dependent segments, time dependency is brought to the forefront and the problem is solved. The average number of lags was established as 5 in the ACF-PACF graphs within each MIS group. The ACF-PACF graph can be found in Figure A.1. Except Feed-Forward Neural Network, since the *nnetar* function covers the finding best past lags itself. In this study, the lag number in embedding matrices has been integrated into the models in Random Forest, SVM, XgBoost, BRNN, LSTM techniques accordingly.

- **Splitting Datasets:** For both statistical methods and machine learning methods in each MIS group, the last 3 of 44 observations were reserved as test data and the remaining data as training data. In the implementation of all machine learning models, the last 20% of the training data was used as validation data to tune the parameters of models. The diagram of splitting dataset is given below. Then, the tuned parameters were implemented onto all training data, including validation, to forecast 3 months ahead in the test dataset. The validation set remains static for each group.

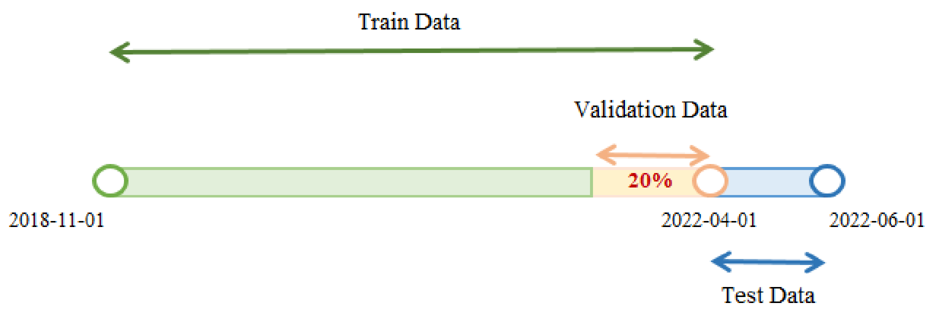

Figure 4.1. The Diagram of Splitting Dataset

4.3 Model Implementation

All models except the LSTM model were installed via R. The dataset was converted into a nested format according to the MIS groups in the model application part in R. In the LSTM model implemented using Python, each MIS group is converted to list format. Data pre-processing, data splitting, model construction, and parameter tuning were constructed separately for each MIS group.

It is a challenge to provide a numerical explanation of the estimated parameter, tuning parameter, total number of neurons and hidden layers of each model, as nine models are implemented for each 25 groups separetely. Instead, the model implementation section will go over the specifics and operation of statistical and machine learning techniques on this data.

4.3.1 Statistical Models

In the statistical forecasting parts, ARIMA, ETS, and TBATS models were used to forecast 3 months ahead of 25 different MIS groups. The assumptions of the models were checked, and the features of the functions used were considered.

4.3.1.1 ARIMA

As it is explained in the methodology chapter, the forecast is made after the trend is eliminated. To predict future periods, the autoregressive model (AR) first determines how many lagged series are necessary for addition to the parameter p. The series is intended to become stationary by using the difference(I) as the d parameter in the second section. The equation is then completed by the addition of the moving average model (MA), the q parameter, and the number of lags forecasts errors. The model is constructed using the original dataset because the ARIMA function incorporates a Box-Cox transformation within itself by using *lambda="auto"* argument in the *auto.arima* function. While the model is being constructed, the appropriate AR and MA components are the ones that minimize the Corrected Akaike Information Criterion (AICc), which is the default criteria in the auto.arima function. The expression of AICc is given below.

$$AICc = \frac{2kn}{n-k-1} - 2\ln(\hat{L}).$$

(3.55)

The series length, the number of parameters to be predicted, and the maximum value of the likelihood function using the observed data, are indicated by n, k, and $\hat{L}$ correspondingly. The *auto.arima* function used selects the best model that minimizes the chosen information criterion and estimates the parameters of the model using Maximum Likelihood Estimation. Hence, the normality assumption is crucial.

4.3.1.2 ETS

In this study, the *ets* function in R has been used to establish the ETS model. The ETS model can distinguish between addition and multiplication as the model's type using a function that utilizes exponential smoothing methods. The best model is selected using the *ets* function relies on the AIC, AICc, and BIC criteria minimizing. The normality assumption was confirmed before the model was constructed, and Box-Cox transformation was used for non-normal series (Svetunkov, 2022).

4.3.1.3 TBATS

TBATS model is a technique that includes trigonometric seasonality, Box-Cox transformation, ARMA errors, trend, and seasonal components. The best model is selected according to the lowest AIC value. This model is implemented via R with the *tbats* function.

4.3.2 Machine Learning Models

In the machine learning forecasting part, Support Vector Machine, Random Forest, Bayesian Regularized Neural Network, Feed Forward Neural Network, XgBoost, and LSTM models were utilized to forecast 3 months ahead of 25 different MIS groups. The lag number was set to 5 in the time embedding matrix for all machine learning techniques. As previously stated, since many MIS groups are predicted, the models cannot be expressed numerically. Instead, the principles of their general application will be explained.

4.3.2.1 Support Vector Machine

The SVM model setup utilized the *train* function from the *caret* package. SVM kernel parameter was assessed to constitute radial basis function and the method was chosen as *svmRadial* for all MIS groups. Moreover, two hyperparameters in this function can be tuned, which are the cost parameter, which can be defined as the penalty term in SVM, and the gamma parameter, which controls the effect distance. The optimal parameters of the model are obtained each series separately by using the *train* function in the *caret* package.

4.3.2.2 Random Forest

Random Forest model was established by using *randomForest* function in R. The parameters utilized in the function are *mrty* and *ntree*. Since the time lag number is 5, the *mtry* value is taken from 1 to 4, and the *ntree* value is determined from 200 to 2100. The parameters that provide the minimum MAPE and RMSE value for each MIS group are provided by the tuneRF function and defined as the best parameter.

4.3.2.3 XgBoost

The *caret* package's *train* function's "xgbTree" method was used to implement the XGBoost model. Model tuning parameters assist to find the best model on its own via the *train* function. Tuning has been done on the Caret package's boosting iteration (nrounds), max tree depth (max depth), shrinkage term (eta), minimum loss reduction (gamma), subsample ratio of columns (colsample by tree), and subsample percentage (subsample) parameters.

The ranges determined for these parameters are defined by the *expand.grid* function. The eta helps to avoid overfitting problem and its range is from 0 to 1. In this study, it is chosen on (0.02, 0.025, 0.001) interval. The gamma parameter indicates the minimal loss reduction necessary to split. Three numbers were generated with the gamma parameter being uniform from 0 to 4. In this study, boosting iterations, which is nround, from 100 to 800 were selected by 100 increments. Subsample ratio of columns parameter supplies to select subsample when constructing the tree. The range of colsample_bytree is 0, 0.1, and 0.2. Although the maximum depth of the tree is default 6, and it is used (1, 2, 3) interval in this study. The minimum child weight is that minimum sum of instance weight (hessian) needed in a child. The values of minimum child weight are between 10 and 15. Subsample indicates the percentage of each tree's observations that are drawn at random. In this study, three values are applied in the range which is uniformly distributed from 0.7 to 1.

All specified parameter ranges have been tuned separately for each group, and the caret function has been used to determine the optimum parameters, and models have been developed.

4.3.2.4 Feed-Forward Neural Network

The Feed-Forward Neural Network model was constructed by using nnetar function in R for each MIS group. A feed-forward neural network with one hidden layer performs the function. The function itself calculates weights in the hidden layer. However, the learning rate and epoch number information are not available for the *nnetar* function. The parameters that can be tuned for this function are the number of neurons and Box-Cox transformation. Neuron hyperparameter is tuned over Box-Cox applied dataset. The number of neurons was taken from 1 to 5 and the number of neurons providing the lowest RMSE value was selected for each MIS group. Moreover, the number of lags is calculated within the function and the network is trained for the given data utilizing back propagation, with all prior lags initiated as inputs.

4.3.2.5 LSTM

The Long Short Term Memory method, in contrast to other models, was constructed by using *keras* package in Python because it needs careful hyperparameter tuning which is very limited in R. The time embedding was applied on each group, just like in previous machine learning models, along with normalization and min-max scaling procedures for each MIS group.

Input data sequence parameters are time step window and model batch size are 5 and 4 respectively. The time step window was used to create sequence length. In the hyperparameter search parameters are tune epoch, max trials, which are 100 and 25 respectively. The target of the adjustment algorithm was determined as objective *val_loss*, early stopping patience 10, learning rate reduce patience 2, learning rate reduce factor 0.5.

In this study, many various layers, number of neurons, drop out and layer weight initialization methods were applied. It was challenging to determine the ideal architecture because each MIS group in the dataset had a limited observation and a unique structure. In this study, the minimum error value in the experimentally applied model parameters was reached in the model structure specified below.

The model was constructed using one LSTM layer. The *hp.int()* function, which enables us to specify the space search of hyperparameters, was used to select the optimal number of units within the layer. The *relu* activation function is used and the number of cells is defined from 1 to 5 with a step of 1. Return sequence should also be set to True since it allows another LSTM to use the output.

A drop layer was added after the layers were constructed to prevent the neural network from over-learning. Neurons are randomly concealed in the dropout layer based on the dropout rate. The *hp.choice()* function is used in this study to select between 0.2, 0.4, 0.6 and 0.7 as the ideal dropout value. The range of drop out values has been kept wide to prevent overlearning and underlearning while modeling 25 different MIS groups at the same time. The lstm layer was concatenated to one fully connected layer. It is choosen an optimal value between 1-5 with a step of 1 and the activation function is selected as *relu*. The learning rate is set as 0.01 or 0.1 with the *hp.choice()* function. In the selected MIS groups, seven different weight initiation methods from the layer weight initiator methods were applied. These are random normal, random uniform, glorot normal, glorot uniform, variance scaling, zero-weight initialized methods.

Glorot uniform initializer, also called Xavier uniform initializer, was used to set the initial random weights of Keras layers, which outperformed most groups in the applied methods. After adding the model to the last fully connected layer, the model should be compiled. Three parameters are used for this step, loss, optimizer, and metric. The term *loss* refers to the metric that will be used to indicate the model's error in the training stage, which is chosen as *mse* in this study. The model is evaluated using a metric called *mape* which resembles loss but is not utilized during training.

The Adam optimization technique was proposed by Kingma *et al.* in 2014. This method is based on adaptive estimations of lower-order moments and is used to optimize stochastic objective functions with first-order gradients. They claimed that their optimizer is computationally effective, ideal for issues with little memory needs, lots of data, and/or many parameters, as well as suitable for the non-stationary and noisy slope issues we encountered in our study (Kingma *et al.*, 2014).

Adam optimizer was proposed in the car spare part demand forecast made in 2021 (Chandriah *et al.*, 2021). It has been stated that it is difficult to predict since the demand for automobile spare parts is constantly repeated. In this study, Adam optimizer was used because the number of warranty claims has a non-stationary and complex structure for each month in service group.

After compiling the model with the existing trial hyperparameters, it is defined keras tuner object and the learning rate schedule and early stopping were determined.

The *EarlyStopping* function was implemented with three parameters: monitor, patience, and mode, in Keras to help stop training when the model is no longer developed. The monitor shows what the model will be stopped early by considered, and the mode includes how the metric value we consider in the monitor should be (min or max). Patience is the number of epochs without improvement after which training will be early stopped. In this study, the monitor is defined *val_loss*, which is the value of cost function. Patience is adjusted to stop early if training does not improve within 10 epochs and mode is defined as minimum.

ReduceLROnPlateau which allows to reduce the learning rate when a particular metric stop improving was used as the learning rate schedule. The three parameters are implemented for the early stop function are also included in this function. In addition, the factor parameter, which helps to reduce the learning rate, was added as 0.5 and the *min_lr* parameter was utilized as the lowest learning rate as 0.02 in the study.

The dashboard of Weights&Biases is applied to illustrate loss, validation loss, accuracy, validation accuracy and all metrics are recorded by using *WandbCallback()* for each 25 MIS groups and all trial models are saved.

The optimal hyperparameter value for the dataset and models of all groups was derived via Bayesian optimization.

For each of the 25 groups, the optimal hyperparameter values were identified, and then 3-horizon prediction and inverse transformation were performed out on these values.

4.3.2.6 Bayesian Regularized Neural Network

The Bayesian regularization for the feed-forward neural networks model with the *brrn* function is configured separately for each of the 25 groups. Two-layer neural networks can accommodate the function. The number of neurons is the only variable in the package that can be altered. The model was constructed using the lowest

RMSE value for each group to determine the number of neurons, which ranged from 1 to 10. The number of epochs, α, β, E_d, and E_w are tuned within the function. The inputs and outputs are scaled within the function. The Gauss-Newton algorithm is used to optimize the weights, which are given a normal distribution as the prior distribution. The 25 different models are trained by using back propagation.

4.4 Empirical Analysis

4.4.1 Comparison of Modelling Performances

The performances of the nine models applied for each MIS group are evaluated in this section, according to the RMSE and MAPE metrics. The structure of the models and the methods used to prevent the models from overlearning are discussed. Also, optimized parameter values while applying the models are given in Appendix B.

The forecasting performances of the implemented models in all MIS groups were compared by applying non-parametric statistical tests for the MAPE metric and considering the computational time of models. In this way, the company can easily decide and implement the best model for the forecasting of the number of claims in the warranty processes.

All months in services groups are, on average, non-stationary. The series in Figure 4.2 were selected depending on MIS groups with different patterns, as has been seen above, the groups are not, on average, stationary.

At certain points in time, the total number of claims for each group decreased significantly. The most important reason is that the removal of many product series from production and the commissioning of different series with new engines. This change, which was applied from the production, was also reflected in the failure request in the warranty data. This steep decline in MIS groups and the short time interval in the data adversely affected the performance of the models.

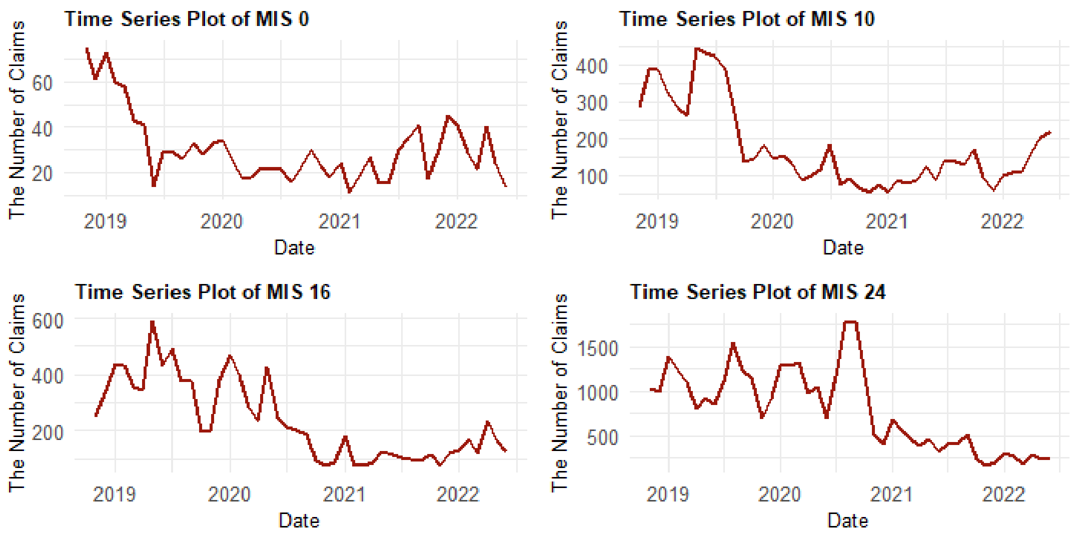

Figure 4.2. The Time Series Plots of MIS 0, MIS 10, MIS 16 and MIS 24

In the first figure, the MIS 0 group exhibited a significant drop in the first half of 2019, whereas the MIS 10 group declined after the first half of 2019. Although the MIS 16 group had ups and downs from 2019 to the end of 2020, there was a decline after the first half of 2020. Lastly, the number of claims in the MIS 24 showed a considerable increment in the last quarter of 2020, then represented a sharp decrease in 2021 and 2022.

It is also observed from the graph that the steep decline in MIS 24 is observed later than MIS 0. This is because it takes time for MIS groups to reflect the actions taken during the production phase. While its reflection to MIS 0 can be seen right away, its reflection to MIS 24 requires longer time. The impact of groups on each other is excluded because the product's month of use is assessed when determining product quality.

Table 4.1. and Table 4.2. present the RMSE and MAPE values for test data of both statistical and machine learning methods applied to the 25 months in services group, respectively. The MAPE values for the train data of the machine learning methods applied to 25 months, respectively, in the services group are in Table 4.3.

Table 4.1. The MAPE of All Methods of Test Dataset for each MIS Group

MIS	ARIMA	ETS	TBATS	SVM	RF	XGBoost	NNETAR	LSTM	BRNN
0	40.51	37.08	**36.50**	47.65	43.39	50.33	41.17	44.25	54.79
1	**18.80**	26.63	26.66	23.67	30.96	24.13	42.01	33.59	23.49
2	65.87	33.84	33.17	34.22	**17.68**	28.24	63.75	38.82	21.73
3	**10.31**	11.40	11.55	20.04	16.51	45.09	15.35	23.78	21.32
4	4.76	15.25	13.33	29.32	**3.27**	44.75	10.23	53.70	28.90
5	28.62	33.85	28.24	28.23	**3.75**	49.37	20.46	60.52	23.30
6	31.58	46.77	46.99	29.98	**9.59**	31.77	13.63	45.92	29.85
7	21.74	29.24	28.47	43.46	**9.95**	38.09	26.72	46.83	21.42
8	13.08	15.25	10.51	16.62	**3.98**	39.04	13.49	40.76	21.20
9	21.84	36.16	35.78	42.10	**5.84**	31.38	43.57	32.85	16.72
10	41.89	79.09	78.45	33.77	**9.59**	28.05	40.76	16.79	30.16
11	31.45	36.78	36.62	22.90	29.30	**20.90**	22.64	28.07	24.49
12	16.80	18.91	21.38	**15.14**	26.41	33.62	16.89	28.20	16.97
13	29.86	45.59	55.47	**14.81**	15.42	18.81	20.79	20.61	24.64
14	21.80	14.67	14.68	9.41	**8.87**	21.54	15.30	21.75	13.99
15	76.98	32.81	32.75	29.79	24.84	37.41	22.10	**21.65**	27.53
16	26.14	35.95	35.95	24.04	23.56	34.15	34.84	30.27	**23.45**
17	25.88	**8.98**	9.29	70.76	14.79	32.38	30.74	34.10	24.82
18	44.54	22.22	21.59	28.17	**18.89**	24.89	54.69	22.46	28.79
19	10.92	10.58	**8.35**	19.20	11.89	18.87	41.00	12.82	17.05
20	36.35	57.27	48.67	27.00	**2.53**	27.45	42.11	23.69	21.90
21	27.77	43.30	43.30	26.89	**12.88**	16.42	18.79	14.97	26.74
22	30.64	38.00	37.93	15.57	**6.87**	17.67	28.90	9.40	9.75
23	21.25	**17.00**	17.37	17.69	21.60	20.78	22.96	58.47	23.56
24	27.37	38.83	38.35	24.17	**10.25**	23.30	15.37	30.32	25.20

Table 4.2. The RMSE of All Methods of Test Dataset for each MIS Group

MIS	ARIMA	ETS	TBATS	SVM	RF	XGBoost	NNETAR	LSTM	BRNN
0	12.03	11.13	**10.88**	11.25	11.68	10.99	11.58	12.07	11.57
1	**46.66**	54.36	54.63	58.51	50.28	66.79	68.10	79.97	57.28
2	114.53	96.88	94.31	65.02	**32.44**	62.31	110.95	76.22	41.20
3	**35.04**	38.04	37.34	54.33	38.69	111.30	42.46	63.84	65.12
4	14.76	33.67	30.19	72.95	**9.25**	110.57	26.58	132.38	74.72
5	78.33	66.48	58.11	74.62	**9.75**	128.81	63.57	157.32	63.11
6	80.53	79.90	80.13	87.49	**23.99**	81.44	42.82	112.97	87.00
7	52.84	52.84	51.96	95.63	**21.48**	84.63	64.58	100.39	47.50
8	30.10	30.11	22.56	38.10	**9.31**	85.33	34.70	88.59	47.40
9	47.62	56.60	56.18	88.85	**14.55**	66.92	90.58	70.65	42.73
10	85.69	87.94	87.57	81.60	**20.49**	61.29	83.26	39.14	56.41
11	82.49	65.63	65.43	64.68	**48.07**	53.87	66.20	50.65	70.29
12	49.94	40.87	42.57	43.42	**39.47**	65.31	47.57	52.80	51.85
13	53.13	53.12	59.32	30.22	**28.68**	32.15	39.91	35.92	50.29
14	34.24	28.84	28.85	**14.47**	15.23	33.88	28.46	34.50	26.37
15	103.16	79.47	79.30	42.42	**37.23**	55.02	66.94	39.82	39.61
16	69.66	62.41	62.40	66.00	**56.93**	80.33	87.39	62.44	63.82
17	53.29	21.75	**20.89**	130.08	26.94	61.79	62.94	63.31	52.51
18	86.70	39.09	38.12	66.52	**35.82**	57.13	99.17	43.09	59.11
19	**18.23**	**18.23**	19.55	40.15	19.99	38.14	76.07	24.88	32.41
20	67.24	67.25	60.51	50.72	**6.20**	50.97	78.43	45.60	41.05
21	46.71	47.54	47.54	42.53	**19.50**	25.56	36.23	22.08	38.43
22	51.89	45.91	45.86	24.75	**12.34**	34.59	49.35	16.33	18.84
23	30.17	30.17	31.68	**29.86**	33.07	36.18	32.76	84.91	35.42
24	76.97	76.96	81.94	95.07	**29.45**	63.07	51.16	97.28	93.62

Table 4.3. The MAPE of All Methods of Train Dataset for each MIS Group

MIS	SVM	RF	XGBoost	NNETAR	LSTM	BRNN
0	24.97	34.13	32.60	22.53	20.35	28.54
1	26.93	32.29	36.16	28.09	19.41	26.92
2	16.91	33.03	43.78	20.26	23.51	23.39
3	27.85	32.74	47.73	20.37	30.12	28.99
4	9.87	36.93	48.88	24.79	28.90	33.38
5	28.32	39.63	51.88	21.01	12.23	27.31
6	25.77	39.30	48.53	24.18	11.90	30.64
7	51.51	54.14	57.30	33.01	23.57	36.28
8	30.35	52.84	55.29	8.53	22.49	34.31
9	46.45	37.11	57.62	19.81	25.39	29.44
10	10.27	32.35	48.07	20.69	43.23	26.51
11	34.53	34.15	45.87	22.30	37.02	27.18
12	38.33	42.49	45.33	34.31	36.60	32.97
13	36.51	47.85	58.17	8.68	27.44	26.56
14	39.48	39.47	64.06	24.57	26.72	28.18
15	32.83	33.59	63.73	29.36	28.94	25.36
16	28.20	38.35	42.60	10.66	36.20	28.33
17	17.13	43.17	49.46	30.11	34.76	26.42
18	28.16	43.88	51.64	11.46	27.54	30.14
19	27.06	40.83	53.97	8.15	34.24	32.18
20	36.02	46.38	53.49	12.54	40.06	27.14
21	29.03	34.33	49.69	21.83	22.63	31.10
22	19.95	24.84	50.49	13.27	23.70	20.31
23	18.32	26.60	56.54	24.78	22.90	19.72
24	20.77	29.04	54.41	23.00	42.95	13.10

In statistical approaches, ARIMA was the best model only in the MIS 1 and MIS 3 groups, while the ETS approach was the best in 2 groups, MIS 17 and MIS 23. TBATS was the model that showed the best performance in the MIS 0 and MIS 19 group.

In machine learning algorithms, the RF is the model that shows the low error values in 14 of 25 groups as seen from the table. Also, RF is the most effective and simplest method to apply for this dataset. The LSTM model, a Deep Learning method, was performed successfully in one group, MIS 16. Moreover, it was observed that the XGBoost algorithm outperformed for the MIS 11 among all models with respect to MAPE. SVM was the more efficient than other ML models in 2 groups out of 25 groups. BRNN was only the most successful model in one group. NNETAR was not able to make the best-performing prediction within all MIS groups.

When the MAPE values for the test and train data sets in Table 4.2 and Table 4.3 were compared, it was observed that the applied methods MAPE of test values are considerably higher than MAPE of train values. It shows that implemented methods overfitted some MIS groups. Finding the appropriate model was quite challenging as the dataset was limited to 44 months and each group had a different pattern. The parameter values of the models were optimized so that the methods would not overlearn the MIS groups.

Since the working environment of XGBoost and RF methods is flexible in R, the appropriate model architecture has been established. According to the MAPE values for test and train data sets, all machine learning methods were overfitted only in the MIS 0 group. In order to prevent overlearning in other MIS groups, the model parameters used for this purpose were tuned. The number of ntrees is limited in the RF method. In the XGBoost model, on the other hand, overlearning is prevented by decreasing the max depth, subsample, eta values, and increasing the gamma and minchild weight parameters.

It has been determined that feed forward neural network and bayesian regularized neural network models overlearning in some MIS groups, but the causes of overfitting could not be investigated in detail because the brnn and nnetar functions used in R do not allow changes in the model's layer, initial weight, and epoch number.

In order to cope with overlearning in the LSTM model, different numbers of layers, neurons, drop out values, and weight initializers have been tried. Although the model architects were chosen quite basic due to the small dataset, overfitting was observed in some models. To investigate the reason for this situation, the MIS 0 group, which is overfitting in all machine learning methods, was chosen. In the applied LSTM model, the MIS 0 group was examined by operating different weight starters. The plots of MAPE values of train and validation datasets for each epoch with different weight initializers are in Appendix C. In these graphs, it has been observed that there is no overlearning when the models work with different weight initializers.

In this study, while other machine learning algorithms applied for the MIS 0 group could not handle with overlearning, the problem was solved thanks to the flexibility of the LSTM environment.

According to the model performances of the weight initialize methods in Table 4.4, the most successful model was created with an initializer weight to zero. It outperformed the others among the applied statistical and machine learning methods. The plots of actual and predicted values according to the applied models with weight initializers plots are in Appendix C.

Table 4.4. MAPE and RMSE Values of the LSTM Models Established with Different Weight Initializers

	MAPE	RMSE
Glorot Normal Initializer	36.68	11.03
Variance Scaling Weight Initializer	37.71	10.87
Random Normal Initializer	39.18	10.92
Random Uniform Initializer	38.50	10.94
Initializer Weight to Zero	36.60	11.09

It has been proven in this way that the LSTM model will perform well using a different weight initializer. However, glorot uniform weight initialization was used in LSTM, which was more successful in most MIS groups since the aim of this thesis was to find the model structure that performed the best in all MIS groups on average.

According to the Table 4.1., it can be said that the Random Forest model is the most successful on average, but this may not be a certain result since the MAPE values of the models are very close to each other in some MIS groups. Due to this reason, we consider a formal comparison tool to make a certain decision about the performances of the models. Since the performance measures do not satisfy the assumptions of the paramteric tests, a non-parametric approach is considered. It is known that the non-parametric tests do not assume normal distributions or homogeneity of variance. Because of this reason, it can be applied to classification accuracies, error ratios or any other measure (Demšar, 2006). The Kruskal-Wallis test, which is a non-parametric test, was applied for MAPE values to compare whether the performances of the applied models were different from each other.

The hypotheses for the test are:

H_0: The medians of the MAPE values of the models applied are equal.

H_1: The medians of the MAPE values of the models applied are not equal.

Since the p-value of tests is smaller than the significance level, 0.05, the performances of models are not equal. Moreover, Wilcoxon signed-rank test was utilized to find which models' performances differ from each other.

H_0: True location shift is equal to 0.

H_1: True location shift is not equal to 0.

According to the result of Wilcoxon signed-rank test, there is a difference between the median of MAPE for LSTM-BRNN and XGBoost-BRNN, since the p-values of tests are smaller than 0.05, which is given Table 4.3. Moreover, the median MAPE of RF is different and smaller than all other models.

Table 4.5. The P-Value of Wilcoxon Signed-Rank Test

	ARIMA	ETS	TBATS	SVM	RF	XGBoost	NNETAR	LSTM	BRNN
ARIMA	-	0.39	0.53	0.83	**0**	0.41	0.8	0.39	0.2
ETS	0.39	-	0.8	0.38	**0**	1	0.97	0.88	0.07
TBATS	0.53	0.8	-	0.46	**0**	0.97	0.95	0.76	0.1
SVM	0.83	0.38	0.46	-	**0**	0.24	0.8	0.29	0.32
RF	**0**	**0**	**0**	**0**	-	**0**	**0**	**0**	**0**
XGBoost	0.41	1	0.97	0.24	**0**	-	0.55	0.83	**0.03**
NNETAR	0.8	0.97	0.95	0.8	**0**	0.55	-	0.62	0.44
LSTM	0.39	0.88	0.76	0.29	**0**	0.83	0.62	-	**0.04**
BRNN	0.2	0.07	0.1	0.32	**0**	**0.03**	0.44	**0.04**	-

In addition to this comparison, the MAPE values of the models are shown in Figure 4.3. According to the boxplot, the NNETAR, ETS, TBATS and LSTM have more variations and higher error values compared to other models.

On the other hand, RF being more accurate than the other models has less variation and lower error values. Finally, it can be observed that some models create some big error values in their model performances.

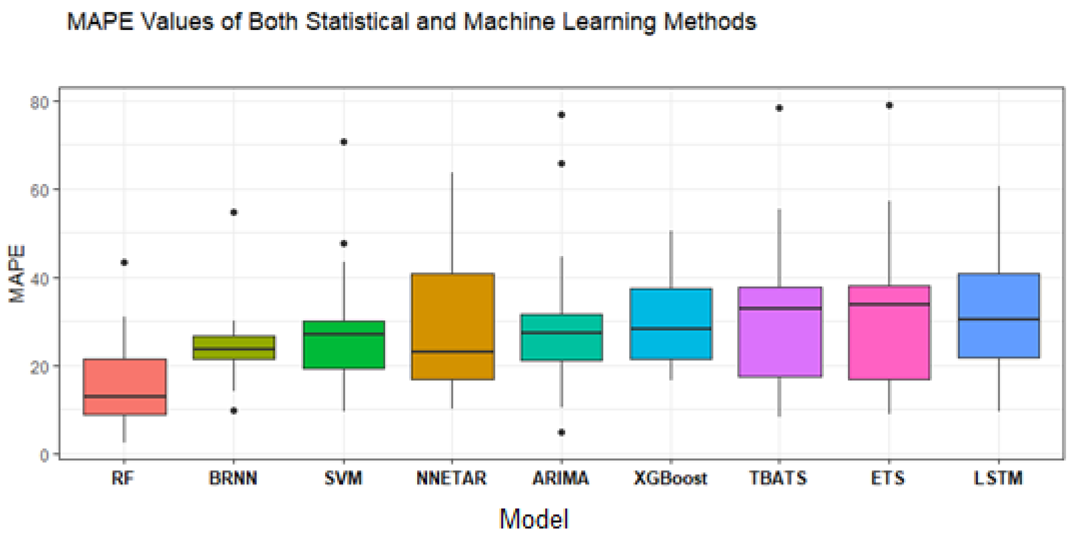

Figure 4.3. The Average of MAPE Values of All Models

4.4.2 Computational Times

Since the training and parameter optimization of all MIS groups are included in a single cell in the structure of machine learning methods, the total running times of the models are given. The *sys.time()* function was utilized in R and ipython-autotime is implemented to measure the time it takes to execute each cell in Python. The calculated computation time for the models includes model training, validating, and forecasting steps since the whole MIS groups are used together for the modeling procedure.

The statistical approaches are implemented in a short time of period compared to machine learning models since the parameter tunning process is not applied to them. For example, the ARIMA model, predicting 15 MIS groups the best among the statistical methods, estimated 25 groups in a total of 2.561 seconds.

The XGBoost model, which predicted all MIS groups in the longest time, took almost 2 hours. The biggest reason for this is that the number of parameters optimized in XGBoost is eight, compared to other models implemented in R, and the parameter range is wider. SVM, RF and NNETAR forecast 25 MIS groups in 24.12115 minutes, 2.025247 minutes, and 19.33765 seconds respectively. Cost and gamma parameters in the SVM model, ntree and mtry numbers in the RF model, and the number of neurons in the function and the lambda parameter required for Box-Cox transformation in the NNETAR model are optimized. Considering the MAPE value, the RF model predicted 14 of 25 groups with the least error and the least computation time. The LSTM model, which is the only method implemented in Python, is forecasted in 4 hours 26 minutes 40 seconds. While constructing the LSTM model, drop out value, the number of units in LSTM layers and the number of units in dense layers, batch size, epochs, learning rate are tuned according to the validation loss value. The BRNN approach predicted 25 groups in 40.117 seconds.

Table 4.6. The Computation Time for Model Training

Model	Computation Time
ARIMA	2.561 secs
ETS	0.594 secs
TBATS	11.874 secs
SVM	24.12115 mins
RF	2.025247 mins
XGBoost	1 hr 42 min 47 secs
NNETAR	19.33765 secs
LSTM	4hr 26 min 40 secs
BRNN	13.76523 secs

CHAPTER 5

CONCLUSION AND FUTURE STUDIES

The chip crisis in the global automotive industry is constantly changing its production plans. The claim rate is frequently used in the analysis of warranty data. However, the fact that the number of production, which directly affects this rate, cannot be predicted due to the crisis, causes uncertainty in business processes. Studies in the literature have generally focused on the claim rate. In the thesis, many of the related researches are introduced at the beginning and the theoretical foundations of the methods used are discussed in detail.

In this study, a 3-month consecutive forecasting of the total number of claims was constructed for 25 different months in services groups by using the warranty data from the automotive industry. Nine different models were utilized: statistical methods ARIMA, ETS, and TBATS and machine learning algorithms SVM, RF, XGBoosting, Feed Forward Neural Network, LSTM, and BRNN.

The prediction performance of the models in the study was compared according to the RMSE and MAPE values, and in addition, not only the prediction accuracy but also the computation time required to predict were compared. With the non-parametric tests applied to compare the performances of the models, it has been observed that the most successful approach to be applied for the company is Random Forest. When the forecast values of the applied models were shared with the company, the forecasting of the number of claims for each MIS group positively affected the business processes of the company. Due to the results of this study, the warranty process can now be carried out from the number of claims instead of the claim rate in the crisis related to production.

Overall, the Random Forest approach has helped to predict models with low mean absolute percentage error. While it outperformed the other models in 14 groups in machine learning methods, it was the second most successful model in 4 groups in total. The number of *ntree* of the most successful models varies between 300 and 1300, and the number of *mtry* varies between 1 and 4.

Considering the trends of the groups, MIS 7-8, MIS 9 -10, MIS 17- 19, have similar patterns over time. The remaining groups' overall claim distribution over time, however, differs from one another. In this study, it was shown that the models that performed well for groups with comparable patterns shared similarities.

First of all, RF has the lowest MAPE value in machine learning algorithms in MIS 7 and MIS 8 groups, while BRNN and NNETAR are the second most successful models. Although the best parameter values used for these groups are close to each other for BRNN and NNETAR, the *ntree* and *mtry* values used for RF are different from each other. In statistical methods, while ARIMA(0,1,0) which is a random walk model for MIS 7 performed better, TBATS model was the best performing model for MIS 8, which may be due to the higher non linearity of MIS 8.

Besides that, machine learning methods, RF is the best model for MIS 9 and MIS 10, BRNN and LSTM, respectively, are the second best models for these MIS groups. The statistical method indicated that the least error for both groups was the ARIMA(0,1,0) model. Finally, the statistical methods, ETS and TBATS, models outperformed ML models for the MIS 17, MIS 19 and MIS 23 groups.

Considering the statistical methods, the ARIMA model, which is superior to other statistical models in estimating the future values of the series, forecast 15 groups out of 25 groups with the lowest error value. While TBATS showed the lowest error value for 7 groups, ETS was the most successful model in 4 groups.

The change of trend and seasonality patterns within the dataset is not important for a good model, because models with good performance also incorporate this mode of change into the model (Hyndman *et al.*, 2007). However, the dataset utilized in this thesis is noisy and does not contain enough observations to understand the pattern of the models. Therefore, the metric values measuring accuracy were quite high in some groups.

ARIMA, ETS, TBATS, and NNETAR models mean absolute percentage value was more than 50% in 2 groups out of 25 groups. The BRNN, XGBoost, and SVM models on the other hand, have more than 50% MAPE in one group. While the MAPE values of the RF model did not exceed 50% in any group, the error rate was higher than 50% in 3 of 25 groups in the LSTM model. Statistical methods forecast in MIS groups 17-19-23 with half the MAPE value of machine learning methods.

Although the Bayesian Weight Optimization technique of the LSTM model successfully optimized the parameters, there were overlearning problems in some MIS groups. To avoid this, the model architecture was chosen small and the range of dropout values was kept high. The MAPE metrics of the models implemented using various weight initializers were very different from each other. Since the study suggested the best model, glorot uniform weight initializer was used, which helps to create a model with the least error value. This study has shown that the use of weight initializer, dropout values , and small architecture affects LSTM model performance in small datasets.

The *brnn* and *nnetar* functions used in R are very limited in use as they do not allow changing the number of layers and weights. At the same time, there is no learning rate information in these functions, and there is no epoch information in the *nnetar* function. This situation, which caused a limited study area, prevented the investigation and prevention of overfitting in some MIS groups.

Finally, Table 4.1. and Table 4.2. show the MAPE and RMSE values for each group of the nine models used in this study. It has been observed that the models with the highest prediction accuracy are generally selected from the machine learning models. Specifically, it can be said that the RF may be the best option for forecasting due to having the overall best performance concerning error measures and requiring a short time.

However, we also know that the performance of statistical and machine learning approaches can be improved by increasing the number of observations in the MIS groups utilized to further improve the performances obtained from this study. By giving the parameter tuning section more contemplated in future investigations, some convergence or training issues can be resolved.

In the modeling part, different models that have been proven as successful in time series forecasting such as Recurrent Neural Network and Convolutional Neural Networks can be applied. Although the univariate time series is applied, the inclusion of many variables, such as the wear rate of the product, defective parts, production period, precipitation, and climate used in addition to the factors affecting the number of claims may affect the performance of the model.

Finally, the hybrid approach, where ML models and statistical approaches are combined, may also have an impact on the performance of the model.

REFERENCES

Akbarov, A., & Wu, S. (2012). Warranty claim forecasting based on weighted maximum likelihood estimation. *Quality and Reliability Engineering International*, *28*(6), 663–669. https://doi.org/10.1002/qre.1399.

Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., & Baskurt, A. (2011). Sequential deep learning for human action recognition. *Lecture Notes in Computer Science*, 29–39. https://doi.org/10.1007/978-3-642-25446-8_4.

Beaumont, C., Makridakis, S., Wheelwright, S. C., & McGee, V. E. (1984). Forecasting: Methods and applications. *The Journal of the Operational Research Society, 35*(1), 79. https://doi.org/https://doi.org/10.2307/2581936

Bilgili, M., Arslan, N., Şekertekin, A., & Yaşar, A. (2022). Application of long short-term memory (LSTM) neural network based on deep learning for electricity energy consumption forecasting. *Turkish Journal of Electrical Engineering and Computer Sciences*, *30*(1). https://doi.org/10.3906/elk-2011-14

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory - COLT '92*. https://doi.org/10.1145/130385.130401

Bouzerdoum, M., Mellit, A., & Massi Pavan, A. (2013). A hybrid model (SARIMA–SVM) for short-term power forecasting of a small-scale grid-connected photovoltaic plant. *Solar Energy*, *98*, 226–235. https://doi.org/10.1016/j.solener.2013.10.002

Box, G. E., Jenkins, G. M., & Reinsel, G. (1970). Time series analysis: Forecasting and control Holden-day San Francisco. *BoxTime Series Analysis: Forecasting and Control Holden Day1970*.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/a:1010933404324.

Brockett, P. L., Cooper, W. W., Golden, L. L., & Pitaktong, U. (1994). A neural network method for obtaining an early warning of insurer insolvency. *The Journal of Risk and Insurance, 61*(3), 402. https://doi.org/10.2307/253568

Brown, R. G. (1959). Statistical forecasting for inventory control. New York: McGraw-Hill.

Brownlee, J. (2020). *Data preparation for machine learning: Data cleaning, feature selection, and data transforms in Python*. Machine Learning Mastery.

Burden, F., & Winkler, D. (2008). Bayesian regularization of neural networks. *Methods in Molecular Biology (Clifton, N.J.), 458,* 25–44. https://doi.org/10.1007/978-1-60327-101-1_3

Chandriah, K. K., & Naraganahalli, R. V. (2021). RNN / LSTM with modified Adam optimizer in deep learning approach for automobile spare parts demand forecasting. *Multimedia Tools and Applications, 80*(17). https://doi.org/10.1007/s11042-021-10913-0

Charles, W., & Chase, J. (2013). Measuring forecast performance. *Demand-Driven Forecasting: A Structured Approach to Forecasting,* 103–124. https://doi.org/10.1002/9781118691861.ch4

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16),* 785–794. https://doi.org/10.1145/2939672.2939785

Chughtai, F., & Zayed, T. (2008). Infrastructure condition prediction models for sustainable sewer pipelines. *Journal of Performance of Constructed Facilities, 22*(5), 333–341. https://doi.org/10.1061/(asce)0887-3828(2008)22:5(333)

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/bf00994018

Dan Foresee, F., & Hagan, M. (1997). Gauss-Newton approximation to bayesian learning. *Proceedings of International Conference on Neural Networks (ICNN'97), 3*, 1930–1935. https://doi.org/10.1109/icnn.1997.614194

De Livera, A. M., Hyndman, R. J., & Snyder, R. D. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, *106*(496), 1513–1527. https://doi.org/10.1198/jasa.2011.tm09771

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7, 1-30.

Fredette, M., & Lawless, J. F. (2007). Finite-horizon prediction of recurrent events, with application to forecasts of warranty claims. *Technometrics*, *49*(1), 66–80. https://doi.org/10.1198/004017006000000390

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5). https://doi.org/10.1214/aos/1013203451

Geary, R. C., Wold, H., & Whittle, P. (1956). A study in the analysis of stationary time series. *The Economic Journal*, *66*(262), 327–330. https://doi.org/10.2307/2227977

Géron, A. (2019). *Hands-on machine learning with scikit-Learn, keras, and tensorflow: Concepts, tools, and techniques to build intelligent systems* (2nd ed.). O'Reilly Media, Inc.

Gos, M., Krzyszczak, J., Baranowski, P., Murat, M., & Malinowska, I. (2020). 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, 281*, 107827. https://doi.org/10.1016/j.agrformet.2019.107827

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5–6), 602–610. https://doi.org/10.1016/j.neunet.2005.06.042

Guelman, L. (2012). Gradient boosting trees for auto insurance loss cost modeling and prediction. *Expert Systems with Applications*, *39*(3), 3659–3667. https://doi.org/10.1016/j.eswa.2011.09.058

Hardesty, L. (2017, April 14). *Explained: Neural networks*. MIT News | Massachusetts Institute of Technology. https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, *20*(1), 5–10. https://doi.org/10.1016/j.ijforecast.2003.09.015

Hu, X. J., & Lawless, J. F. (1997). Pseudolikelihood estimation in a class of problems with response-related missing covariates. *Canadian Journal of Statistics*, *25*(2), 125–142. https://doi.org/10.2307/3315727

Hu, X. J., Lawless, J. F., & Suzuki, K. (1998). Nonparametric estimation of a lifetime distribution when censoring times are missing. *Technometrics*, *40*(1), 3–13. https://doi.org/10.1080/00401706.1998.10485477

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and practice* (3rd ed.). Otexts.

Hyndman, R. J., & Kostenko, A. V. (2007). Minimum sample size requirements for seasonal forecasting models. *Foresight: The International Journal of Applied Forecasting*, *6*, 12–15. https://ideas.repec.org/a/for/ijafaa/y2007i6p12-15.html

James, G., Witten, D., & Hastie, T. (2021). *An introduction to statistical learning: with applications in R* (2nd ed.). Springer. https://doi.org/10.1007/978-1-0716-1418-1

Kalbfleisch, J. D., & Lawless, J. F. (1988). Estimation of reliability in field-performance studies. *Technometrics*, *30*(4), 365. https://doi.org/10.2307/1269797

Kim, J., Hong, J., & Park, H. (2018). Prospects of deep learning for medical imaging. *Precision and Future Medicine*, *2*(2), 37–52. https://doi.org/10.23838/pfm.2018.00030

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. https://doi.org/ https://doi.org/10.48550/arXiv.1412.6980

Krenker, A., Bešter, J., & Kos, A. (2011). *Introduction to the artificial neural networks*. In (Ed.), Artificial Neural Networks - Methodological Advances and Biomedical Applications. IntechOpen. https://doi.org/10.5772/15751

Lawless, J. F., Kalbfleisch, J. D., & Blumenthal, S. (1992). Some issues in the collection and analysis of field reliability data. *Survival Analysis: State of the Art*, 141–152. https://doi.org/10.1007/978-94-015-7983-4_9

Lawless, J. F., & Nadeau, C. (1995). Some simple robust methods for the analysis of recurrent events. *Technometrics*, *37*(2), 158–168. https://doi.org/10.1080/00401706.1995.10484300

Lewis, N. D. (2016). *Deep time series forecasting with python: An intuitive introduction to deep learning for applied time series modeling*. CreateSpace Independent Publishing Platform.

Li, D., Ling, S., & Tong, H. (2012). On moving-average models with feedback. *Beroulli*, *18*(2). https://doi.org/10.3150/11-bej352

Li, W., Yin, Y., Quan, X., & Zhang, H. (2019). Gene expression value prediction based on XGBoost algorithm. *Frontiers in Genetics*, *10*. https://doi.org/10.3389/fgene.2019.01077

Liaw, A., & Wiener, M. (2002). Classification and regression by randomforest. *R News*, *2*(3), 18–22. http://CRAN.R-project.org/doc/Rnews/

Majeske, K. D. (2007). A non-homogeneous poisson process predictive model for automobile warranty claims. *Reliability Engineering & System Safety*, *92*(2), 243–251. https://doi.org/10.1016/j.ress.2005.12.004

Marshall, S. E., & Chukova, S. (2009). On analysing warranty data from repairable items. *Quality and Reliability Engineering International, 26*(1), 43–52. https://doi.org/10.1002/qre.1032

Mayer, H., Gomez, F., Wierstra, D., Nagy, I., Knoll, A., & Schmidhuber, J. (2006). A system for robotic heart surgery that learns to tie knots using recurrent neural networks. *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 543-548. https://doi.org/10.1109/iros.2006.282190

Murthy, D. (2004). Product warranty logistics: Issues and challenges. *European Journal of Operational Research*, *156*(1), 110–126. https://doi.org/10.1016/s0377-2217(02)00912-8

Naing, W. Y. N., & Htike, Z. Z. (2015). Forecasting of monthly temperature variations using random forests. *ARPN Journal of Engineering and Applied Sciences*, *10*(21), 10109–10112.

Nguyen, D., & Widrow, B. (1990). Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, *10*(3), 18–23. https://doi.org/10.1109/37.55119

Nielsen, M. A. (2015). *Neural networks and deep learning* (Vol. 25) [E-book]. Determination press. http://neuralnetworksanddeeplearning.com

Ozdemir, O., & Yozgatlıgil, C. (2020). *Performance comparison of machine learning methods and traditional time series methods for forecasting* (Unpublished master's dissertation), Middle East Technical University Natural and Applied Sciences, Ankara, Turkey

Ozel, T., & Davim, P. J. (2009). *Intelligent Machining* (1st ed.). Wiley-ISTE.

Pai, P.-F., Lin, K.-P., Lin, C.-S., & Chang, P.-T. (2010). Time series forecasting by a seasonal support vector regression model. *Expert Systems with Applications, 37*(6), 4261–4265. https://doi.org/10.1016/j.eswa.2009.11.076

Qiu, X., Zhang, L., Nagaratnam Suganthan, P., & Amaratunga, G. A. J. (2017). Oblique random forest ensemble via least square estimation for time series forecasting. *Information Sciences, 420,* 249–262. https://doi.org/10.1016/j.ins.2017.08.060

Rai, B., & Singh, N. (2005). Forecasting warranty performance in the presence of the ‘maturing data’ phenomenon. *International Journal of Systems Science, 36*(7), 381–394. https://doi.org/10.1080/00207720500139930

Ribeiro, M. H., & Coelho, L. (2020). Ensemble approach based on bagging, boosting and stacking for short-term prediction in agribusiness time series. *Applied Soft Computing, 86,* 105837. https://doi.org/10.1016/j.asoc.2019.105837

Samsudin, R., Shabri, A., & Saad, P. (2010). A comparison of time series forecasting using support vector machine and artificial neural network model. *Journal of Applied Sciences, 10*(11), 950–958. https://doi.org/10.3923/jas.2010.950.958

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning, 5*(2), 197–227. https://doi.org/10.1007/bf00116037

Schmidhuber, J., Wierstra, D., & Gomez, F. J. (2005, July). Evolino: Hybrid neuroevolution / optimal linear search for sequence prediction. *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 853–858. https://doi.org/10.5555/1642293.1642430

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2019). The performance of LSTM and BiLSTM in forecasting time series. *2019 IEEE International Conference on Big Data (Big Data),* 3285–3292. https://doi.org/10.1109/bigdata47090.2019.9005997

Svetunkov, I. (2022a, August 4). *Forecasting and analytics with ADAM*. Forecasting and Analytics with ADAM. https://openforecast.org/adam/

Von Oertzen, T., & Boker, S. M. (2009). Time delay embedding increases estimation precision of models of intraindividual variability. *Psychometrika, 75*(1), 158–175. https://doi.org/10.1007/s11336-009-9137-9

Wang, X., & Xie, W. (2018). Two-dimensional warranty: A literature review. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 232*(3), 284–307. https://doi.org/10.1177/1748006x17742776

Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management Science, 6(*3), 324–342. https://doi.org/10.1287/mnsc.6.3.324

Wu, S. (2012). Warranty data analysis: A review. *Quality and reliability engineering international, 28*(8), 795–805. https://doi.org/10.1002/qre.1282

Wu, S., & Akbarov, A. (2011). Support vector regression for warranty claim forecasting. *European Journal of Operational Research*, *213*(1), 196–204. https://doi.org/10.1016/j.ejor.2011.03.009

Wu, X., Zhang, C., & Du, W. (2021). An analysis on the crisis of "Chips shortage" in automobile industry ——based on the double influence of COVID-19 and trade friction. *Journal of Physics: Conference Series*. 2021 3rd International Conference on Electronic Engineering and Informatics (EEI 2021), Dali, China. https://doi.org/10.1088/1742-6596/1971/1/012100

Yan, S. (2017). *Understanding lstm and its diagrams.* Medium. Retrieved October 23, 2021, from https://blog.mlreview.com/understanding-lstm-andits-diagrams-37e2f46f1714.

Yang, K., & Cekecek, E. (2004). Design vulnerability analysis and design improvement by using warranty data. *Quality and Reliability Engineering International, 20*(2), 121–133. https://doi.org/10.1002/qre.617

Yu, P.-S., Yang, T.-C., Chen, S.-Y., Kuo, C.-M., & Tseng, H.-W. (2017). Comparison of random forests and support vector machine for real-time radar-derived rainfall forecasting. *Journal of Hydrology, 552*, 92–104. https://doi.org/10.1016/j.jhydrol.2017.06.020

Yu, W., Guan, G., Li, J., Wang, Q., Xie, X., Zhang, Y., Huang, Y., Yu, X., & Cui, C. (2021). Claim amount forecasting and pricing of automobile insurance based on the BP neural network. *Complexity, 2021*, 1–17. https://doi.org/10.1155/2021/6616121

Yule, G. U. (1926). Why do we sometimes get nonsense-correlations between time-series? A study in sampling and the nature of time-series. *Journal of the Royal Statistical Society*, *89*(1), 1-64. https://doi.org/10.2307/2341482

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing, 50,* 159–175. https://doi.org/10.1016/s0925-2312(01)00702-0

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, *14*(1), 35–62. https://doi.org/https://doi.org/10.1016/S0169-2070(97)00044-7

Zhang, L., Bian, W., Qu, W., Tuo, L., & Wang, Y. (2021). Time series forecast of sales volume based on XGBoost. *Journal of Physics: Conference Series, 1873*(1), 012067. https://doi.org/10.1088/1742-6596/1873/1/012067

APPENDICES

A. The ACF-PACF Plot of MIS 6

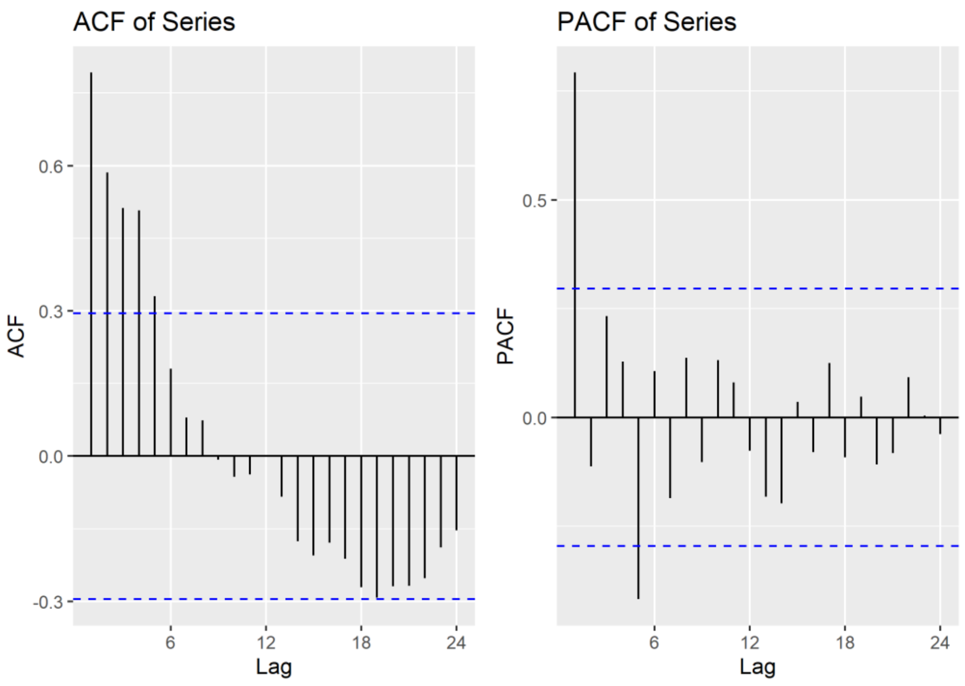

Figure A.1. The ACF- PACF Plot of MIS 6

B. The Tuned Hyperparameters of Machine Learning Methods

Table B.1. The Tuned Hyperparameters of Machine Learning Methods for MIS 0

Model	Tuned Parameters
RF	ntree:400, mtry:4
SVM	cost:0.00003, gamma:256
XGBoost	eta:0.025, max depth:3, gamma:0.9, colsample by tree:0, min child weight:12, subsample:0.78
NNETAR	neuron:1, non-seasonal lag:1
BRNN	alpha:2.14, beta:3.83, gamma:3.86, Ed:4.33, Ew:0.9, p:4, n:37, neurons:2, epoch:19
LSTM	lstm layer(1st):5, dropout value:0.2, dense layer(1st):3, learning rate:0.1 epoch:10

Table B.2. The Tuned Hyperparameters of Machine Learning Methods for MIS 1

Model	Tuned Parameters
RF	ntree:200, mtry:2
SVM	cost:0.00024, gamma:128
XGBoost	eta:0.02, max depth:3, gamma:0.9, colsample by tree:0, min child weight:10, subsample:0.78
NNETAR	neuron:1, non-seasonal lag:1
BRNN	alpha:1.26, beta:4.49, gamma:4.64, Ed:3.61, Ew:1.84, p:4, n:37, neurons:6, epoch:24
LSTM	lstm layer(1st):1, dropout value:0.4, dense layer(1st):2, learning rate:0.01 epoch:14

Table B.3. The Tuned Hyperparameters of Machine Learning Methods for MIS 2

Model	Tuned Parameters
RF	ntree:400, mtry:1
SVM	cost:0.00003, gamma:2048
XGBoost	eta:0.025, max depth:2, gamma:0.7, colsample by tree:0, min child weight:10, subsample:0.84
NNETAR	neuron:5, non-seasonal lag:1
BRNN	alpha:1.25, beta:7.32, gamma:5.48, Ed:2.15, Ew:2.19, p:4, n:37, neurons:8, epoch:23
LSTM	lstm layer(1st):4, dropout value:0.2, dense layer(1st):1, learning rate:0.01 epoch:10

Table B.4. The Tuned Hyperparameters of Machine Learning Methods for MIS 3

Model	Tuned Parameters
RF	ntree:1100, mtry:4
SVM	cost:0.00003, gamma:4096
XGBoost	eta:0.025, max depth:3, gamma:0.7, colsample by tree:0, min child weight:12, subsample:0.84
NNETAR	neuron:5, non-seasonal lag:1
BRNN	alpha:1.54, beta:4.38, gamma:4.49, Ed:3.71, Ew:1.46, p:4, n:37, neurons:10, epoch:36
LSTM	lstm layer(1st):3, dropout value:0.2, dense layer(1st):3, learning rate:0.1 epoch:21

Table B.5. The Tuned Hyperparameters of Machine Learning Methods for MIS 4

Model	Tuned Parameters
RF	ntree:200, mtry:3
SVM	cost:0.00003, gamma:512
XGBoost	eta:0.025, max depth:1, gamma:0.8, colsample by tree:0, min child weight:15, subsample:0.92
NNETAR	neuron:2, non-seasonal lag:1
BRNN	alpha:2.07, beta:6.32, gamma:4.43, Ed:2.58, Ew:1.07, p:4, n:37, neurons:2, epoch:15
LSTM	lstm layer(1st):5, dropout value:0.2, dense layer(1st):1, learning rate:0.1 epoch:14

Table B.6. The Tuned Hyperparameters of Machine Learning Methods for MIS 5

Model	Tuned Parameters
RF	ntree:1300, mtry:1
SVM	cost:0.00098, gamma:512
XGBoost	eta:0.025, max depth:3, gamma:0.7, colsample by tree:0, min child weight:15, subsample:0.84
NNETAR	neuron:2, non-seasonal lag:2
BRNN	alpha:0.83, beta:12.17, gamma:4.97, Ed:1.32, Ew:3, p:4, n:37, neurons:1, epoch:12
LSTM	lstm layer(1st):3, dropout value:0.2, dense layer(1st):3, learning rate:0.01 epoch:14

Table B.7. The Tuned Hyperparameters of Machine Learning Methods for MIS 6

Model	Tuned Parameters
RF	ntree:1100, mtry:3
SVM	cost:0.00098, gamma:256
XGBoost	eta:0.025, max depth:3, gamma:0.7, colsample by tree:0, min child weight:10, subsample:0.78
NNETAR	neuron:5, non-seasonal lag:1
BRNN	alpha:1.11, beta:10.81, gamma:5.05, Ed:1.48, Ew:2.28, p:4, n:37, neurons:1, epoch:16
LSTM	lstm layer(1st):2, dropout value:0.2, dense layer(1st):4, learning rate:0.01 epoch:13

Table B.8. The Tuned Hyperparameters of Machine Learning Methods for MIS 7

Model	Tuned Parameters
RF	ntree:300, mtry:3
SVM	cost:0.00195, gamma:32
XGBoost	eta:0.02, max depth:2, gamma:0.9, colsample by tree:0, min child weight:15, subsample:0.78
NNETAR	neuron:1, non-seasonal lag:1
BRNN	alpha:1.09, beta:6.71, gamma:4.99, Ed:2.39, Ew:2.29, p:4, n:37, neurons:1, epoch:17
LSTM	lstm layer(1st):2, dropout value:0.2, dense layer(1st):4, learning rate:0.1 epoch:33

Table B.9. The Tuned Hyperparameters of Machine Learning Methods for MIS 8

Model	Tuned Parameters
RF	ntree:300, mtry:4
SVM	cost:0.125, gamma:4
XGBoost	eta:0.02, max depth:3, gamma:0.8, colsample by tree:0, min child weight:15, subsample:0.92
NNETAR	neuron:4, non-seasonal lag:5
BRNN	alpha:1.15, beta:7.58, gamma:4.94, Ed:2.12, Ew:2.14, p:4, n:37, neurons:1, epoch:16
LSTM	lstm layer(1st):3, dropout value:0.7, dense layer(1st):1, learning rate:0.01 epoch:43

Table B.10. The Tuned Hyperparameters of Machine Learning Methods for MIS 9

Model	Tuned Parameters
RF	ntree:600, mtry:3
SVM	cost:0.00049, gamma:64
XGBoost	eta:0.02, max depth:3, gamma:0.8, colsample by tree:0, min child weight:15, subsample:0.78
NNETAR	neuron:2, non-seasonal lag:1
BRNN	alpha:1.15, beta:8.73, gamma:4.99, Ed:1.83, Ew:2.16, p:4, n:37, neurons:1, epoch:18
LSTM	lstm layer(1st):3, dropout value:0.4, dense layer(1st):3, learning rate:0.01 epoch:14

Table B.11. The Tuned Hyperparameters of Machine Learning Methods for MIS 10

Model	Tuned Parameters
RF	ntree:900, mtry:2
SVM	cost:0.0625, gamma:4
XGBoost	eta:0.02, max depth:2, gamma:0.8, colsample by tree:0, min child weight:12, subsample:0.78
NNETAR	neuron:2, non-seasonal lag:1
BRNN	alpha:1.09, beta:6.69, gamma:4.84, Ed:2.4, Ew:2.22, p:4, n:37, neurons:1, epoch:14
LSTM	lstm layer(1st):1, dropout value:0.6, dense layer(1st):2, learning rate:0.1 epoch:10

Table B.12. The Tuned Hyperparameters of Machine Learning Methods for MIS 11

Model	Tuned Parameters
RF	ntree:1100, mtry:2
SVM	cost:0.0625, gamma:1
XGBoost	eta:0.02, max depth:3, gamma:0.8, colsample by tree:0, min child weight:12, subsample:0.78
NNETAR	neuron:4, non-seasonal lag:1
BRNN	alpha:1.24, beta:7.29, gamma:4.92, Ed:2.2, Ew:1.99, p:4, n:37, neurons:1, epoch:15
LSTM	lstm layer(1st):3, dropout value:0.2, dense layer(1st):2, learning rate:0.01 epoch:30

Table B.13. The Tuned Hyperparameters of Machine Learning Methods for MIS 12

Model	Tuned Parameters
RF	ntree:500, mtry:2
SVM	cost:0.00024, gamma:512
XGBoost	eta:0.02, max depth:2, gamma:0.7, colsample by tree:0, min child weight:15, subsample:0.92
NNETAR	neuron:1, non-seasonal lag:1
BRNN	alpha:1.24, beta:7.44, gamma:4.98, Ed:2.15, Ew:2.01, p:4, n:37, neurons:1, epoch:18
LSTM	lstm layer(1st):1, dropout value:0.2, dense layer(1st):3, learning rate:0.01 epoch:31

Table B.14. The Tuned Hyperparameters of Machine Learning Methods for MIS 13

Model	Tuned Parameters
RF	ntree:300, mtry:1
SVM	cost:0.00003, gamma:4096
XGBoost	eta:0.02, max depth:2, gamma:0.8, colsample by tree:0, min child weight:12, subsample:0.78
NNETAR	neuron:4, non-seasonal lag:3
BRNN	alpha:1.01, beta:8.71, gamma:5.11, Ed:1.83, Ew:2.52, p:4, n:37, neurons:1, epoch:14
LSTM	lstm layer(1st):1, dropout value:0.2, dense layer(1st):4, learning rate:0.01 epoch:14

Table B.15. The Tuned Hyperparameters of Machine Learning Methods for MIS 14

Model	Tuned Parameters
RF	ntree:1400, mtry:2
SVM	cost:0.00012, gamma:1024
XGBoost	eta:0.02, max depth:1, gamma:0.9, colsample by tree:0, min child weight:15, subsample:0.92
NNETAR	neuron:2, non-seasonal lag:1
BRNN	alpha:1.47, beta:8.14, gamma:4.6, Ed:1.99, Ew:1.56, p:4, n:37, neurons:10, epoch:51
LSTM	lstm layer(1st):3, dropout value:0.4, dense layer(1st):1, learning rate:0.01 epoch:25

Table B.16. The Tuned Hyperparameters of Machine Learning Methods for MIS 15

Model	Tuned Parameters
RF	ntree:500, mtry:1
SVM	cost:0.0625, gamma:2
XGBoost	eta:0.02, max depth:3, gamma:0.8, colsample by tree:0, min child weight:12, subsample:0.78
NNETAR	neuron:4, non-seasonal lag:1
BRNN	alpha:1.25, beta:7.75, gamma:5.04, Ed:2.06, Ew:2.01, p:4, n:37, neurons:1, epoch:17
LSTM	lstm layer(1st):4, dropout value:0.6, dense layer(1st):4, learning rate:0.01 epoch:19

Table B.17. The Tuned Hyperparameters of Machine Learning Methods for MIS 16

Model	Tuned Parameters
RF	ntree:600, mtry:1
SVM	cost:0.00006, gamma:512
XGBoost	eta:0.025, max depth:1, gamma:0.8, colsample by tree:0, min child weight:10, subsample:0.84
NNETAR	neuron:4, non-seasonal lag:5
BRNN	alpha:1.48, beta:4.68, gamma:5, Ed:3.42, Ew:1.69, p:4, n:37, neurons:1, epoch:15
LSTM	lstm layer(1st):3, dropout value:0.2, dense layer(1st):4, learning rate:0.01 epoch:14

Table B.18. The Tuned Hyperparameters of Machine Learning Methods for MIS 17

Model	Tuned Parameters
RF	ntree:400, mtry:1
SVM	cost:0.00003, gamma:4096
XGBoost	eta:0.025, max depth:1, gamma:0.9, colsample by tree:0, min child weight:12, subsample:0.78
NNETAR	neuron:1, non-seasonal lag:1
BRNN	alpha:0.94, beta:5.69, gamma:5.91, Ed:2.73, Ew:3.13, p:4, n:37, neurons:10, epoch:44
LSTM	lstm layer(1st):4, dropout value:0.2, dense layer(1st):2, learning rate:0.01 epoch:48

Table B.19. The Tuned Hyperparameters of Machine Learning Methods for MIS 18

Model	Tuned Parameters
RF	ntree:400, mtry:1
SVM	cost:0.00006, gamma:8192
XGBoost	eta:0.001, max depth:1, gamma:0.7, colsample by tree:0, min child weight:15, subsample:0.92
NNETAR	neuron:3, non-seasonal lag:4
BRNN	alpha:0.91, beta:5.74, gamma:5.04, Ed:2.78, Ew:2.76, p:4, n:37, neurons:1, epoch:34
LSTM	lstm layer(1st):4, dropout value:0.2, dense layer(1st):1, learning rate:0.01 epoch:54

Table B.20. The Tuned Hyperparameters of Machine Learning Methods for MIS 19

Model	Tuned Parameters
RF	ntree:1300, mtry:3
SVM	cost:0.00195, gamma:64
XGBoost	eta:0.001, max depth:2, gamma:0.7, colsample by tree:0, min child weight:10, subsample:0.78
NNETAR	neuron:5, non-seasonal lag:10
BRNN	alpha:0.94, beta:4.53, gamma:6.61, Ed:3.35, Ew:3.53, p:4, n:37, neurons:5, epoch:54
LSTM	lstm layer(1st):4, dropout value:0.2, dense layer(1st):3, learning rate:0.01 epoch:15

Table B.21. The Tuned Hyperparameters of Machine Learning Methods for MIS 20

Model	Tuned Parameters
RF	ntree:300, mtry:1
SVM	cost:0.125, gamma:2
XGBoost	eta:0.001, max depth:1, gamma:0.8, colsample by tree:0, min child weight:12, subsample:0.92
NNETAR	neuron:5, non-seasonal lag:6
BRNN	alpha:0.8, beta:5.01, gamma:7.98, Ed:2.9, Ew:4.97, p:4, n:37, neurons:10, epoch:30
LSTM	lstm layer(1st):5, dropout value:0.2, dense layer(1st):4, learning rate:0.01 epoch:13

Table B.22. The Tuned Hyperparameters of Machine Learning Methods for MIS 21

Model	Tuned Parameters
RF	ntree:700, mtry:1
SVM	cost:0.0625, gamma:0.5
XGBoost	eta:0.001, max depth:1, gamma:0.8, colsample by tree:0, min child weight:12, subsample:0.92
NNETAR	neuron:0, non-seasonal lag:1
BRNN	alpha:1.87, beta:4.86, gamma:4.33, Ed:3.36, Ew:1.16, p:4, n:37, neurons:10, epoch:70
LSTM	lstm layer(1st):3, dropout value:0.2, dense layer(1st):5, learning rate:0.01 epoch:99

Table B.23. The Tuned Hyperparameters of Machine Learning Methods for MIS 22

Model	Tuned Parameters
RF	ntree:200, mtry:1
SVM	cost:0.0625, gamma:2
XGBoost	eta:0.001, max depth:3, gamma:0.8, colsample by tree:0, min child weight:10, subsample:0.84
NNETAR	neuron:1, non-seasonal lag:1
BRNN	alpha:1.11, beta:6.64, gamma:4.99, Ed:2.41, Ew:2.24, p:4, n:37, neurons:1, epoch:17
LSTM	lstm layer(1st):4, dropout value:0.2, dense layer(1st):5, learning rate:0.01 epoch:21

Table B.24. The Tuned Hyperparameters of Machine Learning Methods for MIS 23

Model	Tuned Parameters
RF	ntree:500, mtry:1
SVM	cost:0.03125, gamma:4
XGBoost	eta:0.001, max depth:2, gamma:0.9, colsample by tree:0, min child weight:10, subsample:0.84
NNETAR	neuron:1, non-seasonal lag:1
BRNN	alpha:0.77, beta:8.46, gamma:9.96, Ed:1.6, Ew:6.46, p:4, n:37, neurons:4, epoch:37
LSTM	lstm layer(1st):1, dropout value:0.2, dense layer(1st):5, learning rate:0.01 epoch:46

Table B.25. The Tuned Hyperparameters of Machine Learning Methods for MIS 24

Model	Tuned Parameters
RF	ntree:300, mtry:1
SVM	cost:0.0625, gamma:1
XGBoost	eta:0.001, max depth:2, gamma:0.9, colsample by tree:0, min child weight:12, subsample:0.92
NNETAR	neuron:2, non-seasonal lag:1
BRNN	alpha:0.38, beta:25.4, gamma:21.12, Ed:0.31, Ew:28.04, p:4, n:37, neurons:6, epoch:74
LSTM	lstm layer(1st):5, dropout value:0.2, dense layer(1st):1, learning rate:0.01 epoch:25

C. LSTM Models Implemented with Weight Initialization Techniques for MIS 0

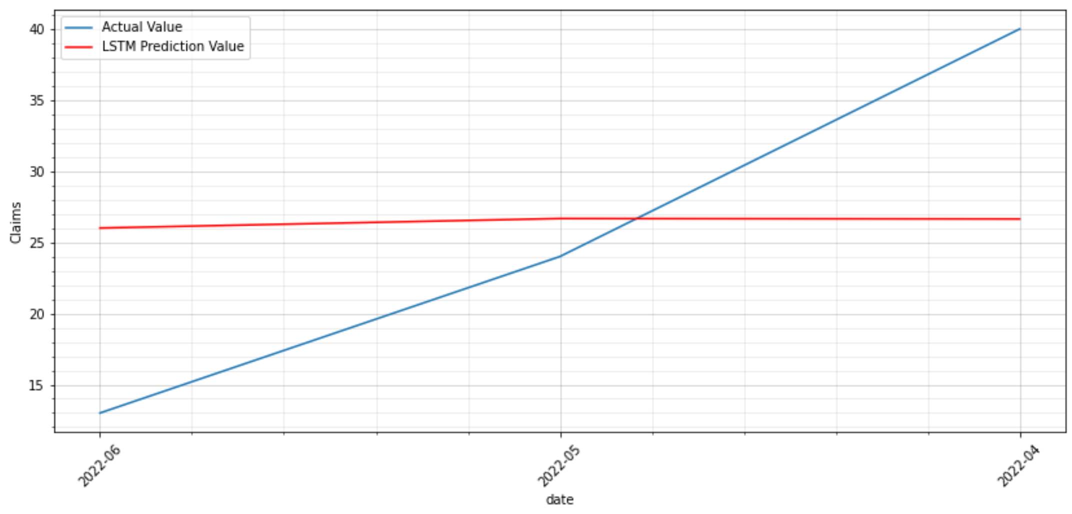

Figure C.1. The Plot of Actual and Predicted Claims with Variance Scaling Weight Initialization

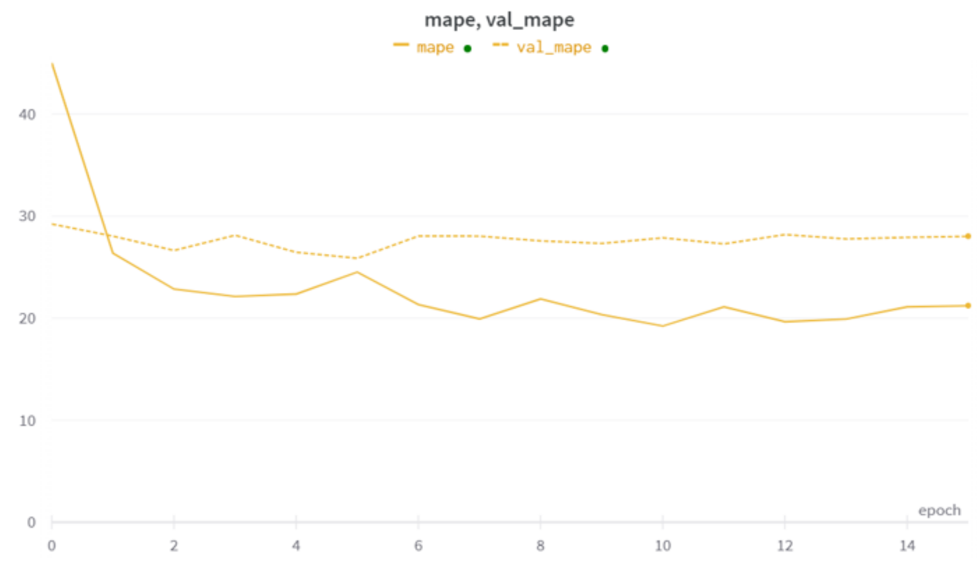

Figure C.2. The Plot of MAPE Values of Train and Validation Sets with Variance Scaling Weight Initialization

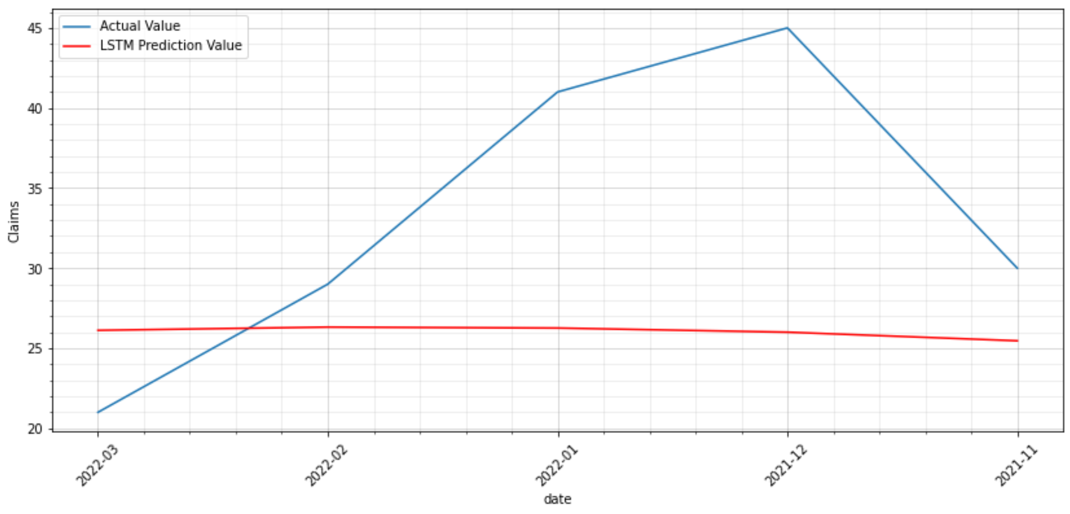

Figure C.3. The Plot of Actual and Predicted Claims with Variance Scaling Weight Initialization on Validation Dataset

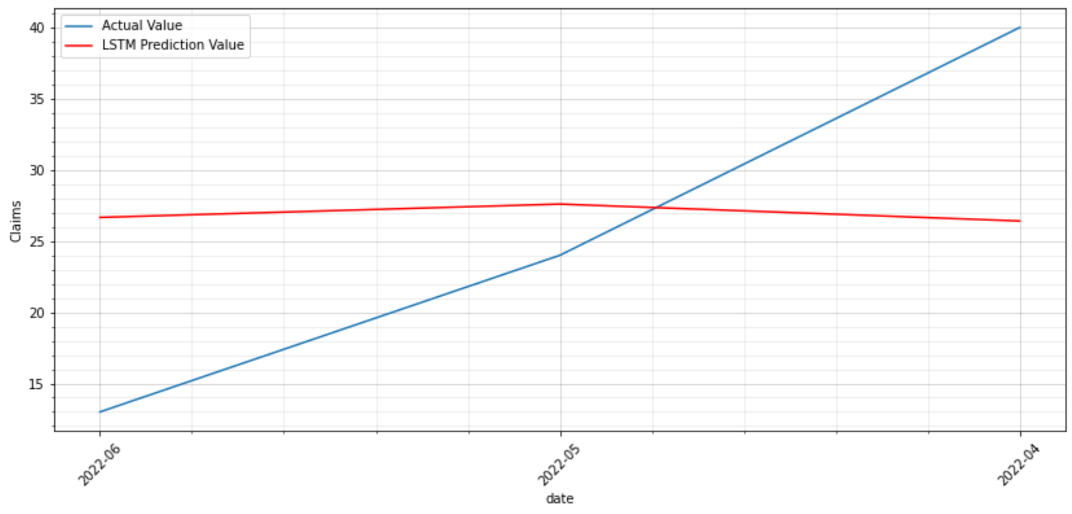

Figure C.4. The Plot of Actual and Predicted Claims with Random Normal Weight Initialization

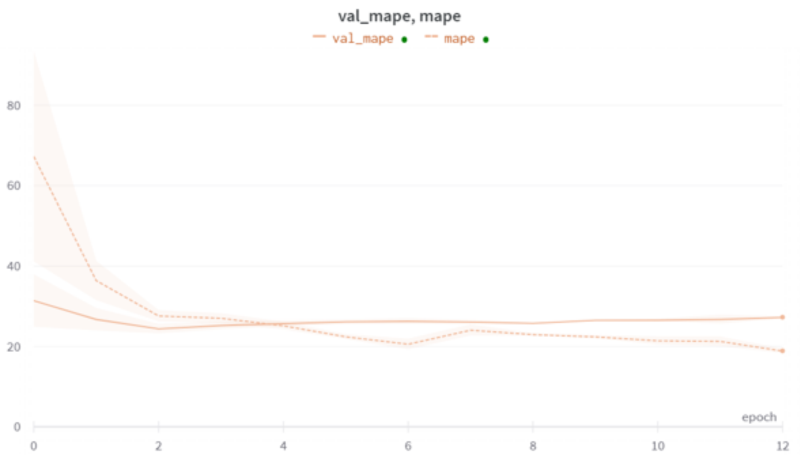

Figure C.5. The Plot of MAPE Values of Train and Validation Sets with Random Normal Weight Initialization

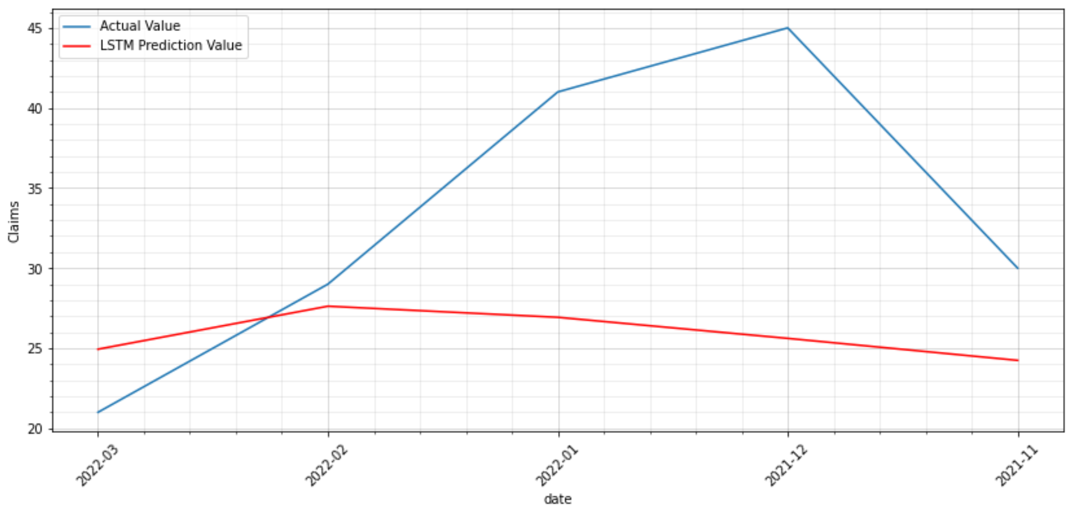

Figure C6. The Plot of Actual and Predicted Claims with Random Normal Weight Initialization on Validation Dataset

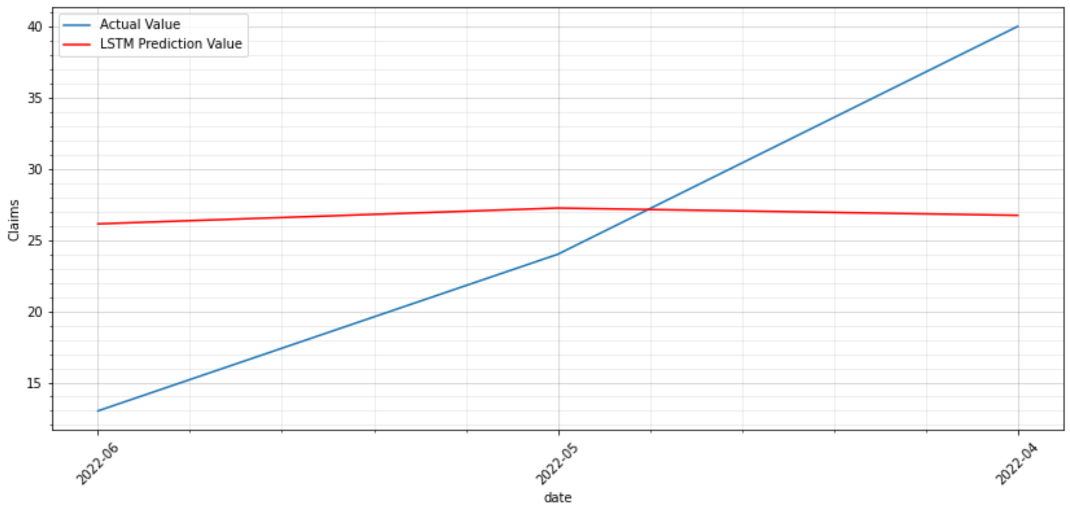

Figure C.7. The Plot of Actual and Predicted Claims with Random Uniform Weight Initialization

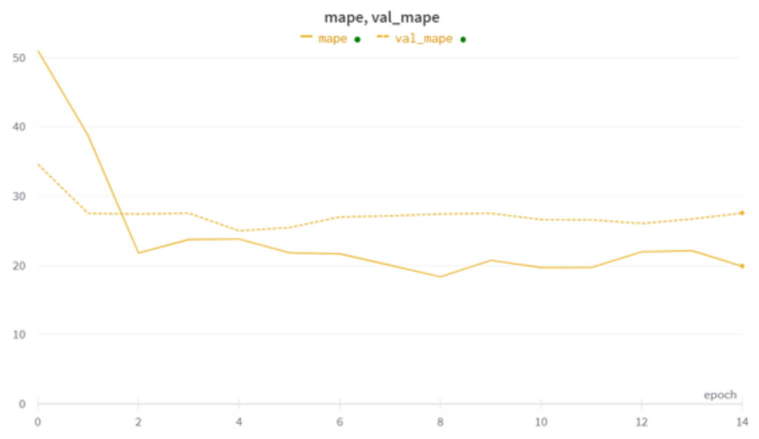

Figure C.8. The Plot of MAPE Values of Train and Validation Sets with Random Uniform Weight Initialization

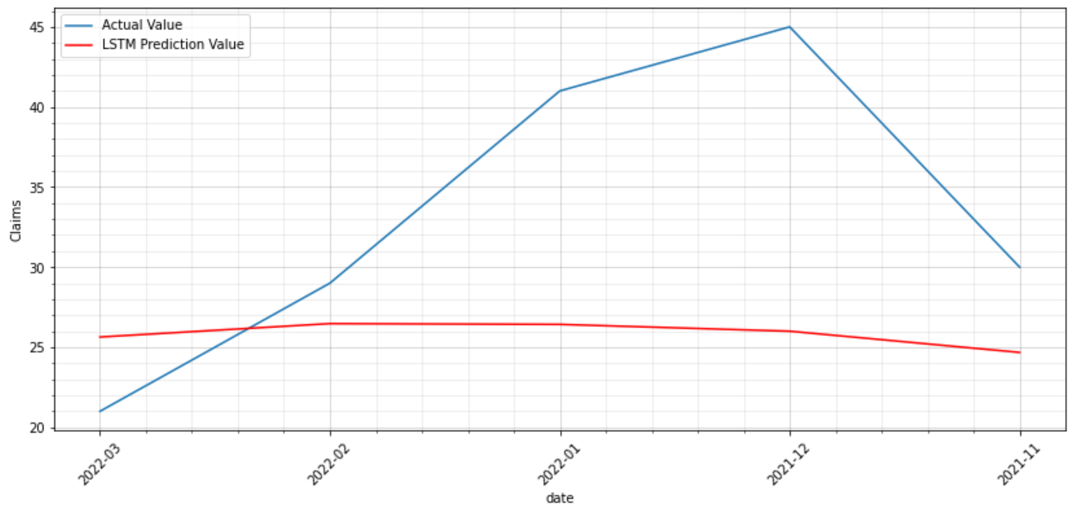

Figure C.9. The Plot of Actual and Predicted Claims with Random Uniform Weight Initialization on Validation Dataset

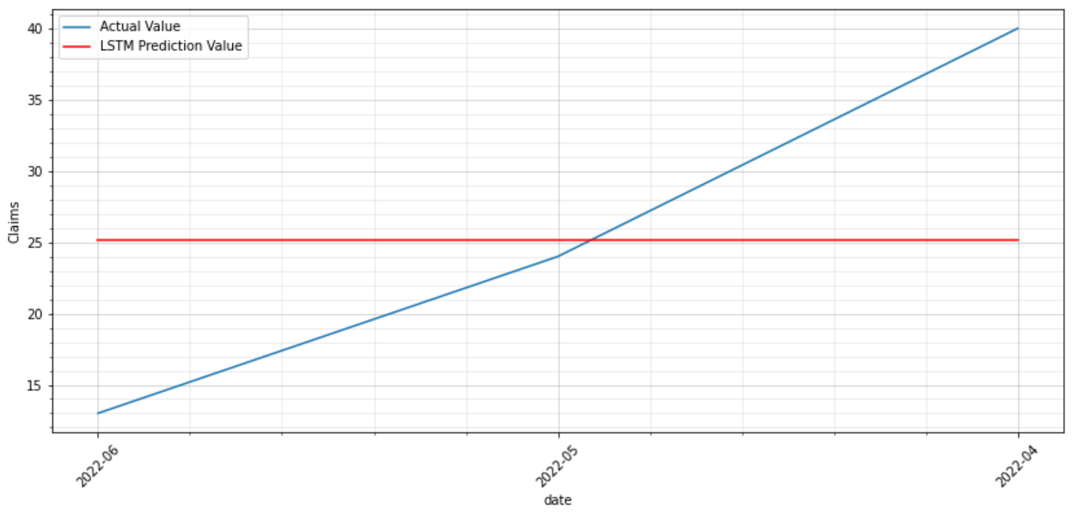

Figure C.10. The Plot of Actual and Predicted Claims with Zero Weight Initialization

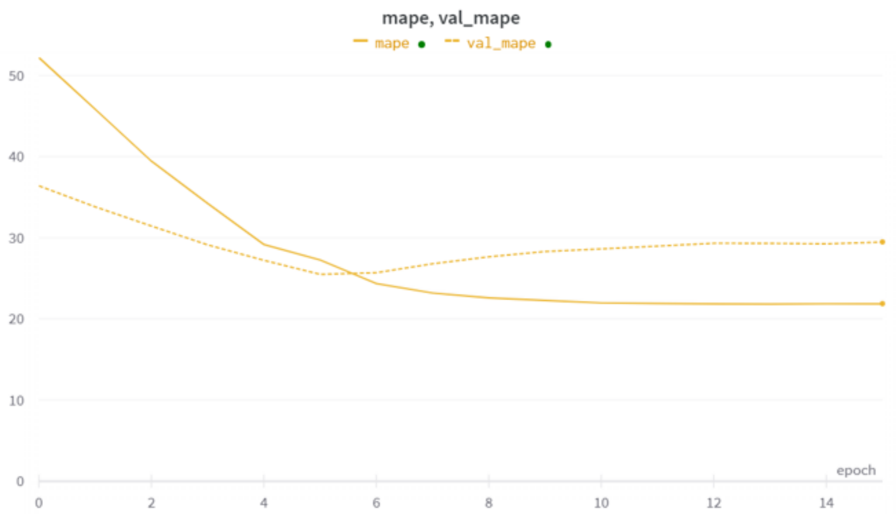

Figure C.11. The Plot of MAPE Values of Train and Validation Sets with Zero Weight Initialization

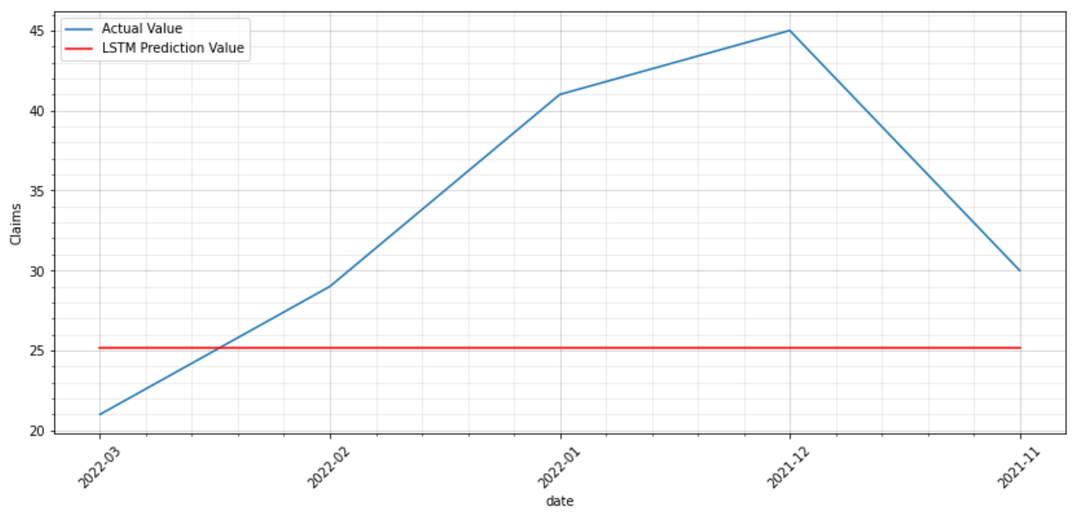

Figure C.12. The Plot of Actual and Predicted Claims with Zero Weight Initialization on Validation Dataset

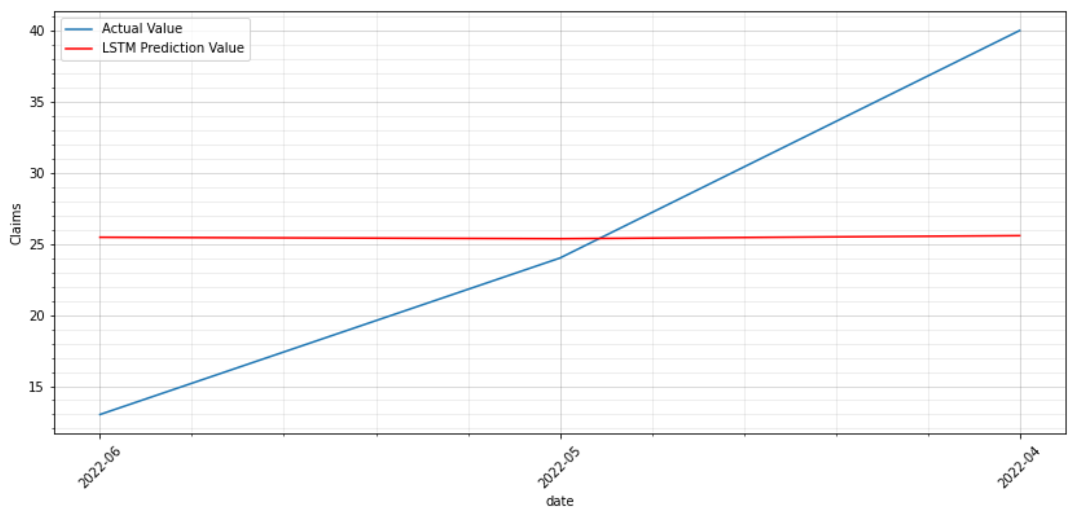

Figure C.13. The Plot of Actual and Predicted Claims with Glorot Normal Weight Initialization

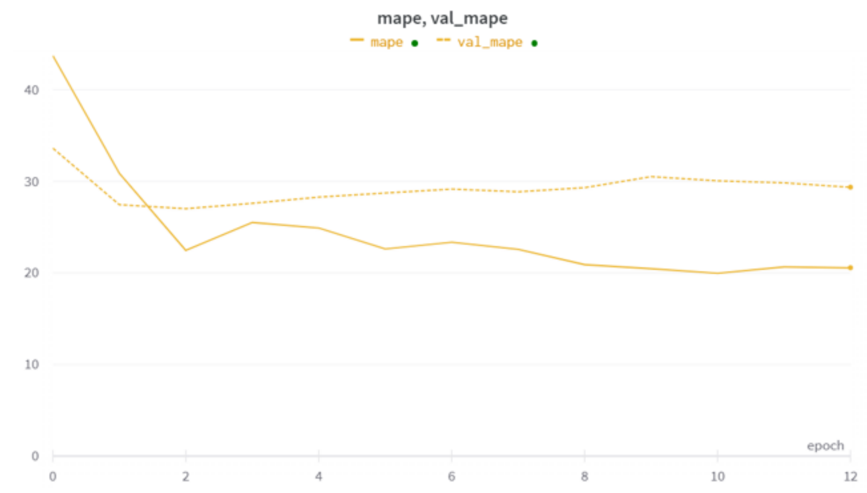

Figure C.14. The Plot of MAPE Values of Train and Validation Sets with Glorot Normal Weight Initialization

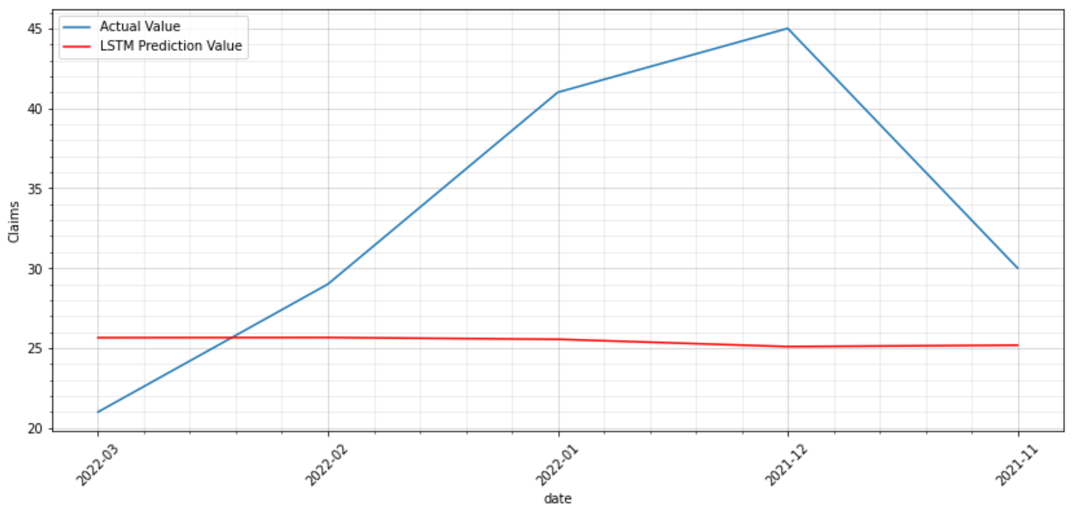

Figure C.15. The Plot of Actual and Predicted Claims with Glorot Normal Weight Initialization on Validation Dataset

Table C.1. Parameters of Models were tuned with Five Different Weight Initialization Methods

	Tuned Parameters
Glorot Normal Initializer	lstm layer(1st):2, dropout value:0.2, dense layer(1st):3, learning rate:0.01 epoch:15
sVariance Scaling Weight Initializer	lstm layer(1st):5, dropout value:0.2, dense layer(1st):3, learning rate:0.01 epoch:13
Random Normal Initializer	lstm layer(1st):1, dropout value:0.2, dense layer(1st):2, learning rate:0.01 epoch:10
Random Uniform Initializer	lstm layer(1st):1, dropout value:0.2, dense layer(1st):2, learning rate:0.01 epoch:11
Initializer Weight to Zero	lstm layer(1st):1, dropout value:0.2, dense layer(1st):3, learning rate:0.01 epoch:15