Forecasting Daily Foreign Tourists for a Tour Operator in Thailand

Mr. Pornpawit Niamjoy

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Industrial Engineering
Department of Industrial Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2019

การพยากรณ์จำนวนนักท่องเที่ยวต่างชาติรายวันสำหรับผู้ประกอบการทัวร์แห่งหนึ่งในประเทศไทย

นายพรภวิษย์ เนียมจ้อย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมอุตสาหการ ภาควิชาวิศวกรรมอุตสาหการ
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2562

Thesis Title	Forecasting Daily Foreign Tourists for a Tour Operator in Thailand
By	Mr. Pornpawit Niamjoy
Field of Study	Industrial Engineering
Thesis Advisor	NANTACHAI KANTANANTHA, Ph.D.

Accepted by the FACULTY OF ENGINEERING, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Engineering

.... Dean of the FACULTY OF ENGINEERING
(Professor SUPOT TEACHAVORASINSKUN, D.Eng.)

THESIS COMMITTEE

.... Chairman
(Associate Professor DARICHA SUTIVONG, Ph.D.)

.... Thesis Advisor
(NANTACHAI KANTANANTHA, Ph.D.)

.... Examiner
(Associate Professor NARAGAIN PHUMCHUSRI, Ph.D.)

.... External Examiner
(Associate Professor Chansiri Singhtaun, D.Eng.)

พรภวินย์ เนียมจ้อย : การพยากรณ์จำนวนนักท่องเที่ยวต่างชาติรายวันสำหรับผู้ประกอบการทัวร์แห่งหนึ่งในประเทศไทย. (Forecasting Daily Foreign Tourists for a Tour Operator in Thailand) อ.ที่ปรึกษาหลัก : อ. ดร.นันทชัย กานตานันทะ

ผู้ประกอบการทัวร์มีบทบาทสำคัญอย่างมากในอุตสาหกรรมการท่องเที่ยวซึ่งเป็นอุตสาหกรรมที่มีความสำคัญอย่างมากของระบบเศรษฐกิจของประเทศไทย การทำให้ได้ซึ่งค่าพยากรณ์จำนวนนักท่องเที่ยวรายวันสำหรับผู้ประกอบทัวร์ที่แม่นยำสูงนั้นเป็นสิ่งที่มีความสำคัญอย่างมากในการบริหารรายได้และการจัดการของบริษัทผู้ประกอบการทัวร์ เช่น การจัดหามัคคุเทศก์ หรือ ยานพาหนะ ให้เหมาะสมในแต่ละวัน ซึ่งงานวิจัยฉบับนี้ได้มีการนำเสนอและเปรียบเทียบตัวแบบพยากรณ์ที่จะนำไปเลือกใช้ให้เหมาะสมสำหรับบริษัทผู้ประกอบการทัวร์กรณีศึกษาโดยตัวแบบการพยาการณ์ที่ถูกนำมาใช้ในงานวิจัยฉบับนี้ประกอบไปด้วย Seasonal Autoregressive Integrated Moving Average model (SARIMA), Seasonal Autoregressive Integrated Moving Average model with exogenous variables model (SARIMAX), Trigonometric ARMA errors, trend and multiple seasonal patterns (TBATS), โครงข่ายประสาทเทียม (ANN) และ Long Short-Term Memory (LSTM) ซึ่งเป็นโครงข่ายประสาทเทียบประเภทหนึ่งซึ่งถูกสร้างขึ้นมาให้ประมวลผลข้อมูลที่มีลักษณะเป็นลำดับ จากผลการวิจัยที่ถูกชี้วัดด้วยการประเมินค่าความคลาดเคลื่อนสัมบูรณ์เฉลี่ย (MAE) และค่าร้อยละความคลาดเคลื่อนสัมบูรณ์เฉลี่ย (MAPE) พบว่าตัวแบบโครงข่ายประสาทเทียมนั้นเป็นตัวแบบพยากรณ์ที่เหมาะสมที่สุดสำหรับทัวร์ทั้ง 3 ประเภท แม้ว่าในการทดสอบกับข้อมูลตรวจสอบไขว้นั้น ตัวแบบ SARIMAX จะให้ผลลัพธ์ที่ดีกว่าในทัวร์ A และ B แต่ก็ไม่ได้มีความแตกต่างอย่างมีนัยสำคัญทางสถิติกับตัวแบบโครงข่ายประสาทเทียมที่มีการใช้ทรัพยากรน้อยกว่า และเมื่อเปรียบเทียบผลลัพธ์ที่ได้จากการทดสอบด้วยข้อมูลทดสอบระหว่างตัวแบบโครงข่ายประสาทเทียมกับตัวแบบพยากรณ์เดิมซึ่งเป็นวิธีการที่บริษัทกรณีศึกษาใช้อยู่ในปัจจุบันนั้นพบว่าสามารถลดความผิดพลาดจากการพยากรณ์ลงได้โดยค่าร้อยละความคลาดเคลื่อนสัมบูรณ์เฉลี่ยลดลงจาก 53.37 % เหลือเพียง 15.89% สำหรับทัวร์ A และจากผลค่าความคลาดเคลื่อนสัมบูรณ์เฉลี่ยสามารถลดความคลาดเคลื่อนจาก 15.73 คนเหลือเพียง 4.437 คน หรือลดลง 71.79% สำหรับทัวร์ A จาก 2.50 คนเหลือเพียง 1.684 คน หรือลดลง 32.64% สำหรับทัวร์ B และ จาก 3.08 คนเหลือเพียง 1.687 คน หรือลดลง 45.24% สำหรับทัวร์ C

สาขาวิชา	วิศวกรรมอุตสาหการ	ลายมือชื่อนิสิต	....
ปีการศึกษา	2562	ลายมือชื่อ อ.ที่ปรึกษาหลัก	....

6170221621 : MAJOR INDUSTRIAL ENGINEERING
KEYWORD: forecasting, Tourism Industry, Tour Operators, SARIMA, SARIMAX, TBATS, ANN, Neural Network, LSTM, Machine Learning

Pornpawit Niamjoy : Forecasting Daily Foreign Tourists for a Tour Operator in Thailand . Advisor: NANTACHAI KANTANANTHA, Ph.D.

Tour operators are playing an important role in the tourism industry which is an essential part of industries for Thailand's economy. Accurate tourist forecasting of daily tourist demands for tour operators is very important in revenue management and planning of tour operators, such as providing a guide or vehicle for each day. This research has presented and compared the forecasting models that will be selected to be suitable for the tour operator for a case-study company. The forecasting models used in this research consist of Seasonal Autoregressive Integrated Moving Average model (SARIMA), Seasonal Autoregressive Integrated Moving Average model with exogenous variables model (SARIMAX), Trigonometric ARMA errors, trend and multiple seasonal patterns (TBATS), artificial neural networks (ANN) and Long Short-Term Memory (LSTM) which is a type of neural network created to process sequential information. Based on the results, which were evaluated by the Mean Absolute Error (MAE) and the Mean Percentage Absolute Error (MAPE), it was found the artificial neural network model is the most suitable for all tours. In testing results with the cross-validation data, the SARIMAX model provides better results on Tour A and B, however it does not have a statistically significant difference with neural networks which using fewer resources. When comparing the testing results with testing data from the artificial neural network model with the same day last year model, which is the method currently used by the case study company, it was found that the predictive errors decrease from 53.37% of MAPE to only 15.89% and from 15.73 of MAE to only 4.437 or decreasing by 71.79%, from 2.50 to 1.684 or 32.64% and from 3.08 to 1.687 or 45.24% for Tours A, B and C, respectively.

Field of Study:	Industrial Engineering	Student's Signature ....
Academic Year:	2019	Advisor's Signature ....

ACKNOWLEDGEMENTS

This thesis could not be completed smoothly without these people. First, I would like to extend my deepest gratitude to my professors. Firstly, Dr. Nantachai Kantanantha is my advisor who helped and advised about forecasting models in the Machine learning section, including analysis of the results, which has led to a tremendous improvement in forecasting accuracy and is always encouraging. Secondly, Assoc. Prof. Naragain Phumchusri, Ph.D. who gave advice on tour data analysis, including cooperation with the case study company and time series models guidance and introducing new models. The advice of the professor helped to create many new ideas. Thank you to the case study company for your cooperation and assistance in facilitating information. In addition, I would like to thank my parents for all the encouragement and support of all things. Finally, thank you to my friends who have listened to my problems and encouraged all along.

Pornpawit Niamjoy

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

Chapter 1 Introduction

1.1 Tourism Industry Overview

Thailand ranks among the top destination countries for tourism because of a combination of its distinct lifestyle, unique landscape, and glamorous architecture. As one of the country's largest economic sectors, tourism job creators, export drives, and prosperity generators across the kingdom, the direct contribution of travel and tourism to GDP was THB 1.9 trillion, 12% of total GDP in 2019. Besides its direct economic impact, the industry has a significant indirect contribution to the economy such as investment spending and domestic purchases of goods and services relating to the sector.

Figure 1 shows the number of inbound tourists in millions which can be seen to continuously increase over the years. Tour operators form a crucial part of the industry by bringing a memorable experience for inbound tourists. There are approximately 7,500 registered travel agents and tour operators in Thailand. The majority of them are relatively small and do not have a well-organized management system in place. Moreover, data has not been properly collected, stored, and processed. Consequently, very few studies, if any, have been conducted in this area to aid in the understanding of customer behavior and hence capture additional values.

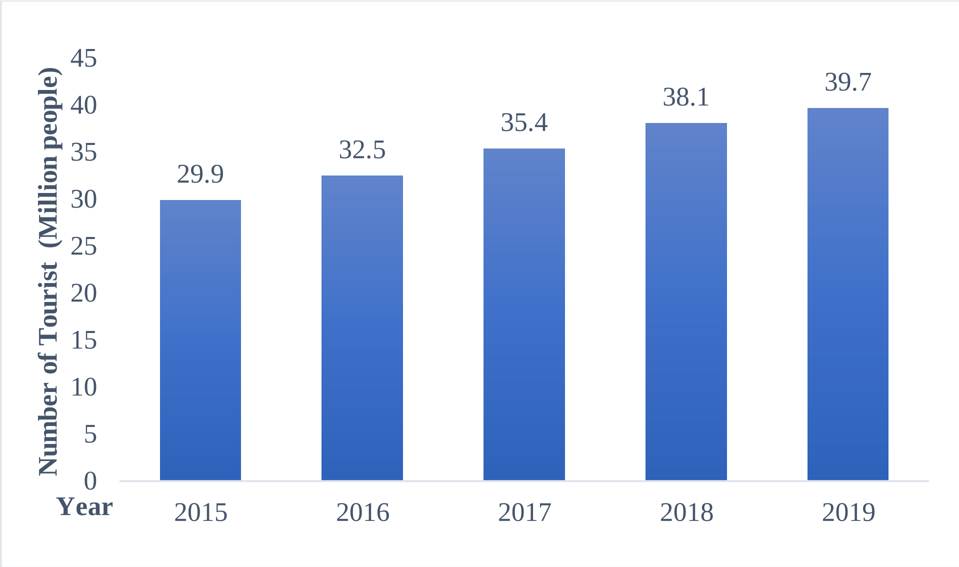


Figure 1: The number of tourists visiting Thailand each year (2015-2019)

1.2 Case-study Tour Operator Information

The case-study company is a leading tour operator in Thailand and was established in 2011. Visitors need to reserve tour tickets online in advance. A variety of tour programs are offered including night trips, day trips, and evening trips. This tour operator focuses on providing experiences such as ultimate historical and cultural exploration. Figure 2 shows the number of attendants in each tour in 2018 for which the case-study company has collected data for a total of 7 tour programs: namely, Tour A - Tour G. According to the data, Tour A, Tour B, and Tour C account for 95 percent of the total tour attendants. Therefore, this study will focus on forecasting models for these three tours.

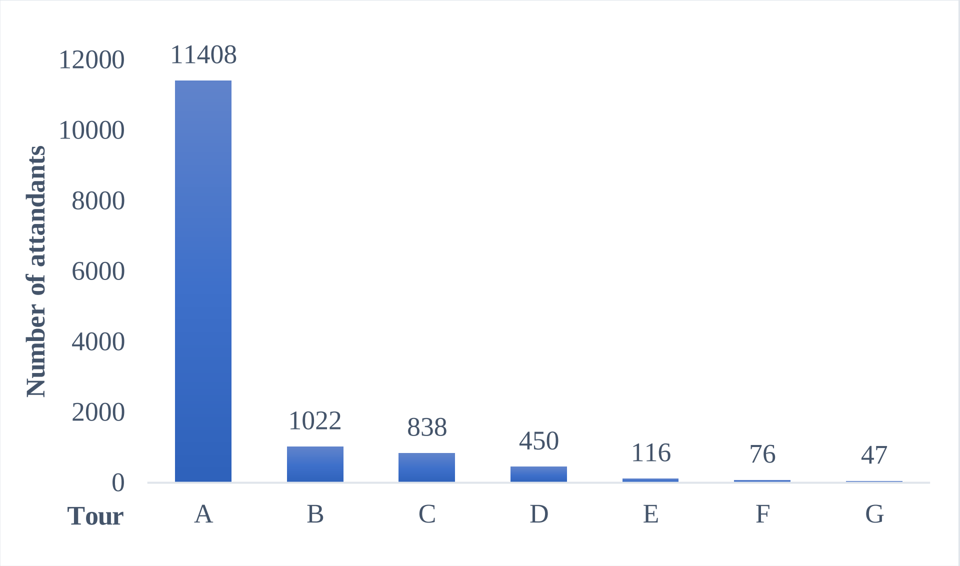


Figure 2: The number of attendants in each tour (2019)

Brief descriptions of focused tours are as follows:

1. Night Tour (Tour A) offers customers night eateries around Bangkok via local transportation. In addition, the tour riders will experience cultural landmarks and some places in Bangkok that are unknown to tourists. Tour A operates every day from 6 p.m. onwards.

2. Day Tour (Tour B) offers their customers to experience the local community and explore diverse regional Thai cuisines. The tour riders will get to learn how to order food like Thais. Tour B operates every day from 9 a.m. onwards.

3. Evening Tour (Tour C) offers their customers street food in Chinatown by unveiling top-notch Thai Chinese street vendors. The tour riders will get to visit religious landmarks and their history. Tour C operates every day from 6 p.m.

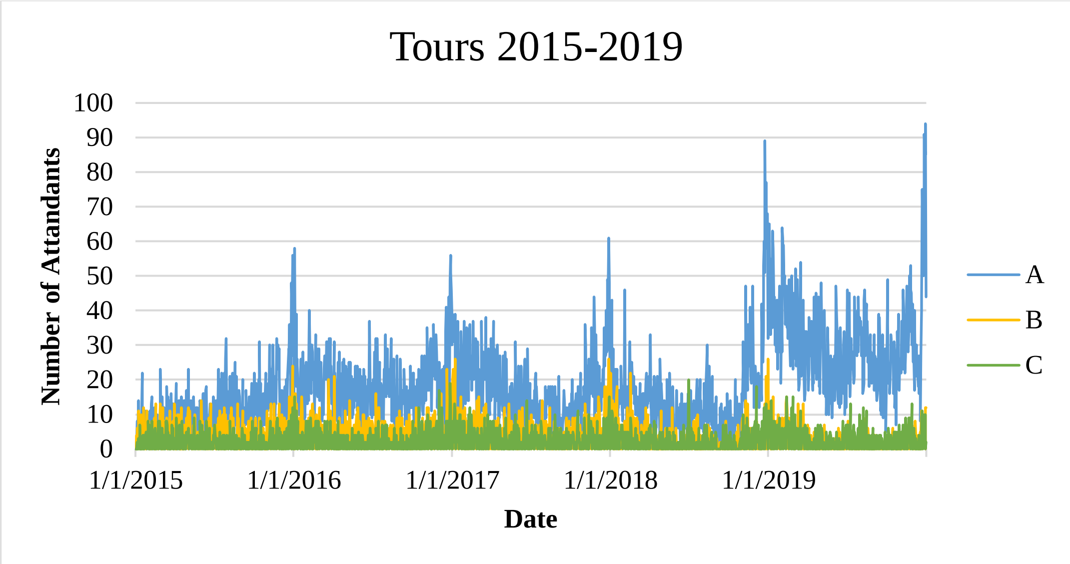


Figure 3: Time series plot of daily demand for each tour from January 2015 to December 2019

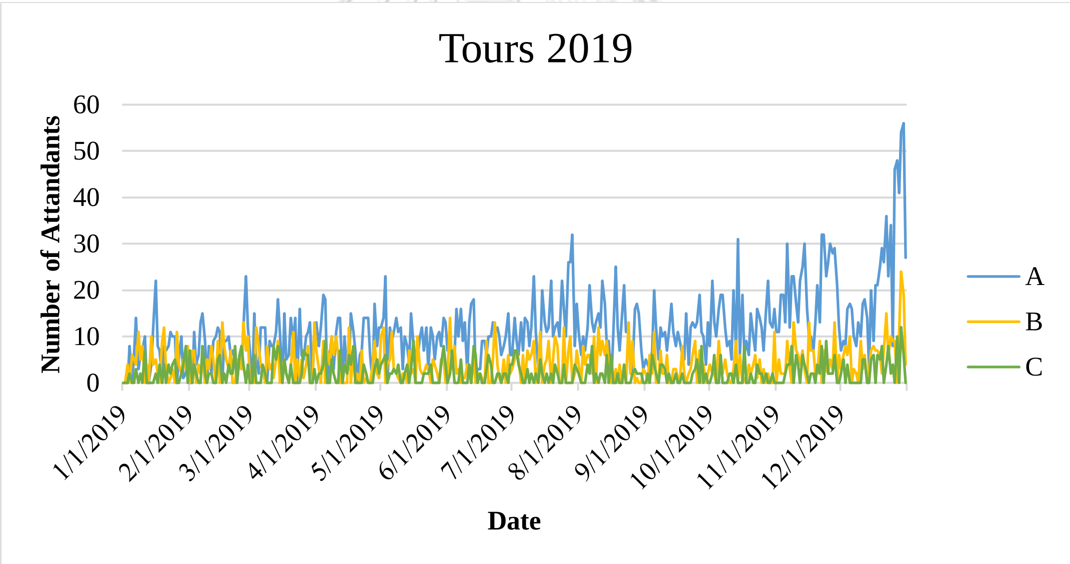


Figure 4: Time series plot of daily demand for each tour from January 2019 to December 2019

Figure 3 shows the time series plot of daily demand for the case study tour operator for Tour A-C from January 2015 to December 2019. It can be noticed that the demand for Tour A for the last year is higher than all the past years. Figure 4 only presents the data from January 2019 to December 2019. It can be seen that the high season occurs in the beginning and the ending of the year. Another observation is that seasonality does not only happen on a monthly basis but also on a daily basis. For instance, Saturday demand is usually higher than the other days. Hence, this demonstrates a double seasonal pattern for the tour demand of this tour operator. It is interesting to explore the appropriate forecasting methods to accurately forecast this type of data for this tour operator. Specifically, this thesis aims to propose time series forecasting models for tour demand and to find insights on how to obtain accurate forecast results.

1.3 Forecasting

It is advantageous to the management team to have advanced knowledge of tour demand since there are things to plan ahead, for example, transportation and tour guides. These resources are outsourced. The company needs to guarantee their schedule for approximately one week in advance. The result of this paper can provide the tour operator management team with reliable forecasting methods for each tour. Also, the insights can assist the management team with the knowledge of trends, seasonality patterns, and affecting variables that explain the data pattern.

1.4 Problem Statement

In the present day, a case study tour operator forecast the demand by using Same Day Last Year method. This method is a nearly Naïve method. The difference is that the Naïve method uses the data of the same date from the last year, but the Same Day Last Year uses the day in the past year, for example, using the first Saturday of January to predict the value of the first Saturday of January for the next year. Table 1 shows the error of this forecasting method.

Table 1: The accuracy measurement of existing forecast method

	Tour A	Tour B	Tour C
MAE	17.164	3.019	3.031
MAPE	55%	-	-

1.5 Objectives

The objective of this thesis is to find suitable forecasting models which improve MAPE more than 50% for Tour A and reduce MAE to be less than 2 people per day for Tour B and Tour C for daily tourist demand which is the number of daily tour attendants for a case-study tour operator in Thailand.

1.6 Scopes

1. This thesis uses three tour data, namely tour A, tour B and tour C, which account for 95% of the tour attendants of the case-study tour operator.

2.This thesis focuses on time series analysis which are SARIMA, SARIMAX, TBATS and machine learning models which are ANN and LSTM.

3.This thesis separates data into training data (1461 Days in 2015 – 2018), cross validation data (182 days in 2019) and testing data (183 days in 2019).

4. The accuracy performance of forecasting models is evaluated in terms of mean absolute error (MAE) for tours A, B and C, and mean absolute percentage error (MAPE) for tour A. They are calculated by the following equations.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|A_t - F_t| \quad (1)$$

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \quad (2)$$

- n is number of samples to measurement

- A_t is actual value

- F_t is forecast value

1.7 Outcomes

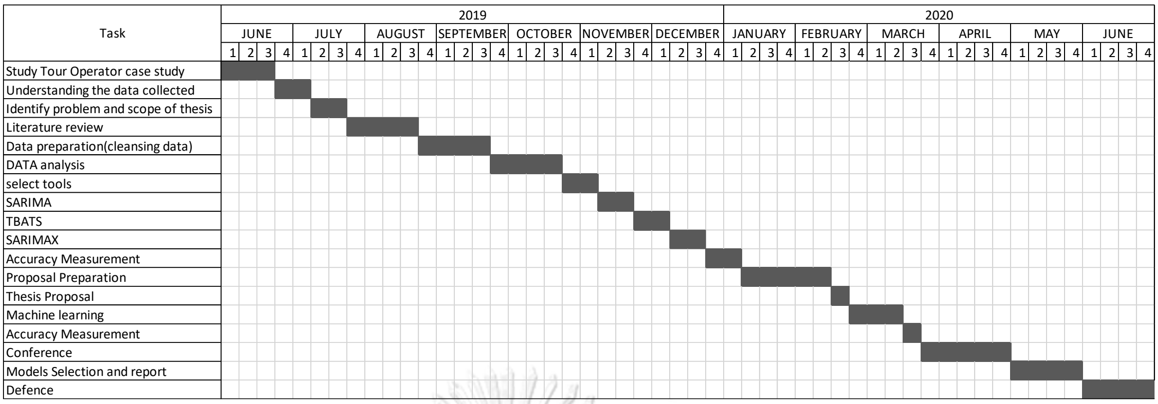
A suitable forecasting model for each considered tour's daily tourist demand for the case study tour operator in Thailand.

1.8 Benefits

1. The accurate forecasting can help a company estimate their resources such as guide, vehicle and other services.

2. Understanding customer behaviors to improve services.

1.9 Research Timeline



Chapter 2 Literature Review

The objective of this research is to forecast the daily inbound tourism of tour operators. The author analyzes the time series plot that the daily data of tours A, B, and C with a seasonal pattern. This literature review focuses on tour operator forecasts with daily data. According to the previous study, tour operator forecasting research has never existed before. Therefore, it is concluded that this thesis is the first research that forecasts demand tour operators. According to the reasons stated, this literature review focuses on a similar type of data and in the tourism industry such as forecasting number of inbound tourism to each country, forecasting hotel occupancy demand, and car rental business that most are seasonal pattern data.

2.1 Daily forecasting researches

Many researchers have tried to forecast with daily data. Most of them have chosen and compared many models to find the model that made the most accuracy for the data. In 2000, (Prybutok, Yi, and Mitchell 2000) forecasted daily maximum ozone concentration by comparing three models – neural network model, regression model and ARIMA model. The data from 1 June 1994 to 30 September 1994 (4 months) were used to predict the data from 1 to10 October 1994 (10 days). MAD and RMSE are considered as accuracy measurements. The result shows that the neural network was more accurate than ARIMA and regression. In 2005, (Osowski and Garanty 2007) forecasted daily meteorological pollution using a support vector machine (SVM) and a wavelet decomposition. The accuracy was measured by mean absolute error and the relative (normalized) error. In 2007, (Taylor 2007) predicted many daily products to set the level of safety stock by using the exponentially weighted quantile regression as a forecasting model and the mean absolute error as an accuracy measurement. In the past few years, there was much research focusing on neural networks which are more accurate than typical forecasting models For example, in 2007, (Paoli et al. 2010) forecasted the preprocessed daily solar radiation time series by neural networks and compared with the reference methods such as ARIMA, Bayesian Inference, Markov Chains, and k-Nearest-Neighbors using the mean absolute error and RMSE as the accuracy measurement. Many researchers have used hybrid models by combining typical models. For example, (Divino and McAleer 2010) forecasted daily international mass tourism to Peru by using GARCH–DLY, GARCH–DLYMA, GJR–DLY, GJR–DLYMA, EGARCH–DLY, EGARCH–DLYMA models. However, the typical models such as SARIMA, Holt-Winters are still famous for the present. (Arunraj and Ahrens 2015) forecasted daily food sales by a hybrid seasonal autoregressive integrated moving average (SARIMA), SARIMAX and quantile regression and used mean absolute percentage error and RMSE as the measurements.

Table 2: Summary of daily forecast researches

Study	**Modeling**	**Forecasting**
(Prybutok, Yi, and Mitchell 2000)	ANN, SARIMA	Ozone concentration
(Osowski and Garanty 2007)	SVM and wavelet decomposition	Daily meteorological pollution
(Taylor 2007)	Exponentially weighted quantile regression	Daily supermarket products
(Paoli et al. 2010)	ANN	Solar radiation
(Divino and McAleer 2010)	GARCH	Mass tourism to Peru
(Arunraj and Ahrens 2015)	SARIMA, SARIMAX	Daily food sales

2.2 Review of forecasting in tourism industry

Since there is no research related to tour operator forecasts, this literature review will focus on the main types of tourism industry which are accommodation and transportation. Thereby, the main focus of this section will be on the hotel industry and car rental forecasting.

For hotel industry tourism, in 2000, (Rajopadhye et al. 2001) used Holt-Winters and the combined method to forecast hotel room demand. For the car rental industry in 2003 (Wan 2012) forecasted car rental demand by SARIMA. After that in 2007, the researcher (YüKsel 2007) compared Holt-Winters and ARIMA in forecasted hotel room demand. BATS and TBATS have gained a reputation in dealing with non-linear data during these past few years For example, in 2016 (Pereira 2016) compared Holt-Winters, Double season Holt-Winters, BATS and TBATS by using naïve method as a benchmark to forecast high frequency daily occupancy data from 300 rooms of Portuguese's four-star hotel. He mentioned that the daily time series were dissimilar to monthly, quarterly or annual data because daily time series presented high frequency and complex seasonal patterns.

In the tourism industry, most literature reviews focus on forecasting the number of tourists (Witt and Song 2001). The number of demands for a specific company (e.g. hotel room (Weatherford and Kimes 2003) and car rental demand (Wan 2012)). The most common time series model for the tourism industry was SARIMA which can capture seasonal components with higher accuracy than other typical models. Time series models were commonly used to forecast in the hotel industry. Other models that draw many attention recently are SARIMAX which can capture more than one seasonal component and TBAT which is the extension of BATS model with double seasonal Holt-Winters integrated with Box-Cox transformation to handle with non-linear data, and with ARMA model to account for autocorrelation in time series by residuals.

Table 3: Summary of tourism industry forecast

Study	Modeling	Forecasting
(Witt and Song 2001)	SARIMA	Tourism flows
(Rajopadhye et al. 2001)	Holt-Winters	Hotel room demand
(Weatherford and Kimes 2003)	SARIMA	Hotel revenue management
(YüKsel 2007)	Holt-Winters, TBATS	Hotel demand
(Wan 2012)	SARIMA	Beijing Car Rental
(Pereira 2016)	BATS, TBATS	Hotel revenue management

From Tables 2 and 3, the author decides to choose SARMA, SARIMAX and TBATS as the forecast modeling.

Forecast Modeling

SARIMA: seasonal autoregressive integrated moving average

BATS: double seasonal Holt-Winters, integrated with Box-Cox transformation

TBATS: Trigonometric BATS

2.2.1 ARIMA

ARIMA model has known as autoregressive integrated moving average models or ARIMA (p, d, q) model which were developed from ARMA model (autoregressive moving average) by increasing the differencing part that can change data from non-stationarity to stationarity. ARIMA model is one of the most famous time series models used to predict the future value from historic data. The original models consisted of the AR (autoregressive) part which focuses on the regression of own lagged values and MA (moving average) part that showed the regression error in the past. Later, ARMIA models has assorted models such as Seasonal ARIMA (SARIMA), ARIMA with exogenous variables (ARIMAX) and Seasonal ARIMA with exogenous variable (SARIMAX)

2.2.2 BATS and TBATS

The original BATS model forecasting method was presented by (De Livera 2010) and (De Livera, Hyndman, and Snyder 2011) . BATS model was a developed model of double seasonal Holt-Winters by integrating with Box-Cox transformation for non-linear data and ARMA model to account for autocorrelation in time series by residuals (De Livera 2010). This shows that BATS model prediction accuracy is better than simple time series models. However, the BATS model still has a disadvantage in high frequency and data that has many seasonal components or complex seasonality. Later, (De Livera, Hyndman, and Snyder 2011) proposed the TBATS (Trigonometric BATS) model by including trigonometric functions into the BATS model. This makes TBATS perform better than BATS when fitting with high-frequency data and can reduce model parameters. Thus, TBATS can apply with high-frequency data, a non-integer seasonal period (i.e., 365.25 for daily data with a leap year), and non-nested periods.

2.2.3 ANN

An artificial neural network is one of the most famous models in the machine learning model. It was a non-linear statistical machine learning model. First, it is created to recognize complex data patterns. Then, the NN models were made by creating a computational model for neural networks (McCulloch and Pitts 1943). Also, the model continues developing and applying to many fields. In the present day, there are many types of neural networks. The ANN model was applied in many applications such as classification, prediction, and time series. ANN is mostly used in time series to forecast numbers with the time series variable which the data will be input to train models. Figure 5 shows the structure of the ANN method.

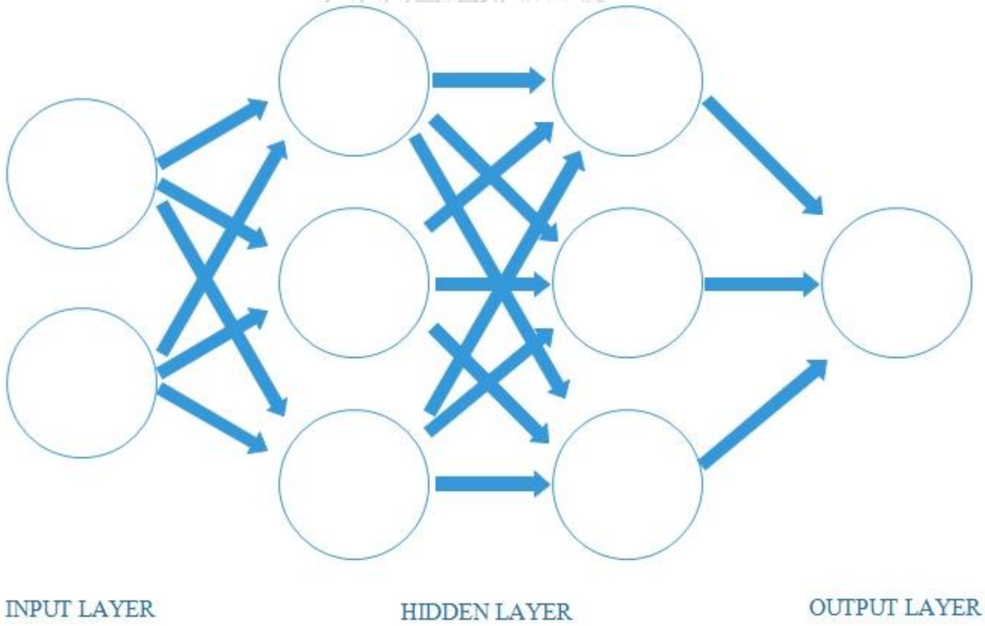


Figure 5: structure of ANN model (tutorialspoint 2020)

2.2.4 LSTM

Long short-term memory was proposed by (Hochreiter and Schmidhuber 1997) It was created from a constant error carousel unit dealing with the vanishing gradient problem in the RNN model. The LSTM model has the same structure as the RNN model. The advantage of this model is that it can process sequences of data not only a single data point like ANN. The famous field of LSTM is voice recognition, text recognition, and time-series data prediction since they can capture lags between time periods in a time series. The structure of the LSTM unit consists of a cell, an input gate, an output gate and a forget gate. The cell recognizes numbers across time intervals and the gates that control the flow of cells' in-out data i. Figure 6 shows the structure of the LSTM method. The LSTM cell structure will be explained in the next chapter.

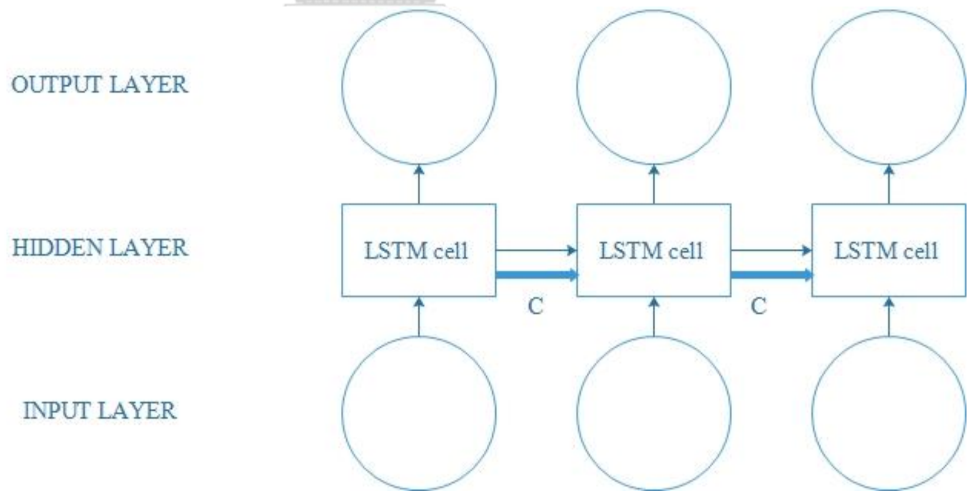


Figure 6: structure of the LSTM model (colah'sblog 2015,August 27)

2.3 Reviews of Forecasting Accuracy Measures

Table 4: Summary of Accuracy measures

Study	Accuracy measures	Forecasting
(Prybutok, Yi, and Mitchell 2000)	MAD, RMSE	Ozone concentration
(Osowski and Garanty 2007)	MAE, relative (normalized) error.	Daily meteorological pollution
(Taylor 2007)	MAE	Daily supermarket products
(Paoli et al. 2010)	MAE, RMSE	Solar radiation
(Arunraj and Ahrens 2015)	MAPE, RMSE	Daily food sales

According to Table 4, the mean absolute error (MAE) and mean absolute percentage error (MAPE) are chosen because the MAE can show actual error while MAPE can show the percentage of the error.

Accuracy Measures

MAD: mean absolute deviation

MAE: mean absolute error

MAPE: mean absolute percentage error

RMSE: root mean square error

2.3.1 Mean absolute error (MAE)

Mean Absolute Error is one of the most famous measurements because it uses the difference of actual observation and directly forecasts the numbers. MAE has the absolute value that differentiate positive and negative errors. If it is not for the absolute value, the error will probably be zero due to the positive and negative errors. MAE is calculated by the following equation:

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|A_t - F_t| \tag{1}$$

- n is number of samples to measurement

- A_t is actual value

- F_t is forecast value

2.3.2 Mean absolute percentage error (MAPE)

Mean Absolute Percentage Error is one of the most famous measurements that researchers use because it shows errors in percentage making it easy to understand. MAPE is the mean ratio of error and actual number. MAPE is calculated by the following equation.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \tag{2}$$

2.3.3 Sliding

There are many ways to measure accuracy. The sliding method is one of them. A big advantage of this method is that the training set data can be adapted to every model making it be able to stay updated and suitable for trending data.

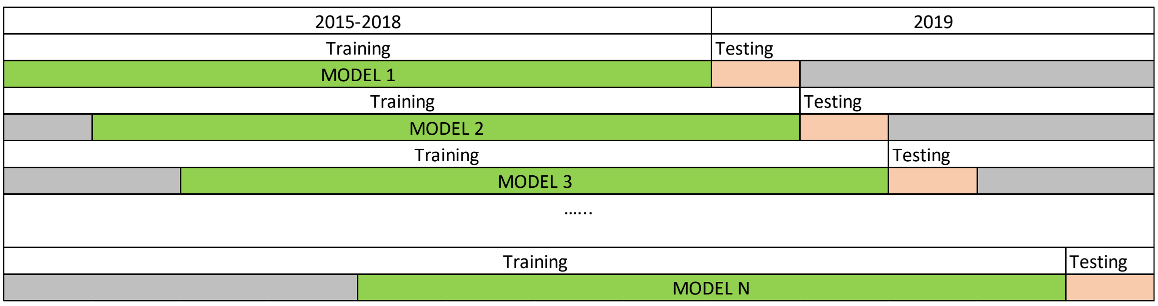


Figure 7: Splitting data for each model

Figure 7 shows how this method splits the data for each model that was updated every period of time.

2.4 Summary of timeseries models forecasting

ARMA errors, trends and multiple seasonal patterns (BATS) along with Trigonometric BATS (TBATS) were proposed by De Livera for the first time in 2010 and De Livera together with Hyndman, and Snyder in 2011.The main difference between these three models is how they perform seasonal component. SARIMAX is improved from SARIMA by repairing the big disadvantage which is its ability to perform only one seasonal component. TBATS has become a famous model in recent years since it was shown that TBATS can handle complex seasonal time series variations (De Livera, Hyndman, and Snyder 2011).

Chapter III: Methodology

The objective of this research is to propose forecasting models that can accurately estimate the number of attendants for the case-study tour operator company. This section contains data information used in this research and the explanation of focused forecasting methods.

3.1 The Data

This study aims to analyze historical data on the tour demand for this tour operator to identify patterns or trends for providing insights for the future estimation. To investigate the trend and seasonality of tour daily demand data, the time series plot of inbound tourist attendants is examined. The total data obtained from the company is the daily tour demand from January 2015 to December 2019 for Tours A, B, and C. The data are divided into 3 parts: January 2015 to December 2018 for the training set, January 2019 to 1st July, 2019 (182 days) for the cross validation set and 2nd July, 2019 to December 2019 (183 days) for the Test set. Time series plots from Figures 8, 9 and 10 show examples of time series data of the tours A, B and C respectively. The pattern of time series clearly shows the components of trend and seasonality.

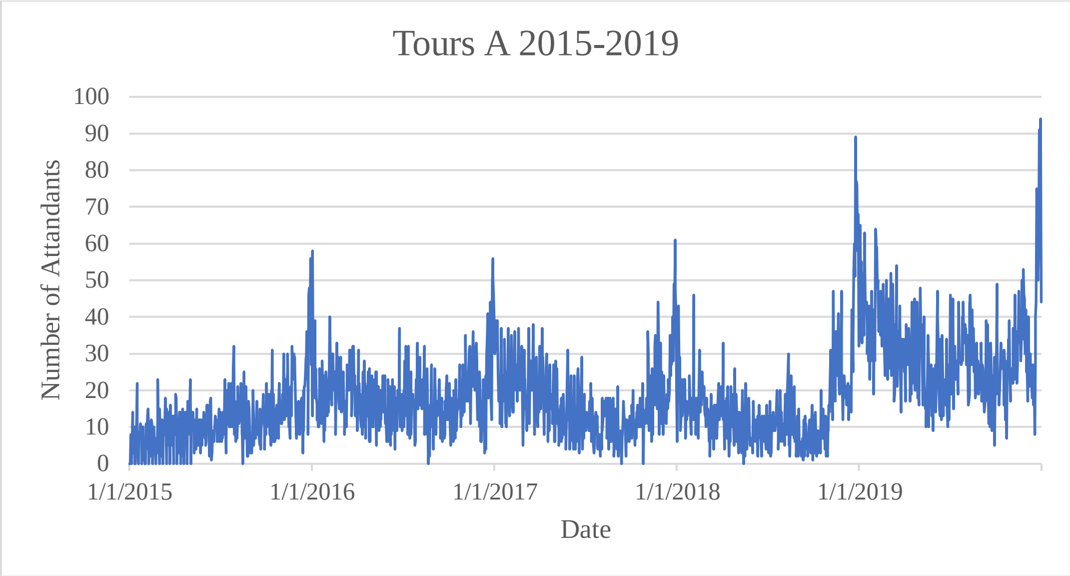


Figure 8: Time series data of the tour A

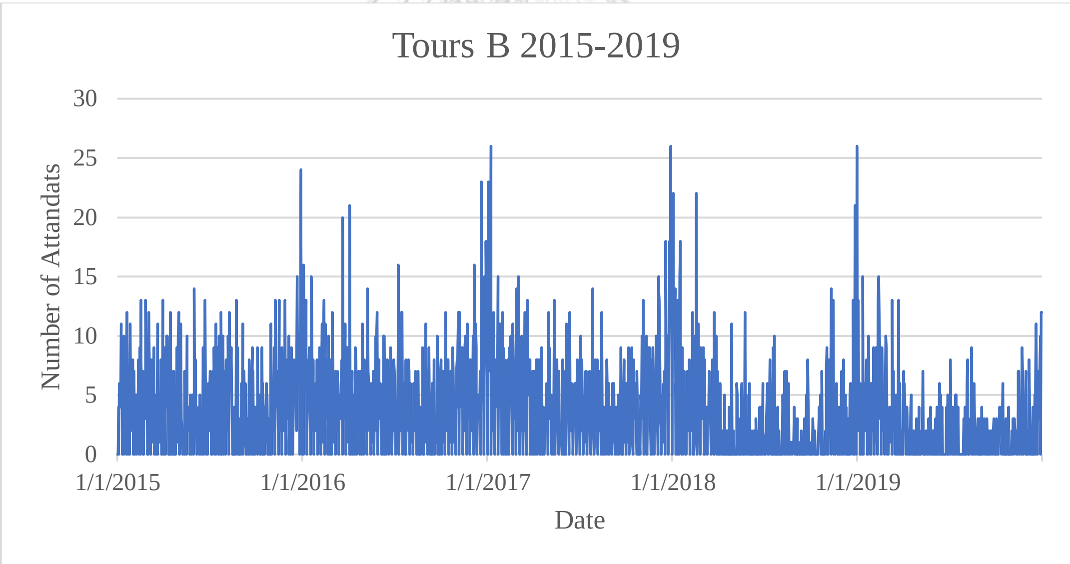


Figure 9: Time series data of the tour B

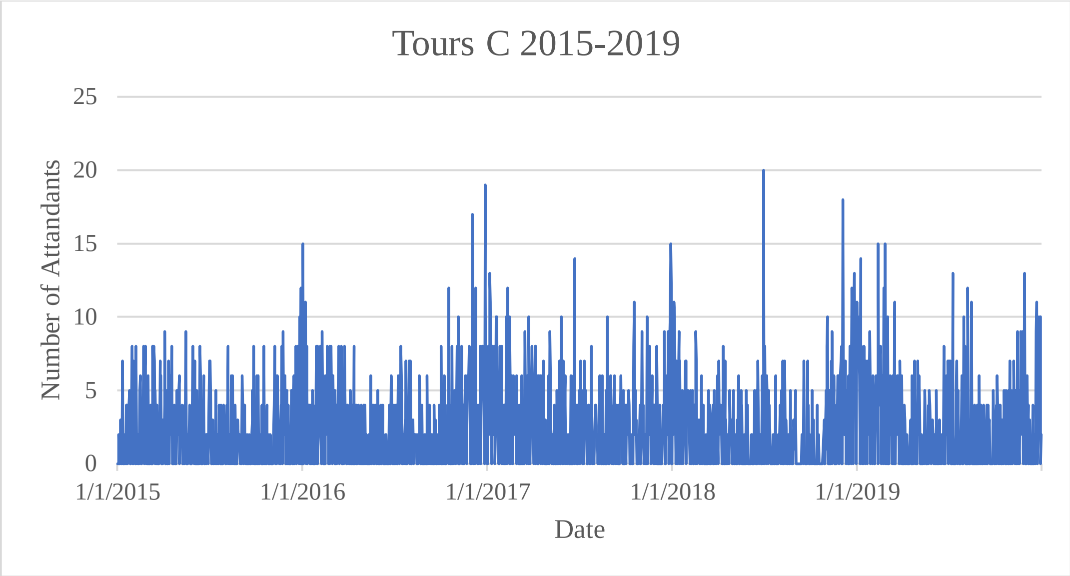


Figure 10: Time series data of the tour C

3.1.1 Data transformation

A case study of the tour operator company collects the transaction data in its own program which can import to Microsoft Excel as shown in Figure 11. One row represents one booking.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Booking ID	Customer	Product Name	Departure	Created	Payment Status	Booking Status	Departure Type	Product Type	Option Time	Age	Nationality	Email	Phone	Attendant
2				1/22/2019	1/1/2019					18:00					2
3				1/11/2019	1/1/2019					19:00					2
4				1/2/2019	1/1/2019					18:00					2
5				1/2/2019	1/1/2019					16:00					2
6				1/10/2019	1/1/2019					19:00					4
7				1/3/2019	1/1/2019					8:45					1
8				1/2/2019	1/1/2019					18:00					2
9				1/2/2019	1/1/2019					18:00					2
10				1/2/2019	1/1/2019					18:30					1
11				1/2/2019	1/1/2019					18:00					2
12				1/3/2019	1/1/2019					18:00					2
13				1/3/2019	1/1/2019					18:00					4
14				1/12/2019	1/1/2019					18:30					2
15		CONFIDENCIAL		2/6/2019	1/1/2019		CONFIDENCIAL			19:00		CONFIDENCIAL			2
16				1/19/2019	1/2/2019					19:00					2
17				1/14/2019	1/2/2019					19:00					2
18				1/17/2019	1/2/2019					10:00					2
19				1/7/2019	1/2/2019					18:30					3
20				2/8/2019	1/2/2019					18:30					2
21				2/16/2019	1/2/2019					19:00					2
22				1/2/2019	1/2/2019					18:00					1
23				1/19/2019	1/2/2019					7:00					1
24				2/7/2019	1/2/2019					19:00					2
25				4/5/2019	1/2/2019					19:00					2
26				1/8/2019	1/2/2019					18:00					2
27				2/3/2019	1/2/2019					19:00					4
28				1/4/2019	1/2/2019					18:30					2
29				1/7/2019	1/2/2019					19:00					2

Figure 11: Shows the raw transaction data

The data is transformed into time-series data and cleaned by the pivot table in Microsoft Excel as shown in Figure 12.

	A	B	C	D
1	DATE	A	B	C
2	1/1/2015	0	0	0
3	1/2/2015	0	0	0
4	1/3/2015	4	0	0
5	1/4/2015	0	2	8
6	1/5/2015	6	0	0
7	1/6/2015	5	0	6
8	1/7/2015	4	3	14
9	1/8/2015	11	0	3
10	1/9/2015	5	2	6
11	1/10/2015	8	0	7
12	1/11/2015	0	7	10
13	1/12/2015	4	0	0
14	1/13/2015	0	0	0
15	1/14/2015	10	0	4
16	1/15/2015	4	0	12
17	1/16/2015	5	2	22
18	1/17/2015	2	0	8
19	1/18/2015	0	4	7
20	1/19/2015	9	0	0
21	1/20/2015	12	4	8
22	1/21/2015	3	0	7
23	1/22/2015	0	4	8
24	1/23/2015	1	2	11
25	1/24/2015	3	4	10
26	1/25/2015	0	5	10
27	1/26/2015	11	0	0
28	1/27/2015	4	0	5
29	1/28/2015	2	2	10

Figure 12: The example of time series data

Figure 12 shows the sum of attendants in each day ranked by date. Then plot ACF plot to inspect linear relation between time lags this is a initial step of time series analysis.

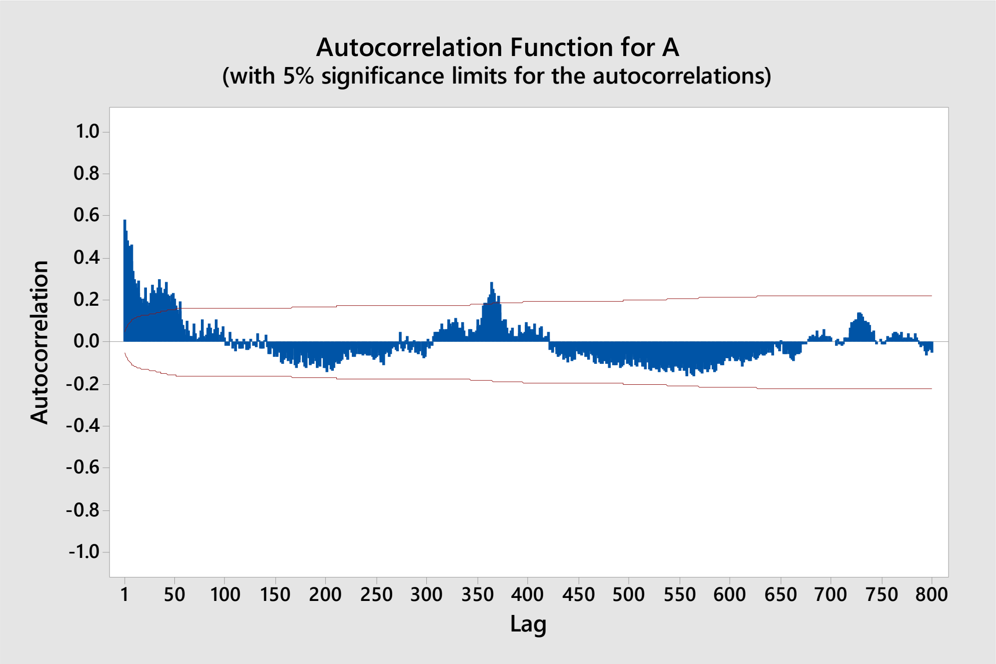


Figure 13: ACF plot of Tour A with 800 lags

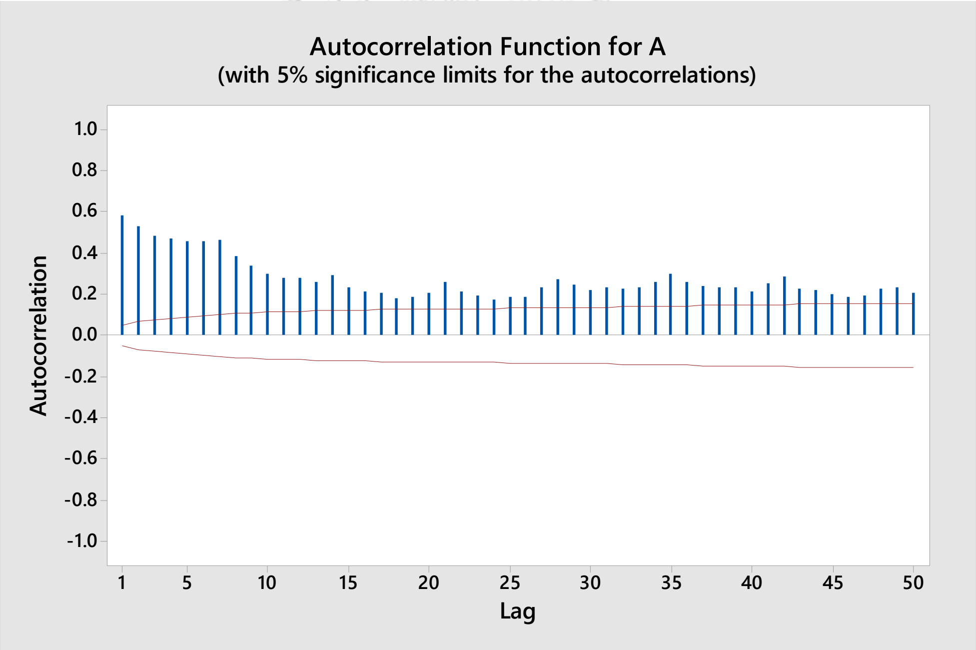


Figure 14: ACF plot of Tour A with 50 lags

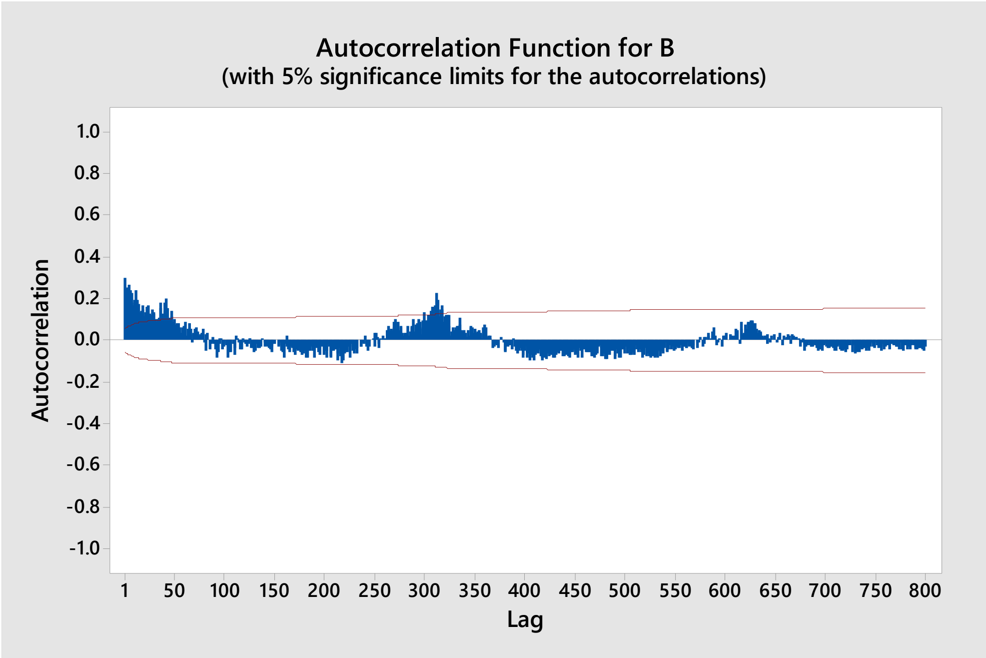


Figure 15: ACF plot of Tour B with 800 lags

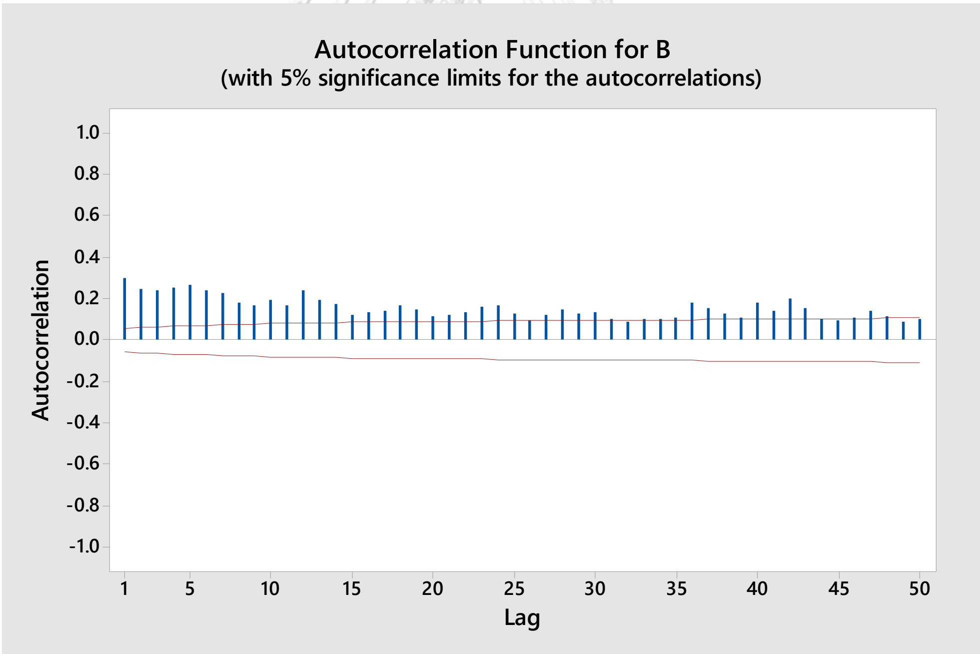


Figure 16: ACF plot of Tour B with 50 lags

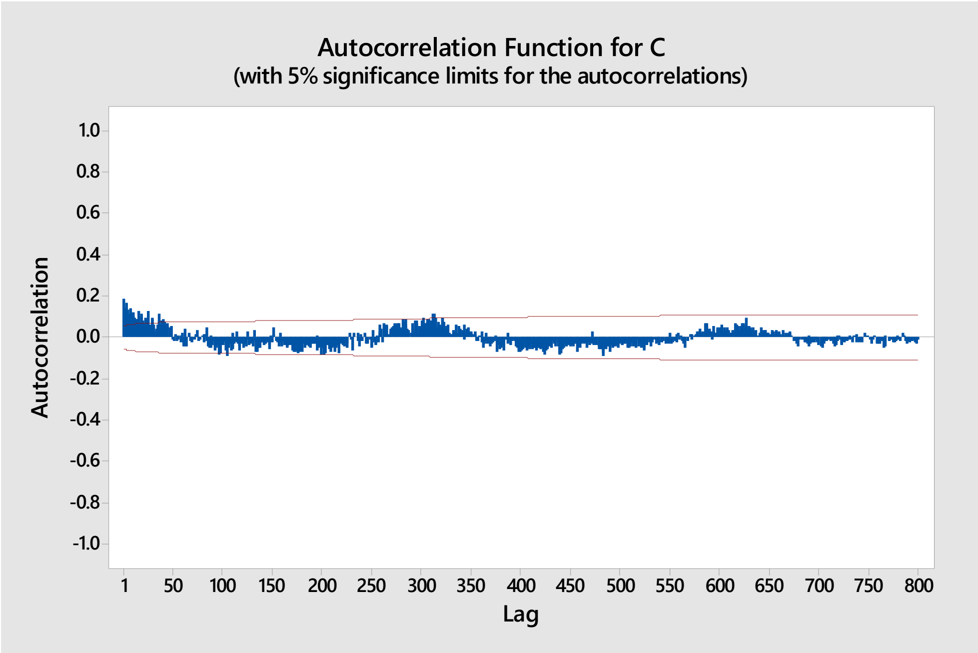


Figure 17: ACF plot of Tour C with 800 lags

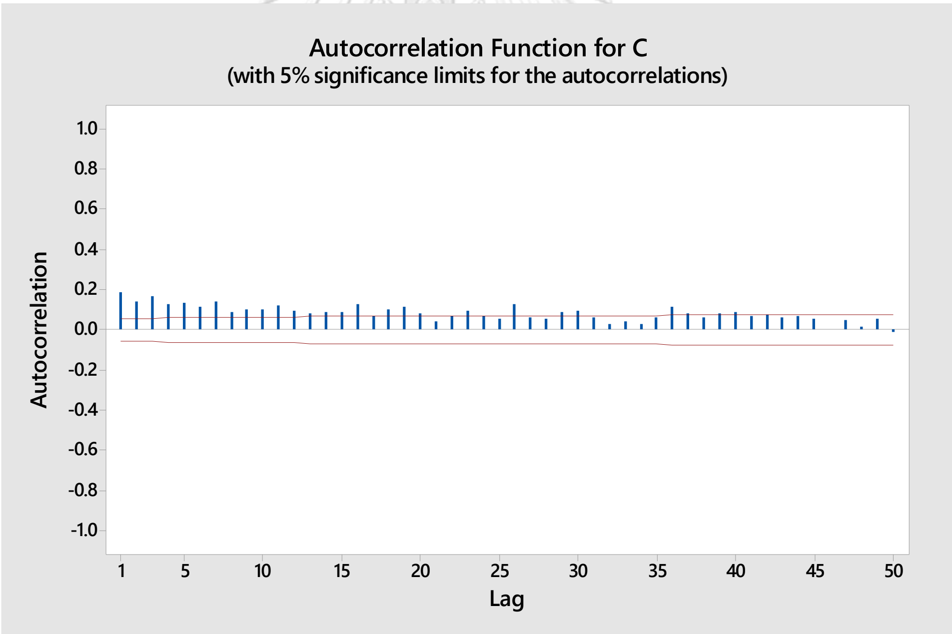


Figure 18: ACF plot of Tour C with 50 lags

Figures 13 and 14 show that tour A has a yearly pattern with 365 days period and weekly pattern with 7 days period. Figures 15, 16, 17 and 18 show that Tours B and C have a yearly pattern with 313 days period and weekly pattern with 6 days period because of from in lag 6 and 313 graph reached significant level.

3.1.2 External Variable

In this thesis, several models will use external factors data as predictive variables. Data is weekly collected on every Friday to forecast Saturday to the next Friday.

- Weekday to be a dummy variable due to the data have a weekly season then these weekday dummy variables will make the different weight of different day.

- Month to be a dummy variable because the data have a yearly season as weekly then the month dummy variables will help different for each month.

- The first 30-time lags in the ACF plot of Tours A, B and C reached the significant level that show these lags should have relation with the actual values, but the forecast takes place every Saturday. Hence, this time lag will be the data until Friday only. Lags 1-30 are used to forecast Saturday. Lags 2-31 are used to forecast Sunday. Lags 3-32 are used to forecast Monday. Lags 4-33 are used to forecast Tuesday. Lags 5-34 are used to forecast Wednesday. Lags 6-35 are used to forecast Thursday and lags 7-36 are tour used to forecast Friday as shown in Figure 19.

30 lag data						Forcasting Week						
Thursday	.....	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday
lag 1-30						Forcasting						
lag 2-31						lag 1	Forcasting					
lag 3-32						lag 2	lag 1	Forcasting				
lag 4-33						lag 3	lag 2	lag 1	Forcasting			
lag 5-34						lag 4	lag 3	lag 2	lag 1	Forcasting		
lag 6-35						lag 5	lag 4	lag 3	lag 2	lag 1	Forcasting	
lag 7-36						lag 6	lag 5	lag 4	lag 3	lag 2	lag 1	Forcasting

Figure 19: First 30 lags variables

- Actual booking data until Friday of the previous week before. This variable is chosen because most tourists booked tours before the tour dates then this variable should be very important variable to forecasting actual value. The preparation of this variable as examples shown in Figure 20.

Booking Data					Forcasting Week						
....	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday
Actual Booking Data until Friday					Forcasting						
Actual Booking Data until Friday					< 1 day data	Forcasting					
Actual Booking Data until Friday					< 2 day booking data		Forcasting				
Actual Booking Data until Friday					< 3 day booking data			Forcasting			
Actual Booking Data until Friday					< 4 day booking data				Forcasting		
Actual Booking Data until Friday					< 5 day booking data					Forcasting	
Actual Booking Data until Friday					< 6 day booking data						Forcasting

Figure 20: Actual booking data until Friday

From the variables above, they could be divided into 48 variables (17 dummy variables and 31 continuous variables). These variables will be used as x components in the SARIMAX model, input of the ANN model and the LSTM model.

After that the important variables are determined by stepwise method. The results show in Table 5 and the full stepwise steps is attached in the Appendix.

Table 5 : The important variables chosen by stepwise

Variables	Tour A	Tour B	Tour C
Weekday dummy variables	Tuesday Wednesday Thursday Friday Saturday	Tuesday Thursday Friday Saturday	Monday Friday Saturday
Month dummy variables	July November December	January March April December	January February May November December
First 30-time lags	Lags 1, 2, 3, 5 and 6	Lags 1, 2, 3, 4, 5, 6, 12, 24 and 28	Lags 1, 2, 3, 5, 11, 25, 26 and 28
Actual advance booking data	Chosen	Chosen	Chosen

3.1.3 Data preparation

According to the following method, the data is split into 2 sets for each model because this thesis uses 4 years (1,451 days) data as a training data set, half-year (182 days) as a cross-validation data set and another half-year (183 days) as a testing dataset. This thesis forecasts and changes parameters every week for 26 weeks. As a result, There are 26 models for each forecasting method as shown in Figures 21, 22 and 23.

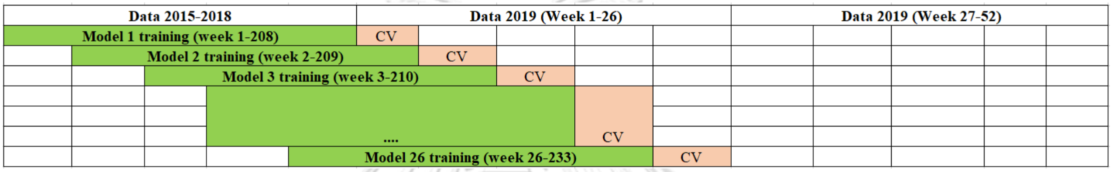


Figure 21: Split training and cross validation data set for 26 weeks (for time series model)

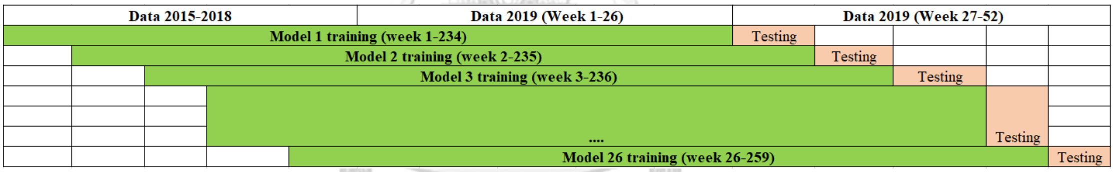


Figure 22: Split training and testing data set for 26 weeks (for time series model)

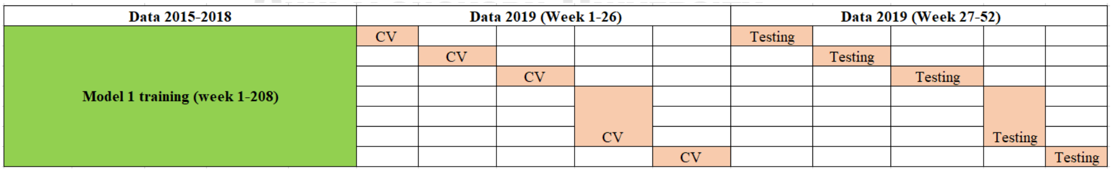


Figure 23: Split training, cross validation and testing data set for 26 weeks (for ANN and LSTM model)

Even though tours B and C operate 6 days a week, the models still have 26 weeks but the period change from 7 days to 6 days then training data will have 4 years (1,253 days include leap year) and test data have 1 year (156 days).

3.2 Forecasting Models

3.2.1 ARIMA

ARIMA models are famously used in time series forecasting. This is known as autoregressive integrated moving average models or ARIMA (p, d, q) models. P autoregressive, d is a number of times difference for its stationary, and q is moving average. This case study found a clearly seasonal component. This study suggests and focuses on seasonal autoregressive integrated moving average models or SARIMA (p, d, q) (P, D, Q) models which can be expressed as:

$$\Phi(L^s)\,\phi(L)\,\Delta^d\Delta_s^d y_t \;=\; \theta_0\Theta(L^s)\,\theta(L)\,\varepsilon_t \tag{3}$$

s is the seasonal length. In this study, s = 7 for weekly, s = 365.25 for a year for tour A, s = 6 for weekly and s = 313.25 for a year for Tours B and C. Daily data includes leap year. L is the lag operator.

Δ^d is the difference operator which d is the order of differencing

Δ_s^d is the order of seasonal differencing.

These different operators can be applied to find y_t transformed from non-stationary time series to stationary. Additionally, the data contains the type of seasonal component weekly and yearly. As a consequence, this study will focus on the SARIMA model (p, d, q) (P, D, Q) s along with exogenous variables or SARIMAX. SARIMAX can be expressed as:

$$\Phi(B)\phi P(1-B_s)Dy_t = c + X_t\beta + \theta_0 B\Theta Q(B_s)\varepsilon_t \tag{4}$$

B is the backshift operator (B$Y_t \;=\; Y_{t-1}$)

$\emptyset$ and θ are the autoregressive and moving averages coefficients, respectively. Φ and Θ are also autoregressive and moving averages coefficients, respectively but in a seasonal term,

β is an exogenous variable which is added from SARIMA (independent).

SARIMA and SARIMAX models can be identified using the following steps;

Step 1: Plot the time-series plots to examine seasonality and trend components or consider the type of seasonal component.

Step 2: If the data contains a trend or seasonal component from step 1, use seasonal and nonseasonal differencing to turn the data to stationary series.

Step 2.1: Only for SARIMAX, define extra seasonal components as exogenous.

Step 3: Plot the ACF and PACF after transforming to stationary data to consider initial p and q, respectively.

Step 4: Use the least-squares method to estimate the parameters to select the model.

Step 5: Test normalization of the residuals and autocorrelations by using the Ljung-box test.

3.2.2 TBATS

These following equations are TBATS models that are extended from the BATS model. This adaptation is called the TBATS model (equations 5 to 8) (De Livera, Hyndman, and Snyder 2011).

$$y_t^{(\omega)} = l_{t-1} + \phi b_{t-1} + \sum_{i=1}^{T} S_{t-1}^{(i)} + d_t \tag{5}$$

$$S_t^{(\omega)} = \sum_{i=1}^{k_i} S_{j,t}^{(i)} \tag{6}$$

$$S_{j,t}^{(i)} = S_{j,t-1}^{(i)} cos\lambda_j^{(i)} + S_{j,t-1}^{*(i)} sin\lambda_j^{(i)} + \gamma_1^{(i)} d_t \tag{7}$$

$$S_{j,t}^{*(i)} = -S_{j,t-1} sin\lambda_j^{(i)} + S_{j,t-1}^{*(i)} cos\lambda_j^{(i)} + \gamma_2^{(i)} d_t \tag{8}$$

k_i is the order of harmonics required for the i^{th} seasonal component.

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

$\gamma_1^{(i)}, \gamma_2^{(i)}$ are smoothing parameters.

BATS and TBATS models are estimated using these following steps (De Livera, Hyndman, and Snyder 2011)

Step 1: Due to the BATS model framework, there are 24 models to be considered for each series. These frameworks consist of 16 model combinations to consider each B, A, T, S components and 8 additional models that consider a damped trend component. Thus, all available (i.e., ϕ=1 if considered having no damping components have to be specified in the first step. ω = 1 as having no Box-Cox transformation. P = q = 0 as having no AR and MA residual adjustment is the considered model.) In the TBATS model, the seed state of state-space models is described as a random vector.

Step 2: Estimate the damping parameter, the Box-Cox parameter, coefficient of ARMA components and the smoothing parameters for initial states X_0 of models. These parameters are estimated by using three appropriate estimation criteria. These different estimation criteria are considered for non-linear optimization as follows:

(1) Maximize the log-likelihood of the estimates (MLE)

(2) Minimize the root mean square error of the original data (RMSE)

(3) Minimize the root mean square error of the transformed data ($RMSE_T$)

Step 3: Select the best available models by Akaike information criterion (AIC) to compare results among models. The following ARMA fitting follows these steps;

(1) Assume that an ARMA residual adjustment is not necessary by setting p = 0, q = 0.

(2) Explore the values of p and q in all possible ARMAs, and the ARMA (p, q) chosen can be minimized AIC.

The number of harmonics for TBATS models was selected by constantly adding harmonics, and by testing the significance using F-tests.

Step 4: Predict values by using the best model from the previous 3 steps.

3.2.3 ANN

Artificial Neural Network model used in this thesis would be explained by these following parameters:

The loss function or Cost function is how the model computes the error by comparing predicted values which are predicted from the model and the actual values. Then, the Gradient which depends on weight and bias is calculated by a backpropagation method to make Gradient descent that can lower loss or error. Nowadays, the ANN model has many types of loss functions which depend on the application to match and make the best result. This thesis uses Mean Square Error (MSE) as a loss function because MSE has a slope that can change due to the error. Gradient will be low with low errors while high errors make high Gradient. Thus, this is the advantage. MSE is expressed as:

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(A_t - F_t)^2 \quad (9)$$

- n is number of samples to measurement

- A_t is actual value

- F_t is forecast value

Figure 5 shows the overview of the ANN model. Figure 24 (TowardsDataScience 2017, August 16) shows the components of each hidden unit (cell).

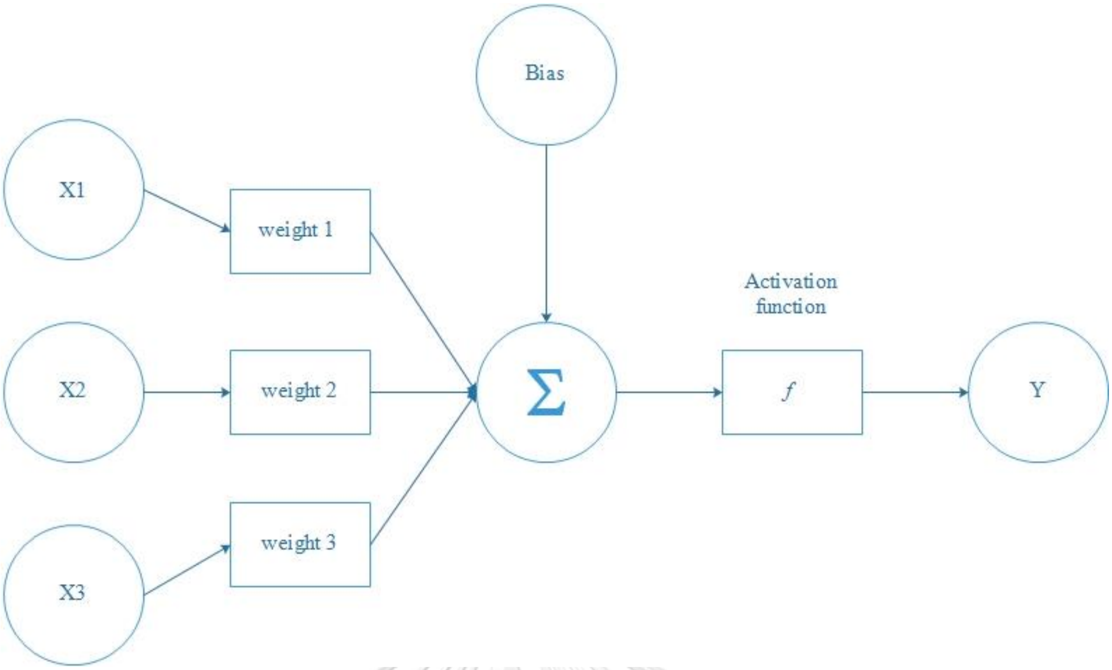


Figure 24: Structure of ANN hidden unit (cell) (TowardsDataScience 2017, August 16)

The equation of every hidden could be calculated by following equation.

$$a_j^i = \sigma\left(\left(\sum_m w_{jk}^i a_k^{i-1}\right) + b_j^i\right) \tag{10}$$

a_j^i is the activation (output) of the neuron j^{th} in layer i^{th}

σ is the activation function

w_{jk}^i is the weight of the neuron k^{th} from previous neuron in layer $(i-1)^{th}$ to the neuron j^{th} in the layer i^{th}

b_j^i is a bias of the neuron j^{th} in layer i^{th}

Activation function

Most research related to the ANN model uses Sigmoid Function and Rectified Linear Unit (ReLU).

-The sigmoid function is the function with S-Curve that can clearly be explained and has output range between 0-1 as shown in the Figure 25.

$$S(x) = 1/(1 + e^{-x}) \tag{11}$$

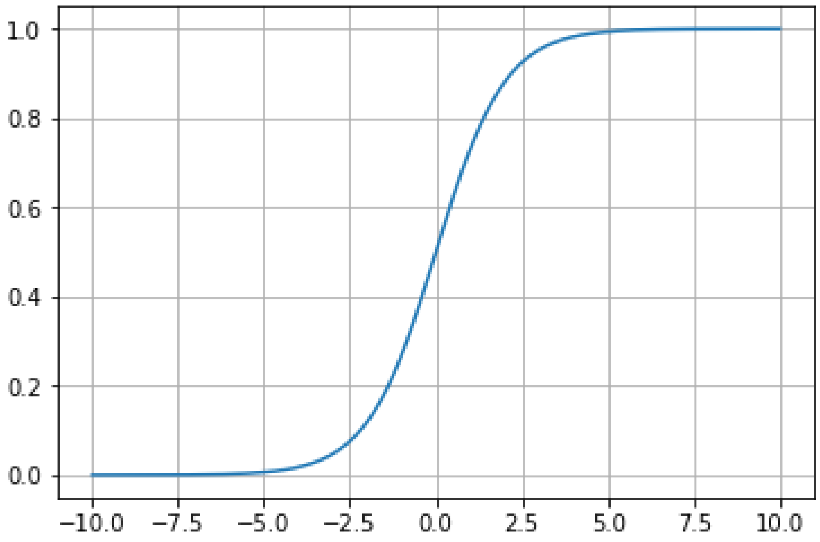


Figure 25: Sigmoid function curve

- ReLU has Slope = 1 when the input has a positive value that can fix the Vanishing Gradient problem as shown in Figure 26.

$$R(x) = \max\,(0, x) \tag{12}$$

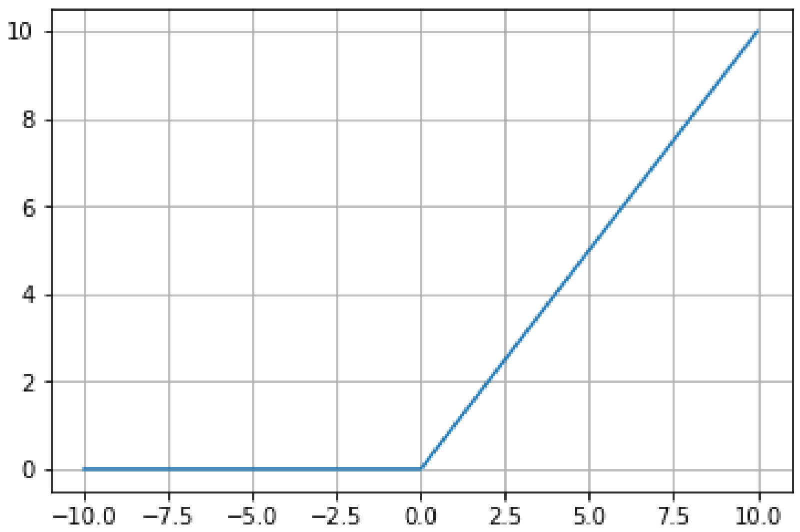


Figure 26: Rectified Linear Unit plot

Train the ANN model by following these steps;

Step 1: Randomize the initial weights (fix seed).

Step 2: Implement forward propagation.

Step 3: Implement the loss function.

Step 4: Implement backpropagation to compute partial derivatives.

Step 5: Use “adam” algorithm optimization (An algorithm based on the optimization of stochastic objective functions with little memory requirements and are well suited for the large number of data with the noise problem) to minimize the cost function (Kingma and Ba 2014) .

Computation code of ANN in python is shown in the appendix.

3.2.4 LSTM

LSTM is one of the RNN types (recurrent neural network) which is a Neural Network. The difference between ANN and LSTM is the hidden units (cells). LSTM cell structure is shown in Figure 27 (colah'sblog 2015,August 27).

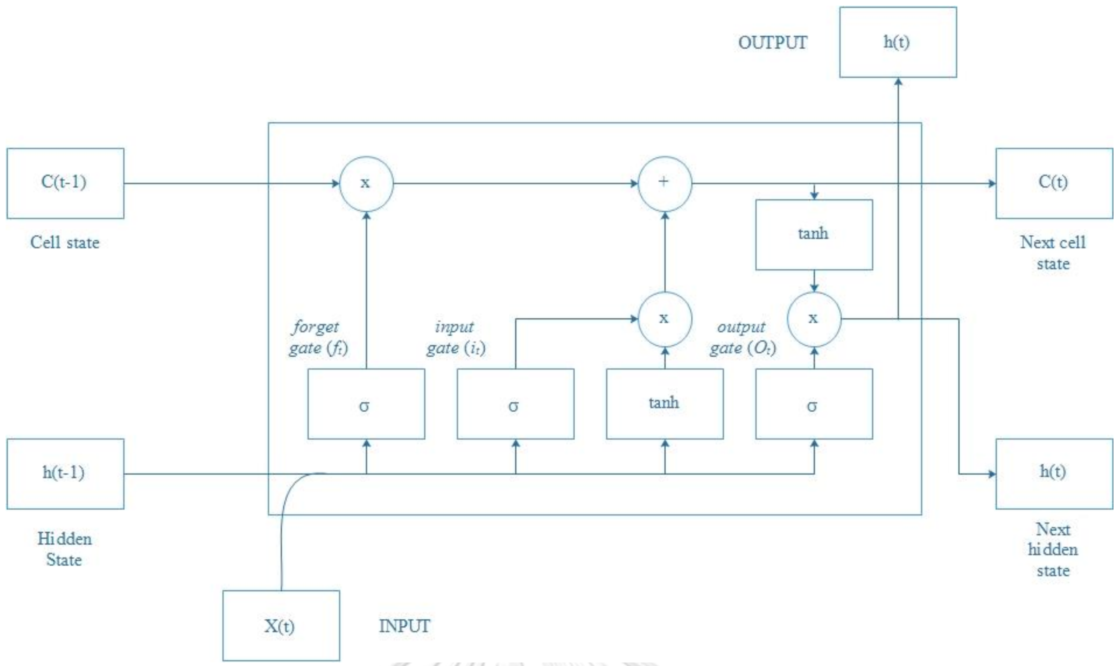


Figure 27: Structure of LSTM hidden unit (cell) (colah'sblog 2015,August 27)

Forget Gate (f_t) using sigmoid function to control forgetting previous data is shown in the equation (13).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{13}$$

Input Gate using sigmoid function to decide which value will be updated by combining with $\tilde{C}_t$ is shown in the equations (14) and (15).

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{14}$$

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{15}$$

Updated cell state is shown in the equation (16).

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{16}$$

Output Gate (O_t) uses sigmoid function to decide parts of the cell state to be an output (17).

$$O_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{17}$$

Updated cell state uses tanh made value between −1 to 1 and multiplies by the output gate (18)

$$h_t = O_t \cdot tanh(C_t) \tag{18}$$

Train LSTM model by following these steps;

Step 1: Create sequential data from time-series data.

Step 2: Randomize the initial weights (fix seed).

Step 3: Implement forward propagation.

Step 4: Implement the loss function.

Step 5: Implement backpropagation to compute partial derivatives.

Step 6: Use “adam” algorithm optimization (An algorithm based on the optimization of stochastic objective functions which had little memory requirements and are well suited for the large number of data with the noise problem) to minimize the cost function (Kingma and Ba 2014).

Computation code of LSTM in python is shown in the appendix.

Chapter IV: Results and Discussion

4.1 Results

This thesis consists of two types of forecasting models (Time series models and Machine Learning models). Time series models use a sliding method to separate the data into a training set and a cross-validation set. This method makes 26 sets of data for 26 models to forecast each weekly demand of weeks 1 to 26 (first half) of 2019. Due to the advantage of Machine Learning models that can make a good prediction accuracy in short and middle-range periods whereas Time series models can make a good prediction accuracy only in short-range period, the first experiment of ANN models is made to compare the forecasts of cross validation set by using a sliding method (26 models) and a long-range periods model that forecasts 26 weeks ahead. The next experiment is to compare the forecasting accuracy between each activation function in the structure of the models. For LSTM models, the experiment compares only the activation functions because LSTM is a huge model with many parameters that cannot perform 26 models in a single time. Finally, the models which make the most accuracy for the cross-validation set will be chosen to forecast the test set (weeks 27-52) of 2019 to determine the general error of the model.

4.1.1 Same Day Last Year

Figures 28, 29 and 30 show the forecasts of Tours A, B and C of the Same Day Last Year method which is the currently used model and will be used at a benchmark for all forecasting models.

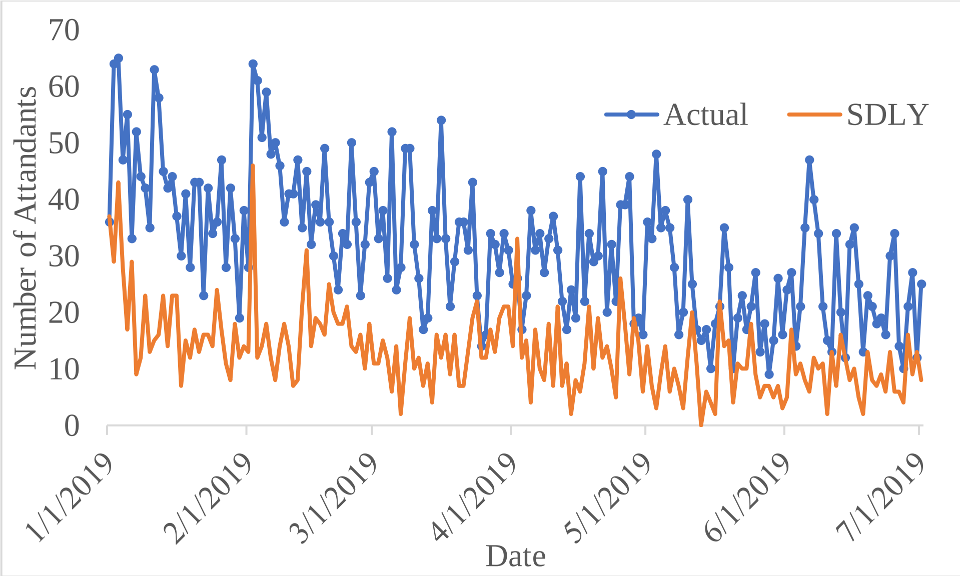


Figure 28: Same Day Last Year forecasts compared with the actual value of Tour A

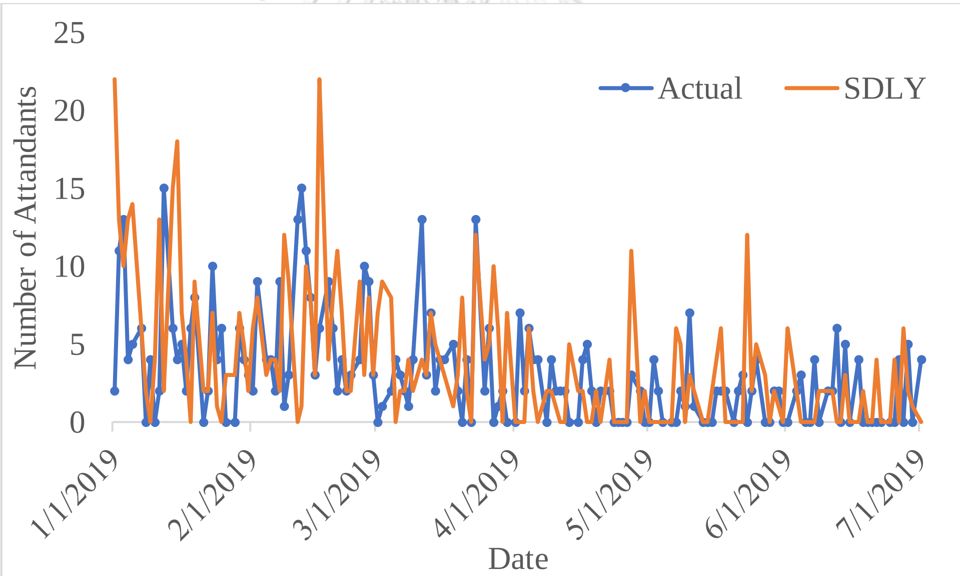


Figure 29: Same Day Last Year forecasts compared with the actual value of Tour B

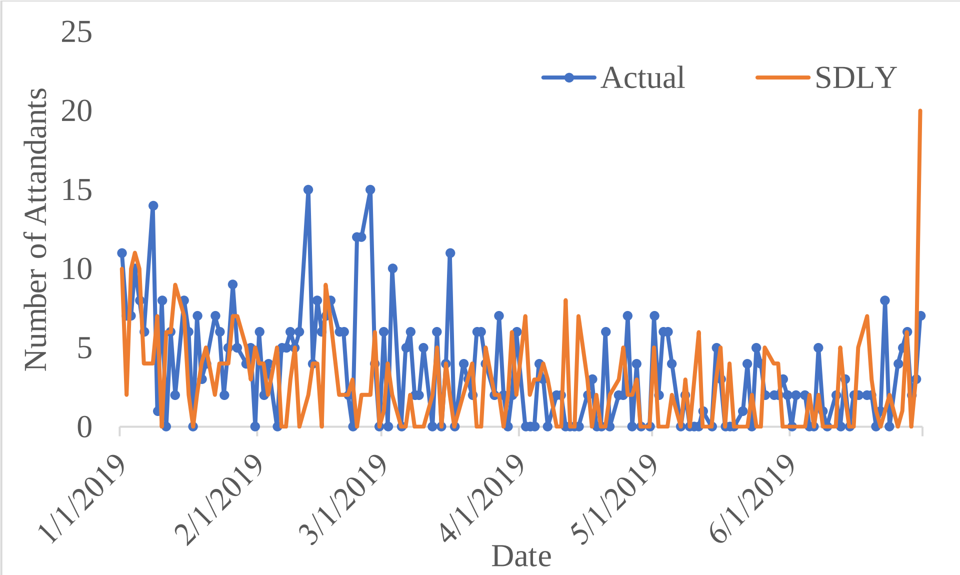


Figure 30: Same Day Last Year forecasts compared with the actual value of Tour C

4.1.2 Seasonal ARIMA Model

Figures 31, 32 and 33 show the actual values and the predicted values of Tours A, B and C respectively by the SARIMA model on the cross validation set (182 days from January 2019 to 1st July, 2019 for Tour A, and 156 days for Tours B and C). Tables 18, 19 and 20 show that the Mean Absolute Error of Tours A, B and C using the SARIMA model can be reduced from 18.648, 3.179 and 2.949 to 8.28, 2.519 and 2.397 respectively, which are less than the Same Day Last Year method (the model currently used by the company).

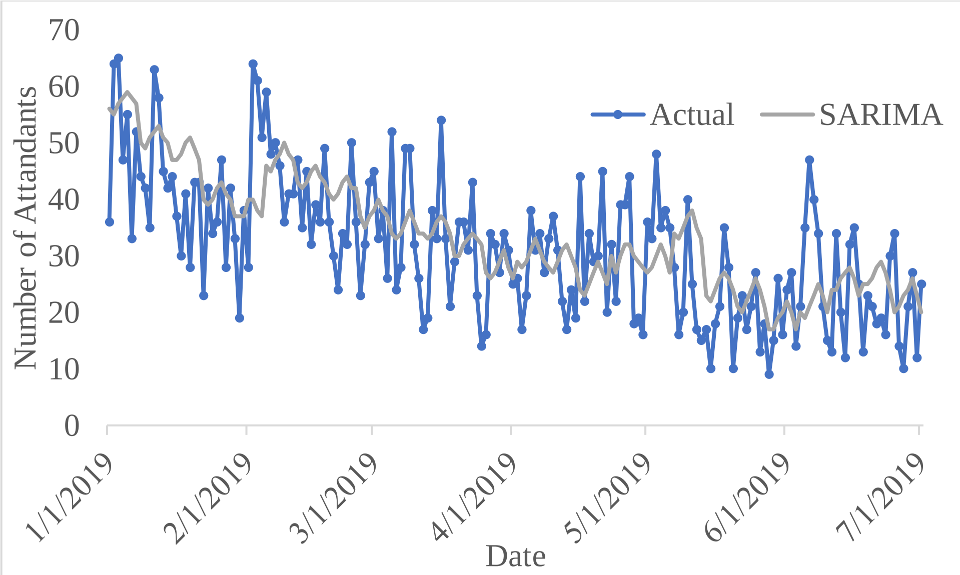


Figure 31: Seasonal ARIMA forecasts compared with the actual value of tour A

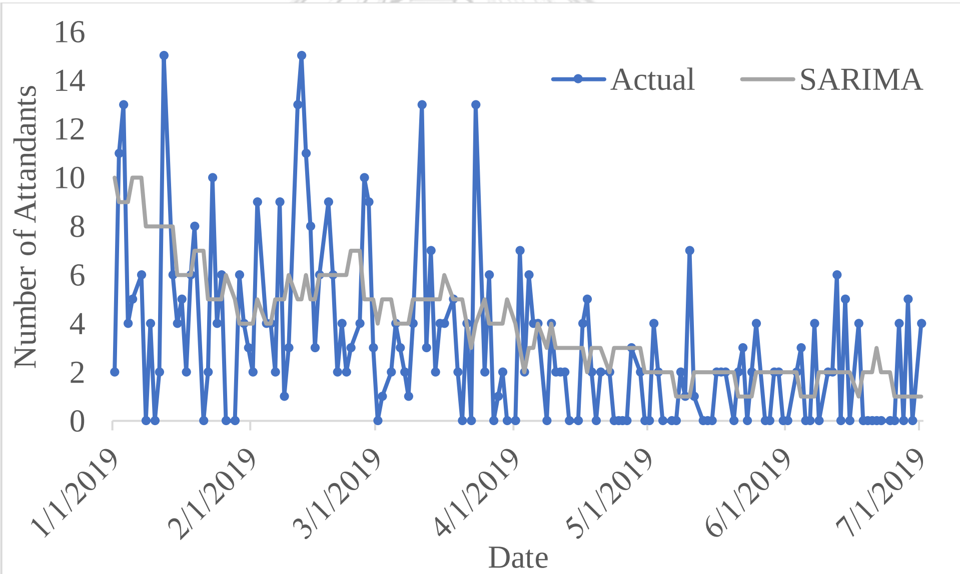


Figure 32: Seasonal ARIMA forecasts compared with the actual value of Tour B

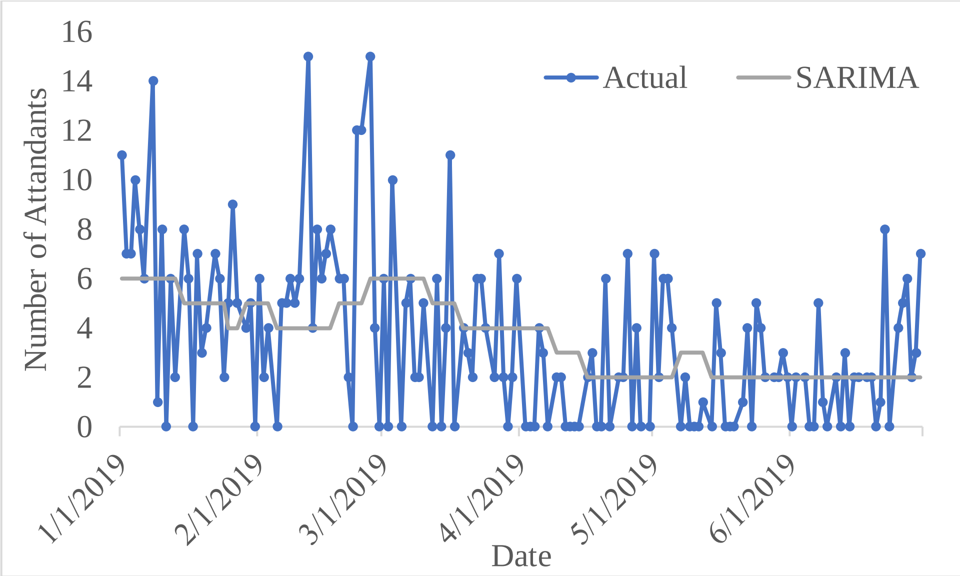


Figure 33: Seasonal ARIMA forecasts compared with the actual number of Tour C

Figure 34 shows the flowchart of SARIMA parameter selection to forecast week by week until week 26. Because every model has different training data by sliding method that make 26 sets of parameters, Autoregressive, moving average, and integrated factor for both trend and seasonal components (p, d, q, P, D, Q) of all models are adjusted to make the lowest AIC and MAPE.

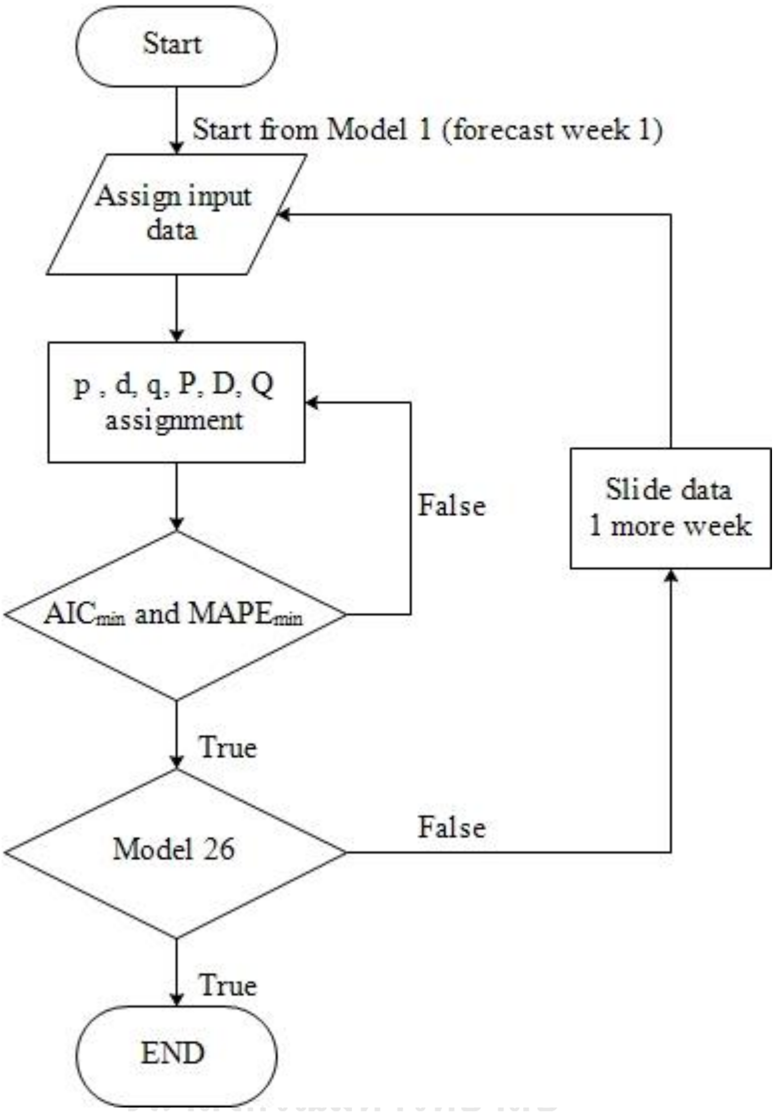


Figure 34: Flowchart showing SARIMA parameter selection

4.1.3 Seasonal ARIMAX Model

SARIMAX models in this part are separated into two parts. The SARIMAX model with Fourier variables for capturing yearly seasonal and the SARIMAX model with external variables is explained in the Chapter 3.

4.1.3.1 Seasonal ARIMAX Model with Fourier variables

Figures 35, 36 and 37 show the actual values and the predicted values of Tours A, B, and C by the SARIMAX model with Fourier variables on the cross validation set (182 days from January 2019 to 1st July, 2019 for Tour A, and 156 days for Tours B and C). Tables 18,19 and 20 show the reduction of Mean Absolute Error of Tours A, B and C from 18.648, 3.179 and 2.949 to 8.077, 2.436 and 2.327 respectively. The reduced numbers are the comparison of MAE using the SARIMAX model and the Same Day Last Year method.

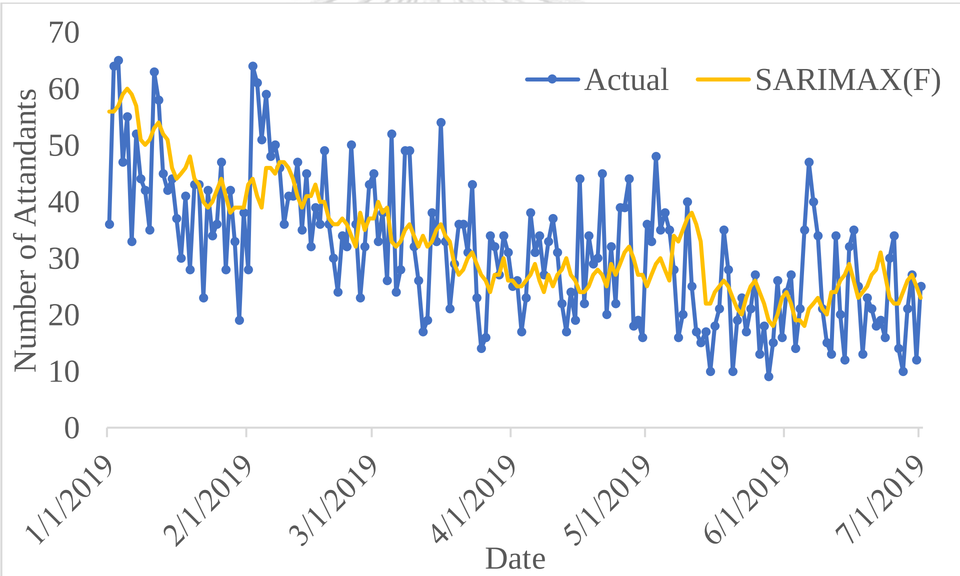


Figure 35: SARIMAX forecasts with Fourier variables compared with the actual value of Tour A

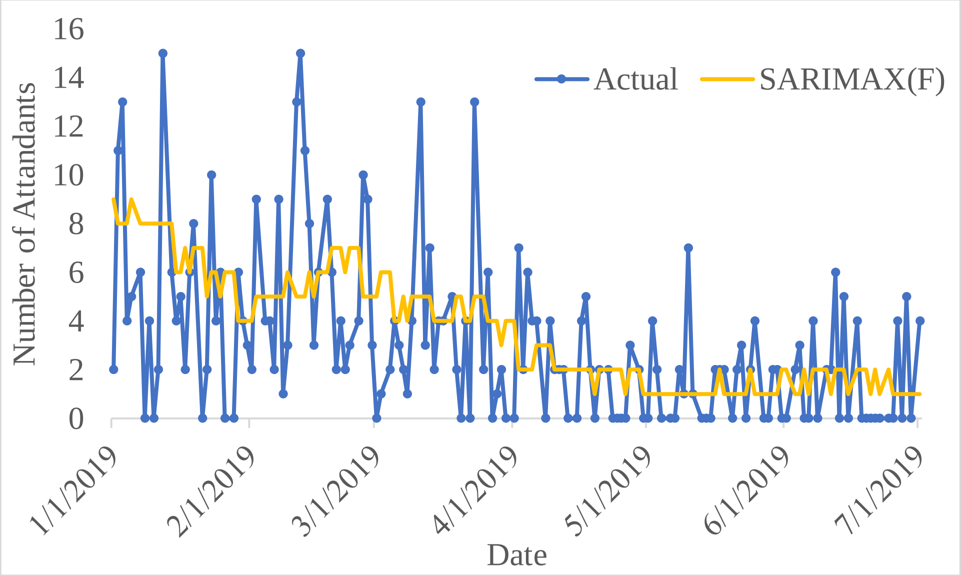


Figure 36: SARIMAX forecasting with Fourier variables compared with the actual value of Tour B

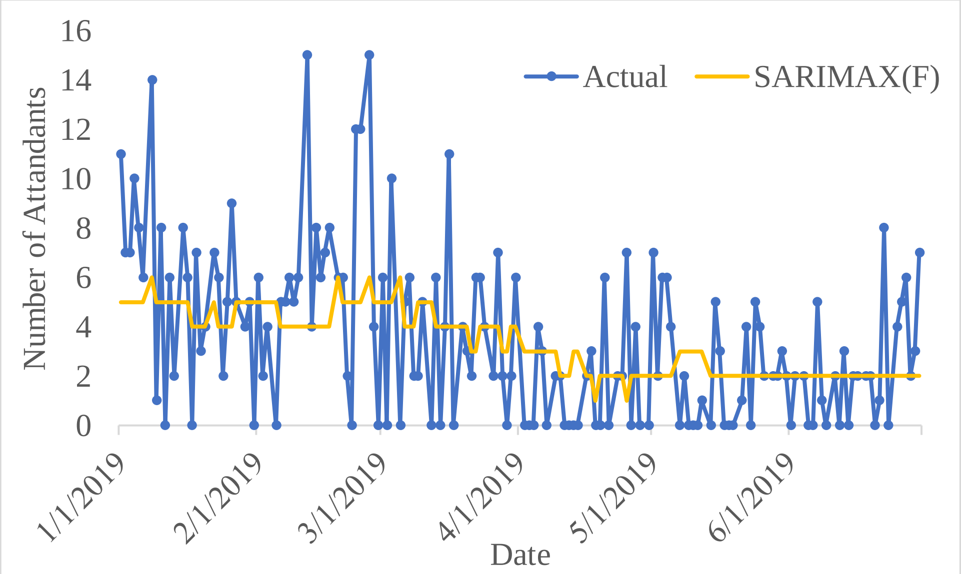


Figure 37: SARIMAX forecasts with Fourier variables compared with the actual value of Tour C

4.1.3.2 Seasonal ARIMAX Model with External Variables

Figures 38, 39 and 40 show the actual values and the predicted values of Tours A, B and C by the SARIMAX model with external variables on the cross validation set (182 days from January 2019 to 1st July, 2019 for Tour A, and 156 days for Tours B and C). The reductions of Mean Absolute Error of Tours A, B and C using the SARIMAX model compared to the Same Day Last Year method which are from 18.648, 3.179 and 2.949 to 4.275, 1.179 and 1.487 respectively is shown in Tables 17,18 and 19.

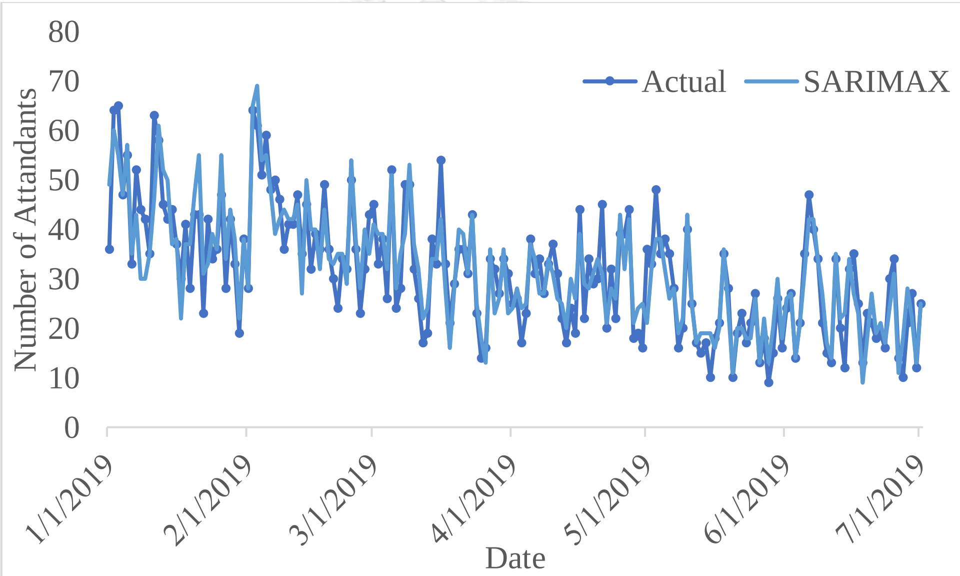


Figure 38: SARIMAX forecasts with external variables compared with the actual value of Tour A

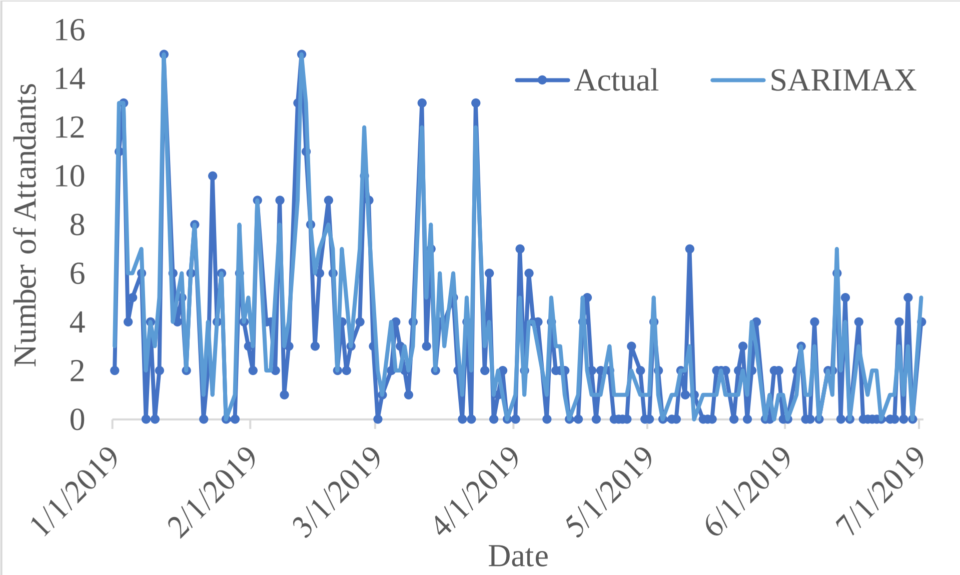


Figure 39: SARIMAX forecasts with external variables compared with the actual value of Tour B

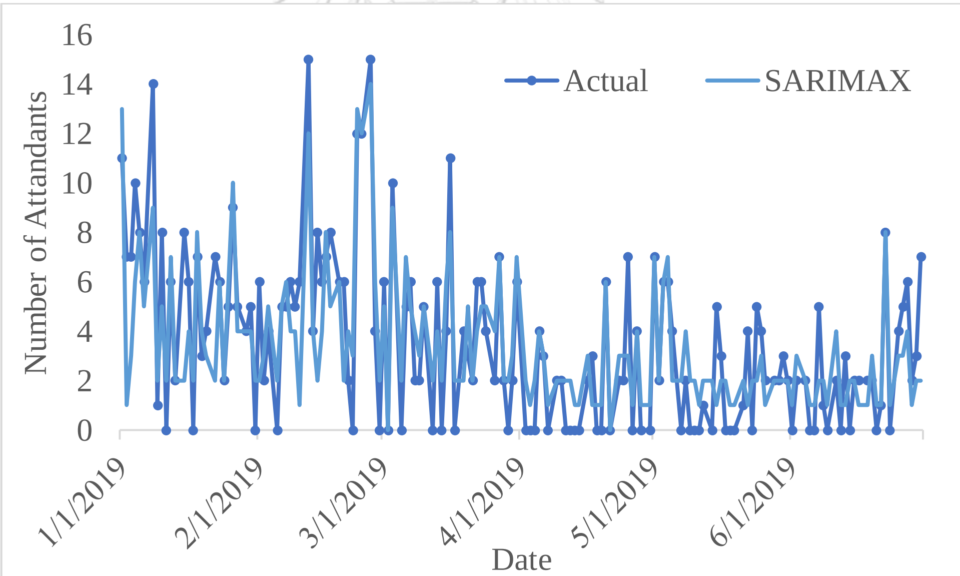


Figure 40: SARIMAX forecasts with external variables compared with the actual value of Tour C

Figure 41 shows the flowchart of each SARIMAX model parameter selection to forecast week by week until week 26. Because every model has different training data and external variables by sliding method that make 26 sets of parameters, Autoregressive, moving average, and integrated factor for both trend and seasonal components (p, d, q, P, D, Q) of all models are adjusted to make the lowest AIC and MAPE.

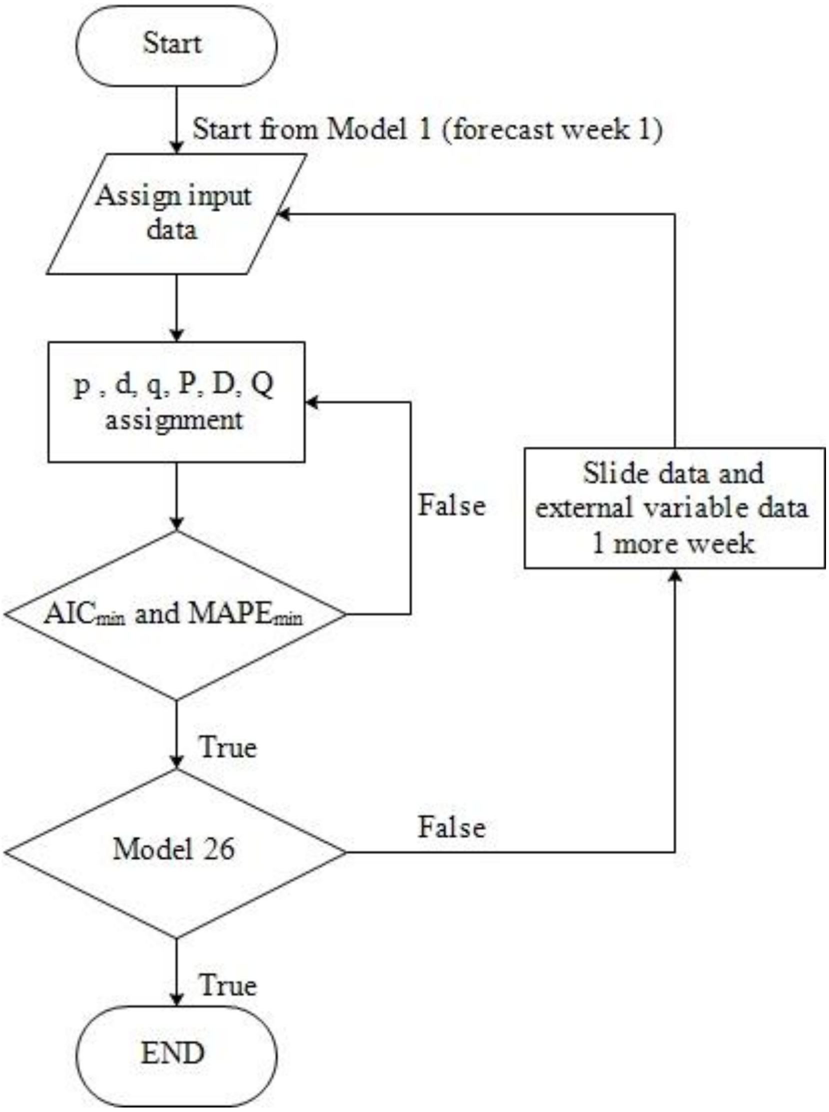


Figure 41: Flowchart showing each SARIMAX model parameter selection

Table 6: Comparison of measurement for SARIMA with exogenous variables

	Tour	SARIMAX (Fourier variables)	SARIMAX (External variables)
MAE	Tour A	8.077	4.275
	Tour B	2.436	1.179
	Tour C	2.327	1.487
RMSE	Tour A	9.835	5.497
	Tour B	3.184	1.573
	Tour C	3.043	1.971
MAPE	Tour A	31.51%	15.19%
	Tour B	-	-
	Tour C	-	-

Figures 42-44 show the actual values and the predicted values and Table 6 show the error of different X variables of Tours A, B and C by the SARIMAX model with Fourier variables and the SARIMAX model with external variables on the cross validation set (182 days from January 2019 to 1st July, 2019 for Tour A, and 156 days for Tours B and C). According to Table 5, the accuracy of all Tours can be improved more than 50%. The main difference of these models is the exogenous variable that the first model uses second-order Fourier as variables to capture another seasonal period while the second model uses advance amount of booking until every Friday, historical data at time lag 1st-7th and dummy variables for weekdays and months.

4.1.4 TBATS Model

Figures 45, 46 and 47 show the actual values and the predicted values of Tour A by the TBATS model on the cross validation set (182 days from January 2019 to 1st July, 2019 for Tour A, and 156 days for Tours B and C). Tables 18,19 and 20 show the reductions of Mean Absolute Error of Tours A, B and C using the SARIMAX model compared to the Same Day Last Year method which are from 18.648, 3.179 and 2.949 to 8.308, 2.410 and 2.423 respectively.

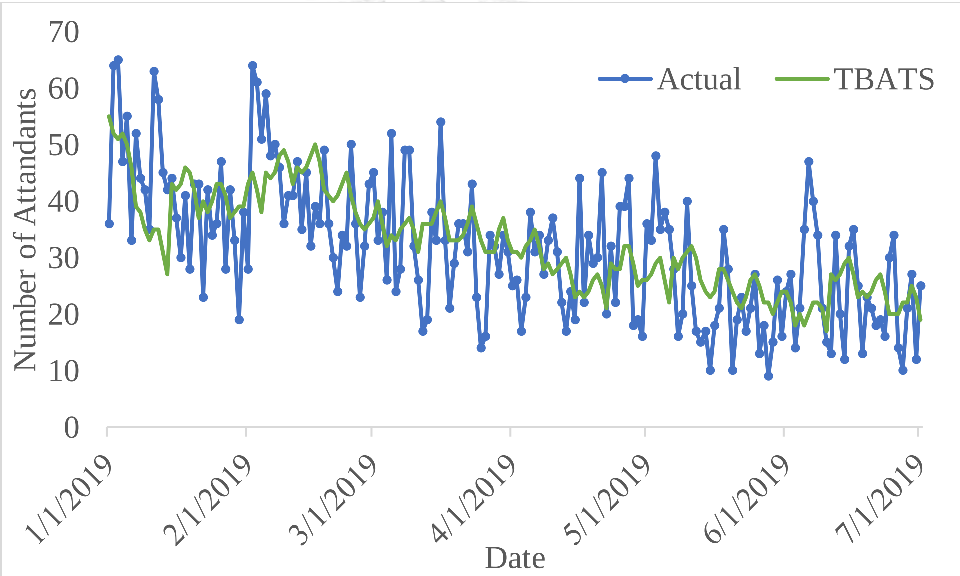


Figure 42: TBATS forecasts compared to the actual value of Tour A

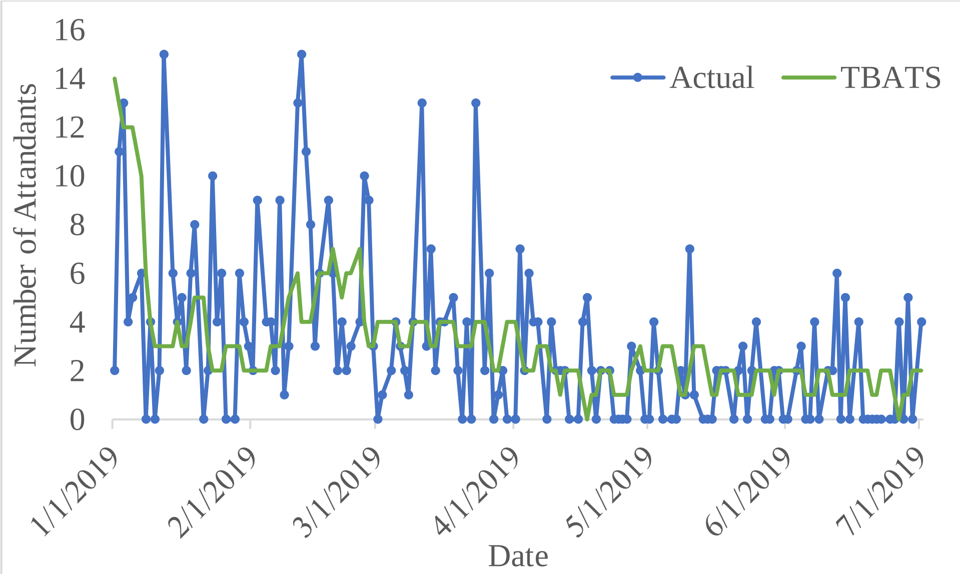


Figure 43: TBATS forecasts compared to the actual number of Tour B

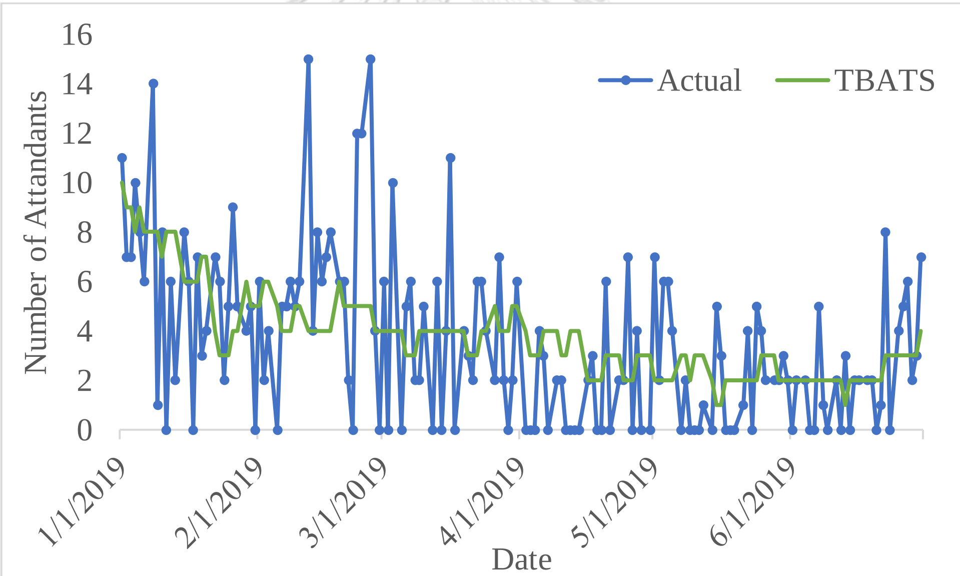


Figure 44: TBATS forecasts compared to the actual value of Tour C

Figure 45 shows the flowchart of TBATS model parameter selection to forecast week by week until week 26. Because every model has different training data by sliding method that make 26 sets of parameters (α, β, γ), box-cox transformation parameter (ω), dampening parameter (ϕ), and auto regressive. Moving average component (p, q) of all models are adjusted to make the maximum log likelihood of the estimates (MLE), the minimum Root Mean Square Error of the original data

($RMSE$) and transformed data ($RMSE_T$),the lowest Akaike information criterion (AIC), and MAPE.

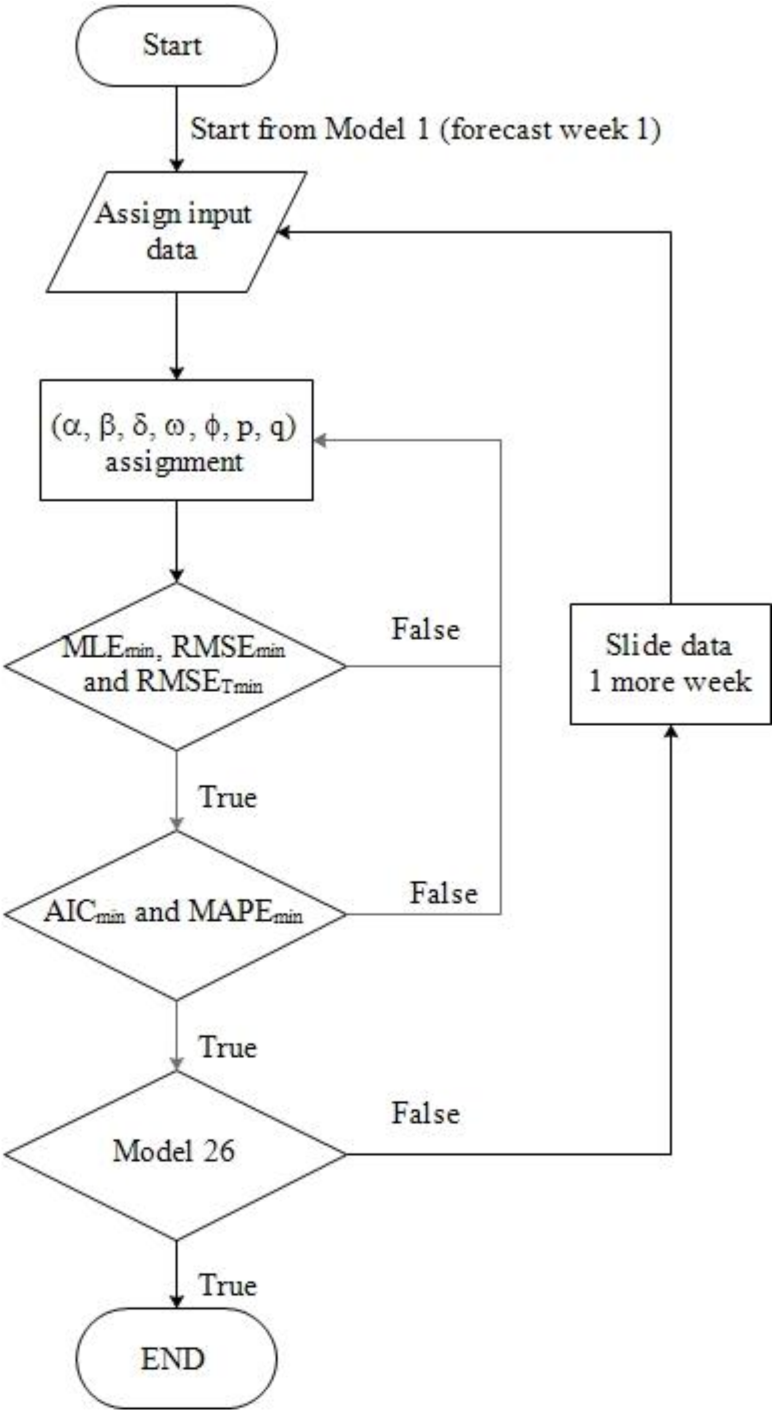


Figure 45: Flowchart showing each TBATS model parameter selection

4.1.5 ANN Models

Since ANN models are also well for predict middle range periods so the next experiment will focus on 1 model to forecast 26 weeks in a single time. The straight benefit of 1 model compare with 26 models can reduce time usage to train model 26 times. Shown in Figure 49.

ANN model structure has many parameters that can adjust to find the minimum Mean Absolute Error (MAE).

Parameters:

1. Batch size is the number of samples which is fed to train the model. The batch size ranges from 1 to n (number of training samples).

- batch size = 1 (stochastic gradient descent) means that the model will update parameters from the backpropagation process. Every single sample fitting will require low memory usage.

- 1 < batch size < n (mini-batch gradient descent) is very famous in the present day. Not only is it more accurate than the full batch size due to its parameter update frequency and lower memory requirement, but also uses less time to train the model than the batch size =1.

- batch size = n (batch gradient descent) is the fastest method. It works well with a small number of samples. However, it will need high memory usage with a large number of samples.

The following experiment compares the number of batch sizes from 1 to 10 by fixing a number of epochs at 1 and 50 along with a number of hidden units at 10 and 100.

Table 7: Comparison of batch sizes with fixed epoch at 1 for tour A

Epoch = 1					
10 hidden units	Batch size	MAE	RMSE	MAPE	Time usage (µs/sample)
	1	5.876	8.228	56.571	800-1000
	2	5.999	8.402	58.172	400-550
	3	6.066	8.483	59.039	260-300
	4	6.071	8.540	58.433	200-220
	5	6.098	8.852	52.690	160-180
	6	6.977	10.143	47.722	130-160
	7	8.619	11.982	50.700	115-130
	8	10.352	13.681	58.866	95-115
	9	11.521	14.776	66.013	85-100
	10	12.500	15.671	73.032	80-95
100 hidden units	Batch size	MAE	RMSE	MAPE	Time usage (µs/sample)
	1	4.502	5.895	46.083	800-1000
	2	4.737	6.305	47.428	400-550
	3	4.957	6.670	48.728	260-300
	4	5.135	6.979	50.405	200-220
	5	5.254	7.265	49.739	160-180
	6	5.349	7.476	49.328	130-160
	7	5.414	7.624	48.977	115-130
	8	5.496	7.742	49.482	95-115
	9	5.550	7.833	51.031	85-100
	10	5.625	7.947	51.786	80-95

Table 8: Comparison of batch sizes with fixed epoch at 50 for tour A

Epoch = 50					
10 hidden units	Batch size	MAE	RMSE	MAPE	Time usage (µs/sample)
	1	2.913	3.878	26.159	800-1000
	2	3.113	4.183	28.235	400-550
	3	3.100	4.067	29.309	260-300
	4	3.086	4.040	29.952	200-220
	5	3.175	4.128	31.280	160-180
	6	3.125	4.062	30.413	130-160
	7	3.270	4.267	32.155	115-130
	8	3.283	4.217	32.814	95-115
	9	3.303	4.237	33.172	85-100
	10	3.379	4.334	33.519	80-95
100 hidden units	Batch size	MAE	RMSE	MAPE	Time usage (µs/sample)
	1	1.674	2.241	14.426	800-1000
	2	1.818	2.374	17.815	400-550
	3	1.711	2.319	15.461	260-300
	4	1.848	2.491	16.513	200-220
	5	1.894	2.549	17.352	160-180
	6	2.007	2.671	18.664	130-160
	7	2.033	2.688	18.656	115-130
	8	2.142	2.831	21.074	95-115
	9	2.169	2.847	21.337	85-100
	10	2.143	2.845	19.856	80-95

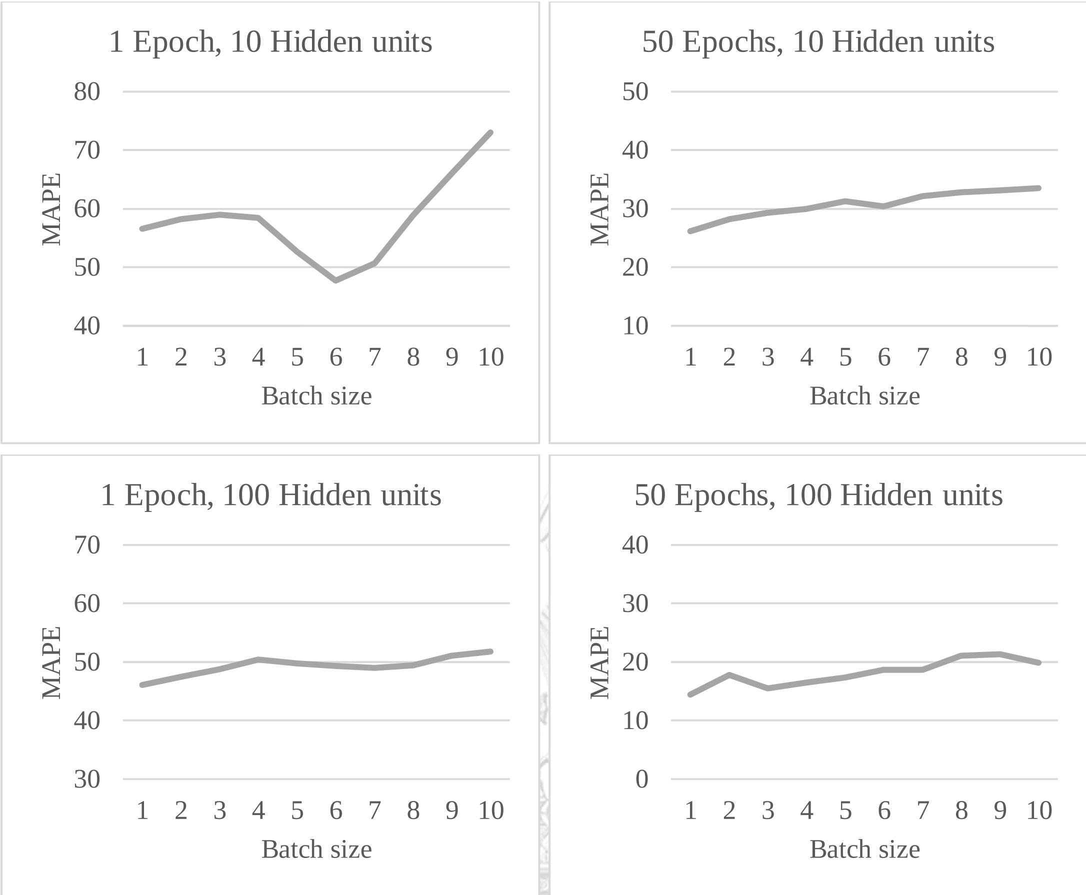


Figure 46: Comparison of batch sizes with fixed epochs and hidden units for Tour A

Tables 7, 8 and Figure 46 show the error comparison in using different batch sizes (1 to 10) which consist of two types of batch size (batch size 1 is stochastic gradient descent and batch sizes 2 – 10 is mini-batch gradient descent). At low epoch, errors will be increased due to the higher numbers of batch size. Yet, errors at the same level happened at high epoch. Also, the increasing batch size can reduce plenty of time usage. According to the results, it is concluded that mini-batch gradient descent with high epoch is more suitable for this thesis than the stochastic gradient descent. As a consequence, batch sizes 10, 20, 32, and 64 will be used in this thesis.

2. Epoch (iteration) is the number of complete passes through the training set. Epoch 10, 30, 50, 70 and 100 are used in the experiments.

3. 10, 30, 50, 70 and 100 hidden units are the numbers of hidden units in each hidden layer used in the experiments.

4. 1-2 hidden layers are the numbers of hidden layers used in the experiments.

5. An activation function is applied in hidden and output layers; Sigmoid and ReLU functions are used in hidden layers, but only ReLU is used in the output layer.

According to the above parameter settings, ANN model structure will have 11,520 possible combinations; (4 sets of batch size * 5 sets of epochs * 96 (5-100) sets of hidden units * ((1 hidden layer * 2 activation functions) + (2 hidden layers * 2 activation functions)). It will take a very long time to train all the combinations. Therefore, the experiments are divided into 2 main steps as shown in Figure 47.

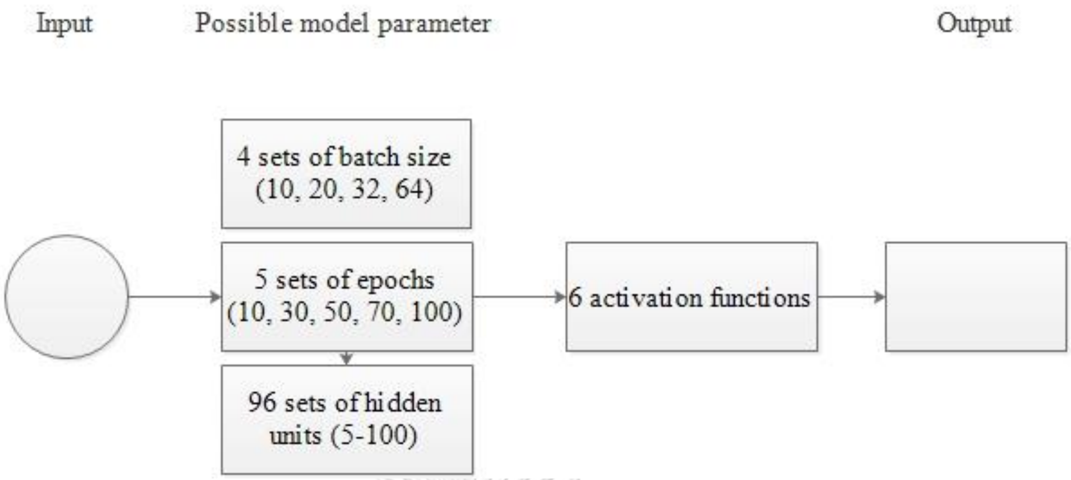


Figure 47: 2 steps to find the best parameters

The first step is to find the best match of batch size and epoch with sample number of hidden units for each combination (4 sets of batch size * 5 sets of epochs * * 96 (5-100) sets of hidden units). The best match for every combination of activation function and hidden layers are shown in the following figure.

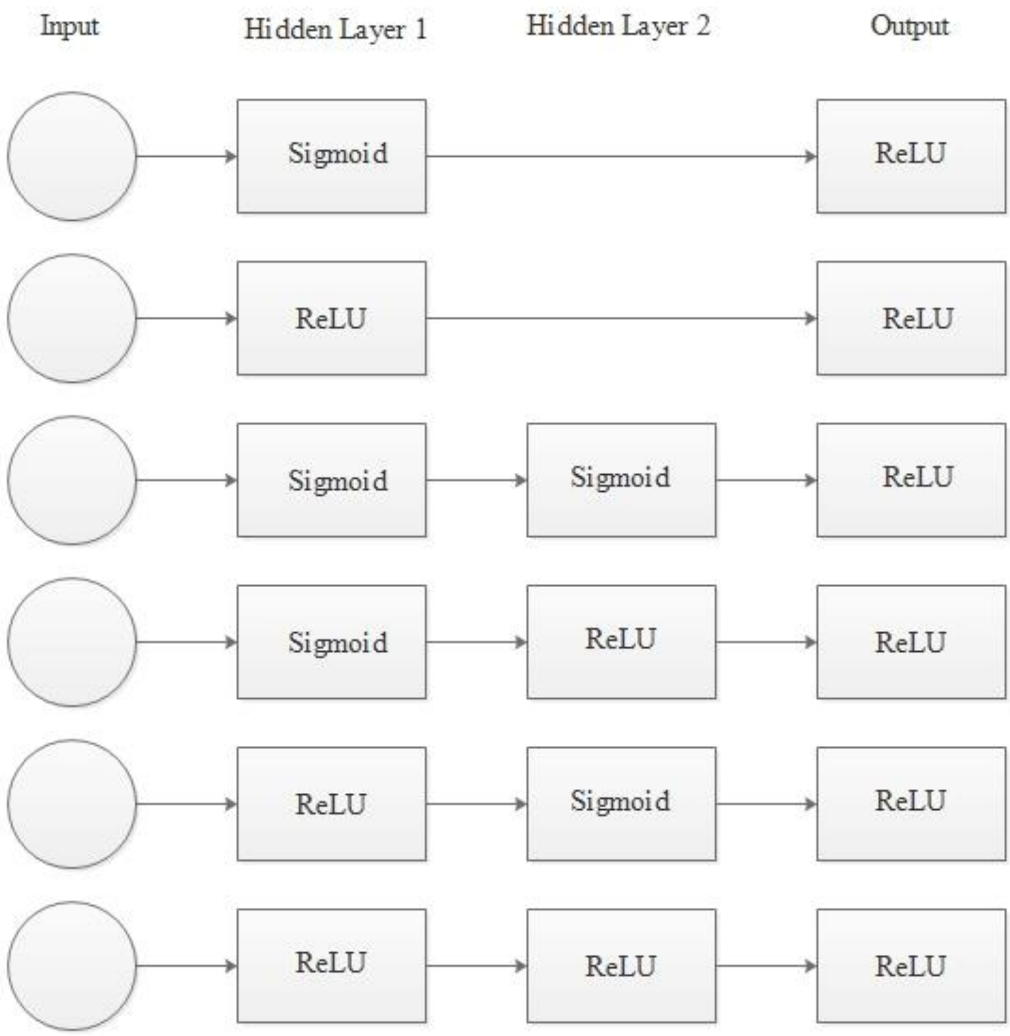


Figure 48: All possible combinations between activation function and hidden layers

Step 1: Choose batch size, epoch and number of hidden units which makes the minimum MAE.

Step 2: After getting all parameters, use them to find the best type of activation function from 6 possible combinations which are shown in Figure 51.

Table 9 : The results of all possible structure for training ANN model Tour A

Model Structure	Seed	Batch size	Epoch	Hidden layer	Hidden unit	MAE	RMSE	MAPE
ReLU	1	10	100	1	100	2.676	3.593	23.58%
Sigmoid	1	10	70	1	70	3.190	4.300	28.59%
ReLU, ReLU	1	10	100	2	100	1.263	1.691	11.88%
Sigmoid, Sigmoid	1	10	50	2	98	3.278	4.256	33.07%
ReLU, Sigmoid	1	10	100	2	93	1.827	2.445	15.81%
Sigmoid, ReLU	1	10	50	2	91	3.272	4.251	32.06%

After all models were constructed, the final step is to recheck random seeds to find which one is better than the initial seed (1).

Table 10: Comparison of numbers of different seeds with 2 ReLU layer, 10 batch size, 100 epoch and 100 hidden units

SEED	MAE	MASE	MAPE
1	1.263	1.691	11.88%
2	1.105	1.555	9.72%
3	1.665	2.232	16.45%
4	1.769	2.332	18.02%
5	1.384	1.874	12.34%
6	1.282	1.740	11.87%
7	1.403	1.927	12.54%
8	1.365	1.835	12.15%
9	1.120	1.522	10.34%
10	1.577	2.018	15.86%
20	1.516	2.055	13.78%
30	1.673	2.153	15.82%
40	15.635	18.541	100.0%
50	1.400	1.868	14.15%
60	1.424	1.871	14.41%
70	1.491	2.012	13.14%
80	1.463	1.970	13.48%
90	1.456	1.961	12.98%
100	1.366	1.876	12.67%

From the result in Tables 9 and 10, The best structure of ANN model for forecasting 26 weeks ahead is 2 hidden layers with the ReLU activation function which made 1.105 MAE and 9.72% MAPE.

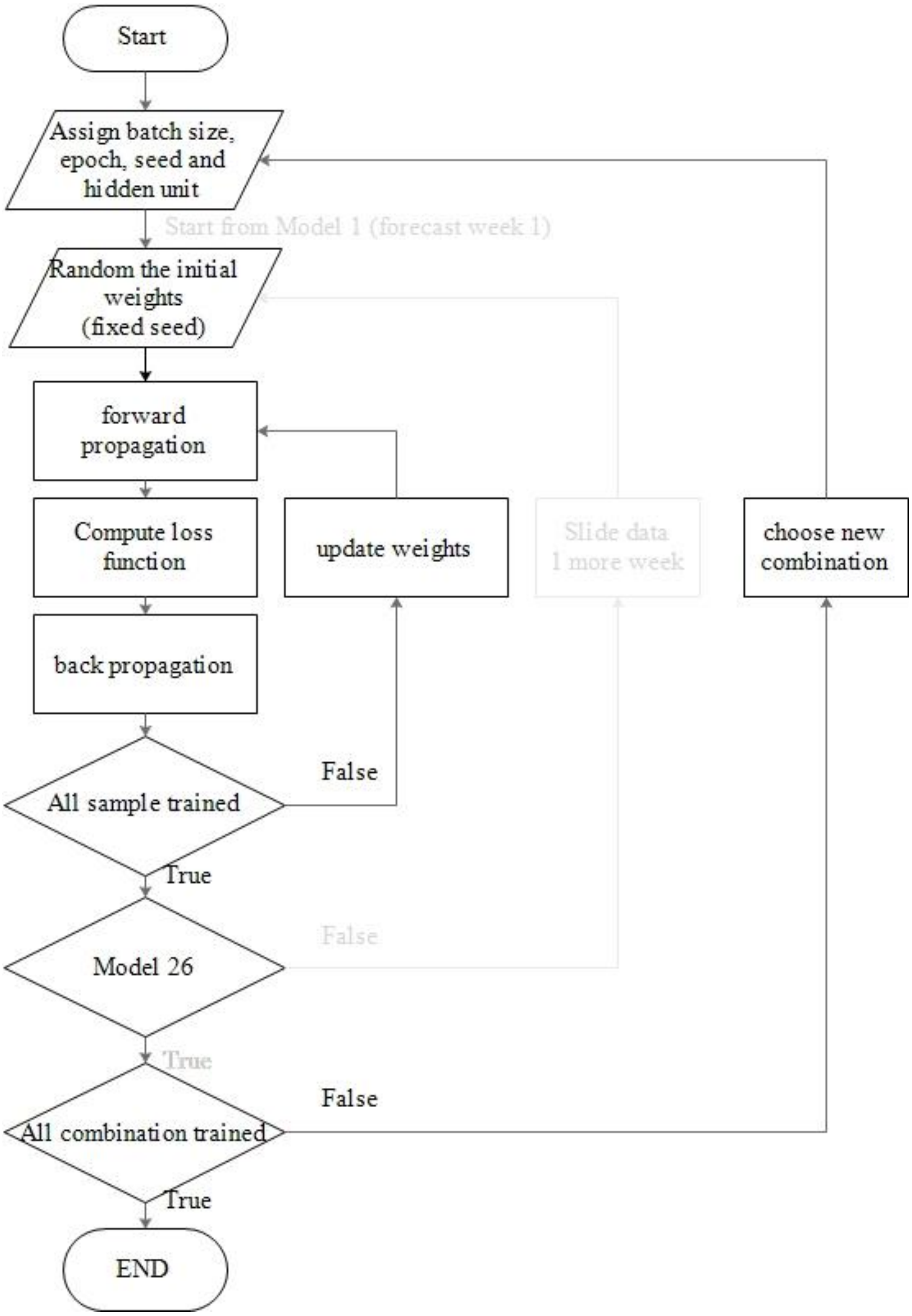


Figure 49: Flowchart showing ANN model parameter selection (1 model)

Table 11: The results of all possible structure for ANN 1 model Tour A (cross-validations)

Model Structure	Seed	Batch size	Epoch	Hidden layer	Hidden unit	MAE	RMSE	MAPE
ReLU	90	10	100	1	100	4.654	6.445	15.71%
Sigmoid	80	10	70	1	70	4.857	6.399	15.91%
ReLU, ReLU	2	10	100	2	100	6.115	7.630	24.63%
Sigmoid, Sigmoid	3	10	50	2	98	4.418	5.900	15.28%
ReLU, Sigmoid	1	10	100	2	93	5.484	7.111	19.66%
Sigmoid, ReLU	3	10	50	2	91	4.621	6.213	15.37%

Table 12: The results of all possible structure for ANN 1 model Tour B (cross-validations)

Model Structure	Seed	Batch size	Epoch	Hidden layer	Hidden unit	MAE	RMSE	MAPE
ReLU	1	10	100	1	100	1.5	2.106	-
Sigmoid	5	10	100	1	53	1.333	1.790	-
ReLU, ReLU	1	10	100	2	100	2.032	2.975	-
Sigmoid, Sigmoid	70	10	100	2	93	1.365	1.827	-
ReLU, Sigmoid	60	10	100	2	100	2.609	3.982	-
Sigmoid, ReLU	1	10	100	2	77	1.301	1.752	-

Table 13: The results of all possible structure for ANN 1 model Tour C (cross-validations)

Model Structure	Seed	Batch size	Epoch	Hidden layer	Hidden unit	MAE	RMSE	MAPE
ReLU	100	10	100	1	97	1.731	2.262	-
Sigmoid	60	10	70	1	99	1.462	2.066	-
ReLU, ReLU	80	20	100	2	99	1.910	2.534	-
Sigmoid, Sigmoid	70	32	70	2	100	1.481	2.123	-
ReLU, Sigmoid	1	10	100	2	90	2.179	2.913	-
Sigmoid, ReLU	1	10	100	2	95	1.532	2.099	-

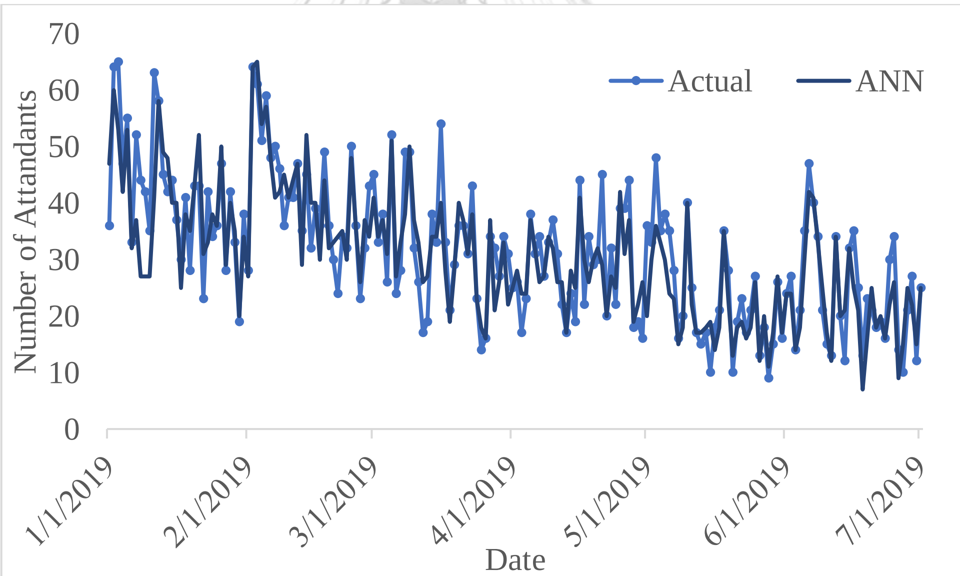


Figure 50: ANN model (2 hidden layers with ReLU) forecasting compared with the actual value for Tour A

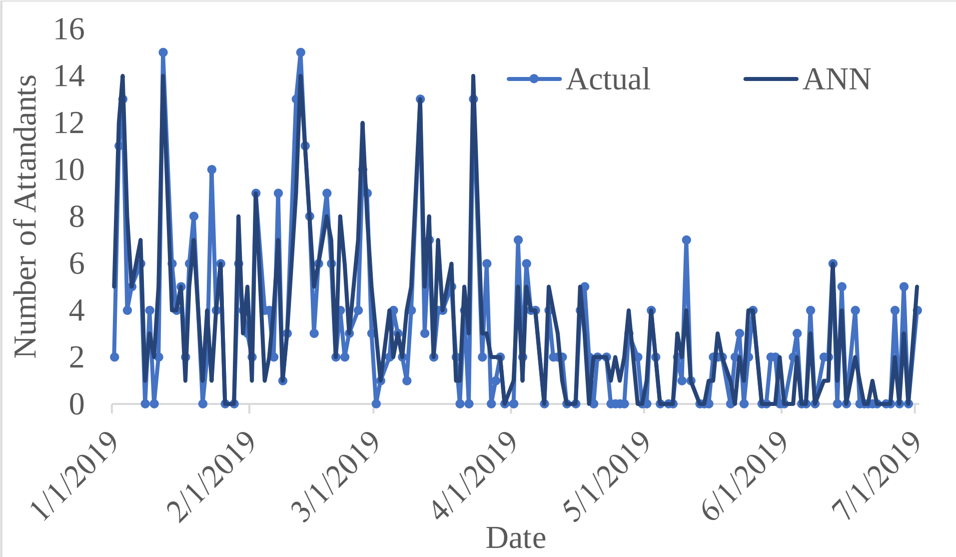


Figure 51: ANN model (2 hidden layers with ReLU) forecasting compared with the actual value for Tour B

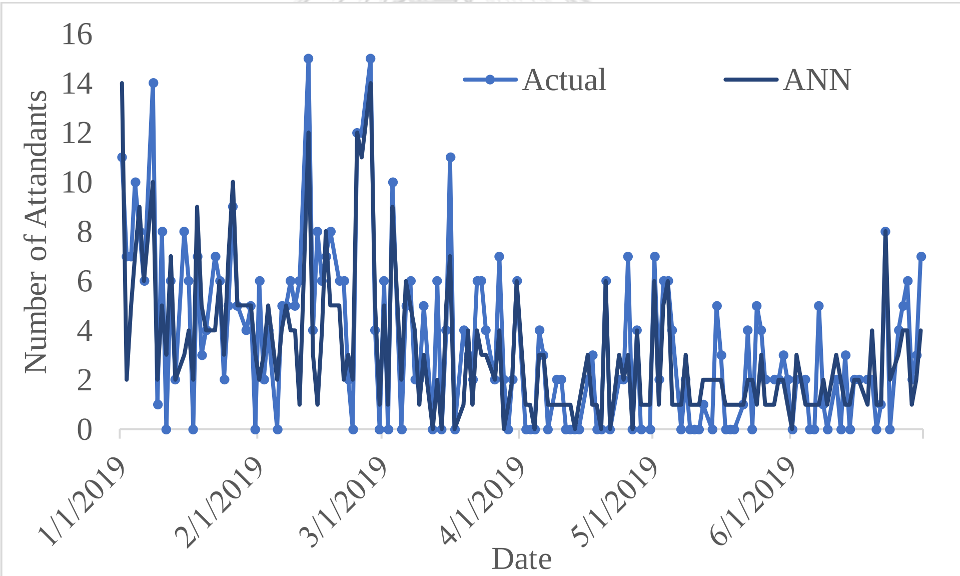


Figure 52: ANN model (1 hidden layer with ReLU) forecasting compared with the actual value for Tour C

Tables 11, 12, 13 along with Figures 50, 51 and 52 show the actual values and the predicted values of Tour A by the ANN model for the cross-validation set of data (182 days from January 2019 to 1 July 2019 for Tour A and 156 days for Tours B and C). They show that the ANN model can reduce the numbers of mean absolute error of

Tours A, B and C from 18.648, 3.179 and 2.949 to 4.418, 1.301 and 1.462 respectively. The former numbers are collected from the Same Day Last Year method which is the existing model used by the company.

4.1.6 LSTM Models

LSTM model is a sequential Machine learning model that the structure parameters is almost similar to ANN models which is explained in 4.1.5. The difference is parameter Cell state and Hidden state can remember the previous state. The LSTM model has a lot of parameters making this thesis cannot train 26 models to forecast 26 weeks like other models. Thus, only 1 model will be built to forecast 26 weeks similar to the second part of the ANN models.

Parameters:

1. Batch size is the number of data which is fed to train the model. The batch size can be assigned from 1 – n (number of input). Batch size 10, 20, 32, and 64 will be used in this thesis.

2. Epoch (iteration) is the amount of time used to train all examples. 10, 30, 50, 70, and 100 are used in these experiments.

3. Hidden node is the number of hidden units for each hidden layer. 5-100 neurons are used in the experiments of this thesis.

4. Since this thesis focuses on the machine learning 1-2 hidden layers for the experiments.

5. The activation function is a type of function which is used in the LSTM cell. The experiment of LSTM model has 1,920 (4 batch size* 5 epoch *96 neurons)

combination steps to find the best batch size, epoch, and the number of hidden units. However, the LSTM model only focuses on the ReLU activation function since the case-study data cannot fit with Sigmoid function in the LSTM model that the output data is 0 or the inverse maximum value is already transformed.

The experiment will be separated into 2 parts depending on the input variables. The first experiment uses only sequential data with data of the previous month to predict the next 7 days (The forecasting always takes place on Saturday) sequential data as shown in Figure 53. Figure 54 shows a flowchart of these models.

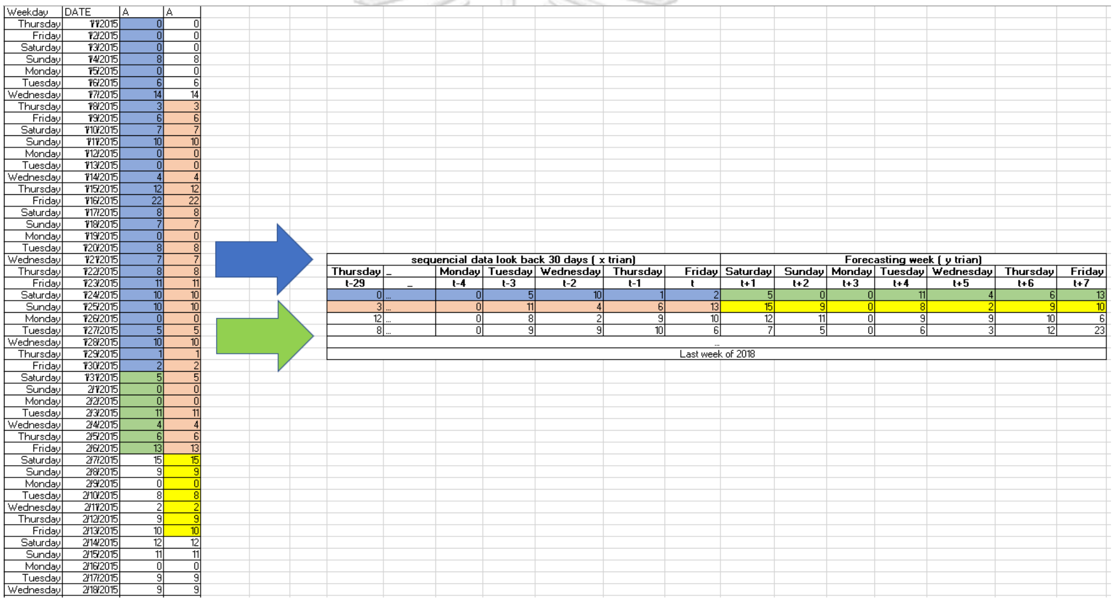


Figure 53: Transformation of time series data to sequential data (input and output of LSTM model)

Another part uses the same input as the ANN model.

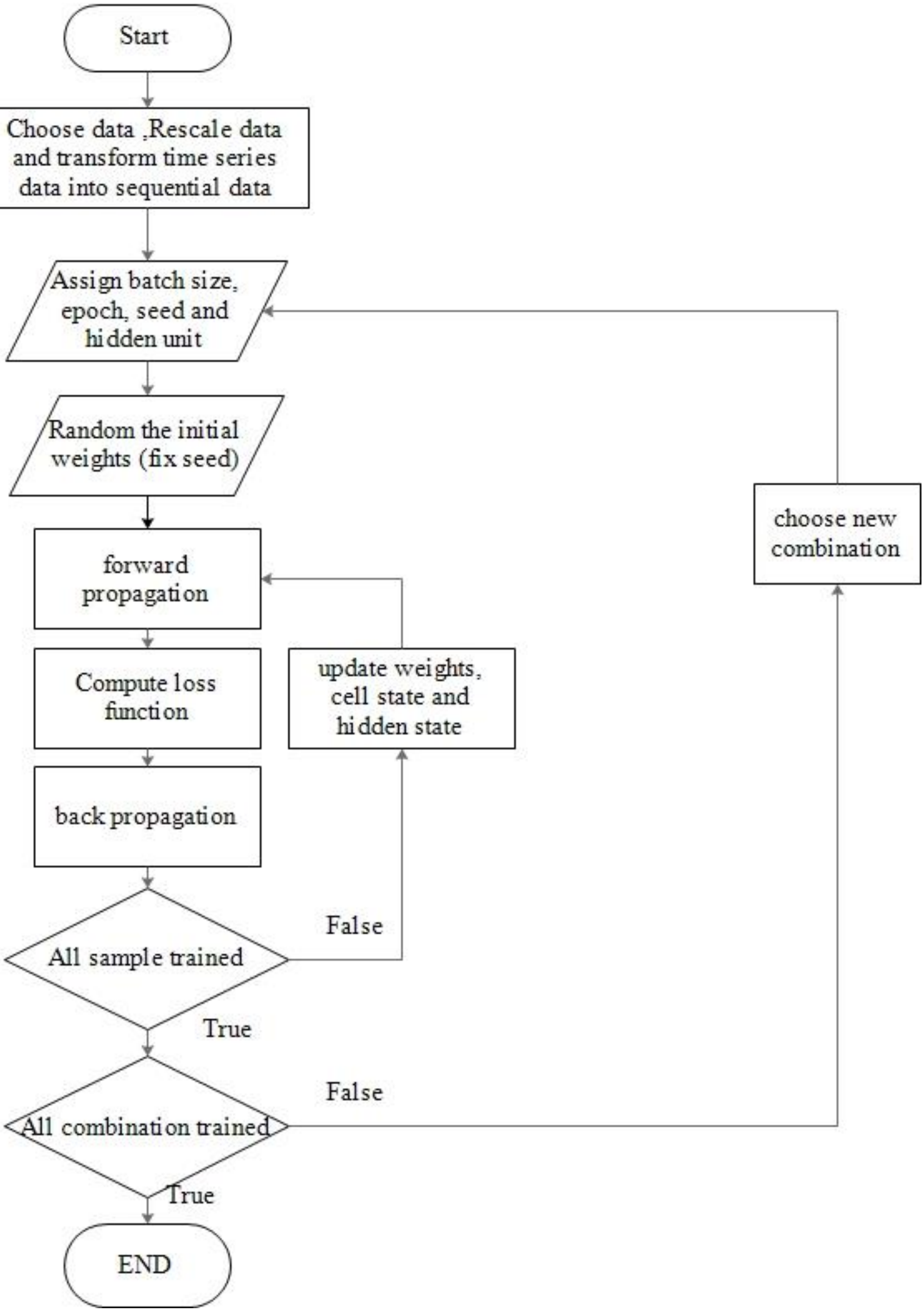


Figure 54: Flowchart showing LSTM model parameter selection (Sequential input and output)

Tables 14, 15 and 16, and Figures 55-60 show the actual values and the predicted values of Tour A by the LSTM model for the cross-validation set of data (182 days from January 2019 to 1st July, 2019 for Tour A, and 156 days for Tours B and C). They show that the Mean Absolute Error of Tour A using the LSTM model can be reduced from 18.648, 3.179 and 2.949 to 4.659, 1.737 and 1.756 respectively, which are less than the Same Day Last Year method (the existing model used by the company).

Table 14: The results of all possible structure for LSTM model Tour A

Input	Model Structure	Seed	Batch size	Epoch	Hidden layer	Hidden unit	MAE	RMSE	MAPE
External variable	ReLU	1	20	100	1	76	4.659	6.252	15.54%
	ReLU, ReLU	1	10	100	2	77	5.110	6.705	17.63%
Sequential data	ReLU	1	64	70	1	23	9.615	11.934	39.12%
	ReLU, ReLU	1	64	70	2	59	12.467	15.279	50.13%

Table 15: The results of all possible structure for LSTM model Tour B

Input	Model Structure	Seed	Batch size	Epoch	Hidden layer	Hidden unit	MAE	RMSE	MAPE
External variable	ReLU	60	10	100	1	98	1.609	2.243	-
	ReLU, ReLU	50	10	100	2	99	1.737	2.538	-
Sequential data	ReLU	1	10	10	1	24	2.327	3.253	-
	ReLU, ReLU	1	20	100	2	72	2.763	3.476	-

Table 16: The results of all possible structure for LSTM model Tour C

Input	Model Structure	Seed	Batch size	Epoch	Hidden layer	Hidden unit	MAE	RMSE	MAPE
External variable	ReLU	10	10	100	1	98	1.756	2.307	-
	ReLU, ReLU	1	10	100	2	97	2.013	2.658	-
Sequential data	ReLU	7	10	100	1	96	2.692	3.432	-
	ReLU, ReLU	5	10	100	2	95	2.769	3.566	-

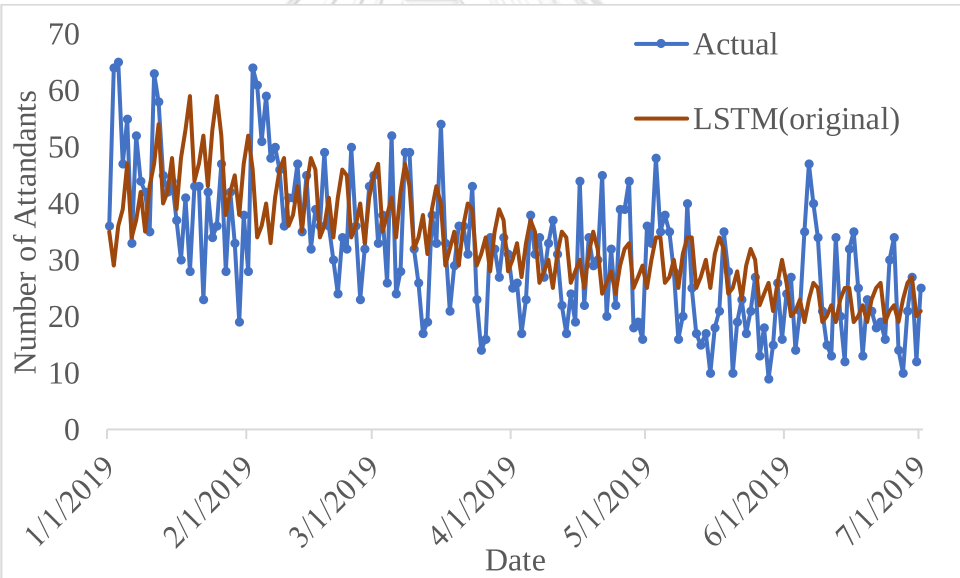


Figure 55: LSTM model (2 hidden layers with ReLU) forecasting compared with the actual value for Tour A (sequential data input and output)

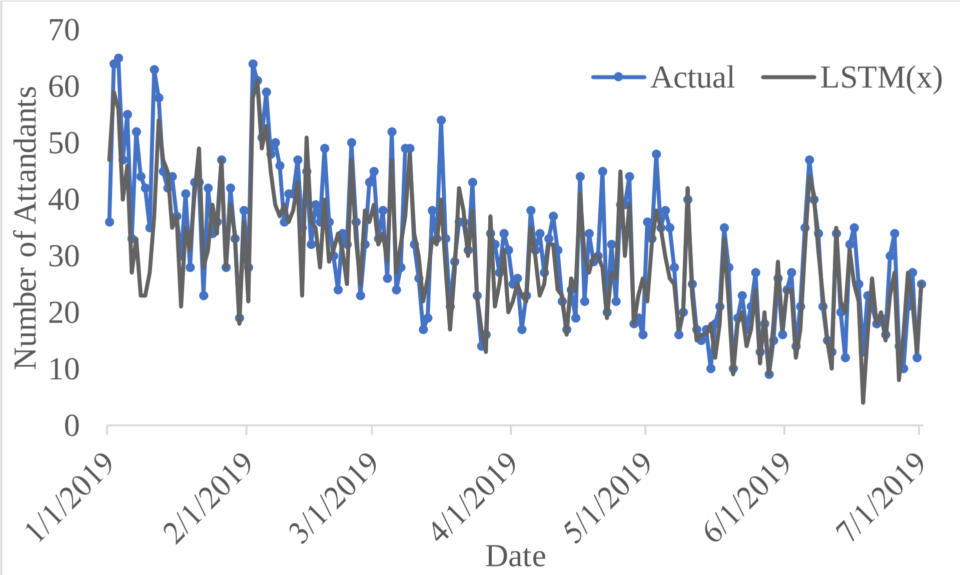


Figure 56: LSTM model (2 hidden layers with ReLU) forecasting compared with the actual value for Tour A (external input variable)

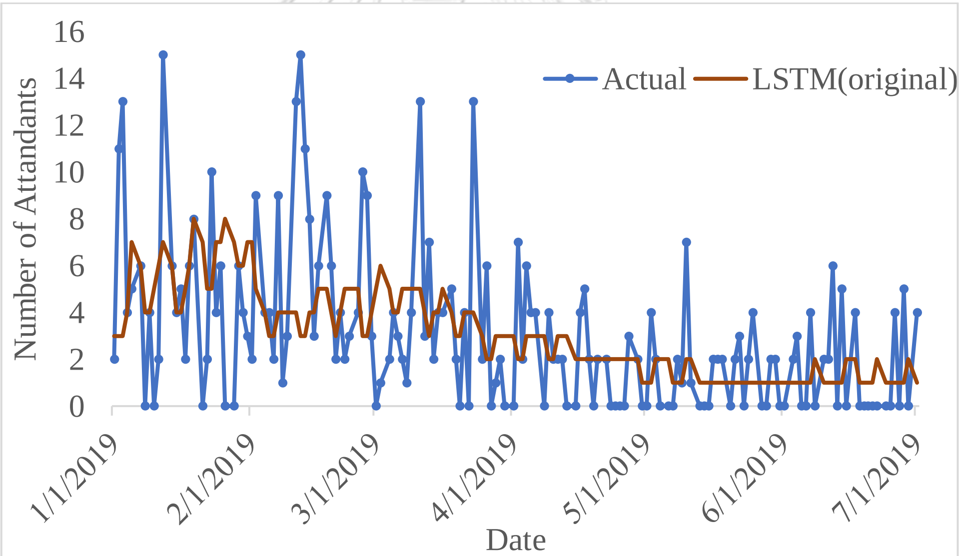


Figure 57: LSTM model (1 hidden layer with ReLU) forecasting compared with the actual value for Tour B (sequential data input and output)

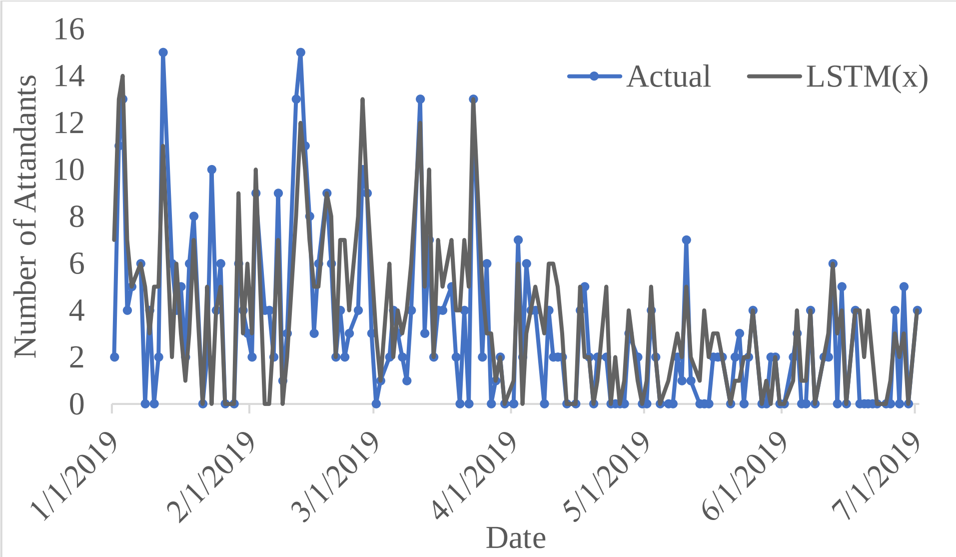


Figure 58: LSTM model (1 hidden layer with ReLU) forecasting compared with the actual value for Tour B (external input variable)

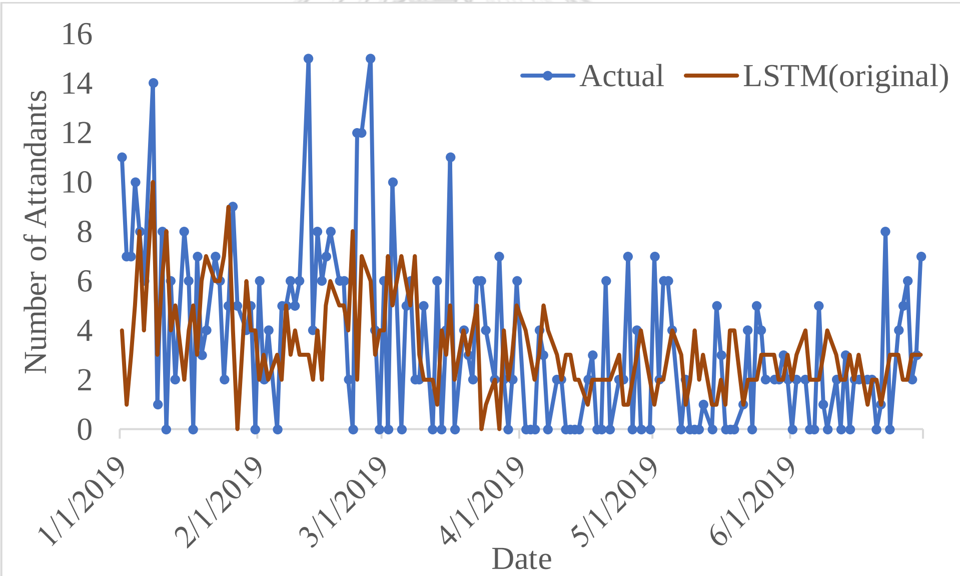


Figure 59: LSTM model (2 hidden layers with ReLU) forecasting compared with the actual value for Tour C (sequential data input and output)

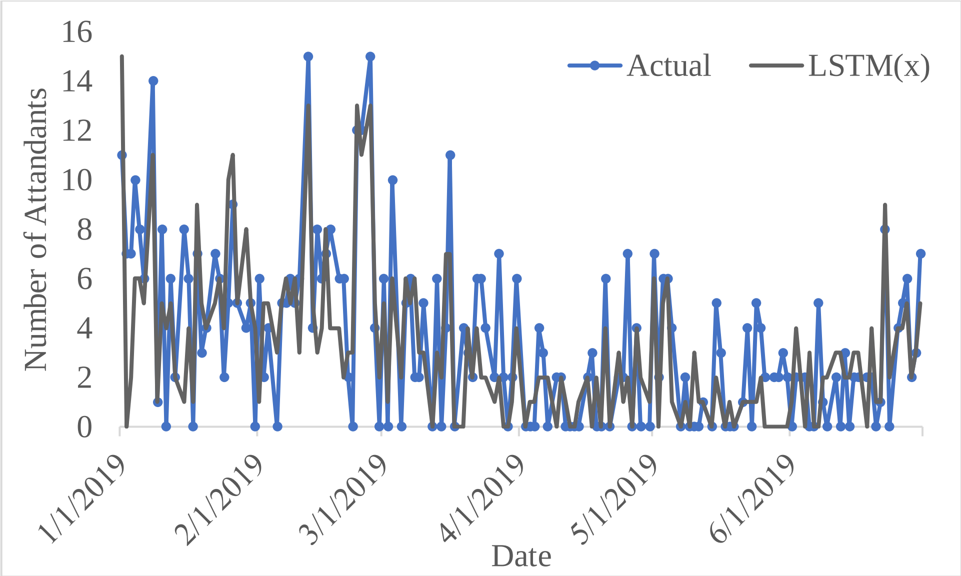


Figure 60: LSTM model (2 hidden layers with ReLU) forecasting compared with the actual value for Tour C (external input variable)

MAPE is basically a famous measurement, but its big disadvantage is it cannot measure when the actual value is zero. Since both Tours B and C have many zero values, MAPE is only used for Tour A.

Tables 14, 15, and 16 show the comparison of the new models and the existing model. It is found that Tour A which has a high mean value can reduce Mean Absolute Error from 18.648 to 4.659. This proves that the LSTM model can be dramatically improved. Moreover, Tours B and C which have low mean values and high standard deviations can reduce the Mean Absolute Error of Tour B from 3.179 to 1.609 and Tour C from 2.949 to 1.756.

4.2 Model Comparisons and Selection

Table 17 : the comparison of the error between the actual value and forecast value from each forecasting model for Tour A.

Model	Models Tour A	MAE	RMSE	MAPE
SARIMA	SARIMA	8.280	9.985	32.26%
SARIMAX	SARIMAX (Fourier Variable)	8.077	9.835	31.51%
	SARIMAX (External variable)	4.275	5.497	15.19%
TBATS	TBATS	8.308	10.099	31.94%
ANN	ANN (ReLU, ReLU, ReLU)	6.115	7.630	24.63%
	ANN (ReLU, ReLU)	4.654	6.445	15.71%
	ANN (ReLU, Sigmoid, ReLU)	5.484	7.111	19.66%
	ANN (Sigmoid, Sigmoid, ReLU)	4.418	5.900	15.28%
	ANN (Sigmoid, ReLU)	4.857	6.399	15.91%
	ANN (Sigmoid, ReLU, ReLU)	4.621	6.213	15.37%
LSTM	LSTM (original) (ReLU, ReLU, ReLU)	12.467	15.279	50.13%
	LSTM (original) (ReLU, ReLU)	9.615	11.934	39.12%
	LSTM (External variable) (ReLU, ReLU, ReLU)	5.110	6.705	17.63%
	LSTM (External variable) (ReLU, ReLU)	4.659	6.252	15.54%

Table 18 : the comparison of the error between the actual value and forecast value from each forecasting model for Tour B.

Model	Models Tour B	MAE	RMSE	MAPE
SARIMA	SARIMA	2.519	3.237	-
SARIMAX	SARIMAX (Fourier Variable)	2.436	3.184	-
	SARIMAX (External variable)	1.179	1.573	-
TBATS	TBATS	2.313	3.208	-
ANN	ANN (ReLU, ReLU, ReLU)	2.032	2.975	-
	ANN (ReLU, ReLU)	1.5	2.106	-
	ANN (ReLU, Sigmoid, ReLU)	2.609	3.982	-
	ANN (Sigmoid, Sigmoid, ReLU)	1.365	1.827	-
	ANN (Sigmoid, ReLU)	1.333	1.790	-
	ANN (Sigmoid, ReLU, ReLU)	1.301	1.752	-
LSTM	LSTM (original) (ReLU, ReLU, ReLU)	2.763	3.476	-
	LSTM (original) (ReLU, ReLU)	2.327	3.253	-
	LSTM (External variable) (ReLU, ReLU, ReLU)	1.737	2.538	-
	LSTM (External variable) (ReLU, ReLU)	1.609	2.243	-

Table 19 : the comparison of the error between the actual value and forecast value from each forecasting model for Tour C.

Model	Models Tour C	MAE	RMSE	MAPE
SARIMA	SARIMA	2.397	3.111	-
SARIMAX	SARIMAX (Fourier Variable)	2.327	3.043	-
	SARIMAX (External variable)	1.487	1.971	-
TBATS	TBATS	2.423	3.121	-
ANN	ANN (ReLU, ReLU, ReLU)	1.910	2.534	-
	ANN (ReLU, ReLU)	1.731	2.262	-
	ANN (ReLU, Sigmoid, ReLU)	2.179	2.913	-
	ANN (Sigmoid, Sigmoid, ReLU)	1.481	2.123	-
	ANN (Sigmoid, ReLU)	1.462	2.066	-
	ANN (Sigmoid, ReLU, ReLU)	1.532	2.099	-
LSTM	LSTM (original) (ReLU, ReLU, ReLU)	2.769	3.566	-
	LSTM (original) (ReLU, ReLU)	2.692	3.432	-
	LSTM (External variable) (ReLU, ReLU, ReLU)	2.013	2.658	-
	LSTM (External variable) (ReLU, ReLU)	1.756	2.307	-

Table 20 : the comparison of the time usage from each model using laptop with Intel Core i5-8300H and 24 GB DDR4 RAM.

Model	**Time usage**
SARIMA	120 minutes
SARIMAX	150 minutes
TBATS	180 minutes
ANN	2 minutes
LSTM	3 minutes

The results from Tables 16-18 show the comparison of the error between the actual value and forecast value. The ANN model is the most accurate for Tour C. Tour A and Tour B also works well with the ANN model, but the SARIMAX model makes a slightly better MAE. To compare the performance of every model, Tukey Pairwise Comparisons and the Randomized Complete Block Designs (RCBD) (Figures 61-69) are tested by the Absolute Error (as shown in Table 21) at 95% confidence interval.

Table 21:Absolute error of Tour A

	ABS ERROR					
DATE	SDLY	SARIMA	SARIMAX	TBATS	ANN	LSTM
1/1/2019	1	20	13	19	11	11
1/2/2019	35	9	4	12	4	5
1/3/2019	22	8	10	14	12	9
1/4/2019	19	11	0	5	5	7
1/5/2019	38	4	2	5	2	9
1/6/2019	4	25	2	13	1	6
1/7/2019	43	5	9	13	15	19
1/8/2019	32	6	14	6	17	21
1/9/2019	19	7	12	7	15	19
1/10/2019	22	16	0	2	8	8
1/11/2019	48	11	17	28	22	26
1/12/2019	42	5	3	23	0	4
1/13/2019	22	6	7	14	4	2
…						
End of Cross validation data set						

Method

Null hypothesis	All means are equal
Alternative hypothesis	Not all means are equal
Significance level	α = 0.05

Equal variances were assumed for the analysis.

Factor Information

Factor	Levels	Values
Factor	6	SDLY, SARIMA, SARIMAX, TBATS, ANN, LSTM

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Factor	5	28685	5737.06	148.07	0.000
Error	1086	42078	38.75		
Total	1091	70763			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
6.22459	40.54%	40.26%	39.88%

Means

Factor	N	Mean	StDev	95% CI
SDLY	182	18.648	11.265	(17.743, 19.554)
SARIMA	182	8.280	5.596	(7.375, 9.186)
SARIMAX	182	4.275	3.466	(3.369, 5.180)
TBATS	182	8.308	5.758	(7.402, 9.213)
ANN	182	4.159	3.963	(3.254, 5.065)
LSTM	182	4.071	3.660	(3.166, 4.977)

Pooled StDev = 6.22459

Tukey Pairwise Comparisons

Grouping Information Using the Tukey Method and 95% Confidence

Factor	N	Mean	Grouping		
SDLY	182	18.648	A		
TBATS	182	8.308		B	
SARIMA	182	8.280		B	
SARIMAX	182	4.275			C
ANN	182	4.159			C
LSTM	182	4.071			C

Means that do not share a letter are significantly different.

Figure 61: Randomized Complete Block Designs (RCBD) of all forecasting model in Tour A

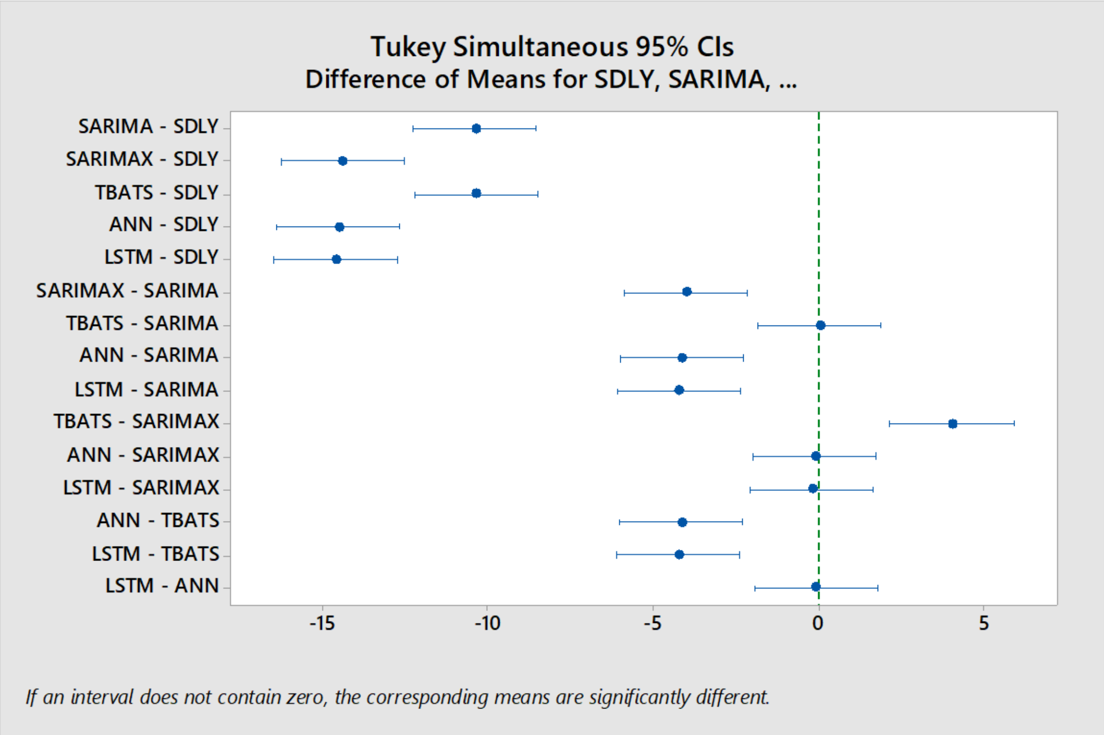


Figure 62: Tukey Simultaneous difference of mean for all forecasting model in Tour A

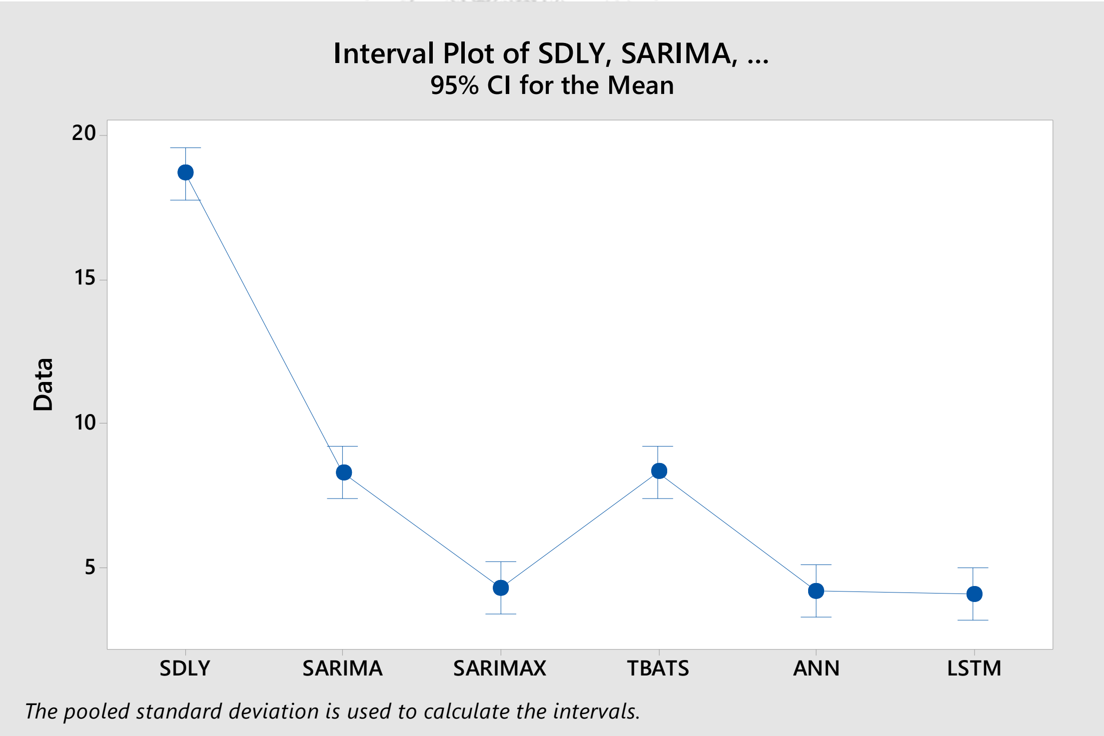


Figure 63: Interval Plot of forecasting model in Tour A

Method

Null hypothesis	All means are equal
Alternative hypothesis	Not all means are equal
Significance level	α = 0.05

Equal variances were assumed for the analysis.

Factor Information

Factor	Levels	Values
Factor	6	SDLY, SARIMA, SARIMAX, TBATS, ANN, LSTM

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Factor	5	489.4	97.870	21.16	0.000
Error	930	4300.6	4.624		
Total	935	4790.0			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
2.15042	10.22%	9.73%	9.05%

Means

Factor	N	Mean	StDev	95% CI
SDLY	156	3.179	3.645	(2.842, 3.517)
SARIMA	156	2.519	2.040	(2.181, 2.857)
SARIMAX	156	1.1795	1.0441	(0.8416, 1.5174)
TBATS	156	2.410	2.316	(2.072, 2.748)
ANN	156	1.3013	1.1773	(0.9634, 1.6392)
LSTM	156	1.609	1.568	(1.271, 1.947)

Pooled StDev = 2.15042

Tukey Pairwise Comparisons

Grouping Information Using the Tukey Method and 95% Confidence

Factor	N	Mean	Grouping		
SDLY	156	3.179	A		
SARIMA	156	2.519	A	B	
TBATS	156	2.410		B	
LSTM	156	1.609			C
ANN	156	1.3013			C
SARIMAX	156	1.1795			C

Means that do not share a letter are significantly different.

Figure 64: Randomized Complete Block Designs (RCBD) of all forecasting model in Tour B

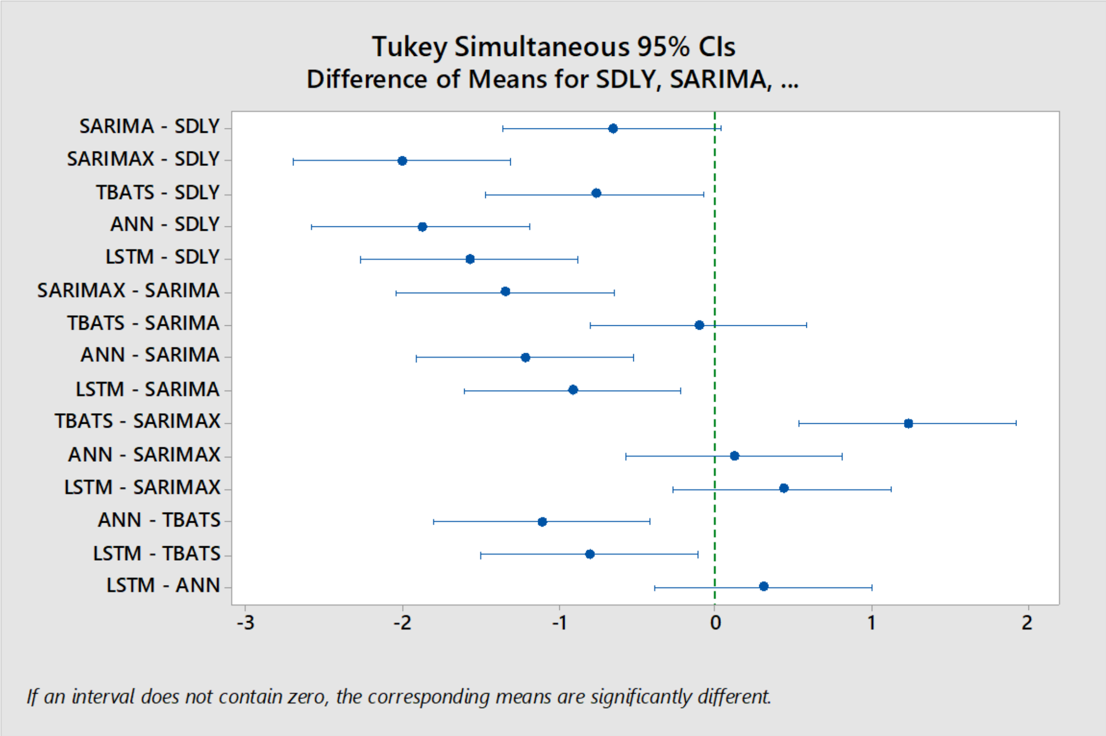


Figure 65: Tukey Simultaneous difference of mean for all forecasting model in Tour B

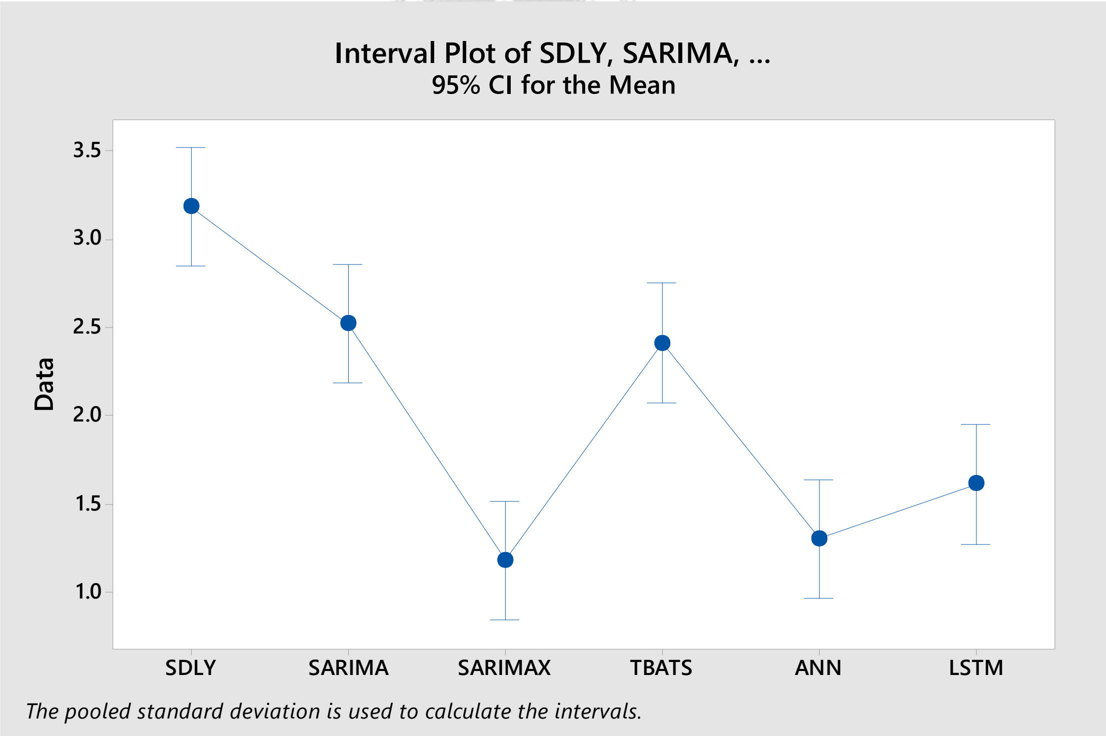


Figure 66: Interval Plot of forecasting model in Tour B

Method

Null hypothesis	All means are equal
Alternative hypothesis	Not all means are equal
Significance level	α = 0.05

Equal variances were assumed for the analysis.

Factor Information

Factor	Levels	Values
Factor	6	SDLY, SARIMA, SARIMAX, TBATS, ANN, LSTM

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Factor	5	282.6	56.527	15.64	0.000
Error	930	3361.5	3.615		
Total	935	3644.1			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
1.90119	7.76%	7.26%	6.56%

Means

Factor	N	Mean	StDev	95% CI
SDLY	156	2.949	2.784	(2.650, 3.247)
SARIMA	156	2.397	1.989	(2.099, 2.696)
SARIMAX	156	1.487	1.298	(1.188, 1.786)
TBATS	156	2.423	1.974	(2.124, 2.722)
ANN	156	1.462	1.465	(1.163, 1.760)
LSTM	156	1.756	1.500	(1.458, 2.055)

Pooled StDev = 1.90119

Tukey Pairwise Comparisons

Grouping Information Using the Tukey Method and 95% Confidence

Factor	N	Mean	Grouping	
SDLY	156	2.949	A	
TBATS	156	2.423	A	
SARIMA	156	2.397	A	
LSTM	156	1.756		B
SARIMAX	156	1.487		B
ANN	156	1.462		B

Means that do not share a letter are significantly different.

Figure 67: Randomized Complete Block Designs (RCBD) of all forecasting model in Tour C

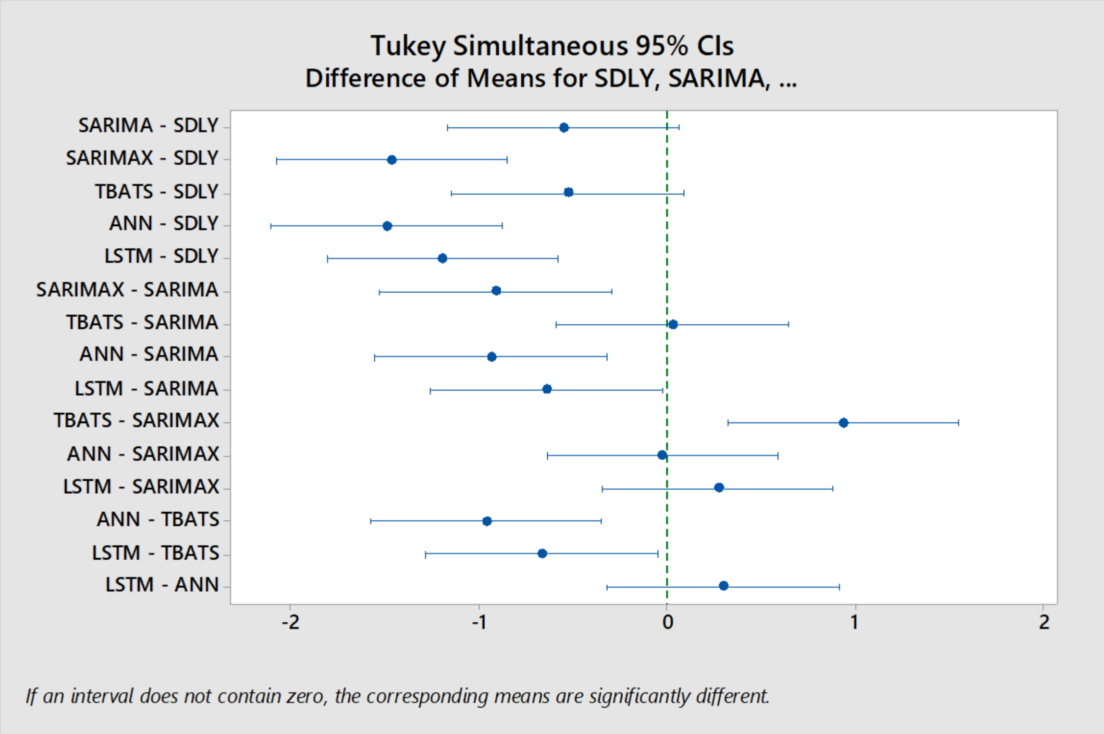


Figure 68: Tukey Simultaneous difference of mean for all forecasting model in Tour C

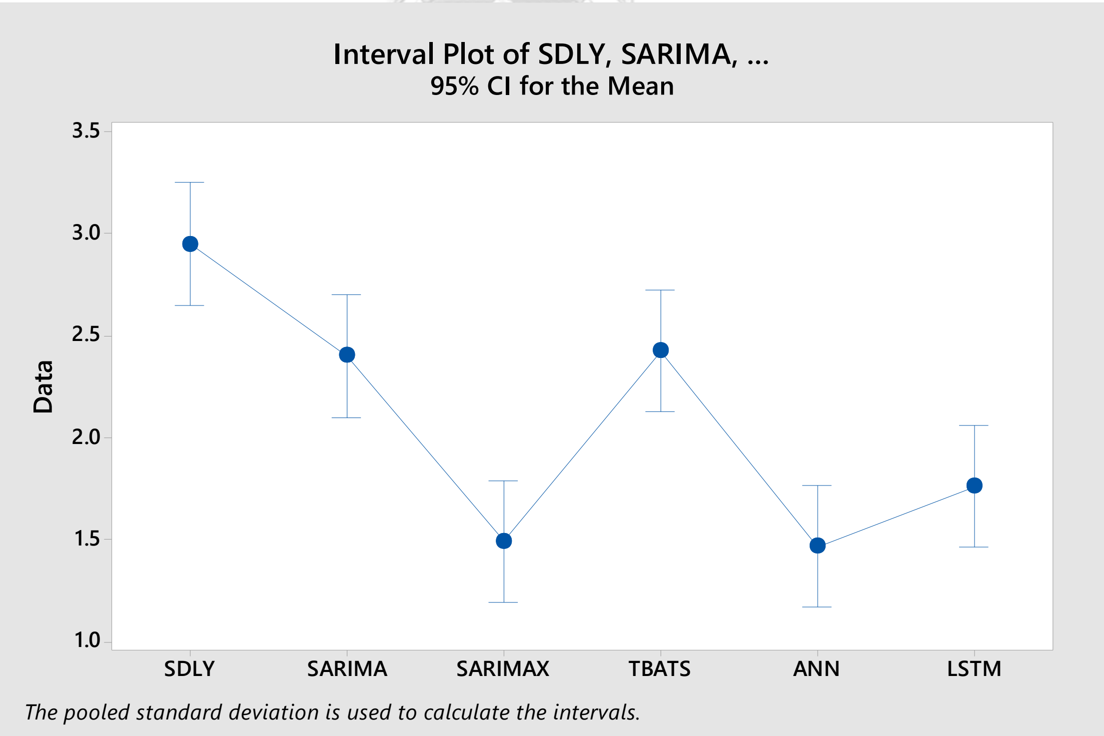


Figure 69: Interval Plot of forecasting model in Tour C

4.3 Testing set

The results of Tukey Pairwise Comparisons and Randomized Complete Block Designs (RCBD) which test the blocked date for inspect different of forecast values in each model in Figures 68-76 show that the cross-validation forecast means of SARIMAX model, ANN model and LSTM model are not significantly different. Consequently, the ANN model is suggested for all Tours of this case study company. Since the ANN model does not only provide the most accurate among all models in Tour C, but also works well in Tours A and B although the SARIMAX model makes more accuracy. In addition, the ANN model uses less parameters and runs faster than the LSTM models as shown in Table 20. The results of forecasting using the testing data by the ANN model with the same structure as the cross-validation forecasting are shown in Table 22 and Figures 70-72.

Table 22: The results of forecasting using testing data by the ANN model

ANN model	MAE	RMSE	MAPE
TOUR A	4.437	6.471	15.89%
TOUR B	1.191	1.684	-
TOUR C	1.369	1.687	-

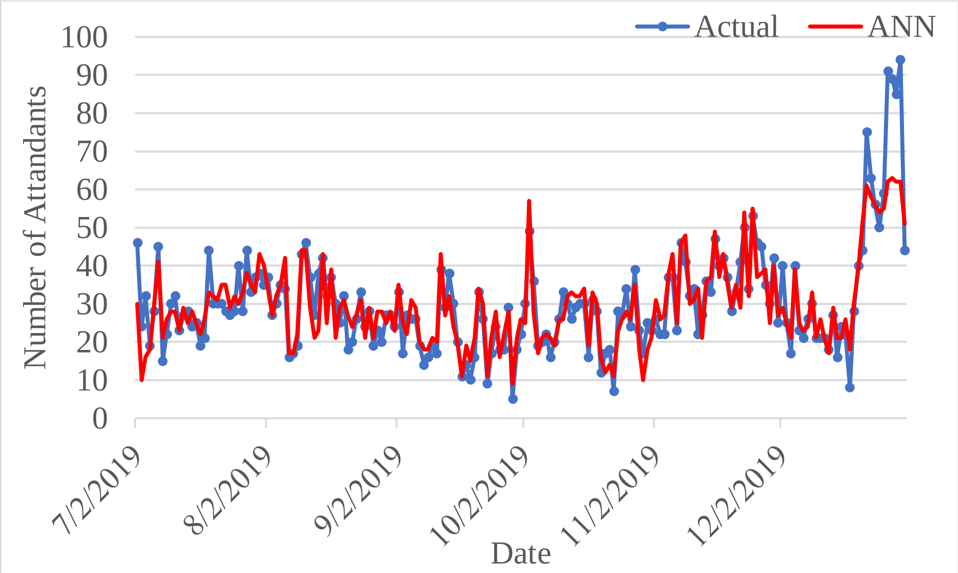


Figure 70: Forecasting Tour A testing data by ANN model with same structure as the cross-validation forecasting

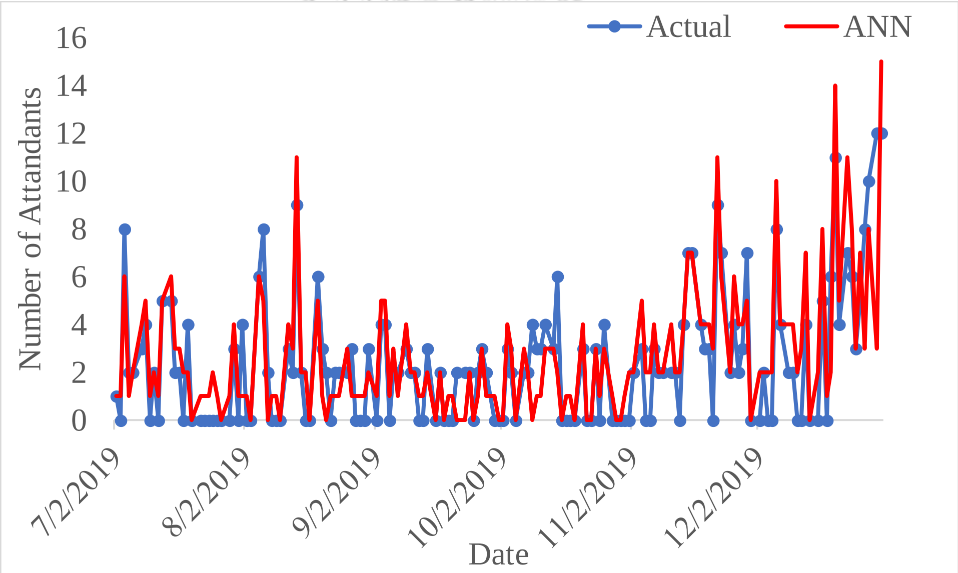


Figure 71: Forecasting Tour B testing data by ANN model with same structure as the cross-validation forecasting

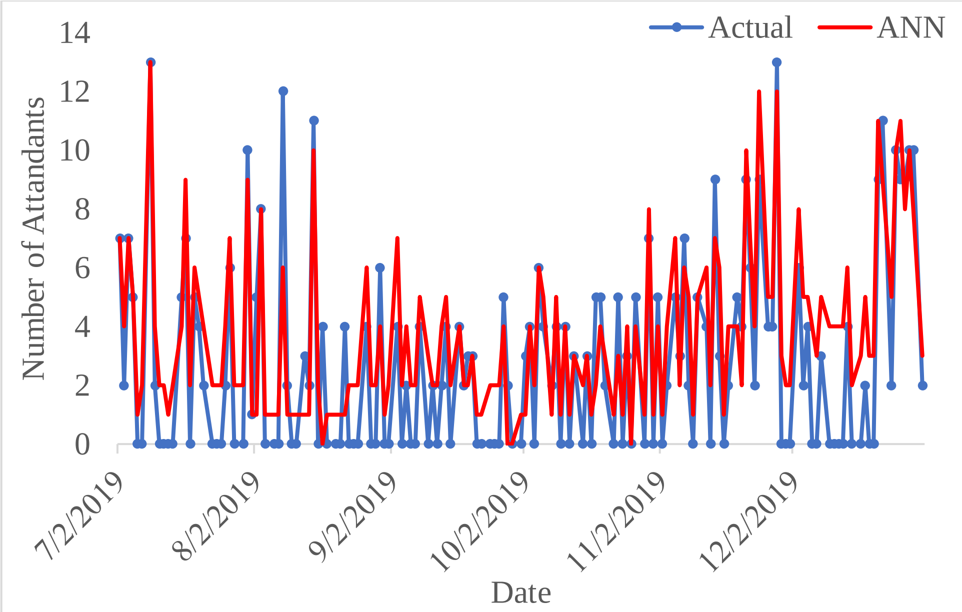


Figure 72: Forecasting Tour C testing data by ANN model with same structure as the Table 23 cross validation forecasting

Chapter V: Conclusion and Future Work

5.1 Conclusion

The objective of this thesis is to find suitable forecasting models for daily tourist demand by using time-series models and machine learning models to forecast tourist demand in every ending of each week. The results of Tukey Pairwise Comparisons and Randomized Complete Block Designs (RCBD) of Tour A and Tour B can be separated into 3 groups while Tour C can be separated into 2 groups; SDLY model is in the same group of SARIMA and TBATS. SDLY models (an existing model used in the case study) are the one with the worst accuracy for Tour A and Tour B compared with other models based on the observation results of the input variables in the remaining 2 groups. The second group containing SARIMA model, TBATS model, and SDLY model (only Tour C) use historical data to be an input. The last group containing SARIMAX model, ANN model, and LSTM model use external variables. If it can be chosen only one model, the best one would be ANN model which make the most accuracy in Tour C. However, the results of Tours A and B are different. They showed that the SARIMAX model is more accurate than the ANN model, but the ANN model is chosen because the results of RCBD indicate that the ANN model is in the same group with the SARIMAX and the LSTM models which means that the accuracies of these three models are not statistically different. Besides, the ANN model employs less parameters resulting in lower memory usage. Not to mention that the ANN model is faster and more comfortable to use compared to the SARIMAX model. Although the SARIMAX is a time series model, it is preferred to use for the short-range predictions. Thus, it requires a weekly update as shown in the thesis. Finally, in the testing data, the ANN model can improve the accuracy of MAE from 15.73 to 4.437 and MAPE from 53.37% to 15.89% in Tour A while in Tour B and Tour C, the MAEs decrease from 2.50 to 1.191 and from 3.08 to 1.687, respectively.

5.2 Recommendations for model improvement

There are several ways to improve models to make more accuracy. The first way is trying other models and another way is to improve or try other parameters.

1. Looking for other activation functions.
2. Try to use the hidden units which have more than 100 neurons.
3. Try to find a better seed that has more than 100 seeds.
4. Add more hidden layers than this thesis added for deeper researching.
5. Try to use the Epochs with the number above 100.
6. Batch size1 should be used with high-performance equipment.
7. Find and add more external variables.

5.3 Recommendations for case company

The results of this thesis show that the ANN model is the most suitable forecasting model which can reduce MAPE to be less than 16% of Tour A and can reduce MAE to be less than 2 people per day for Tour B and Tour C in both Cross validation set and Test set. Therefore, it implies that the ANN model can provide the forecast for the whole next year because the Cross validation set covers the first half year of 2019 and the Test set covers the last half year of 2019. If the case study company employs the ANN model and parameters trained in this thesis to use in real case, the author recommends to use this model for the next year. For the year after next year, if the MAPE is more than 20% or over than acceptable, then the model should be trained with new data. If the MAPE of the model using the new data is still over acceptable then all parameters should be updated.

The case study company is temporarily close due to Covid-19 overspread all over the world that directly impacts the tour operator companies because the foreign tourists can not travel across the countries. After the Covid-19 is eliminated by the vaccine and the tourists start to travel again, the author recommends to use SARIMAX model because this model is good for short period prediction and can follow the trend quickly. In the next year, the company should train the ANN model with the data from previous year.

APPENDIX

Stepwise to analyze external variables of Tour A step by step

Stepwise Selection of Terms

Candidate terms: Xbooked, Xmonth1, Xmonth2, Xmonth3, Xmonth4, Xmonth5, Xmonth6, Xmonth7, Xmonth8, Xmonth10, Xmonth11, Xmonth12, Xday1, Xday3, Xday4, Xday5, Xday6, Xday7, Lag1, Lag2, Lag3, Lag4, Lag5, Lag6, Lag7, Lag8, Lag9, Lag10, Lag11, Lag12, Lag13, Lag14, Lag15, Lag16, Lag17, Lag18, Lag19, Lag20, Lag21, Lag22, Lag23, Lag24, Lag25, Lag26, Lag27, Lag28, Lag29, Lag30

	-----Step 1----		-----Step 2----		-----Step 3----		-----Step 4----	
	Coef	P	Coef	P	Coef	P	Coef	P
Constant	6.125		3.853		4.496		3.910	
Xbooked	1.0753	0.000	0.9614	0.000	1.0350	0.000	1.0365	0.000
Lag5			0.1948	0.000	0.1484	0.000	0.1480	0.000
Xday7					-4.526	0.000	-3.962	0.000
Xday6							3.479	0.000
Lag1								
Xday1								
Xday5								
Lag2								
Xday4								
Xday3								
Lag3								
Xmonth12								
Xmonth11								
Lag6								
Xmonth7								
Lag9								
Xmonth10								
Lag30								
S		5.51120		5.13650		4.93321		4.78624
R-sq		80.11%		82.73%		84.08%		85.02%
R-sq(adj)		80.10%		82.71%		84.05%		84.99%
R-sq(pred)		80.04%		82.64%		83.99%		84.91%
Mallows' Cp		939.02		580.97		397.79		270.34

	-----Step 5----		-----Step 6----		-----Step 7----		-----Step 8----	
	Coef	P	Coef	P	Coef	P	Coef	P
Constant	3.307		3.796		3.437		3.116	
Xbooked	0.9707	0.000	0.9868	0.000	0.9917	0.000	0.9729	0.000
Lag5	0.1112	0.000	0.0938	0.000	0.0845	0.000	0.0682	0.000
Xday7	-3.766	0.000	-4.524	0.000	-4.195	0.000	-3.914	0.000
Xday6	3.494	0.000	2.973	0.000	3.390	0.000	3.557	0.000
Lag1	0.1082	0.000	0.1191	0.000	0.1230	0.000	0.0947	0.000
Xday1			-2.635	0.000	-2.262	0.000	-2.184	0.000
Xday5					1.703	0.000	1.795	0.000
Lag2							0.0689	0.000
Xday4								
Xday3								
Lag3								
Xmonth12								
Xmonth11								
Lag6								
Xmonth7								
Lag9								
Xmonth10								
Lag30								
S		4.69735		4.61778		4.58599		4.55347
R-sq		85.58%		86.08%		86.27%		86.48%
R-sq(adj)		85.54%		86.03%		86.22%		86.42%
R-sq(pred)		85.46%		85.94%		86.12%		86.30%
Mallows' Cp		195.56		130.00		104.71		79.03

	-----Step 9----		----Step 10----		----Step 11----		----Step 12----	
	Coef	P	Coef	P	Coef	P	Coef	P
Constant	2.588		1.443		1.280		1.053	
Xbooked	0.9762	0.000	0.9692	0.000	0.9636	0.000	0.9612	0.000
Lag5	0.0587	0.000	0.0530	0.000	0.0419	0.001	0.0415	0.002
Xday7	-3.423	0.000	-2.294	0.000	-2.142	0.000	-1.893	0.000
Xday6	4.111	0.000	5.221	0.000	5.280	0.000	5.505	0.000
Lag1	0.0888	0.000	0.0958	0.000	0.0871	0.000	0.0858	0.000
Xday1	-1.678	0.000	-0.598	0.138	-0.446	0.272		
Xday5	2.361	0.000	3.482	0.000	3.677	0.000	3.902	0.000
Lag2	0.0822	0.000	0.0874	0.000	0.0745	0.000	0.0758	0.000
Xday4	1.566	0.000	2.684	0.000	2.692	0.000	2.914	0.000
Xday3			2.172	0.000	2.056	0.000	2.267	0.000
Lag3					0.0416	0.003	0.0435	0.002
Xmonth12								
Xmonth11								
Lag6								
Xmonth7								
Lag9								
Xmonth10								
Lag30								
S		4.53065		4.49518		4.48540		4.48566
R-sq		86.62%		86.83%		86.90%		86.89%
R-sq(adj)		86.55%		86.76%		86.82%		86.82%
R-sq(pred)		86.43%		86.63%		86.68%		86.68%
Mallows' Cp		61.44		33.77		26.90		26.12

	----Step 13----		----Step 14----		----Step 15----		----Step 16----	
	Coef	P	Coef	P	Coef	P	Coef	P
Constant	1.112		1.126		1.106		1.011	
Xbooked	0.9570	0.000	0.9558	0.000	0.9527	0.000	0.9529	0.000
Lag5	0.0434	0.001	0.0434	0.001	0.0315	0.024	0.0324	0.020
Xday7	-1.869	0.000	-1.869	0.000	-1.975	0.000	-1.974	0.000
Xday6	5.476	0.000	5.455	0.000	5.252	0.000	5.245	0.000
Lag1	0.0820	0.000	0.0799	0.000	0.0754	0.000	0.0749	0.000
Xday1								
Xday5	3.859	0.000	3.832	0.000	3.698	0.000	3.693	0.000
Lag2	0.0738	0.000	0.0722	0.000	0.0677	0.000	0.0673	0.000
Xday4	2.878	0.000	2.855	0.000	2.801	0.000	2.796	0.000
Xday3	2.241	0.000	2.229	0.000	2.207	0.000	2.206	0.000
Lag3	0.0428	0.002	0.0416	0.003	0.0361	0.010	0.0361	0.010
Xmonth12	1.092	0.006	1.254	0.002	1.300	0.001	1.372	0.001
Xmonth11			1.082	0.006	1.107	0.005	1.180	0.003
Lag6					0.0333	0.014	0.0342	0.012
Xmonth7							0.746	0.049
Lag9								
Xmonth10								
Lag30								
S		4.47730		4.46896		4.46268		4.45909
R-sq		86.95%		87.00%		87.05%		87.07%
R-sq(adj)		86.87%		86.91%		86.95%		86.97%
R-sq(pred)		86.72%		86.76%		86.78%		86.79%
Mallows' Cp		20.42		14.74		10.73		8.88

	----Step 17----		----Step 18----		----Step 19----	
	Coef	P	Coef	P	Coef	P
Constant	0.867		0.758		0.561	
Xbooked	0.9516	0.000	0.9519	0.000	0.9484	0.000
Lag5	0.0283	0.046	0.0288	0.042	0.0276	0.052
Xday7	-1.903	0.000	-1.899	0.000	-1.828	0.000
Xday6	5.356	0.000	5.361	0.000	5.418	0.000
Lag1	0.0730	0.000	0.0727	0.000	0.0712	0.000
Xday1						
Xday5	3.776	0.000	3.778	0.000	3.814	0.000
Lag2	0.0644	0.000	0.0645	0.000	0.0630	0.000
Xday4	2.923	0.000	2.929	0.000	3.010	0.000
Xday3	2.253	0.000	2.253	0.000	2.283	0.000
Lag3	0.0328	0.021	0.0329	0.020	0.0311	0.028
Xmonth12	1.418	0.000	1.476	0.000	1.479	0.000
Xmonth11	1.222	0.002	1.285	0.001	1.437	0.000
Lag6	0.0286	0.040	0.0290	0.037	0.0281	0.043
Xmonth7	0.768	0.043	0.845	0.027	0.917	0.017
Lag9	0.0227	0.077	0.0238	0.064	0.0207	0.110
Xmonth10			0.611	0.111	0.698	0.071
Lag30					0.0195	0.082
S		4.45642		4.45449		4.45195
R-sq		87.10%		87.12%		87.14%
R-sq(adj)		86.99%		87.00%		87.01%
R-sq(pred)		86.79%		86.80%		86.80%
Mallows' Cp		7.75		7.22		6.21

α to enter = 0.15, α to remove = 0.15

Stepwise to analyze external variables of Tour B step by step

Stepwise Selection of Terms

Candidate terms: Xbooked, Xmonth1, Xmonth2, Xmonth3, Xmonth4, Xmonth5, Xmonth6, Xmonth7, Xmonth8, Xmonth10, Xmonth11, Xmonth12, Xday2, Xday3, Xday5, Xday6, Xday7, Lag1, Lag2, Lag3, Lag4, Lag5, Lag6, Lag7, Lag8, Lag9, Lag10, Lag11, Lag12, Lag13, Lag14, Lag15, Lag16, Lag17, Lag18, Lag19, Lag20, Lag21, Lag22, Lag23, Lag24, Lag25, Lag26, Lag27, Lag28, Lag29, Lag30

	-----Step 1----		-----Step 2----		-----Step 3----		-----Step 4----	
	Coef	P	Coef	P	Coef	P	Coef	P
Constant	1.7508		1.9121		1.3567		1.108	
Xbooked	1.0001	0.000	1.0816	0.000	1.0472	0.000	1.0195	0.000
Xday7			-2.264	0.000	-2.044	0.000	-1.964	0.000
Lag3					0.1388	0.000	0.1162	0.000
Lag1							0.0929	0.000
Lag5								
Xmonth12								
Lag4								
Lag28								
Lag6								
Lag12								
Xday5								
Xday6								
Xmonth1								
Lag2								
Lag24								
Xmonth3								
Xmonth4								
Xday3								
S		2.42311		2.28768		2.22042		2.19287
R-sq		64.78%		68.62%		70.46%		71.21%
R-sq(adj)		64.75%		68.58%		70.40%		71.13%
R-sq(pred)		64.68%		68.53%		70.29%		70.99%
Mallows' Cp		399.51		190.62		91.93		52.98

	-----Step 5----		-----Step 6----		-----Step 7----		-----Step 8----	
	Coef	P	Coef	P	Coef	P	Coef	P
Constant	0.983		0.983		0.911		0.812	
Xbooked	1.0079	0.000	0.9997	0.000	0.9946	0.000	0.9921	0.000
Xday7	-1.991	0.000	-1.977	0.000	-1.969	0.000	-1.964	0.000
Lag3	0.1034	0.000	0.1000	0.000	0.0903	0.000	0.0875	0.000
Lag1	0.0818	0.000	0.0752	0.000	0.0692	0.000	0.0670	0.000
Lag5	0.0604	0.000	0.0605	0.000	0.0511	0.001	0.0464	0.002
Xmonth12			0.728	0.000	0.718	0.000	0.715	0.000
Lag4					0.0445	0.003	0.0405	0.008
Lag28							0.0373	0.008
Lag6								
Lag12								
Xday5								
Xday6								
Xmonth1								
Lag2								
Lag24								
Xmonth3								
Xmonth4								
Xday3								
S		2.18185		2.17348		2.16812		2.16388
R-sq		71.52%		71.75%		71.91%		72.04%
R-sq(adj)		71.42%		71.64%		71.78%		71.89%
R-sq(pred)		71.25%		71.43%		71.53%		71.62%
Mallows' Cp		38.14		27.17		20.53		15.52

	-----Step 9----		----Step 10----		----Step 11----		----Step 12----	
	Coef	P	Coef	P	Coef	P	Coef	P
Constant	0.767		0.718		0.659		0.586	
Xbooked	0.9877	0.000	0.9826	0.000	0.9854	0.000	0.9883	0.000
Xday7	-1.980	0.000	-1.985	0.000	-1.933	0.000	-1.858	0.000
Lag3	0.0827	0.000	0.0805	0.000	0.0785	0.000	0.0790	0.000
Lag1	0.0614	0.000	0.0590	0.000	0.0621	0.000	0.0650	0.000
Lag5	0.0399	0.010	0.0373	0.016	0.0378	0.015	0.0350	0.024
Xmonth12	0.724	0.000	0.748	0.000	0.742	0.000	0.736	0.000
Lag4	0.0366	0.017	0.0348	0.023	0.0320	0.037	0.0290	0.061
Lag28	0.0349	0.014	0.0326	0.021	0.0304	0.033	0.0276	0.053
Lag6	0.0366	0.016	0.0320	0.038	0.0328	0.033	0.0338	0.028
Lag12			0.0301	0.042	0.0308	0.037	0.0317	0.032
Xday5					0.309	0.044	0.398	0.013
Xday6							0.318	0.045
Xmonth1								
Lag2								
Lag24								
Xmonth3								
Xmonth4								
Xday3								
S		2.16050		2.15827		2.15611		2.15399
R-sq		72.14%		72.22%		72.29%		72.37%
R-sq(adj)		71.98%		72.04%		72.09%		72.15%
R-sq(pred)		71.68%		71.72%		71.74%		71.77%
Mallows' Cp		11.73		9.58		7.54		5.55

	----Step 13----		----Step 14----		----Step 15----		----Step 16----	
	Coef	P	Coef	P	Coef	P	Coef	P
Constant	0.525		0.483		0.434		0.427	
Xbooked	0.9889	0.000	0.9868	0.000	0.9862	0.000	0.9868	0.000
Xday7	-1.872	0.000	-1.847	0.000	-1.858	0.000	-1.863	0.000
Lag3	0.0817	0.000	0.0776	0.000	0.0774	0.000	0.0787	0.000
Lag1	0.0658	0.000	0.0619	0.000	0.0603	0.000	0.0603	0.000
Lag5	0.0388	0.013	0.0363	0.021	0.0362	0.021	0.0371	0.018
Xmonth12	0.676	0.001	0.644	0.002	0.633	0.002	0.584	0.005
Lag4	0.0323	0.038	0.0299	0.056	0.0298	0.057	0.0308	0.048
Lag28	0.0296	0.039	0.0291	0.041	0.0243	0.095	0.0260	0.075
Lag6	0.0378	0.015	0.0356	0.022	0.0351	0.024	0.0358	0.022
Lag12	0.0379	0.013	0.0367	0.016	0.0335	0.029	0.0352	0.022
Xday5	0.393	0.014	0.399	0.012	0.408	0.011	0.408	0.011
Xday6	0.309	0.052	0.339	0.034	0.347	0.030	0.347	0.030
Xmonth1	-0.446	0.079	-0.469	0.065	-0.532	0.039	-0.615	0.019
Lag2			0.0262	0.092	0.0252	0.105	0.0258	0.096
Lag24					0.0237	0.111	0.0261	0.081
Xmonth3							-0.323	0.111
Xmonth4								
Xday3								
S		2.15250		2.15120		2.15011		2.14902
R-sq		72.42%		72.47%		72.52%		72.57%
R-sq(adj)		72.19%		72.22%		72.25%		72.28%
R-sq(pred)		71.78%		71.79%		71.80%		71.82%
Mallows' Cp		4.48		3.65		3.13		2.62

	----Step 17----		----Step 18----	
	Coef	P	Coef	P
Constant	0.455		0.374	
Xbooked	0.9861	0.000	0.9867	0.000
Xday7	-1.864	0.000	-1.784	0.000
Lag3	0.0781	0.000	0.0790	0.000
Lag1	0.0597	0.000	0.0591	0.000
Lag5	0.0370	0.018	0.0385	0.014
Xmonth12	0.550	0.009	0.550	0.009
Lag4	0.0304	0.052	0.0301	0.054
Lag28	0.0279	0.057	0.0276	0.060
Lag6	0.0359	0.021	0.0356	0.022
Lag12	0.0365	0.017	0.0364	0.018
Xday5	0.406	0.011	0.488	0.004
Xday6	0.344	0.031	0.424	0.012
Xmonth1	-0.661	0.012	-0.665	0.012
Lag2	0.0252	0.104	0.0242	0.119
Lag24	0.0277	0.065	0.0277	0.065
Xmonth3	-0.363	0.075	-0.364	0.074
Xmonth4	-0.323	0.113	-0.322	0.113
Xday3			0.247	0.134
S		2.14795		2.14706
R-sq		72.61%		72.65%
R-sq(adj)		72.30%		72.33%
R-sq(pred)		71.84%		71.84%
Mallows' Cp		2.12		1.90

α to enter = 0.15, α to remove = 0.15

Stepwise to analyze external variables of Tour C step by step

Stepwise Selection of Terms

Candidate terms: Xbooked, Xmonth1, Xmonth2, Xmonth3, Xmonth4, Xmonth5, Xmonth6, Xmonth7, Xmonth8, Xmonth10, Xmonth11, Xmonth12, Xday1, Xday3, Xday4, Xday6, Xday7, Lag1, Lag2, Lag3, Lag4, Lag5, Lag6, Lag7, Lag8, Lag9, Lag10, Lag11, Lag12, Lag13, Lag14, Lag15, Lag16, Lag17, Lag18, Lag19, Lag20, Lag21, Lag22, Lag23, Lag24, Lag25, Lag26, Lag27, Lag28, Lag29, Lag30

	-----Step 1----		-----Step 2----		-----Step 3----		-----Step 4----	
	Coef	P	Coef	P	Coef	P	Coef	P
Constant	1.3118		1.4181		1.3669		1.1999	
Xbooked	1.0187	0.000	1.0394	0.000	1.0260	0.000	1.0157	0.000
Xday7			-0.839	0.000	-0.822	0.000	-0.794	0.000
Xmonth1					1.017	0.000	0.871	0.000
Lag3							0.0640	0.000
Xday1								
Lag1								
Xmonth11								
Lag5								
Xmonth12								
Xmonth2								
Lag28								
Xmonth5								
Lag26								
Lag2								
Lag11								
Xday6								
Lag25								
S		1.89730		1.87257		1.85559		1.84640
R-sq		61.80%		62.82%		63.51%		63.90%
R-sq(adj)		61.78%		62.77%		63.44%		63.80%
R-sq(pred)		61.70%		62.69%		63.29%		63.62%
Mallows' Cp		119.54		77.75		49.70		35.10

	-----Step 5----		-----Step 6----		-----Step 7----		-----Step 8----	
	Coef	P	Coef	P	Coef	P	Coef	P
Constant	1.2888		1.1853		1.1643		1.0910	
Xbooked	1.0260	0.000	1.0188	0.000	1.0166	0.000	1.0139	0.000
Xday7	-0.898	0.000	-0.886	0.000	-0.883	0.000	-0.898	0.000
Xmonth1	0.870	0.000	0.794	0.000	0.852	0.000	0.759	0.000
Lag3	0.0601	0.000	0.0546	0.001	0.0518	0.001	0.0482	0.003
Xday1	-0.460	0.000	-0.464	0.000	-0.461	0.000	-0.474	0.000
Lag1			0.0457	0.004	0.0424	0.007	0.0391	0.014
Xmonth11					0.450	0.008	0.431	0.012
Lag5							0.0375	0.019
Xmonth12								
Xmonth2								
Lag28								
Xmonth5								
Lag26								
Lag2								
Lag11								
Xday6								
Lag25								
S		1.83951		1.83507		1.83149		1.82880
R-sq		64.19%		64.38%		64.55%		64.67%
R-sq(adj)		64.07%		64.24%		64.38%		64.49%
R-sq(pred)		63.88%		64.03%		64.14%		64.20%
Mallows' Cp		24.47		18.01		13.00		9.50

	-----Step 9----		----Step 10----		-----Step 11----		-----Step 12----	
	Coef	P	Coef	P	Coef	P	Coef	P
Constant	1.0934		1.0932		1.1662		1.122	
Xbooked	1.0074	0.000	1.0022	0.000	1.0044	0.000	1.0054	0.000
Xday7	-0.892	0.000	-0.887	0.000	-0.883	0.000	-0.883	0.000
Xmonth1	0.831	0.000	0.909	0.000	0.967	0.000	0.998	0.000
Lag3	0.0444	0.006	0.0402	0.013	0.0404	0.013	0.0411	0.011
Xday1	-0.470	0.000	-0.465	0.000	-0.475	0.000	-0.475	0.000
Lag1	0.0340	0.034	0.0299	0.064	0.0306	0.058	0.0306	0.057
Xmonth11	0.481	0.005	0.534	0.002	0.511	0.003	0.547	0.002
Lag5	0.0350	0.029	0.0310	0.056	0.0325	0.045	0.0334	0.039
Xmonth12	0.358	0.041	0.423	0.017	0.449	0.012	0.483	0.007
Xmonth2			0.366	0.049	0.417	0.026	0.451	0.017
Lag28					-0.0309	0.053	-0.0306	0.055
Xmonth5							0.304	0.073
Lag26								
Lag2								
Lag11								
Xday6								
Lag25								
S		1.82689		1.82516		1.82352		1.82219
R-sq		64.77%		64.86%		64.95%		65.02%
R-sq(adj)		64.56%		64.63%		64.69%		64.74%
R-sq(pred)		64.23%		64.27%		64.31%		64.34%
Mallows' Cp		7.33		5.45		3.72		2.52

	-----Step 13----		-----Step 14----		-----Step 15----		-----Step 16----	
	Coef	P	Coef	P	Coef	P	Coef	P
Constant	1.057		1.014		0.957		0.907	
Xbooked	1.0021	0.000	1.0011	0.000	1.0013	0.000	1.0022	0.000
Xday7	-0.867	0.000	-0.855	0.000	-0.865	0.000	-0.814	0.000
Xmonth1	0.943	0.000	0.895	0.000	0.805	0.000	0.808	0.000
Lag3	0.0400	0.014	0.0379	0.019	0.0377	0.020	0.0370	0.022
Xday1	-0.468	0.000	-0.461	0.000	-0.471	0.000	-0.421	0.002
Lag1	0.0306	0.057	0.0285	0.078	0.0276	0.087	0.0288	0.075
Xmonth11	0.563	0.001	0.532	0.003	0.528	0.003	0.525	0.003
Lag5	0.0339	0.036	0.0314	0.053	0.0301	0.064	0.0290	0.075
Xmonth12	0.460	0.010	0.421	0.020	0.399	0.028	0.395	0.029
Xmonth2	0.405	0.033	0.374	0.050	0.341	0.074	0.339	0.076
Lag28	-0.0332	0.038	-0.0357	0.026	-0.0357	0.026	-0.0372	0.021
Xmonth5	0.307	0.070	0.311	0.066	0.323	0.056	0.323	0.056
Lag26	0.0285	0.076	0.0284	0.076	0.0280	0.081	0.0294	0.067
Lag2			0.0271	0.096	0.0269	0.099	0.0284	0.082
Lag11					0.0267	0.101	0.0256	0.116
Xday6							0.201	0.129
Lag25								
S		1.82091		1.81985		1.81884		1.81806
R-sq		65.09%		65.16%		65.22%		65.27%
R-sq(adj)		64.79%		64.84%		64.87%		64.90%
R-sq(pred)		64.36%		64.37%		64.38%		64.37%
Mallows' Cp		1.39		0.65		-0.01		-0.28

	-----Step 17----	
	Coef	P
Constant	0.857	
Xbooked	1.0033	0.000
Xday7	-0.811	0.000
Xmonth1	0.766	0.000
Lag3	0.0376	0.020
Xday1	-0.423	0.002
Lag1	0.0285	0.078
Xmonth11	0.531	0.003
Lag5	0.0289	0.076
Xmonth12	0.369	0.043
Xmonth2	0.301	0.119
Lag28	-0.0399	0.014
Xmonth5	0.325	0.055
Lag26	0.0271	0.093
Lag2	0.0277	0.090
Lag11	0.0254	0.119
Xday6	0.210	0.113
Lag25	0.0240	0.137
S		1.81733
R-sq		65.32%
R-sq(adj)		64.93%
R-sq(pred)		64.37%
Mallows' Cp		-0.48

α to enter = 0.15, α to remove = 0.15

SARIMA computation code in Spyder (Python) for Tour A

```python
@author: PORNPAWIT NIAMJOY
"""
import numpy as np
import pandas as pd
data = pd.read_csv('A.csv',index_col=0)
data.head()
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
data.index = pd.to_datetime(data.index)
data.columns = ['TourlistA']
data.plot()
pyplot.show()
from pmdarima.arima import auto_arima
ypreds=[]
ny = 365 #numday of year
nd = 7 #numday per week
wf =26 #numweek forecast
ycv = data.iloc[(len(data)-ny):(len(data)-ny+(nd*wf))]
#check
i=0
y_to_train = data.iloc[(i*nd):(len(data)-ny+(i*nd))]

for i in range(0, wf):
        y_to_train = data.iloc[(i*nd):(len(data)-ny+(i*nd))]
        model = auto_arima(y_to_train, seasonal=True, m=nd)
        model.fit(y_to_train)
        ypred = model.predict(n_periods=nd)
        ypreds = np.append(ypreds,ypred)

ycv= ycv.to_numpy()
ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
ypreds= ypreds[:(nd*wf)]
ypreds = np.round(ypreds)
ypreds[ypreds < 0 ] =0.0

def mape(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_pred - y_true) / y_true)) * 100

print('Test MAE:', mean_absolute_error(ycv, ypreds))
print('Test RMSE:',np.sqrt(mean_squared_error(ycv, ypreds)))

xxx=[]
for x in range (0, len(ycv)):
        if ycv[x][0]==0:
                xxx=np.append(xxx,x)
ycvs =np.delete(ycv, np.s_[xxx], axis=0)
ypredss =np.delete(ypreds, np.s_[xxx], axis=0)

print('Test MAPE:',mape(ycvs, ypredss))
```

SARIMA computation code in Spyder (Python) for Tour B

```
@author: PORNPAWIT NIAMJOY
"""
import numpy as np
import pandas as pd
data = pd.read_csv('B.csv',index_col=0)
data.head()
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
data.index = pd.to_datetime(data.index)
data.columns = ['TourlistB']
data.plot()
pyplot.show()
from pmdarima.arima import auto_arima
ypreds=[]
ny = 313 #numday of year
nd = 6 #numday per week
wf =26 #numweek forecast
ycv = data.iloc[(len(data)-ny):(len(data)-ny+(nd*wf))]
#check
i=0
y_to_train = data.iloc[(i*nd):(len(data)-ny+(i*nd))]

for i in range(0, wf):
        y_to_train = data.iloc[(i*nd):(len(data)-ny+(i*nd))]
        model = auto_arima(y_to_train, seasonal=True, m=nd)
        model.fit(y_to_train)
        ypred = model.predict(n_periods=nd)
        ypreds = np.append(ypreds,ypred)

ycv= ycv.to_numpy()
ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
ypreds= ypreds[:(nd*wf)]
ypreds = np.round(ypreds)
ypreds[ypreds < 0 ] =0.0

def mape(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_pred - y_true) / y_true)) * 100

print('Test MAE:', mean_absolute_error(ycv, ypreds))
print('Test RMSE:',np.sqrt(mean_squared_error(ycv, ypreds)))

xxx=[]
for x in range (0, len(ycv)):
        if ycv[x][0]==0:
                xxx=np.append(xxx,x)
ycvs =np.delete(ycv, np.s_[xxx], axis=0)
ypredss =np.delete(ypreds, np.s_[xxx], axis=0)

print('Test MAPE:',mape(ycvs, ypredss))
```

SARIMA computation code in Spyder (Python) for Tour C

```
@author: PORNPAWIT NIAMJOY
"""
import numpy as np
import pandas as pd
data = pd.read_csv('C.csv',index_col=0)
data.head()
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
data.index = pd.to_datetime(data.index)
data.columns = ['TourlistC']
data.plot()
pyplot.show()
from pmdarima.arima import auto_arima
ypreds=[]
ny = 313 #numday of year
nd = 6 #numday per week
wf =26 #numweek forecast
ycv = data.iloc[(len(data)-ny):(len(data)-ny+(nd*wf))]
#check
i=0
y_to_train = data.iloc[(i*nd):(len(data)-ny+(i*nd))]

for i in range(0, wf):
        y_to_train = data.iloc[(i*nd):(len(data)-ny+(i*nd))]
        model = auto_arima(y_to_train, seasonal=True, m=nd)
        model.fit(y_to_train)
        ypred = model.predict(n_periods=nd)
        ypreds = np.append(ypreds,ypred)

ycv= ycv.to_numpy()
ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
ypreds= ypreds[:(nd*wf)]
ypreds = np.round(ypreds)
ypreds[ypreds < 0 ] =0.0

def mape(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_pred - y_true) / y_true)) * 100

print('Test MAE:', mean_absolute_error(ycv, ypreds))
print('Test RMSE:',np.sqrt(mean_squared_error(ycv, ypreds)))

xxx=[]
for x in range (0, len(ycv)):
        if ycv[x][0]==0:
                xxx=np.append(xxx,x)
ycvs =np.delete(ycv, np.s_[xxx], axis=0)
ypredss =np.delete(ypreds, np.s_[xxx], axis=0)

print('Test MAPE:',mape(ycvs, ypredss))
```

SARIMAX computation code in Spyder (Python) for Tour A

```
@author: PORNPAWIT NIAMJOY
"""

import numpy as np
import pandas as pd
data = pd.read_csv('A.csv',index_col=0)
exog = pd.read_csv('exogA.csv',index_col=0)
data.head()
exog.head()
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
data.index = pd.to_datetime(data.index)
exog.index = pd.to_datetime(exog.index)
data.columns = ['TourlistA']
data.plot()
pyplot.show()
from pmdarima.arima import auto_arima
ypreds=[]
ny = 365 #numday of year
nd = 7 #numday per week
wf =26 #numweek forecast
pe =13 #penalty shift+13 for A +11 for BC
ycv = data.iloc[(len(data)-ny):(len(data)-ny+(nd*wf))]
#check
i=0
y_to_traincheck = data.iloc[pe+(i*nd):(len(data)-ny+(i*nd))]
exog_to_traincheck = exog.iloc[((i*nd)):(len(data)-ny+(i*nd)-pe)]
exog_to_testcheck = exog.iloc[(len(data)-ny+(i*nd)-pe):(len(data)-ny+(i*nd)+nd-pe)]
for i in range(0, wf):
        y_to_train = data.iloc[pe+(i*nd):(len(data)-ny+(i*nd))]
        exog_to_train = exog.iloc[((i*nd)):(len(data)-ny+(i*nd)-pe)]
        exog_to_test = exog.iloc[(len(data)-ny+(i*nd)-pe):(len(data)-ny+(i*nd)+nd-pe)]
        arima_exog_model = auto_arima(y=y_to_train, exogenous=exog_to_train, seasonal=True, m=nd)
        ypred = arima_exog_model.predict(n_periods=nd, exogenous=exog_to_test)
        ypreds = np.append(ypreds,ypred)
ycv= ycv.to_numpy()
ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
ypreds= ypreds[:(nd*wf)]
ypreds = np.round(ypreds)
ypreds[ypreds < 0 ] =0.0
def mape(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_pred - y_true) / y_true)) * 100
print('Test MAE:', mean_absolute_error(ycv, ypreds))
print('Test RMSE:',np.sqrt(mean_squared_error(ycv, ypreds)))
xxx=[]
for x in range (0, len(ycv)):
        if ycv[x][0]==0:
                xxx=np.append(xxx,x)
ycvs =np.delete(ycv, np.s_[xxx], axis=0)
ypredss =np.delete(ypreds, np.s_[xxx], axis=0)
print('Test MAPE:',mape(ycvs, ypredss))
```

SARIMAX computation code in Spyder (Python) for Tour B

```python
@author: PORNPAWIT NIAMJOY
"""

import numpy as np
import pandas as pd
data = pd.read_csv('B.csv',index_col=0)
exog = pd.read_csv('exogB.csv',index_col=0)
data.head()
exog.head()
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
data.index = pd.to_datetime(data.index)
exog.index = pd.to_datetime(exog.index)
data.columns = ['TourlistB']
data.plot()
pyplot.show()
from pmdarima.arima import auto_arima
ypreds=[]
ny = 313 #numday of year
nd = 6 #numday per week
wf =26 #numweek forecast
pe =11 #penalty shift+13 for A +11 for BC
ycv = data.iloc[(len(data)-ny):(len(data)-ny+(nd*wf))]
#check
i=0
y_to_traincheck = data.iloc[pe+(i*nd):(len(data)-ny+(i*nd))]
exog_to_traincheck = exog.iloc[((i*nd)):(len(data)-ny+(i*nd)-pe)]
exog_to_testcheck = exog.iloc[(len(data)-ny+(i*nd)-pe):(len(data)-ny+(i*nd)+nd-pe)]
for i in range(0, wf):
        y_to_train = data.iloc[pe+(i*nd):(len(data)-ny+(i*nd))]
        exog_to_train = exog.iloc[((i*nd)):(len(data)-ny+(i*nd)-pe)]
        exog_to_test = exog.iloc[(len(data)-ny+(i*nd)-pe):(len(data)-ny+(i*nd)+nd-pe)]
        arima_exog_model = auto_arima(y=y_to_train, exogenous=exog_to_train, seasonal=True, m=nd)
        ypred = arima_exog_model.predict(n_periods=nd, exogenous=exog_to_test)
        ypreds = np.append(ypreds,ypred)
ycv= ycv.to_numpy()
ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
ypreds= ypreds[:(nd*wf)]
ypreds = np.round(ypreds)
ypreds[ypreds < 0 ] =0.0
def mape(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_pred - y_true) / y_true)) * 100
print('Test MAE:', mean_absolute_error(ycv, ypreds))
print('Test RMSE:',np.sqrt(mean_squared_error(ycv, ypreds)))
xxx=[]
for x in range (0, len(ycv)):
        if ycv[x][0]==0:
                xxx=np.append(xxx,x)
ycvs =np.delete(ycv, np.s_[xxx], axis=0)
ypredss =np.delete(ypreds, np.s_[xxx], axis=0)
print('Test MAPE:',mape(ycvs, ypredss))
```

SARIMAX computation code in Spyder (Python) for Tour C

```
@author: PORNPAWIT NIAMJOY
"""

import numpy as np
import pandas as pd
data = pd.read_csv('C.csv',index_col=0)
exog = pd.read_csv('exogC.csv',index_col=0)
data.head()
exog.head()
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
data.index = pd.to_datetime(data.index)
exog.index = pd.to_datetime(exog.index)
data.columns = ['TourlistC']
data.plot()
pyplot.show()
from pmdarima.arima import auto_arima
ypreds=[]
ny = 313 #numday of year
nd = 6 #numday per week
wf =26 #numweek forecast
pe =11 #penalty shift+13 for A +11 for BC
ycv = data.iloc[(len(data)-ny):(len(data)-ny+(nd*wf))]
#check
i=0
y_to_traincheck = data.iloc[pe+(i*nd):(len(data)-ny+(i*nd))]
exog_to_traincheck = exog.iloc[((i*nd)):(len(data)-ny+(i*nd)-pe)]
exog_to_testcheck = exog.iloc[(len(data)-ny+(i*nd)-pe):(len(data)-ny+(i*nd)+nd-pe)]
for i in range(0, wf):
        y_to_train = data.iloc[pe+(i*nd):(len(data)-ny+(i*nd))]
        exog_to_train = exog.iloc[((i*nd)):(len(data)-ny+(i*nd)-pe)]
        exog_to_test = exog.iloc[(len(data)-ny+(i*nd)-pe):(len(data)-ny+(i*nd)+nd-pe)]
        arima_exog_model = auto_arima(y=y_to_train, exogenous=exog_to_train, seasonal=True, m=nd)
        ypred = arima_exog_model.predict(n_periods=nd, exogenous=exog_to_test)
        ypreds = np.append(ypreds,ypred)
ycv= ycv.to_numpy()
ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
ypreds= ypreds[:(nd*wf)]
ypreds = np.round(ypreds)
ypreds[ypreds < 0 ] =0.0
def mape(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_pred - y_true) / y_true)) * 100
print('Test MAE:', mean_absolute_error(ycv, ypreds))
print('Test RMSE:',np.sqrt(mean_squared_error(ycv, ypreds)))
xxx=[]
for x in range (0, len(ycv)):
        if ycv[x][0]==0:
                xxx=np.append(xxx,x)
ycvs =np.delete(ycv, np.s_[xxx], axis=0)
ypredss =np.delete(ypreds, np.s_[xxx], axis=0)
print('Test MAPE:',mape(ycvs, ypredss))
```

TBATS computation code in Spyder (Python) for Tour A

```
@author: PORNPAWIT NIAMJOY
"""

import numpy as np
import pandas as pd
data = pd.read_csv('A.csv',index_col=0)
data.head()
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
data.index = pd.to_datetime(data.index)
data.columns = ['TourlistA']
data.plot()
pyplot.show()
from tbats import TBATS, BATS
ypreds=[]
ny = 365 #numday of year
nd = 7 #numday per week
wf =26 #numweek forecast
ycv = data.iloc[(len(data)-ny):(len(data)-ny+(nd*wf))]
#check
i=0
y_to_train = data.iloc[(i*nd):(len(data)-ny+(i*nd))]


for i in range(0, wf):
        y_to_train = data.iloc[(i*nd):(len(data)-ny+(i*nd))]
        estimator = TBATS(seasonal_periods=[nd, 365.25])
        model = estimator.fit(y_to_train)
        ypred = model.forecast(steps=nd)
        ypreds = np.append(ypreds,ypred)
ycv= ycv.to_numpy()
ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
ypreds= ypreds[:(nd*wf)]
ypreds = np.round(ypreds)
ypreds[ypreds < 0 ] =0.0
def mape(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_pred - y_true) / y_true)) * 100
print('Test MAE:', mean_absolute_error(ycv, ypreds))
print('Test RMSE:',np.sqrt(mean_squared_error(ycv, ypreds)))
xxx=[]
for x in range (0, len(ycv)):
        if ycv[x][0]==0:
                xxx=np.append(xxx,x)
ycvs =np.delete(ycv, np.s_[xxx], axis=0)
ypredss =np.delete(ypreds, np.s_[xxx], axis=0)
print('Test MAPE:',mape(ycvs, ypredss))
```

TBATS computation code in Spyder (Python) for Tour B

```
@author: PORNPAWIT NIAMJOY
"""

import numpy as np
import pandas as pd
data = pd.read_csv('B.csv',index_col=0)
data.head()
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
data.index = pd.to_datetime(data.index)
data.columns = ['TourlistB']
data.plot()
pyplot.show()
from tbats import TBATS, BATS
ypreds=[]
ny = 313 #numday of year
nd = 6 #numday per week
wf =26 #numweek forecast
ycv = data.iloc[(len(data)-ny):(len(data)-ny+(nd*wf))]
#check
i=0
y_to_train = data.iloc[(i*nd):(len(data)-ny+(i*nd))]


for i in range(0, wf):
        y_to_train = data.iloc[(i*nd):(len(data)-ny+(i*nd))]
        estimator = TBATS(seasonal_periods=[nd, 365.25])
        model = estimator.fit(y_to_train)
        ypred = model.forecast(steps=nd)
        ypreds = np.append(ypreds,ypred)
ycv= ycv.to_numpy()
ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
ypreds= ypreds[:(nd*wf)]
ypreds = np.round(ypreds)
ypreds[ypreds < 0 ] =0.0
def mape(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_pred - y_true) / y_true)) * 100
print('Test MAE:', mean_absolute_error(ycv, ypreds))
print('Test RMSE:',np.sqrt(mean_squared_error(ycv, ypreds)))
xxx=[]
for x in range (0, len(ycv)):
        if ycv[x][0]==0:
                xxx=np.append(xxx,x)
ycvs =np.delete(ycv, np.s_[xxx], axis=0)
ypredss =np.delete(ypreds, np.s_[xxx], axis=0)
print('Test MAPE:',mape(ycvs, ypredss))
```

TBATS computation code in Spyder (Python) for Tour C

```
@author: PORNPAWIT NIAMJOY
"""

import numpy as np
import pandas as pd
data = pd.read_csv('C.csv',index_col=0)
data.head()
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
data.index = pd.to_datetime(data.index)
data.columns = ['TourlistC']
data.plot()
pyplot.show()
from tbats import TBATS, BATS
ypreds=[]
ny = 313 #numday of year
nd = 6 #numday per week
wf =26 #numweek forecast
ycv = data.iloc[(len(data)-ny):(len(data)-ny+(nd*wf))]
#check
i=0
y_to_train = data.iloc[(i*nd):(len(data)-ny+(i*nd))]


for i in range(0, wf):
        y_to_train = data.iloc[(i*nd):(len(data)-ny+(i*nd))]
        estimator = TBATS(seasonal_periods=[nd, 365.25])
        model = estimator.fit(y_to_train)
        ypred = model.forecast(steps=nd)
        ypreds = np.append(ypreds,ypred)
ycv= ycv.to_numpy()
ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
ypreds= ypreds[:(nd*wf)]
ypreds = np.round(ypreds)
ypreds[ypreds < 0 ] =0.0
def mape(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_pred - y_true) / y_true)) * 100
print('Test MAE:', mean_absolute_error(ycv, ypreds))
print('Test RMSE:',np.sqrt(mean_squared_error(ycv, ypreds)))
xxx=[]
for x in range (0, len(ycv)):
        if ycv[x][0]==0:
                xxx=np.append(xxx,x)
ycvs =np.delete(ycv, np.s_[xxx], axis=0)
ypredss =np.delete(ypreds, np.s_[xxx], axis=0)
print('Test MAPE:',mape(ycvs, ypredss))
```

ANN computation code in Spyder (Python) for Tour A

```
@author: PORNPAWIT NIAMJOY
"""
import numpy as np
import pandas as pd
import tensorflow as tf
tf.__version__
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from keras.models import Sequential
from keras.layers import Dense
from numpy.random import seed
seed(1)
tf.random.set_seed(1)
#--------setting-----------------------------------
ny = 365 #numday of year
nd = 7*26 #numday per period(week)
fp =1 #forecast period(week)
lb =4 #lookback week
#--------Importing the dataset----------------------
datasetx = pd.read_csv('exogA.csv')
X = datasetx.iloc[:,1:].values
datasety = pd.read_csv('A.csv')
y = datasety.iloc[:,1:].values
#--------look back----------------------------------
def create_dataset(dataset, look_back=1):
    X, Y = [], []
    for i in range(len(dataset)-look_back):
        a = dataset[(i):(i+look_back), 0]
        if i%7 ==0 :
                X.append(a)
                X.append(a)
                X.append(a)
                X.append(a)
                X.append(a)
                X.append(a)
                X.append(a)
        Y.append(dataset[i + look_back, 0])
    return np.array(X), np.array(Y)
#--------create traning data-------------------------
look_back = 2+(7*lb)
X_lag, Y = create_dataset(y ,look_back)
if len(Y)!= len(X_lag):
        X_lag =X_lag[:len(Y)]
Xbook=X[look_back:,:1]
Xori=X
X=np.concatenate(( Xbook, X_lag), axis=1)
#---------dummy var---------------------------------
X = np.concatenate(( X, Xori[look_back:,7:]), axis=1)
#---------var---------------------------------------
#ycv = y[(len(y)-ny):(len(y)-ny+(nd*fp))]#cv
#ycv = y[(len(y)-ny+(fp*nd)):] #test
ycv = y[2+(7*lb):+(len(y)-ny)] #trian
ypreds=[]
mae=[]
rmse=[]
mapes=[]
```

```
measure=[]
s2=[]
n2=[]
e2=[]
b2=[]
maemin=100
#-------------mape--------------------------------
def mape(y_true, y_pred):
        y_true, y_pred = np.array(y_true), np.array(y_pred)
        return np.mean(np.abs((y_pred - y_true) / y_true)) * 100
#------------model--,2,----------------------------------
for s in [1]:#seed
        for n in range(5,101):#hidden unit(neuron)
                for e in [10,30,50,70,100]:#epoch
                        for b in [10,20,32,64]:#batch
                                for p in range(0, fp):
                                        tf.random.set_seed(s)
                                        #--------train cv split---------------------------------
                                        y_to_train = Y[(p*nd):(len(Y)-ny+(p*nd))]
                                        #y_to_cv = Y[(len(Y)-ny+(p*nd)):(len(Y)-ny+(p*nd)+nd)]#CV
                                        #y_to_cv = Y[(len(Y)-ny+(p*nd)+(fp*nd)):] #test
                                        x_to_train = X[(p*nd):(len(X)-ny+(p*nd))]
                                        #x_to_cv = X[(len(X)-ny+(p*nd)):(len(X)-ny+(p*nd)+nd)]#CV
                                        #x_to_cv = X[(len(X)-ny+(p*nd)+(fp*nd)):] #test
                                        #--------Scaling-------------------------------------
                                        from sklearn.preprocessing import MinMaxScaler
                                        from sklearn.preprocessing import StandardScaler
                                        sc = MinMaxScaler(feature_range=(0, 1))
                                        sc.fit(x_to_train)
                                        x_to_train = sc.transform(x_to_train)
                                        x_to_cv = sc.transform(x_to_cv)
                                        #----------ANN-----------------------------------------
                                        ann = tf.keras.models.Sequential()
                                        ann.add(tf.keras.layers.Dense(units=n, activation='sigmoid'))
                                        ann.add(tf.keras.layers.Dense(units=n, activation='sigmoid'))
                                        ann.add(tf.keras.layers.Dense(units=1, activation='relu'))
                                        ann.compile(optimizer = 'adam', loss = 'mean_squared_error', metrics = ['accuracy'])
                                        ann.fit(x_to_train, y_to_train, batch_size = b, epochs = e)
                                        ypred = ann.predict(x_to_cv)
                                        ypreds = np.append(ypreds,ypred)
                                        #---------------------------------------------------------
                                        print('seed=:',s)
                                        print('hidden unit=:',n)
                                        print('epoch=:',e)
                                        print('batch size=:',b)
                                        print('period=:',p,'/',fp)
                                ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
                                ypreds = np.round(ypreds)
                                ypreds[ypreds < 0 ] =0.0
                                s2 = np.append(s2,s)
                                n2 = np.append(n2,n)
                                e2 = np.append(e2,e)
                                b2 = np.append(b2,b)
                                print('Test MAE:', mean_absolute_error(ycv, ypreds))
                                mae=np.append(mae,mean_absolute_error(ycv, ypreds))
                                print('Test RMSE:',np.sqrt(mean_squared_error(ycv, ypreds)))
                                rmse=np.append(rmse,np.sqrt(mean_squared_error(ycv, ypreds)))
                                #-----delete 0 row for mape cal-----------------------------
                                delete=[]
                                for x in range (0, len(ycv)):
                                        if ycv[x][0]==0:
                                                delete=np.append(delete,x)
                                ycvs =np.delete(ycv, np.s_[delete], axis=0)
                                ypredss =np.delete(ypreds, np.s_[delete], axis=0)
                                print('Test MAPE:',mape(ycvs, ypredss))
                                mapes=np.append(mapes,mape(ycvs, ypredss))
                                if mean_absolute_error(ycv, ypreds)<maemin:
                                        mapemin=mape(ycvs, ypredss)
                                        masemin=np.sqrt(mean_squared_error(ycv, ypreds))
                                        maemin=mean_absolute_error(ycv, ypreds)
                                        smin=s
                                        nmin=n
                                        emin=e
                                        bmin=b
                                        ypredbest = ypreds
                                ypreds=[]
s2 = np.reshape(s2, (s2.shape[0], 1))
n2 = np.reshape(n2, (n2.shape[0], 1))
e2 = np.reshape(e2, (e2.shape[0], 1))
b2 = np.reshape(b2, (b2.shape[0], 1))
maes = np.reshape(mae, (mae.shape[0], 1))
rmses = np.reshape(rmse, (rmse.shape[0], 1))
mapess = np.reshape(mapes, (mapes.shape[0], 1))
measure = np.concatenate((maes,rmses,mapess,s2,n2,e2,b2),axis=1)
```

ANN computation code in Spyder (Python) for Tour B

```
@author: PORNPAWIT NIAMJOY
"""
import numpy as np
import pandas as pd
import tensorflow as tf
tf.__version__
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from keras.models import Sequential
from keras.layers import Dense
from numpy.random import seed
seed(1)
tf.random.set_seed(1)
#--------setting-----------------------------------
ny = 313 #numday of year
nd = 6*26 #numday per period(week)
fp =1 #forecast period(week)
lb =4 #lookback week
#--------Importing the dataset----------------------
datasetx = pd.read_csv('exogB.csv')
X = datasetx.iloc[:,1:].values
datasety = pd.read_csv('B.csv')
y = datasety.iloc[:,1:].values
#--------Look back---------------------------------
def create_dataset(dataset, look_back=1):
    X, Y = [], []
    for i in range(len(dataset)-look_back):
        a = dataset[(i):(i+look_back), 0]
        if i%6 ==0 :
                X.append(a)
                X.append(a)
                X.append(a)
                X.append(a)
                X.append(a)
                X.append(a)
                #X.append(a)
        Y.append(dataset[i + look_back, 0])
    return np.array(X), np.array(Y)
#--------create traning data------------------------
look_back = 2+(6*lb)
X_lag, Y = create_dataset(y ,look_back)
if len(Y)!= len(X_lag):
        X_lag =X_lag[:len(Y)]
Xbook=X[look_back:,:1]
Xori=X
X=np.concatenate(( Xbook, X_lag), axis=1)
#---------dummy var-----------------------------------
X = np.concatenate(( X, Xori[look_back:,7:]), axis=1)
#---------var-----------------------------------------
#ycv = y[(len(y)-ny):(len(y)-ny+(nd*fp))]#cv
#ycv = y[(len(y)-ny+(fp*nd)):] #test
ycv = y[2+(6*lb):+(len(y)-ny)] #trian
ypreds=[]
mae=[]
rmse=[]
mapes=[]
```

```python
measure=[]
s2=[]
n2=[]
e2=[]
b2=[]
maemin=100
#-------------mape---------------------------------
def mape(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_pred - y_true) / y_true)) * 100
#-----------model--,2,-----------------------------------
for s in [1]:#seed
    for n in range(5,101):#hidden unit(neuron)
        for e in [10,30,50,70,100]:#epoch
            for b in [10,20,32,64]:#batch
                for p in range(0, fp):
                    tf.random.set_seed(s)
                    #--------train cv split-----------------------------------
                    y_to_train = Y[(p*nd):(len(Y)-ny+(p*nd))]
                    #y_to_cv = Y[(len(Y)-ny+(p*nd)):(len(Y)-ny+(p*nd)+nd)]#CV
                    #y_to_cv = Y[(len(Y)-ny+(p*nd)+(fp*nd)):] #test
                    x_to_train = X[(p*nd):(len(X)-ny+(p*nd))]
                    #x_to_cv = X[(len(X)-ny+(p*nd)):(len(X)-ny+(p*nd)+nd)]#CV
                    #x_to_cv = X[(len(X)-ny+(p*nd)+(fp*nd)):] #test
                    #--------Scaling------------------------------------------
                    from sklearn.preprocessing import MinMaxScaler
                    from sklearn.preprocessing import StandardScaler
                    sc = MinMaxScaler(feature_range=(0, 1))
                    sc.fit(x_to_train)
                    x_to_train = sc.transform(x_to_train)
                    x_to_cv = sc.transform(x_to_cv)
                    #----------ANN----------------------------------------------
                    ann = tf.keras.models.Sequential()
                    ann.add(tf.keras.layers.Dense(units=n, activation='sigmoid'))
                    ann.add(tf.keras.layers.Dense(units=n, activation='sigmoid'))
                    ann.add(tf.keras.layers.Dense(units=1, activation='relu'))
                    ann.compile(optimizer = 'adam', loss = 'mean_squared_error', metrics = ['accuracy'])
                    ann.fit(x_to_train, y_to_train, batch_size = b, epochs = e)
                    ypred = ann.predict(x_to_cv)
                    ypreds = np.append(ypreds,ypred)
                    #---------------------------------------------------------
                    print('seed=:',s)
                    print('hidden unit=:',n)
                    print('epoch=:',e)
                    print('batch size=:',b)
                    print('period=:',p,'/',fp)
                ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
                ypreds = np.round(ypreds)
                ypreds[ypreds < 0 ] =0.0
                s2 = np.append(s2,s)
                n2 = np.append(n2,n)
                e2 = np.append(e2,e)
                b2 = np.append(b2,b)
                print('Test MAE:', mean_absolute_error(ycv, ypreds))
                mae=np.append(mae,mean_absolute_error(ycv, ypreds))
                print('Test RMSE:',np.sqrt(mean_squared_error(ycv, ypreds)))
                rmse=np.append(rmse,np.sqrt(mean_squared_error(ycv, ypreds)))
                #-----delete 0 row for mape cal-----------------------------
                delete=[]
                for x in range (0, len(ycv)):
                    if ycv[x][0]==0:
                        delete=np.append(delete,x)
                ycvs =np.delete(ycv, np.s_[delete], axis=0)
                ypredss =np.delete(ypreds, np.s_[delete], axis=0)
                print('Test MAPE:',mape(ycvs, ypredss))
                mapes=np.append(mapes,mape(ycvs, ypredss))
                if mean_absolute_error(ycv, ypreds)<maemin:
                    mapemin=mape(ycvs, ypredss)
                    masemin=np.sqrt(mean_squared_error(ycv, ypreds))
                    maemin=mean_absolute_error(ycv, ypreds)
                    smin=s
                    nmin=n
                    emin=e
                    bmin=b
                    ypredbest = ypreds
                ypreds=[]
s2 = np.reshape(s2, (s2.shape[0], 1))
n2 = np.reshape(n2, (n2.shape[0], 1))
e2 = np.reshape(e2, (e2.shape[0], 1))
b2 = np.reshape(b2, (b2.shape[0], 1))
maes = np.reshape(mae, (mae.shape[0], 1))
rmses = np.reshape(rmse, (rmse.shape[0], 1))
mapess = np.reshape(mapes, (mapes.shape[0], 1))
measure = np.concatenate((maes,rmses,mapess,s2,n2,e2,b2),axis=1)
```

ANN computation code in Spyder (Python) for Tour C

```
@author: PORNPAWIT NIAMJOY
"""
import numpy as np
import pandas as pd
import tensorflow as tf
tf.__version__
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from keras.models import Sequential
from keras.layers import Dense
from numpy.random import seed
seed(1)
tf.random.set_seed(1)
#--------setting-----------------------------------
ny = 313 #numday of year
nd = 6*26 #numday per period(week)
fp =1 #forecast period(week)
lb =4 #lookback week
#--------Importing the dataset---------------------
datasetx = pd.read_csv('exogC.csv')
X = datasetx.iloc[:,1:].values
datasety = pd.read_csv('C.csv')
y = datasety.iloc[:,1:].values
#--------look back---------------------------------
def create_dataset(dataset, look_back=1):
    X, Y = [], []
    for i in range(len(dataset)-look_back):
        a = dataset[(i):(i+look_back), 0]
        if i%6 ==0 :
                X.append(a)
                X.append(a)
                X.append(a)
                X.append(a)
                X.append(a)
                X.append(a)
                #X.append(a)
        Y.append(dataset[i + look_back, 0])
    return np.array(X), np.array(Y)
#--------create traning data------------------------
look_back = 2+(6*lb)
X_lag, Y = create_dataset(y ,look_back)
if len(Y)!= len(X_lag):
        X_lag =X_lag[:len(Y)]
Xbook=X[look_back:,:1]
Xori=X
X=np.concatenate(( Xbook, X_lag), axis=1)
#---------dummy var---------------------------------
X = np.concatenate(( X, Xori[look_back:,7:]), axis=1)
#---------var---------------------------------------
#ycv = y[(len(y)-ny):(len(y)-ny+(nd*fp))]#cv
#ycv = y[(len(y)-ny+(fp*nd)):] #test
ycv = y[2+(6*lb):+(len(y)-ny)] #trian
ypreds=[]
mae=[]
rmse=[]
mapes=[]
```

```python
measure=[]
s2=[]
n2=[]
e2=[]
b2=[]
maemin=100
#-------------mape---------------------------------
def mape(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_pred - y_true) / y_true)) * 100
#-----------model--,2,----------------------------------
for s in [1]:#seed
    for n in range(5,101):#hidden unit(neuron)
        for e in [10,30,50,70,100]:#epoch
            for b in [10,20,32,64]:#batch
                for p in range(0, fp):
                    tf.random.set_seed(s)
                    #--------train cv split----------------------------------
                    y_to_train = Y[(p*nd):(len(Y)-ny+(p*nd))]
                    #y_to_cv = Y[(len(Y)-ny+(p*nd)):(len(Y)-ny+(p*nd)+nd)]#CV
                    #y_to_cv = Y[(len(Y)-ny+(p*nd)+(fp*nd)):] #test
                    x_to_train = X[(p*nd):(len(X)-ny+(p*nd))]
                    #x_to_cv = X[(len(X)-ny+(p*nd)):(len(X)-ny+(p*nd)+nd)]#CV
                    #x_to_cv = X[(len(X)-ny+(p*nd)+(fp*nd)):] #test
                    #--------Scaling------------------------------------------
                    from sklearn.preprocessing import MinMaxScaler
                    from sklearn.preprocessing import StandardScaler
                    sc = MinMaxScaler(feature_range=(0, 1))
                    sc.fit(x_to_train)
                    x_to_train = sc.transform(x_to_train)
                    x_to_cv = sc.transform(x_to_cv)
                    #----------ANN---------------------------------------------
                    ann = tf.keras.models.Sequential()
                    ann.add(tf.keras.layers.Dense(units=n, activation='sigmoid'))
                    ann.add(tf.keras.layers.Dense(units=n, activation='sigmoid'))
                    ann.add(tf.keras.layers.Dense(units=1, activation='relu'))
                    ann.compile(optimizer = 'adam', loss = 'mean_squared_error', metrics = ['accuracy'])
                    ann.fit(x_to_train, y_to_train, batch_size = b, epochs = e)
                    ypred = ann.predict(x_to_cv)
                    ypreds = np.append(ypreds,ypred)
                    #----------------------------------------------------------
                    print('seed=:',s)
                    print('hidden unit=:',n)
                    print('epoch=:',e)
                    print('batch size=:',b)
                    print('period=:',p,'/',fp)
                ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
                ypreds = np.round(ypreds)
                ypreds[ypreds < 0 ] =0.0
                s2 = np.append(s2,s)
                n2 = np.append(n2,n)
                e2 = np.append(e2,e)
                b2 = np.append(b2,b)
                print('Test MAE:', mean_absolute_error(ycv, ypreds))
                mae=np.append(mae,mean_absolute_error(ycv, ypreds))
                print('Test RMSE:',np.sqrt(mean_squared_error(ycv, ypreds)))
                rmse=np.append(rmse,np.sqrt(mean_squared_error(ycv, ypreds)))
                #-----delete 0 row for mape cal---------------------------
                delete=[]
                for x in range (0, len(ycv)):
                    if ycv[x][0]==0:
                        delete=np.append(delete,x)
                ycvs =np.delete(ycv, np.s_[delete], axis=0)
                ypredss =np.delete(ypreds, np.s_[delete], axis=0)
                print('Test MAPE:',mape(ycvs, ypredss))
                mapes=np.append(mapes,mape(ycvs, ypredss))
                if mean_absolute_error(ycv, ypreds)<maemin:
                    mapemin=mape(ycvs, ypredss)
                    masemin=np.sqrt(mean_squared_error(ycv, ypreds))
                    maemin=mean_absolute_error(ycv, ypreds)
                    smin=s
                    nmin=n
                    emin=e
                    bmin=b
                    ypredbest = ypreds
                ypreds=[]
s2 = np.reshape(s2, (s2.shape[0], 1))
n2 = np.reshape(n2, (n2.shape[0], 1))
e2 = np.reshape(e2, (e2.shape[0], 1))
b2 = np.reshape(b2, (b2.shape[0], 1))
maes = np.reshape(mae, (mae.shape[0], 1))
rmses = np.reshape(rmse, (rmse.shape[0], 1))
mapess = np.reshape(mapes, (mapes.shape[0], 1))
measure = np.concatenate((maes,rmses,mapess,s2,n2,e2,b2),axis=1)
```

LSTM computation code in Spyder (Python) for Tour A

```
@author: PORNPAWIT NIAMJOY
"""
import numpy as np
import pandas as pd
import tensorflow as tf
tf.__version__
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
from numpy.random import seed
seed(1)
tf.random.set_seed(1)
#--------setting-----------------------------------
ny = 365 #numday of year
nd = 7 #numday per period(week)
fp = 26 #forecast period(week)
lb = 4 #lookback week
#--------Importing the dataset----------------------
datasety = pd.read_csv('A.csv')
y = datasety.iloc[:,1:].values
#--------look back----------------------------------
def create_dataset(dataset, look_back=1):
    X, Y = [], []
    for i in range(round((len(dataset)-look_back-nd+1)/7)):
        a = dataset[(i*7):((i*7)+look_back), 0]
        X.append(a)
        b = dataset[((i*7)+look_back):((i*7)+look_back+nd), 0]
        Y.append(b)
    return np.array(X), np.array(Y)
#--------create traning data------------------------
look_back = 2+(7*lb)
X_lag, Y = create_dataset(y ,look_back)
X = X_lag
#---------var------------------------------------------
#ycv = y[(len(y)-ny):(len(y)-ny+(nd*fp))]#cv
#ycv = y[(len(y)-ny+(fp*nd)):] #test
ycv = y[2+(7*lb):+(len(y)-ny)] #trian
ypreds=[]
mae=[]
rmse=[]
mapes=[]
measure=[]
s2=[]
n2=[]
e2=[]
b2=[]
mapemin=100
#-------------mape---------------------------------
def mape(y_true, y_pred):
        y_true, y_pred = np.array(y_true), np.array(y_pred)
        return np.mean(np.abs((y_pred - y_true) / y_true)) * 100
#-----------model---------------------------------------
for s in [1]:#seed
        for n in range(5,101):#hidden unit(neuron)
                for e in [10,30,50,70,100]:#epoch
                        for b in [10,20,32,64]:#batch
                                tf.random.set_seed(s)
                                p=0
                                y_to_train = Y[:204]
                                x_to_train = X[:204]
                                #--------Scaling---------------------------------------
                                from sklearn.preprocessing import MinMaxScaler
                                from sklearn.preprocessing import StandardScaler
                                sc = MinMaxScaler(feature_range=(0, 1))
                                sc.fit(x_to_train)
                                x_to_train = sc.transform(x_to_train)
                                #---- reshape input to be [samples, time steps, features]----
                                X_train = np.reshape(x_to_train, (x_to_train.shape[0], 1, x_to_train.shape[1]))
                                #----------LSTM-----------------------------------------
                                model = Sequential()
                                model.add(LSTM(n,return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2]),activation='relu'))
                                model.add(LSTM(n,activation='relu'))
                                model.add(Dense(units=7))
                                model.compile(loss='mean_squared_error', optimizer='adam', metrics = ['accuracy'])
                                model.fit(X_train, y_to_train, epochs=e, batch_size=b)
                                #model.summary()
                                #-------------predict------------------------------------
                                for p in range(0, fp+1):
                                        x_to_cv = X[204:]
                                        #x_to_cv = X[:204]
                                        x_to_cv = sc.transform(x_to_cv)
                                        X_cv = np.reshape(x_to_cv, (x_to_cv.shape[0], 1, x_to_cv.shape[1]))
                                        #train_predict = model.predict(X_train)
                                        ypred = model.predict(X_cv)
                                        ypreds = np.append(ypreds,ypred)
                                #ypreds = ypreds[3:3+182]
                                #-------------------------------------------------------
                                print('seed=:',s)
                                print('hidden unit=:',n)
                                print('epoch=:',e)
                                print('batch size=:',b)
                                print('period=:',p,'/',fp)
                                ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
                                ypreds = np.round(ypreds)
                                ypreds[ypreds < 0 ] =0.0
                                s2 = np.append(s2,s)
                                n2 = np.append(n2,n)
                                e2 = np.append(e2,e)
                                b2 = np.append(b2,b)
                                print('Test MAE:', mean_absolute_error(ycv, ypreds))
                                mae=np.append(mae,mean_absolute_error(ycv, ypreds))
                                print('Test RMSE:',np.sqrt(mean_squared_error(ycv, ypreds)))
                                rmse=np.append(rmse,np.sqrt(mean_squared_error(ycv, ypreds)))
```

```python
                #-----delete 0 row for mape cal-----------------------------
                delete=[]
                for x in range (0, len(ycv)):
                        if ycv[x][0]==0:
                                delete=np.append(delete,x)
                ycvs =np.delete(ycv, np.s_[delete], axis=0)
                ypredss =np.delete(ypreds, np.s_[delete], axis=0)
                print('Test MAPE:',mape(ycvs, ypredss))
                mapes=np.append(mapes,mape(ycvs, ypredss))
                if mape(ycvs, ypredss)<mapemin:
                        mapemin=mape(ycvs, ypredss)
                        masemin=np.sqrt(mean_squared_error(ycv, ypreds))
                        maemin=mean_absolute_error(ycv, ypreds)
                        smin=s
                        nmin=n
                        emin=e
                        bmin=b
                        ypredbest = ypreds
                ypreds=[]
s2 = np.reshape(s2, (s2.shape[0], 1))
n2 = np.reshape(n2, (n2.shape[0], 1))
e2 = np.reshape(e2, (e2.shape[0], 1))
b2 = np.reshape(b2, (b2.shape[0], 1))
maes = np.reshape(mae, (mae.shape[0], 1))
rmses = np.reshape(rmse, (rmse.shape[0], 1))
mapess = np.reshape(mapes, (mapes.shape[0], 1))
measure = np.concatenate((maes,rmses,mapess,s2,n2,e2,b2),axis=1)
```

LSTM computation code in Spyder (Python) for Tour B

```python
@author: MOS
"""
import numpy as np
import pandas as pd
import tensorflow as tf
tf.__version__
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
from numpy.random import seed
import random as rn
import os
os.environ["CUDA_DEVICE_ORDER"] = "PCI_BUS_ID"
os.environ["CUDA_VISIBLE_DEVICES"] = ""
#--------setting------------------------------------
ny = 313 #numday of year
nd = 6 #numday per period(week)
fp = 26 #forecast period(week)
lb = 4 #lookback week
#--------Importing the dataset----------------------
datasety = pd.read_csv('B.csv')
y = datasety.iloc[:,1:].values
#--------look back------------------------------------
def create_dataset(dataset, look_back=1):
    X, Y = [], []
    for i in range(round((len(dataset)-look_back-nd+1)/6)):
        a = dataset[(i*6):((i*6)+look_back), 0]
        X.append(a)
        b = dataset[((i*6)+look_back):((i*6)+look_back+nd), 0]
        Y.append(b)
    return np.array(X), np.array(Y)
#--------create traning data-------------------------
look_back = 2+(6*lb)
X_lag, Y = create_dataset(y ,look_back)
X = X_lag
#---------var------------------------------------------
#ycv = y[(len(y)-ny):(len(y)-ny+(nd*fp))]#cv
#ycv = y[len(y)-9-157:len(y)-9] #test
ycv = y[2+(6*lb):+(len(y)-ny)-2] #trian
ypreds=[]
mae=[]
rmse=[]
mapes=[]
measure=[]
s2=[]
n2=[]
e2=[]
b2=[]
maemin=100
#-------------mape---------------------------------
def mape(y_true, y_pred):
        y_true, y_pred = np.array(y_true), np.array(y_pred)
        return np.mean(np.abs((y_pred - y_true) / y_true)) * 100
```

```
#-----------model-------------------------------------
for s in [1]:#seed
        for n in range(5,101):#hidden unit(neuron)
                for e in [10,30,50,70,100]:#epoch
                        for b in [10,20,32,64]:#batch
                                y_to_train = Y[:204]
                                x_to_train = X[:204]
                                #---------Scaling--------------------------------------------
                                from sklearn.preprocessing import MinMaxScaler
                                from sklearn.preprocessing import StandardScaler
                                #sc = StandardScaler()
                                sc = MinMaxScaler(feature_range=(0, 1))
                                sc.fit(x_to_train)
                                x_to_train = sc.transform(x_to_train)
                                #---- reshape input to be [samples, time steps, features]----
                                X_train = np.reshape(x_to_train, (x_to_train.shape[0], 1, x_to_train.shape[1]))
                                #----------LSTM---------------------------------------------------
                                rn.seed(s)
                                np.random.seed(s)
                                tf.random.set_seed(s)
                                model = Sequential()
                                #model.add(LSTM(n,return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2]),activation='relu'))
                                model.add(LSTM(n, input_shape=(X_train.shape[1], X_train.shape[2]),activation='relu'))
                                #model.add(LSTM(n,activation='relu'))
                                model.add(Dense(units=6))
                                model.compile(loss='mean_squared_error', optimizer='adam', metrics = ['accuracy'])
                                model.fit(X_train, y_to_train, epochs=e, batch_size=b)
                                #model.summary()
                                #--------------predict-------------------------------------
                                for p in range(0, fp+1):
                                        rn.seed(s)
                                        np.random.seed(s)
                                        tf.random.set_seed(s)
                                #x_to_cv = X[230:]#test
                                x_to_cv = X[204:]#cv
                                #x_to_cv = X[:204]#train
                                x_to_cv = sc.transform(x_to_cv)
                                X_cv = np.reshape(x_to_cv, (x_to_cv.shape[0], 1, x_to_cv.shape[1]))
                                #train_predict = model.predict(X_train)
                                ypred = model.predict(X_cv)
                                ypreds = np.append(ypreds,ypred)
                                ypreds = ypreds[2:2+156]
                                #------------------------------------------------------------
                                print('seed=:',s)
                                print('hidden unit=:',n)
                                print('epoch=:',e)
                                print('batch size=:',b)
                                print('period=:',p,'/',fp)
                                ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
                                ypreds = np.round(ypreds)
                                ypreds[ypreds < 0 ] =0.0
                                s2 = np.append(s2,s)
                                n2 = np.append(n2,n)
                                e2 = np.append(e2,e)
                                b2 = np.append(b2,b)
```

LSTM computation code in Spyder (Python) for Tour C

```
@author: MOS
"""
import numpy as np
import pandas as pd
import tensorflow as tf
tf.__version__
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
from numpy.random import seed
import random as rn
import os
os.environ["CUDA_DEVICE_ORDER"] = "PCI_BUS_ID"
os.environ["CUDA_VISIBLE_DEVICES"] = ""
#--------setting--------------------------------------
ny = 313 #numday of year
nd = 6 #numday per period(week)
fp = 26 #forecast period(week)
lb = 4 #lookback week
#--------Importing the dataset----------------------
dataset = pd.read_csv('C.csv')
y = datasety.iloc[:,1:].values
#--------Look back----------------------------------
def create_dataset(dataset, look_back=1):
    X, Y = [], []
    for i in range(round((len(dataset)-look_back-nd+1)/6)):
        a = dataset[(i*6):((i*6)+look_back), 0]
        X.append(a)
        b = dataset[((i*6)+look_back):((i*6)+look_back+nd), 0]
        Y.append(b)
    return np.array(X), np.array(Y)
#--------create traning data------------------------
look_back = 2+(6*lb)
X_lag, Y = create_dataset(y ,look_back)
X = X_lag
#---------var-----------------------------------------
#ycv = y[(len(y)-ny):(len(y)-ny+(nd*fp))]#cv
#ycv = y[len(y)-9-157:len(y)-9] #test
ycv = y[2+(6*lb):+(len(y)-ny)-2] #trian
ypreds=[]
mae=[]
rmse=[]
mapes=[]
measure=[]
s2=[]
n2=[]
e2=[]
b2=[]
maemin=100
#-------------mape--------------------------------
def mape(y_true, y_pred):
        y_true, y_pred = np.array(y_true), np.array(y_pred)
        return np.mean(np.abs((y_pred - y_true) / y_true)) * 100
#-----------model-------------------------------------
```

```
#------------model-----------------------------------
for s in [1]:#seed
        for n in range(5,101):#hidden unit(neuron)
                for e in [10,30,50,70,100]:#epoch
                        for b in [10,20,32,64]:#batch
                                y_to_train = Y[:204]
                                x_to_train = X[:204]
                                #---------Scaling-------------------------------------------
                                from sklearn.preprocessing import MinMaxScaler
                                from sklearn.preprocessing import StandardScaler
                                #sc = StandardScaler()
                                sc = MinMaxScaler(feature_range=(0, 1))
                                sc.fit(x_to_train)
                                x_to_train = sc.transform(x_to_train)
                                #---- reshape input to be [samples, time steps, features]----
                                X_train = np.reshape(x_to_train, (x_to_train.shape[0], 1, x_to_train.shape[1]))
                                #----------LSTM-----------------------------------------------
                                rn.seed(s)
                                np.random.seed(s)
                                tf.random.set_seed(s)
                                model = Sequential()
                                #model.add(LSTM(n,return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2]),activation='relu'))
                                model.add(LSTM(n, input_shape=(X_train.shape[1], X_train.shape[2]),activation='relu'))
                                #model.add(LSTM(n,activation='relu'))
                                model.add(Dense(units=6))
                                model.compile(loss='mean_squared_error', optimizer='adam', metrics = ['accuracy'])
                                model.fit(X_train, y_to_train, epochs=e, batch_size=b)
                                #model.summary()
                                #--------------predict-------------------------------------------
                                for p in range(0, fp+1):
                                        rn.seed(s)
                                        np.random.seed(s)
                                        tf.random.set_seed(s)
                                #x_to_cv = X[230:]#test
                                x_to_cv = X[204:]#cv
                                #x_to_cv = X[:204]#train
                                x_to_cv = sc.transform(x_to_cv)
                                X_cv = np.reshape(x_to_cv, (x_to_cv.shape[0], 1, x_to_cv.shape[1]))
                                #train_predict = model.predict(X_train)
                                ypred = model.predict(X_cv)
                                ypreds = np.append(ypreds,ypred)
                                ypreds = ypreds[2:2+156]
                                #-----------------------------------------------------------
                                print('seed=:',s)
                                print('hidden unit=:',n)
                                print('epoch=:',e)
                                print('batch size=:',b)
                                print('period=:',p,'/',fp)
                                ypreds = np.reshape(ypreds, (ypreds.shape[0], 1))
                                ypreds = np.round(ypreds)
                                ypreds[ypreds < 0 ] =0.0
                                s2 = np.append(s2,s)
                                n2 = np.append(n2,n)
                                e2 = np.append(e2,e)
                                b2 = np.append(b2,b)
                                print('Test MAE:', mean_absolute_error(ycv, ypreds))
                                mae=np.append(mae,mean_absolute_error(ycv, ypreds))
                                print('Test RMSE:',np.sqrt(mean_squared_error(ycv, ypreds)))
                                rmse=np.append(rmse,np.sqrt(mean_squared_error(ycv, ypreds)))
                                #-----delete 0 row for mape cal-------------------------------
                                delete=[]
                                for x in range (0, len(ycv)):
                                        if ycv[x][0]==0:
                                                delete=np.append(delete,x)
                                ycvs =np.delete(ycv, np.s_[delete], axis=0)
                                ypredss =np.delete(ypreds, np.s_[delete], axis=0)
                                print('Test MAPE:',mape(ycvs, ypredss))
                                mapes=np.append(mapes,mape(ycvs, ypredss))
                                if mean_absolute_error(ycv, ypreds)<maemin:
                                        mapemin=mape(ycvs, ypredss)
                                        masemin=np.sqrt(mean_squared_error(ycv, ypreds))
                                        maemin=mean_absolute_error(ycv, ypreds)
                                        smin=s
                                        nmin=n
                                        emin=e
                                        bmin=b
                                        ypredbest = ypreds
                                ypreds=[]
s2 = np.reshape(s2, (s2.shape[0], 1))
n2 = np.reshape(n2, (n2.shape[0], 1))
e2 = np.reshape(e2, (e2.shape[0], 1))
b2 = np.reshape(b2, (b2.shape[0], 1))
maes = np.reshape(mae, (mae.shape[0], 1))
rmses = np.reshape(rmse, (rmse.shape[0], 1))
mapess = np.reshape(mapes, (mapes.shape[0], 1))
measure = np.concatenate((maes,rmses,mapess,s2,n2,e2,b2),axis=1)
```

REFERENCES

Arunraj, Nari Sivanandam, and Diane Ahrens. 2015. 'A hybrid seasonal autoregressive integrated moving average and quantile regression for daily food sales forecasting', *International Journal of Production Economics*, 170: 321-35.

colah'sblog. 2015,August 27. 'Understanding LSTM Networks'. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

De Livera, Alysha M. 2010. 'Modeling time series with complex seasonal patterns using exponential smoothing', Monash University.

De Livera, Alysha M, Rob J Hyndman, and Ralph D Snyder. 2011. 'Forecasting time series with complex seasonal patterns using exponential smoothing', *Journal of the American statistical association*, 106: 1513-27.

Divino, Jose Angelo, and Michael McAleer. 2010. 'Modelling and forecasting daily international mass tourism to Peru', *Tourism Management*, 31: 846-54.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. 'Long Short-term Memory', *Neural computation*, 9: 1735-80.

Kingma, Diederik, and Jimmy Ba. 2014. 'Adam: A Method for Stochastic Optimization', *International Conference on Learning Representations*.

McCulloch, Warren S., and Walter Pitts. 1943. 'A logical calculus of the ideas immanent in nervous activity', *The bulletin of mathematical biophysics*, 5: 115-33.

Osowski, Stanislaw, and Konrad Garanty. 2007. 'Forecasting of the daily meteorological pollution using wavelets and support vector machine', *Engineering Applications of Artificial Intelligence*, 20: 745-55.

Paoli, Christophe, Cyril Voyant, Marc Muselli, and Marie-Laure Nivet. 2010. 'Forecasting of preprocessed daily solar radiation time series using neural networks', *Solar Energy*, 84: 2146-60.

Pereira, Luis Nobre. 2016. 'An introduction to helpful forecasting methods for hotel revenue management', *International Journal of Hospitality Management*, 58: 13-23.

Prybutok, Victor R, Junsub Yi, and David Mitchell. 2000. 'Comparison of neural network models with ARIMA and regression models for prediction of Houston's daily maximum ozone concentrations', *European Journal of Operational Research*, 122: 31-40.

Rajopadhye, Mihir, Mounir Ben Ghalia, Paul P Wang, Timothy Baker, and Craig V Eister. 2001. 'Forecasting uncertain hotel room demand', *Information sciences*, 132: 1-11.

Taylor, James W. 2007. 'Forecasting daily supermarket sales using exponentially weighted quantile regression', *European Journal of Operational Research*, 178: 154-67.

TowardsDataScience. 2017, August 16. 'Statistics is Freaking Hard: WTF is Activation function'. https://towardsdatascience.com/statistics-is-freaking-hard-wtf-is-activation-function-df8342cdf292.

tutorialspoint. 2020. 'Artificial Intelligence - Neural Networks'. https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm.

Wan, Feng Jiao. 2012. "Research on the demand forcast of Beijing car rental industry based on system dynamics." In *Applied Mechanics and Materials*, 267-70. Trans Tech Publ.

Weatherford, Larry R, and Sheryl E Kimes. 2003. 'A comparison of forecasting methods for hotel revenue management', *International journal of forecasting*, 19: 401-15.

Witt, Stephen, and Haiyan Song. 2001. 'Forecasting future tourism flows.' in, *Tourism and hospitality in the 21st century* (Elsevier).

YüKsel, Sedat. 2007. 'An integrated forecasting approach to hotel demand', *Mathematical and Computer Modelling*, 46: 1063-70.

VITA

NAME	พรภวิษย์ เนียมจ้อย
DATE OF BIRTH	25 มกราคม 2538
PLACE OF BIRTH	สงขลา, ไทย
INSTITUTIONS ATTENDED	จุฬาลงกรณ์มหาวิทยาลัย
HOME ADDRESS	129/5 ม.3 ซอย วัดศรีประวัติ ต.ศาลากลาง อ.บางกรวย จ.นนทบุรี 11130