A comparative study between statistical and machine learning methods for forecasting retail sales

Shanmugapriya Murugavel

Student No: R00195696

Supervisor: Francisco Hernandez

For the module DATA9003 – Research Project as part of the Master of Science in Data Science and Analytics, Department of Mathematics, 11th December 2022

Declaration of Authorship

I, Shanmugapriya Murugavel, declare that this thesis titled '**A comparative study between statistical and machine learning methods for forecasting retail sales**' and the work presented in it are my own. I confirm that,

- This work was done wholly or mainly while in candidature for the Masters' degree at Munster Technological University
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Munster Technological University or any other institution, this has been clearly stated
- Where I have consulted the published work of others, this is always clearly attributed
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work
- I have acknowledged all main sources of help
- I understand that my project documentation may be stored in the library at MTU, and may be referenced by others in the future

Signed: Shanmugapriya Murugavel

Date : 08.12.2022

Acknowledgement

Throughout the journey of this dissertation and my course, I have received a great deal of support and assistance from all my professors.

I am extremely grateful to my supervisor, Professor Francisco Hernandez, for his continued support and guidance throughout this project. Your extreme support and meaningful feedback pushed me to this high level of thinking. I would also like to express my gratitude to Professor Aengus Daly for his expertise and ideas for the improvement of this research work and his recommendations in formulating the research question and methodology.

Further, I would like to mention my mentor Dr Vincent Cregan for his valuable guidance throughout my studies and all other lecturers for giving me this opportunity. I could not have undertaken this journey without my manager and accountant at Centra who provided me with the dataset. To conclude, I like to thank my family and friends for all their unconditional support in completing this intense Masters' programme.

Table of Contents

List of Figures

Abstract

Forecasting techniques are widely used in various sectors. From predicting climatic changes, healthcare systems, agriculture crop simulation, and stock market management to business forecasting including demand, supply, and sales forecasting. Sales prediction is one of the forecasting methods which predicts retailing of products that are to be sold in the future. Sales forecast additionally serves as a tool for identifying benchmarks and determining incremental results of new initiatives, planning resources to meet demand, and projecting future costs. The forecasting process helps retailers to stock products based on customer requirements in turn improving the supply chain management process. During unexpected situations like the pandemic, sales of a few items such as milk, bread and toilet paper were completely unpredictable. This is an example where sales prediction plays a major impact. The purpose of this paper is to compare the performance in predicting sales of various machine learning and statistical time series models. As there are strong seasonal fluctuations observed, several regression approaches like LSTM, Linear Regression and Random Forests were used to predict future sales using the historically available data. Finally, the models were compared based on the RMSE scores to evaluate the performances.

1. Introduction

Growth in technologies has widely influenced business performances across all sectors. Many industries have completely transformed their interactions and services towards their customers. These new innovations in various fields have helped businesses to better understand customer needs for the future. Sales forecasting is one of the most common and essential tasks that is performed in both small businesses and multinational conglomerates. This also supports companies to stay on track with their desired goals, effectively allocate resources and manage the forthcoming growth of the business. Accurate sales forecasts help a healthy business environment and make the shareholders happy as sales are considered the lifeline of any company. The main aim behind sales forecasting is to precisely estimate sales performances.

Sales forecasts mostly use historical data and industry trends to analyse weekly, monthly, quarterly, and annual sales. This forecasted information is used for making impactful business decisions like allocating resources, hiring new staff during busy hours, increasing inventory stock, or managing future sales. Sales forecasting is not just predicting the sales numbers, instead, it provides business leaders with the necessary information to make the right decisions that help the business achieve better goals. It is considered extensive among both the industrialists and the manufacturers which helps in better understanding the shelf-life of the goods and determining their production. In earlier days, companies produce supplies without considering demand and sales, which in turn led to the loss of revenue in both surplus and shortage situations.

Many factors are considered while selecting a suitable method before prediction, which includes the relativity and availability of the historical data, a forecasted time period, the time available to make the analysis, the benefit for the company through the forecast made and also the purpose of the forecast. Considering all these above factors, suitable techniques are

considered for the analysis and the results are then compared, to choose the most appropriate technique depending on each case. However, the forecasting process is still challenging to make predictions as the raw data collected is mixed up with high trends, seasonal variations and the sales offer made during the period.

There are several tools used in the industry for forecasting. Some of these tools include Microsoft Excel, CRM (Customer Relationship Management), Sales analytical tools, project management tools and accounting software such as QuickBooks. The sales forecast is mainly built on assumptions.

A combination of Artificial Intelligence, Time Series, and Statistics is changing traditional forecasting methods. These methods are data-driven and are mostly based on quantitative analysis of past information. Changes in retail demand over the years, the global economy, weather, and the introduction of new products are some of the external factors which impact the patterns of these data. In the past, sales prediction was seen as an arduous task. But modern forecasting techniques have become game changers. Modern techniques like Machine Learning and Artificial Intelligence mainly use probability and statistics for forecasting sales.

This thesis aims to develop a sales prediction model which can accurately predict the future sales of the retail store using its historical data. The dataset was collected from a local retail store. Five different techniques: i) Holt-Winters' exponential smoothing techniques; ii) Auto-Regressive Integrated Moving Average (ARIMA); iii) a statistical approach using Linear Regression; iv) a time series approach based on Long Short-Term Memory (LSTM); v) and a machine learning approach based on Random Forest were considered.

These models were then compared based on their accuracy and the value of residuals. Linear regression analysis is used to predict the value of a dependent variable based on the value of an independent variable. The LSTM are a kind of Recurrent Neural Network (RNN) having a

memory which helps in analysing a time series [1]. Random Forest combines the output of multiple decision trees to obtain the desired value. Having work experience in the retail industry, this field was chosen as an interest.

1.1 Time Series Regression Models

The time series regression model uses sequential data collected over a time period at different points in regular intervals. Time series analysis helps in answering questions like, what will be the effect of variable Y of a change in variable X *over time*? Here, Y is considered the dependent variable and X is the independent variable. In the dataset collected, the sales of the store are analysed over the years. So, Sales data becomes the dependent variable Y – *to be predicted* and time is the independent variable X – *already available.* Since the analysis to be performed is based on the time period, Time Series analysis is chosen as the major method of implementation.

Sequential sales data is collected starting from October 2015 to April 2022. The data is collected over a span of seven years, in a weekly periodicity. To better explain, how time series can be helpful in further sales analysis consider the following example. To forecast monthly sales y using a total number of customers visited x as a predictor. The forecasted variable y is also called the explained or dependent variable [2]. The predictor variable x is also called the explanatory or the independent variable.

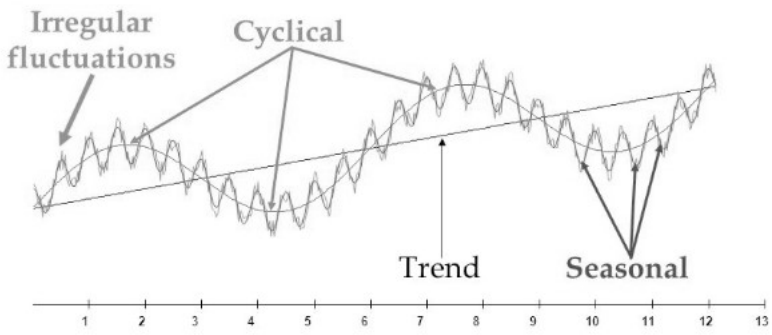


Figure 1: Components of a Time Series [3]

A time series can be identified into four different components: the seasonal variation, the cyclical variation, the secular trend and the residual variation. The combination of these four components results in an irregular time series. An irregular time series is represented in above *figure 1.*

Apart from these components, there exists another term periodicity which also plays an important role in a time series. It is usually described as the number of time points at which the data is recorded during one completion of a seasonal pattern. The periodicity can be collected hourly, daily, weekly, monthly or even quarterly or annually. Here, the dataset is of weekly and monthly periodicity.

1.2 Machine Learning/Artificial Intelligence Regression Models

The processes and algorithms which simulate a machine to mimic human intelligence is called Artificial Intelligence. Personal assistant models like Alexa, and Siri that mainly understands spoken language, modern web search engines, personalised ads, and self-driving vehicles are some everyday applications of AI. AI/ML models are used in various industries namely: Healthcare, Telecommunications, Financial Services, Insurance/Recharge renewals, Automobile, etc.,

The major advantage of choosing a machine learning technique for performing the sales/demand forecasting here for the available dataset is that these ML models are designed in such a way to handle thousand metrics of data at a single time and have the capacity to handle them accurately. Machine Learning involves the process of training a model to analyse the sales from the historical data available, learn how each input factor impacts a weighted output, and finally use the testing model to predict future sales [3].

Here LSTM, the Long Short-Term Memory method is chosen due to its capability to handle long-term dependencies among the data. These work on the technology of Recurrent Neural Networks (RNN), which remembers the previous information and use them for processing the current input.

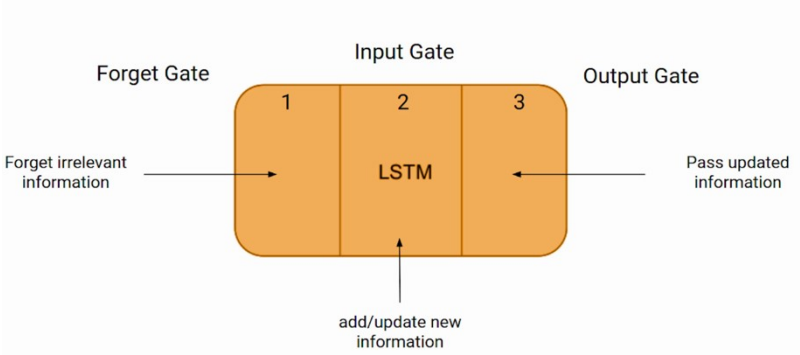


Figure 2: Architecture of an LSTM cell [4]

Here the first part of the Forget gate decides whether the information coming from the previous timestamp is to be remembered or not. If the data feels irrelevant the previous data is forgotten. In the second part, known as the Input gate the cell tries learning new information from the current input. In the output gate, the information is passed from the current timestamp.

2. Literature Review

This section discusses the research papers related to sales forecasting and how it is being implemented by industries of all domains, especially retail. There have been relatively few works related to this topic of study.

The eXtreme Gradient Boosting (XGBoost) Machine Learning technique was utilized in [5] to accurately forecast sales amounts. The Walmart retail dataset from the Kaggle competition was used for this analysis. The reason behind selecting the XGBoost technique was that it was more scalable and efficient, making it ten times faster than other existing machine learning models. The efficiency of the algorithm was evaluated using the RMSE score obtained - 0.655, which was about 16.3% lower when compared to other Linear Regression models [6] and 15.4% lower than the Ridge Regression method [7].

One similar work was done in [8] which analysed the same Walmart Sales data based on Light Gradient Boost Method (LightGBM). The results obtained indicated the RMSE of the model was 2.09, which was better when compared to linear regression models and Support Vector Machine models (SVM). Finally, the model with the highest accuracy was selected to make future predictions. This motivated the study to perform a comparison by designing a model using linear regression.

Many other studies have also excelled in the same accurate results but using different implementation techniques such as Time Series modelling. One such noticeable work was from [9]. The analysis was performed based on historical sales data from Amazon.com a leading e-commerce website that is available worldwide. At first, three popular forecasting approaches – Holt-Winters exponential smoothing, Autoregressive integrated moving average (ARIMA) and Neural network autoregression were performed. The sales data forecasted Amazon's quarterly

sales in the year 2019. Later the forecasted sales were tested against the actual sales numbers in 2019 to compare the results. The accuracy was measured using Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE). The results showed that seasonal ARIMA gave the most accurate results as compared to the other two methods. Above are some of the standard methods used in this type of analysis and a few similar approaches are to be used in this study.

In the comparative study between LSTM and ARIMA conducted in [10]. The models predicted sales for one day ahead and the results were evaluated using RMSE and MAE together along with a t-test. The study concludes that the LSTM model was much more promising in forecasting sales in retail when compared to the ARIMA models. A similar approach is to be performed in this research based on LSTM models.

Even though various comparative studies between traditional and modern forecasting methods have been conducted in the past, findings are mixed with regard to the linearity of the data. For example, in a comparative study for forecasting electric load between the ARIMA model and

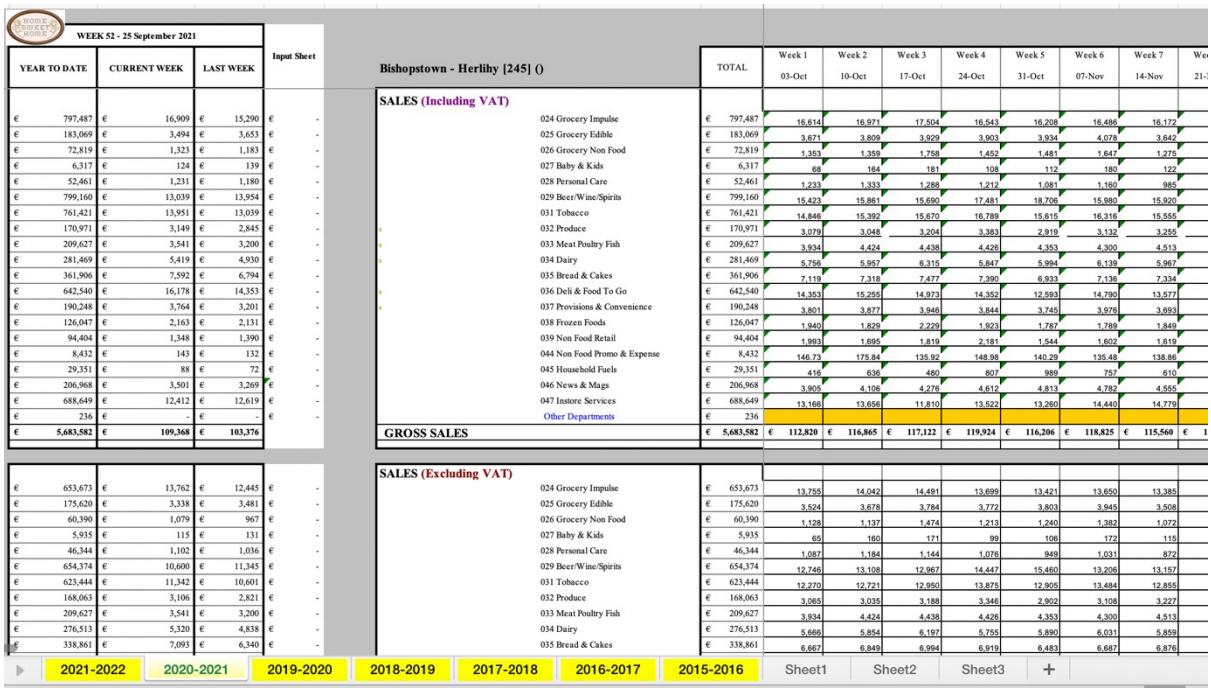
Few other results from [11], showed Neural Networks had better predictions than the linear ARIMA model. But the case of a comparative study, conducted with nonlinear data [12], proved Neural Networks were not as effective as their linear time series models.

On the other hand, no study has been conducted to compare a traditional statistical model, a time series-based model and a modern machine learning model to our knowledge. The next sections of the paper are organised to perform this comparative research. The research methodology of the study is discussed in the next section.

3. Methodology

3.1 Dataset

The sales data used in this study was obtained from a retail store based in Cork in Microsoft Excel format. The total sampling period examined is from October 2015 to May 2022. As part of this research, sales is the main component to be analysed. Figure 3 gives us a visual representation of the dataset.



WEEK 52 - 25 September 2021

YEAR TO DATE	CURRENT WEEK	LAST WEEK	Input Sheet	Bishopstown - Herlihy [245] ()	TOTAL	Week 1 03-Oct	Week 2 10-Oct	Week 3 17-Oct	Week 4 24-Oct	Week 5 31-Oct	Week 6 07-Nov	Week 7 14-Nov
				SALES (Including VAT)								
€ 797,487	€ 16,909	€ 15,290	€ -	024 Grocery Impulse	€ 797,487	16,614	16,971	17,504	16,543	16,208	16,486	16,172
€ 183,069	€ 3,494	€ 3,653	€ -	025 Grocery Edible	€ 183,069	3,671	3,809	3,929	3,903	3,934	4,078	3,642
€ 72,819	€ 1,323	€ 1,183	€ -	026 Grocery Non Food	€ 72,819	1,353	1,359	1,758	1,452	1,481	1,647	1,275
€ 6,317	€ 124	€ 139	€ -	027 Baby & Kids	€ 6,317	68	164	181	108	112	180	122
€ 52,461	€ 1,231	€ 1,180	€ -	028 Personal Care	€ 52,461	1,233	1,333	1,288	1,212	1,081	1,160	985
€ 799,160	€ 13,039	€ 13,954	€ -	029 Beer/Wine/Spirits	€ 799,160	15,423	15,861	15,690	17,481	18,706	15,980	15,920
€ 761,421	€ 13,951	€ 13,039	€ -	031 Tobacco	€ 761,421	14,846	15,392	15,670	16,789	15,615	16,316	15,555
€ 170,971	€ 3,149	€ 2,845	€ -	032 Produce	€ 170,971	3,079	3,048	3,204	3,383	2,919	3,132	3,255
€ 209,627	€ 3,541	€ 3,200	€ -	033 Meat Poultry Fish	€ 209,627	3,934	4,424	4,438	4,426	4,353	4,300	4,513
€ 281,469	€ 5,419	€ 4,930	€ -	034 Dairy	€ 281,469	5,758	5,957	6,315	5,847	5,994	6,139	5,967
€ 361,906	€ 7,592	€ 6,794	€ -	035 Bread & Cakes	€ 361,906	7,119	7,318	7,477	7,390	6,933	7,136	7,334
€ 642,540	€ 16,178	€ 14,353	€ -	036 Deli & Food To Go	€ 642,540	14,353	15,255	14,973	14,352	12,593	14,790	13,577
€ 190,248	€ 3,764	€ 3,201	€ -	037 Provisions & Convenience	€ 190,248	3,801	3,877	3,946	3,844	3,745	3,976	3,693
€ 126,047	€ 2,163	€ 2,131	€ -	038 Frozen Foods	€ 126,047	1,940	1,829	2,229	1,923	1,787	1,789	1,849
€ 94,404	€ 1,348	€ 1,390	€ -	039 Non Food Retail	€ 94,404	1,993	1,695	1,819	2,181	1,544	1,602	1,610
€ 8,432	€ 143	€ 132	€ -	044 Non Food Promo & Expense	€ 8,432	146.73	175.84	135.92	148.98	140.29	135.48	138.86
€ 29,351	€ 88	€ 72	€ -	045 Household Fuels	€ 29,351	416	636	480	807	989	757	610
€ 206,968	€ 3,501	€ 3,269	€ -	046 News & Mags	€ 206,968	3,905	4,106	4,276	4,612	4,813	4,782	4,555
€ 688,649	€ 12,412	€ 12,619	€ -	047 Instore Services	€ 688,649	13,166	13,656	11,810	13,522	13,260	14,440	14,779
€ 236	€ -	€ -	€ -	Other Departments	€ 236							
€ 5,683,582	€ 109,368	€ 103,376		GROSS SALES	€ 5,683,582	€ 112,820	€ 116,865	€ 117,122	€ 119,924	€ 116,206	€ 118,825	€ 115,560
				SALES (Excluding VAT)								
€ 653,673	€ 13,762	€ 12,445	€ -	024 Grocery Impulse	€ 653,673	13,755	14,042	14,491	13,699	13,421	13,650	13,385
€ 175,620	€ 3,338	€ 3,481	€ -	025 Grocery Edible	€ 175,620	3,524	3,678	3,784	3,772	3,803	3,945	3,508
€ 60,390	€ 1,079	€ 967	€ -	026 Grocery Non Food	€ 60,390	1,128	1,137	1,474	1,213	1,240	1,382	1,072
€ 5,935	€ 115	€ 131	€ -	027 Baby & Kids	€ 5,935	65	160	171	99	106	172	115
€ 46,344	€ 1,102	€ 1,036	€ -	028 Personal Care	€ 46,344	1,087	1,184	1,144	1,076	949	1,031	872
€ 654,374	€ 10,600	€ 11,345	€ -	029 Beer/Wine/Spirits	€ 654,374	12,746	13,108	12,967	14,447	15,460	13,206	13,157
€ 623,444	€ 11,342	€ 10,601	€ -	031 Tobacco	€ 623,444	12,270	12,721	12,950	13,875	12,905	13,484	12,855
€ 168,063	€ 3,106	€ 2,821	€ -	032 Produce	€ 168,063	3,065	3,035	3,188	3,346	2,902	3,108	3,227
€ 209,627	€ 3,541	€ 3,200	€ -	033 Meat Poultry Fish	€ 209,627	3,934	4,424	4,438	4,426	4,353	4,300	4,513
€ 276,513	€ 5,320	€ 4,838	€ -	034 Dairy	€ 276,513	5,666	5,854	6,197	5,755	5,890	6,031	5,859
€ 338,861	€ 7,093	€ 6,340	€ -	035 Bread & Cakes	€ 338,861	6,667	6,849	6,994	6,919	6,483	6,687	6,876

2021-2022 | 2020-2021 | 2019-2020 | 2018-2019 | 2017-2018 | 2016-2017 | 2015-2016 | Sheet1 | Sheet2 | Sheet3 | +

Figure 3: Sample data as received

The sales data included details like the Sales including VAT (Value-Added Tax), Sales excluding VAT, net Sales, Profit, Weekly Scanning Margin, Participation of various departments, Customer Count, Average Customer Spending amount, Employee wages details and weekly wastage information. Data was extracted from only what details were required for the analysis. Here the analysis is based on sales forecasting, only the Sales including VAT details are used. The other details like the Sales excluding VAT, Weekly Scanning Margin, Employee wages and Participation were removed as part of the data extraction process.

3.1.1: Software

Most statisticians prefer using R for performing these time series-based analyses. But here in this study, the complete analysis is performed in the Python 3 environment. Python was preferred over R for easing the analysis in the same environment. Python has a wide range of libraries that can support both statistical methods and machine learning methods. Also, previous knowledge of working with Python was an added advantage. Jupyter notebooks and Google Collaboratory were the tools used for performing the analysis.

3.1.2: Pre-processing: Cleaning the data

The sales data gathered was then imported into the Python working environment as a data frame. A data frame is described as a data structure of 2-dimension often containing rows and columns with data. Once, loaded the data is first transformed into a time series format for further analysis. Check for any inconsistencies and null values are followed. There were no inconsistent or missing data found. After this, the *DATE* is found to be an *object* categorical type which was converted into the 'date' format using the pandas – *pd.to_datetime( )* function in Python.

```
In [57]: pd.to_datetime(Centra_sales['Date'])

Out[57]: 0     2015-03-10
         1     2015-10-10
         2     2015-10-17
         3     2015-10-24
         4     2015-10-31
                  ...
         341   2022-04-16
         342   2022-04-23
         343   2022-04-30
         344   2022-07-05
         345   2022-05-14
         Name: Date, Length: 346, dtype: datetime64[ns]
```

Figure 4: Code snap - conversion of DATE into proper datetime64 format

This converts the date into the standard *YYYY-MM-DD* format. The columns were renamed using lower-case letters to make the names consistent. Proceeding further to ease the analysis the weekly periodic data was converted to monthly periodicity using internal loops. Now the observed resultant data is the monthly overall sales of the store in euros.

3.2 Exploratory Data Analysis (EDA)

The process of EDA is useful for analysing the data set using various visualization methods. The main characteristics of the dataset are summarised while performing EDA. This EDA process also helps in identifying the outliers or some hidden relations between the variables. Exploratory data analysis can be performed using various tools.

Below are the images from the EDA process for this dataset.

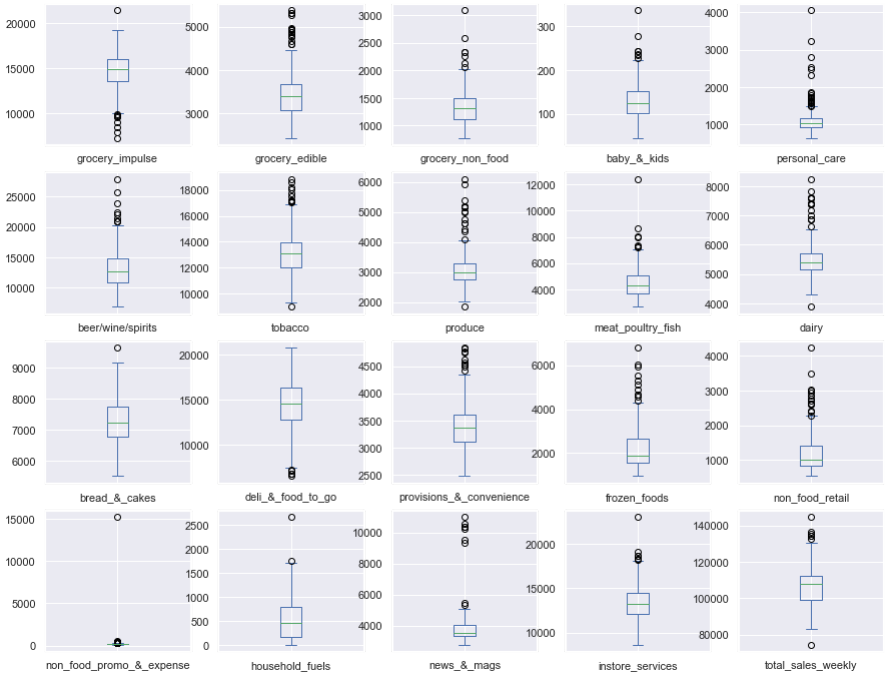


Figure 5: Boxplots of categorical sales

From the boxplots, it can observe that Impulse – Grocery contributes a major part to the overall sales of the store.

The correlation coefficient matrix indicates the correlation coefficient values for different variables. The correlation coefficient indicates the strength of a relationship. Equation 1 helps in calculating the correlation coefficient between two variables.

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[\, n\Sigma x^2 - (\Sigma x)^2 \,][\, n\Sigma y^2 - (\Sigma y)^2 \,]}}$$

Equation 1: Correlation Coefficient

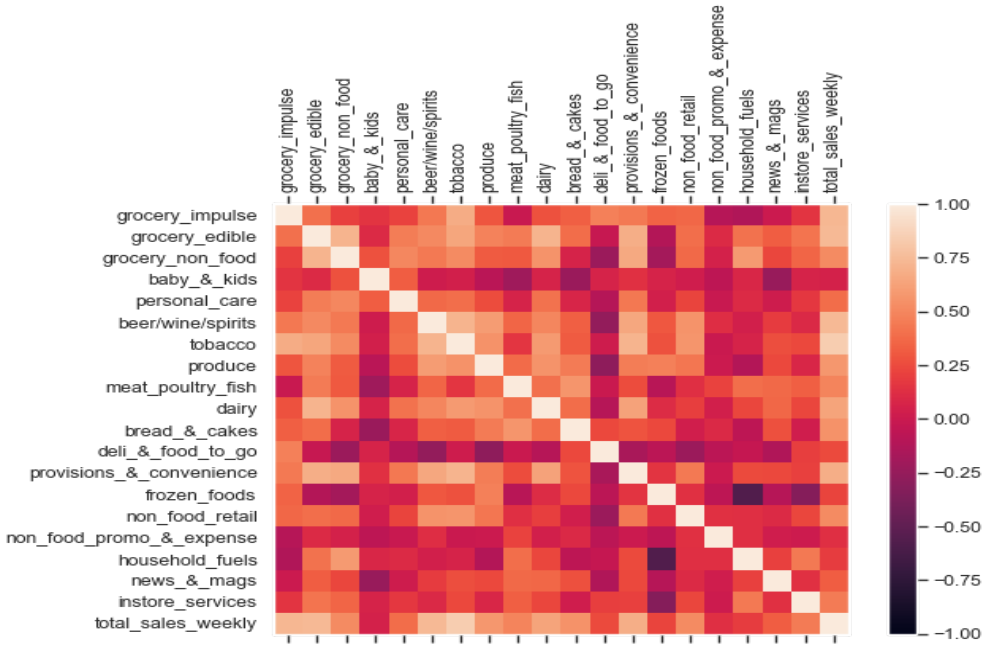


Figure 6: Correlation coefficient matrix of multivariate sales data

Figure 6 indicates the correlation coefficient matrix of the dataset. This plot is considered to be very powerful as it helps in summarizing large datasets with multiple variables. The darker the box is, the higher the correlation coefficient values are. From the above-obtained graph, it can be observed that the values *'frozen_food' and 'household_fuels'* have the highest coefficients.

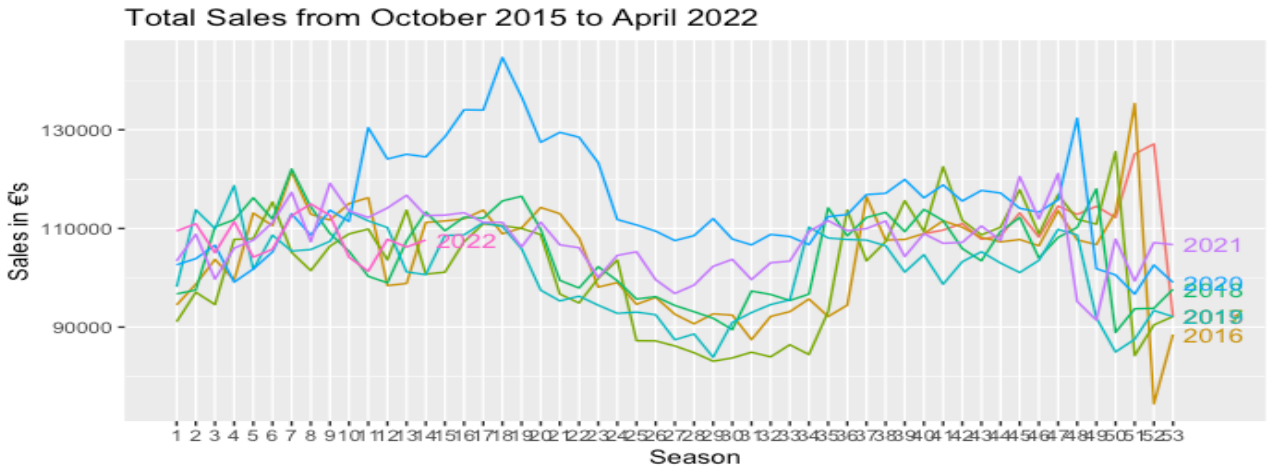


Figure 7: Year-wise sales represented in a line graph

From figure 7, it is observed that the sales were highest in the year 2020 in the middle of the pandemic. Similarly, it can be inferred that the average sales were high in 2021 when compared to the overall sales.

3.3 Understanding a time-series

Followed by data transformation, and visualisation the time series data is plotted against its time period. To better understand the data, the seasonal decomposition of the original time series data is plotted. The *seasonal_decompose( )* function is used for separating the trend, seasonal and cyclical variations, and residual components. A time series is usually represented by equation 2.

$$y_t = f(S_t, T_t, R_t)$$

Equation 2: General equation of a time series

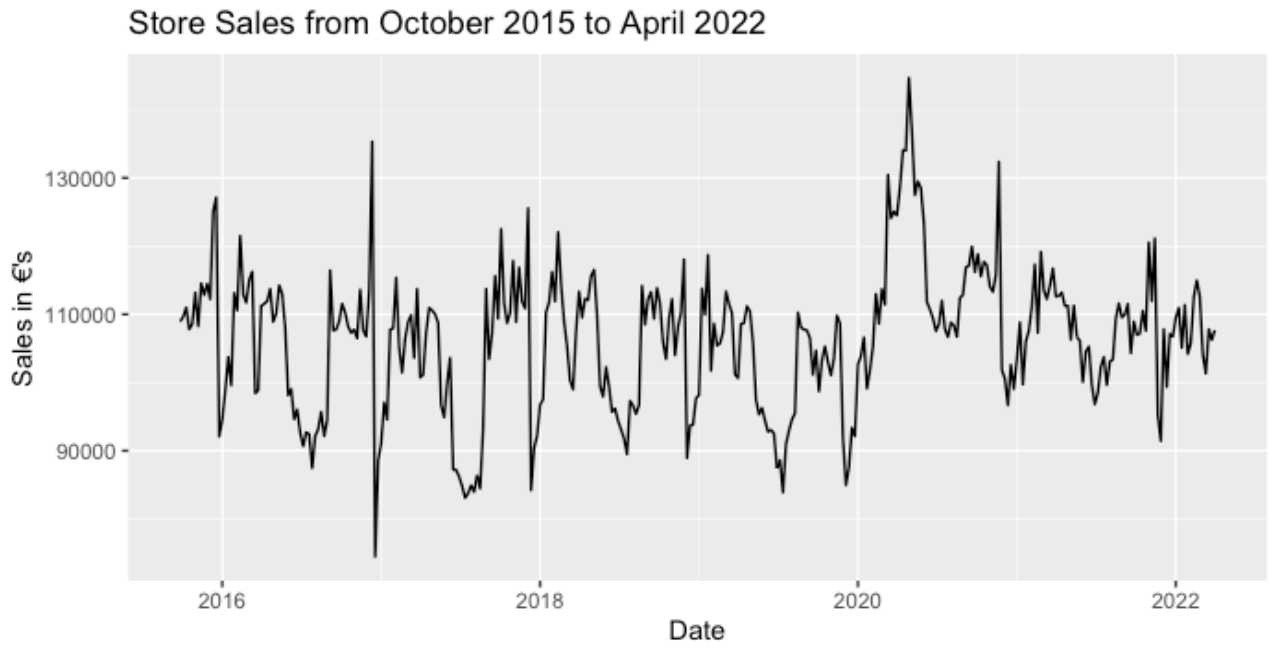


Figure 8: Weekly sales data from October 2015 to April 2022

There are no null values. As shown above, the average weekly sales of the store from 2016 to 2021 were € 106,292. The maximum weekly sales during this period were € 144,661 and the minimum weekly sales were € 74,465.

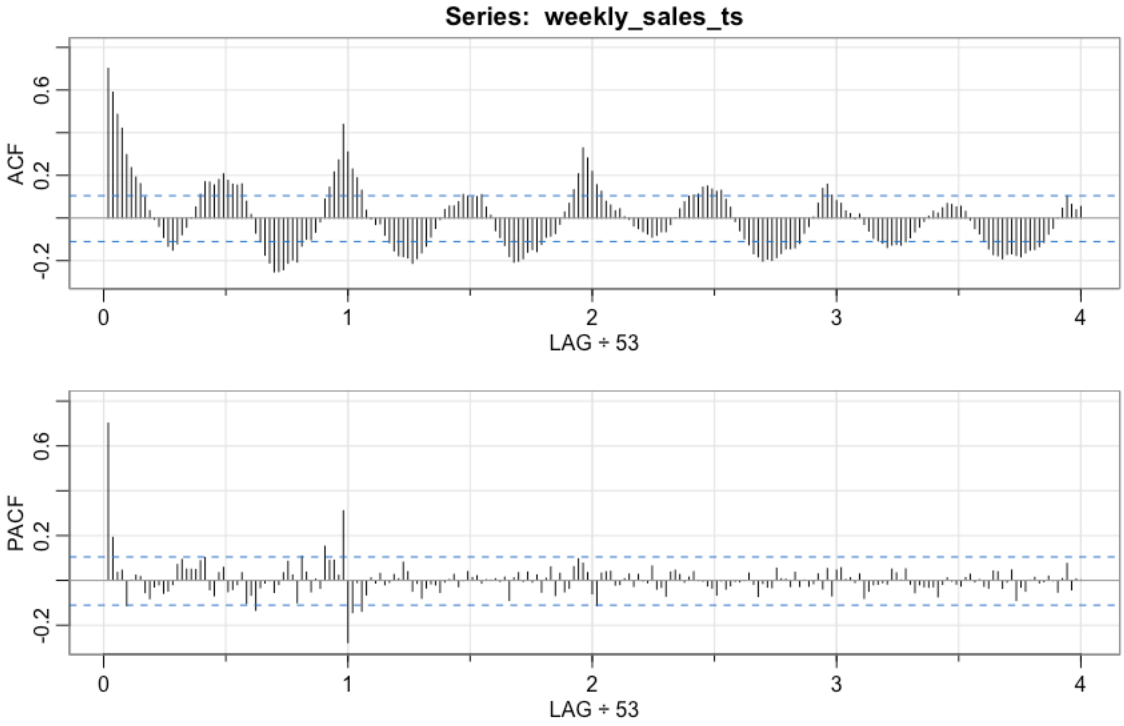


Figure 9: Autocorrelation plots for weekly sales

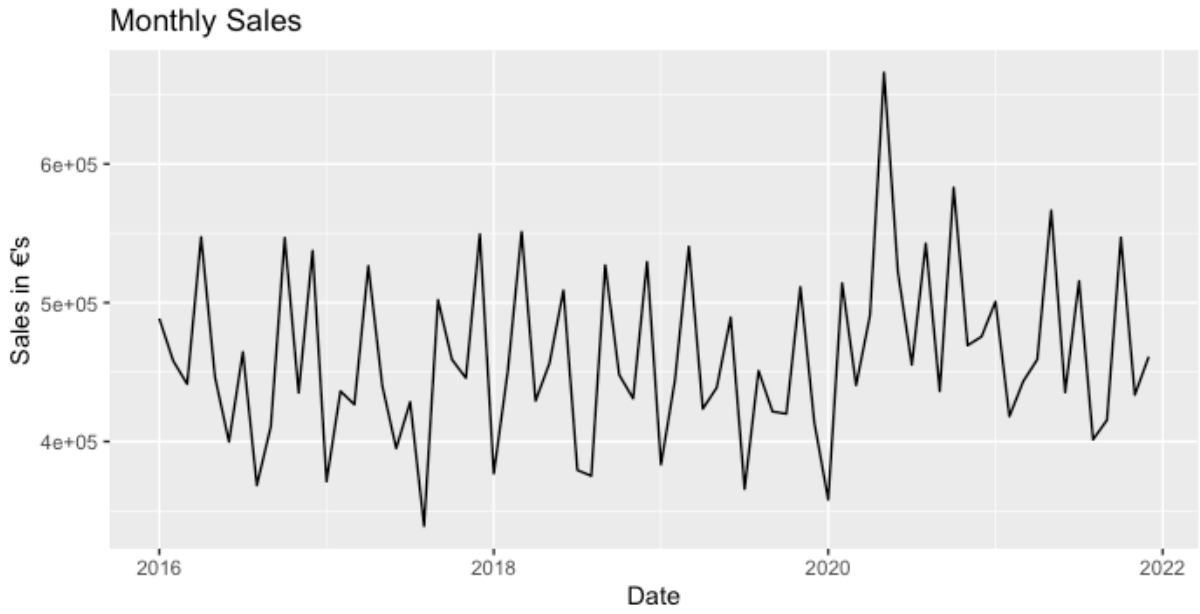


Figure 10: Monthly sales data from October 2015 to April 2022

From the above monthly plot of the original data, it can be observed that there is a repeating pattern and a cyclical variation. Monthly sales of the store were € 460,857 whereas during peaks it even reached € 665,802.

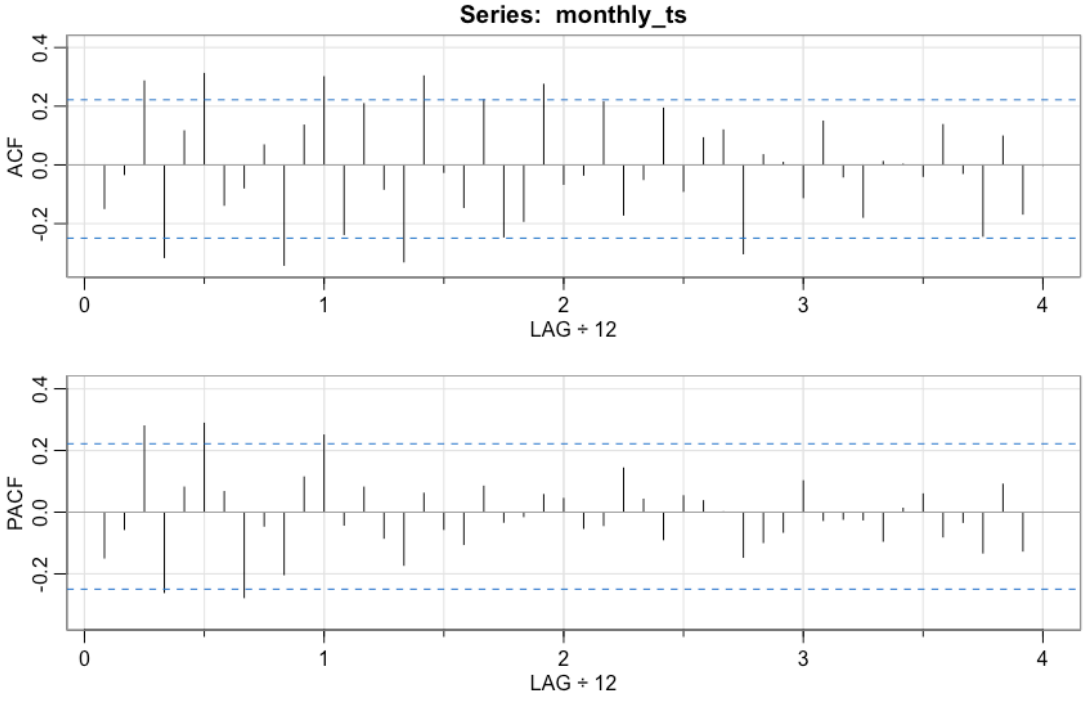


Figure 11: Autocorrelation plots for monthly sales

Interpreting the above seasonal decomposition, the trend is considered over a long period of time and can either be an upward/downward trend or constant also. Here, from the above *figure 11,* it is observed that there is an increase in the trend, especially during the pandemic. Seasonal variation refers to the trend which occurs within a year. These seasonal changes are likely to repeat over the subsequent years.

In the observed decomposition also there exists a seasonality every year. There are peak sales during the months of April and December and low sales during the summer months. Usually, a wave-like movement is observed. This is caused due to externally affecting features like the market demand, introduction of new products and offers, festivals, etc,

Apart from these components, there always exists some extreme values in the data which are called outliers or residuals. These residuals build up the residual variation component of the time series which is used in ARIMA modelling.

3.4 Decomposition of the time series

Time series exhibits a variety of patterns: trend, seasonality and cyclical variation. When a time series is decomposed into its components it improves the forecast accuracy. There are two forms of classical decomposition: additive decomposition and multiplicative decomposition.

For an additive decomposition, this is done by subtracting the trend estimates from the time series. For a multiplicative decomposition, this is done by dividing the series by the trend values. The additive model is useful when the seasonal variation is relatively constant over time. The multiplicative model is useful when the seasonal variation increases over time.

Additive Decomposition of Weekly Sales:

In *additive* time series, the individual components are added together. Whereas, in *multiplicative* time series the individual components are multiplied together. The formula for each of the models is given as follows.

$$y_t = T_t + S_t + R_t$$

Equation 3: Additive time series formula

where,

y_t *--- the data in time period t*

T_t *--- trend component and cycle component at time t*

S_t *--- Seasonal component at time t*

R_t *--- Residual component at time t*

The plot below shows the observed series, the trend line, the seasonal pattern and the random part of the time series. Here the seasonal pattern is the regularly repeating pattern.

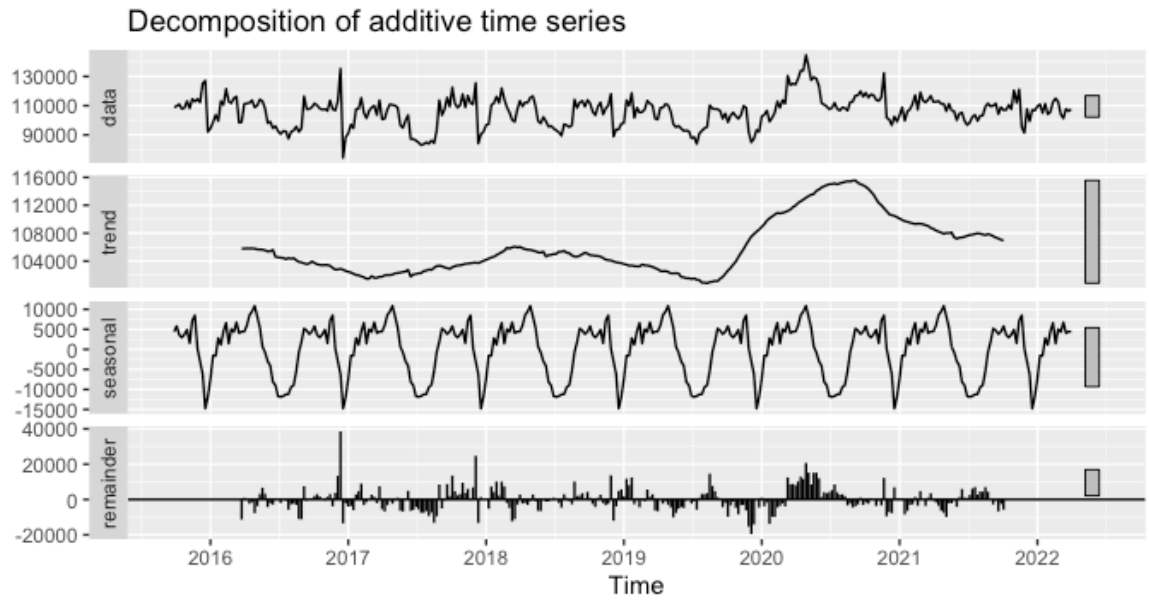


Figure 12: Additive decomposition of weekly sales

The first row is the actual time series, followed by its decomposition. The trend in the second line has a cyclical movement. Having ups and downs over a long time period. In 2020, a dramatic increase is due to the pandemic. The squares at the right end of the lots describe the data's scale. The smaller the box is the higher the scale of the data. If the square boxes are of similar size it means they both have the same impact on the variability of the data. The residuals from the remainder part are the ones which lead to randomness. The smaller the box is, the least is the randomness. When the seasonality is removed, there is a repeating pattern observed every 12 months.

Additive Decomposition of Monthly Sales:

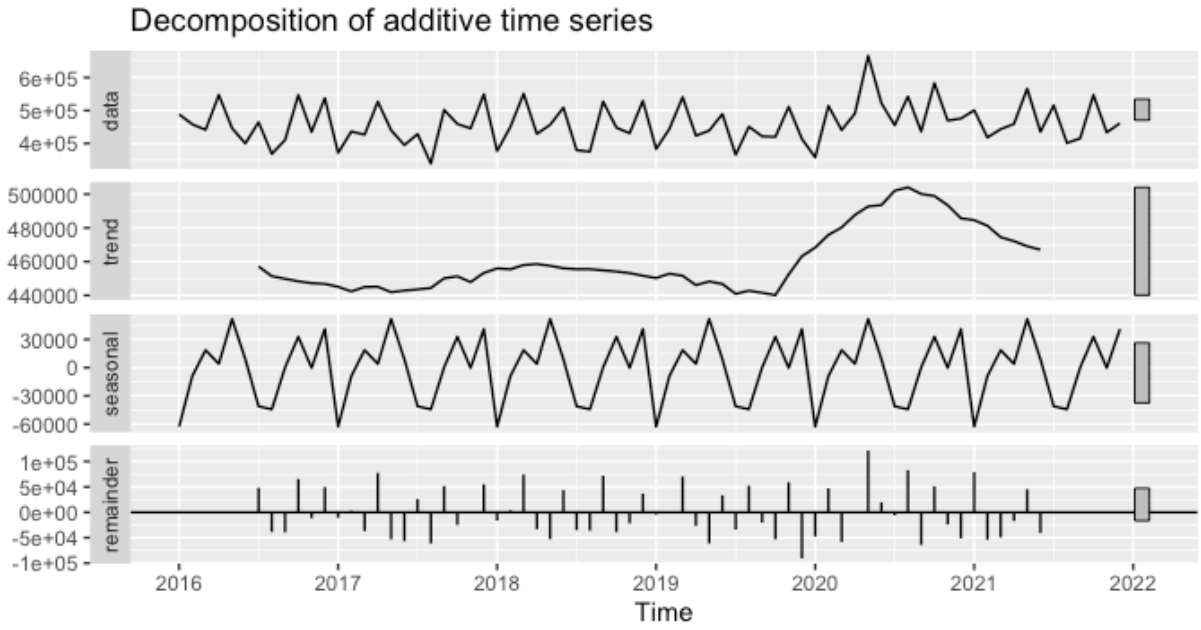


Figure 13: Additive decomposition of monthly sales

A similar pattern is observed for the monthly sales data also.

Multiplicative Decomposition of Weekly Sales:

The multiplicative decomposition argues that time series data is a function of the product of its components. It can be usually identified from its variation. If the magnitude of the seasonal component changes with time, then the time series is multiplicative. Here the magnitude of the seasonal component grows as time increases.

$$y_t = T_t \times S_t \times R_t$$

Equation 4: Multiplicative time series formula

where,

y_t *--- the data in time period t*

T_t *--- trend component and cycle component at time t*

S_t *--- Seasonal component at time t*

R_t *--- Residual component at time t*

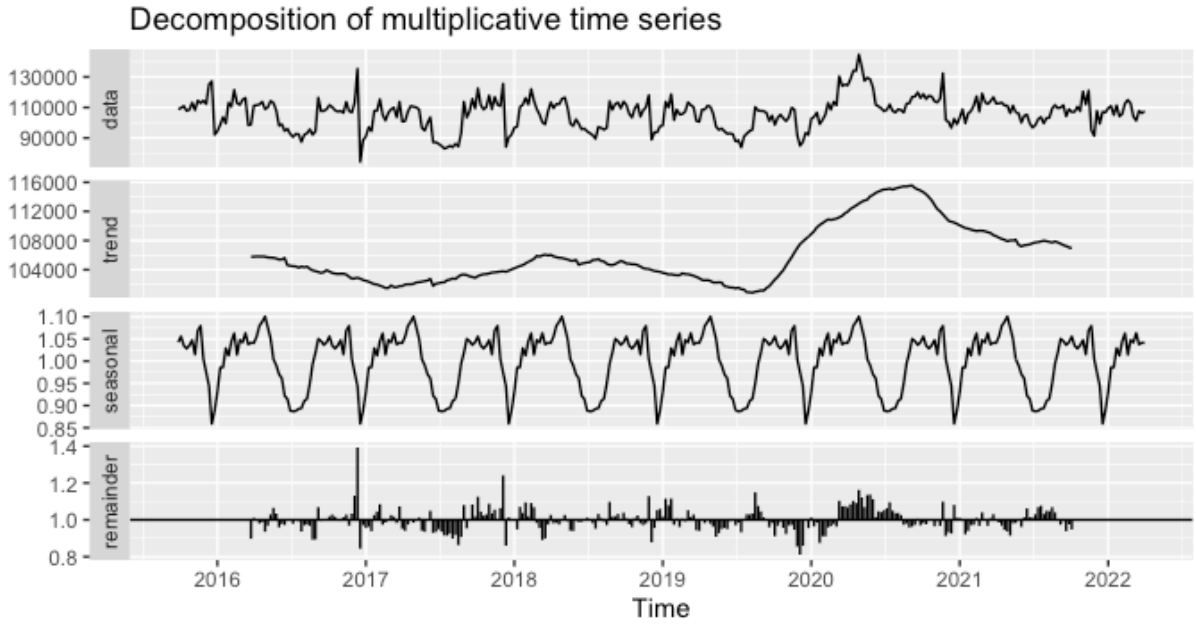


Figure 14: Multiplicative decomposition of weekly sales

Multiplicative Decomposition of Monthly Sales:

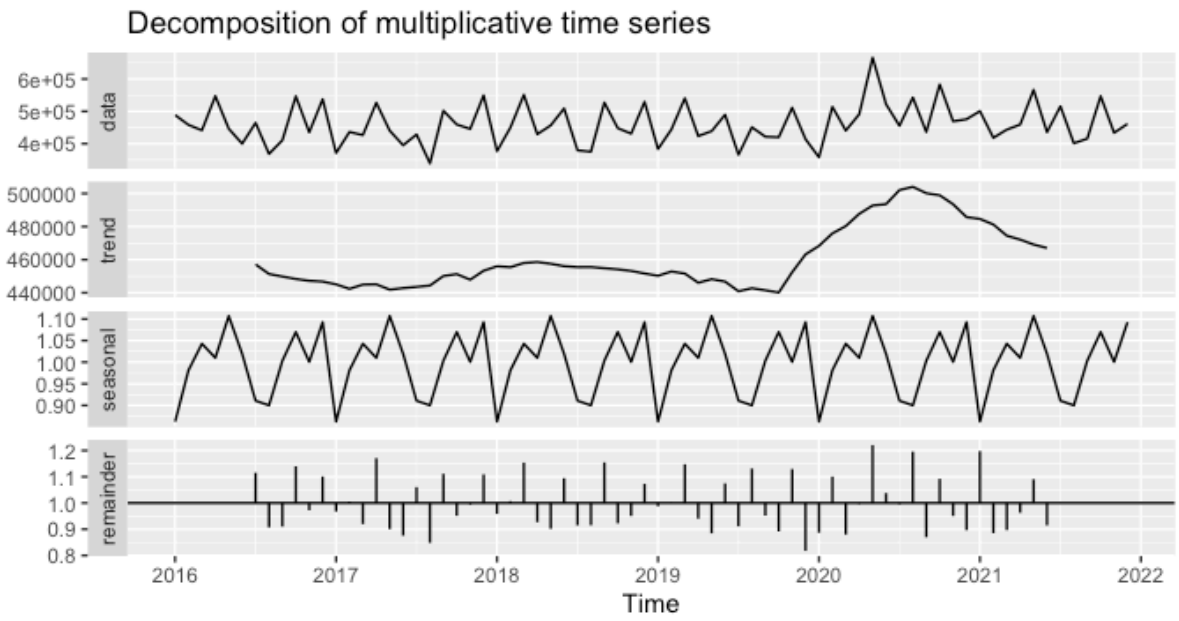


Figure 15: Multiplicative decomposition of monthly sales

From the above graph, it can be noticed long periods of apparent trend ups and downs are present. The time series is non-stationary with respect to mean. Taking 1st difference may result in a stationary time series. Here it can also be observed that the variance of the time series is not constant. To make it constant, logarithms are used to remove or stabilise the variance of the time series. The seasonal difference is used to remove the peaks and the ordinary difference to remove the trend.

3.5 Statistical Methods

Under the traditional statistical methods of forecasting, some of the classical methods like HoltWinter, ARIMA, and SARIMA can be used for forecasting sales of historical data. In this analysis, Holt-Winters' and ARIMA models are used to analyse and forecast sales.

3.5.1: Time Series Analysis using Holt-Winters' exponential smoothing:

Holt-Winters' exponential smoothing is one of the oldest methods of time series forecasting techniques which considers trend and seasonality while forecasting. This method has three major aspects for performing the predictions.

i. **Exponential Smoothing:** Simple exponential smoothing – Forecasting when the data contains no trend or seasonality.
ii. **Holt's Smoothing:** Holt's exponential smoothing – Forecasting when the data has a trend but no seasonality.
iii. **Holt-Winters' Smoothing:** Holt-Winters' exponential smoothing – Forecasting technique which includes seasonality along with the trend.

Holt Winter's classical method is best suited for the Sales dataset Time series because the time series contains both trend and seasonality. This method forecasts future values by extending the simple exponential smoothing to capture the trend and seasonality of the time series using triple exponential smoothing.

3.5.2: Time Series Analysis using ARIMA: Auto-Regressive Integrated Moving Average

In the ARIMA model, a linear combination of historical values and errors is used for predicting future values. The ARIMA is a combination of the AR part, Integration and the MA part. An ARIMA(*p, d, q*)*(P, D, Q)* model where *p* is the number of non-seasonal autoregressive terms (AR) and *q* is the number of non-seasonal moving average terms (MA) and *d* is the degree of non-seasonal differencing and *P, D, Q* refers to the same for a seasonal component respectively.

Auto-Regression

The AR (*p*) model is given as follows:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

Equation 5: Equation for the Auto-regressive part of the ARIMA model

Integration

The I (*d*) is the number of differences taken to make the time-series stationary.

Moving Average

The MA (*q*) model is given as follows:

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

Equation 6: Equation for the Moving-average part of the ARIMA model

Combining all of the three types of models above gives the resulting ARIMA(p,d,q) model.

ARIMA

$$\hat{y}'_t = c + \phi_1 y'_{t-1} + \phi_2 y'_{t-2} + \ldots + \phi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q} + \epsilon_t$$

Equation 7: General equation of the ARIMA model

where,

- *εt is defined as white noise or randomness.*
- *ϕp depends on the order p selected*

- θq *depends on the order q selected*
- $c = (1 - \sum^{p} \phi) \cdot \mu$

Stationarity Tests:

Stationarity describes how a particular time series variable changes over a time period. Transformations are made on the original data to obtain stationarity. A time series is considered stationary when the below conditions are satisfied:

- Constant mean.
- Constant variance (Homoscedasticity).
- The autocorrelation pattern is constant throughout the time series.

ARIMA is based on the assumption that the data should be stationary and univariate. A time series is said to be stationary if its mean, variance, and covariance are all invariant with respect to time. The weekly sales data does not show a constant variance and constant mean, in addition, the data shows some repeating patterns over time, making it different from white noise. Monthly may be considered a non-stationary process.

However, transformations can be always done to remove the trend and seasonality on the original time series data. These transformations include differentiating the time series to remove the trend and taking log transformations to stabilize the variance in the data.

Removing Trend and Seasonality:

Time series datasets may contain trends and seasonality, which may need to be removed prior to further modelling. Trends can result in a varying mean over time, and seasonality can result in a changing variance over time, both of which define a time series as being non-stationary.

Stationary datasets are those that have a stable mean and stable variance. Differencing and log transformations on widely used data can make a time series stationary.

i) Stationary Time Series:

The observations in stationary time series data are not dependent on time. Time series is stationary if they do not have a trend or seasonal effects. Summary statistics calculated on the time series are consistent over time, like the mean or the variance of the observations. When a time series is stationary, it can be easier to model. Statistical modelling methods assume or require the time series to be stationary.

ii) Non-Stationary Time Series:

Observations from a non-stationary time series show seasonal effects, trends, and other structures that depend on the time index. Summary statistics like the mean and variance do change over time, providing a drift in the concepts a model may try to capture. Classical time series analysis and forecasting methods are concerned with making non-stationary time series data stationary by identifying and removing trends and removing stationary effects.

Differentiation and log transformations are not the only determinants of stationarity. There are formal statistical tests such as the Augmented Dickey-Fuller test and the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) tests. These tests, check for the stationarity of a series around a deterministic trend.

Part 1 – Monthly Sales data

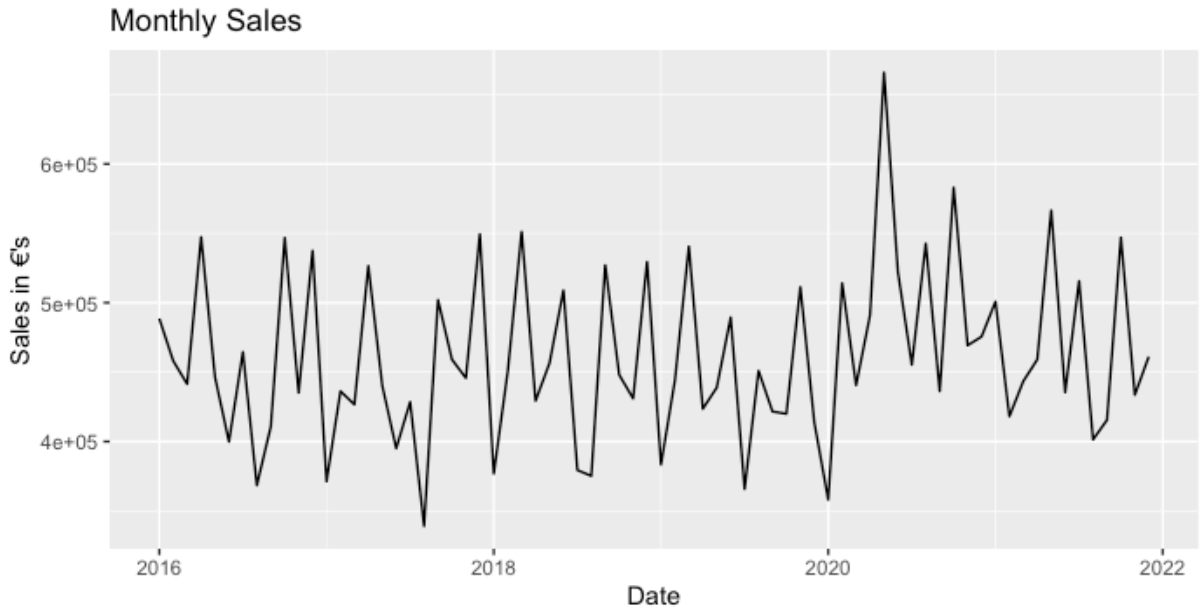


Figure 16: Monthly sales data from October 2015 to April 2022

Tests for Stationarity:

a) Dickey-Fuller Test:

To check the stationarity of the time series Augmented Dickey-Fuller test is performed. This can be imported from the *statsmodels library* using the *adfuller*( ) function.

```
> adf.test(monthly_ts, alternative = "stationary",k = 0)

        Augmented Dickey-Fuller Test

data:  monthly_ts
Dickey-Fuller = -9.9116, Lag order = 0, p-value =
0.01
alternative hypothesis: stationary

Warning message:
In adf.test(monthly_ts, alternative = "stationary", k = 0) :
  p-value smaller than printed p-value
```

Figure 17: Code snap showing the ADF test for monthly time series

The null hypothesis H_o states that the time series is not stationary.
The alternative hypothesis H_a states that the time series is stationary.

Since the p-value (0.01) is less than the significance level (0.05), we have strong evidence to reject the null hypothesis and conclude that the time series is stationary. Figure 17 shows the results from the test for stationarity.

b) KPSS Test:

The Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test figures out if a time series is stationary around a mean or linear trend, or is non-stationary due to a unit root. A stationary time series is one where statistical properties like the mean and variance are constant over time.

The null hypothesis H_o states that the time series is stationary with a trend.
The alternative hypothesis H_a states that the time series is not stationary.

```
> kpss.test(monthly_ts)

        KPSS Test for Level Stationarity

data:  monthly_ts
KPSS Level = 0.28017, Truncation lag parameter = 3, p-value = 0.1

Warning message:
In kpss.test(monthly_ts) : p-value greater than printed p-value
```

Figure 18: Code snap showing KPSS test for monthly time-series

Since the p-value (0.01) is less than the significance level (0.05), we have strong evidence to reject the null hypothesis and conclude that the time series is not stationary. This contradicts the results obtained from Dicky Fuller test. Hence the time series needs to be transformed to make it stationary.

Manipulate the Time Series to make it stationary:

Stationarity of the time series can be observed by looking at a line plot of the series over time. Signs of trends, seasonality or other random components in the series are indicators of a non-stationary series. A more accurate method would be a statistical test, such as the KPSS test or the Dickey-Fuller test. Differencing is one of the methods of transforming time series data. It can remove the time dependence on the data and helps stabilize the mean of the time series by removing changes in the level of a time series, and eliminating trend and seasonality. Differencing is performed by subtracting the previous observation from the current observation. Another such method is log transformation. Here the difference between consecutive observations is taken. Log differences can be adjusted to suit the specific temporal structure. For a time series with a seasonal component, the log may be expected to be the period (width) of the seasonality. Violation of stationarity creates estimation problems for ARIMA models.

Below were the steps carried out to transform the time series data stationary to be fed into an ARIMA model.

Step 1: Remove the variability

Remove the variability of the time series by applying a log. A log transformation converts a multiplicate time series model to an additive time series model. The time series now displays constant variance as shown below.

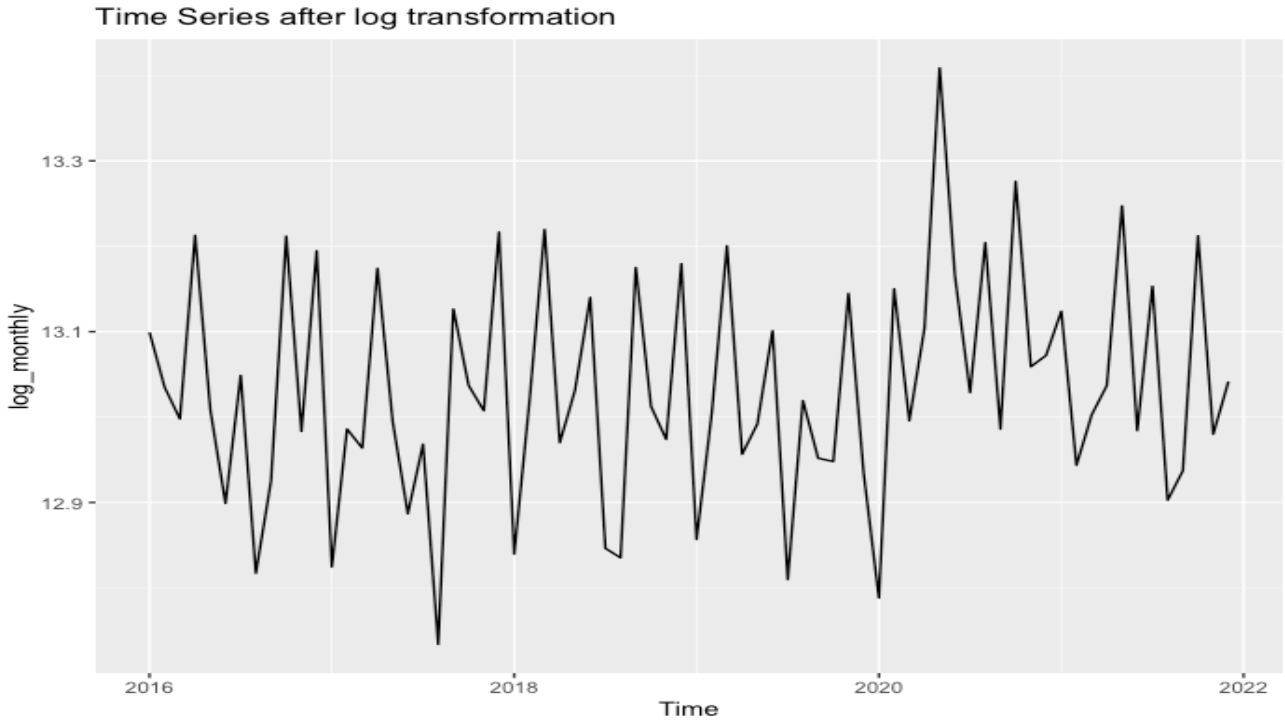


Figure 19: Time Series plot after log transformation

a) **Dickey-Fuller Test:**

```
> adf.test(log_monthly, alternative = "stationary",k = 0)

        Augmented Dickey-Fuller Test

data:  log_monthly
Dickey-Fuller = -9.9436, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(log_monthly, alternative = "stationary", k = 0) :
  p-value smaller than printed p-value
```

Figure 20: Code snap showing the ADF test after log transformation

a) **KPSS Test:**

```
> kpss.test(log_monthly)

        KPSS Test for Level Stationarity

data:  log_monthly
KPSS Level = 0.28658, Truncation lag parameter = 3, p-value = 0.1

Warning message:
In kpss.test(log_monthly) : p-value greater than printed p-value
```

Figure 21: Code snap showing the KPSS test for log-transformed data

The KPSS test now confirms that the time series is not stationary with a level or trend by giving a p-value (0.1) which is greater than the significance level (0.05). Hence we fail to reject the null hypothesis and conclude that the time series is stationary.

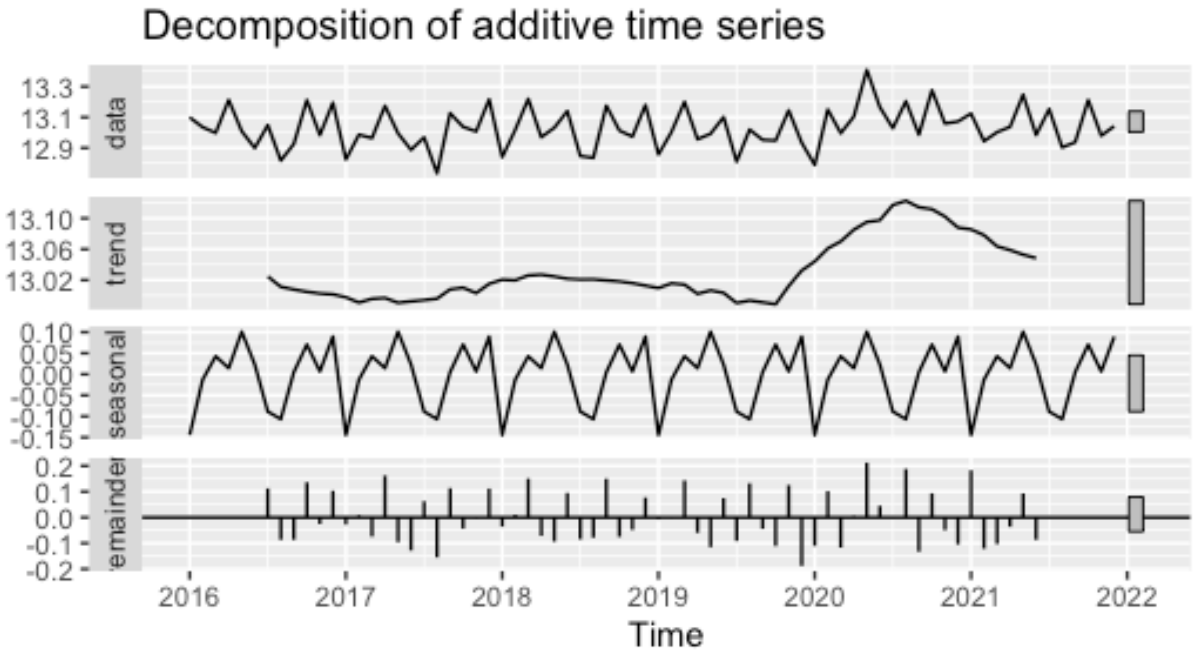


Figure 22: Additive decomposition of the log-transformed data

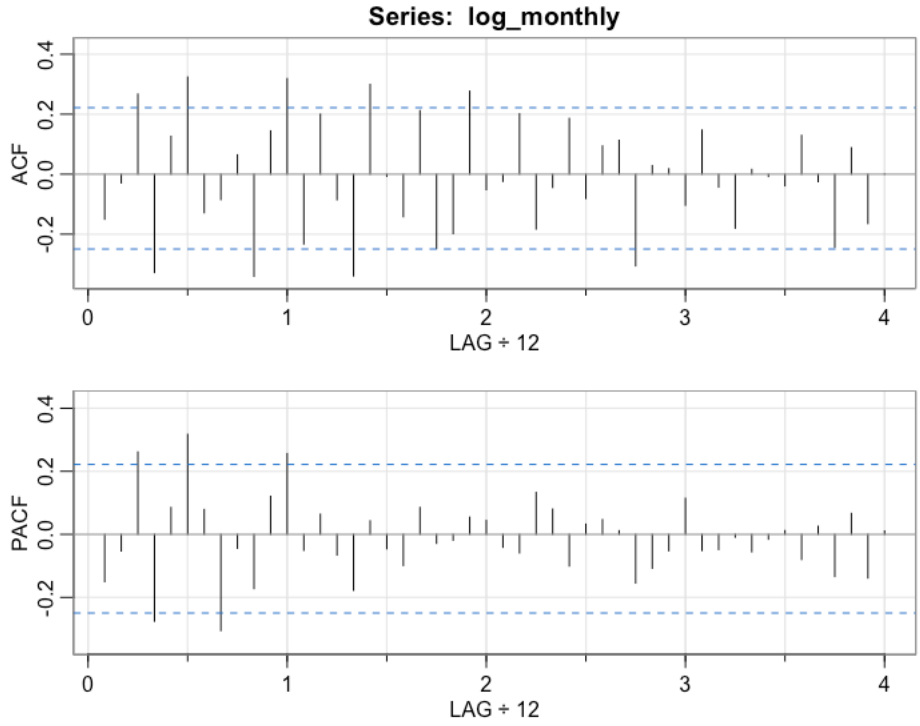


Figure 23: Autocorrelation plots for monthly sales after log transformation

Step 2: Seasonal Differencing

Remove the trend by differentiating the time series. Since we observe a linear trend, a 1st order differentiation is sufficient. If the trend is exponential, then a 2nd order differentiation is required. If the trend is cubic, then a 3rd order differentiation is required. As shown below, the trend has the least impact on the time series post-differentiation.

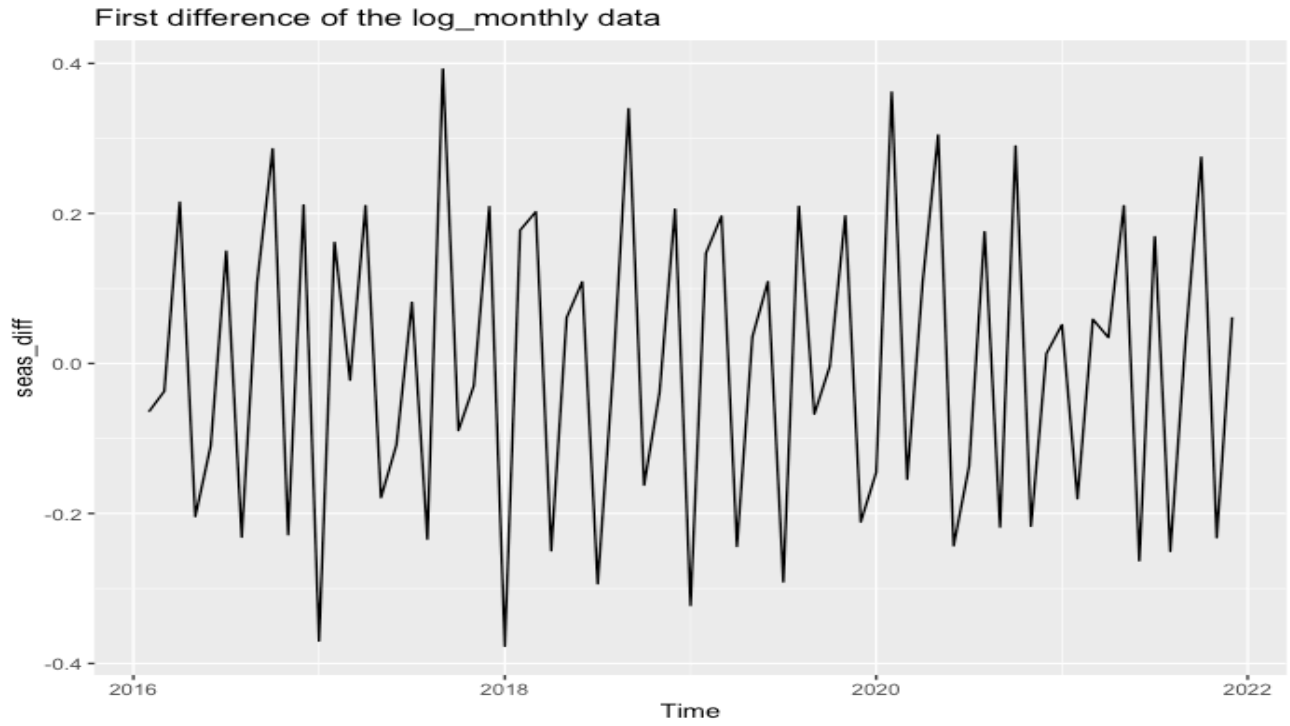


Figure 24: Time Series plot after differencing the log-transformed data

a) Dickey-Fuller Test:

```
> adf.test(seas_diff, alternative = "stationary",k = 0)

        Augmented Dickey-Fuller Test

data:  seas_diff
Dickey-Fuller = -15.248, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(seas_diff, alternative = "stationary", k = 0) :
  p-value smaller than printed p-value
```

Figure 25: Code snap of ADF test after the first difference on log_monthly data

b) KPSS Test:

```
> kpss.test(seas_diff)

        KPSS Test for Level Stationarity

data:  seas_diff
KPSS Level = 0.024286, Truncation lag parameter = 3, p-value = 0.1

Warning message:
In kpss.test(seas_diff) : p-value greater than printed p-value
```

Figure 26: Code snap of KPSS test after the first difference on log_monthly data

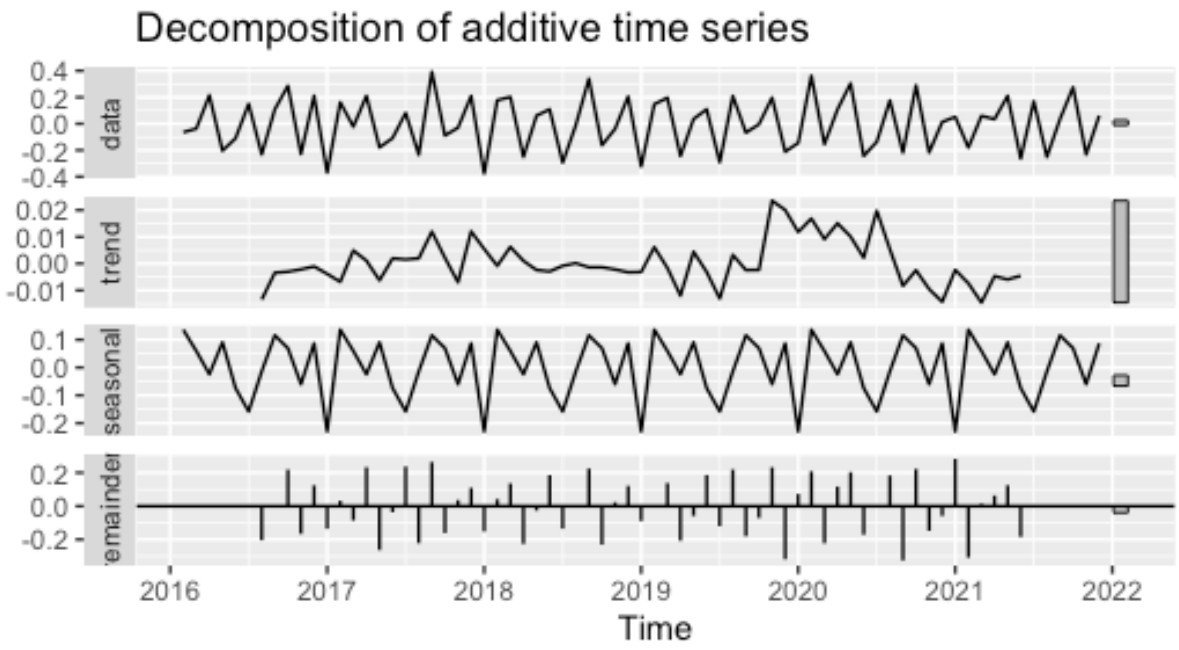


Figure 27: Additive decomposition of the differenced log-transformed data

The autocorrelation plot highlights that although most of the data are within the 95% limit, there is still a peak every 4 periods. This indicates periodicity in the time series and hence the seasonality must be removed.

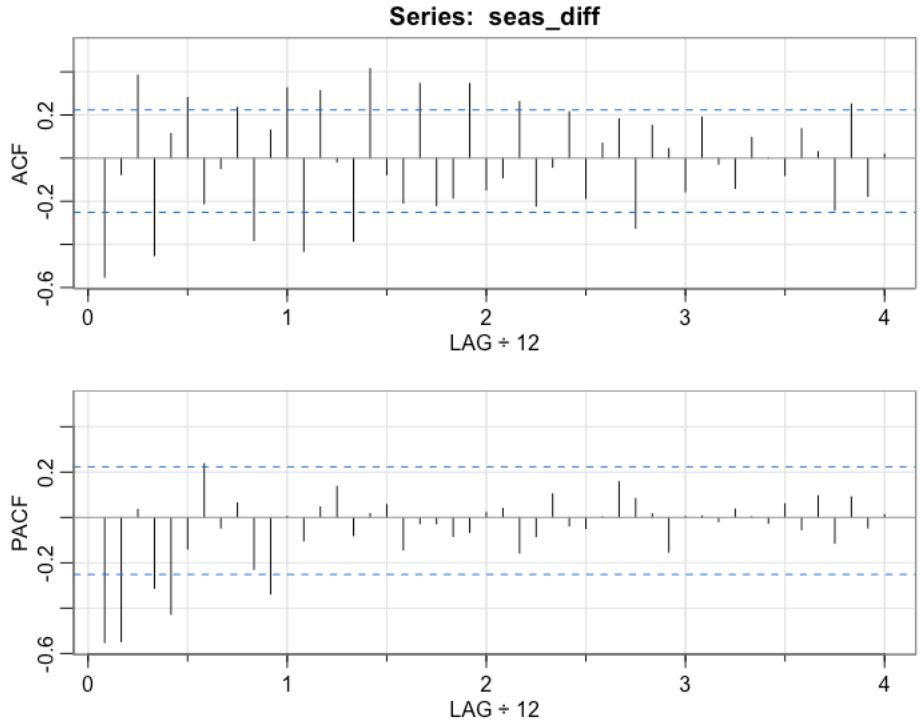


Figure 28: Autocorrelation plots after the first difference of log_monthly

Step 3: Remove the seasonality by applying differentiation. As shown below, the time series is now converted into white noise with mean 0 and constant variance.

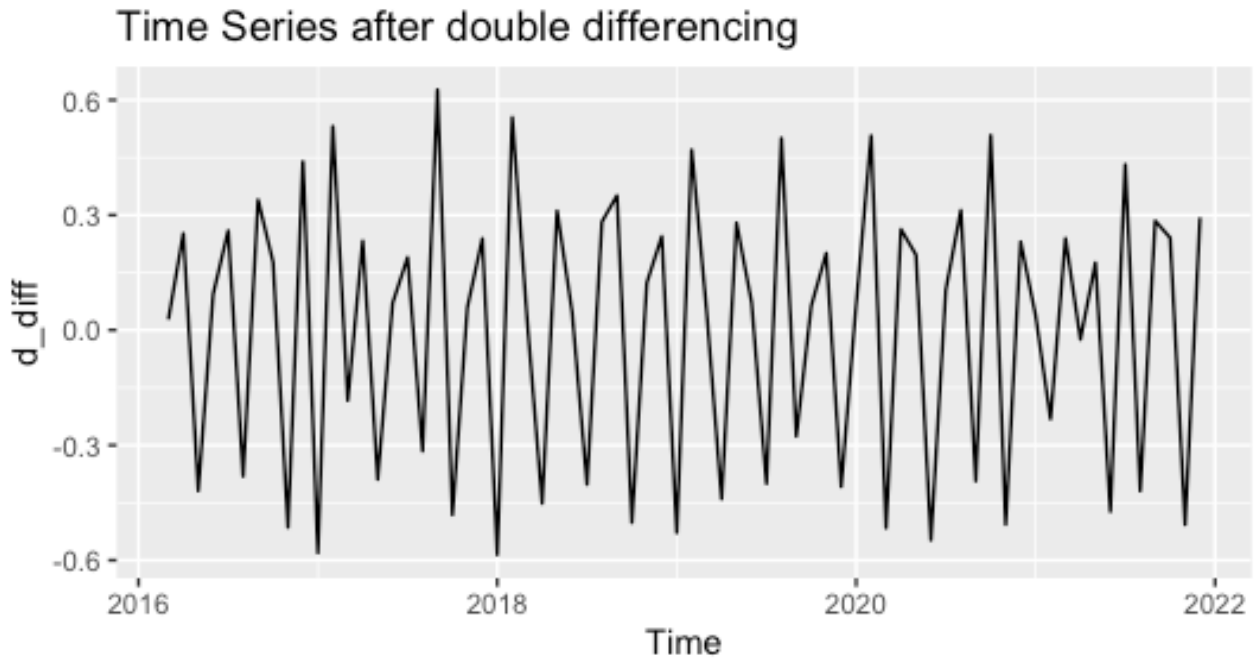


Figure 29: Time Series plot after double differencing the log-transformed data

a) Dickey-Fuller Test:

```
> adf.test(d_diff, alternative = "stationary",k = 0)

        Augmented Dickey-Fuller Test

data:  d_diff
Dickey-Fuller = -17.736, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(d_diff, alternative = "stationary", k = 0) :
  p-value smaller than printed p-value
```

Figure 30: Code snap of ADF test after the second difference on log_monthly

b) KPSS Test:

```
> kpss.test(d_diff)

        KPSS Test for Level Stationarity

data:  d_diff
KPSS Level = 0.02027, Truncation lag parameter = 3, p-value = 0.1

Warning message:
In kpss.test(d_diff) : p-value greater than printed p-value
```

Figure 31: Code snap of KPSS test after the second difference on log_monthly

The below decomposition plot highlights that trend and seasonality have been removed from the time series and the data is ready for feeding into an ARIMA model.

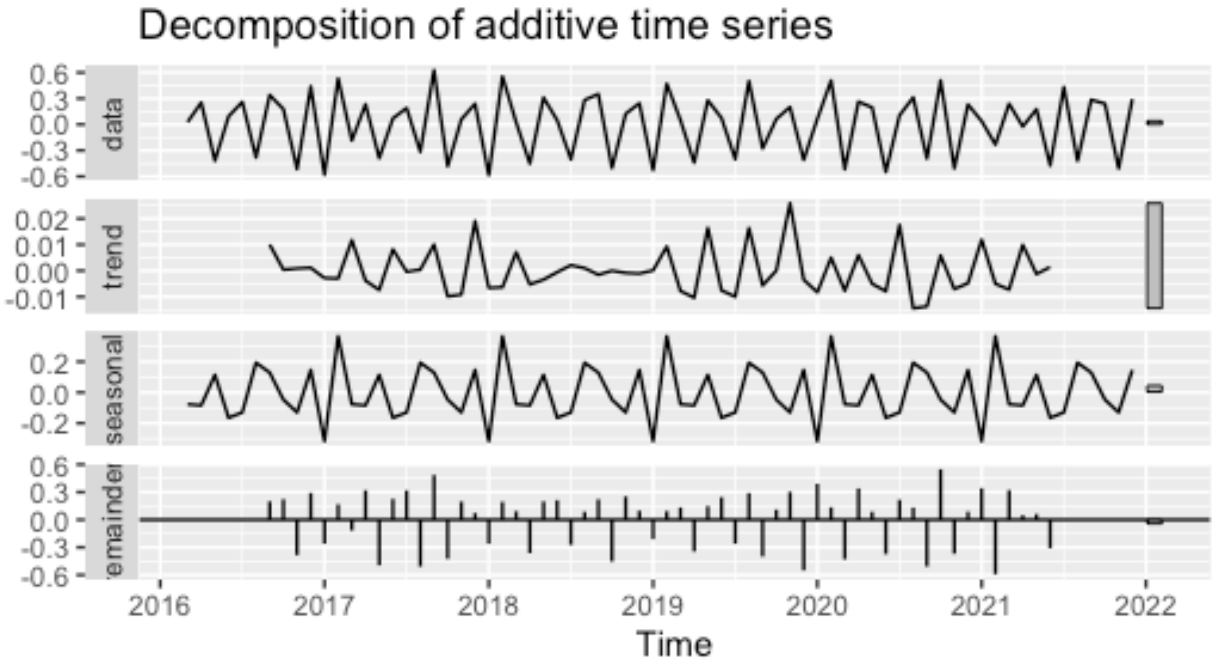


Figure 32: Additive decomposition of the double differenced log-transformed data

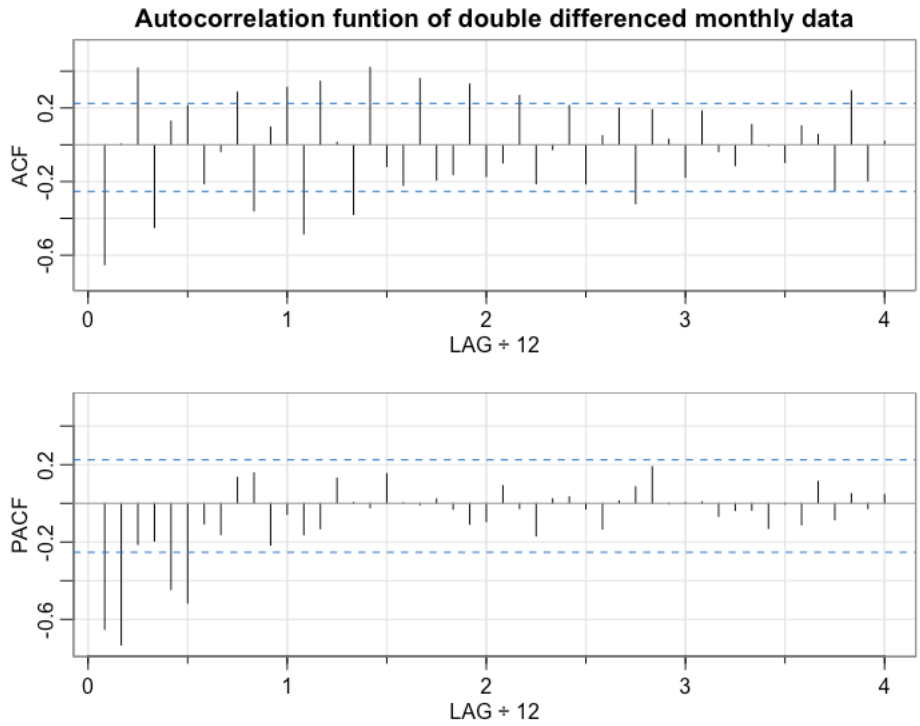


Figure 33: Autocorrelation plots after the second difference of log_monthly

The autocorrelation plot highlights that there is no periodicity but still there are few bars outside of the 95% limit.

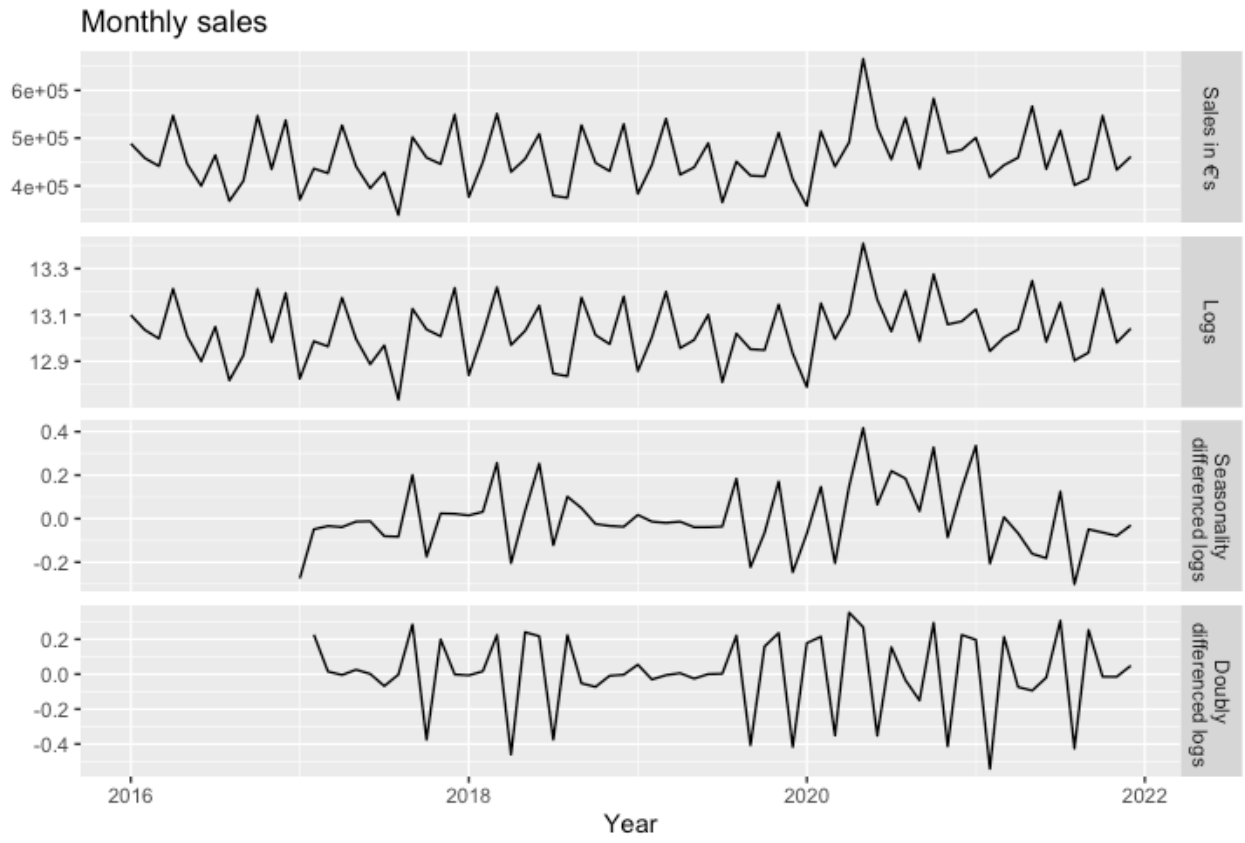


Figure 34: Comparison plot of the monthly sales

Figure 34 represents the different transformations carried to the monthly time series data. These transformations consist of a log transformation and a double differentiation to stabilize the variance in the original sales data. It can be observed that the trend is removed from all the data. Combining the log-transformed and double differenced monthly data, it became like white noise as seen in the last row of figure 34.

After getting a stationary process, the next in ARIMA modelling is the identification of the p, d, and q parameters of the model. This process requires the use of Autocorrelation and partial autocorrelation function. The ACF and PACF measure the level of dependence between observations in a time series in a set of lags. ACF helps identify which lags have significant correlations, it also helps understand the patterns of the time series which are of great help for time series modelling as ACF helps access the randomness and stationarity of a time series as well as the presence of trend and seasonal patterns.

The SARIMA model has been chosen to process this time series since it has a seasonal component. Many models were derived from this figure and their training scores are summarized in the next table.

	AIC	AICc	BIC
ARIMA(0,1,1,0,0,0,12)	-77.59	-77.41	-73.06
ARIMA(1,0,0,0,0,0,12)	-84.01	-83.41	-74.9
ARIMA(1,1,1,0,0,0,12)	-77.14	-76.54	-68.09
ARIMA(1,1,0,0,0,0,12)	-46.18	-45.82	-39.39
Auto.arima - ARIMA(0,0,0,0,0,1,12)	-91.97	-91.61	-85.14

Table 1: ARIMA models for monthly data (frequency = 12)

The models are selected if their AIC, and RMSE represent the minimum, and MAE is used to decide if the two models have similar RMSE and AIC. Table 1 shows the best ARIMA models for monthly periodic data. ARIMA(0,0,0,0,0,1,12) having the least AIC value of -91.97 outperforms the rest of the models. The auto.arima model was found to be the best. The Ljung Box-test was performed on the mode. Followed by this, future predictions were made.

A similar analysis of the weekly dataset was carried out as follows.

Part 2 – Weekly Sales data

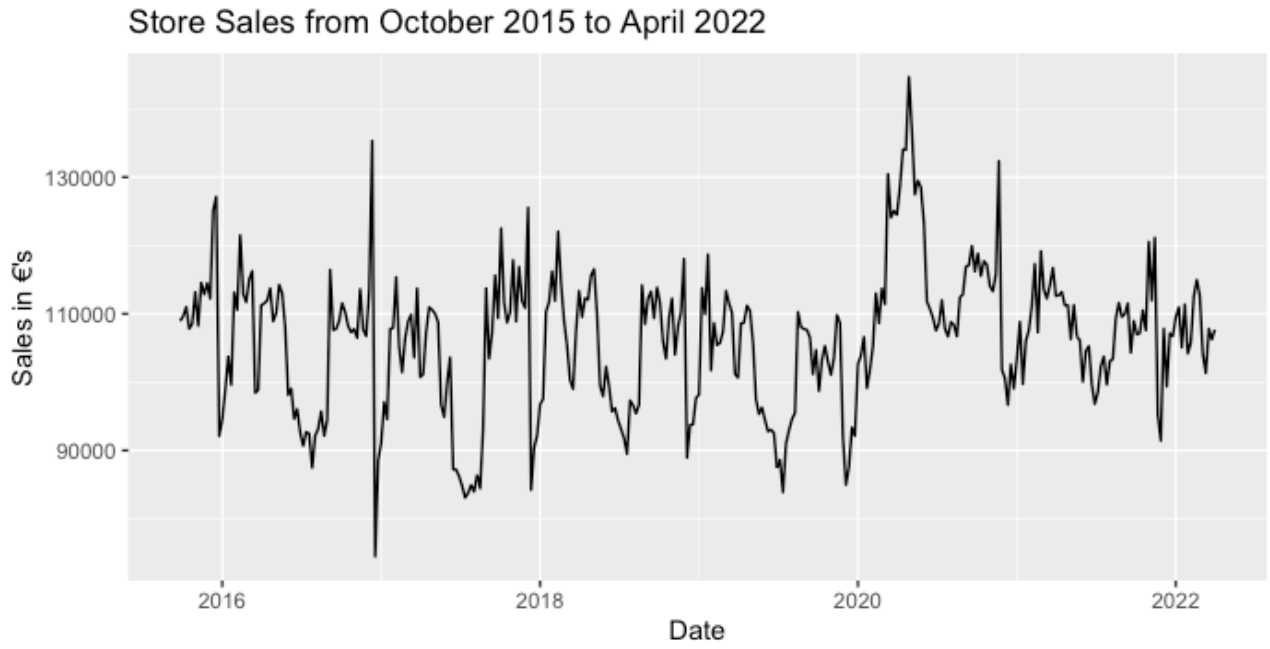


Figure 35: Weekly sales data from October 2015 to April 2022

Tests for Stationarity:

a) Dickey-Fuller Test:

```
> adf.test(weekly_sales_ts, alternative = "stationary",k = 0)

        Augmented Dickey-Fuller Test

data:  weekly_sales_ts
Dickey-Fuller = -7.8701, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(weekly_sales_ts, alternative = "stationary", k = 0) :
  p-value smaller than printed p-value
```

Figure 36: Code snap showing the ADF test for weekly time-series

b) KPSS Test:

```
> kpss.test(weekly_sales_ts)

        KPSS Test for Level Stationarity

data:  weekly_sales_ts
KPSS Level = 0.40716, Truncation lag parameter = 5, p-value = 0.07407
```

Figure 37: Code snap showing KPSS test for weekly time-series

Step 1: Remove the variability

Remove the variability of the time series by applying a log.

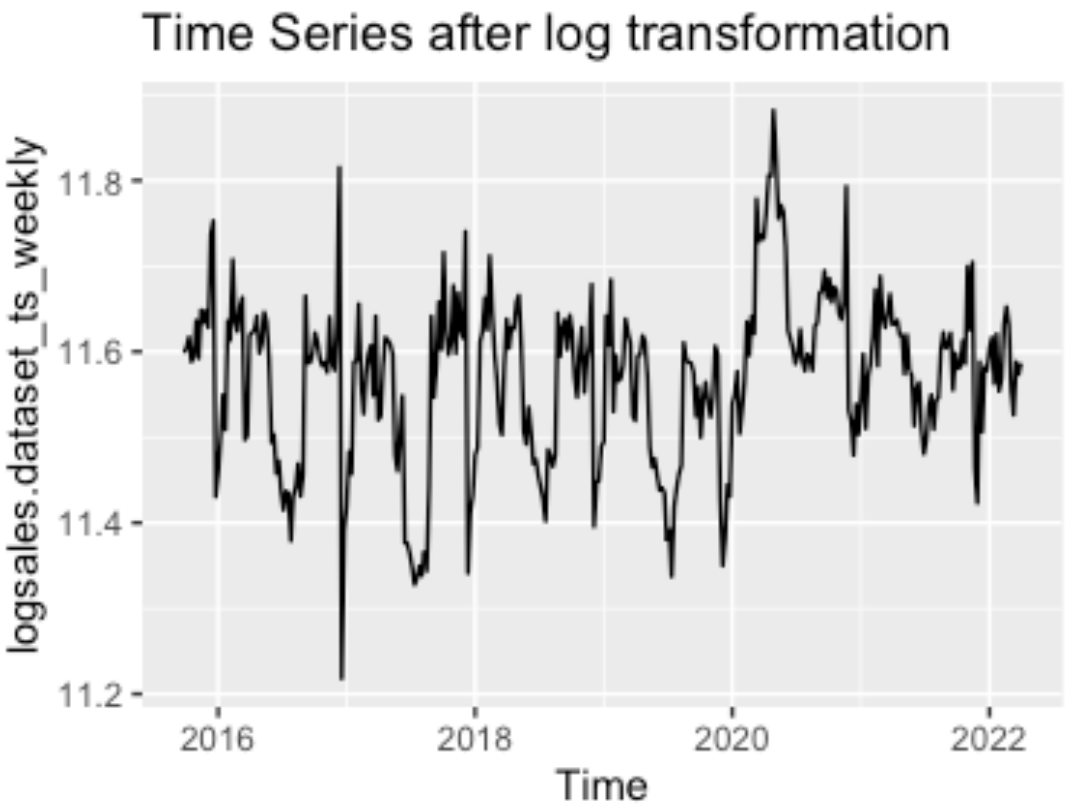


Figure 38: Time Series plot after log transformation

a) Dickey-Fuller Test:

```
> adf.test(log_weekly, alternative = "stationary",k = 0)

        Augmented Dickey-Fuller Test

data:  log_weekly
Dickey-Fuller = -7.801, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(log_weekly, alternative = "stationary", k = 0) :
  p-value smaller than printed p-value
```

Figure 39: Code snap showing the ADF test for log_weekly

b) KPSS Test:

```
> kpss.test(log_weekly)

        KPSS Test for Level Stationarity

data:  log_weekly
KPSS Level = 0.4224, Truncation lag parameter = 5, p-value = 0.0675
```

Figure 40:Code snap showing KPSS test for log_weekly

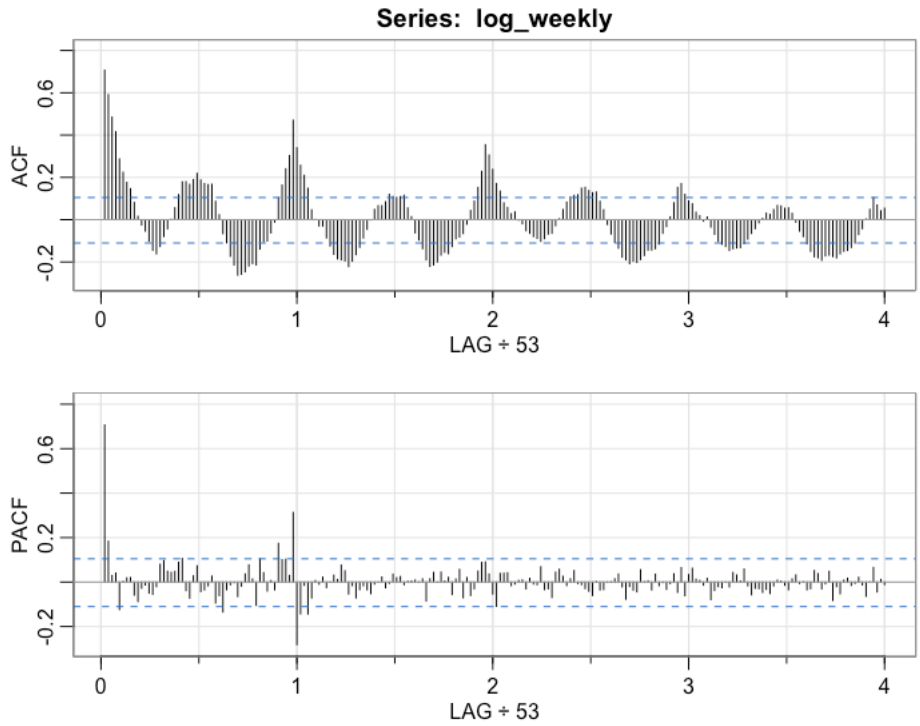


Figure 41: Autocorrelation plots for log_monthly

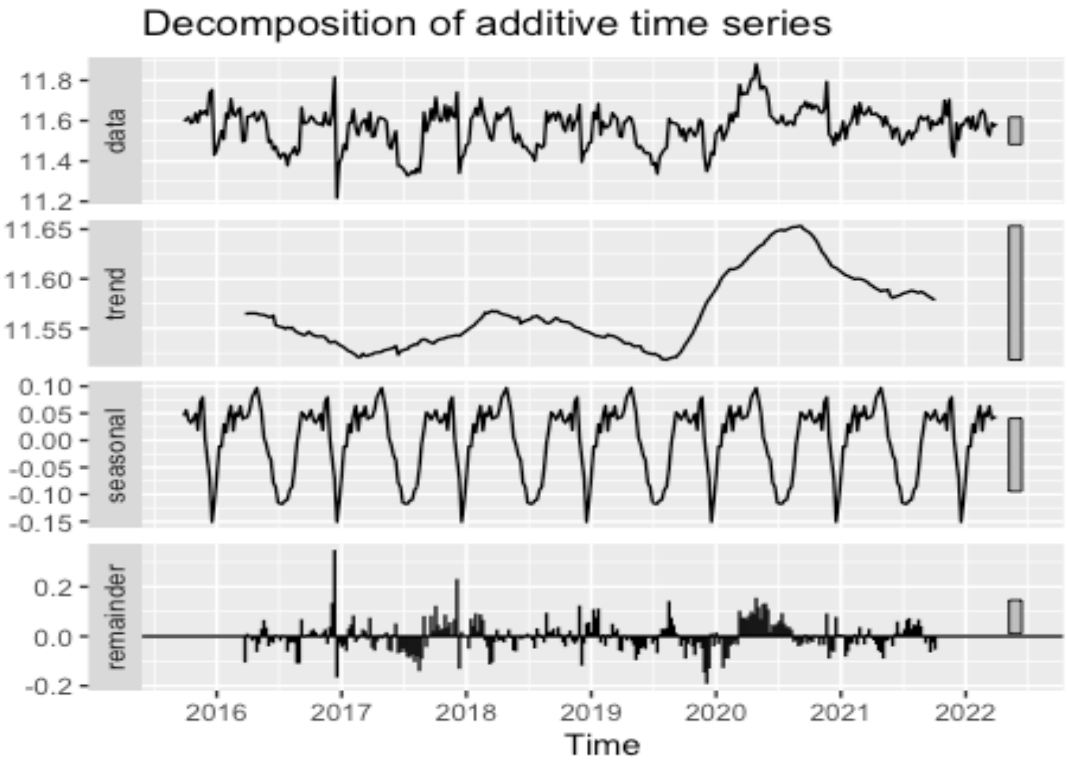


Figure 42: Additive decomposition of the log-transformed weekly data

Step 2: Seasonal Differencing

Remove the trend by differentiating the time series.

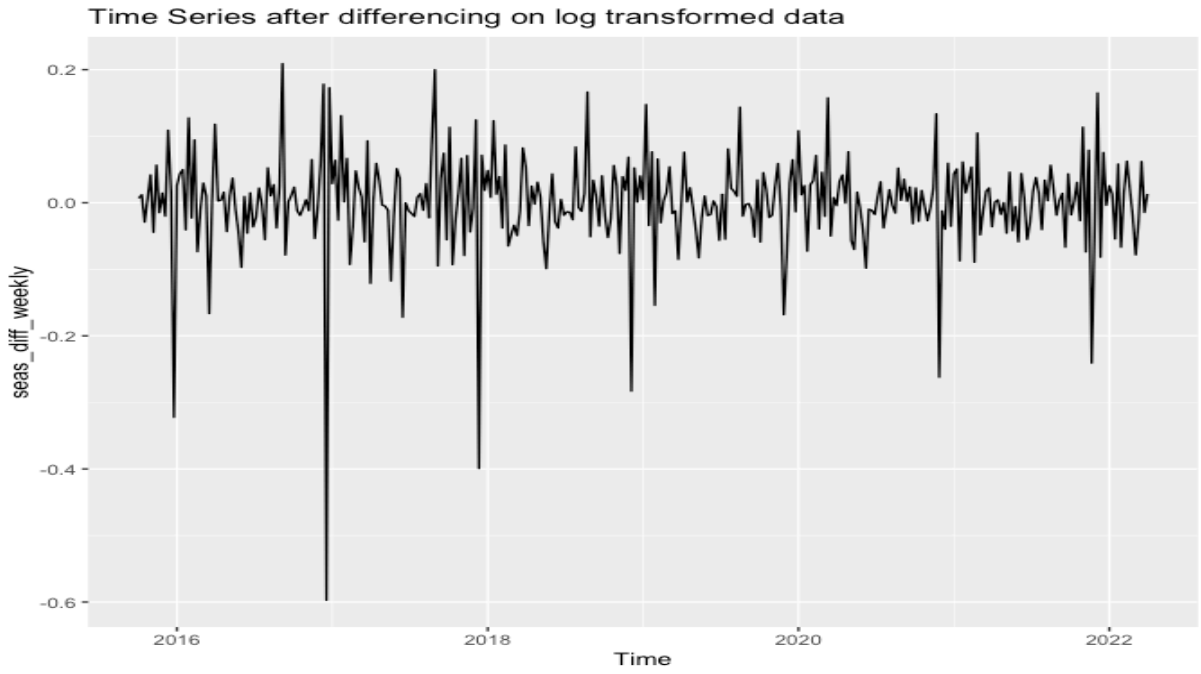


Figure 43: Differencing on log-transformed weekly data

a) Dickey-Fuller Test:

```
> adf.test(seas_diff_weekly, alternative = "stationary",k = 0)

        Augmented Dickey-Fuller Test

data:  seas_diff_weekly
Dickey-Fuller = -25.263, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(seas_diff_weekly, alternative = "stationary", k = 0) :
  p-value smaller than printed p-value
```

Figure 44: Code snap showing the ADF test for the first differenced log_weekly

b) KPSS Test:

```
> kpss.test(seas_diff_weekly)

        KPSS Test for Level Stationarity

data:  seas_diff_weekly
KPSS Level = 0.012009, Truncation lag parameter = 5, p-value = 0.1

Warning message:
In kpss.test(seas_diff_weekly) : p-value greater than printed p-value
```

Figure 45: Code snap showing the KPSS test for the first differenced log_weekly

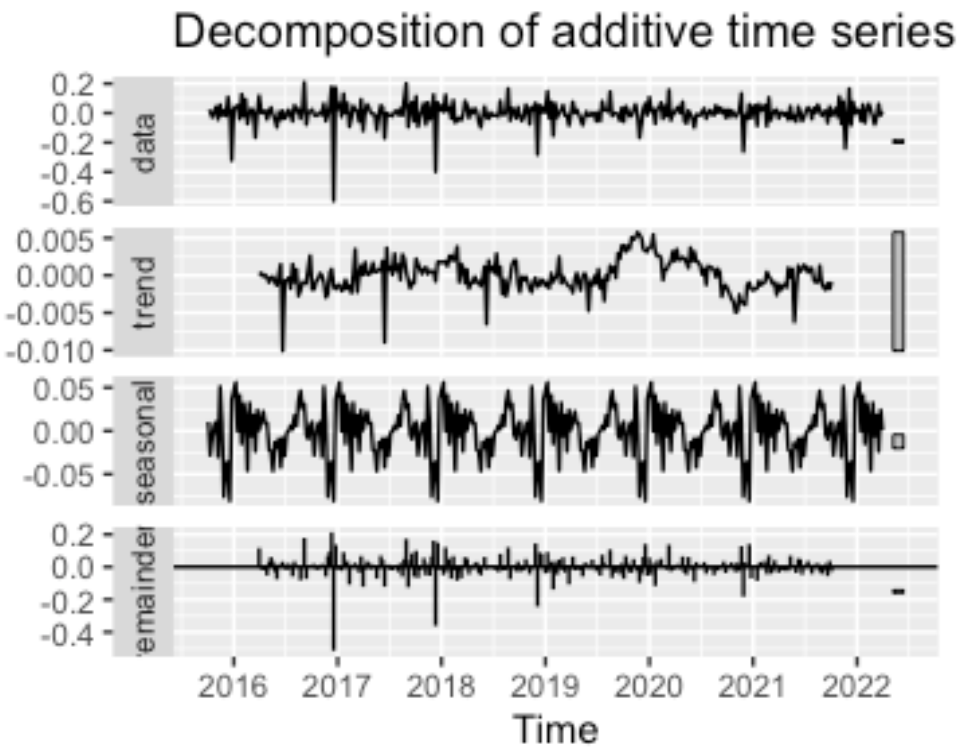


Figure 46: Additive decomposition of the above data

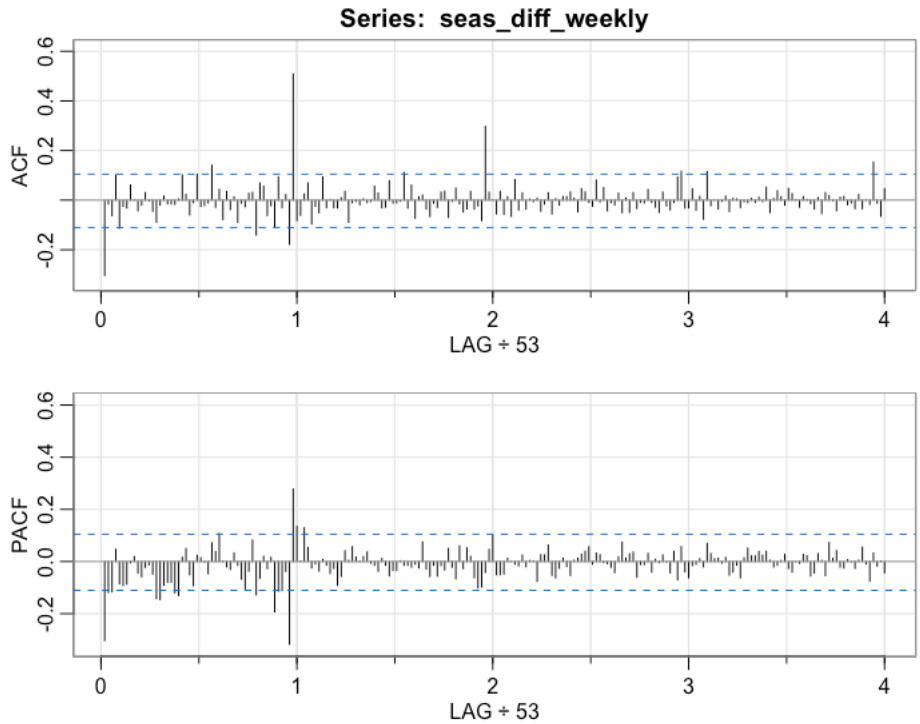


Figure 47: Autocorrelation plots for the first difference log_monthly

Step 3: Remove the seasonality by applying differentiation. Double differencing is done to the log-transformed data.

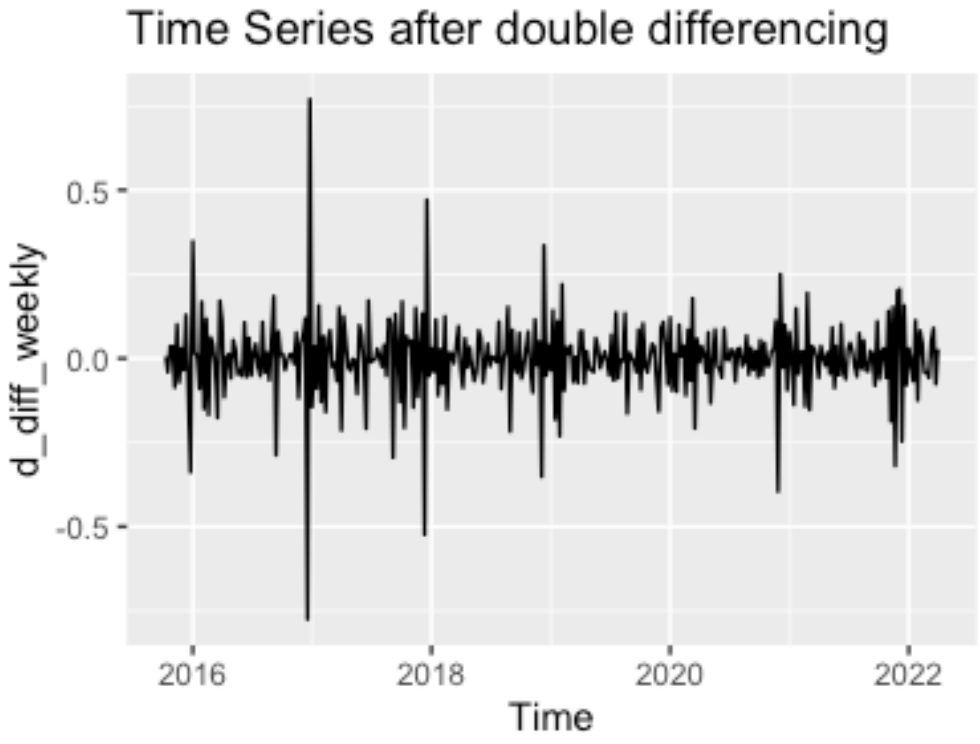


Figure 48: Time Series plot after double differencing the log-transformed data

a) **Dickey-Fuller Test:**

```
> adf.test(d_diff_weekly, alternative = "stationary",k = 0)

        Augmented Dickey-Fuller Test

data:  d_diff_weekly
Dickey-Fuller = -37.485, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(d_diff_weekly, alternative = "stationary", k = 0) :
  p-value smaller than printed p-value
```

Figure 49: Code snap showing the ADF test for the second differenced log_weekly

b) **KPSS Test:**

```
> kpss.test(d_diff_weekly)

        KPSS Test for Level Stationarity

data:  d_diff_weekly
KPSS Level = 0.0086577, Truncation lag parameter = 5, p-value = 0.1

Warning message:
In kpss.test(d_diff_weekly) : p-value greater than printed p-value
```

Figure 50: Code snap showing the KPSS test for the second differenced log_weekly

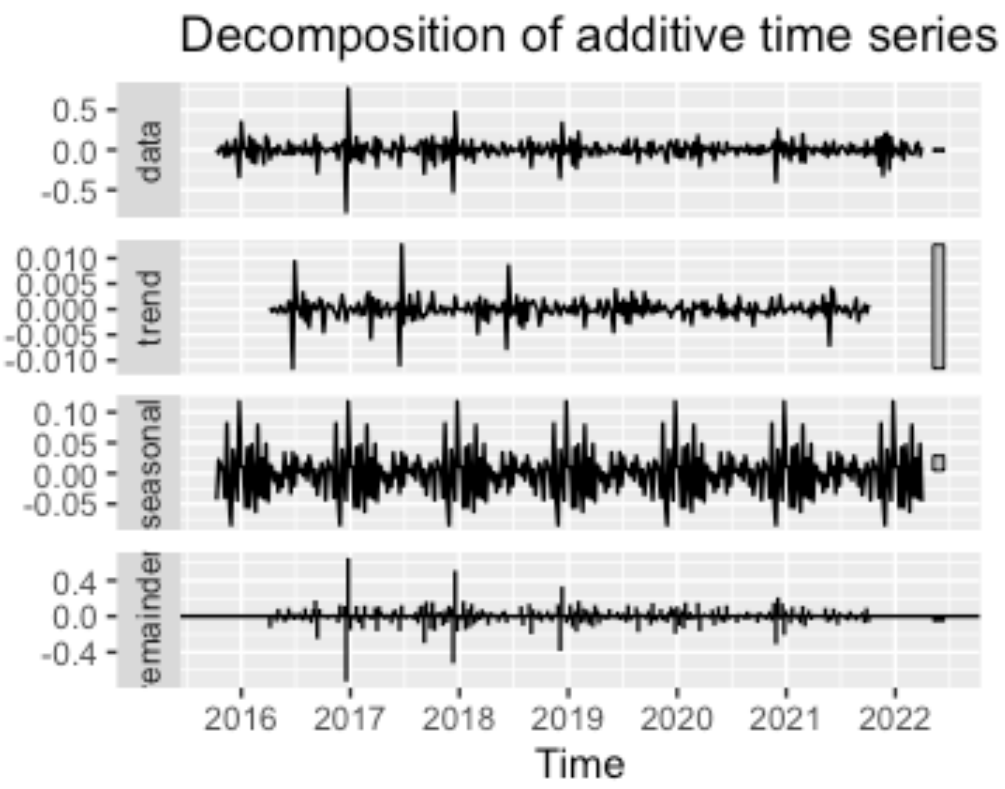


Figure 51: Decomposition after double differencing the log-transformed data

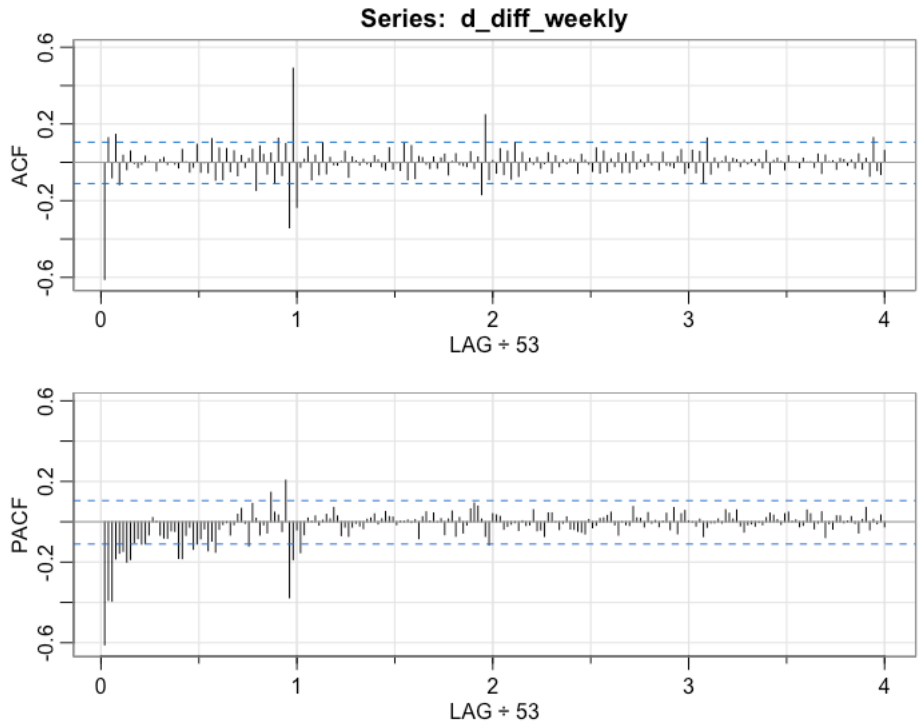


Figure 52: Autocorrelation plots for the second difference log_monthly

For a better understanding of the transformed time series, a comparison plot is given below in figure 53. At first, the log transformation is performed on the original weekly sales to remove variability. Followed by this, the log-transformed data is differentiated twice to get rid of the seasonal differencing. As a result of these transformations, a graph similar to white noise is obtained in the final row of figure 53.

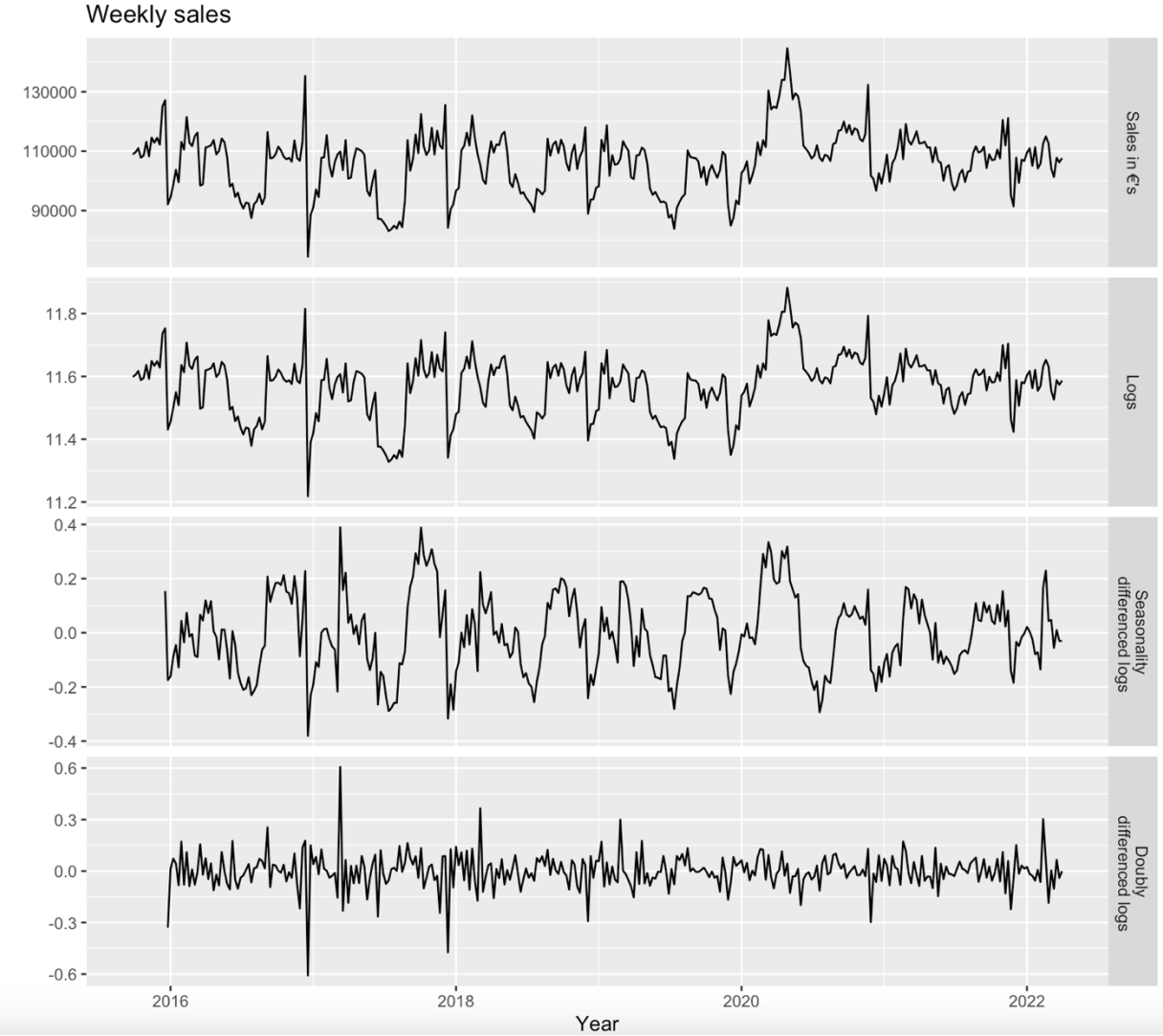


Figure 53: Comparison plot of the weekly sales

The above decomposition plot highlights that trend and seasonality have been removed from the time series and the data is ready for feeding into an ARIMA model. ARIMA models describe the autocorrelation in the data.

	AIC	AICc	BIC
ARIMA(0,1,1,0,0,0,53)	-844.49	-844.42	-832.96
ARIMA(1,0,0,0,0,0,53)	-860.82	-860.7	-845.44
ARIMA(1,1,1,0,0,0,53)	-844.17	-844.05	-828.8
ARIMA(3,1,0,0,0,1,53)	-844.16	-843.91	-821.1
Auto.arima - ARIMA(0,0,1,2,0,0,53)	-847.62	-847.51	-832.25

Table 2: ARIMA models for monthly data (frequency = 53)

The models are selected if their AIC, and RMSE represent the minimum, and MAE is used to decide if the two models have similar RMSE and AIC. Table 1 shows the best ARIMA models for weekly periodic data. ARIMA(1,0,0,0,0,0,53) having the least AIC value of -860.82 outperforms the rest of the models including the auto.arima ARIMA(0,0,1,2,0,0,53) . The Ljung Box-test was performed on the mode. Followed by this, future predictions were made.

3.6 Machine Learning Methods

3.6.1: Long Short-Term Memory

Long Short-Term Memory networks or LSTMs are capable of handling long-term dependencies which are a part of the Recurrent Neural Networks (RNN). LSTM is the widely used method among other machine learning models as they avoid the problem of long-term dependency. To know more about how LSTM works refer to [13].

RNN is a particular form of Artificial Neural Network which mainly performs on sequential data. Generally, when performing a time series-based analysis the check for stationary is conducted. If the dataset is not stationary, it is first converted into a stationary dataset before performing the analysis. But the advantage of the current neural networks does not need stationary data. This is one of the major reasons for choosing LSTM over other classical models as the dataset in this case is not stationary. These models are capable of learning complex patterns of the data when compared to the ARIMA and SARIMA models. On the other hand, if the model is not able to make good predictions changing the dataset into a stationary form can be a good option for obtaining better results.

The length of the dataset is analysed and a split for the training and testing part is done. The first 60 values are reserved in the training set. The last 12 months of analysis are reserved in the testing set and later will be used to compare the results with the predicted values. Using *MinMaxScalar*( ) the dataset is converted into a scale of values 0 to 1. The importance of converting this is to avoid the difference in the magnitude of the data. Next, the scalar values are fit into the training set which is also converted to scalar train and test sets. The most challenging part while designing the LSTM model was to format the data exactly to provide input to the RNN model.

```
[8]  # splitting test and train 20:80
     train = df.iloc[:60]
     test = df.iloc[60:]

[9]  scaler.fit(train)
     scaled_train = scaler.transform(train)
     scaled_test = scaler.transform(test)

[10] scaled_train[:10]

     array([[0.63310491],
            [0.33582366],
            [0.42808304],
            [0.45759662],
            [0.36419868],
            [0.31326994],
            [0.63725817],
            [0.3280558 ],
            [0.18613477],
            [0.38373769]])

     # define generator
     input_n =  3
     features_n = 1
     generator = TimeseriesGenerator(scaled_train, scaled_train, length = input_n, batch_size= 1)
```

Figure 54: Code for scalar transformation and input transformation

This was solved using the library TimeseriesGenerator. It uses a sequence of the first 3 (n =3) values to predict the upcoming 4th value. The lines of codes in above figure 3.8 explains them. Finally, an LSTM layer with 100 neurons and the activation function as "*relu*" is designed. To compare the predictions a graph is plotted against the original values and for numerical comparison, the RMSE function from the SciKit learning libraries is used.

3.6.2: Simple Linear Regression

Linear Regression analysis defines a linear relationship between a dependent variable and an independent variable. It also fits a straight line which approximates the relationship between both these variables. The mathematical equation for the simple linear regression is given as:

$$Y = a + bX + e$$

Equation 8: Simple Linear Regression formula

where Y is the predicted value of the dependent variable y for every specified value of the independent variable X, a is the intercept of the line, b is the regression coefficient, e is the error and X is an independent variable.

```
[9] from sklearn.linear_model import LinearRegression
    LR_model = LinearRegression()
```

```
x1,x2,x3,y=df['Sales_LastMonth'],df['Sales_2Months_back'], df['Sales_3Months_back'], df['SALES']
x1,x2,x3,y=np.array(x1),np.array(x2),np.array(x3),np.array(y)
x1,x2,x3,y=x1.reshape(-1,1),x2.reshape(-1,1),x3.reshape(-1,1),y.reshape(-1,1)
x_final=np.concatenate((x1,x2,x3),axis=1)
print(x_final)
```

```
[[478939. 448795. 545926.]
 [488582. 478939. 448795.]
 [458066. 488582. 478939.]
 [441426. 458066. 488582.]
 [547283. 441426. 458066.]
 [446257. 547283. 441426.]
 [399887. 446257. 547283.]
 [464450. 399887. 446257.]
```

Figure 55: Codes from moving average calculation

Regression analysis helps the business organisation to better understand their data points which makes decision-making easier. Here, in this study analysing the historical sales data can make us understand what are the top sellers of the store and which areas of sales are to be improved. The data obtained is also of linear form which makes linear regression possible. Sales are the dependent variable to be predicted based on the independent variable time. Initially, this analysis was performed with the help of the *SciKit learning* library, which was not successful. Later, the same analysis was carried out in R using the *'tslm' function – Fit a linear model with time series components*. The model is capable of fitting linear models to time series including trend and seasonality components. The moving average values were calculated to train the model. Once the model is trained and fitted, the predicted values are plotted against the actual sales to compare the performance. Similarly, the RMSE is also taken to measure the accuracy.

3.6.3: Random Forest Regression

The Random Forest algorithm is considered one of the easiest machine learning algorithms and the most widely used due to its uniqueness and simplicity of execution. The decision trees approach is carried out where each tree generated depends on the sampled random vector value. The forests which are considered the ensembles of the decision trees are trained using the bagging method. In the end, several decision trees are combined into a single tree. As Random forest algorithms are capable of handling overfitting it is much preferred over other models. While analysing high-dimensional datasets like the one in this case, the model makes use of random subsets of features which improves its performance.

```
[10] from sklearn.ensemble import RandomForestRegressor
     RF_model = RandomForestRegressor(n_estimators=100, max_features=3, random_state=1)

[ ]  from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = x_final[:-30], x_final[-30:],y[:-30],y[-30:]

[ ]  RF_model.fit(X_train, y_train)
     LR_model.fit(X_train,y_train)
```

Figure 56: Fitting of the Random Forest model

Some of the reasons for selecting random forests over the other models were: this algorithm is not influenced by the outliers and secondly, the noise in the dataset is identified properly and is not included during the pattern recognition by the model. Once the model is fitted using the suitable libraries from SciKit learning, the performance of the model is evaluated using its accuracy, RMSE and the confusion matrix.

4. Results

Comparison of Predicted Sales of different models

The below plots are results obtained from different models designed.

4.1 Forecasts using Holt-Winters' exponential smoothing:

The first plot is the HW multiplicative, the second is the HW additive and the third is the additive damped model. HW multiplicative model can catch up with the trend and seasonality much better than the additive model.

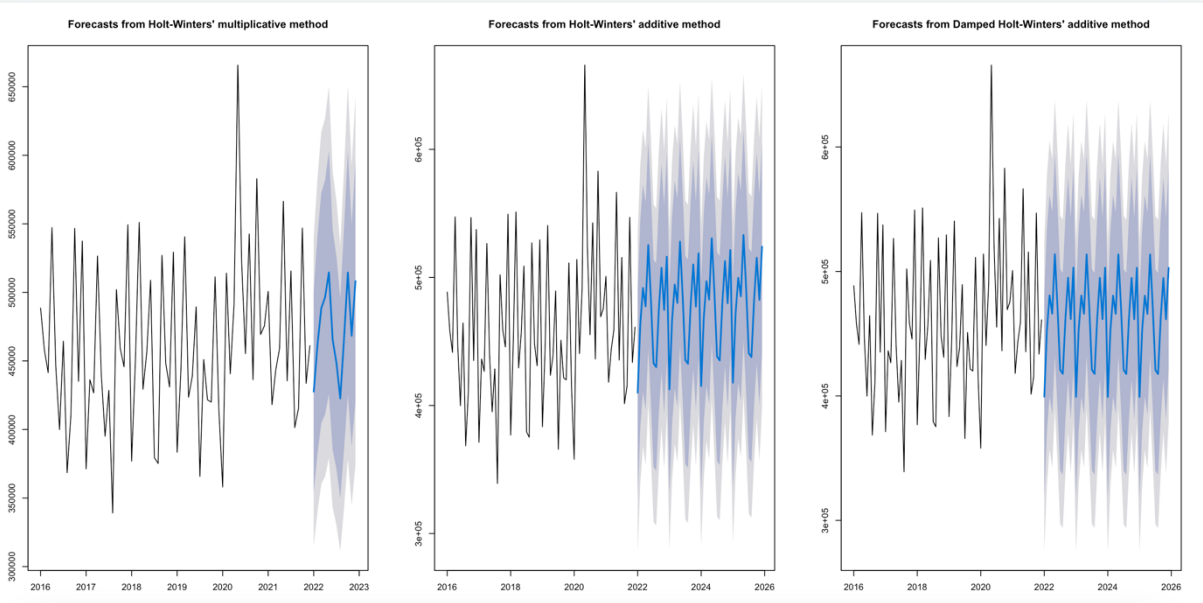


Figure 57: Forecasts from Holt-Winters' methods

As referred from the table below, HW multiplicative model is the best for this time series since it has the lowest AIC, BIC and RMSE.

Model Name	Akaike Information Criterion (AIC)	Bayesian Information Criterion (BIC)	Root Mean Square Error (RMSE)
Holt-Winters' multiplicative method	1913.863	1952.567	55007.86
Holt-Winters' additive model	1915.333	1954.036	55630.13
Damped Holt-Winters' additive model	1916.213	1957.193	55199.04

Table 3: ARIMA models for monthly data (frequency = 53)

4.2 Forecasts using ARIMA – monthly data

For the monthly sales time series – the auto.arima - ARIMA(0,0,0,0,0,1,12) was the best-fitted model. This model has given the lowest AIC of -91.97. For the above-stated ARIMA model, the residuals are normally distributed and have constant variance. Almost all the bars in the ACF plot are within the 95% confidence interval.

```
> #Auto-ARIMA
> auto.arma11_monthly = auto.arima(log(monthly_ts))
> auto.arma11_monthly
Series: log(monthly_ts)
ARIMA(0,0,0)(0,0,1)[12] with non-zero mean

Coefficients:
         sma1     mean
      0.4025  13.0322
s.e.  0.1175   0.0191

sigma^2 = 0.015:  log likelihood = 48.98
AIC=-91.97   AICc=-91.61   BIC=-85.14
```

Figure 58: Auto-ARIMA results for monthly sales

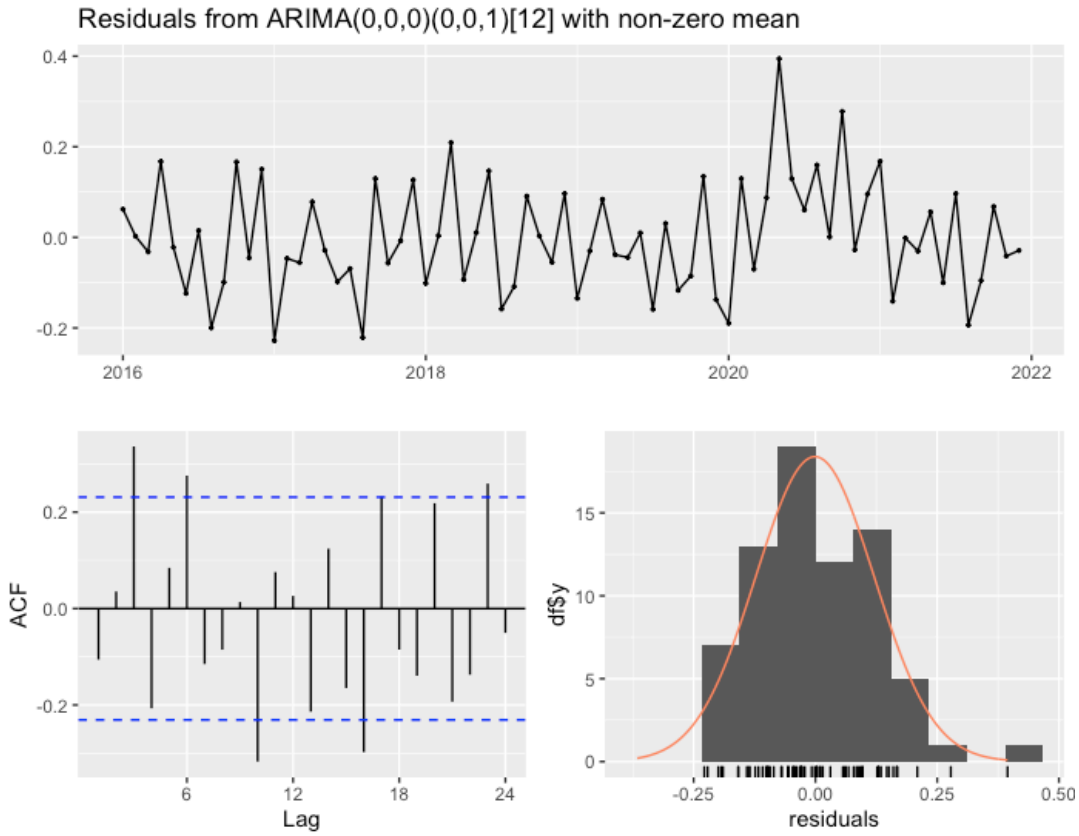


Figure 59: ARIMA residuals - Monthly data, p = 12

```
> ar.mod_monthly = Arima(log(monthly_ts), order=c(0, 0, 0),
+                 seasonal = c(0, 0, 1),
+                 include.drift = TRUE)
> ar.mod_monthly
Series: log(monthly_ts)
ARIMA(0,0,0)(0,0,1)[12] with drift

Coefficients:
         sma1  intercept  drift
      0.3825    13.0122  5e-04
s.e.  0.1209     0.0362  9e-04

sigma^2 = 0.01518:  log likelihood = 49.19
AIC=-90.38   AICc=-89.78   BIC=-81.27
```

Equation of the ARIMA (0,0,0,0,0,1,12) : $y_{t = 13.0122 + 0.3825 * e_{t-1}}$

The Ljung-box test confirms that the residuals of the selected model are independent.

```
> checkresiduals(auto.arma11_monthly)

        Ljung-Box test

data:  Residuals from ARIMA(0,0,0)(0,0,1)[12] with non-zero mean
Q* = 36.205, df = 12, p-value = 0.0003003

Model df: 2.   Total lags used: 14
```

Figure 60: Ljung-Box test for monthly data

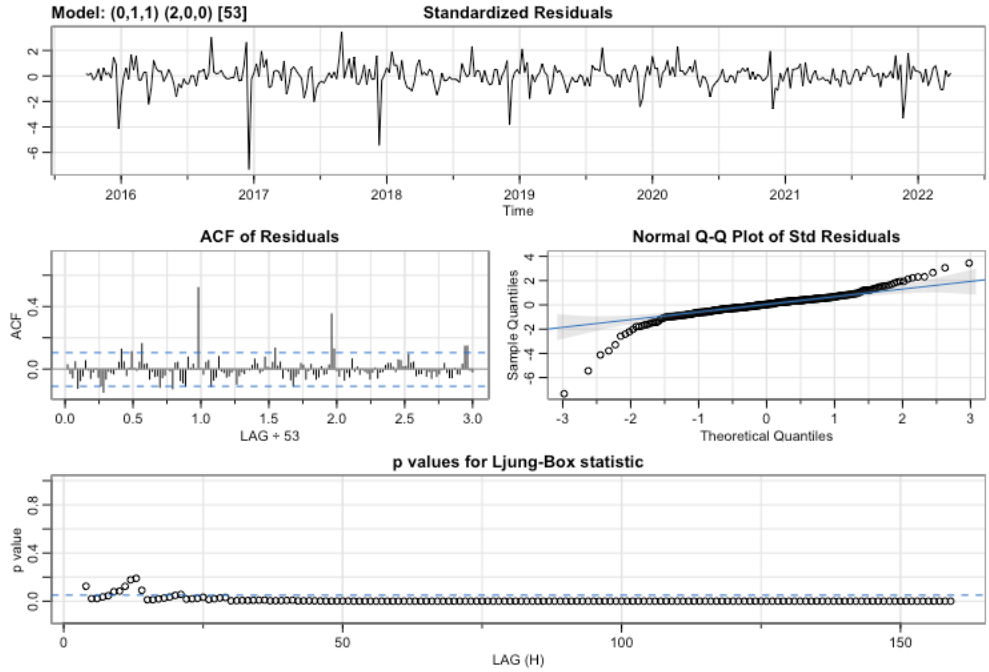


Figure 61:Residuals plot from ARIMA – monthly

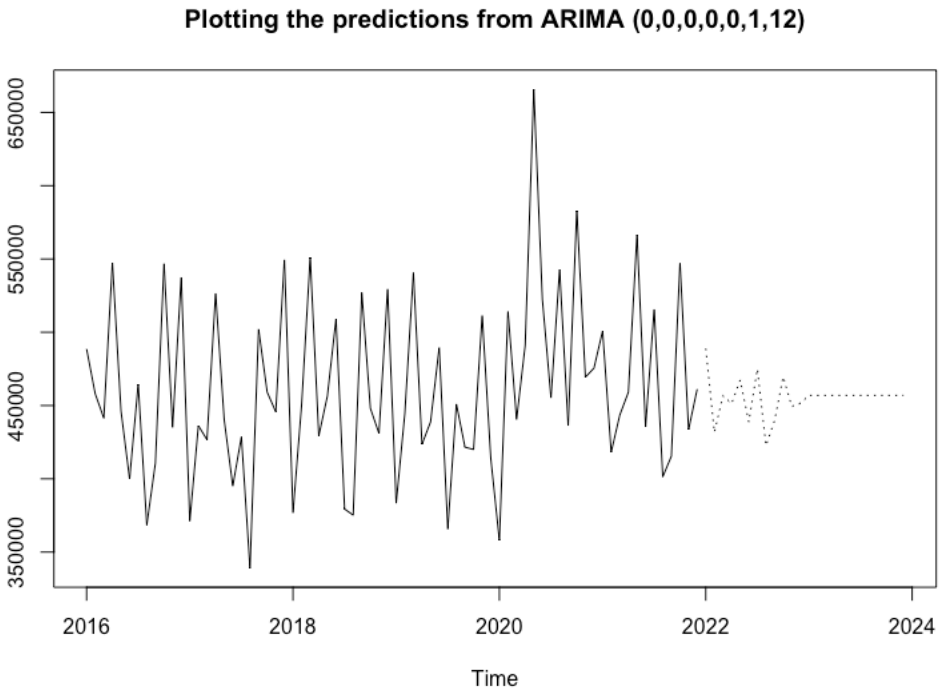


Figure 62: Predictions from ARIMA – monthly

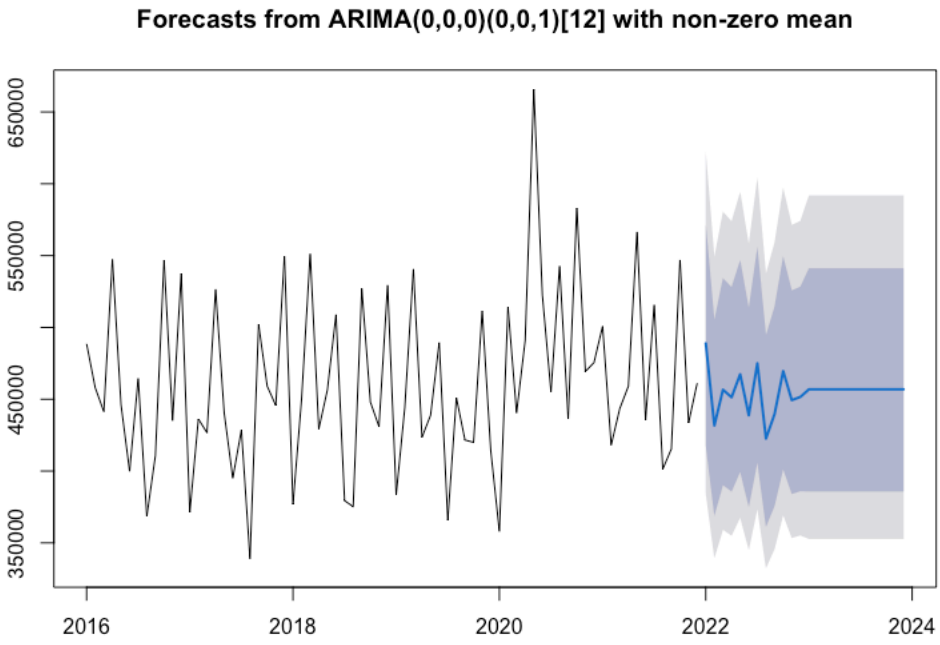


Figure 63: Forecasts from ARIMA – monthly

4.3 Forecasts using ARIMA – weekly data

For the weekly sales time series – the auto.arima - ARIMA(1,0,0,0,0,0,53) was the best-fitted model. This model has given the lowest AIC of -91.97. For the above-stated ARIMA model, the residuals are normally distributed and have constant variance. Almost all the bars in the ACF plot are within the 95% confidence interval.

```
> # ARIMA models
> #AR model
> ar.mod_weekly = Arima(log(weekly_sales_ts), order=c(1, 0, 0),
+                       seasonal = c(0, 0, 0),
+                       include.drift = TRUE)
> ar.mod_weekly
Series: log(weekly_sales_ts)
ARIMA(1,0,0) with drift

Coefficients:
         ar1  intercept  drift
      0.6971    11.5434  2e-04
s.e.  0.0383     0.0241  1e-04

sigma^2 = 0.004786:  log likelihood = 434.41
AIC=-860.82   AICc=-860.7   BIC=-845.44
```

Figure 64: Auto-ARIMA results for weekly sales

Equation of the ARIMA (1,0,0,0,0,1,53) : $y_{t = 11.5434 + 0.6971 * e_{t-1}}$

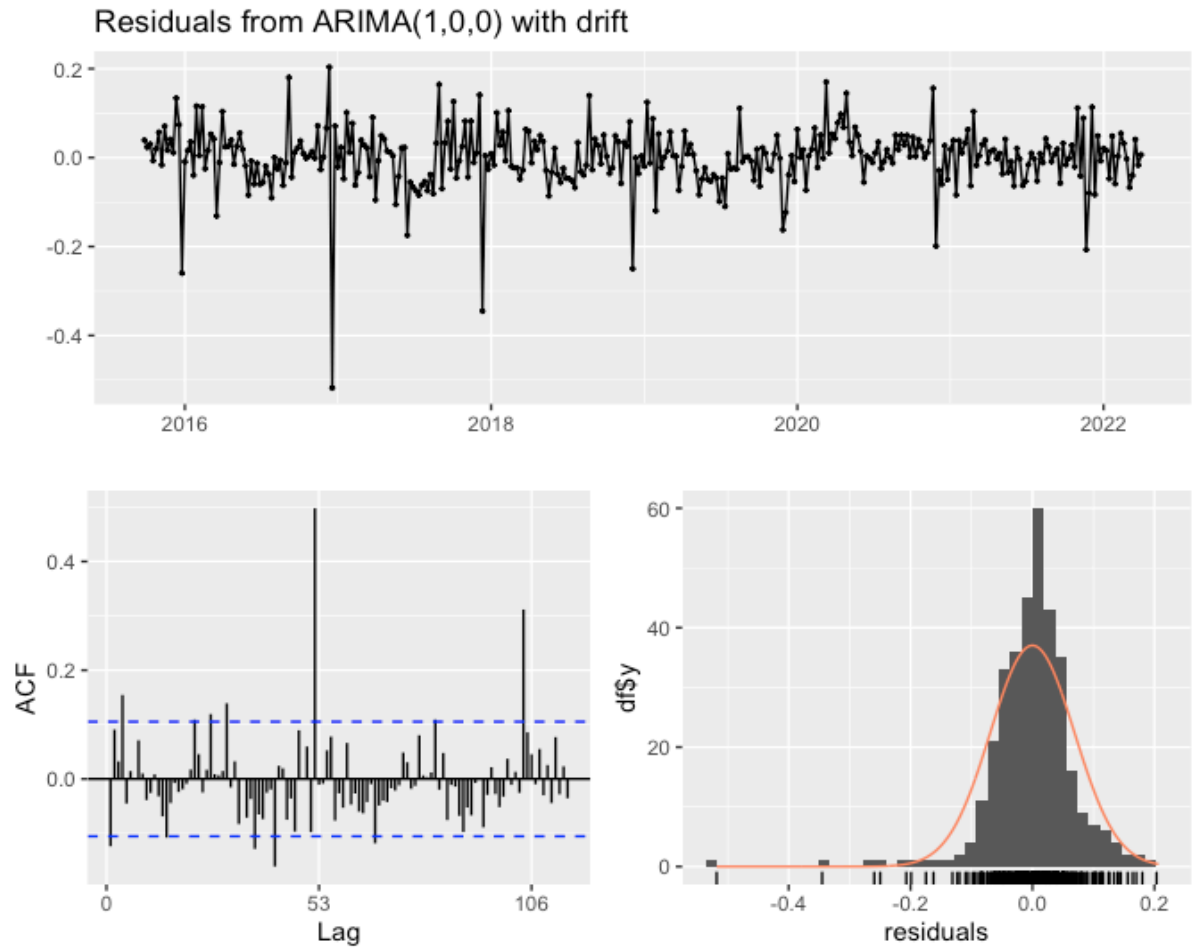


Figure 65: ARIMA residuals - Weekly data, p = 53

The Ljung-box test confirms that the residuals of the selected model are independent.

```
> checkresiduals(ar.mod_weekly)

        Ljung-Box test

data:  Residuals from ARIMA(1,0,0) with drift
Q* = 212.76, df = 66, p-value < 2.2e-16

Model df: 3.   Total lags used: 69
```

Figure 66: Ljung-Box test for weekly data

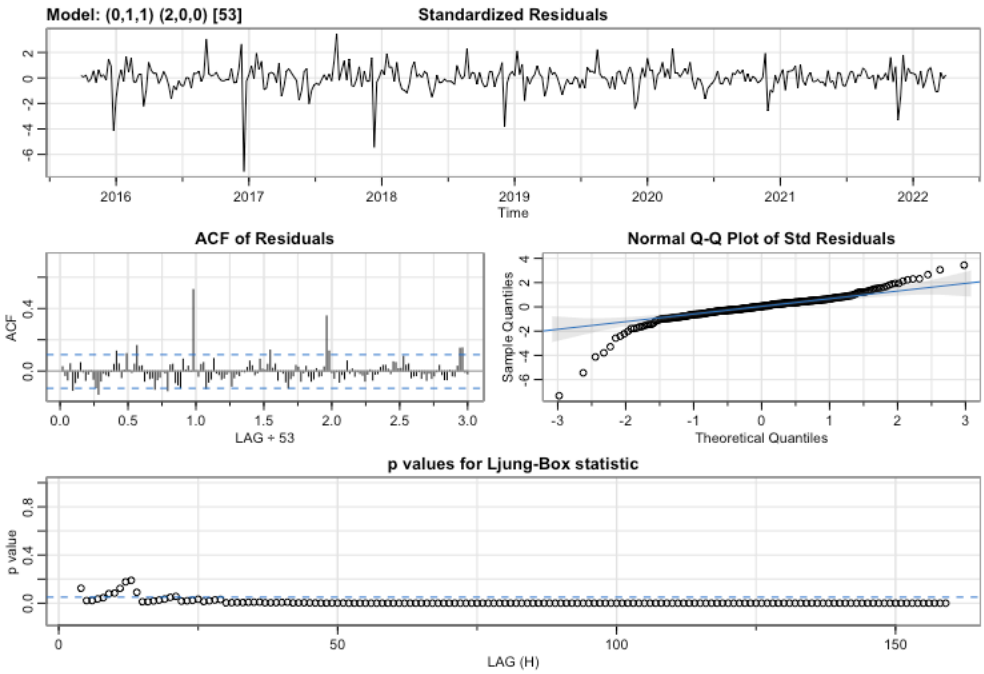


Figure 67: Residual plot of weekly data

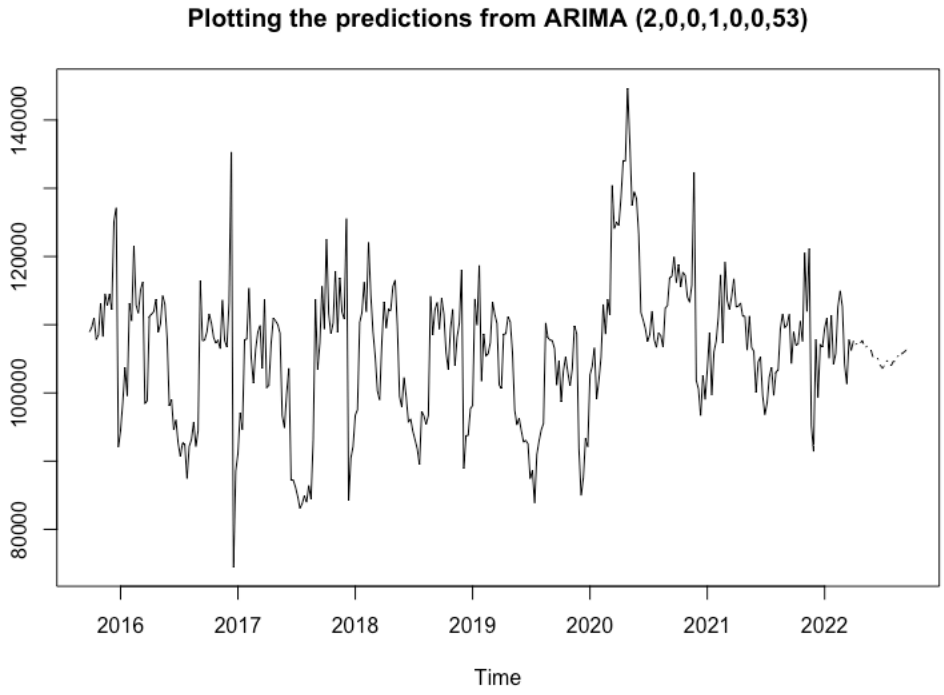


Figure 68: Predictions from ARIMA – weekly

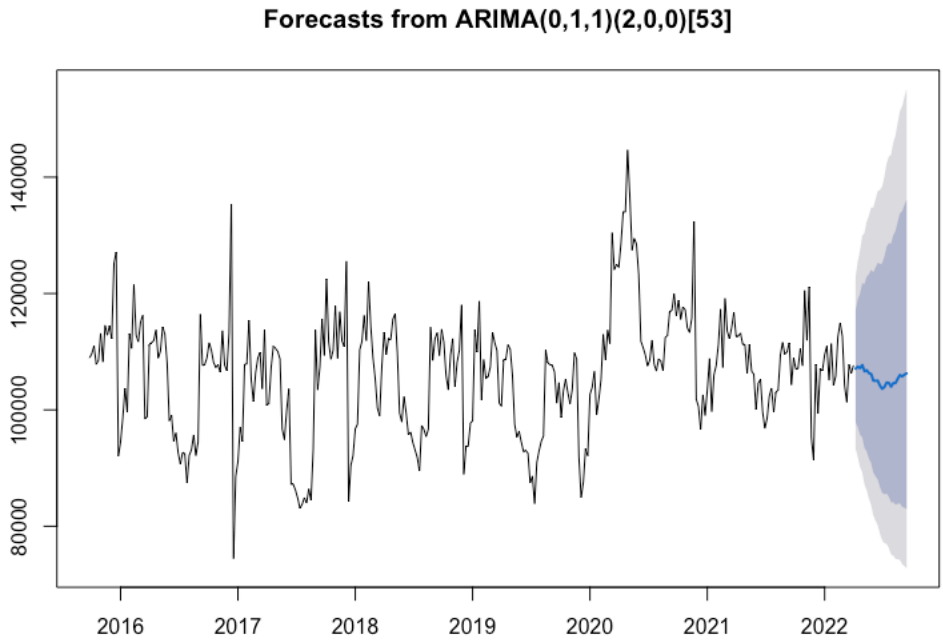


Figure 69: Forecasts from ARIMA – weekly

4.4 Forecasts using LSTM

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100)               40800

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 40,901
Trainable params: 40,901
Non-trainable params: 0
_________________________________________________________________
```

Figure 70: Results from the LSTM model

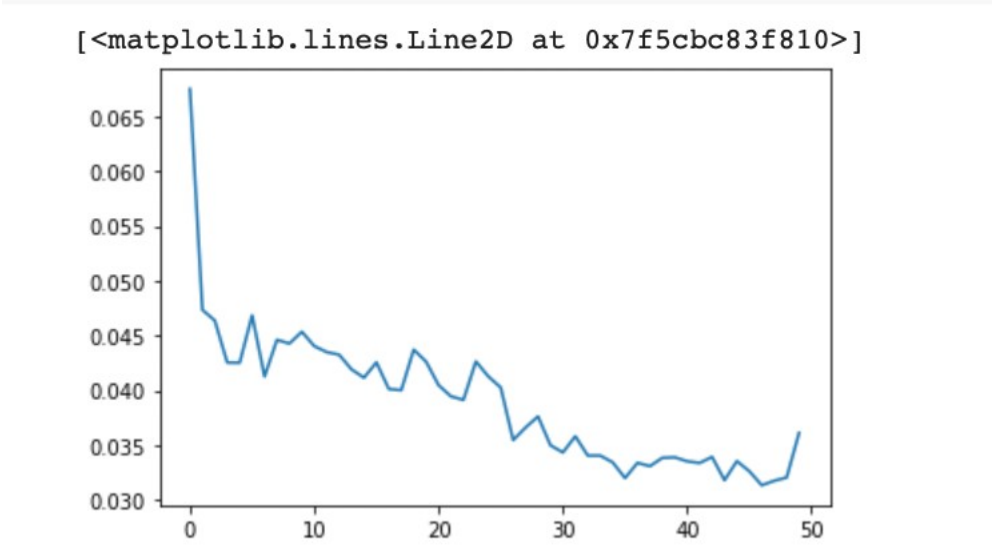


Figure 71: Plot of the losses per epoch

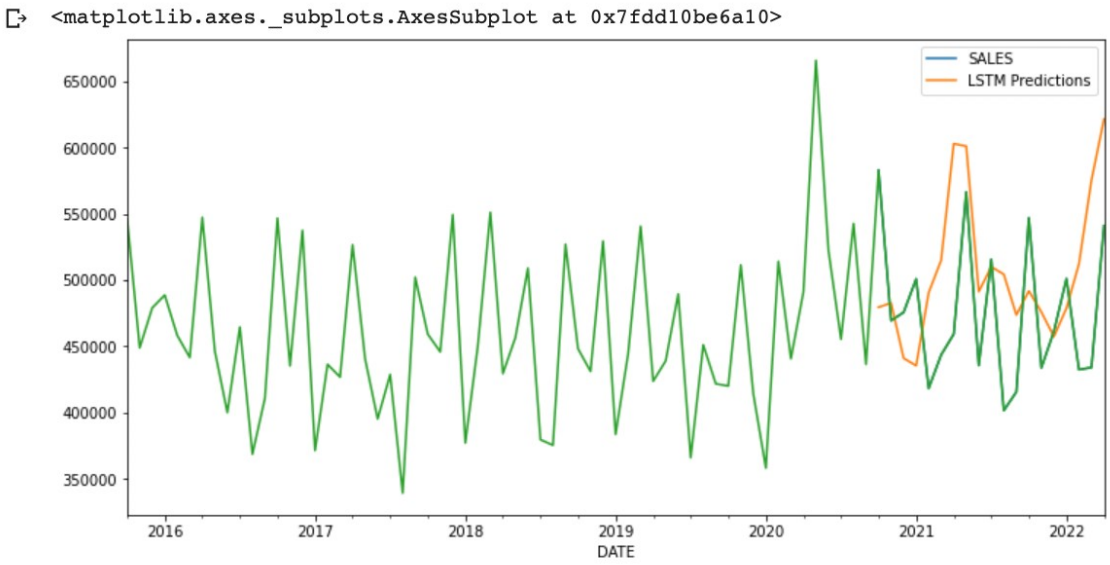


Figure 72: Time series plot of actual sales and the LSTM Predicted sales

4.5 Forecasts using Linear Regression

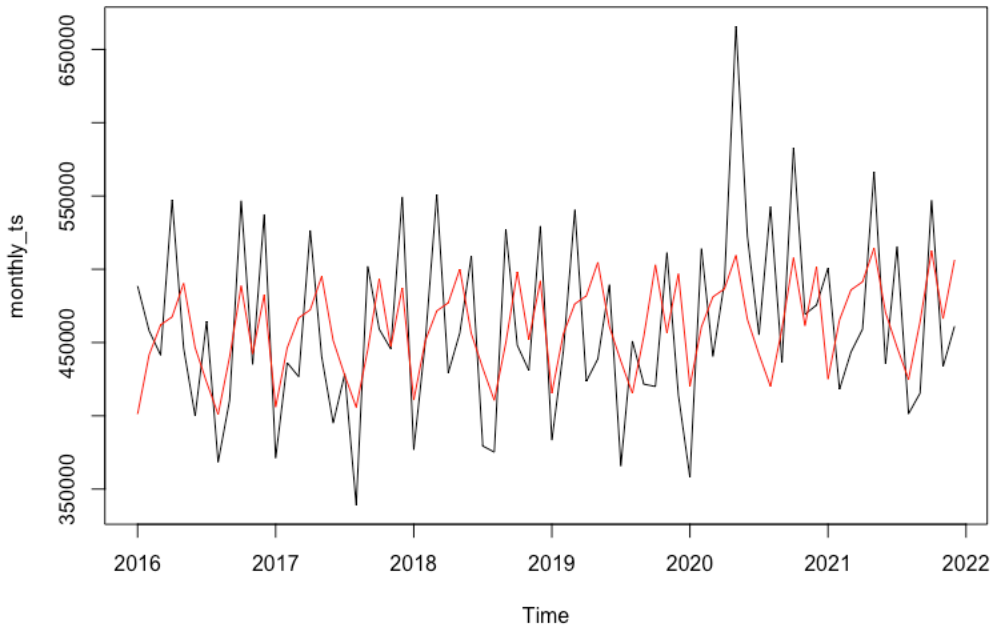


Figure 73: Sales predictions from the Linear model – monthly

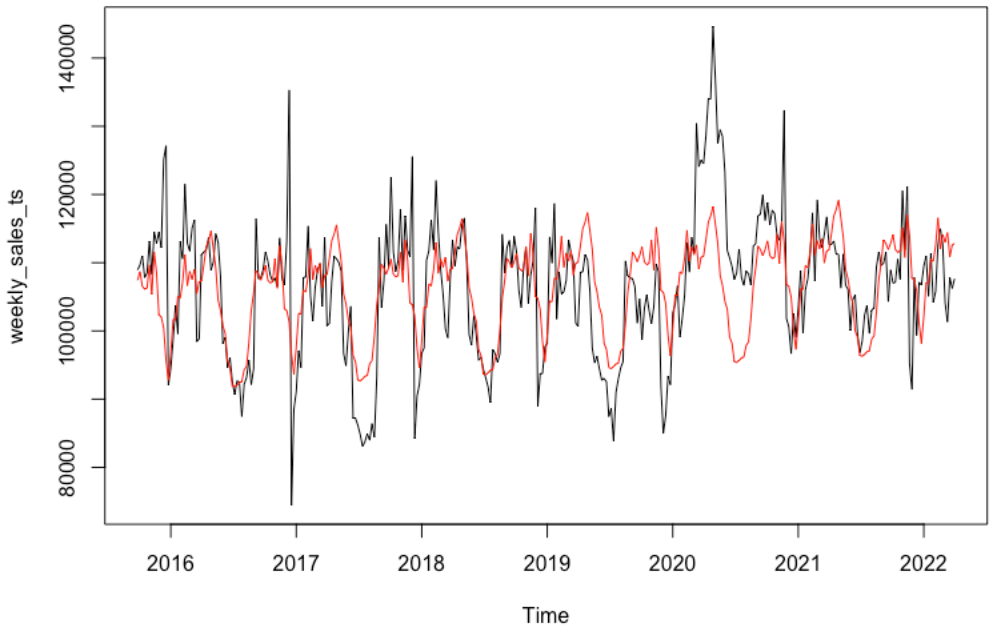


Figure 74: Sales predictions from the Linear model – weekly

4.6 Forecasts using Random Forest

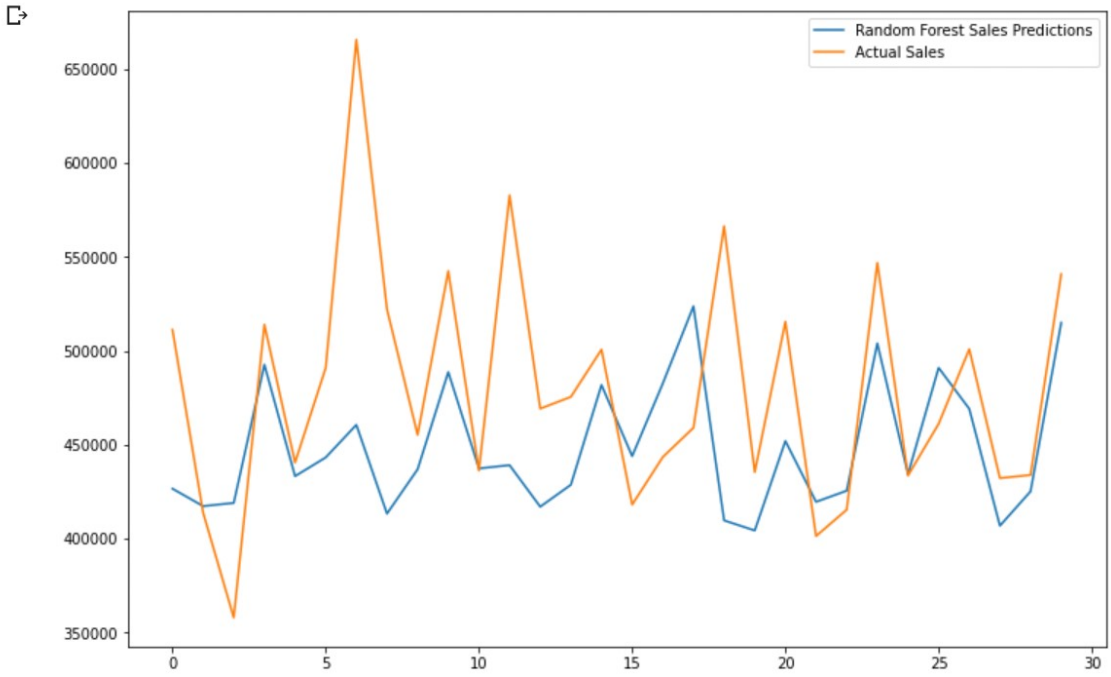


Figure 75: Sales predictions from the Random Forest model

4.7 Performance Summary of the different models

Models	RMSE values
LSTM	61260.98
Random Forest Regressor	67903.81

Table 4: Comparison of the algorithms' performances based on RMSE

5. Discussion and Conclusion

The objective of this thesis was to perform a comparison of the different statistical and machine learning approaches in predicting the sales of a retail store based on the historical data collected. The research carried out leads to a major added advantage to the businesses. It helps in a better decision-making process where each individual department of good sales is tracked and helps monitor where an improvement is required. For a business to compete in the current market, it is necessary for making such sales predictions in advance to meet the requirements of the customers. The RMSE values were used for the comparison of the final performances of the model. All three different models proved to be effective in terms of the predicted sales outputs. LSTM model proved to be the closest among the predictions. The lower the RMSE, the better the predictions were able to “fit” within the dataset.

For future work, the same study can be conducted for the weekly periodic dataset. Some other works may be like including more complex deep learning models and hybrid algorithms to compare with the existing models.

One of the major limitations faced during this study was the computational power to run the python codes. To overcome this Python modules were processed on the Google Colaboration environment over the Jupyter notebooks.

References

Referencing style used is IEEE.

[1] Andrew S. Denney & Richard Tewksbury (2012): "*How to Write a Literature Review, Journal of Criminal Justice Education"*.

[2] "*Statistical Methods for Sales Forecasting in Retail Industry"*. https://www.linkedin.com/pulse/statistical-methods-sales-forecasting-retail-industryajay-bidyarthy.

[3] Ofoegbu, Kenneth. "*A Comparative Analysis of Four Machine Learning Algorithms to Predict Product Sales for a Retail Store*", n.d., 64.

[4] Liu N, Ren S, Choi T, Hui C and Ng S 2013 "*Sales Forecasting for Fashion Retailing Service Industry: A Review 2013"*.

[5] *"What is Artificial Intelligence (AI),"* https://www.ibm.com/cloud/learn/what-isartificial-intelligence.

[6] Krishna A, Akhilesh V, Aich A and Hegde C 2018 "*Sales-forecasting of Retail Stores using Machine Learning Techniques"* Proc. 2018 3rd Int. Conf. Comput. Syst. Inf. Technol. Sustain. Solut. CSITSS 2018 160–6.

[7] Wang J and Liu L 2019 *"A Selection of Advanced Technologies for Demand Forecasting in the Retail Industry"*.

[8] Arif M A I, Sany S I, Nahin F I and Rabby A S A 2020 "*Comparison Study: Product Demand Forecasting with Machine Learning for Shop"* 171–6.

[9] Investopedia. *"What Does Statistics Study?"* https://www.investopedia.com/terms/s/statistics.asp.

[10] *"What Is AI/ML and Why Does It Matter to Your Business?"* https://www.redhat.com/en/blog/what-aiml-and-why-does-it-matter-your-business.

[11] Hyndman, R.J., & Athanasopoulos, G. (2018) "*Forecasting: principles and practice*" 2nd edition, OTexts: Melbourne, Australia. OTexts.com/fpp2.

[12] P.-F. Pai and C.-S. Lin, "*A hybrid arima and support vector machines model in stock price forecasting*", Omega, vol. 33, no. 6, pp. 497–505, 2005, ISSN: 0305-0483.

[13] Armstrong J S and Collopy F 1998 *"Integration of statistical methods and judgment for time series forecasting : principles from empirical research".*

Appendix:

For monthly data, another best fit model is as follows.

ARIMA (3,1,0,0,0,1,12) having another lowest AIC of -84.11. All the lags of the ACF were within the 95% C.I.

```
> #model4
> arima310_monthly = Arima(log(monthly_ts), order=c(3, 1, 0),
+                          seasonal = c(0, 0, 1),
+                          include.drift = TRUE)
> arima310_monthly
Series: log(monthly_ts)
ARIMA(3,1,0)(0,0,1)[12] with drift

Coefficients:
          ar1      ar2      ar3    sma1    drift
      -0.8997  -0.5972  -0.0349  0.5678  -0.0021
s.e.   0.1191   0.1429   0.1205  0.1285   0.0083

sigma^2 = 0.01498:  log likelihood = 48.06
AIC=-84.11   AICc=-82.8   BIC=-70.54
> checkresiduals(arima310_monthly)

        Ljung-Box test

data:  Residuals from ARIMA(3,1,0)(0,0,1)[12] with drift
Q* = 25.419, df = 9, p-value = 0.002541

Model df: 5.   Total lags used: 14
```

Figure 76: ARIMA results for another fit model - monthly data

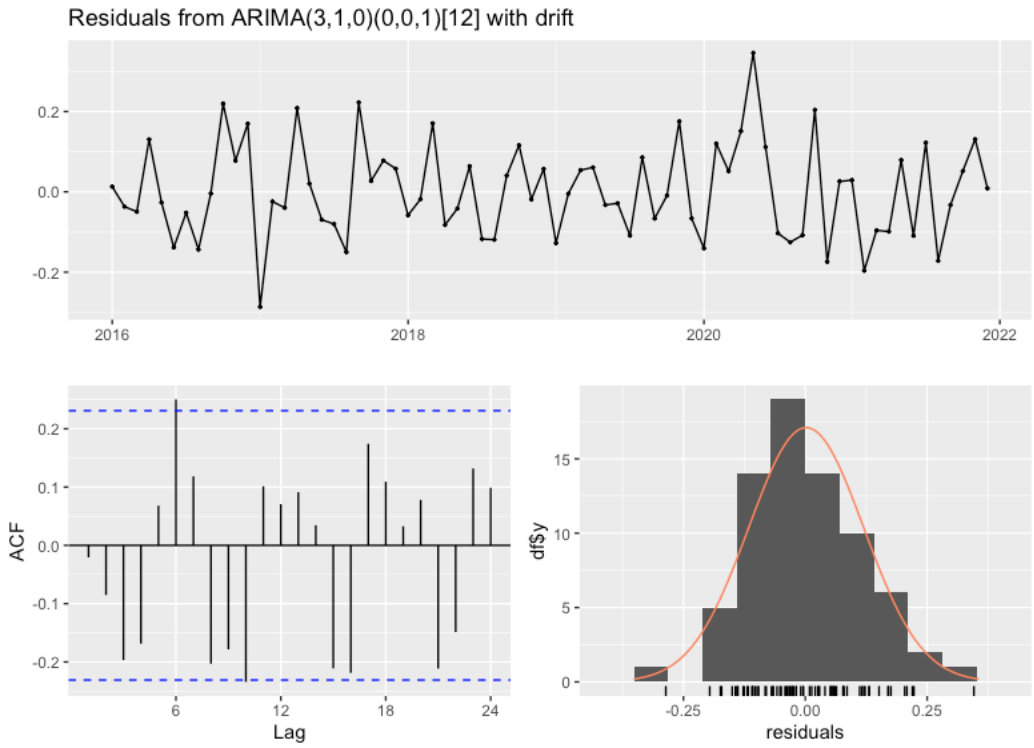


Figure 77: ARIMA residuals - Monthly data, p = 12