C O V E N T R Y
U N I V E R S I T Y

Faculty of Engineering, Environment and Computing

School of Computing, Electronics and Mathematics

Data Science
7150CEM - Data Science Project
Analysing Customer Behaviour and Sales Trend based on the Time Stamp

Author: Bharathi Cherukuri

SID: 12088229

Supervisor: **Dr. Marwan Fuad**

Submitted in partial fulfilment of the requirements for the Degree of Master of Science in Data Science

Academic Year: 2022/23

Declaration of Originality

I declare that this project is all my own work and has not been copied in part or in whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines, internet etc.) has been acknowledged by citation within the main report to an item in the References or Bibliography lists. I also agree that an electronic copy of this project may be stored and used for the purposes of plagiarism prevention and detection.

Statement of copyright

I acknowledge that the copyright of this project report, and any product developed as part of the project, belong to Coventry University. Support, including funding, is available to commercialise products and services developed by staff and students. Any revenue that is generated is split with the inventor/s of the product or service. For further information please see www.coventry.ac.uk/ipr or contact ipr@coventry.ac.uk.

Statement of ethical engagement

I declare that a proposal for this project has been submitted to the Coventry University ethics monitoring website (https://ethics.coventry.ac.uk/) and that the application number is listed below (Note: Projects without an ethical application number will be rejected for marking)

Signed: Bharathi Cherukuri Date: 09/12/2022

Please complete all fields.

First Name:	Bharathi
Last Name:	Cherukuri
Student ID number	12088229
Ethics Application Number	P141766
1st Supervisor Name	Marwan Fuad
2nd Supervisor Name	Nachiketa Chakraborty

This form must be completed, scanned and included with your project submission to Turnitin`. Failure to append these declarations may result in your project being rejected for marking.

Abstract

One of the most popular data science techniques, Time Series Forecasting is used in a variety of applications, including stock trading, predicting corporate sales, and weather forecasting. In today's business intelligence, for businesses involved in manufacturing, marketing, logistics, wholesale, and retail, sales forecasting is crucial. It enables businesses to design strategies that are better for their long-term success, estimate revenue from sales, and manage resources effectively. Sales predictions can be used to set benchmarks, assess the incremental effects of fresh initiatives, allocate resources in line with anticipated demand, and forecast upcoming budgets. Machine learning algorithms can assist organisations in predicting consumer behaviour by using data from previous transactions and demographic information, particularly in sales forecasting. The sales data of each and every individual item is now tracked by supermarkets and shopping centres in order to forecast future consumer demand. In this project I am going to analyse the sales reports of a real-life time series dataset by using supervised machine learning techniques, also exploring the reports in Tableau and show forecasting analysis.

This project provides and discusses the results of applying different Regression techniques to the sales dataset. Here I am applying some of the Machine learning techniques like Linear Regression, Random Forest Regression, Decision Tree Regression, XGBoost Regression, Forecasting Models like Seasonal Auto-Regression Integrated Moving Average (SARIMA) model and Recurrent Neural Network models (LSTM's, GRU's, Bidirectional LSTM's) on the Prakruthivanam Sales Data. We have four steps. The first step is exploratory data analysis (EDA), the second is data pre-processing, the third is implementing regression algorithms, and the fourth and final step is to calculate accuracy using the Mean Square Error of these Models and comparing the results. The EDA analysis mainly focus on the columns like Date, Day of the week, time of purchase, Total items purchased per bill, Total bill value, payment type, Total sale amount per day, etc. for the betterment of future sales. EDA involves creating some visualizations on different columns of the dataset so that client can get better understanding of the flow in the business. Data Pre-processing involves removing unwanted columns (which can be decided using EDA) and converting the columns in such a way that to make it ready for the model implementation. In the, third step we split the entire data into train 70% and test 30% and applying the model to train data and make predictions on the test data. Final step is to measure the accuracy. Prediction outcomes and algorithm performance metrics were obtained after the aforementioned techniques

were implemented in a Jupiter notebook using Python programming. Also Implement Forecasting using Tableau.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my sincere gratitude to Madhu Sir the Proprietor of Prakruthivanam Store, Hyderabad for accepting my request to use store data and sharing the information about the store. Also I would like Thank the General Manager Trijan, Prakruthivanam for explaining me the details of the data and clarifying my doubts. Most importantly, I am very grateful to my supervisor Dr. Marwan Fuad for guiding me to start with this kind of real time data and always pushing me to do my best.

1 Introduction

Retailer expansion and market competitiveness are both on the rise in the most populous cities. Retailers are now a days concentrating on Analysing their Sales and investing their time and money to get better understanding about their sales trend to attract customers for increasing their business in this competitive world. Sales targets are the benchmarks that each person or department must meet in order to fulfil the objectives of the business. This is accomplished by forecasting consumer behaviour using information from prior transactions. This enables businesses to make precise projections and plan for upcoming occurrences. The basic ideas behind sellers and buyers are supply and demand. The ultimate goal is to multiply with the customers. Businesses can effectively manage their cash flow and allocate resources for future expansion with the help of an accurate sales forecast. Here comes the major part of the Data Science, Machine Learning Techniques, and Forecasting Methods etc. are used by businesses to forecast sales and revenue.
Machine Learning is an evolving field where several methods are being developed and tested on different datasets daily. Time series Forecasting is a method that allows companies to use this data to develop accurate plans that can impact how they do business. Time-series forecasting is a technique that uses historical data to predict future sales using machine learning models, deep learning models, traditional statistical models, and hybrid models. Predicting/forecasting the Sales accurately is critical for these kinds of organizations in order to plan the staff, stock, offers etc. This Project aims to apply some regression model to forecast the future sales of Prakruthivanam Organic Store.

1.1 Background to the Project

In Hyderabad, Prakruthivanam is one of the companies in the organic food retailers industry. Due to the numerous advantages that organic food provides, demand has suddenly increased. Organic food is beneficial for overall health because it is free of chemicals and preservatives. Prakruthivanam in Kphb Colony, Hyderabad was established in December 2018, is one of the top players in the category Organic Food Retailers. They have customers from all over India and even from foreign countries like UK, USA etc. They take online orders and ship all over the world. For both local and out-of-town consumers, Prakruthivanam serves as a one-stop shop.

There are many challenges faced by entrepreneurs especially in retail business for delivering on customer experience. The customer experience plays a vital role in deciding

the success of retail business. Retaining existing clients while attracting new ones is one of the biggest challenges our client Prakruthivanam faces. So our client wants to know the future sales so that he can plan to arrange more stock, staff on high sale days and plan some weekly offers/ festival offers at those days having low sale days to increase the sale. In this project, Time series daily sales data of Prakruthivanam store has been collected and implemented some of the machine learning techniques and also compare the accuracy and choose the best model.

1.2 Project Objectives

This study's main goal is to forecast future sales by analysing historical data. Given that the retail food industry is one of the industries with the fastest growth rates, time series forecasting is crucial today. Making the wrong decisions at the wrong time may cause the business to close down quickly. This is due to lack of analysis, some entrepreneur’s not able to implement their ideas on right time which in turn results to losses. So, this study is aimed to provide benefits to Prakruthivanam to increase their customers and get profit in their business.

The study aims to accomplish the following things, or outcomes:

- To thoroughly analyse the dataset and create some visualizations.
- To clean data by detecting null values, replacing duplicates.
- To use exploratory data analysis techniques to gain understanding of the features that are present.
- To determine the impact of each feature on the target variable.
- To check Stationarity of data using Augmented Dickey Fuller test and plotted the rolling statistics.
- Splitting the dataset to obtain training and testing data
- Applying different Regression models and comparing the results of methods used.
- Choosing the most effective method for making future predictions.
- Forecasting future sales using Tableau analysis.

1.3 Overview of This Report

Machine learning techniques have benefited many industries, and there are numerous applications that use them. Machine learning applications have proven effective in a variety of disciplines, including healthcare, marketing, education, weather forecasting etc. Similar benefits of machine learning techniques may be applied in the retail industry to forecast future sales so that businesses can prepare for increased profitability and take timely action during slow periods.
The report is divided into ten sections. A brief synopsis of each chapter follows:

Chapter one contains an introduction to the problem, background information, the study's goals and objectives, and lastly an overview of the report that provides a summary of each section.

The second chapter includes a review of related work by other researchers in the field.

The third chapter is a brief explanation of theoretical topics of the techniques used for the research.

The requirements for implementing the model are presented in Chapter four. These requirements include data that we got from the client and functional requirements for this project.

Fifth chapter includes the design framework which follows the recommended methodologies.

The sixth chapter covers analysing and cleaning the dataset before applying the model. The Machine Learning methods are implemented in Chapter seven.

The results obtained for different models are compared and discussed in Chapter eight.

The ninth chapter covers project management, including the anticipated schedule, risk management and mitigation techniques, and the methods used to assess the models. Chapter ten contains the study's conclusion, future work and student reflections. Eleventh chapter contains References.

2 Literature Review

This section's goal is to expand upon the research done by numerous other academics in the field of machine learning in time series sales forecasting. It is essential to carry out literature review research so that current methodologies can be examined and how other author's analyses are done should be reviewed in order to motivate upcoming researchers to create a new, innovative, distinctive, and robust system. This section includes an introduction that lists categories that researchers have identified based on the volume and nature of the data. The techniques they employed and the findings they reached were noted and listed followed by the conclusion stating the models I inspired from previous researchers.

2.1 Introduction

The forecasting models were broadly categorised by the researchers into four major categories based on their experience of using data by applying various methodologies. They are Machine Learning Models, Deep Learning Models, Traditional Statistical Models and Hybrid Models. The below figure 2.1 shows the models that are appropriate for forecasting over specific time periods. (Ratnadip Adhikari, 2013)

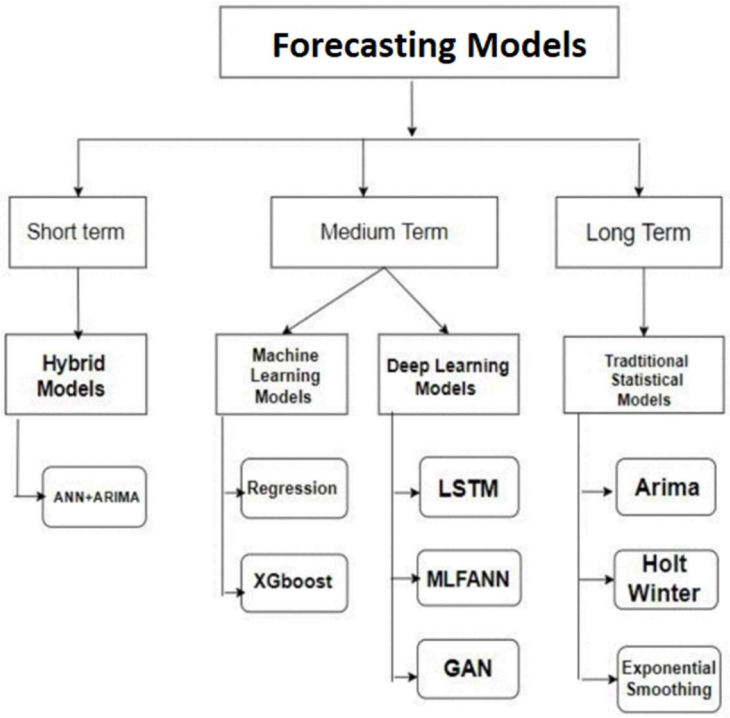


Figure 2.1 forecasting methodologies

Traditional Statistical Models

Traditional models rely on historical sales data provided by the organisation to forecast future product demand. Time series data is used to forecast these models.

Machine Learning Models

These techniques can be used to forecast demand by estimating the quantity of a product that will be bought over a specific time period. ML offers more accurate and consistent forecasting compared to conventional statistical models, reveals data's hidden patterns, improves adaptability to changes, and builds a stable system.

Deep Learning Models

Because they can control massive amounts of data and model non-linear data, deep learning approaches have recently gained a lot of popularity in many areas. Comparing deep learning methods to machine learning and conventional statistical models, it has been found that they are more accurate and have better prediction.

Hybrid Models

Since finding an effective forecasting technique is very difficult, we typically use a variety of models and compare the outcomes to determine which model is best. We're not sure, though, if it will be able to provide accurate results for a future time. So many researchers have suggested the use of hybrid models to address these problems.

2.2 Findings from Previous Research

- ➔ A study by Jamal Fattah and Latifa on models using Box–Jenkins time series approach were used in this journal. It is used for forecasting demand of food products reveals that this model is quite reliable and can be used for forecasting demand in food manufacturing industries, retail stores, and supply chains, as well as in stock management of these products. Several models were developed using historical demand data, and the best one was chosen using four performance criteria: SBC, AIC, standard error, and maximum likelihood. Out of all the applied models, ARIMA (1, 0, 1) gives the better results as the AIC value is lower compared with other results. (Jamal Fattah, 2018)
- ➔ A study on time series sales forecasting by Manisha and Vijayalakshmi shows that Instead of relying on a single model, combining them reduces the risk. The forecasted values are calculated by combining a consistent model for a series in the test phase (the last 24 points of a series). In this paper, the statistical methods like ARIMA and Hot-Winter models have been used to forecast these components. The results of combined model and Hot-Winter method were compared. As results

were better than Hot-Winter method, combined model was chosen as best model. (Mrs. Manisha Gahirwal)

- ➔ A study on sales Time Series Forecasting using machine learning models by Bohdan M. Pavlyshenko says that Sales forecasting is a regression problem rather than a time series problem. The author says that when compared to time series methods, regression approaches for sales forecasting can often produce better results. Sales show patterns, generalisation allows to obtain more precise results that are resistant to sales noise. We can get even more precise results if we have a short time frame. The effect of machine-learning generalisation allows to make predictions even with a small number of historical sales data, which is useful when launching a new product or store (Pavlyshenko, 2019).
- ➔ A Journal by C. CATAL, K. ECE, B. ARSLAN and A.AKBULUT studied the impact of Regression and Time Series Analysis methods on the sales forecasting problem was investigated. The experimental results show that regression techniques outperform and are more accurate than time series analysis techniques. With a coefficient of determination of 0.97, the boosted Decision Tree Regression algorithm was the best predictor for sales forecasting (C. CATAL, 2019).
- ➔ In this study a hybrid system using an ARIMA model and Artificial Neural Networks models is proposed by Zhang, Peter and it has led to more accurate results in the restaurant industry. The hybrid model makes use of ARIMA and ANN's unique advantages in linear and nonlinear modelling. The combination method can be an efficient way to boost forecasting performance for complex problems with both linear and nonlinear correlation structures. By using Hybrid models, the overfitting issue that is more closely associated to neural network models can be mitigated by fitting the ARIMA model to the data first (Zhang, 2003).

- ➔ In this study by Yihang Zhu, Yinglei Zhao proposes a different hybrid Holt-Winters and SVM model for spring onion seed demand forecasting. Because statistical-based models only use historical data, their findings are linear and seasonal in nature. The results of this investigation demonstrated that the suggested hybrid model beats statistically based models and other cutting-edge machine learning models and offers precise predictions (Yihang Zhu, 2019).

- ➔ A study on comparing Neural Network models and Box-Jenkins Model (SARIMA) by Ong Hong Choon and javin Lee Tze Chuin says that the models DSTL (Deseasonalized and Detrending model) and SARIMA are the two best models in

forecasting. Between these two models, DSTL outperforms SARIMA, indicating that neural networks can provide more accurate predictions than the SARIMA model on monthly weather report of malaysia (Hong Choon Ong, 2008).

➔ This paper on a study by Bahri and Vahidnia on time series forecasting using a hybrid model smoothing decomposition and machine learning techniques combined with LSTM and CNN. The forecast in this case was created by recombining the residuals and predicted intrinsic mode functions (IMFs). For all non-stationary and non-linear data, the Smoothing Decomposition technique-LSTM-CNN method obtained 99% accuracy (Mona Z. Bahri, 2022).

2.3 Conclusion

With the aid of this study on earlier research, I gained a lot of knowledge and was surprised by the implementation of combining different methods and techniques and creating new hybrid models which gives better results. So many researchers are experimenting on hybrid models and I was motivated to keep working on my project and will now try to focus on applying neural networks and machine learning techniques for forecasting sales in order to find the model that performs the best.

3 Methodology

3.1 Research Methodology

The standard technique known as agile methodology was adopted for this project's work because it enables data to be prioritised and models to be built in accordance with the project's business objectives and needs. Reproducibility is essential for success since data science processes are iterative. This is where these techniques may assure that a data science project delivers better results. For any methodology after completing all the necessary phases, the model should not be left untreated. Because this is mostly about data, some extra information may be required at any time, necessitating a return to prior phases and redoing from there. In addition, an appropriate update to the model should be made based on feedback, and deployment should continue until improved results are obtained. Because of these reasons I prefer to choose this methodology for my project work. The below figure 3.1 indicates the flow of this model. (Hamilton, 2022)

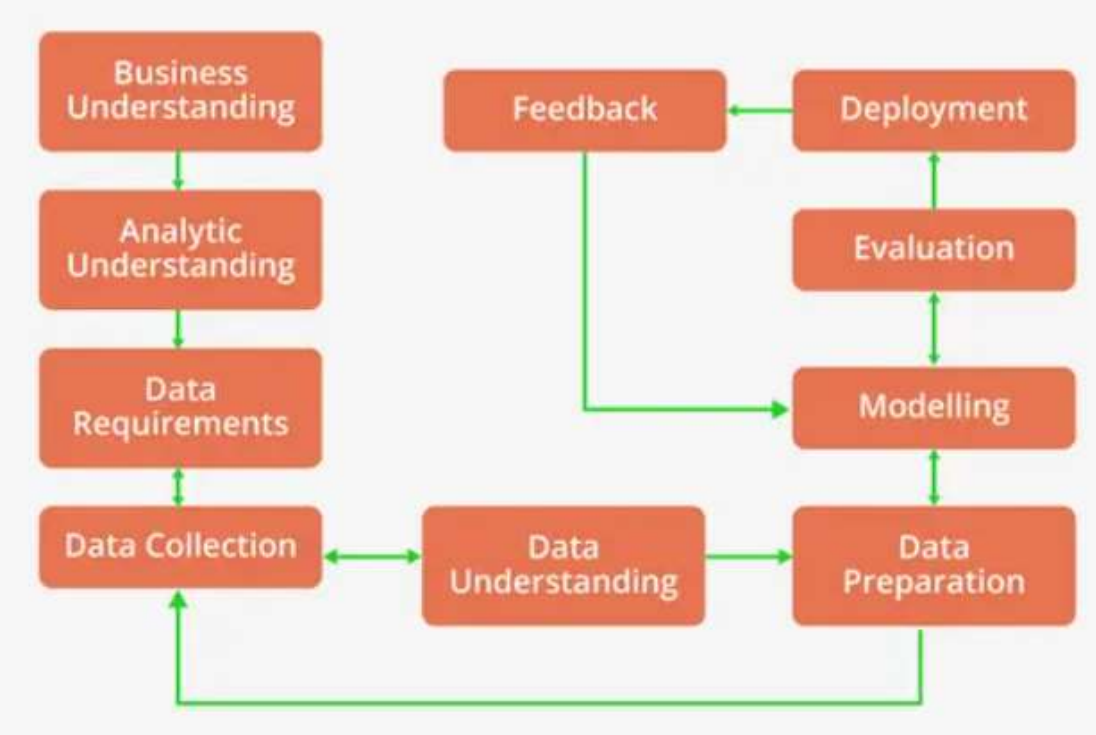


Figure 3.1 Flow of model

I can relate my project work with all the above stages of this methodology as first three stages Business Understanding, Analytic Understanding and Data Requirements as Understanding the problem statement, the next three stages Data Collection, Data Understanding and Data Preparation as Data collection, Exploratory Data Analysis and Pre-processing. The last stages Modelling, Evaluation, Deployment and Feedback as Implementation and calculating the performance of the models

3.2 Time Series Forecasting Techniques

With the assumption that future trends will resemble past trends, time series forecasting is a method for predicting future events by analysing historical patterns. In order to predict future values, models fitted to historical data are used in forecasting. For prediction problems with a time component, time series forecasting is necessary because it provides a statistical framework for making effective and efficient plans. From sales forecasting to weather forecasting, time series models have a wide range of applications. Time series models have been proven to be among the most successful forecasting methods in decisions involving a factor of uncertainty about the future. (Time series forecasting methods, n.d.)

The following are some examples of where we apply this method:

- Forecasting power demand to determine if another power producing unit should be built in the next five years.
- Forecasting call volumes in order to book workers at a call centre the following week.
- Forecasting inventory requirements in order to store goods to satisfy demand.
- Forecasting supply and demand to optimise logistics operations and other supply chain issues.
- Predicting breakdowns and service requirements to reduce downtime and maintain safety standards.
- Forecasting infection rates to improve disease control and outbreak prevention programmes.
- Predicting consumer ratings to forecasting product sales.

Below figure 3.2 shows the flow of time series model.

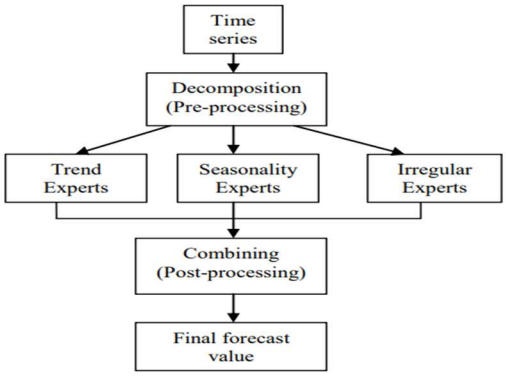


Figure 3.2 Time series flow

Before going into the models we need to get to know about some of the important terms of time series data. (Mrs. Manisha Gahirwal)

1. Trend – Data movement is either upward or downward over time
2. Seasonality – seasonal variations
3. Noise or irregularity – sudden spikes at random intervals
4. Cyclicity – similar patterns that repeats after large interval of time
5. Stationarity -- which means it must be consistent through time. In simple terms we can say that it should have constant mean, constant variance and no seasonality.

Before applying any statistical model to a Time Series, we need to check for stationarity. Stationarity check can be done in four ways.

1. Visual inspection - Plot the moving average or standard deviation to observe if it changes over time. It's a visual trick.
2. Global vs Local check – find the mean of entire time series data and then find mean of local time period and then see if its matching
3. Augmented Dickey-Fuller (ADF) Test - ADF Test gives us the p-value, lesser the p value says that the data is stationary.
4. KPPS Test. - It is opposite to ADF Test which also gives us the p-value, greater the p value says that the data is stationary.

By using one of the above tests we can get to know that the data is stationary or not. It the data is stationary we can continue with the model implementation, If not, we have to convert it to a stationary series.

There are many methods to convert non-stationary data to stationary data

1. Differencing: subtracting the values of time series with values of previous time period. First order differencing y (t)-y (t-1)

We can also go to 2nd order, 3rd order which ever convert our data to stationary

2. Log operations to smooth exponential curves (Log (exp(x)=x)
3. Seasonal differencing y (t)-y (t-N) – subtract each value from the value in the previous time cycle.

After completing all the necessary checks and conversions, we can implement our machine learning models on out data. Below are some of the models I implemented in this research project.

3.2.1 Time Series Regression

Based on independent variables, regression models a target prediction value. Regression develops a goal prediction value based on independent variables.

3.2.1.1 Linear Regression

Based on a given independent variable, linear regression is used to forecast the value of a dependent variable (y) (x). This regression strategy thus establishes a linear relationship between the input variables x and y. (output). In this kind of analysis, the coefficients of a linear equation with one or more independent variables are chosen because they are the ones that most accurately predict the value of the dependent variable. The differences between the expected and actual output values are minimised by linear regression by fitting a straight line or surface. (The linear model, n.d.)

Linear Regression can be represented as Y = m X + c as shown in the below figure 3.3.

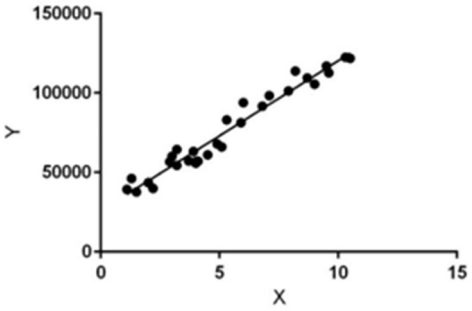


Figure 3.3 graphical representation of Linear Regression

3.2.1.2 Random Forest Regression

A supervised learning technique "Random Forest Regression" performs regression using the ensemble learning approach. A more accurate forecast is produced by the ensemble learning method than by using a single model by combining predictions from various machine learning algorithms. (Random Forest Regression in Python, n.d.)

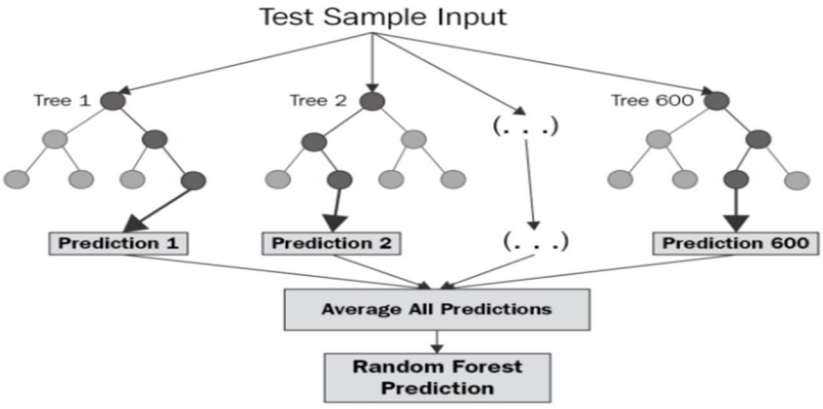


Figure 3.4 pictorial representation of Random Forest Regression

Figure 3.4 above depicts a Random Forest's structure. As you can see, the trees are running parallel to one another without making contact. The first step is to randomly choose k data points from the practise set. On the basis of these k data points, construct a decision tree. Repeat this process until you have created N trees. Assign the new data point to the average of all predicted y values by using each of your N-tree trees to forecast the value of y for the relevant new data point. An efficient and precise regression model is a random forest model. Numerous issues, including those with non-linear relationships, are frequently resolved by it.

3.2.1.3 Decision Tree Regression

Decision trees are used to solve classification and regression problems. In the regression problem, they start at the root of the tree and proceed through splits based on variable outcomes until they reach a leaf node and provide the conclusion. They flow visually like trees, hence the name. A model with a tree structure is trained using decision tree regression to forecast data in the future and produce meaningful continuous output after evaluating the attributes of an object. Continuous output means that the output or result isn't discrete, i.e., it's not just a discrete collection of well-known numbers or values.

3.2.1.4 XGBoost Regression

A powerful and effective method for creating supervised regression models is XGBoost. Knowing about its (XGBoost) objective function and base learners allows one to infer the veracity of this claim. A regularisation term and a loss function are two parts of the objective function. It gives data on the difference between actual and predicted values, or how far the model's predictions differ from the actual values. Linear and Logistics are the two most popular loss functions in XGBoost for regression and binary classification, respectively. XGBoost is one of the ensemble learning techniques, which entails training and combining various individual models (also referred to as base learners) to produce a single prediction. In order for bad predictions to cancel out and better ones to add up to final good predictions, XGBoost anticipates having base learners who are uniformly bad at the rest. (XGBoost for Regression, n.d.)

3.2.2 ARIMA (Auto Regressive Integrated Moving Average) Model

Auto Regressive Integrated Moving Average Model contains four models, AR (Auto Regressive), MA (Moving Average), ARMA (Auto Regressive Moving Average), and ARIMA (Auto Regressive Integrated Moving Average) models.

3.2.2.1 AR (Auto Regressive) Model

Regression: used to predict continuous value of an item based on certain parameters.

Auto: Uses its own past values to predict future values.

Below Equations shows the 1st and 2nd order AR model

$Y_t = C_1 y_{t-1} + C_2$ → AR (1): 1st order Auto Regression

$Y_t = C_1 + C_2 y_{t-1} + C_3 y_{t-2}$ → AR (2): 2nd order Auto Regression

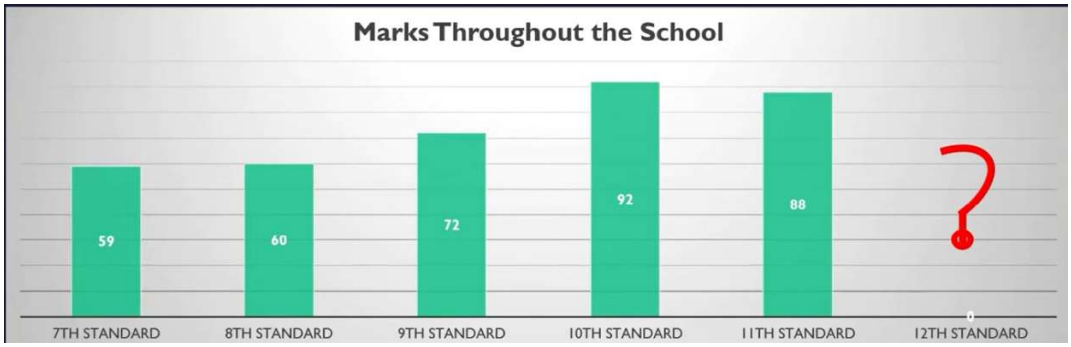


Figure 3.5 example of prediction using auto regression

Using the data from above figure 3.5, we can predict 12th standard marks using AR model of order 1, 2, 3, etc. with the below equations as follows

12th marks = C1 + C2*(11th Marks) + Error (we are not 100% correct) – 1st order AR model

12th marks = C1 + C2 *(11th Marks) + C3 *(10th Marks) + Error (we are not 100% correct) – 2nd order AR model

12th marks = C1 + C2 *(11th Marks) + C3 *(10th Marks) + C4 * (9th Marks) + Error (we are not 100% correct) - 3rd order AR model

We can clearly see that there is a need to establish how the value of the variable is related with its previous time period. For that we need to use PACF to find the order for AR model.

Partial Auto-Correlation Function: Only Direct effect of values in previous time Lags with current time period (8th marks directly impact 12th marks) (TIME SERIES AR,MA, n.d.)c

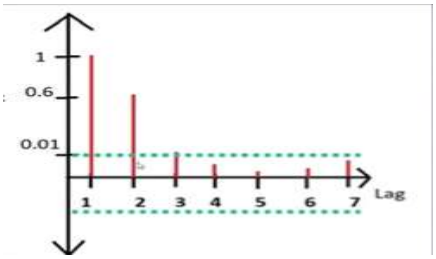


Figure 3.6 PACF plot

$Y_t = C_1 + C_2 y_{t-1} + C_3 y_{t-2}$ → AR (2): 2nd order Auto Regression

In the above figure 3.6 lag 1 and 2 are towards 1 in y axis above the green line and others 3 to 7 lags are nearly 0 so we can say that this is a 2nd order AR model.

3.2.2.2 MA (Moving Average) Model

Models that predict future values of a time series using the past errors. Below Equations shows the 1st, 2nd and qth order MA model

MA: $y_t = \mu + C_1 e_{t-1} + e_t \rightarrow$ MA (1)

MA (2) Model: $y_t = \mu + C_1 e_{t-1} + C_2 e_{t-2} + e_t \rightarrow$ MA (2)

MA (q) Model: $y_t = \mu + C_1 e_{t-1} + \ldots + C_q e_{t-q} + e_t \rightarrow$ MA (q)

In order to choose which model we need to use we need to use ACF plots.

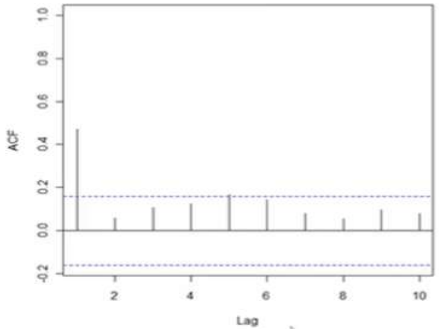


Figure 3.7 ACF plot

In order to choose which model we need to use we need to use ACF plots. Above figure 3.7 shows an ACF plot, Moving average of order 1 should be used. As the 1st value on x axis has value 0.5 which is very good. For rest of the time periods the correlation is very low that is in between the blue dotted lines which is in error band.

3.2.2.3 ARMA (Auto Regressive Moving Average) Model

ARMA is a combination of AR and MA models.

AR: uses past values to make future predictions

$y_t = C_1 + C_2 y_{t-1} + e_t \rightarrow$ AR (1)

MA: uses past errors to make future predictions

$y_t = C_1 + \Theta_1 e_{t-1} + e_t \rightarrow$ MA (1)

ARMA: $y_t = C_1 + C_2 y_{t-1} + \Theta_1 e_{t-1} + e_t \rightarrow$ ARMA (1, 1) $\rightarrow$ ARMA (p, q)

ACF and PACF plots: Measure the correlation between current time period and previous time lags

ACF: Measure direct and indirect effect of previous time lags on current value , used to find order of moving average model.

PACF: Measure only direct effect of previous time lags on current value (it elliminates the indirect effects) , used to find order of auto regression model. Below figure 3.8 shows the ARMA (2,3)

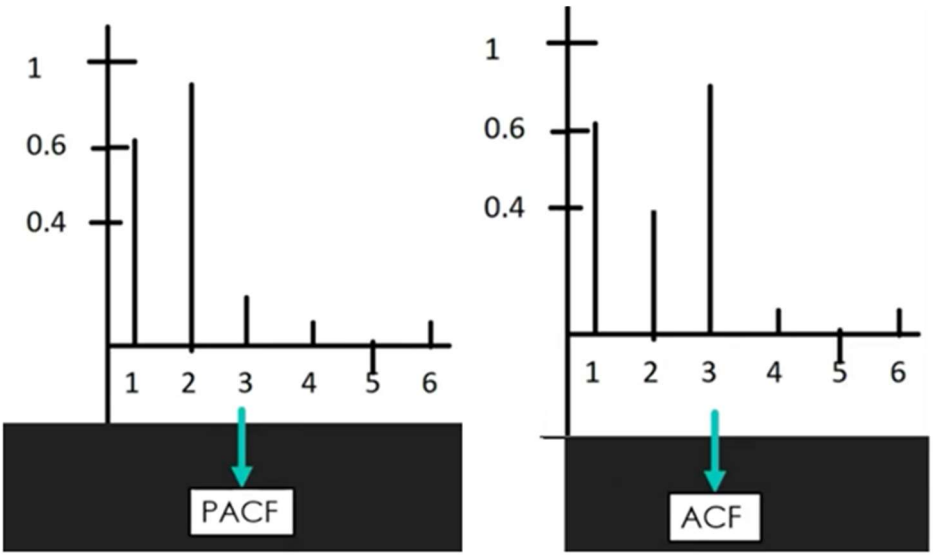


Figure 3.8 ACF and PACF plot

3.2.2.4 ARIMA (Auto Regressive Integrated Moving Average) Model

ARIMA is a time series model designed to understand auto-correlations in time series data.It is effective for short-term forecasting and may be used to generate forecasted values for particular periods that show good outcomes for demand, sales, planning, and production. ARIMA is composed of 3 terms(Auto-Regression p + Integrated d +Moving-Average q)

Auto Regression and Moving Average models are explained above. Here we combine the AR and MA models with the differencing component d called integration. If your dataset is stationary, we can use d value as Zero. If your data set is not stationary, you will almost always need to run a difference operation to make it stationary. Simply we can say that the number of times your time series data must be differed in order to attain stationarity. (ARIMA, n.d.)

p – Order of AR model

d – Order of differencing to get stationary Series – Differencing operation to convert non-stationary series(Increasing mean) to stationary (constant mean,average)

q – Order of MA model

3.2.2.4 SARIMA (Seasonal Auto Regressive Integrated Moving Average) Model

The impact of seasonality in time series has not yet been taken into account. This behaviour is undoubtedly prevalent in many situations, such as shop sales or the overall number of passengers on aeroplanes.

A seasonal ARIMA model or SARIMA is as follows: SARIMA (p, d, q) (P, D, Q) m

As you can see, we added P, D, and Q to the time series' seasonal component. Since they involve seasonal period backshifts, they share the same terms as the non-seasonal

components. The number m in the formula above represents the periods or the year's number of observations. If we were looking at annual data, m would be equal to 12. An AR and MA model's seasonal component can be inferred from the PACF and ACF plots. The modelling process is the same as it is for ARIMA models that are not seasonal. In this instance, we only need to take the extra parameters into account. (SARIMA Using Python – Forecast Seasonal Data, n.d.)

3.2.3 Recurrent Neural Networks (LSTM's, GRU's, Bidirectional LSTM's)

A recurrent neural network (ANN) is a type of artificial neural network (ANN) that is used to perform prediction on sequential or time-series data.

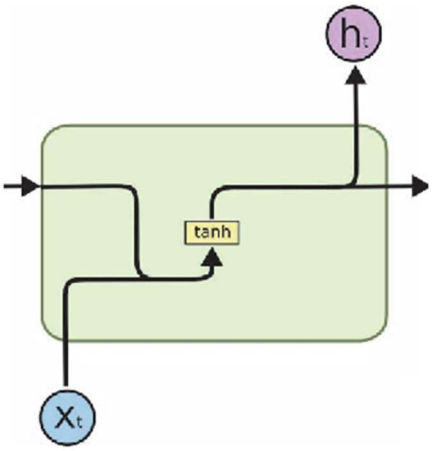


Figure 3.9 RNN Architecture

The above figure 3.9 is a single unit of RNN architecture, it accepts input from the previous step and current state X_t and incorporates Tanh as an activation function; the activation function can change the activation function. It works well for recent information, however it loses information when extended sequences or words are utilised. This problem is known as the vanishing gradients problem. To address this issue, specialised versions of RNN like as LSTM, GTU etc. have been developed. (Petneh´azi)

3.2.3.1 Long Short Term Memory (LSTM)

LSTM is a kind of RNN that can learn long-term sequences.

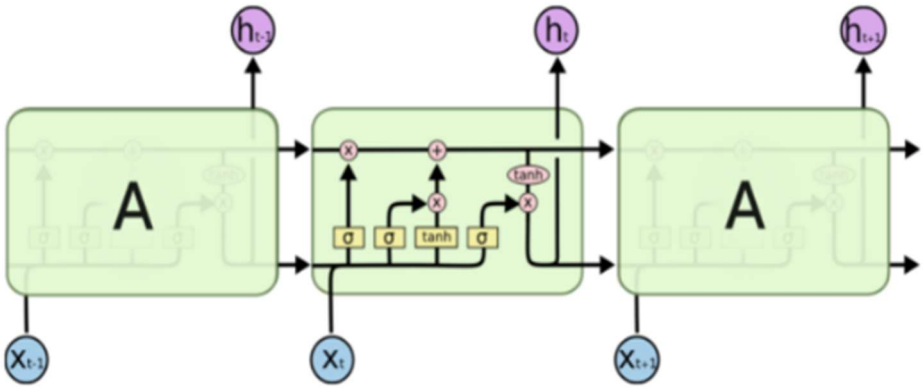


Figure 3.10 LSTM Architecture

As shown in Figure 3.10 above, it gathers data from three different states simultaneously: the short term memory from the previous cell, the long term memory, and finally the current input state. The Input Gate, Forget Gate, and Output Gate are the three gates used by LSTM. What information is stored in long-term memory is determined by the input gate. Whether or not to retain information from long-term memory depends on the forget. The output gate will create new short term memory and send it to the cell in the following time step using the current input, previous short term memory, and newly computed long term memory. (Has¸im Sak)

3.2.3.2 Gated Recurrent Unit or GRU

The RNN and the Gated Recurrent Unit, or GRU, use the same workflow, but the GRU has different operations and gates connected to each unit. As depicted in figure 3.11, GRU contains the gate operating techniques Update gate and Reset gate to address the challenge faced by standard RNN.

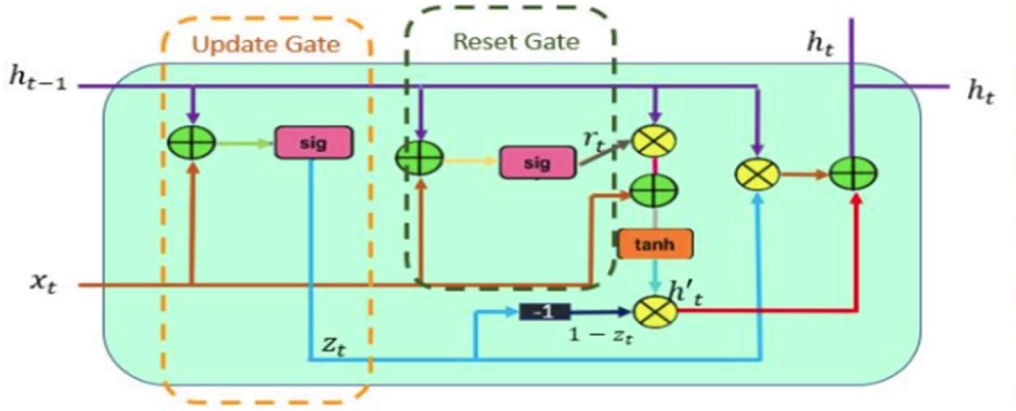


Figure 3.11 GRU Architecture

The update gate is in charge of determining how much prior information must be passed along to the next state. The reset gate is employed in the model to determine how much prior information may be ignored in other words, it determines whether the previous cell state is significant or not. (LENDAVE, n.d.)

3.2.3.3 Bidirectional LSTM

Bidirectional long-short term memory (Bidirectional LSTM) is the process of allowing any neural network to have sequence information in both directions—backwards (future to past) or forward (past to future). Bi-directional input can be made to flow in both directions in order to preserve both the past and the future information. Figure 3.12 below shows the information flow from the backward and forward layers. When tasks requiring sequence to sequence are necessary, BI-LSTM is frequently used. This kind of network can be used for forecasting models, text classification, and speech recognition. (bilstmLayer, n.d.)

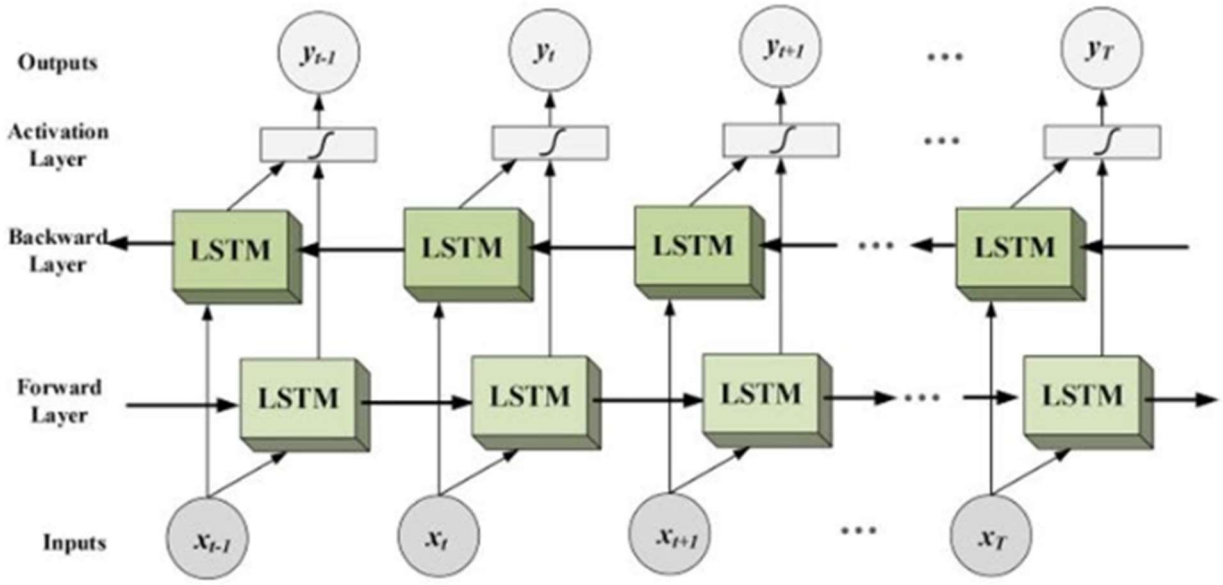


Figure 3.12 Bidirectional LSTM Architecture

4 Requirements

4.1 Data Requirements

Since my project involves forecasting sales at Prakruthivanam Store, I had the opportunity to meet with the management online. They gave me a thorough explanation of their store, the various product categories they carry, the various brands with which they have dealerships, and other information. The inventory's movement both in and out is monitored using the Nukkad Shops Billing System. The customers who purchased the goods were billed in the same system, and the stock they are receiving will be entered into the system by creating a GRN (Goods Receiving Note). As a result, the goods added by the GRN were automatically decreased if any purchases were made. The Nukkad shops billing system provided the information I used for this project, which was then pulled and sent directly to my email address.

The data contains three reports. They are bill-wise sales report, daily sales report, product wise sales report.

Bill-wise sales report contains the data columns like Time, Date, Bill No, Consumer Name, Taxable Amount 0%, Taxable Amount 5%, Taxable Amount 12%, Taxable Amount 18%, CGST, SGST, Total tax amount, Total Bill Value, Total No of Items Purchased, Payment Type, Online/Offline. All these columns data was recorded for each and every bill. It contains 51737 rows and 15 columns.

Daily Sales Report contains the data columns like Date, Total Bills, Invoice Range, Total Taxable amount, Taxable Amount 0%, Taxable Amount 5%, Taxable Amount 12%, Taxable Amount 18%, Total Amount, Cash, Card, Other Modes, Amount Return, Amount Return to Credit, Total Items Sold, Total Quantity, Returned Items Count, and Returned quantity. Based on the Bill wise sales report, daily sales report was generated once their billing day session was closed. It contains 1361 rows and 19 columns.

Product wise sales Report contains the data columns like Name, Brand, UOM, Category, Sub-category, Tax Code, Quantity Sold, No of Bills, Taxable Amount, Revenue, MRP, SP/WP/SPLP/OP, CP. This report was also generated by the system with respect to the product name and its quantity purchased in each bill. It contains 8588 rows and 13 columns.

The below table 4.1 provides a summary of all the Columns and the Column types and description of Bill-wise sales report:

Column Name	Column Description	Data Type
Time	Bill generation time/ purchase time	object
Date	Date of the particular bill	object
Bill No	Bill number , which is a unique value generated in a series	object
Consumer Name	Which is entered by the staff at the time of billing	object
Taxable Amount 0%	If a customer purchase any non-taxable items, that total amount of items will be shown here.	float64
Taxable Amount 5%	If a customer purchase any 5% tax items that total value of items will be shown here.	float64
Taxable Amount 12%	If a customer purchase any 12% tax items that total value of items will be shown here.	float64
Taxable Amount 18%	If a customer purchase any 18% tax items that total value of items will be shown here.	float64
CGST	Central Goods Service Tax	float64
SGST	State Goods Service Tax	float64
Total Tax Amount	Total tax amount of all taxable goods	float64
Total Bill Value	The total amount of all the items purchased	float64
Total No of Items Purchased	Count of the items in each bill	Integer
Payment Type	Mode of payment like cash/card/paytm/google pay/Redeem Points	Object
Online/Offline	Whether the customer payment is through cash is offline, if paid	object

	any other electronic payments is online, if any purchase returns then its refund.	

Table 4.1 Bill-wise Sales Report column Description

The below table 4.2 provides a summary of all the Columns and the Column types and description of Daily sales report:

Column Name	Column Description	Data Type
Date	Specific date	object
Total Bills	No of Bills Generated on a particular date	Int64
Invoice Range	Invoice number of First and last bill generated on that particular day	Object
Total Taxable Amount	Total Amount of all the items which is taxable	Float64
Total Tax amount	Total tax need to be paid to government of the above taxable amount	Float64
Taxable Amount 0%	Total non-taxable amount of all the purchases in a particular day	Float64
Taxable Amount 5%	Total amount of 5% taxable of all the purchases in a particular day	Float64
Taxable Amount 12%	Total amount of 12% taxable of all the purchases in a particular day	Float64
Taxable Amount 18%	Total amount of 18% taxable of all the purchases in a particular day	Float64
Total Amount	Total sale amount in a particular day	Float64
Cash	Total amount received through cash	Float64

Card	Total amount received through card	Float64
Other Modes	Total amount received through other payment modes like google pay, paytm, redeem points etc.	Int64
Amount Return	Total amount returned for any purchase returns	Float64
Amount Returned to Credit	If amount is returned for any particular customer in credit, that means physical amount is not returned, it's added in the system under credit section and the customer can use that at the time of future purchases.	Int64
Total Items Sold	Total no of products sold	Int64
Total Quantity	quantity of all products sold	Float64
Returned Items Count	Count of no of products returned	Int64
Returned Quantity	Quantity of all returned products	Float64

Table 4.2 Daily Sales Report column Description

The below table 4.3 provides a summary of all the Columns and the Column types and description of Product Wise sales report:

Column Name	Column Description	Data Type
Name	Name of the Product	Object
Brand	Brand Name to which the product belongs to	Object
UOM	Units Of Measure	Object
Category	Category of the product	Object
Sub-category	Sub-category of the product	Object

Tax Code	Tax Code to which the product belongs to. There are Different tax codes like 0,5,12 and 18	Integer
Quantity Sold	Total quantity of items sold till date	Float64
No of Bills	Count of bills in which this item is sold	Int64
Taxable Amount	Total taxable amount of that item	Float64
Revenue	Total amount of all the quantity sold	Float64
MRP	Manufacturers recommended price of that particular product	Int64
SP	Selling Price	Float64
CP	Cost Price	Float64

Table 4.3 Product wise sales Report column Description

4.2 Functional Requirements

The functional requirement for this project are mentioned below:

1. The data must be pre-processed and all the features must be extracted to get better results.
2. Using Jupiter notebook which was installed using Anaconda. Also installed Tableau Desktop to implement the sales forecasting using the sales reports.
3. All the libraries required to implement the model must be imported.
4. Accuracy must be calculated for all the models and compare the results.
5. The code must be error free and can be reproducible.

5 Analysis

I am using all the three reports and done some basic Exploratory Analysis and created some visualisations. So first step is to import all the necessary libraries such as Numpy, pandas, Matplotlib, Seaborn, Sklearn, statsmodels, pmdarima, pandas.plotting, math libraries for analysing the data, model construction and calculating results.

Firstly I am using pandas library to read all the three input files and storing the excel file into dataframes so that we can use that pandas dataframe throughout the project. The below figure 5.1 shows the first and last five records of the bill wise sales report.

```
In [5]: #read the dataset
        Billwise_sales = pd.read_csv("Billwise-Sales-Report-2019-2022.csv")
        Billwise_sales.head(5)
```

Out[5]:

	TIME	DATE	BILL_NO	CONSUMER_NAME	Taxable_Amount_0%	Taxable_Amount_5%	Taxable_Amount_12%	Taxable_Amount_18%	CGST	SG
0	09:12pm	30/09/2022	AA/2223/7106	Praveen Kphb	0.0	0.000	0.000	9745.763	877.119	877.
1	09:11pm	30/09/2022	AA/2223/7105	Praveen Kphb	225.0	219.048	0.000	177.966	21.493	21.
2	08:43pm	30/09/2022	AA/2223/7104	Guest	0.0	0.000	0.000	10169.492	915.254	915.
3	08:15pm	30/09/2022	AA/2223/7103	Venkata Ratnam 6th Phase	0.0	791.398	0.000	75.451	26.576	26.
4	08:08pm	30/09/2022	AA/2223/7102	Raghavendra Vivekananda Nagar	0.0	380.952	410.714	0.000	34.167	34.

```
In [6]: Billwise_sales.tail(5)
```

Out[6]:

	TIME	DATE	BILL_NO	CONSUMER_NAME	Taxable_Amount_0%	Taxable_Amount_5%	Taxable_Amount_12%	Taxable_Amount_18%	CGST	SGST
51732	01:55pm	01/01/2019	A-A000128	ravi l	205.0	0.0	0.0	0.0	0.0	0.0
51733	01:48pm	01/01/2019	A-A000127	balaji	880.0	0.0	0.0	0.0	0.0	0.0
51734	01:40pm	01/01/2019	A-A000126	ramu	225.0	0.0	0.0	0.0	0.0	0.0
51735	01:37pm	01/01/2019	A-A000125	ratna	390.0	0.0	0.0	0.0	0.0	0.0
51736	01:29pm	01/01/2019	A-A000124	Ramesh Hydernagar	770.0	0.0	0.0	0.0	0.0	0.0

Figure 5.1 head and tail of bill wise sales report

The below figure 5.2 shows the first and last five records of the product wise sales report.

```
#read the dataset
product_sales = pd.read_csv("product wise sales report 2019-2022.csv")
product_sales.head(5)
```

	NAME	BRAND	UOM	CATEGORY	SUB CATEGORY	TAX CODE	QTY SOLD	NO. OF BILLS	TAXABLE AMOUNT	REVENUE	MRP	SP/WP/SPLP/OP	CP
0	Rice Nippattu 100g	Prakruthivanam	1 grm	Snacks	Home Made	5	432.0	329	24685.71	25920.0	60	60.0	42.0
1	Real Aam Rs.5	Go Desi	1 pc	Miscellaneous	Miscellaneous	5	7098.0	1312	33800.00	35490.0	5	5.0	4.0
2	Coconut Oil 1lt	Prakruthivanam	1 pc	Grocery & Staples	Cooking Oil	5	636.0	499	327085.71	343440.0	540	540.0	430.0
3	Kiwi - 250g	Dry Fruits	1 grm	Miscellaneous	Miscellaneous	0	16.0	15	2288.00	2288.0	143	143.0	110.0
4	Jaggery Powder-500g	Prakruthivanam	1 kg	Miscellaneous	Miscellaneous	0	61.0	37	3965.00	3965.0	65	65.0	45.0

```
product_sales.tail(5)
```

	NAME	BRAND	UOM	CATEGORY	SUB CATEGORY	TAX CODE	QTY SOLD	NO. OF BILLS	TAXABLE AMOUNT	REVENUE	MRP	SP/WP/SPLP/OP	CP
8583	Tamarind-500 Gms-pvs	Prakruthivanam	1 pc	Miscellaneous	Miscellaneous	0	1.0	1	89.3	89.3	95	89.3	0.0
8584	Shampoo -600 Ml -pvs	Prakruthivanam	1 pc	Miscellaneous	Miscellaneous	0	5.0	1	1363.0	1363.0	290	272.6	0.0
8585	Udara Rawa - 500 Gms - Pvs	Prakruthivanam	1 pc	Miscellaneous	Miscellaneous	0	1.0	1	80.0	80.0	85	80.0	0.0
8586	Udara Flour - 500 Gms - Pvs	Prakruthivanam	1 pc	Miscellaneous	Miscellaneous	0	1.0	1	85.0	85.0	80	85.0	0.0
8587	Arikalu (kodo) Millet - 1kg - Pvs	Prakruthivanam	1 pc	Miscellaneous	Miscellaneous	0	2.0	2	380.0	380.0	160	190.0	0.0

Figure 5.2 head and tail of product wise sales report

The below figure 5.3 shows the first and last five records of the daily sales report.

```
daily_sales = pd.read_csv("daily_sale_report_2019-2022.csv")
daily_sales.head(5)
```

	DATE	TOTAL BILLS	INVOICE RANGE	TOTAL TAXABLE AMOUNT	TOTAL TAX AMOUNT	Taxable Amount 0%	Taxable Amount 5%	Taxable Amount 12%	Taxable Amount 18%	TOTAL AMOUNT	CASH	CARD	OTHER MODES	AMOUNT RETURN	AMOUNT RETURN TO CREDIT	TOTAL ITEMS SOLD
0	30-Sep-22	46	AA/2223/7062, AA/2223/7106	73200.11	9962.67	6108.72	23686.13	2235.86	47278.12	89272.0	6816.0	1160.0	81870	574.0	0	164
1	29-Sep-22	35	AA/2223/7027, AA/2223/7061	26105.68	3436.52	6879.00	9180.00	1151.79	15773.90	36421.0	2735.0	2880.0	31446	640.0	0	105
2	28-Sep-22	44	AA/2223/6983, AA/2223/7026	43762.30	4439.50	7969.76	25776.88	1445.36	16540.06	56172.0	17160.0	11502.0	28670	1160.0	0	168
3	27-Sep-22	33	AA/2223/6951, AA/2223/6982	23461.43	1909.57	3638.00	16638.10	2508.93	4314.41	29009.0	5185.0	5402.0	18942	520.0	0	120
4	26-Sep-22	27	AA/2223/6924, AA/2223/6950	23631.84	1660.16	1553.00	19357.14	1285.71	2988.98	26845.0	6080.0	0.0	21325	560.0	0	108

```
daily_sales.tail(5)
```

DATE	TOTAL BILLS	INVOICE RANGE	TOTAL TAXABLE AMOUNT	TOTAL TAX AMOUNT	Taxable Amount 0%	Taxable Amount 5%	Taxable Amount 12%	Taxable Amount 18%	TOTAL AMOUNT	CASH	CARD	OTHER MODES	AMOUNT RETURN	AMOUNT RETURN TO CREDIT	TOTAL ITEMS SOLD	TOTAL QTY	RE
05-Jan-19	62	A-A000386, A-A000446	0.0	0.0	31980.0	0.0	0.0	0.0	31980.0	13485.0	18720.0	280	505.0	0	228	266.0	
04-Jan-19	115	A-A000271, A-A000385	0.0	0.0	45510.0	0.0	0.0	0.0	45510.0	39360.0	5890.0	330	70.0	0	348	377.0	
03-Jan-19	39	A-A000232, A-A000270	0.0	0.0	33920.5	0.0	0.0	0.0	33921.0	9275.0	24516.0	135	5.0	0	163	224.0	
02-Jan-19	47	A-A000186, A-A000231	0.0	0.0	36140.0	0.0	0.0	0.0	36140.0	22115.0	14335.0	0	310.0	0	246	292.0	
01-Jan-19	63	A-A000124, A-A000185	0.0	0.0	37420.0	0.0	0.0	0.0	37420.0	9305.0	26435.0	2040	360.0	0	297	329.0	

Figure 5.3 head and tail of daily sales report

By seeing the columns and information, I can see that the bill wise report and daily sales report have the simillar kind of information.so from now, I am using billwise report to continue my further analysis.

In order to proceed with the analysis, the data should be clean without any null values. We can check the null values by using the isnull() function. If there are few missing values we can ignore them and drop the rows. If there are more missing values we can take the mean value and replace the missing rows with that value and proceed with the analysis. The below figure 5.4 shows the count of missing values from the three reports. As we have zero null values, we can proceed with our analysis.

```
#Look for missing values
Billwise_sales.isnull().sum()
```

```
TIME                          0
DATE                          0
BILL NO                       0
CONSUMER NAME                 0
Taxable Amount 0%             0
Taxable Amount 5%             0
Taxable Amount 12%            0
Taxable Amount 18%            0
CGST                          0
SGST                          0
TOTAL TAX AMOUNT              0
TOTAL BILL VALUE              0
TOTAL NO OF ITEMS PURCHASED   0
PAYMENT TYPE                  0
ONLINE/OFFLINE                0
dtype: int64
```

```
product_sales.isnull().sum()
```

```
NAME              0
BRAND             0
UOM               0
CATEGORY          0
SUB CATEGORY      0
TAX CODE          0
QTY SOLD          0
NO. OF BILLS      0
TAXABLE AMOUNT    0
REVENUE           0
MRP               0
SP/WP/SPLP/OP     0
CP                0
dtype: int64
```

Figure 5.4 checking for null values

5.1 Billwise sales report Exploratory Data Analysis

We can check the datatypes and columns information using info() function.Below figure 5.5 shows the information of billwise and daily sales information. We can see that the data type of date and time column are object. They are stored as sting. We need to convert date and time to datetime64[ns]

```
Billwise_sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51817 entries, 0 to 51816
Data columns (total 15 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   TIME                        51817 non-null  object
 1   DATE                        51817 non-null  object
 2   BILL NO                     51817 non-null  object
 3   CONSUMER NAME               51817 non-null  object
 4   Taxable Amount 0%           51817 non-null  float64
 5   Taxable Amount 5%           51817 non-null  float64
 6   Taxable Amount 12%          51817 non-null  float64
 7   Taxable Amount 18%          51817 non-null  float64
 8   CGST                        51817 non-null  float64
 9   SGST                        51817 non-null  float64
 10  TOTAL TAX AMOUNT            51817 non-null  float64
 11  TOTAL BILL VALUE            51817 non-null  float64
 12  TOTAL NO OF ITEMS PURCHASED 51817 non-null  int64
 13  PAYMENT TYPE                51817 non-null  object
 14  ONLINE/OFFLINE              51817 non-null  object
dtypes: float64(8), int64(1), object(6)
memory usage: 5.9+ MB
```

Figure 5.5 bill wise sales data information

Using to_datetime function I am converting the date and time columns to datetime format and adding two columns as shown in below figure 5.6. One column is month_year which I have extracted month and year from date, and the other one is Hour column which I extracted hour from the time. Also converted time in 24hours format.

```
Billwise_sales['DATE'] = pd.to_datetime(Billwise_sales['DATE'], format = "%d/%m/%Y")
Billwise_sales['TIME'] = pd.to_datetime(Billwise_sales['TIME'],format="%I:%M%p").dt.strftime('%H:%M')
Billwise_sales['month_year'] = pd.to_datetime(Billwise_sales['DATE']).dt.to_period('M')
Billwise_sales['Hour'] = pd.to_datetime(Billwise_sales['TIME']).dt.hour
Billwise_sales
```

	TIME	DATE	BILL_NO	CONSUMER_NAME	Taxable_Amount_0%	Taxable_Amount_5%	Taxable_Amount_12%	Taxable_Amount_18%	CGST	SGST
0	21:12	2022-09-30	AA/2223/7106	Praveen Kphb	0.0	0.000	0.000	9745.763	877.119	877.119
1	21:11	2022-09-30	AA/2223/7105	Praveen Kphb	225.0	219.048	0.000	177.966	21.493	21.493
2	20:43	2022-09-30	AA/2223/7104	Guest	0.0	0.000	0.000	10169.492	915.254	915.254
3	20:15	2022-09-30	AA/2223/7103	Venkata Ratnam 6th Phase	0.0	791.398	0.000	75.451	26.576	26.576
4	20:08	2022-09-30	AA/2223/7102	Raghavendra Vivekananda Nagar	0.0	380.952	410.714	0.000	34.167	34.167
...	...	...	...	...	...	...	...	...	...	...
51732	13:55	2019-01-01	A-A000128	ravi l	205.0	0.000	0.000	0.000	0.000	0.000
51733	13:48	2019-01-01	A-A000127	balaji	880.0	0.000	0.000	0.000	0.000	0.000
51734	13:40	2019-01-01	A-A000126	ramu	225.0	0.000	0.000	0.000	0.000	0.000
51735	13:37	2019-01-01	A-A000125	ratna	390.0	0.000	0.000	0.000	0.000	0.000
51736	13:29	2019-01-01	A-A000124	Ramesh Hydernagar	770.0	0.000	0.000	0.000	0.000	0.000

51737 rows × 17 columns

Figure 5.6 Date column conversion

The below figure 5.7 shows the dataframe after adding the month_year and hour columns.

Billwise_sales

unt_18%	CGST	SGST	TOTAL_TAX_AMOUNT	TOTAL_BILL_VALUE	TOTAL_NOOF_ITEMS_PURCHASED	PAYMENT_TYPE	ONLINE/OFFLINE	month_year	Hour
9745.763	877.119	877.119	1754.237	11500.0	1	Paytm	Online	2022-09	21
177.966	21.493	21.493	42.986	665.0	6	Paytm	Online	2022-09	21
0169.492	915.254	915.254	1830.508	12000.0	1	Paytm	Online	2022-09	20
75.451	26.576	26.576	53.151	920.0	5	Cash	Offline	2022-09	20
0.000	34.167	34.167	68.333	860.0	7	Card	Online	2022-09	20

Figure 5.7 bill wise sales dataframe after adding new columns

The below figure 5.8 shows the count of unique values of all the columns in our dataframe.

```
Billwise_sales.nunique()
```

```
TIME                          967
DATE                         1361
BILL_NO                     51737
CONSUMER_NAME                6921
Taxable_Amount_0%            2237
Taxable_Amount_5%            2490
Taxable_Amount_12%            742
Taxable_Amount_18%           1093
CGST                        10714
SGST                        10714
TOTAL_TAX_AMOUNT            10629
TOTAL_BILL_VALUE             3703
TOTAL_NOOF_ITEMS_PURCHASED     65
PAYMENT_TYPE                    9
ONLINE/OFFLINE                  3
month_year                     48
Hour                           20
dtype: int64
```

Figure 5.8 count of unique values

The below figure 5.9 shows the payment_Type column and its values with their count. By seeing these values we can say that out of all the mode of payments, most of the customers (around 21K bills) are using cash transactions, then around 15K bills, customers use paytm, for 14k bills, customers use card for paying their bills.

```
Billwise_sales['PAYMENT_TYPE'].value_counts()
```

```
Cash             21903
Paytm            15294
Card             14007
NEFT               177
Credit             148
Google Pay         100
Redeem Points       71
Dunzo               19
Swiggy              18
Name: PAYMENT_TYPE, dtype: int64
```

Figure 5.9 payment type column value counts

Below figure 5.10 shows the plot of payment type with respect to count of the payment type. The below data frame with payment type as index and Total bill value as column shows that out of all payments, amount received through paytm is more, second is card payment and the third is cash payments. I think this column won't have any impact on total sales.

```
sns.countplot("PAYMENT_TYPE", data=Billwise_sales)
Billwise_sales[['PAYMENT_TYPE','TOTAL_BILL_VALUE']].groupby(['PAYMENT_TYPE']).sum()
```

```
C:\Users\Bharathi Kondaveeti\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
as a keyword arg: x. From version 0.12, the only valid positional argument will be `
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

PAYMENT_TYPE	TOTAL_BILL_VALUE
Card	17592731.00
Cash	15580472.37
Credit	313757.00
Dunzo	12446.00
Google Pay	140247.00
NEFT	835729.00
Paytm	17720675.00
Redeem Points	124439.00
Swiggy	15949.00

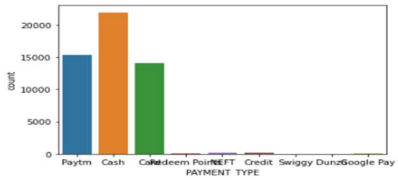


Figure 5.10 payment type column plot with respect to total sales

The below figure 5.11 shows the Online/offline column and its values with their counts. Also plotted the same. By seeing this we can say that both online payments are more compared to offline payments. By this we can say that this column won't have much impact on the total sales.

```
Billwise_sales['ONLINE/OFFLINE'].value_counts()
```

```
Online     29833
Offline    21171
Refund       733
Name: ONLINE/OFFLINE, dtype: int64
```

```
sns.countplot("ONLINE/OFFLINE", data=Billwise_sales)
```

```
  warnings.warn(
```

```
<AxesSubplot:xlabel='ONLINE/OFFLINE', ylabel='count'>
```

Figure 5.11 Online/Offline column plot

Below figure 5.12 shows the plot of customer bills with respect to day of the week. I have created day of the week column using date column and day_name() function. It groups all days from Monday's to Sunday's and gives a cumulative report. By seeing the plot, we can say that Saturday and Sunday's customer visits are little high compared to other days, but overall there's not much difference.

```
Billwise_sales['Dayofweek']= Billwise_sales['DATE'].dt.day_name()
plt.figure(figsize=(8,5))
sns.countplot(x='Dayofweek',data=Billwise_sales, palette='rainbow')
plt.title("DAY OF THE WEEK")
```

```
Text(0.5, 1.0, 'DAY OF THE WEEK')
```

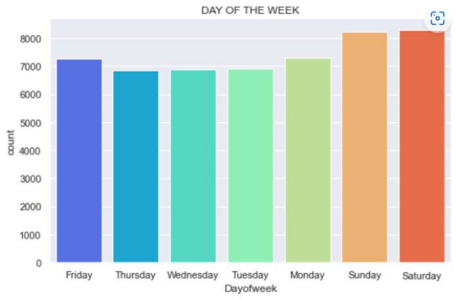


Figure 5.12 plot with respect to day of the week

Below figure 5.13 shows the plot with respect yearly customer bills. I have created year column using year function on date column. It groups all bills from particular years. We have 2019 January to 2022 September data. By seeing the plot, we can sh6blay that 2019 has high sales, 2020 has low sales due to COVID-19 pandemic and 2021 sales increased and now 2022 sale is little less as we have only 9 months data.

```
Billwise_sales['Year'] = pd.DatetimeIndex(Billwise_sales['DATE']).year
plt.figure(figsize=(8,5))
sns.countplot(x='Year',data=Billwise_sales, palette='rainbow')
plt.title("YEARLY SALES")
```

```
Text(0.5, 1.0, 'YEARLY SALES')
```

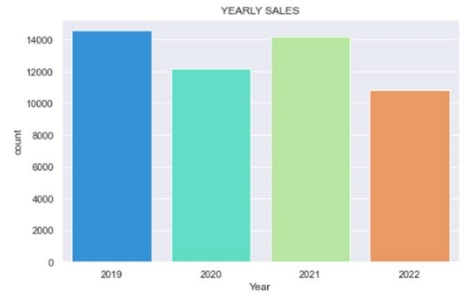


Figure 5.13 plot with respect to year

The below figure 5.14 shows the plot of Month-year and customer bills count.

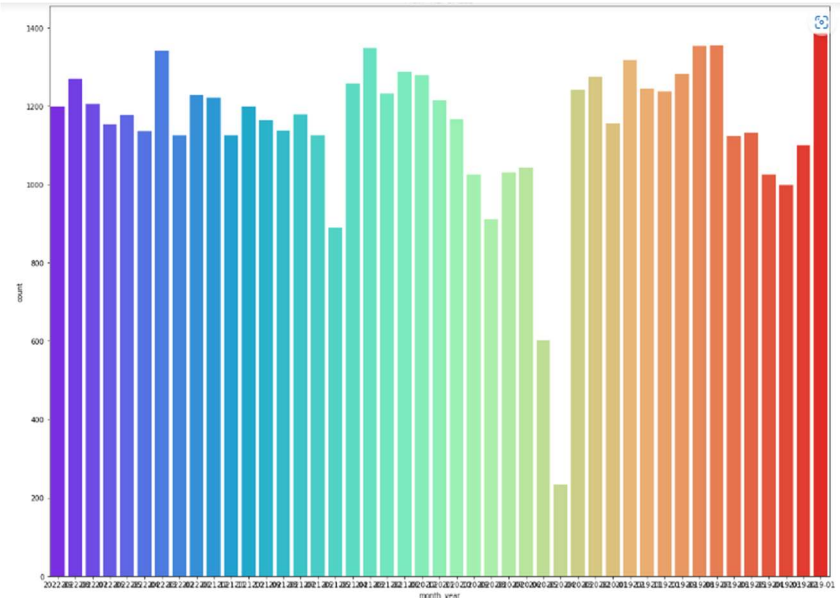


Figure 5.14 plot with respect to month

The maximum and minimum period as well as the number of bills per month are shown in the below figure 5.15. As can be seen, the highest sale was recorded in January of 2019 and the lowest in April of 2020.

```
In [7]: Billwise_sales['month_year'].max()
Out[7]: Period('2022-09', 'M')

In [8]: Billwise_sales['month_year'].min()
Out[8]: Period('2019-01', 'M')

In [9]: Billwise_sales['month_year'].value_counts()
Out[9]: 2019-01    1385
        2019-07    1355
        2019-08    1354
        2021-03    1349
        2022-03    1341
        2019-12    1318
        2021-01    1288
        2019-09    1283
        2020-12    1279
        2020-02    1275
        2022-08    1270
        2021-04    1258
        2019-11    1245
        2020-03    1241
        2019-10    1238
        2021-02    1232
        2022-01    1229
        2021-12    1221
        2020-11    1214
        2022-07    1206
        2022-09    1198
        2021-10    1198
        2021-07    1178
        2022-05    1177
        2020-10    1167
        2021-09    1164
        2020-01    1156
        2022-06    1153
        2021-08    1137
        2022-04    1135
        2019-05    1132
        2021-06    1126
        2021-11    1126
        2022-02    1125
        2019-06    1123
        2019-02    1100
        2020-06    1044
        2020-07    1030
        2019-04    1026
        2020-09    1025
        2019-03     999
        2020-08     911
        2021-05     890
        2020-05     603
        2020-04     233
        Freq: M, Name: month_year, dtype: int64
```

Figure 5.15 Total no of bills with respect to month

The min and max functions on the date column are shown in the below figure 5.16, and when they are subtracted, the result is the total number of days of data we have.

```
Billwise_sales['DATE'].min()
Timestamp('2019-01-01 00:00:00')

Billwise_sales['DATE'].max()
Timestamp('2022-09-30 00:00:00')

Billwise_sales['DATE'].max() - Billwise_sales['DATE'].min()
Timedelta('1368 days 00:00:00')
```

Figure 5.16 functions using Date time

Below figure 5.17 shows the plot of total no of items purchased with respect to count of bills.

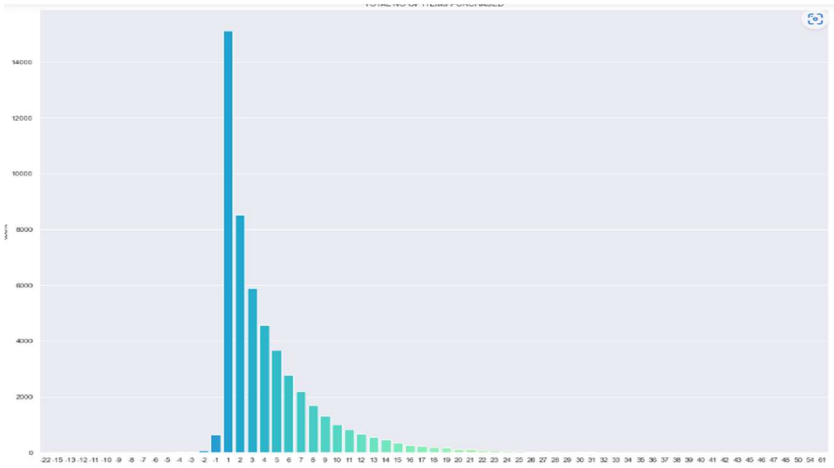


Figure 5.17 count of bills with respect to items purchased

The below figures 5.18 shows data of specified time range.

```
filt = (Billwise_sales['DATE'] >= pd.to_datetime('2019-01-01')) & (Billwise_sales['DATE'] < pd.to_datetime('2020-01-01'))
Billwise_sales.loc[filt]
```

	TIME	DATE	BILL_NO	CONSUMER_NAME	Taxable_Amount_0%	Taxable_Amount_5%	Taxable_Amount_12%	Taxable_Amount_18%	CGST	SGST
37179	08:57pm	2019-12-31	A/1920/10962	Jyoshna	710.0	0.0	0.0	0.0	0.0	0.0
37180	08:55pm	2019-12-31	A/1920/10961	Kumar	130.0	0.0	0.0	0.0	0.0	0.0
37181	08:48pm	2019-12-31	A/1920/10960	Jagadeesh	220.0	0.0	0.0	0.0	0.0	0.0
37182	08:25pm	2019-12-31	A/1920/10959	Guest	50.0	0.0	0.0	0.0	0.0	0.0
37183	07:59pm	2019-12-31	A/1920/10958	Vijay Kumar 9th Phase	1420.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...
51732	01:55pm	2019-01-01	A-A000128	ravi l	205.0	0.0	0.0	0.0	0.0	0.0
51733	01:48pm	2019-01-01	A-A000127	balaji	880.0	0.0	0.0	0.0	0.0	0.0
51734	01:40pm	2019-01-01	A-A000126	ramu	225.0	0.0	0.0	0.0	0.0	0.0
51735	01:37pm	2019-01-01	A-A000125	ratna	390.0	0.0	0.0	0.0	0.0	0.0
51736	01:29pm	2019-01-01	A-A000124	Ramesh Hydernagar	770.0	0.0	0.0	0.0	0.0	0.0

14558 rows × 16 columns

Figure 5.18 getting data from a specified range

The below figure 5.19 shows the sum of hourly total sale and total number of items purchased on a particular day grouped by date and hour columns.

```
sorteddf = Billwise_sales.groupby(['DATE','Hour'])['TOTAL_BILL_VALUE','TOTAL_NOOF_ITEMS_PURCHASED'].sum()
sorteddf
```

```
C:\Users\BHARAT~1\AppData\Local\Temp/ipykernel_24312/198528915.py:1: FutureWarning: Indexing with multiple keys (implicitly con
verted to a tuple of keys) will be deprecated, use a list instead.
  sorteddf = Billwise_sales.groupby(['DATE','Hour'])['TOTAL_BILL_VALUE','TOTAL_NOOF_ITEMS_PURCHASED'].sum()
```

		TOTAL_BILL_VALUE	TOTAL_NOOF_ITEMS_PURCHASED
DATE	Hour		
	13	3295.0	27
	14	6585.0	51
2019-01-01	15	1185.0	10
	16	5065.0	40
	17	3080.0	29
...	...	...	...
	17	3500.0	15
	18	14710.0	4
2022-09-30	19	340.0	3
	20	13780.0	13
	21	12165.0	7

14077 rows × 2 columns

Figure 5.19 sum of sales and items grouped by date and hour

The below figure 5.20 shows the sum of total sale and total number of items purchased on daily basis grouped by date column.

```
sorted_df2 = Billwise_sales.groupby('DATE')['TOTAL_BILL_VALUE','TOTAL_NOOF_ITEMS_PURCHASED'].sum().reset_index()
sorted_df2
```

```
C:\Users\BHARAT~1\AppData\Local\Temp/ipykernel_24312/473875180.py:1: FutureWarning: Indexing with multiple keys (:
verted to a tuple of keys) will be deprecated, use a list instead.
  sorted_df2 = Billwise_sales.groupby('DATE')['TOTAL_BILL_VALUE','TOTAL_NOOF_ITEMS_PURCHASED'].sum().reset_index(
```

	DATE	TOTAL_BILL_VALUE	TOTAL_NOOF_ITEMS_PURCHASED
0	2019-01-01	37420.0	296
1	2019-01-02	36140.0	245
2	2019-01-03	33921.0	163
3	2019-01-04	45510.0	348
4	2019-01-05	31980.0	224
...	...	...	...
1356	2022-09-26	26845.0	108
1357	2022-09-27	29009.0	119
1358	2022-09-28	56172.0	168
1359	2022-09-29	36421.0	105
1360	2022-09-30	89272.0	163

1361 rows × 3 columns

Figure 5.20 sum of sales and items grouped by date

The below figure 5.21 shows the sum of total sale and total number of items purchased on monthly basis grouped by month_year column.

```
Billwise_sales.groupby('month_year')['TOTAL_BILL_VALUE','TOTAL_NOOF_ITEMS_PURCHASED'].sum()
```

```
C:\Users\BHARAT~1\AppData\Local\Temp/ipykernel_24312/1402473909.py:1: FutureWarning: Indexing w
nverted to a tuple of keys) will be deprecated, use a list instead.
  Billwise_sales.groupby('month_year')['TOTAL_BILL_VALUE','TOTAL_NOOF_ITEMS_PURCHASED'].sum()
```

month_year	TOTAL_BILL_VALUE	TOTAL_NOOF_ITEMS_PURCHASED
2019-01	855053.00	5657
2019-02	667027.00	4349
2019-03	699745.00	4249
2019-04	788361.00	4567
2019-05	803643.00	5018
2019-06	1092992.00	4829
2019-07	1029301.00	6173
2019-08	1082340.00	5683
2019-09	1554605.00	5144
2019-10	1087014.00	5070
2019-11	971971.00	4820
2019-12	1084130.00	5236
2020-01	1026015.00	4379
2020-02	1160771.00	5008
2020-03	1354048.00	5325
2020-04	169301.00	708
2020-05	648946.00	2084
2020-06	1117881.00	3890
2020-07	1206249.00	4396
2020-08	1154275.00	4027
2020-09	1204686.00	4493

Figure 5.21 sum of sales and items grouped by month and year

The below figure 5.22 shows the plot of total sales with respect to every month_year column.

```
Billwise_sales.plot.line(x = 'month_year', y = 'TOTAL_BILL_VALUE',
                         color = 'salmon', title = 'Monthly Sales')
```

```
<AxesSubplot:title={'center':'Monthly Sales'}, xlabel='month_year'>
```

Figure 5.22 plot of monthly sales

The below figure 5.23 show plots for total sale and total no of items purchased with respect to every month.

Figure 5.23 plot for monthly sales and items purchased

5.2 Product wise sale report Exploratory Data Analysis

The product wise sales report contains data from January 2019 to September 2022 with respect to each product like product name, brand, category and sub-category to which it belongs to, the quantity sold and in number of bills the product was sold, total revenue got from that product etc. The below figure 5.24 shows the total count of products in each Brand. We can see that there are a total of 79 brands. But as per the information I got from the store management, there are only 18 brands. All the remaining entries are duplicate names like as shown in the below figure, for Prakruthivanam Brand there are duplicates like Prakruthivanamm, Prakruthiv, Prakruthiivannam, prakruthivanam, PRAKRUTHIVANAM, PRAKRUTHIV etc. so I replaced all the duplicate brands in the dataframe.

```
product_sales['BRAND'].value_counts()
```

```
Prakruthivanam                  5488
Clay Pots                        395
Herbal Strategi                  312
Arogya Rahasya                   306
Dry Fruits                       258
                                 ...
Sweetee Palmyrah Palm Jaggery      1
Prakruthivanamm                    1
Arogya Rahasya                     1
Dry Fruits                         1
PRAKRUTHIV                         1
Name: BRAND, Length: 79, dtype: int64
```

```
product_sales['BRAND'].unique()
```

```
array(['Prakruthivanam', 'Go Desi', 'Dry Fruits', 'Timbaktu', 'Pure&Sure',
       'Arogya Rahasya', 'Surabhi Healthy Life', 'GO DESI', 'Radico',
       'Water Filter', 'Herbal Strategi', 'Sri venkateswara Enterprises',
       'Timbaktu ', 'GO DESi', 'Native Circle', 'GCC', 'Herbal Strategi ',
       'Dathu Naturals', 'The Kind Earth', 'Mittise', 'NATIVE CIRCLE ',
       'Cast Iron', 'Clay Pots', 'NATIVE CIRCLE', 'Native Circle ',
       'Pure & Sure', 'Miscellaneous', 'Absolute Ghee', 'Absolute Milk',
       'Water Filter ', 'Native Cvircle', 'Cloth Bag', 'Akshaya',
       'Healthy Life', 'Vaarahi', 'Noor Pans', 'Travels/courier', 'Iron',
       'Surabhi', 'Native Cicrle', 'Noor Pan', 'Gcc',
       'Sweetee Palmyrah Palm Jaggery', 'Gouthami Agro Industries',
       'Gou Ganga', 'Prakruthivanamm', 'Dry Fruits ',
       'Sree Chandan Corporation ', 'Pure & Sue', 'Gouthami Agro',
       'Sree Chandan Corporation', 'Arogya Rahasya ',
       'Shree Chandan Corporation', 'Sweetee Palmyrah Palm Jaggery ',
       'Soap Stone', 'Manam Foods', 'Prakruthiv', 'Prakruthiivannam',
       'Kimia', 'Sree Chandan Corporotion', 'DHAPUUR GREEN',
       'prakruthivanam', 'PRAKRUTHIVANAM', 'PRAKRUTHIV',
       'Herbal Strategic', 'Tamarind-250 Gms-pvs', 'Sadguru Sri Sri',
```

Figure 5.24 Brand column values

After replacing all the duplicate name, the below figure 5.25 shows 18 unique values in Brand column.

```
product_sales.BRAND.replace({'Sweetee Palmyrah Palm Jaggery':"Prakruthivanam",'Prakruthivanamm':"Prakr
product_sales.BRAND.replace({'Surabhi':"Surabhi Healthy Life",'Healthy Life':"Surabhi Healthy Life",'Su
product_sales.BRAND.replace({'Timbaktu ' : "Timbaktu" }, inplace = True)
product_sales.BRAND.replace({'Sree Chandan Corporation ':"Dry Fruits",'Shree Chandan Corporation':"Dry
product_sales.BRAND.replace({'Herbal Strategi':"Herbal Strategi",'Herbal Strategi ':"Herbal Strategi",
product_sales.BRAND.replace({'Rathnam':"Miscellaneous",'Travels/courier':"Miscellaneous",'Cloth Bag':"
product_sales.BRAND.replace({'Clay Pots' : "Cookware" }, inplace = True)
product_sales.BRAND.replace({'Native Circle':"Native Circle",'NATIVE CIRCLE ':"Native Circle",'NATIVE C
product_sales.BRAND.replace({'Water Filter ' : "Water Filter", 'Water Filt' : "Water Filter"}, inplace
product_sales.BRAND.replace({'Danthu Naturals' : "Dathu Ayurvedam", 'Dathu Naturals' : "Dathu Ayurveda
product_sales.BRAND.replace({'Absolute Ghee' : "Absolute Milk" }, inplace = True)
product_sales.BRAND.replace({'Arogya Rahasya ':"Arogya Rahasya",'Gou Ganga':"Arogya Rahasya",'ArogyaRa
product_sales.BRAND.replace({'Cast Iron':"Cookware",'Soap Stone':"Cookware",'Woodenware':"Cookware",'No
product_sales.BRAND.replace({'Clay pot' : "Cookware" }, inplace = True)
```

```
product_sales['BRAND'].unique()
```

```
array(['Prakruthivanam', 'Go Desi', 'Dry Fruits', 'Timbaktu',
       'Pure & Sure', 'Arogya Rahasya', 'Surabhi Healthy Life', 'Radico',
       'Water Filter', 'Herbal Strategi', 'Puro', 'Native Circle',
       'Dathu Ayurvedam', 'The Kind Earth', 'Mittise', 'Cookware',
       'Miscellaneous', 'Absolute Milk'], dtype=object)
```

Figure 5.25 replacing duplicates values

The below figure 5.26 shows the value count of products in each brand.

```
product_sales['BRAND'].value_counts()
```

```
Prakruthivanam          5587
Cookware                 563
Dry Fruits               400
Herbal Strategi          323
Arogya Rahasya           322
Pure & Sure              310
Miscellaneous            194
Timbaktu                 188
Go Desi                  174
Native Circle            106
Mittise                   79
The Kind Earth            77
Dathu Ayurvedam           61
Water Filter              59
Radico                    53
Surabhi Healthy Life      51
Puro                      25
Absolute Milk             16
Name: BRAND, dtype: int64
```

Figure 5.26 final list of brands

The below figure 5.27 shows the brand wise sale which is having highest MRP, and also Quantity sold, no of bills, and revenue with respect to brand.

```
product_sales.groupby('BRAND')['MRP'].nlargest(1)
```

```
BRAND
Absolute Milk          1104     1500
Arogya Rahasya          168      580
Cookware               7475     2400
Dathu Ayurvedam        2960      600
Dry Fruits              171      815
Go Desi                 184      500
Herbal Strategi         515      700
Miscellaneous          1715     2000
MittiSe                1233     1150
Native Circle          1000      400
Prakruthivanam         4757     7500
Pure & Sure             587     1100
Puro                    122      110
Radico                  394      699
Surabhi Healthy Life   1416     2000
The Kind Earth         2476      150
Timbaktu                679     1750
Water Filter             90    12000
Name: MRP, dtype: int64
```

```
df = product_sales.groupby('BRAND')['QTY SOLD','NO. OF BILLS','REVENUE'].sum()
df
```

```
C:\Users\BHARAT~1\AppData\Local\Temp/ipykernel_16960/3293698101.py:1: FutureWar
ltiple keys (implicitly converted to a tuple of keys) will be deprecated, use a
  df = product_sales.groupby('BRAND')['QTY SOLD','NO. OF BILLS','REVENUE'].sum(
```

BRAND	QTY SOLD	NO. OF BILLS	REVENUE
Absolute Milk	205.000	185	167757.06
Arogya Rahasya	15159.800	10427	1567560.42
Cookware	5196.067	4617	1612946.43
Dathu Ayurvedam	2492.000	2047	284253.61
Dry Fruits	3883.000	3012	544159.84
Go Desi	35288.000	8807	244847.59
Herbal Strategi	8011.000	6246	1044497.93
Miscellaneous	5939.065	3941	633118.47
MittiSe	793.000	738	235767.53
Native Circle	1589.000	1406	332713.82
Prakruthivanam	240726.000	167515	36211196.33
Pure & Sure	5190.000	4405	984891.30
Puro	853.000	677	93188.30
Radico	304.000	289	71904.99
Surabhi Healthy Life	862.000	729	585713.09
The Kind Earth	3496.000	1915	188741.49
Timbaktu	4233.000	3445	408576.44
Water Filter	825.000	782	7272513.07

Figure 5.27 extracting information related to brand

The below plots in the figure 5.28 are between brand vs revenue and brand vs quantity sold. By these plots we can easily say that Prakruthivanam brand has highest sales.

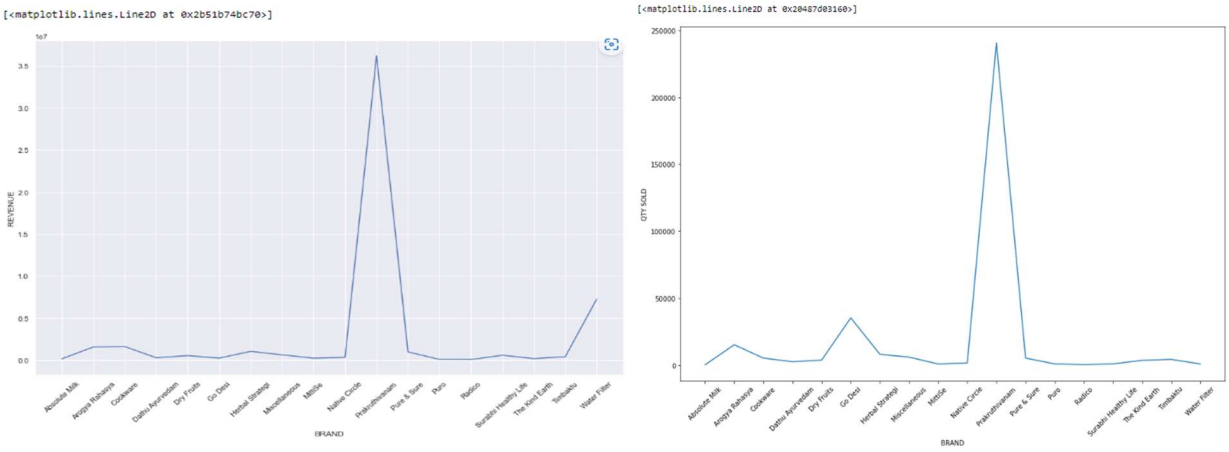


Figure 5.28 ploting brand with respect to revenue

6 Design

The figure 6.1 below shows the project's pipeline.

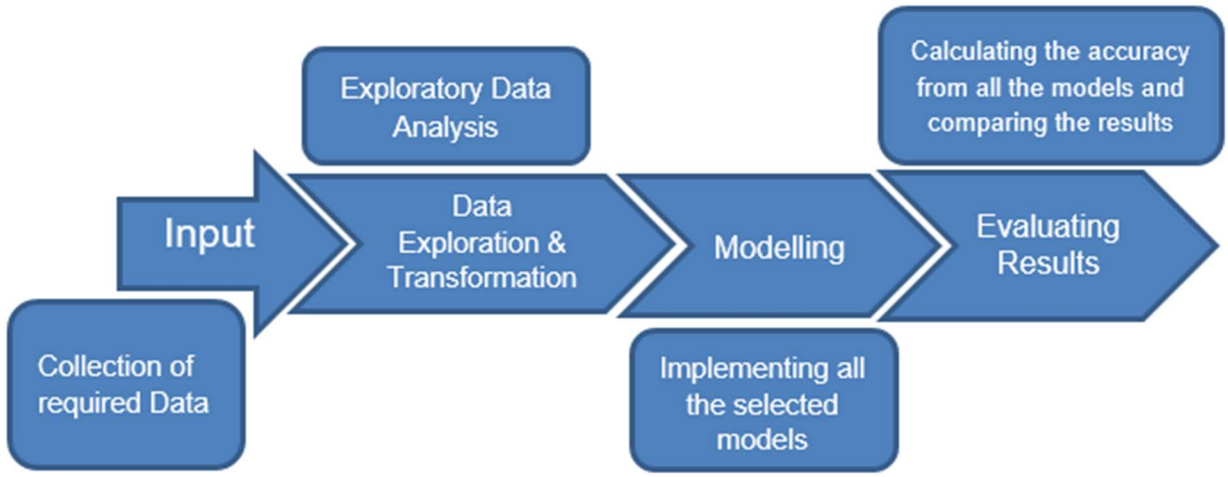


Figure 6.1 Pipeline of the Architecture

Supervised machine learning techniques are applied to the development of this model. For making predictions on the selected data, each of the algorithms used in this project was best suited. The success of the retail industry will be greatly aided by the model that predicts sales with a high degree of accuracy.

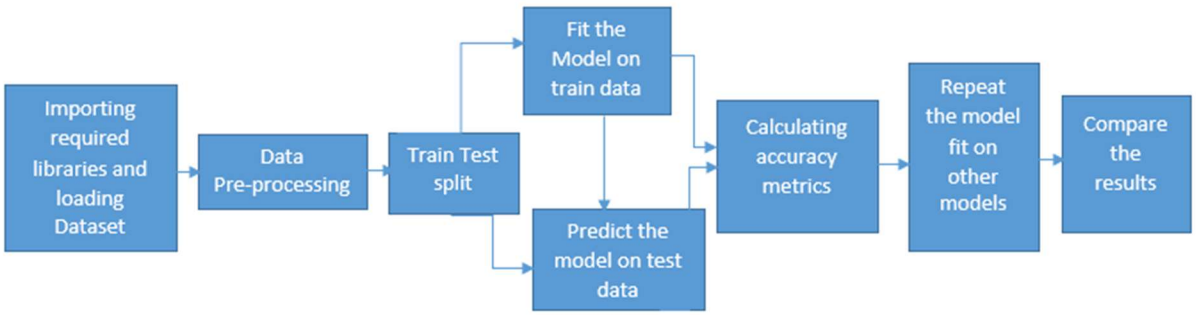


Figure 6.2 Block Diagram of step-wise implementation

The figure 6.2 above shows the stepwise implementation. The architecture's design can be segmented into the various stages.

The first step is to importing required libraries and loading the data into panda's dataframe. Then need to do some Data analysis and plotted the graphs using some plotting libraries. Some insights were taken from other columns. In order to improve decision accuracy, data pre-processing was carried out as a result of the analysis, which involved cleaning the data, removing some features, and grouping some columns. Following all of these steps, there were almost 1361 rows representing instances and 2 columns for applying the model. Then the data was then divided into two parts, with training data making up 70% and testing data 30%.

After splitting the data, the next step was to create models using the training data and predict the model using the testing data. Eight different architectures were put into use at this stage with the same goal of forecasting sales. This has been discussed in detail in Chapter 6 Implementation.

The model was created, and the next step was to validate it. To do this, the Mean Absolute Error and Root Mean Square Error were calculated. The results of Chapter 7 have demonstrated the same. The best model among all the implemented models was chosen after the results of all the models were compared.

7 Implementation

This chapter contains the implementation of the project. The chosen models needed to be implemented after the requirements and models to be used were finalised. Eight different machine learning algorithms were used, as previously mentioned in the methodology.

The below figure 7.1 shows Classical decomposition of a time series by considering the series as an additive combination of the base level, trend, seasonal index and the residual for the Total number of items purchased with respect to month wise. The pattern in trend clearly shows there is no trend in the data. The seasonality plot shows that there is a repeated pattern in the data.

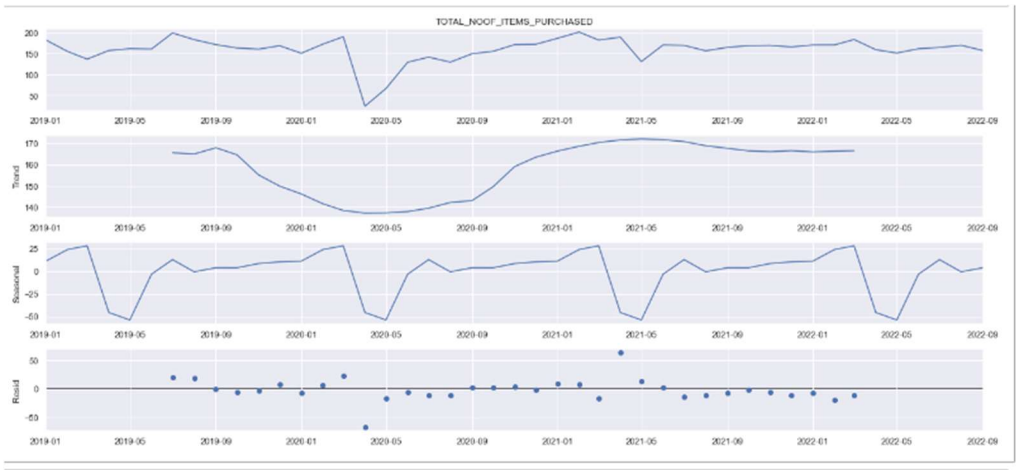


Figure 7.1 Decomposition of Total_noof_items_purchased with respect to each month

The below figure 7.2 shows the decomposition of Time series shows the Trend, seasonality and Residuals for the Total bill value of sales with respect to month wise. The pattern in trend clearly shows there is no trend in the data. The seasonality plot shows that there is a repeated pattern in the data. Both the Total No of items purchased and Total bill value plots seems to be similar. So we can proceed our model implementation with respect to the total bill value and Date.

Figure 7.2 Decomposition of Total sales with respect to each month

The below figure 7.3 shows the plot of total sale along with rolling mean and standard deviation of the monthly sales data.

Figure 7.3 Rolling Mean and Standard Deviation of monthly sales

The below figure 7.4 shows the plot of total sale along with rolling mean and standard deviation of the daily sales data.

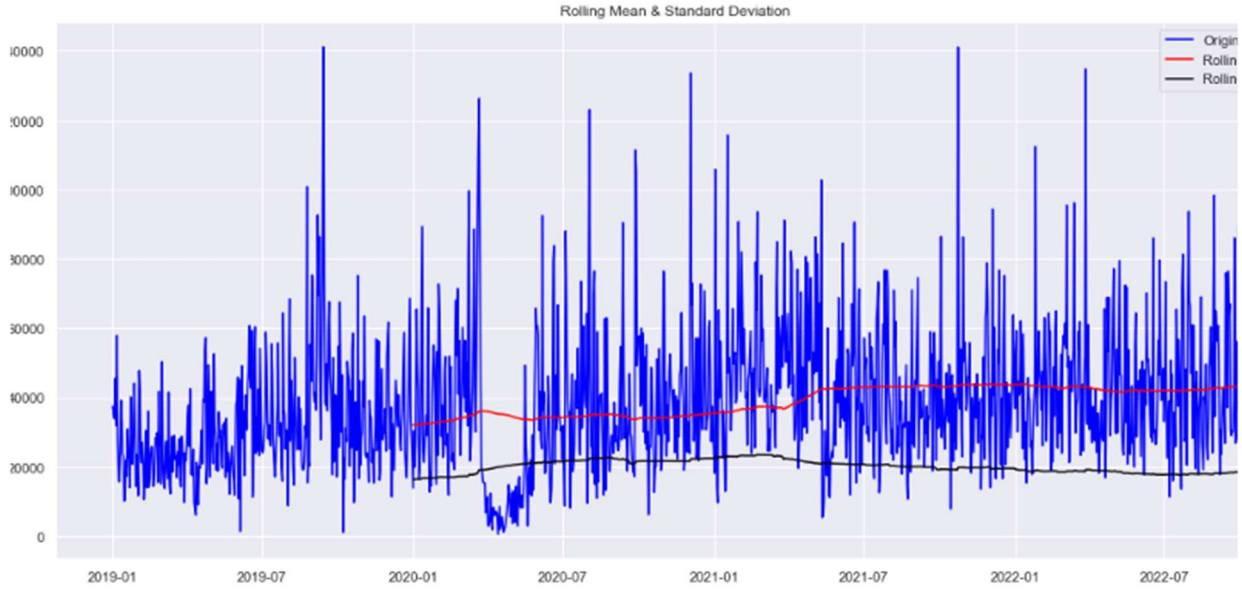


Figure 7.4 Rolling Mean and Standard Deviation of daily sales

In order to implement the forecasting we need the data to be stationary. Here I am using Dickey-Fuller Test to check the stationarity. The below figure 7.5 shows the results of Dickey-Fuller Test. The results shows that the sales data is stationary because p-value is very less than 0.05.

```
Results of Dickey-Fuller Test:
Test Statistics                -5.720473e+00
p-value                         6.960301e-07
#lags Used                      1.300000e+01
Number of Observations Used     1.347000e+03
Critical Value (1%)            -3.435214e+00
Critical Value (5%)            -2.863688e+00
Critical Value (10%)           -2.567914e+00
dtype: float64
```

Figure 7.5 Results of Dickey-Fuller Test

The below figure 7.6 shows the plot between Total Sales Vs Date. The bill wise data is grouped with respect to date.

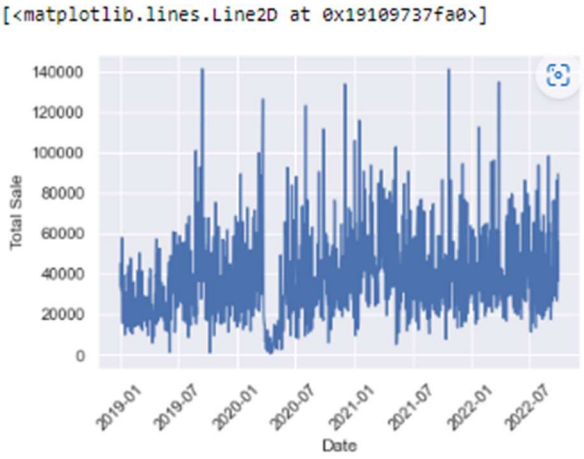


Figure 7.6 Total sale Vs Date

7.1 Regression Models

For better performance of the regression models, I am considering current day sales and previous three day sales which is shown in the below figure 7.7.

DATE	TOTAL_BILL_VALUE	Previous_day_sale	Two_days_before_sale	Three_days_before_sale
2019-01-04	45510.0	33921.0	36140.0	37420.0
2019-01-05	31980.0	45510.0	33921.0	36140.0
2019-01-06	57938.0	31980.0	45510.0	33921.0
2019-01-07	29883.0	57938.0	31980.0	45510.0
2019-01-08	19970.0	29883.0	57938.0	31980.0
...	...	...	...	...
2022-09-26	26845.0	86078.0	38915.0	30231.0
2022-09-27	29009.0	26845.0	86078.0	38915.0
2022-09-28	56172.0	29009.0	26845.0	86078.0
2022-09-29	36421.0	56172.0	29009.0	26845.0
2022-09-30	89272.0	36421.0	56172.0	29009.0

1358 rows × 4 columns

Figure 7.7 final data to proceed with applying regression models

Then the data is further split into train and test. The below figure 7.8 shows the train and test data.

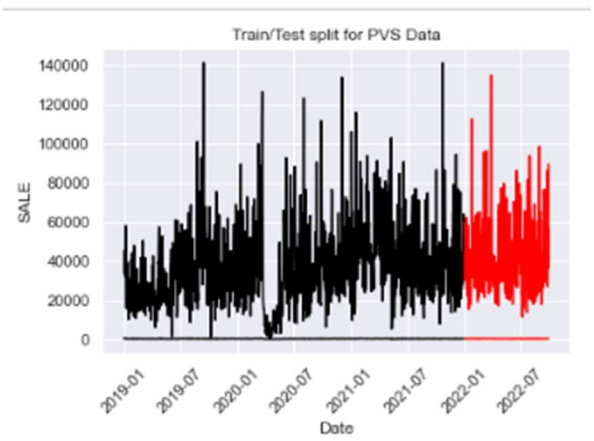


Figure 7.8 Train/Test split

7.1.1 Linear Regression

After splitting the data, the linear regression model is applied on train data and then predict on the test data. The below figure 7.9 shows the plot for test data actuals and predicted sales.

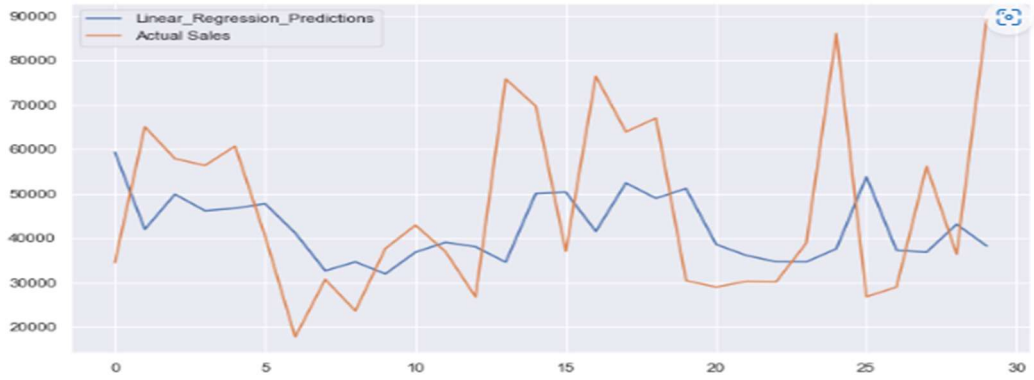


Figure 7.9 Linear regression predictions and actual sales plots

7.1.2 Random Forest Regression

The Random Forest Regression model is applied on train data and then predict on the test data. The below figure 7.10 shows the plot for test data actuals and predicted sales.

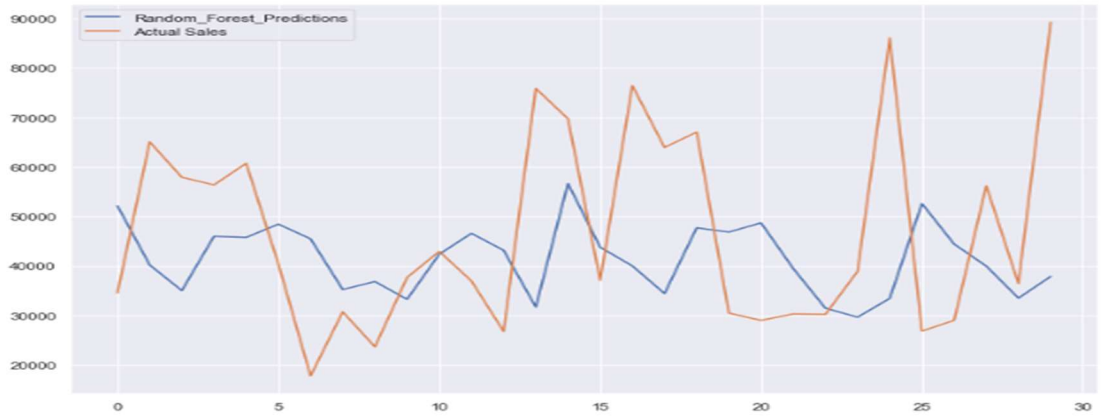


Figure 7.10 Random Forest regression predictions and actual sales plots

7.1.3 Decision Tree Regression

The Decision Tree Regression model is applied on train data and then predict on the test data. The below figure 7.11 shows the plot for test data actuals and predicted sales.

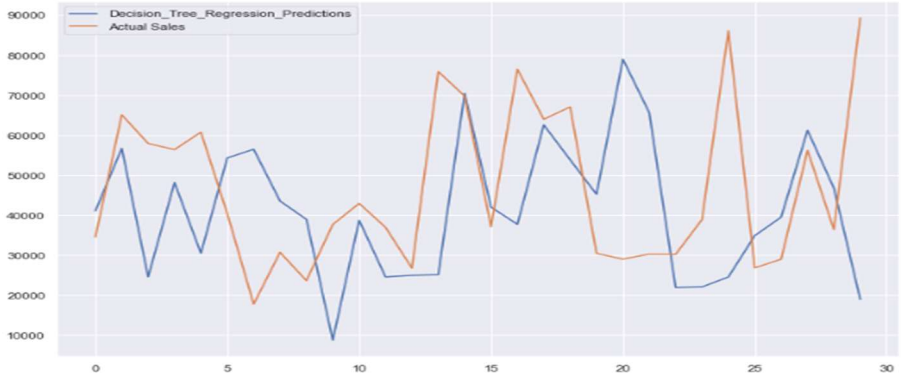


Figure 7.11 Decision Tree regression predictions and actual sales plots

7.1.4 XGBoost Regression

The XGBoost Regression model is applied on train data and then predict on the test data. The below figure 7.12 shows the plot for test data actuals and predicted sales.

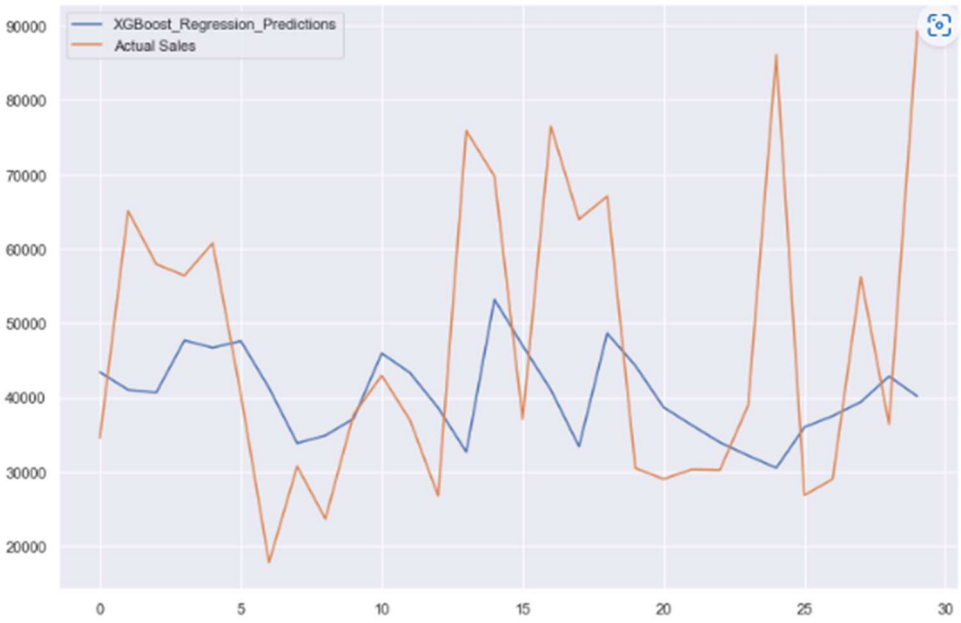


Figure 7.12 XGBoost regression predictions and actual sales plots

7.2 ARIMA (Auto Regressive Integrated Moving Average) Model

In order to implement the ARIMA models, we need to find the parameters p, q and d. Below figure 7.13 shows the graph of auto correlation.

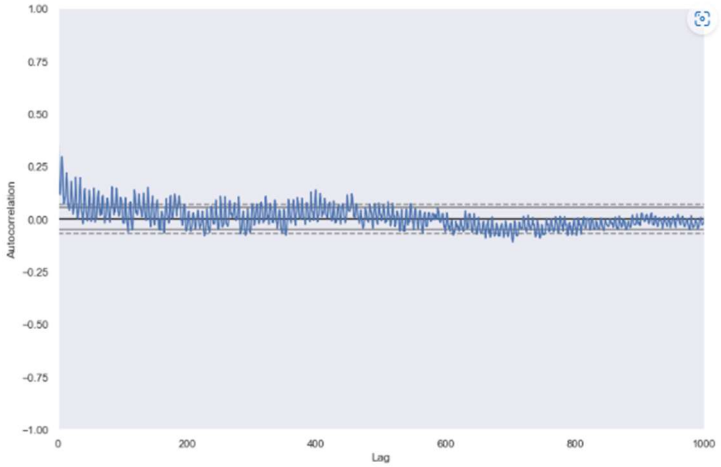


Figure 7.13 Autocorrelation plot

The lag of the autoregressive component was calculated using the PACF which is the parameter p, whereas the parameter q is lag of the moving average component was determined using the ACF. The parameter d is taken from differencing, if the data is stationary then we need to consider d as zero, if the data is not stationary, we can convert the data by using 1st order differencing and check for stationarity. If its stationary, then we

consider d as 1 if it's not stationary, we increment the order and proceed the check and so on.

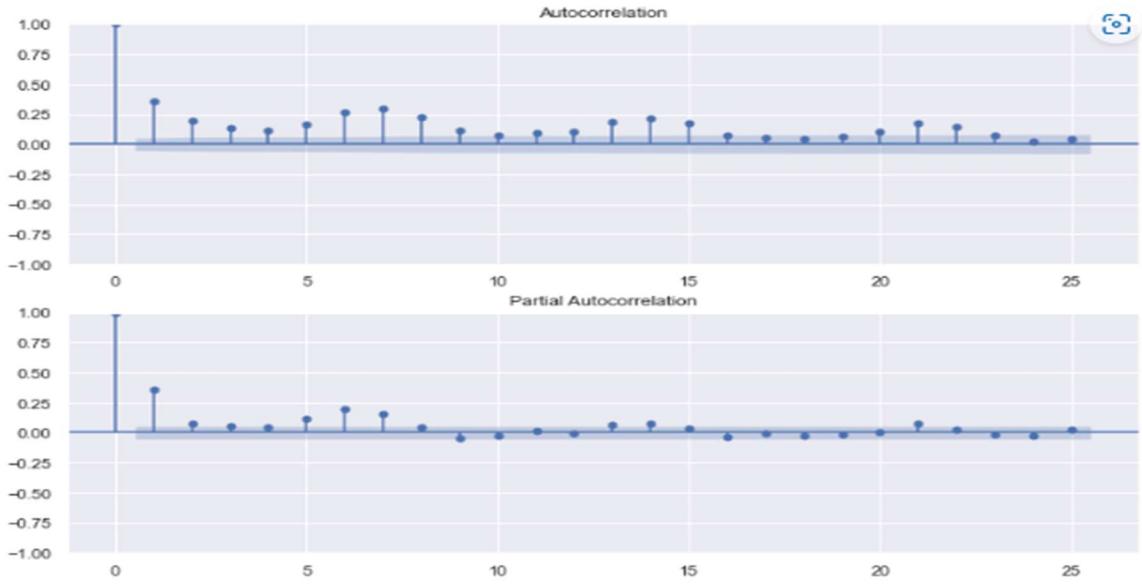


Figure 7.14 ACF and PACF plots

The above figure 7.14 shows the ACF and PACF plots. By using the plot we can calculate the p and q values as p=2/3 and q=5. As our data is stationary, we can take d=0.

7.2.1 AR (Auto Regressive) Model:

The below figure 7.15 shows the summary of AR model with p=2 and p=3. By this we can see p=3 results are better so we consider this p value for further ARIMA model.

AutoReg Model Results

Dep. Variable:	TOTAL_BILL_VALUE	No. Observations:	1331
Model:	AutoReg(2)	Log Likelihood	-14950.287
Method:	Conditional MLE	S.D. of innovations	18589.218
Date:	Wed, 23 Nov 2022	AIC	29908.574
Time:	14:27:34	BIC	29929.343
Sample:	2	HQIC	29916.358
	1331		

	coef	std err	z	P>|z|	[0.025	0.975]
const	2.257e+04	1291.470	17.476	0.000	2e+04	2.51e+04
TOTAL_BILL_VALUE.L1	0.3357	0.027	12.242	0.000	0.282	0.389
TOTAL_BILL_VALUE.L2	0.0750	0.027	2.733	0.006	0.021	0.129

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	2.0444	+0.0000j	2.0444	0.0000
AR.2	-6.5191	+0.0000j	6.5191	0.5000

AutoReg Model Results

Dep. Variable:	TOTAL_BILL_VALUE	No. Observations:	1331
Model:	AutoReg(3)	Log Likelihood	-14937.970
Method:	Conditional MLE	S.D. of innovations	18574.281
Date:	Fri, 25 Nov 2022	AIC	29885.941
Time:	12:50:34	BIC	29911.898
Sample:	3	HQIC	29895.670
	1331		

	coef	std err	z	P>|z|	[0.025	0.975]
const	2.148e+04	1431.881	15.002	0.000	1.87e+04	2.43e+04
TOTAL_BILL_VALUE.L1	0.3320	0.027	12.077	0.000	0.278	0.386
TOTAL_BILL_VALUE.L2	0.0589	0.029	2.037	0.042	0.002	0.116
TOTAL_BILL_VALUE.L3	0.0485	0.028	1.761	0.078	-0.005	0.102

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.7284	-0.0000j	1.7284	-0.0000
AR.2	-1.4717	-3.1255j	3.4547	-0.3200
AR.3	-1.4717	+3.1255j	3.4547	0.3200

Figure 7.15 Summary of Auto Regression Model for p=2 and p=3

The below figure 7.16 shows the predictions of AR model on test data. By seeing the blue plot we can say AR model won't fit for our data.

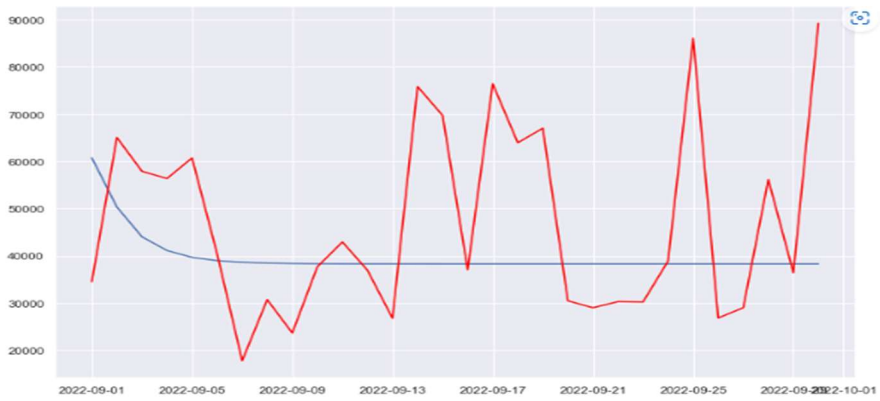


Figure 7.16 Predictions on test data using AR model

7.2.2 MA (Moving Average) Model:

The below figure 7.17 shows the MA model predictions on test data.

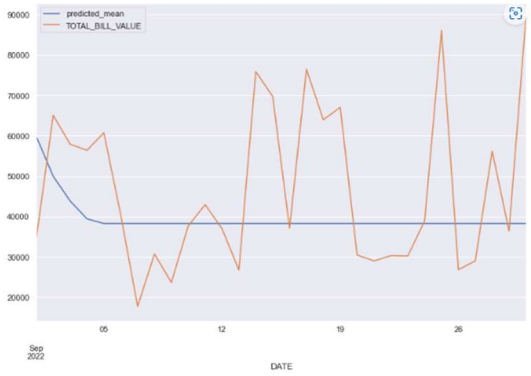


Figure 7.17 Predicting on test data using MA model.

Moving Average is for predicting future dates. So I tried to implement MA model for the future 30 dates from 1 Oct 2022 to 31 Oct 2022 as shown in figure 7.18. By seeing the MA plots, we can clearly say that MA model won't fit for our data.

```
#For Future Dates
index_future_dates=pd.date_range(start='2022-10-01',end='2022-10-31')
#print(index_future_dates)
pred=model1.predict(start=len(sorted_df3)+1,end=len(sorted_df3)+31,typ='levels').rename('MA Predictions')
#print(comp_pred)
pred.index=index_future_dates
print(pred)
```

```
2022-10-01    46934.320156
2022-10-02    43340.448991
2022-10-03    39426.778539
2022-10-04    38454.405121
2022-10-05    38454.405121
2022-10-06    38454.405121
2022-10-07    38454.405121
2022-10-08    38454.405121
2022-10-09    38454.405121
2022-10-10    38454.405121
2022-10-11    38454.405121
2022-10-12    38454.405121
2022-10-13    38454.405121
2022-10-14    38454.405121
2022-10-15    38454.405121
2022-10-16    38454.405121
2022-10-17    38454.405121
2022-10-18    38454.405121
2022-10-19    38454.405121
2022-10-20    38454.405121
2022-10-21    38454.405121
2022-10-22    38454.405121
2022-10-23    38454.405121
2022-10-24    38454.405121
2022-10-25    38454.405121
2022-10-26    38454.405121
2022-10-27    38454.405121
2022-10-28    38454.405121
2022-10-29    38454.405121
2022-10-30    38454.405121
2022-10-31    38454.405121
Freq: D, Name: MA Predictions, dtype: float64
```

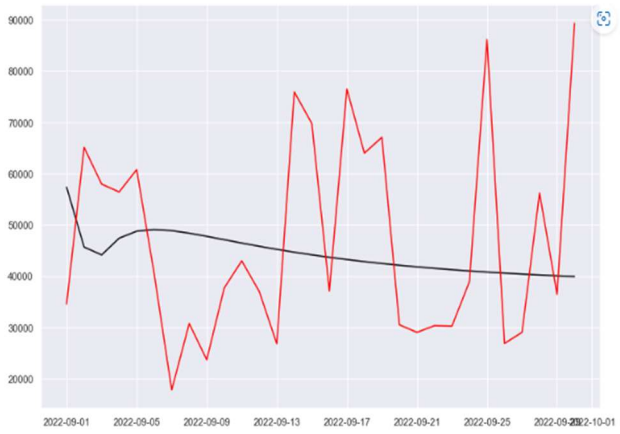


Figure 7.18 predicting future dates using MA model Oct1 to 31

7.2.3 ARMA Model (Auto Regressive Moving Average) Model:

SARIMAX Results

Dep. Variable:	TOTAL_BILL_VALUE	No. Observations:	1331
Model:	ARIMA(3, 0, 4)	Log Likelihood	-14897.545
Date:	Fri, 25 Nov 2022	AIC	29813.089
Time:	13:40:44	BIC	29859.833
Sample:	0	HQIC	29830.607
	- 1331		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	3.827e+04	2193.680	17.444	0.000	3.4e+04	4.26e+04
ar.L1	2.1984	0.018	122.300	0.000	2.163	2.234
ar.L2	-2.1751	0.025	-88.101	0.000	-2.223	-2.127
ar.L3	0.9464	0.018	52.629	0.000	0.911	0.982
ma.L1	-1.9541	0.030	-65.512	0.000	-2.013	-1.896
ma.L2	1.7499	0.053	33.264	0.000	1.647	1.853
ma.L3	-0.5838	0.052	-11.257	0.000	-0.685	-0.482
ma.L4	-0.0890	0.028	-3.153	0.002	-0.144	-0.034
sigma2	3.205e+08	0.277	1.16e+09	0.000	3.21e+08	3.21e+08

Ljung-Box (L1) (Q):	0.01	Jarque-Bera (JB):	915.22
Prob(Q):	0.94	Prob(JB):	0.00
Heteroskedasticity (H):	1.30	Skew:	1.15
Prob(H) (two-sided):	0.01	Kurtosis:	6.35

```
DATE
2022-09-01    51589.910817
2022-09-02    47987.155821
2022-09-03    50500.238481
2022-09-04    51463.204230
2022-09-05    49868.020902
2022-09-06    46645.145929
2022-09-07    43941.129061
2022-09-08    43497.018538
2022-09-09    45351.956476
2022-09-10    47836.615926
2022-09-11    48843.843448
2022-09-12    47409.318758
2022-09-13    44416.455451
2022-09-14    41910.531533
2022-09-15    41553.657390
2022-09-16    43387.191975
2022-09-17    45822.532010
2022-09-18    46850.453461
2022-09-19    45548.437917
2022-09-20    42755.189016
2022-09-21    40419.456341
2022-09-22    40127.955876
2022-09-23    41923.944960
2022-09-24    44295.625783
2022-09-25    45327.136388
2022-09-26    44135.940575
2022-09-27    41518.257808
2022-09-28    39330.837590
2022-09-29    39088.393028
2022-09-30    40835.792651
Name: predicted_mean, dtype: float64
```

Figure 7.19 Summary of ARIMA model and predictions on test data

```
[<matplotlib.lines.Line2D at 0x24afd2299d0>]
```

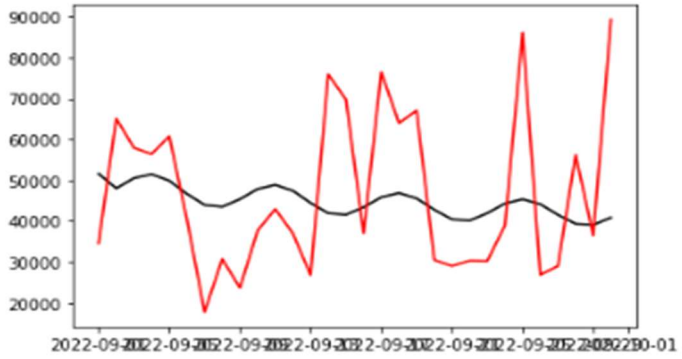


Figure 7.20 Plot using ARIMA Predictions and test data

The above figures 7.19 and 7.20 shows the summary results, prediction values and plotted the same. The plot is better compared to AR and MA models. The below figure 7.21 shows the future predictions and plotted the same.

```
#Make Future predictions:
#index_future_dates=pd.date_range(start='2022-10-

start = len(sorted_df3)+1
end = len(sorted_df3)+31
print('The Future predictions for next month')
pred_future = ARMA_model.predict(start=len(sorted_

pred_future.index=index_future_dates
print(len(pred_future))
```

```
The Future predictions for next month
31
```

```
C:\Users\Bharathi Kondaveeti\anaconda3\lib\site-pa
x is available. Prediction results will be given w
  return get_prediction_index(
```

```
pred_future
```

```
2022-10-01    44157.131377
2022-10-02    43059.932339
2022-10-03    40598.580722
2022-10-04    38542.147312
2022-10-05    38336.586724
2022-10-06    40028.118932
2022-10-07    42247.574196
2022-10-08    43252.974864
2022-10-09    42236.621732
2022-10-10    39916.027229
2022-10-11    37976.705961
2022-10-12    37798.956381
2022-10-13    39430.114480
2022-10-14    41567.191851
2022-10-15    42549.135408
2022-10-16    41603.253940
2022-10-17    39410.638204
2022-10-18    37577.178487
2022-10-19    37420.482620
2022-10-20    38988.788908
```

```
2022-10-21    41042.091390
2022-10-22    41996.500738
2022-10-23    41112.821667
2022-10-24    39037.551778
2022-10-25    37300.706315
2022-10-26    37160.036535
2022-10-27    38664.488881
2022-10-28    40633.997116
2022-10-29    41558.251185
2022-10-30    40730.106159
2022-10-31    38763.205419
Freq: D, Name: predicted_mean, dtype: float64
```

```
from matplotlib import pyplot
pyplot.plot(pred_future,color='red')
```

```
[<matplotlib.lines.Line2D at 0x24afd1c5c40>]
```

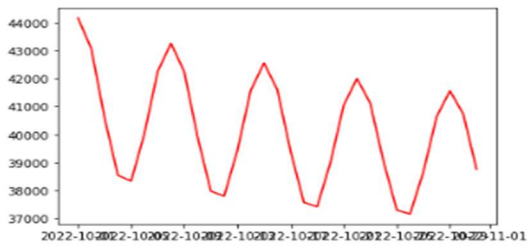
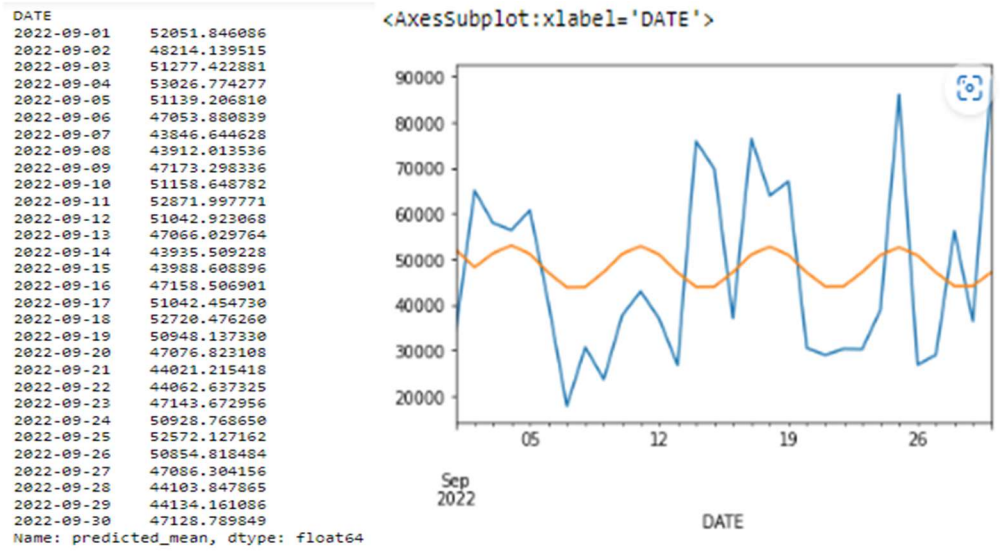


Figure 7.21 future predictions using ARIMA model

7.2.4 SARIMA Model (Seasonal Auto Regressive Integrated Moving Average)

Model:

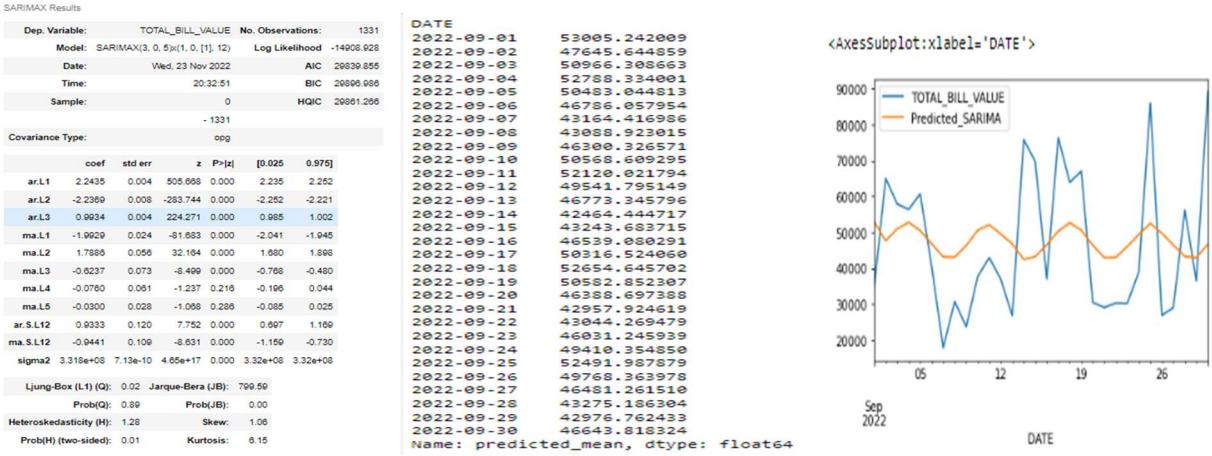
For Seasonal, I consider P, D, Q as zeros and the results are shown in the below figure 7.22.



```
DATE
2022-09-01    52051.846086
2022-09-02    48214.139515
2022-09-03    51277.422881
2022-09-04    53026.774277
2022-09-05    51139.206810
2022-09-06    47053.880839
2022-09-07    43846.644628
2022-09-08    43912.013536
2022-09-09    47173.298336
2022-09-10    51158.648782
2022-09-11    52871.997771
2022-09-12    51042.923068
2022-09-13    47066.029764
2022-09-14    43935.509228
2022-09-15    43988.608896
2022-09-16    47158.506901
2022-09-17    51042.454730
2022-09-18    52720.476260
2022-09-19    50948.137330
2022-09-20    47076.823108
2022-09-21    44021.215418
2022-09-22    44062.637325
2022-09-23    47143.672956
2022-09-24    50928.768650
2022-09-25    52572.127162
2022-09-26    50854.818484
2022-09-27    47086.304156
2022-09-28    44103.847865
2022-09-29    44134.161086
2022-09-30    47128.789849
Name: predicted_mean, dtype: float64
```


Figure 7.22 predictions on test data and the plot of actuals and predicted sales

Now, I consider seasonal parameters P and Q as 1 and D as zero and the results are shown in the below figure 7.23.



SARIMAX Results

Dep. Variable:	TOTAL_BILL_VALUE	No. Observations:	1331
Model:	SARIMAX(3, 0, 5)x(1, 0, [1], 12)	Log Likelihood	-14908.928
Date:	Wed, 23 Nov 2022	AIC	29839.855
Time:	20:32:51	BIC	29896.986
Sample:	0	HQIC	29861.266
	- 1331		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	2.2435	0.004	505.668	0.000	2.235	2.252
ar.L2	-2.2369	0.008	-283.744	0.000	-2.252	-2.221
ar.L3	0.9934	0.004	224.271	0.000	0.985	1.002
ma.L1	-1.9929	0.024	-81.683	0.000	-2.041	-1.945
ma.L2	1.7886	0.056	32.164	0.000	1.680	1.898
ma.L3	-0.6237	0.073	-8.499	0.000	-0.768	-0.480
ma.L4	-0.0760	0.061	-1.237	0.216	-0.196	0.044
ma.L5	-0.0300	0.028	-1.068	0.286	-0.085	0.025
ar.S.L12	0.9333	0.120	7.752	0.000	0.697	1.169
ma.S.L12	-0.9441	0.109	-8.631	0.000	-1.159	-0.730
sigma2	3.318e+08	7.13e-10	4.65e+17	0.000	3.32e+08	3.32e+08

Ljung-Box (L1) (Q):	0.02	Jarque-Bera (JB):	799.59
Prob(Q):	0.89	Prob(JB):	0.00
Heteroskedasticity (H):	1.28	Skew:	1.06
Prob(H) (two-sided):	0.01	Kurtosis:	6.15

```
DATE
2022-09-01    53005.242009
2022-09-02    47645.644859
2022-09-03    50966.308663
2022-09-04    52788.334001
2022-09-05    50483.044813
2022-09-06    46786.057954
2022-09-07    43164.416986
2022-09-08    43088.923015
2022-09-09    46300.326571
2022-09-10    50568.609295
2022-09-11    52120.021794
2022-09-12    49541.795149
2022-09-13    46773.345796
2022-09-14    42464.444717
2022-09-15    43243.683715
2022-09-16    46539.080291
2022-09-17    50316.524060
2022-09-18    52654.645702
2022-09-19    50582.852307
2022-09-20    46388.697388
2022-09-21    42957.924619
2022-09-22    43044.269479
2022-09-23    46031.245939
2022-09-24    49410.354850
2022-09-25    52491.987879
2022-09-26    49768.363978
2022-09-27    46481.261510
2022-09-28    43275.186304
2022-09-29    42976.762433
2022-09-30    46643.818324
Name: predicted_mean, dtype: float64
```


Figure 7.23 results of SARIMA (3, 0, 5) (1, 0, 1, 12) model

The below figure 7.24 shows the future predictions

```
2022-10-01    51985.872011
2022-10-02    50071.171804
2022-10-03    46243.921687
2022-10-04    42893.036269
2022-10-05    42294.896890
2022-10-06    46603.371493
2022-10-07    49569.727710
2022-10-08    51928.718052
2022-10-09    50212.893263
2022-10-10    46130.334246
2022-10-11    43554.004607
2022-10-12    43336.937883
2022-10-13    46240.169694
2022-10-14    49827.884233
2022-10-15    51567.160121
2022-10-16    49705.161932
2022-10-17    45444.008390
2022-10-18    43603.024717
2022-10-19    42759.999293
2022-10-20    46293.817197
2022-10-21    49936.711086
2022-10-22    51324.510253
2022-10-23    50174.206609
2022-10-24    46367.237369
2022-10-25    43348.396721
2022-10-26    43153.607435
2022-10-27    46061.239799
2022-10-28    49440.383170
2022-10-29    50550.653243
2022-10-30    50075.289646
2022-10-31    45780.111106
Freq: D, Name: predicted_mean, dtype: float64
```

Figure 7.24 future predictions using SARIMA (3, 0, 5) (1, 0, 1, 12) model

7.4 Recurrent Neural Networks

Before applying RNN models, first we need to split the data into two train and test data frames as shown in below figure 7.25.

```
train = sorted_df3.iloc[:1200]
test = sorted_df3.iloc[1200:]
```

Figure 7.25 train and test split

Now we need to convert the sales data in the range 0 to 1 using MinMaxScaler as below figure 7.26.

```
scaler.fit(train)
scaled_train = scaler.transform(train)
scaled_test = scaler.transform(test)
```

```
scaled_train[:10]
```

```
array([[0.26216403],
       [0.25306012],
       [0.23727765],
       [0.31970355],
       [0.22347243],
       [0.40809679],
       [0.20855767],
       [0.13805219],
       [0.1080662 ],
       [0.16168678]])
```

Figure 7.26 converting data using minmaxscalar

After implementing the model we have the model with transformed values, now need to convert the sales values using inverse transform function as in figure 7.27.

```
y_test = scaler_y.inverse_transform(y_test)
y_train = scaler_y.inverse_transform(y_train)
```

Figure 7.27 inverse transform

7.4.1 Long Short Term Memory (LSTM)

The below figure 7.28 shows the plot between actuals and predictions using LSTM

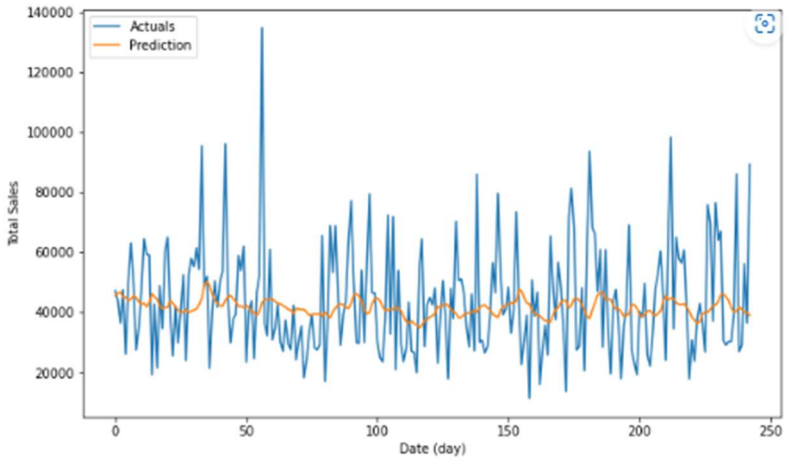


Figure 7.28 plotting test data and predictions using LSTM

7.4.2 Gated Recurrent Unit or GRU

The below figure 7.29 shows the plot between actuals and predictions using GRU

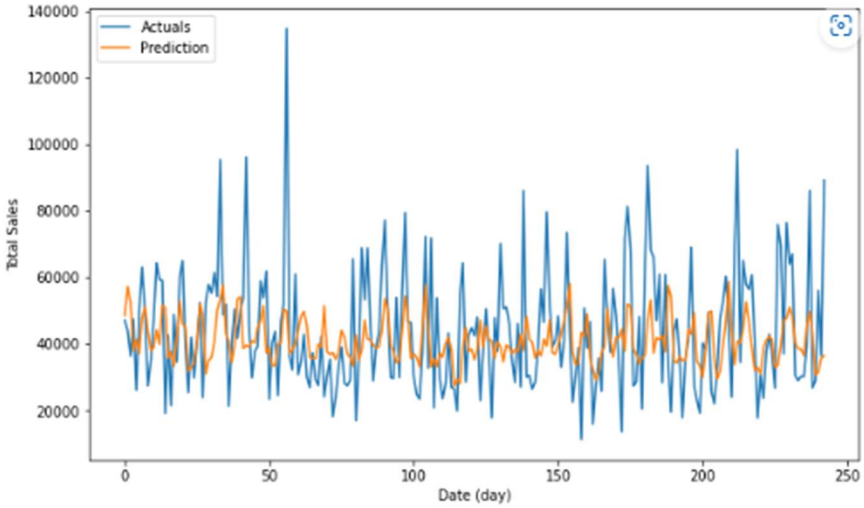


Figure 7.29 plotting test data and predictions using GRU

7.4.3 Bidirectional LSTM

The below figure 7.30 shows the plot between actuals and predictions using Bi-LSTM

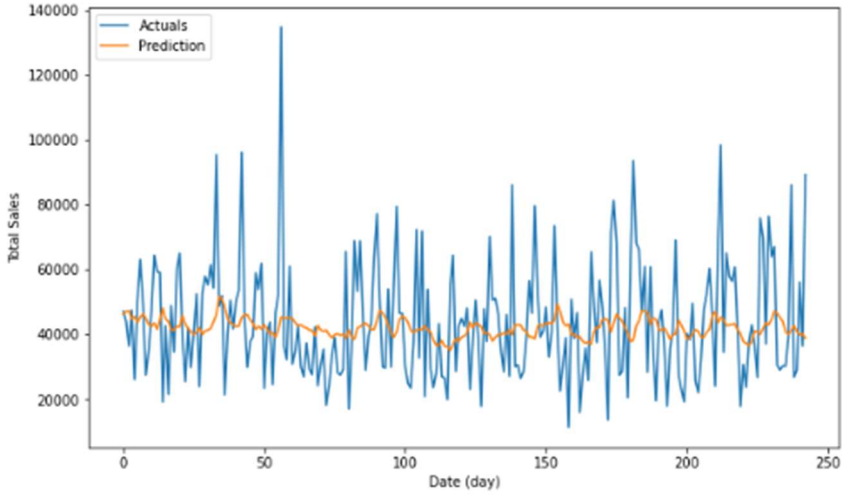


Figure 7.30 plotting test data and predictions using Bi-LSTM

7.5 Forecasting sales using Tableau

It is necessary to analyse time series to understand the data. Due to the fluctuating nature of currency and sales, Tableau Time Series Analysis is frequently used by the retail, financial, and economic sectors. Tableau Time Series Analysis is a very good example of stock market analysis with automated trading algorithms. With Tableau, time series analysis is as easy as dragging and dropping. The dataset is loaded in to the tableau desktop and by using the columns in the tables we can create all the visualizations by dragging the column names into rows and columns. The below figure 7.31 shows the plot of monthly sales where the sum function is applied on total bill value column which is dragged to rows (means y-axis in the plot) and Month in the date was selected and dragged to columns (x-axis in the plot).

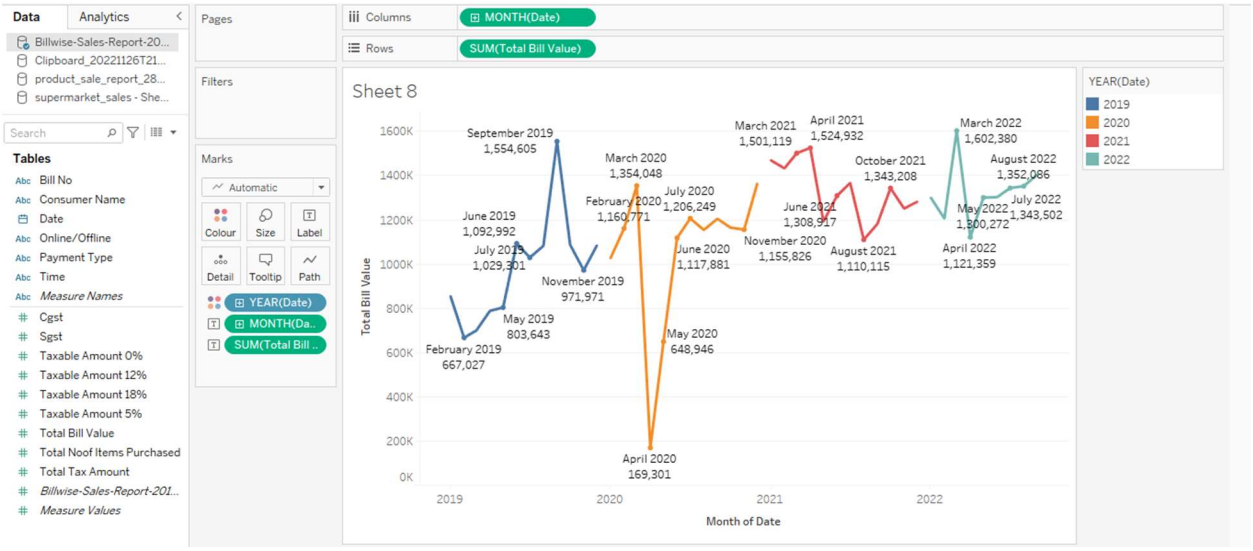


Figure 7.31 Monthly sales using Tableau

We can apply machine learning techniques like regression, forecasting, clustering etc. from the Analytics pane. The below figure 7.32 shows the forecast from September 2022 to December 2023.

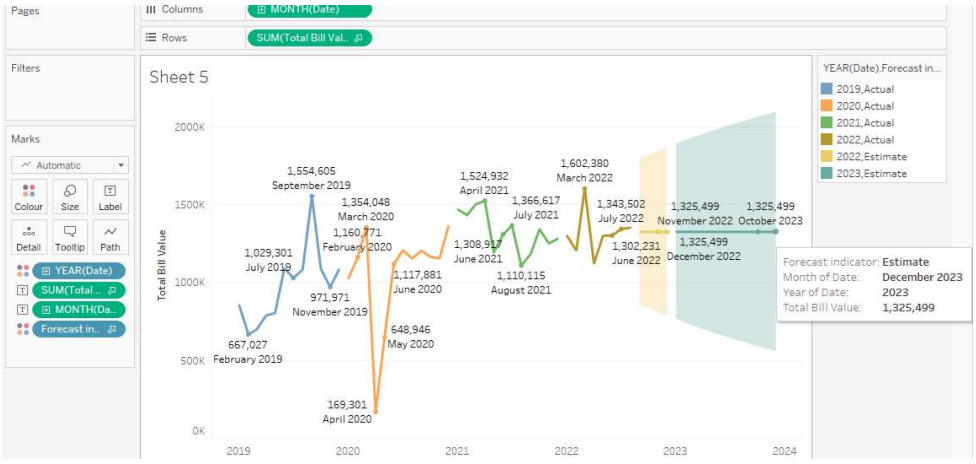


Figure 7.32 forecasting next year sales

We can change the forecasting period, trend and seasonality options using forecast options as shown in the below figure 7.33.

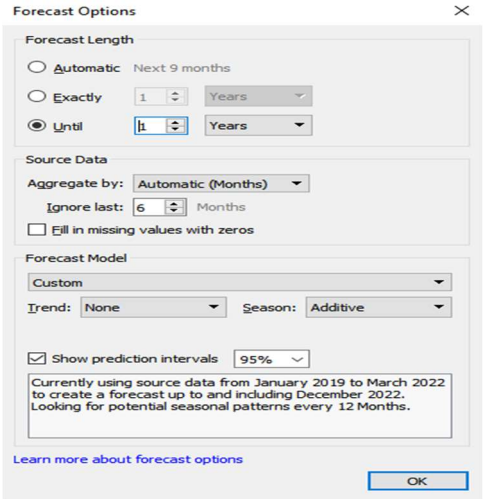


Figure 7.33 adding seasonality in forecasting options

The below figure 7.34 shows the forecast after changing the options by adding seasonality.

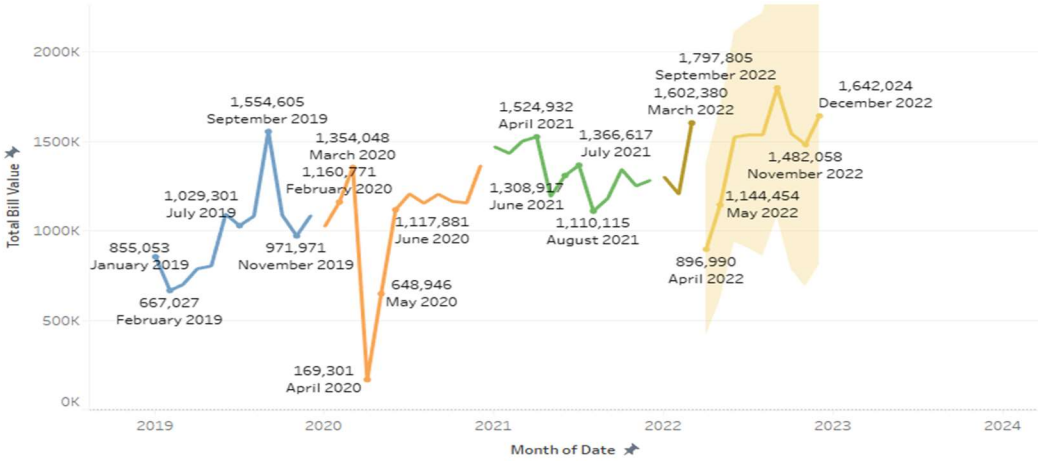


Figure 7.34 forecast of monthly sales with seasonality

The below figure 7.35 shows the results of forecast. MASE is the Mean Absolute Square Error is the performance metrics. It should be in the range 0 to 1 and the best model will be the one with less MASE.

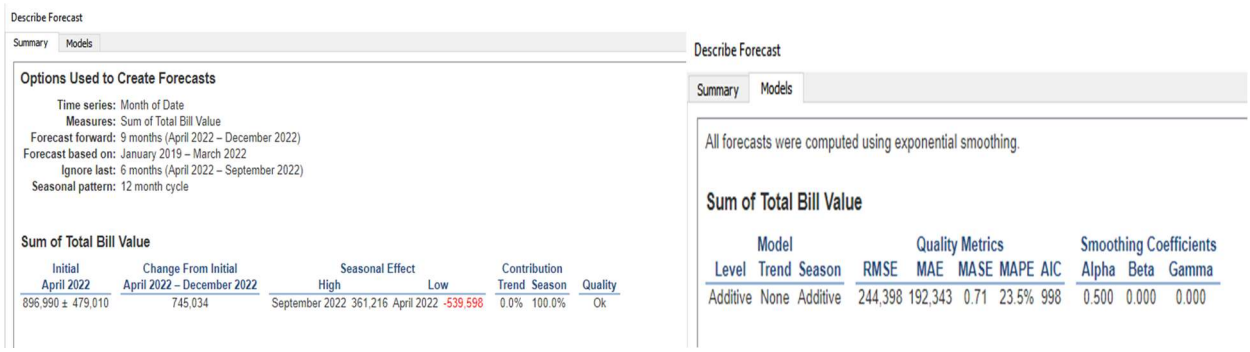


Figure 7.35 forecasting results

The below figure 7.36 shows the changes in forecasting options. Season is multiplicative and trend is additive then there is poor quality of model performance and MASE is 1.11 which is of bad performance as shown in the figure 7.37.

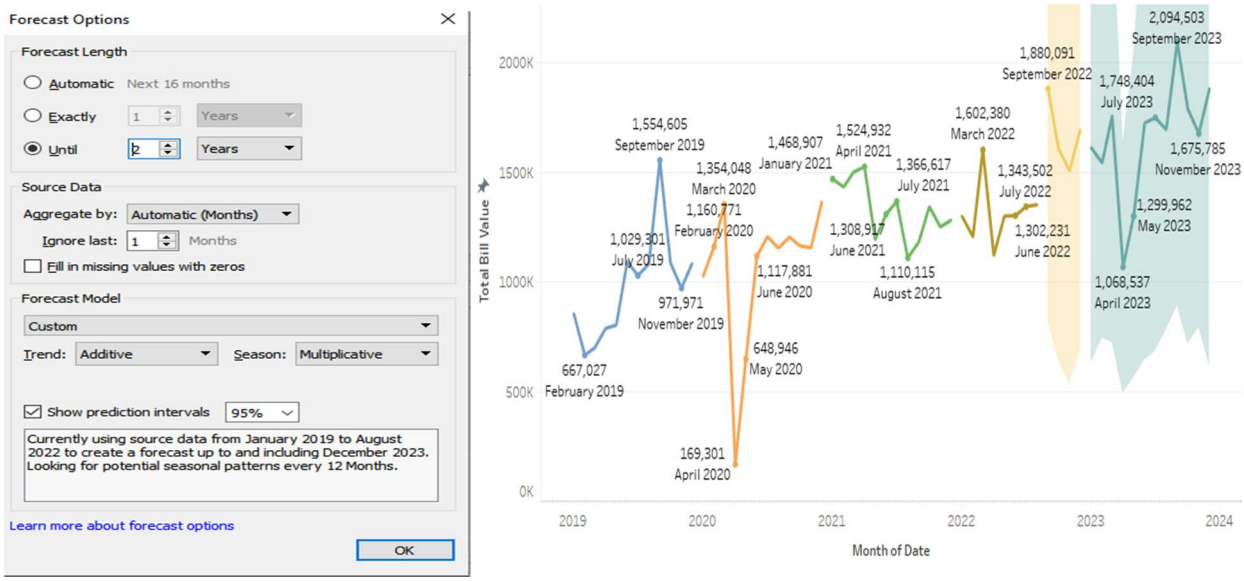


Figure 7.36 additive trend and multiplicative seasonality

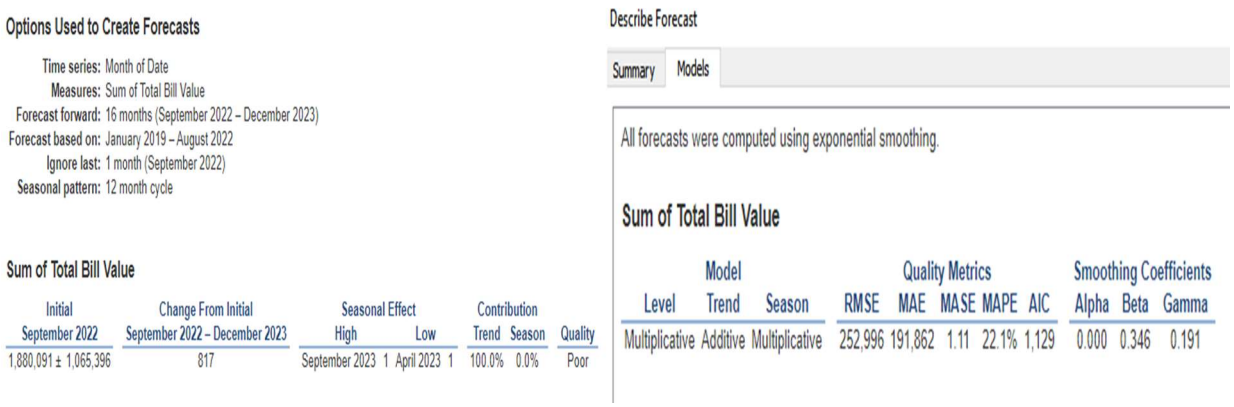


Figure 7.37 Results of additive trend and multiplicative seasonality

The below figure 7.38 shows the changes of forecast with Season is multiplicative and we don't have trend in our data so I choose non for trend and the results are shown in figure 7.39.

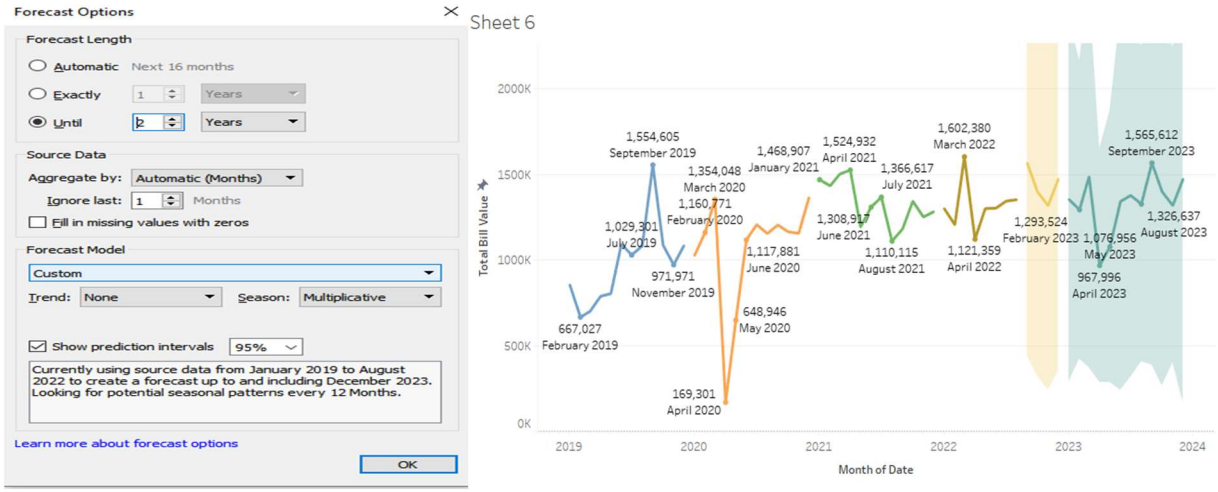


Figure 7.38 forecasting multiplicative seasonality

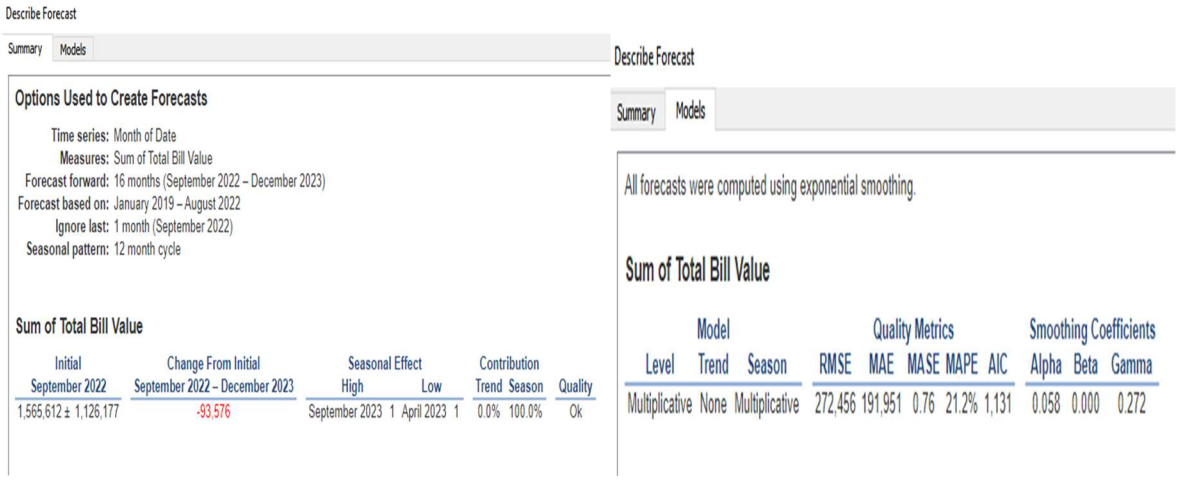


Figure 7.39 results of multiplicative seasonality

The below figure 7.40 shows the changes of forecast with Season as additive.

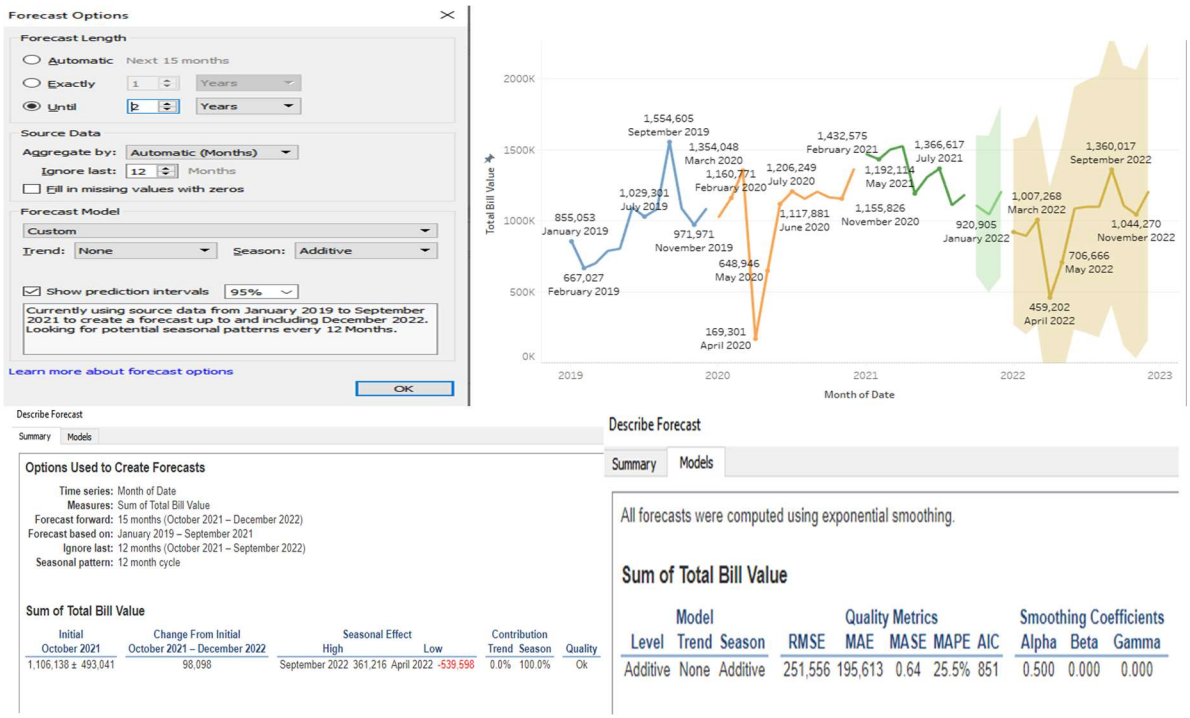


Figure 7.40 forecasting plots and results of additive seasonality

8 Results

In this section, the outcomes of regression models, ARIMA models, and recurrent neural network models were presented. Below figure 8.1 shows the results of the implemented four Regression models.

```
print('Mean Squared Error for Linear Regression Model is:',rmse_lr)
print('Mean Absolute Error for Linear Regression Model is:',mae_lr)
```

```
Mean Squared Error for Linear Regression Model is: 20922.4629299157
Mean Absolute Error for Linear Regression Model is: 16444.800471961942
```

```
rmse_rf=sqrt(mean_squared_error(pred,y_test))
print('Mean Squared Error for Random Forest Model is:',rmse_rf)
```

```
Mean Squared Error for Random Forest Model is: 22101.703245631434
```

```
from sklearn.metrics import mean_absolute_error as mae
mae_rf = mae(y_test,pred)
print('Mean Absolute Error for Random Forest Regression Model is:',mae_rf)
```

```
Mean Absolute Error for Random Forest Regression Model is: 17388.626300000004
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse_dt=sqrt(mean_squared_error(ydt_pred,y_test))
print('Mean Squared Error for Decission Tree Model is:',rmse_dt)
```

```
Mean Squared Error for Decission Tree Model is: 27666.577699606918
```

```
from sklearn.metrics import mean_absolute_error as mae
mae_dt = mae(y_test,ydt_pred)
print('Mean Absolute Error for Decission Tree Regression Model is:',mae_dt)
```

```
Mean Absolute Error for Decission Tree Regression Model is: 21729.38466666667
```

```
Mean Squared Error for XG Boost Model is: 21039.49274023751
```

```
from sklearn.metrics import mean_absolute_error as mae
mae_xgb = mae(y_test,yxgb_pred)
print('Mean Absolute Error for XG Boost Regression Model is:',mae_xgb)
```

```
Mean Absolute Error for XG Boost Regression Model is: 15967.951919270834
```

Figure 8.1 Results of regression models

Below figure 8.2 shows the results of the implemented ARIMA models.

```
#calculate error:
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse_AR=sqrt(mean_squared_error(pred,test))
print('Mean Squared Error for AR Model is:',rmse_AR)
```

```
Mean Squared Error for AR Model is: 21196.820823333055
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt
test['TOTAL_BILL_VALUE'].mean()
rmse_MA=sqrt(mean_squared_error(pred,test['TOTAL_BILL_VALUE']))
print('Mean Squared Error for MA Model is:',rmse_MA)
```

```
Mean Squared Error for MA Model is: 21415.826489972504
```

```
from sklearn.metrics import mean_absolute_error as mae
mae_AR = mae(test,pred)
print('Mean Absolute Error for AR Model is:',mae_AR)
```

```
Mean Absolute Error for AR Model is: 16121.98369843206
```

```
from sklearn.metrics import mean_absolute_error as mae
mae_ma = mae(test['TOTAL_BILL_VALUE'],pred)
print('Mean Absolute Error for MA Model is:',mae_ma)
```

```
Mean Absolute Error for MA Model is: 16417.212182975705
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt
test['TOTAL_BILL_VALUE'].mean()
rmse_ARIMA=sqrt(mean_squared_error(pred,test['TOTAL_BILL_VALUE']))
print('Mean Squared Error for ARIMA Model is:',rmse_ARIMA)
mae_ARIMA = mae(test['TOTAL_BILL_VALUE'],pred)
print('Mean Absolute Error for ARIMA Model is:',mae_ARIMA)
```

```
Mean Squared Error for ARIMA Model is: 19722.18422951012
Mean Absolute Error for ARIMA Model is: 16501.99637979555
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt
test['TOTAL_BILL_VALUE'].mean()
rmse_SARIMA=sqrt(mean_squared_error(s_pred,test['TOTAL_BILL_VALUE']))
print('Mean Squared Error for SARIMA Model is:',rmse_SARIMA)
mae_SARIMA = mae(test['TOTAL_BILL_VALUE'],s_pred)
print('Mean Absolute Error for SARIMA Model is:',mae_SARIMA)
```

```
Mean Squared Error for SARIMA Model is: 19071.87378334275
Mean Absolute Error for SARIMA Model is: 16875.492123665277
```

Figure 8.2 Results of ARIMA models

Below figure 8.3 shows the results of the implemented Recurrent Neural Network models.

```
Bidirectional LSTM:
Mean Absolute Error: 14169.7534
Root Mean Square Error: 18635.3887

LSTM:
Mean Absolute Error: 14354.5546
Root Mean Square Error: 18712.0314

GRU:
Mean Absolute Error: 13734.4055
Root Mean Square Error: 17958.0938
```

Figure 8.3 Results of Recurrent Neural Network models

Below table 8.4 shows the all results of implemented models.

Forecasting Models	Model	Root Mean Square Error	Mean Absolute Error
Machine Learning Models - Regression	Linear Regression	20922.46	16444.800
	Random Forest Regression	22101.70	17388.62
	Decision Tree Regression	27666.577	21729.384
	XGBoost Regression	21039.49	15967.951
Traditional Statistical Models-ARIMA	AR Model	21196.820	16121.983
	MA Model	21415.82	16417.212
	ARIMA Model	19722.184	16501.996
	SARIMA MODEL	19071.873	16875.492
Deep Learning Models-Recurrent Neural Network	LSTM	18712.0314	14354.5546
	GRU	17958.0938	13734.4055
	BI-LSTM	18635.3887	14169.7534

Table 8.4 Results of Time Series Forecasting

The below figure 8.5 shows the results of forecasting using tableau. Out of all the implemented models the below model shows the better results as we got MASE 0.64 which is less than 1.

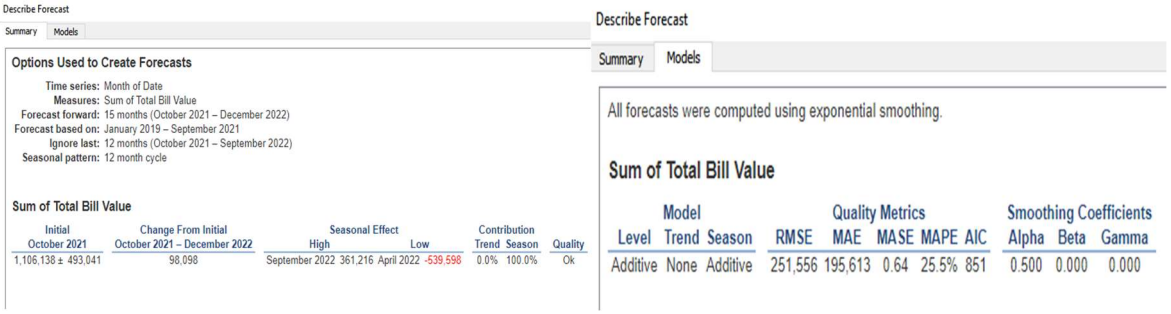


Figure 8.5 results of Tableau forecasting

9 Project Management

The Gantt chart is used to depict the project's schedule. The tasks can be broken down into topic selection, reviewing the available documentation, putting the model into practise, comparing the model, and report writing. Before beginning the work, each task was planned. The study took twelve weeks to complete, starting with the topic search and ending with the submission. The first four weeks were planned and used for topic selection, proposal writing, problem formulation, and ethical application submission. The next step involved reading research and becoming familiar with machine learning methods. After that, the data was gathered, cleaned up, and made suitable for the implementation of the chosen architectures. The time frame for this phase was exceeded. The minor adjustments persisted through the tenth week. In order to finish the documentation on time and within budget, it was begun concurrently with the models' finalisation. The project was essentially finished within the specified timeline with a few minor modifications. On December 9th, 2022, the report submission date will be completed.

9.1 Project Schedule

The below figure 9.1 shows the Gantt chart of this project.

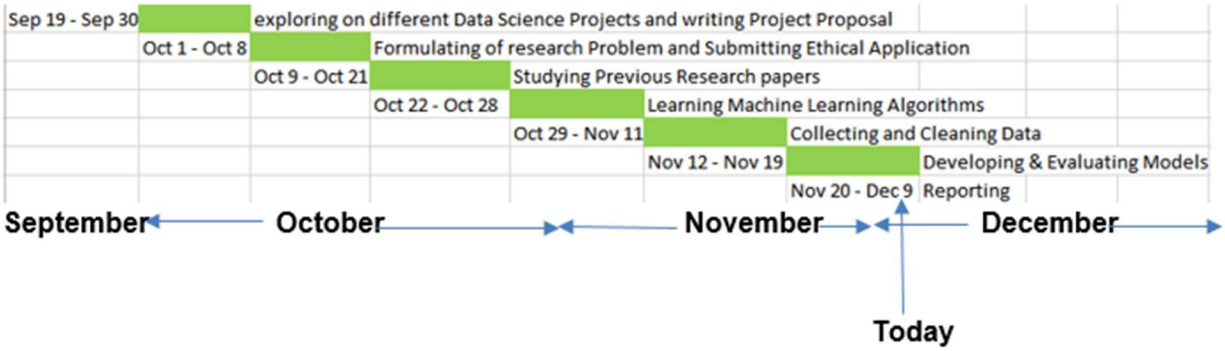


Figure 9.1 Gantt chart Showing Schedule

9.2 Risk Management

Three categories can broadly be used to classify risk management. One is the risk of scope, which refers to anything that is still within the project's scope. With the exception of the hybrid model implementation, this project's entire scope was met. The second risk is time risk, which is related to the project not finishing on time and not having enough time to complete all of the report's goals and objectives. The time risk in this dissertation was kept to a minimum because the report and all of the tasks were finished on time. The last one is project-related resource risks. The management shared access to data storage

during the project's early stages, which was the only resource needed. Therefore, there was no actual resource risk.

9.3 Social, Legal, Ethical and Professional Considerations

The model was applied to data that was gathered from a retail establishment for research purposes. Secondary data is used. There are no social, legal, ethical, or professional considerations, as a result. It doesn't have any private information in it.

10 Conclusion

The main goal of this study is to identify the prediction methods used to assess whether it is possible to predict sales accurately. Utilizing the techniques discussed in this article, it is possible to determine a predictive analysis of a retail store by analysing data using a variety of supervised machine learning techniques, traditional statistical models, and deep learning models. On the basis of Mean Absolute Error and Root Mean Square Error, the effectiveness of all algorithms was compared. The best model can be selected from those with lower values. Out of these three forecasting models, Deep Learning Models performed well and in particular we got less values for Mean Absolute Error and Root Mean Square Error for Gated Recurrent Unit (GRU) for Prakruthivanam Store Sales. For retailers to understand the various aspects of the business plan, the findings could be very helpful. They can use these conclusions to make wise decisions and advance their business.

10.1 Future Work

To achieve better predictions in this study, hybrid models that combine deep learning and traditional statistical models must also be used in addition to the applied models. This study can be expanded by selecting the best algorithm from the used models, and this algorithm must then be integrated with the Prakruthivanam billing software in order to update daily sales automatically at the end of every day and the future sales to be forecasted for the upcoming month. Also need to study more on product wise sales and concentrate on slow moving items.

10.2 Student Reflections

I have learned a lot throughout this project. At first, I faced many obstacles because I wanted my project to use real-time data in a novel and distinctive way. The biggest challenge I've faced since deciding to work on time series forecasting is obtaining real-time data. I contacted a few organisations via phone and email. At first, I received some rejections; later, some of them agreed but didn't respond; however, I persisted, and in the end, Prakruthivanam Store agreed and assisted me with the data I required. To get all the permissions, including the attestation on the consent form, took me close to a month. At the same time, I began learning forecasting methods separately. Since I had never studied deep learning or conventional statistical models before and hadn't had the opportunity to do so in my previous modules, it took me some time to learn and implement

them. In my opinion all of the course's modules, were very helpful to me in finishing this project, by quickly correcting any errors, I was able to implement and complete the methods successfully. I started reporting simultaneously, which took me more than a month, but because I use a systematic approach and plan out all of my work, I was able to finish everything on schedule. Since Bill Wise Sales already covered daily sales, I didn't work on it. I believe that if the sales report and the product wise sales report were linked, I could work much more effectively on the product wise sales report. However, I have already completed some basic analysis. This project will definitely aid me in gaining valuable practical experience with machine learning techniques and teach me time management skills, risk management techniques, and by considering the ethical and social impacts associated with any project.

11 References

ARIMA. (n.d.). Retrieved from https://otexts.com/fpp2/arima.html

bilstmLayer. (n.d.). Retrieved from Mathworks: https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.bilstmlayer.html

C. CATAL, K. E. (2019). Benchmarking of Regression and Time Series Analysis Techniques for Sales Forecasting. *BALKAN JOURNAL OF ELECTRICAL & COMPUTER ENGINEERING, 7*. Retrieved from https://dergipark.org.tr/en/download/article-file/637927

Hamilton, T. (2022). *Agile Methodology*. Retrieved from guru99 : https://www.guru99.com/agile-scrum-extreme-testing.html

Has¸im Sak, A. S. (n.d.). Long Short-Term Memory Recurrent Neural Network Architectures. *arxiv*.

Hong Choon Ong, J. L. (2008). A comparison of neural network methods and Box-Jenkins model in time series analysis. *ResearchGate.*

Jamal Fattah, L. E. (2018, october 30). Forecasting of demand using ARIMA model. *SAGE Journals*. Retrieved from https://journals.sagepub.com/doi/full/10.1177/1847979018808673

LENDAVE, V. (n.d.). *LSTM Vs GRU in Recurrent Neural Network: A Comparative Study*. Retrieved from analyticsindiamag: https://analyticsindiamag.com/lstm-vs-gru-in-recurrent-neural-network-a-comparative-study/

Mona Z. Bahri, S. V. (2022). Time Series Forecasting Using SmoothingEnsemble Empirical Mode Decomposition andMachine Learning Techniques. *Proceedings of the International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. Retrieved from https://www.researchgate.net/publication/364775694_Time_Series_Forecasting_Using_Smoothing_Ensemble_Empirical_Mode_Decomposition_and_Machine_Learning_Techniques

Mrs. Manisha Gahirwal, V. M. (n.d.). Inter Time Series Sales Forecasting. *arxiv.org*. Retrieved from https://arxiv.org/ftp/arxiv/papers/1303/1303.0117.pdf

Pavlyshenko, B. M. (2019). Machine-Learning Models for Sales Time Series Forecasting. *MDPI*. Retrieved from https://www.mdpi.com/2306-5729/4/1/15/htm

Petneh´azi, G. (n.d.). Recurrent Neural Networks for Time SeriesForecasting. *arxiv*. Retrieved from Recurrent Neural Networks for Time SeriesForecasting

Random Forest Regression in Python. (n.d.). Retrieved from greeksforgreeks: https://www.geeksforgeeks.org/random-forest-regression-in-python/

Ratnadip Adhikari, R. (2013, January). An Introductory Study on Time series Modeling and Forecasting. *ResearchGate*. Retrieved from https://www.researchgate.net/publication/235219651_An_Introductory_Study_on_Time_series_Modeling_and_Forecasting

SARIMA Using Python – Forecast Seasonal Data. (n.d.). Retrieved from https://www.wisdomgeek.com/development/machine-learning/sarima-forecast-seasonal-data-using-python/#:~:text=SARIMA%20is%20a%20widely%20used,weather%20forecast%20over%20several%20years.

The linear model. (n.d.). Retrieved from otexts: https://otexts.com/fpp2/regression-intro.html

TIME SERIES AR,MA. (n.d.). Retrieved from https://www.bauer.uh.edu/rsusmel/phd/ec2-3.pdf

Time series forecasting methods. (n.d.). Retrieved from influx data: https://www.influxdata.com/time-series-forecasting-methods/#:~:text=Types%20of%20time%20series%20methods%20used%20for%20forecasting&text=Common%20types%20include%3A%20Autoregression%20(AR,Moving%2DAverage%20(SARIMA).

XGBoost for Regression. (n.d.). Retrieved from greeksforgreeks: https://www.geeksforgeeks.org/xgboost-for-regression/?ref=gcse

Yihang Zhu, Y. Z. (2019). Spring onion seed demand forecasting using a hybrid Holt-Winters and support vector machine model. *PLOS ONE Journals*. Retrieved from https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0219889

Zhang, G. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *ScienceDirect, 50*, 159-175. Retrieved from https://www.sciencedirect.com/science/article/pii/S0925231201007020

Appendix A – Code and Datasets Link:

PVS_PROJECT_IMPLEMENTATION_Regression_models code:

```
#Exploratory Data Analysis
#importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#read the dataset
Billwise_sales = pd.read_csv("Billwise-Sales-Report-2019-2022.csv")
Billwise_sales.head(10)
Billwise_sales.shape
Billwise_sales.describe()
Billwise_sales.dtypes
Billwise_sales.isnull().sum()
Billwise_sales = pd.read_csv("Billwise-Sales-Report-2019-2022.csv")
Billwise_sales['DATE'] = pd.to_datetime(Billwise_sales['DATE'], format =
"%d/%m/%Y")
Billwise_sales['TIME'] =
pd.to_datetime(Billwise_sales['TIME'],format="%I:%M%p").dt.strftime('%H:%M')
Billwise_sales['month_year'] =
pd.to_datetime(Billwise_sales['DATE']).dt.to_period('M')
Billwise_sales['Hour'] = pd.to_datetime(Billwise_sales['TIME']).dt.hour
Billwise_sales
Billwise_sales['Dayofweek']= Billwise_sales['DATE'].dt.day_name()
Billwise_sales
plt.figure(figsize=(30,25))
sns.countplot(x='month_year',data=Billwise_sales, palette='rainbow')
plt.title("MONTHLY SALES")
Billwise_sales['month_year'].max()
Billwise_sales['month_year'].min()
Billwise_sales['month_year'].value_counts()
Billwise_sales['PAYMENT_TYPE'].value_counts()
Billwise_sales['ONLINE/OFFLINE'].value_counts()
sns.countplot("ONLINE/OFFLINE", data=Billwise_sales)
sns.countplot("PAYMENT_TYPE", data=Billwise_sales)
Billwise_sales[['PAYMENT_TYPE','TOTAL_BILL_VALUE']].groupby(['PAYMENT_TYPE'])
.sum()
Billwise_sales['PAYMENT_TYPE'].value_counts()
Billwise_sales.groupby('Dayofweek')['TOTAL_BILL_VALUE','TOTAL_NOOF_ITEMS_PURC
HASED'].sum()
Billwise_sales['Dayofweek']= Billwise_sales['DATE'].dt.day_name()
plt.figure(figsize=(8,5))
sns.countplot(x='Dayofweek',data=Billwise_sales, palette='rainbow')
plt.title("DAY OF THE WEEK")
Billwise_sales['Year'] = pd.DatetimeIndex(Billwise_sales['DATE']).year
plt.figure(figsize=(8,5))
sns.countplot(x='Year',data=Billwise_sales, palette='rainbow')
plt.title("YEARLY SALES")
plt.figure(figsize=(20,15))
sns.countplot(x='TOTAL_NOOF_ITEMS_PURCHASED',data=Billwise_sales,
palette='rainbow')
plt.title("TOTAL NO OF ITEMS PURCHASED")
Billwise_sales
Billwise_sales['DATE'].min()
```

```
Billwise_sales['DATE'].max()
Billwise_sales['DATE'].max() - Billwise_sales['DATE'].min()
filt = (Billwise_sales['DATE'] >= pd.to_datetime('2019-01-01')) &
(Billwise_sales['DATE'] < pd.to_datetime('2020-01-01'))
Billwise_sales.loc[filt]
filt = (Billwise_sales['DATE'] >= pd.to_datetime('2020-01-01')) &
(Billwise_sales['DATE'] < pd.to_datetime('2021-01-01'))
Billwise_sales.loc[filt]
filt = (Billwise_sales['DATE'] >= pd.to_datetime('2021-01-01')) &
(Billwise_sales['DATE'] < pd.to_datetime('2022-01-01'))
Billwise_sales.loc[filt]
filt = (Billwise_sales['DATE'] >= pd.to_datetime('2022-01-01')) &
(Billwise_sales['DATE'] <= pd.to_datetime('2022-09-30'))
Billwise_sales.loc[filt]
sorteddf =
Billwise_sales.groupby(['DATE','Hour'])['TOTAL_BILL_VALUE','TOTAL_NOOF_ITEMS_
PURCHASED'].sum()
sorteddf

sorteddf['TOTAL_BILL_VALUE'].plot(figsize=(20,10))
sorted_df2 =
Billwise_sales.groupby('DATE')['TOTAL_BILL_VALUE','TOTAL_NOOF_ITEMS_PURCHASED
'].sum().reset_index()
sorted_df2
Billwise_sales.groupby('month_year')['TOTAL_BILL_VALUE','TOTAL_NOOF_ITEMS_PUR
CHASED'].sum()
Billwise_sales =
Billwise_sales[['month_year','TOTAL_BILL_VALUE','TOTAL_NOOF_ITEMS_PURCHASED']
].groupby('month_year').sum()
Billwise_sales.reset_index(inplace = True)
Billwise_sales
Billwise_sales.plot.line(x = 'month_year', y = 'TOTAL_BILL_VALUE',
                         color = 'salmon', title = 'Monthly Sales')

ts_df = sorted_df2.set_index('DATE')
ts_df.index
sns.set()
plt.ylabel('Total Sale')
plt.xlabel('Date')
plt.xticks(rotation=45)
plt.plot(ts_df.index, ts_df['TOTAL_BILL_VALUE'],)

train = ts_df[ts_df.index < pd.to_datetime("2022-01-01", format='%Y-%m-%d')]
test = ts_df[ts_df.index > pd.to_datetime("2022-01-01", format='%Y-%m-%d')]

plt.plot(train, color = "black")
plt.plot(test, color = "red")
plt.ylabel('SALE')
plt.xlabel('Date')
plt.xticks(rotation=45)
plt.title("Train/Test split for PVS Data")
plt.show()
tsdf2 = sorted_df2.groupby('DATE')['TOTAL_BILL_VALUE'].sum().reset_index()
tsdf2
tsdf1 =
sorted_df2.groupby('DATE')['TOTAL_NOOF_ITEMS_PURCHASED'].sum().reset_index()
tsdf1

tsdf1 = tsdf1.set_index('DATE')
```

```
tsdf1.index
tsdf2 = tsdf2.set_index('DATE')
tsdf2.index
y1 = tsdf1['TOTAL_NOOF_ITEMS_PURCHASED'].resample('MS').mean()
y2 = tsdf2['TOTAL_BILL_VALUE'].resample('MS').mean()
y1.plot(figsize=(15, 6))
plt.show()
y2.plot(figsize=(15, 6))
plt.show()
from pylab import rcParams
import statsmodels.api as sm
rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(y1, model='additive')
fig = decomposition.plot()
plt.show()

decomposition2 = sm.tsa.seasonal_decompose(y2, model='additive')
fig = decomposition2.plot()
plt.show()
#Determining rolling statistics
rolmean = y2.rolling(window=7).mean()
rolstd = y2.rolling(window=7).std()
print(rolmean,rolstd)
#plot rolling statistics
orig = plt.plot(y2,color='blue',label='Original')
mean = plt.plot(rolmean,color='red',label='Rolling Mean')
std = plt.plot(rolstd,color='black',label = 'Rolling Std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard Deviation')
plt.show(block=False)
#Mean and standard deviation is stationary
sorted_df2.drop('TOTAL_NOOF_ITEMS_PURCHASED',axis=1,inplace=True)
sorted_df2.set_index('DATE',inplace=True)
sorted_df2
#Determining rolling statistics
rolmean = sorted_df2.rolling(window=365).mean()
rolstd = sorted_df2.rolling(window=365).std()
print(rolmean,rolstd)
#plot rolling statistics
orig = plt.plot(sorted_df2,color='blue',label='Original')
mean = plt.plot(rolmean,color='red',label='Rolling Mean')
std = plt.plot(rolstd,color='black',label = 'Rolling Std')
plt.legend(loc='best')
plt.title('Rolling Mean & Standard Deviation')
plt.show(block=False)
#Mean and standard deviation is stationary
#Testing seasonality and stationarity (dicky fuller test)
from statsmodels.tsa.stattools import adfuller

print('Results of Dickey-Fuller Test:')
dftest = adfuller(sorted_df2['TOTAL_BILL_VALUE'],autolag='AIC')
dfoutput = pd.Series(dftest[0:4],index=['Test Statistics','p-value','#lags
Used','Number of Observations Used'])
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value
print(dfoutput)
# Function to test the stationarity
def test_stationarity(df2):
# Determing rolling statistics
```

```
roll_mean = df2.rolling(window=365).mean()
roll_std = df2.rolling(window=365).std()
# Plotting rolling statistics:
orig = plt.plot(df2.resample('W').mean(), color='blue',label='Original')
mean = plt.plot(roll_mean, color='red', label='Rolling Mean')
std = plt.plot(roll_std, color='green', label = 'Rolling Std')
plt.legend(loc='best')
plt.show(block=False)

# Performing Dickey-Fuller test:
print('Results of Dickey-Fuller Test:')
result = adfuller(df2, autolag='AIC')
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print(key, value)
# Testing stationarity of store type a
test_stationarity(sorted_df2)
## Regression Models
sorted_df2['Previous_day_sale']=sorted_df2['TOTAL_BILL_VALUE'].shift(+1)
sorted_df2['Two_days_before_sale']=sorted_df2['TOTAL_BILL_VALUE'].shift(+2)
sorted_df2['Three_days_before_sale']=sorted_df2['TOTAL_BILL_VALUE'].shift(+3)
sorted_df2=sorted_df2.dropna()
sorted_df2
import numpy as np
x1,x2,x3,y=sorted_df2['Previous_day_sale'],sorted_df2['Two_days_before_sale']
,sorted_df2['Three_days_before_sale'],sorted_df2['TOTAL_BILL_VALUE']
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)
final_x=np.concatenate((x1,x2,x3),axis=1)
print(final_x)
X_train,X_test,y_train,y_test=final_x[:-30],final_x[-30:],y[:-30],y[-30:]
# import the Linear regressor
from sklearn.linear_model import LinearRegression
lin_model=LinearRegression()
lin_model.fit(X_train,y_train)
lin_pred=lin_model.predict(X_test)
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (11,6)
plt.plot(lin_pred,label='Linear_Regression_Predictions')
plt.plot(y_test,label='Actual Sales')
plt.legend(loc="upper left")
plt.show()
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse_lr=sqrt(mean_squared_error(lin_pred,y_test))
from sklearn.metrics import mean_absolute_error as mae
mae_lr = mae(y_test,lin_pred)
print('Mean Squared Error for Linear Regression Model is:',rmse_lr)
print('Mean Absolute Error for Linear Regression Model is:',mae_lr)
# import the Random Forest regressor
from sklearn.ensemble import RandomForestRegressor
rf_model=RandomForestRegressor(n_estimators=100,max_features=3,
random_state=1)
rf_model.fit(X_train,y_train)
pred=rf_model.predict(X_test)
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (12,8)
```

```
plt.plot(pred,label='Random_Forest_Predictions')
plt.plot(y_test,label='Actual Sales')
plt.legend(loc="upper left")
plt.show()
rmse_rf=sqrt(mean_squared_error(pred,y_test))
print('Mean Squared Error for Random Forest Model is:',rmse_rf)
from sklearn.metrics import mean_absolute_error as mae
mae_rf = mae(y_test,pred)
print('Mean Absolute Error for Random Forest Regression Model is:',mae_rf)
# import the Decission tree regressor
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X_train,y_train)
ydt_pred = regressor.predict(X_test)
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (12,8)
plt.plot(ydt_pred,label='Decision_Tree_Regression_Predictions')
plt.plot(y_test,label='Actual Sales')
plt.legend(loc="upper left")
plt.show()
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse_dt=sqrt(mean_squared_error(ydt_pred,y_test))
print('Mean Squared Error for Decission Tree Model is:',rmse_dt)
from sklearn.metrics import mean_absolute_error as mae
mae_dt = mae(y_test,ydt_pred)
print('Mean Absolute Error for Decission Tree Regression Model is:',mae_dt)
!pip install xgboost
import xgboost as xgb
xgb_r = xgb.XGBRegressor(objective ='reg:linear',
                  n_estimators = 10, seed = 123)

xgb_r.fit(X_train,y_train)
yxgb_pred = xgb_r.predict(X_test)
plt.rcParams["figure.figsize"] = (12,8)
plt.plot(yxgb_pred,label='XGBoost_Regression_Predictions')
plt.plot(y_test,label='Actual Sales')
plt.legend(loc="upper left")
plt.show()
rmse_xgb=sqrt(mean_squared_error(y_test, yxgb_pred))
print("Mean Squared Error for XG Boost Model is:",rmse_xgb)

from sklearn.metrics import mean_absolute_error as mae
mae_xgb = mae(y_test,yxgb_pred)
print('Mean Absolute Error for XG Boost Regression Model is:',mae_xgb)
import pandas as pd
#AR,MA,ARMA,ARIMA,SARIMA
Billwise_sales = pd.read_csv("Billwise-Sales-Report-2019-2022.csv")
Billwise_sales['DATE'] = pd.to_datetime(Billwise_sales['DATE'], format =
"%d/%m/%Y")
Billwise_sales['TIME'] =
pd.to_datetime(Billwise_sales['TIME'],format="%I:%M%p").dt.strftime('%H:%M')
Billwise_sales['month_year'] =
pd.to_datetime(Billwise_sales['DATE']).dt.to_period('M')
Billwise_sales['Hour'] = pd.to_datetime(Billwise_sales['TIME']).dt.hour
Billwise_sales
sorted_df3 =
Billwise_sales.groupby('DATE')['TOTAL_BILL_VALUE'].sum().reset_index()
sorted_df3 = sorted_df3.set_index('DATE')
```

```
sorted_df3
import matplotlib.pyplot as plt
from pandas.plotting import autocorrelation_plot
autocorrelation_plot(sorted_df3['TOTAL_BILL_VALUE']).set_xlim([0, 1000])
plt.show()
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig =
sm.graphics.tsa.plot_acf(sorted_df3['TOTAL_BILL_VALUE'].iloc[13:],lags=25,ax=
ax1)
ax2 = fig.add_subplot(212)
fig =
sm.graphics.tsa.plot_pacf(sorted_df3['TOTAL_BILL_VALUE'].iloc[13:],lags=25,ax
=ax2)

#split Dataset into train and test
train = sorted_df3['TOTAL_BILL_VALUE'][:len(sorted_df3)-30]
test = sorted_df3['TOTAL_BILL_VALUE'][len(sorted_df3)-30:]
from statsmodels.tsa.ar_model import AutoReg
model = AutoReg(train,lags=3).fit()
print(model.summary())
print(len(train))
print(len(test))
#Make predictions on test data
start = len(train)
end = len(sorted_df3)-1
pred = model.predict(start=start,end=end,dynamic=False)
pred.index=sorted_df3.index[start:end+1]
from matplotlib import pyplot
pyplot.plot(pred)
pyplot.plot(test,color='red')
print(pred)
#calculate error:
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse_AR=sqrt(mean_squared_error(pred,test))
print('Mean Squared Error for AR Model is:',rmse_AR)

from sklearn.metrics import mean_absolute_error as mae
mae_AR = mae(test,pred)
print('Mean Absolute Error for AR Model is:',mae_AR)
#MA Model
print(sorted_df3.shape)
train=sorted_df3.iloc[:-30]
test=sorted_df3.iloc[-30:]
print(train.shape,test.shape)
import statsmodels.api as sm
model=sm.tsa.arima.ARIMA(train['TOTAL_BILL_VALUE'],order=(0,0,4))
model=model.fit()
model.summary()
#make predictions on test set
start=len(train)
end=len(train)+len(test)-1
pred=model.predict(start=start,end=end,typ='levels')#.rename('ARIMA
Predictions')
pred.index=sorted_df3.index[start:end+1]
print(pred)
```

```
pred.plot(legend=True)
test['TOTAL_BILL_VALUE'].plot(legend=True)
from sklearn.metrics import mean_squared_error
from math import sqrt
test['TOTAL_BILL_VALUE'].mean()
rmse_MA=sqrt(mean_squared_error(pred,test['TOTAL_BILL_VALUE']))
print('Mean Squared Error for MA Model is:',rmse_MA)
from sklearn.metrics import mean_absolute_error as mae
mae_ma = mae(test['TOTAL_BILL_VALUE'],pred)
print('Mean Absolute Error for MA Model is:',mae_ma)
#Make future date predictions uding MA model
import statsmodels.api as sm
model2=sm.tsa.arima.ARIMA(sorted_df3['TOTAL_BILL_VALUE'],order=(0,0,4))
model2=model2.fit()
model2.summary()

#For Future Dates
index_future_dates=pd.date_range(start='2022-10-01',end='2022-10-31')
#print(index_future_dates)
pred=model2.predict(start=len(sorted_df3)+1,end=len(sorted_df3)+31,typ='level
s').rename('MA Predictions')
#print(comp_pred)
pred.index=index_future_dates
print(pred)
!pip install statsmodels

#ARMA modeL:
import statsmodels.api as sm
ARMA_model=sm.tsa.arima.ARIMA(train['TOTAL_BILL_VALUE'],order=(3,0,4))
ARMA_model=ARMA_model.fit()
ARMA_model.summary()
#make predictions on test set
start=len(train)
end=len(train)+len(test)-1
pred=ARMA_model.predict(start=start,end=end,typ='levels')
pred.index=sorted_df3.index[start:end+1]
print(pred)

from matplotlib import pyplot
pyplot.plot(pred,color='black')
pyplot.plot(test,color='red')

from sklearn.metrics import mean_squared_error
from math import sqrt
test['TOTAL_BILL_VALUE'].mean()
rmse_ARIMA=sqrt(mean_squared_error(pred,test['TOTAL_BILL_VALUE']))
print('Mean Squared Error for ARIMA Model is:',rmse_ARIMA)
mae_ARIMA = mae(test['TOTAL_BILL_VALUE'],pred)
print('Mean Absolute Error for ARIMA Model is:',mae_ARIMA)
#Make Future predictions:
#index_future_dates=pd.date_range(start='2022-10-01',end='2022-10-31')

start = len(sorted_df3)+1
end = len(sorted_df3)+31
print('The Future predictions for next month')
pred_future =
ARMA_model.predict(start=len(sorted_df3)+1,end=len(sorted_df3)+31,dynamic=Fal
se)
```

```
pred_future.index=index_future_dates
print(len(pred_future))
pred_future
from matplotlib import pyplot
pyplot.plot(pred_future,color='red')

#from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
pacf = plot_pacf(sorted_df3['TOTAL_BILL_VALUE'],lags=25)
acf = plot_acf(sorted_df3['TOTAL_BILL_VALUE'],lags=25)

#Seasonal ARIMA
import statsmodels.api as sm
model1=sm.tsa.statespace.SARIMAX(train['TOTAL_BILL_VALUE'],order=(3,0,5),seas
onal_order=(0,0,0,12))
SARIMA_results1=model1.fit()
SARIMA_results1.summary()
start=len(train)
end=len(train)+len(test)-1
s_pred1=SARIMA_results1.predict(start=start,end=end,typ='levels')#.rename('AR
IMA Predictions')
s_pred1.index=sorted_df3.index[start:end+1]
print(s_pred1)

test['TOTAL_BILL_VALUE'].plot()
s_pred1.plot()
from sklearn.metrics import mean_squared_error
from math import sqrt
test['TOTAL_BILL_VALUE'].mean()
rmse_SARIMA1=sqrt(mean_squared_error(s_pred1,test['TOTAL_BILL_VALUE']))
print('Mean Squared Error for SARIMA Model is:',rmse_SARIMA1)
mae_SARIMA1 = mae(test['TOTAL_BILL_VALUE'],s_pred1)
print('Mean Absolute Error for SARIMA Model is:',mae_SARIMA1)
#Seasonal ARIMA
import statsmodels.api as sm
model=sm.tsa.statespace.SARIMAX(train['TOTAL_BILL_VALUE'],order=(3,0,5),seaso
nal_order=(1,0,1,12))
SARIMA_results=model.fit()
SARIMA_results.summary()
test
start=len(train)
end=len(train)+len(test)-1
s_pred=SARIMA_results.predict(start=start,end=end,typ='levels')#.rename('ARIM
A Predictions')
s_pred.index=sorted_df3.index[start:end+1]
print(s_pred)
residuals=test['TOTAL_BILL_VALUE']-s_pred
residuals
s_pred.plot(kind='kde')
SARIMA_results.resid.plot(kind='kde')

test['Predicted_SARIMA']=s_pred

test[['TOTAL_BILL_VALUE','Predicted_SARIMA']].plot()

from sklearn.metrics import mean_squared_error
from math import sqrt
test['TOTAL_BILL_VALUE'].mean()
rmse_SARIMA=sqrt(mean_squared_error(s_pred,test['TOTAL_BILL_VALUE']))
```

```
print('Mean Squared Error for SARIMA Model is:',rmse_SARIMA)
mae_SARIMA = mae(test['TOTAL_BILL_VALUE'],s_pred)
print('Mean Absolute Error for SARIMA Model is:',mae_SARIMA)
#Make Future predictions:
#index_future_dates=pd.date_range(start='2022-10-01',end='2022-10-31')

start = len(sorted_df3)+1
end = len(sorted_df3)+31
print('The Future predictions using SARIMA for next month')
pred_future1 =
SARIMA_results.predict(start=len(sorted_df3)+1,end=len(sorted_df3)+31,dynamic
=False)
pred_future1.index=index_future_dates
pred_future1
#LSTM,BILSTM,GRU PVS STORE
sorted_df3['TOTAL_BILL_VALUE'].plot(figsize=(15,10))

sorted_df3[:200].plot(figsize=(15,10))

train = sorted_df3.iloc[:1200]
test = sorted_df3.iloc[1200:]
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(train)
scaled_train = scaler.transform(train)
scaled_test = scaler.transform(test)
scaled_train[:10]
from keras.preprocessing.sequence import TimeseriesGenerator
# define generator
n_input = 3
n_features = 1
generator = TimeseriesGenerator(scaled_train, scaled_train, length=n_input,
batch_size=1)

X,y = generator[0]
print(f'Given the Array: \n{X.flatten()}')
print(f'Predict this y: \n {y}')

# We do the same thing, but now instead for 1 months
n_input = 30
generator = TimeseriesGenerator(scaled_train, scaled_train, length=n_input,
batch_size=1)

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM

# define model
model = Sequential()
model.add(LSTM(100, activation='relu', input_shape=(n_input, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
model.summary()
# fit model
model.fit(generator,epochs=50)
loss_per_epoch = model.history.history['loss']
plt.plot(range(len(loss_per_epoch)),loss_per_epoch)
last_train_batch = scaled_train[-30:]
last_train_batch = last_train_batch.reshape((1, n_input, n_features))
```

```
model.predict(last_train_batch)
scaled_test[0]
test_predictions = []

first_eval_batch = scaled_train[-n_input:]
current_batch = first_eval_batch.reshape((1, n_input, n_features))

for i in range(len(test)):

    # get the prediction value for the first batch
    current_pred = model.predict(current_batch)[0]

    # append the prediction into the array
    test_predictions.append(current_pred)

    # use the prediction to update the batch and remove the first value
    current_batch = np.append(current_batch[:,1:,:],[[current_pred]],axis=1)
test_predictions
true_predictions = scaler.inverse_transform(test_predictions)
test['Predictions'] = true_predictions
test
test.plot(figsize=(14,5))
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse=sqrt(mean_squared_error(test['TOTAL_BILL_VALUE'],test['Predictions']))
print('Root Mean Square Error for LSTM model:',rmse)
mae = mae(test['TOTAL_BILL_VALUE'],test['Predictions'])
print('Mean Absolute Error for LSTM Model is:',mae)
#Implementing LSTM,BILSTM and GRU in one function
sorted_df4 =
Billwise_sales.groupby(['DATE'])['TOTAL_BILL_VALUE','TOTAL_NOOF_ITEMS_PURCHAS
ED'].sum().reset_index()
sorted_df4=sorted_df4.set_index('DATE')
sorted_df4
# Split train data and test data
train_size = int(len(sorted_df4)*0.8)
train_dataset, test_dataset =
sorted_df4.iloc[:train_size],sorted_df4.iloc[train_size:]
# Plot train and test data
plt.figure(figsize = (10, 6))
plt.plot(train_dataset.TOTAL_BILL_VALUE)
plt.plot(test_dataset.TOTAL_BILL_VALUE)
plt.xlabel('Date (day)')
plt.ylabel('Daily Sales')
plt.legend(['Train set', 'Test set'], loc='upper right')
print('Dimension of train data: ',train_dataset.shape)
print('Dimension of test data: ', test_dataset.shape)
# Split train data to X and y
X_train = train_dataset.drop('TOTAL_BILL_VALUE', axis = 1)
y_train = train_dataset.loc[:,['TOTAL_BILL_VALUE']]
# Split test data to X and y
X_test = test_dataset.drop('TOTAL_BILL_VALUE', axis = 1)
y_test = test_dataset.loc[:,['TOTAL_BILL_VALUE']]
from sklearn.preprocessing import MinMaxScaler

# Different scaler for input and output
scaler_x = MinMaxScaler(feature_range = (0,1))
scaler_y = MinMaxScaler(feature_range = (0,1))
# Fit the scaler using available training data
```

```
input_scaler = scaler_x.fit(X_train)
output_scaler = scaler_y.fit(y_train)
# Apply the scaler to training data
train_y_norm = output_scaler.transform(y_train)
train_x_norm = input_scaler.transform(X_train)
# Apply the scaler to test data
test_y_norm = output_scaler.transform(y_test)
test_x_norm = input_scaler.transform(X_test)
# Create a 3D input
import numpy as np
def create_dataset (X, y, time_steps = 1):
    Xs, ys = [], []
    for i in range(len(X)-time_steps):
        v = X[i:i+time_steps, :]
        Xs.append(v)
        ys.append(y[i+time_steps])
    return np.array(Xs), np.array(ys)
TIME_STEPS = 30
X_test, y_test = create_dataset(test_x_norm, test_y_norm,
                                TIME_STEPS)
X_train, y_train = create_dataset(train_x_norm, train_y_norm,
                                  TIME_STEPS)
print('X_train.shape: ', X_test.shape)
print('y_train.shape: ', y_train.shape)
print('X_test.shape: ', X_test.shape)
print('y_test.shape: ', y_train.shape)
import scipy
import numpy
import tensorflow as tf
import tensorflow as keras
from tensorflow.keras import Sequential, layers, callbacks
from tensorflow.keras.layers import Dense, LSTM, Dropout, GRU, Bidirectional


# Create BiLSTM model
def create_model_bilstm(units):
    model = Sequential()
    model.add(Bidirectional(LSTM(units = units,
              return_sequences=True),
              input_shape=(X_train.shape[1], X_train.shape[2])))
    model.add(Bidirectional(LSTM(units = units)))
    model.add(Dense(1))
    #Compile model
    model.compile(loss='mse', optimizer='adam')
    return model
# Create LSTM or GRU model
def create_model(units, m):
    model = Sequential()
    model.add(m (units = units, return_sequences = True,
                 input_shape = [X_train.shape[1], X_train.shape[2]]))
    model.add(Dropout(0.2))
    model.add(m (units = units))
    model.add(Dropout(0.2))
    model.add(Dense(units = 1))
    #Compile model
    model.compile(loss='mse', optimizer='adam')
    return model
# BiLSTM
model_bilstm = create_model_bilstm(64)
```

```
# GRU and LSTM
model_gru = create_model(64, GRU)
model_lstm = create_model(64, LSTM)
# Fit BiLSTM, LSTM and GRU
from keras.callbacks import EarlyStopping, ModelCheckpoint

def fit_model(model):
    early_stop = tf.keras.callbacks.EarlyStopping(monitor =
'val_loss',patience = 10)
    history = model.fit(X_train, y_train, epochs = 100,
                        validation_split = 0.2, batch_size = 32,
                        shuffle = False, callbacks = [early_stop])
    return history
history_bilstm = fit_model(model_bilstm)
history_lstm = fit_model(model_lstm)
history_gru = fit_model(model_gru)
# Plot train loss and validation loss
def plot_loss (history):
    plt.figure(figsize = (10, 6))
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.ylabel('Loss')
    plt.xlabel('epoch')
    plt.legend(['Train loss', 'Validation loss'], loc='upper right')
plot_loss (history_bilstm)
plot_loss (history_lstm)
plot_loss (history_gru)
y_test = scaler_y.inverse_transform(y_test)
y_train = scaler_y.inverse_transform(y_train)
# Make prediction
def prediction(model):
    prediction = model.predict(X_test)
    prediction = scaler_y.inverse_transform(prediction)
    return prediction
prediction_bilstm = prediction(model_bilstm)
prediction_lstm = prediction(model_lstm)
prediction_gru = prediction(model_gru)
# Plot true future vs prediction
def plot_future(prediction, y_test):
    plt.figure(figsize=(10, 6))
    range_future = len(prediction)
    plt.plot(np.arange(range_future), np.array(y_test),
             label='Actuals')
    plt.plot(np.arange(range_future),np.array(prediction),
            label='Prediction')
    plt.legend(loc='upper left')
    plt.xlabel('Date (day)')
    plt.ylabel('Total Sales')
plot_future(prediction_bilstm, y_test)
plot_future(prediction_lstm, y_test)
plot_future(prediction_gru, y_test)
# Define a function to calculate MAE and RMSE
def evaluate_prediction(predictions, actual, model_name):
    errors = predictions - actual
    mse = np.square(errors).mean()
    rmse = np.sqrt(mse)
    mae = np.abs(errors).mean()
    print(model_name + ':')
    print('Mean Absolute Error: {:.4f}'.format(mae))
```

```
    print('Root Mean Square Error: {:.4f}'.format(rmse))
    print('')
evaluate_prediction(prediction_bilstm, y_test, 'Bidirectional LSTM')
evaluate_prediction(prediction_lstm, y_test, 'LSTM')
evaluate_prediction(prediction_gru, y_test, 'GRU')
```

Product wise sales EDA:

```
#Exploratory Data Analysis
#importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#read the dataset
product_sales = pd.read_csv("product wise sales report 2019-2022.csv")
product_sales.head(10)
product_sales['BRAND'].value_counts()
product_sales['BRAND'].unique()
product_sales.BRAND.replace({'Sweetee Palmyrah Palm
Jaggery':"Prakruthivanam",'Prakruthivanamm':"Prakruthivanam",'Sweetee
Palmyrah Palm Jaggery
':"Prakruthivanam",'Prakruthiv':"Prakruthivanam",'Prakruthiivannam':"Prakruth
ivanam",'DHAPUUR GREEN' : "Prakruthivanam"
,'prakruthivanam':'Prakruthivanam','PRAKRUTHIVANAM':'Prakruthivanam','PRAKRUT
HIV':'Prakruthivanam','Tamarind-250 Gms-pvs':'Prakruthivanam','Andu Korralu
500gms':'Prakruthivanam','Andu Korra Flour
500g':'Prakruthivanam','Gcc':'Prakruthivanam','GCC':'Prakruthivanam','Akshaya
':'Prakruthivanam'}, inplace = True)
product_sales.BRAND.replace({'Surabhi':"Surabhi Healthy Life",'Healthy
Life':"Surabhi Healthy Life",'Sri venkateswara
Enterprises':"Puro",'Pure&Sure':"Pure & Sure",'Pure & Sue':"Pure & Sure"},
inplace = True)
product_sales.BRAND.replace({'Timbaktu ' : "Timbaktu" }, inplace = True)
product_sales.BRAND.replace({'Sree Chandan Corporation ':"Dry Fruits",'Shree
Chandan Corporation':"Dry Fruits",'Sree Chandan Corporation':"Dry
Fruits",'Kimia':"Dry Fruits",'Manam Foods':"Dry Fruits",'Sree Chandan
Corporotion' : "Dry Fruits" ,'Dry Fruits ':'Dry Fruits'}, inplace = True)
product_sales.BRAND.replace({'Herbal Strategi':"Herbal Strategi",'Herbal
Strategi ':"Herbal Strategi",'Herbal Strategic':"Herbal Strategi",'Go
Desi':"Go Desi",'GO DESI':"Go Desi",'GO DESi' : "Go Desi" }, inplace = True)
product_sales.BRAND.replace({'Rathnam':"Miscellaneous",'Travels/courier':"Mis
cellaneous",'Cloth Bag':"Miscellaneous",'Vaarahi':"Miscellaneous",'Gouthami
Agro':"Miscellaneous",'Gouthami Agro Industries' : "Miscellaneous"
,'Miscellaneous':'Miscellaneous'}, inplace = True)
product_sales.BRAND.replace({'Clay Pots' : "Cookware" }, inplace = True)
product_sales.BRAND.replace({'Native Circle':"Native Circle",'NATIVE CIRCLE
':"Native Circle",'NATIVE CIRCLE':"Native Circle",'Native Circle ':"Native
Circle",'Native Cvircle':"Native Circle",'Native Cicrle' : "Native Circle" },
inplace = True)
product_sales.BRAND.replace({'Water Filter ' : "Water Filter", 'Water Filt' :
"Water Filter"}, inplace = True)
product_sales.BRAND.replace({'Danthu Naturals' : "Dathu Ayurvedam", 'Dathu
Naturals' : "Dathu Ayurvedam"}, inplace = True)
product_sales.BRAND.replace({'Absolute Ghee' : "Absolute Milk" }, inplace =
True)
product_sales.BRAND.replace({'Arogya Rahasya ':"Arogya Rahasya",'Gou
Ganga':"Arogya Rahasya",'ArogyaRaha':"Arogya Rahasya",'Arogya':"Arogya
```

```
Rahasya",'Arogya Rah':"Arogya Rahasya",'Sadguru Sri Sri' : "Arogya Rahasya"
}, inplace = True)
product_sales.BRAND.replace({'Cast Iron':"Cookware",'Soap
Stone':"Cookware",'Woodenware':"Cookware",'Noor Pans':"Cookware",'Noor
Pan':"Cookware",'Prithvimaya' : "Cookware" ,'Iron':'Cookware'}, inplace =
True)
product_sales.BRAND.replace({'Clay pot' : "Cookware" }, inplace = True)
product_sales['BRAND'].unique()
product_sales.nlargest(n=10, columns='QTY SOLD')
product_sales['BRAND'].unique()
product_sales['BRAND'].value_counts()
product_sales.nsmallest(n=50, columns='QTY SOLD')
product_sales.nlargest(n=50, columns='QTY SOLD')
product_sales.groupby('BRAND')['MRP'].nlargest(1)

df = product_sales.groupby('BRAND')['QTY SOLD','NO. OF
BILLS','REVENUE'].sum()
df
plt.figure(figsize=(15,10))

plt.ylabel('REVENUE')
plt.xlabel('BRAND')
plt.xticks(rotation=45)
plt.plot(df.index, df['REVENUE'],)

plt.figure(figsize=(15,10))
plt.ylabel('QTY SOLD')
plt.xlabel('BRAND')
plt.xticks(rotation=45)
plt.plot(df.index, df['QTY SOLD'],)
```

GIT hub Code and Dataset Links:

https://github.com/cherukuribh/Prakruthivanam-Project.git

OneDrive Datasets Link:

Bill wise report:

https://1drv.ms/u/s!AmAHyMNwrXGvbSy1QVE5xt2nVm4?e=dkLl7d

Daily sales report:

https://1drv.ms/u/s!AmAHyMNwrXGvalEwxpuOvChxfHU?e=Jn6KnN

Product wise sales Report:

https://1drv.ms/u/s!AmAHyMNwrXGva0k0rRceWULYpkY?e=ug6o62

OneDrive Code links:

Product wise sales EDA:

https://1drv.ms/u/s!AmAHyMNwrXGvbON38byh__NVb5A?e=WdhMHX

PVS Time series forecasting Implementation:

https://1drv.ms/u/s!AmAHyMNwrXGvbgKCVEVBkw5kvOw?e=VTVCAa

Appendix B – Meeting Records

The below table shows the details of the meetings with my first supervisor Dr. Marwan Fuad.

Meeting	Date	Time
Online Teams Meeting 1	30 September 2022	4.00 PM – 5.00 PM
Online Teams Meeting 2	14 October 2022	2.00 PM – 4.00 PM
Online Teams Meeting 3	28 October 2022	3.00 PM – 4.30 PM
Online Teams Meeting 4	11 November 2022	2.00 PM – 3.00 PM
Online Teams Meeting 5	25 November 2022	2.30 PM – 3.30 PM

Appendix C – Certificate of Ethics Approval

Certificate of Ethical Approval

Applicant:	Bharathi Cherukuri
Project Title:	Analysing customer behaviour and sales trend based on the time stamp

This is to certify that the above named applicant has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Medium Risk

Date of approval:	07 Oct 2022
Project Reference Number:	P141766