Dear Intern

Project report is an inherent component of your internship. We are enclosing a reference table of content for the project report. Depending on the internship project (IT/Non-IT, Technical/Business Domain), you may choose to include or exclude or rename sections from the table of content mentioned below. You can also add additional sections. The key objective of this report is for you to systemically document the project work done.

|  |  |
| --- | --- |
| Internship Project Title | RIO-125: Forecasting System - Project Demand of Products at a Retail Outlet Based on Historical Data |
| Name of the Company | TCS iON |
| Name of the Industry Mentor | Debashis Roy |
| Name of the Institute | ICT Academy of Kerala |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Start Date | End Date | Total Effort (hrs.) | Project Environment | Tools used |
| 05/02/2023 | 06/03/2023 | 125 | Jupyter Notebook | Python |

**TABLE OF CONTENT**

* Acknowledgements
* Objective
* Introduction /
* Methodology
* Process Flow Diagram
* Exploratory Data Analysis
* Analysis of Monthly Trend and Seasonality
* ETS Decomposition
* Trend and Seasonality Analysis of all Stores
* Checking Data Stationarity
* ARIMA Model Building
* Fbprophet Model Building
* Comparison of RMSE of SARIMAX and Fbprophet
* Result
* Conclusion
* References

**ACKNOWLEDGEMENT**

“I would like to express our deep appreciation to all those who have contributed to the successful completion of this project.

Special thanks to our project guide from TCS iON for their invaluable guidance and support. Their expertise and assistance was crucial in making this project a reality.

I also extend our appreciation to ICT Academy of Kerala for offering us the opportunity to work with TCS iON as part of a one-month internship program, which allowed me to gain a deeper understanding of the IT industry's work culture.

I would like to acknowledge the support of my colleagues and friends who have provided me with their suggestions and insights, which have been of immense help in completing this project.

Finally, we would like to thank our families for their love, support and encouragement throughout the project.

Thank you, everyone, for your support and cooperation."

Bottom of Form

**OBJECTIVE**

The objective of the project Forecasting System - Project Demand of Products at a Retail Outlet Based on Historical Data is to predict the demand for products at a retail outlet based on historical sales data.

The goal is to create a forecasting model that can accurately predict future product demand, allowing the retail outlet to make informed decisions regarding inventory management, product sourcing, and sales planning.

The project aims to improve the efficiency and profitability of the retail outlet by reducing the instances of stock shortages or overstocking. The project also provides a platform to analyze sales trends and make data-driven decisions to optimize the retail outlet's operations.

**INTRODUCTION**

"The demand forecasting system has become a critical component in managing the supply chain of modern retail businesses. Accurate forecasting of product demand allows retailers to make informed decisions on inventory management, product ordering, and pricing strategies.

The objective of this project is to develop a forecasting system that will predict the demand for products at a retail outlet based on historical data. This system will provide the retail outlet with valuable insights on the future demand for their products, enabling them to make data-driven decisions to optimize their supply chain and increase profitability.

The project focuses on analyzing the historical sales data of the retail outlet to identify patterns and trends that can be used to make accurate demand forecasts. Advanced statistical techniques and machine learning algorithms were employed to build the forecasting model.

This report details the methodology and results of the project, and highlights the potential impact of the forecasting system on the retail outlet's operations and profitability. The report also includes a discussion of the limitations and future directions for improvement of the forecasting system."

**METHODOLOGY**

Time series forecasting is the process of predicting future values of a time series based on its past behavior. A time series is a sequence of data points that are measured at regular intervals over time. Examples of time series data include stock prices, temperature readings, and sales data.

Time series forecasting techniques can be used to make predictions about future trends and patterns in the data, which can be useful for a wide range of applications, such as demand forecasting, inventory management, and financial forecasting.

Some common techniques for time series forecasting include:

* Autoregressive Integrated Moving Average (ARIMA) models
* Seasonal Autoregressive Integrated Moving Average (SARIMA) models
* Exponential Smoothing (ES) models
* Prophet models (developed by Facebook)

These models take into account the patterns and trends in the historical data to make predictions about future values of the time series. They can also be used to estimate the uncertainty in the predictions and to identify any potential sources of error or bias in the forecasting process. This project is implemented using:

* **ARIMA (AutoRegressive Integrated Moving Average**): This method models the relationship between an observation and a number of lagged observations.
* **SARIMA (Seasonal ARIMA)**: This method is similar to ARIMA, but takes into account seasonality in the data.
* **Prophet:** This is a forecasting method developed by Facebook specifically for time series data. It is based on decomposing the time series into trend, seasonality and holiday components.

**PROCESS FLOW DIAGRAM**

**Model Estimation**

**Start**

Data Preprocessing

**2018**

**Forecasting**

Not Accepted

Accepted

No

**Time Series**

**Transformation**

Yes

**End**

**Load Dataset**

**Sales**

**Dataset**

**Stationarity Check**

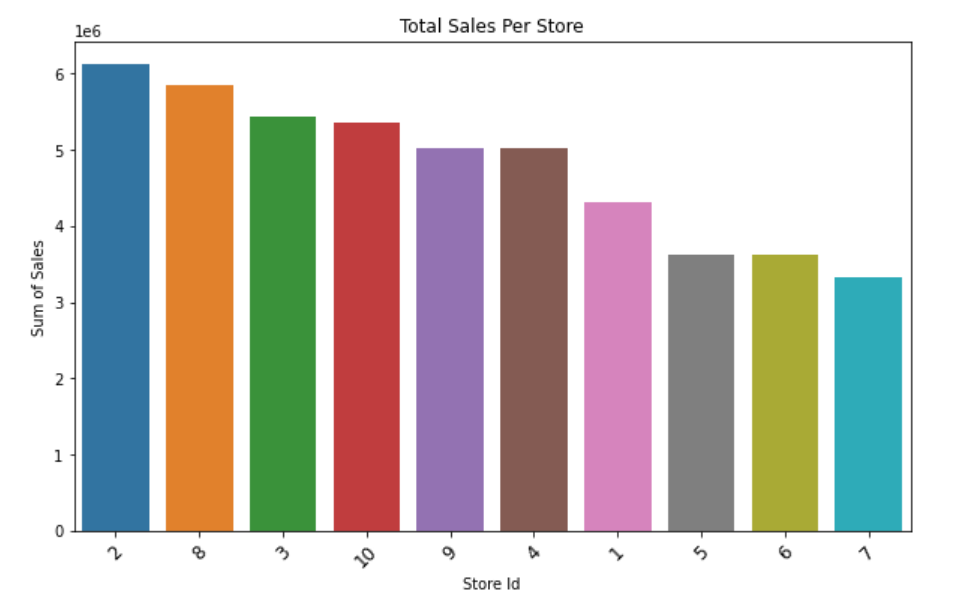
**Model Evaluation**

**Model Identification**

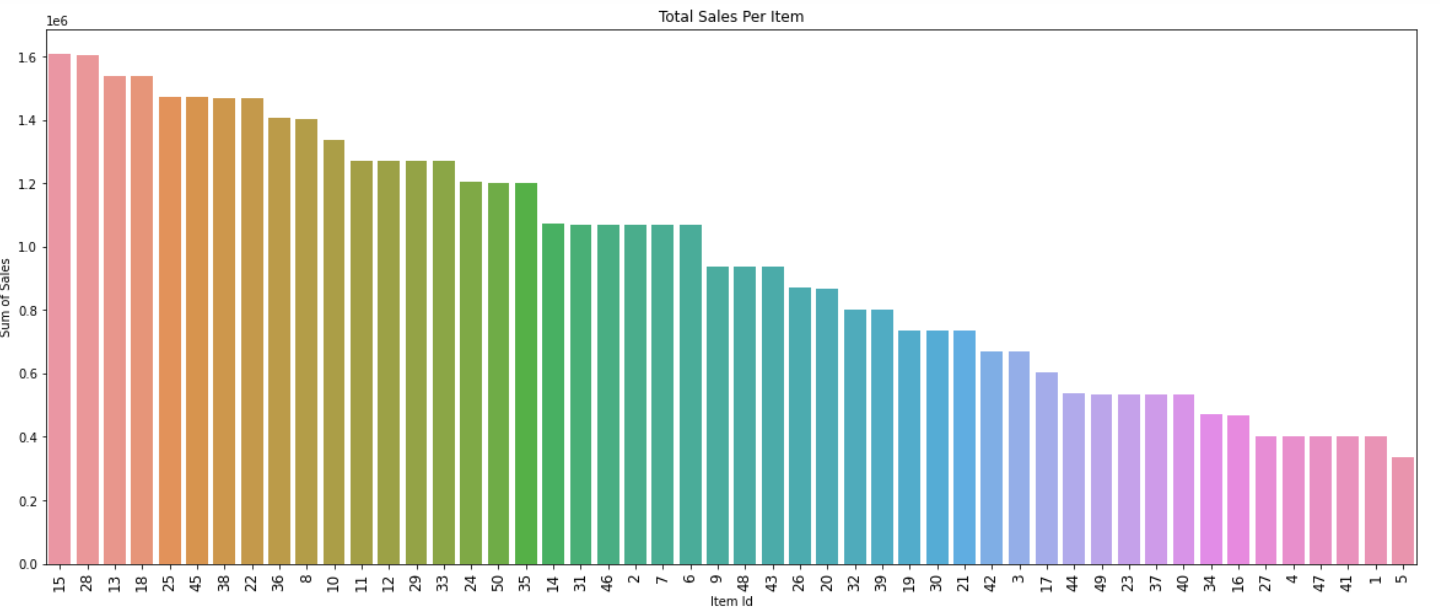
**EXPLORATORY DATA ANALYSIS**

Inferences obtained from dataset are:

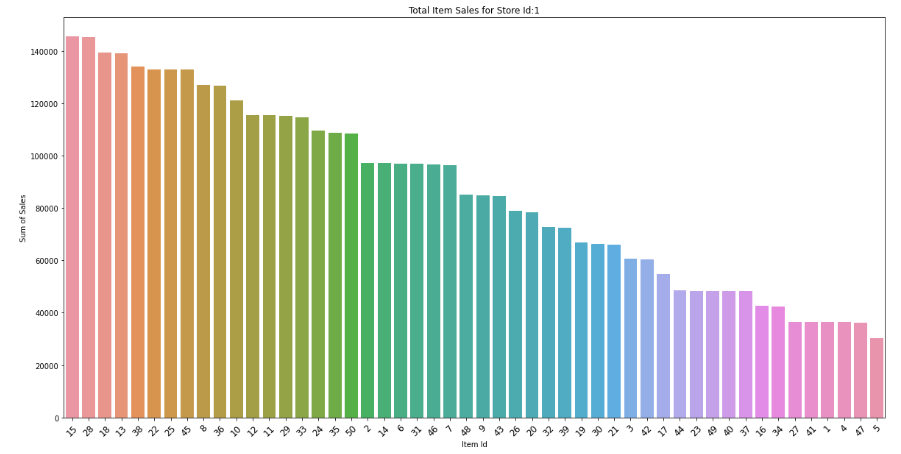
* The training dataset consists of two files, with one containing the target sales column for training and the other used for testing the model.
* The test data includes information from January 1st, 2018 to March 31st, 2018.
* The maximum date in the training set is December 31st, 2017, while the maximum date in the test set is March 31st, 2018.
* The forecast lag size is 90.
* It was determined that during the time period from January 1st, 2013 to December 31st, 2017, the train dataset had 10 stores selling 50 items.
* Information was also gathered on the total sales in each of the 10 stores and the number of the 50 items sold in each store.
* All stores have the same number of unique items.

****

**Figure: Total sales per store**

****

**Figure: Total sales per item**

****

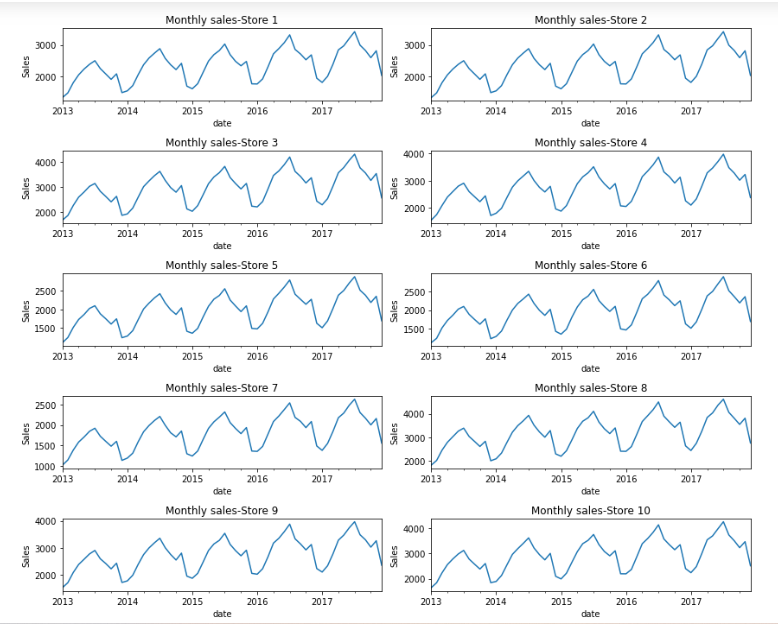
**Figure: Total item sales for store 1**

**DATA PREPROCESSING**

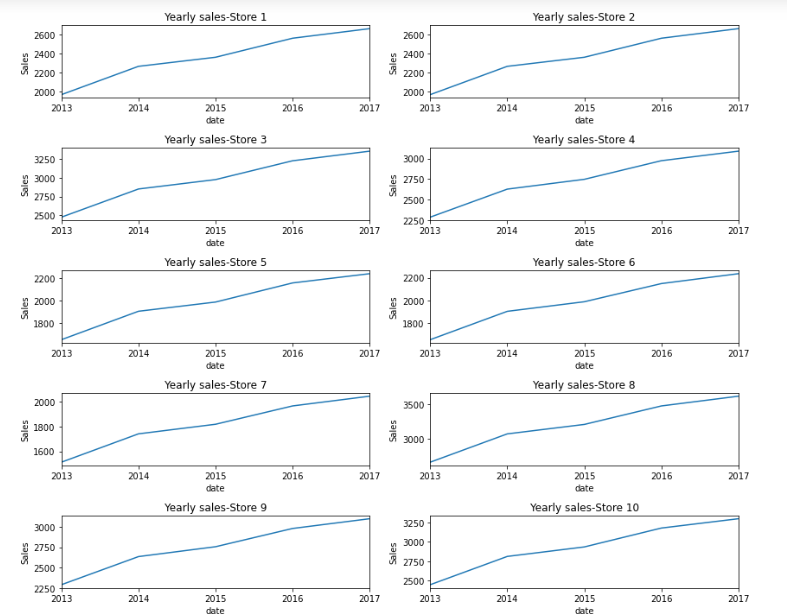
Resampling is a process of changing the time frequency of a time series data. It involves aggregating the data over a different time period, either by increasing or decreasing the frequency of the data.

In the context of sales data, resampling can be done to convert the data from its original frequency (e.g., daily) to a lower frequency such as monthly, quarterly or yearly. This can be useful for time series forecasting because it can help to reduce the noise in the data, identify patterns and trends at a higher level of aggregation, and make it easier to detect seasonal variations.

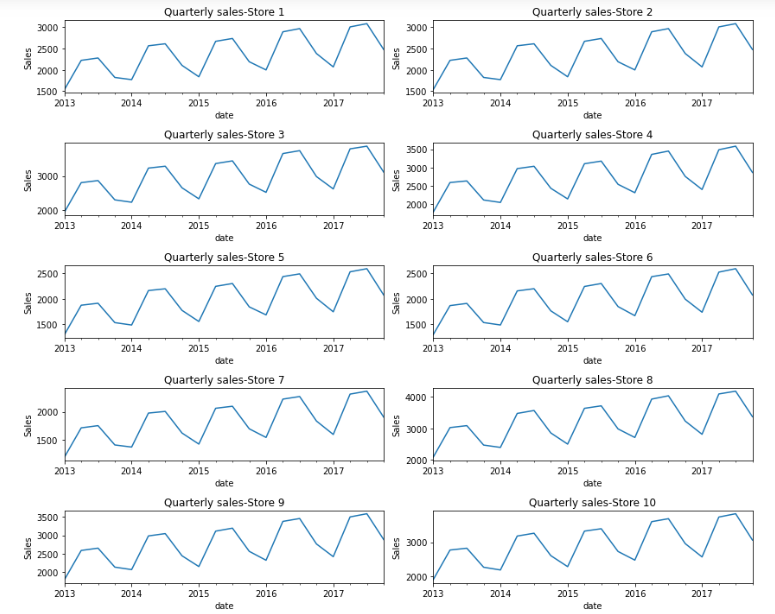
Furthermore, resampling can be used to aggregate data for specific seasons or quarters, allowing you to identify seasonality and trends that may not be apparent in the original data. For example, you can resample the data to a quarterly level to identify any trends that occur during a specific quarter, such as the holiday season.

****

**Figure: Monthly sales per store**

****

**Figure: Yearly sales per store**

****

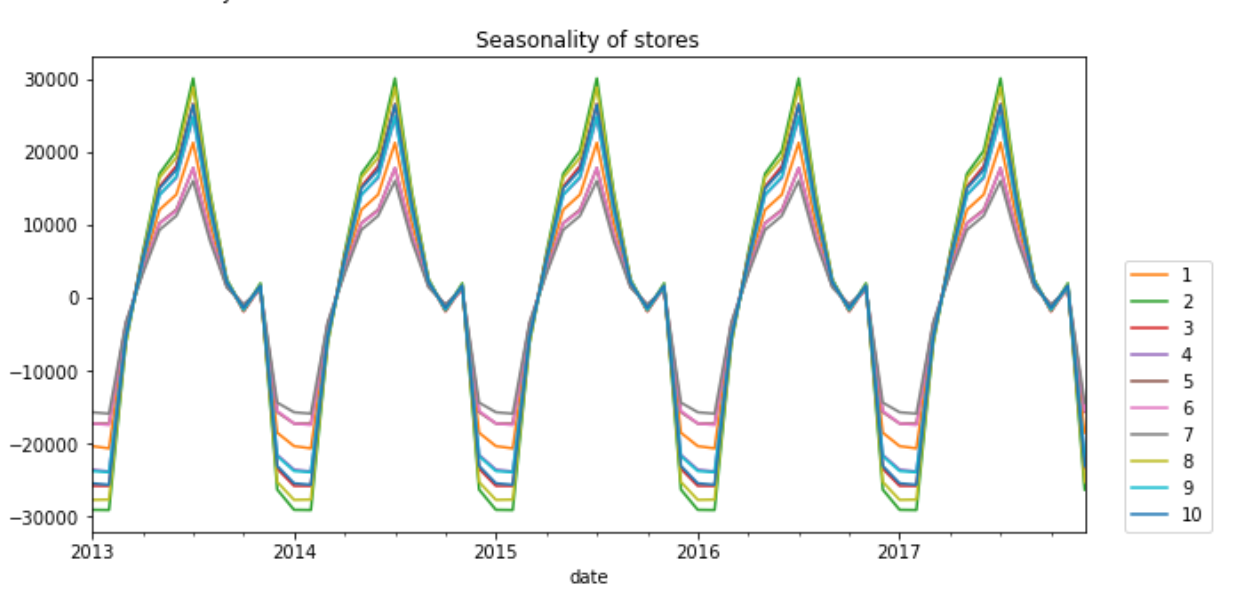
**Figure: Quarterly sales per store**

**ANALYSIS OF MONTHLY TREND AND SEASONALITY**

Analyzing monthly trend and seasonality is an important step in time series analysis, as it helps to identify any recurring patterns or cycles in the data. Here are some steps to perform this analysis:

* Visualize the data: Plot the monthly sales data over time to get a visual representation of the trend and seasonality. This can be done using a line chart or a scatter plot.
* Decompose the time series: Decompose the time series into its trend, seasonal, and residual components using a decomposition method such as additive or multiplicative decomposition. This will help to identify the seasonality and trend components separately.
* Examine the trend: Examine the trend component of the time series to identify any long-term patterns or trends in the data. This can be done using a simple linear regression model or a more complex time series model such as an ARIMA model.
* Examine the seasonality: Examine the seasonal component of the time series to identify anyrecurring patterns or cycles in the data. This can be done using a seasonal plot, which plots the average sales value for each month over multiple years. If the seasonal component is strong, it can be modeled using a seasonal ARIMA model or a seasonal regression model.
* Identify any outliers: Check for any outliers or extreme values in the data, which can distort the trend and seasonality. These outliers can be removed or adjusted to improve the accuracy of the forecasting models.

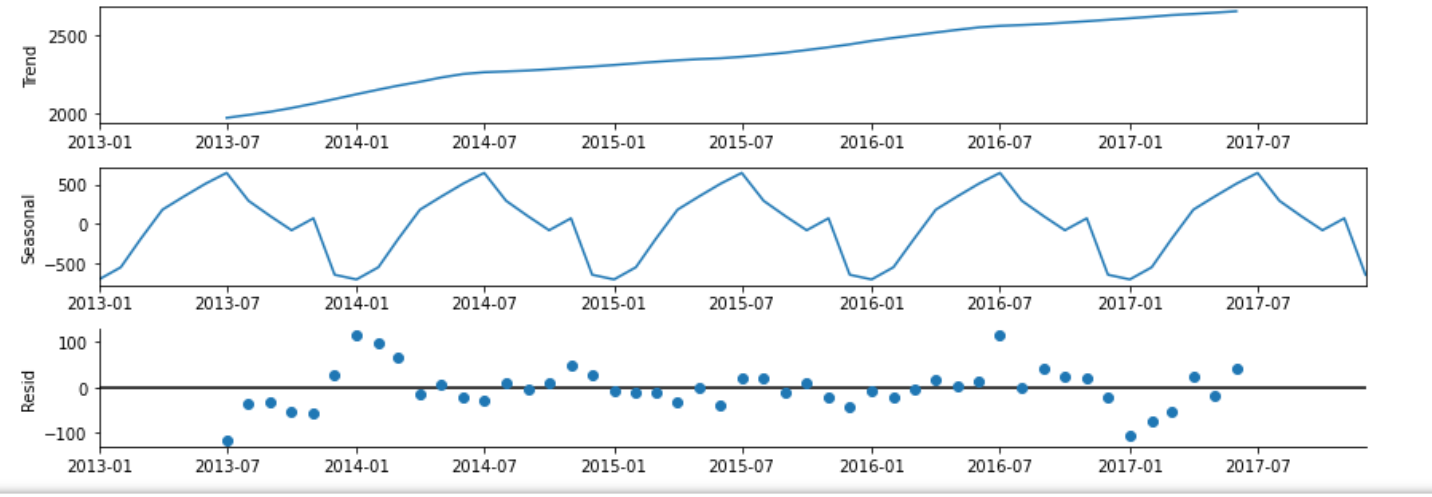
****

****

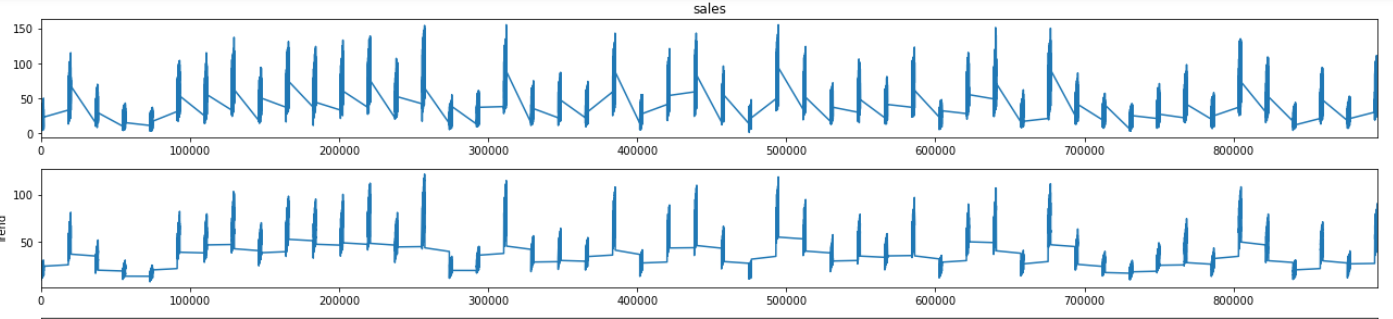
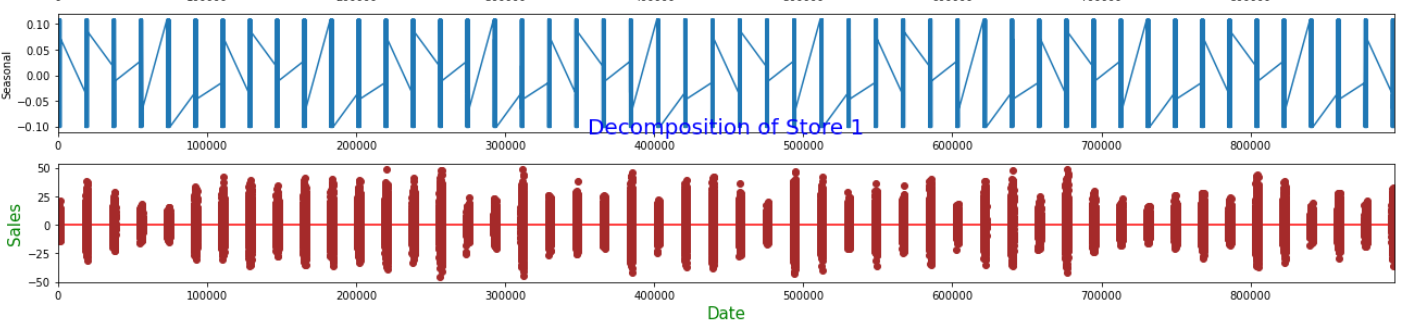
**ETS (ERROR, TREND, SEASONALITY) DECOMPOSITION**

ETS (Error, Trend, Seasonality) decomposition is a method used in time series analysis to separate a time series into its three components: error, trend, and seasonality. This method is often used as a preliminary step in time series analysis to identify the underlying patterns and trends in the data, which can then be used to develop more accurate forecasting models.

The ETS decomposition method assumes that the time series can be modeled using an additive or multiplicative model, where the observed values can be represented as the sum or product of the three components: error, trend, and seasonality. The error component represents the random fluctuations or noise in the data that cannot be explained by the trend or seasonality, while the trend component represents the long-term behavior or direction of the data. The seasonality component represents the recurring patterns or cycles in the data that occur at fixed intervals, such as daily, weekly, or monthly.

****

**TREND AND SEASONALITY ANALYSIS OF ALL STORES**

The resulting plots will show the original sales data, the estimated trend component, the estimated seasonality component, and the residual component for each store. These plots can be used to visually inspect the trend and seasonality patterns in the sales data for each store. For example, if the trend component is increasing over time, it may indicate that the store is experiencing overall growth in sales. If the seasonality component shows a recurring pattern over time, it may indicate that the store experiences fluctuations in sales at fixed intervals, such as holidays or seasonal events.

**CHECKING DATA STATIONARITY**

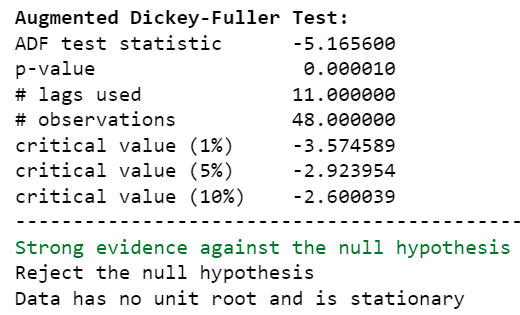
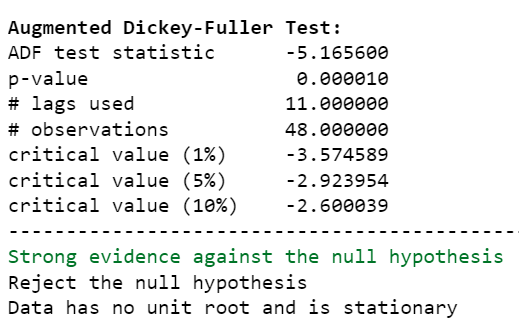
# Augmented Dickey-Fuller Test for testing Seasonality

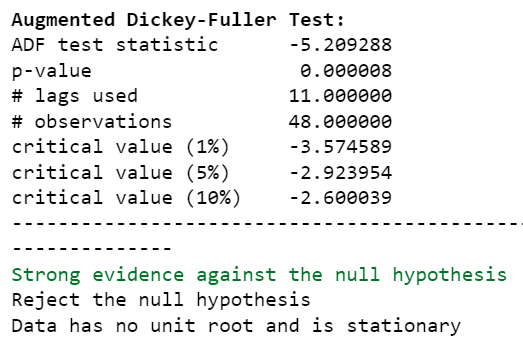
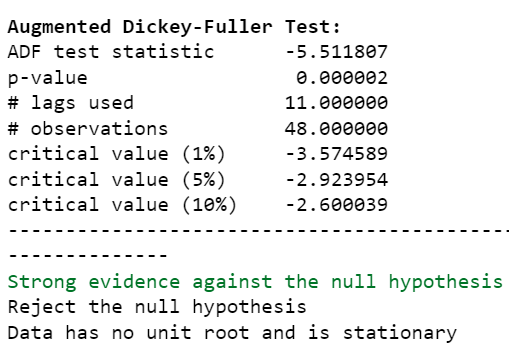
The Augmented Dickey-Fuller (ADF) test is a statistical test used to determine whether a time series has a unit root, which indicates the presence of a trend or seasonality in the data. The ADF test is commonly used to test the null hypothesis that a time series is non-stationary, meaning that it exhibits a trend or seasonality, against the alternative hypothesis that the time series is stationary, meaning that it does not exhibit a trend or seasonality.

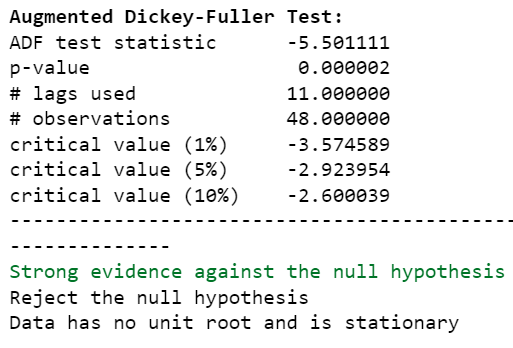
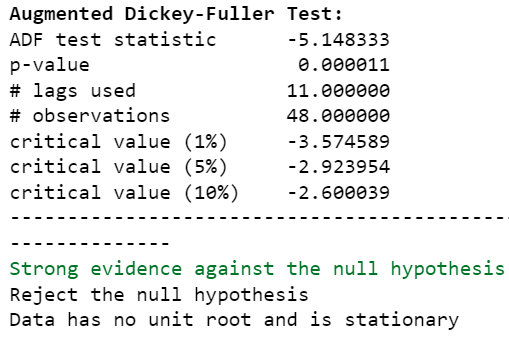
In the context of seasonal time series forecasting, the ADF test can be used to test whether a time series exhibits seasonality. If the ADF test statistic is significantly less than the critical value, then the null hypothesis of non-stationarity can be rejected, indicating that the time series is stationary and does not exhibit seasonality. If the ADF test statistic is not significantly less than the critical value, then the null hypothesis cannot be rejected, indicating that the time series may exhibit seasonality.

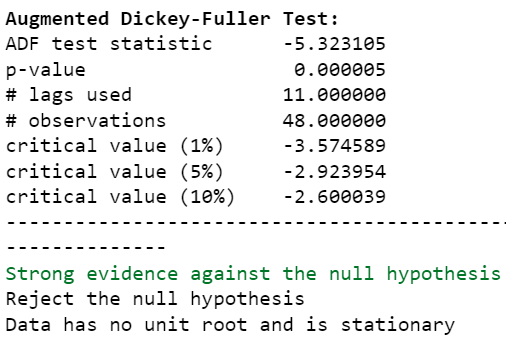
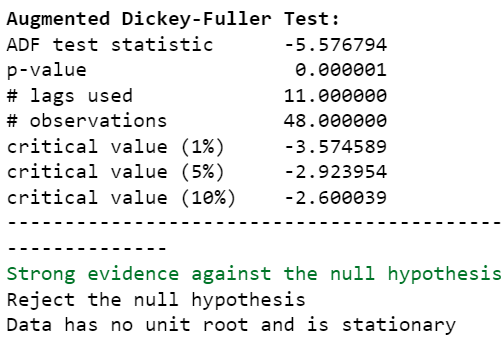
Overall, the ADF test can be a useful tool for identifying whether a time series exhibits seasonality and for determining the appropriate modeling approach for forecasting the time series.

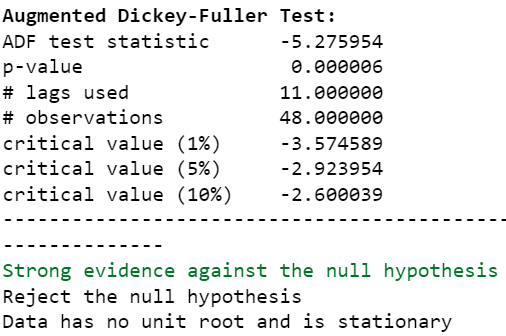
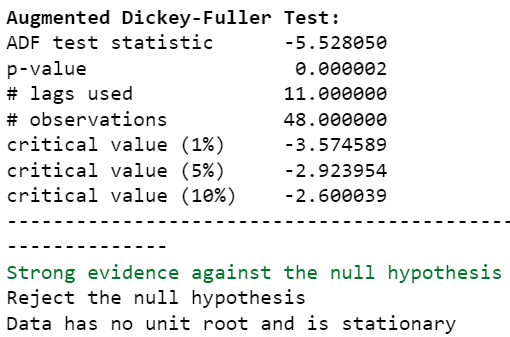
Top of Form

** **

** **

****

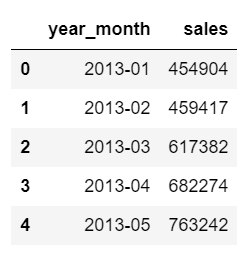
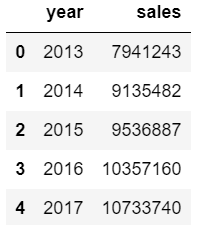
****

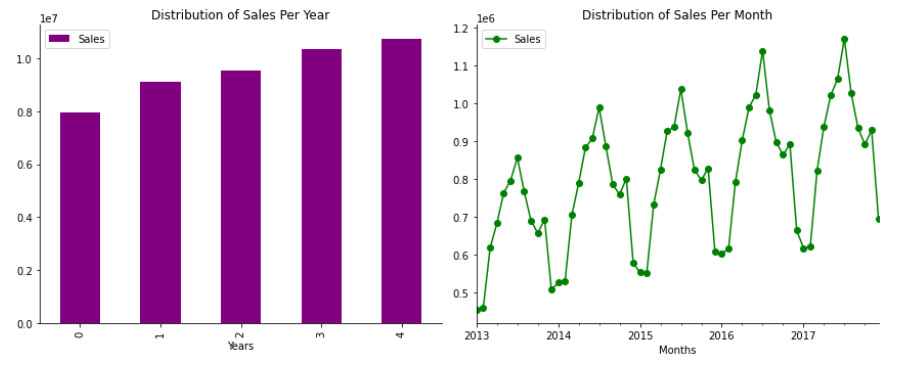
****

Stationarity is a key assumption in time series analysis, and it refers to the property of a time series whose statistical properties such as mean, variance, and autocorrelation remain constant over time. Stationary time series are easier to model and forecast because they exhibit predictable patterns over time. In this context, the finding that the monthly sales of all stores are stationary means that their statistical properties such as mean and variance do not change over time. This is an important finding because it allows us to use a wide range of time series models to capture the underlying patterns in the sales data and make accurate forecasts. It also suggests that there are no long-term trends or seasonal patterns in the sales data that might otherwise complicate our analysis.

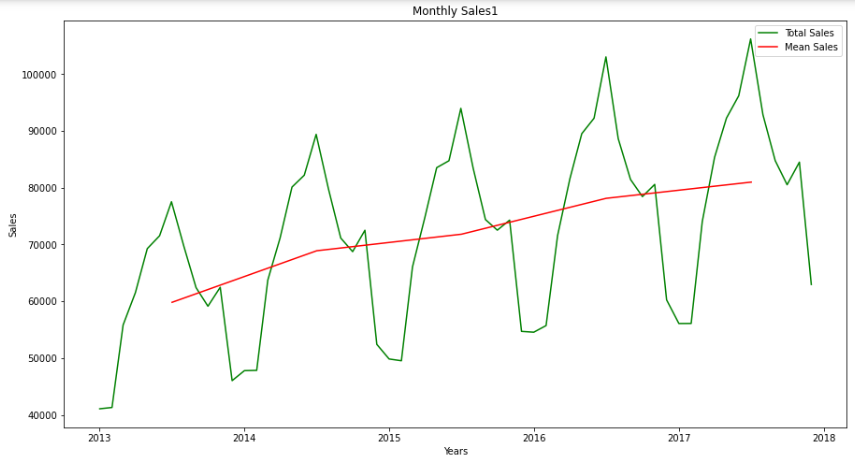
For further analysis, create the total daily, monthly and annual sales data frames:

* Daily Sales Data Frame: Aggregate the sales data by day to calculate the total sales for each day. This will provide a daily view of the sales trend.
* Monthly Sales Data Frame: Group the daily sales data by month to calculate the total sales for each month. This will provide a monthly view of the sales trend.
* Annual Sales Data Frame: Group the monthly sales data by year to calculate the total sales for each year. This will provide an annual view of the sales trend.

**  **

****

**Figure: Daliy sales from 2013 to 2017**

****

Based on the graphs, it is evident that sales for each store increase every month. However, upon conducting the Dickey Fuller test, it was found that the sales data for each store is stationary. As a result, stationary data can be utilized to forecast using ARIMA, SARIMAX, and Fbprophet models.

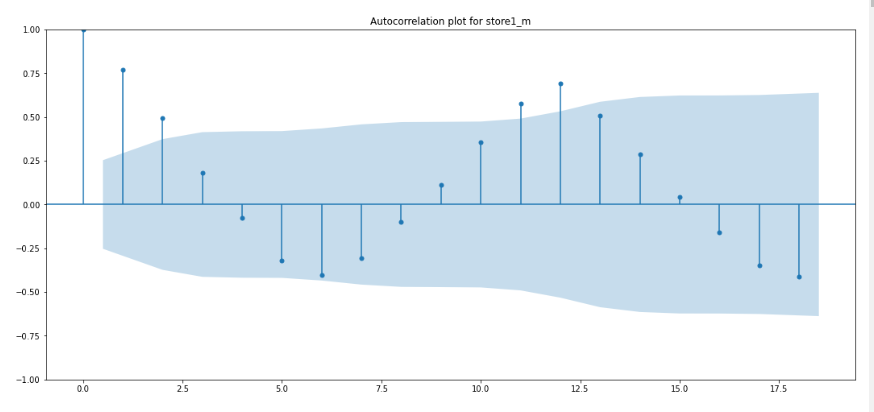
**ARIMA MODEL BUILDING**

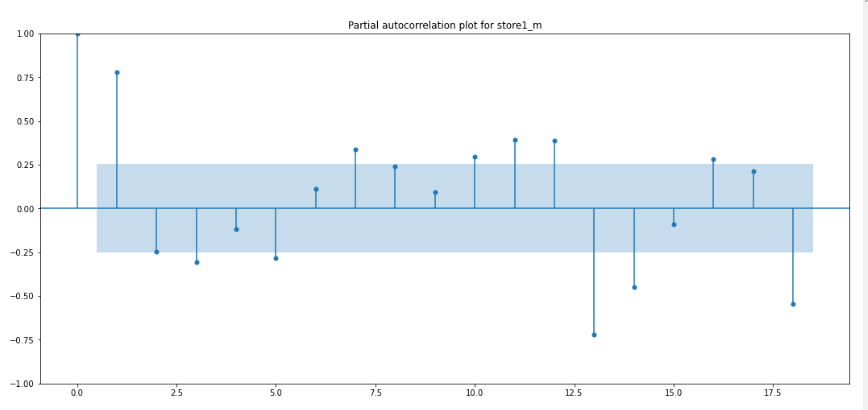
# Finding Autocorrelation and Partial autocorrelation plots for all stores

Autocorrelation (ACF) and partial autocorrelation (PACF) plots are tools used to identify the patterns of correlation between time series data and its lags. ACF measures the correlation between a time series and its lagged values, while PACF measures the correlation between a time series and its lags, controlling for the effect of intermediate lags.

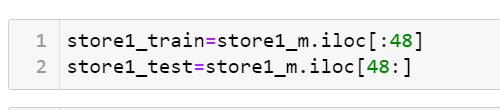
In the context of time series modeling, ACF and PACF plots are used to identify the appropriate orders for an ARIMA model. The ACF plot provides information about the MA(q) term, while the PACF plot provides information about the AR(p) term.

To create ACF and PACF plots for all stores, we can use the plot\_acf() and plot\_pacf() functions from the statsmodels.graphics.tsaplots module. These functions take a time series data as input and return a plot showing the correlation coefficient against the lag values. The significance level of the correlation coefficient is indicated by horizontal lines in the plot. If the correlation coefficient is outside the significance level, it indicates a statistically significant correlation.





# Splitting into Train and Test Data



store1\_m is the time series data for 'Store 1' that contains monthly sales values for a period of three years. The iloc function is used to index the first 48 rows of this data which corresponds to the first four years of monthly sales data for 'Store 1'. The resulting data is assigned to store1\_train.

The remaining data after the first 48 rows (i.e., data from the fifth year of sales) is selected using iloc and assigned to store1\_test. This testing data will be used to evaluate the performance of any models built using the training data.

# Finding ARIMA Order (p, d, q)

The ARIMA model is a time series forecasting model that stands for Autoregressive Integrated Moving Average. It uses three parameters to describe the time series data: p, d, and q.

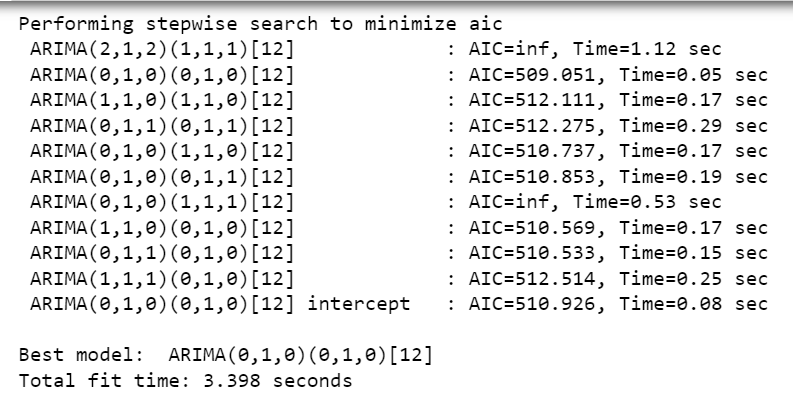
p: the number of autoregressive terms, which refers to the number of lagged observations that are included in the model.

d: the number of times the data needs to be differenced to make it stationary.

q: the number of moving average terms, which refers to the number of lagged forecast errors that are included in the model.

In the best ARIMA model, the values of p, d, and q are all 0, indicating that no autoregressive or moving average terms are needed and that the data only needs to be differenced once to make it stationary. The [12] in the model specification denotes a seasonal component with a period of 12 months.

In some cases, the ARIMA model may not adequately capture the complexity of the time series data. In such cases, a more sophisticated model such as the SARIMAX model may be needed. The SARIMAX model is an extension of the ARIMA model that incorporates additional parameters to capture the effects of seasonality, exogenous variables, and other factors that may impact the time series. By including these additional components, the SARIMAX model can provide more accurate forecasts than the ARIMA model alone. Therefore, ARIMA models may be transformed into SARIMAX models to improve their accuracy.



The best ARIMA model is specified as ARIMA(0,1,0)(0,1,0)[12]. This model has two components:

The non-seasonal component, specified as (0,1,0), which indicates that the data does not require any autoregressive (AR) or moving average (MA) terms to be included in the model, but needs to be differenced once (d=1) to make it stationary.

The seasonal component, specified as (0,1,0)[12], which indicates that the data has a seasonal pattern with a period of 12 months. This component also does not require any AR or MA terms, but needs to be seasonally differenced once (D=1) to make it stationary.

In summary, the best ARIMA(0,1,0)(0,1,0)[12] model suggests that the sales data has a stationary seasonal component with a period of 12 months, but no significant non-seasonal component that requires additional AR or MA terms. This model is chosen based on statistical metrics such as AIC, BIC, and MSE, which evaluate the goodness of fit and accuracy of the model.

# SARIMAX Summary and Diagnostics for Store 1 to Store 10

The store\_result.summary() method provides a summary table that shows various statistical metrics such as the coefficients, standard errors, t-values, p-values, and confidence intervals for each parameter in the SARIMAX model. These metrics can be used to evaluate the goodness of fit and accuracy of the model.

The diagnostic plots are a visual way to assess the fit of the model. They consist of four panels:

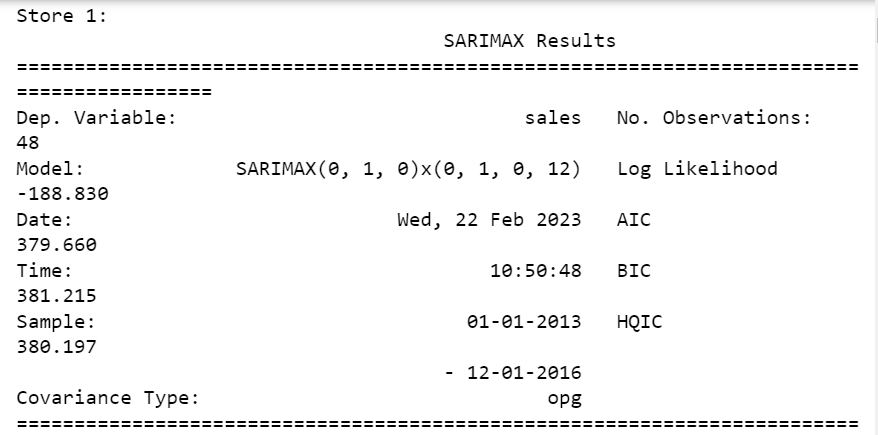
The histogram of the residuals, which should be approximately normally distributed if the model is a good fit.

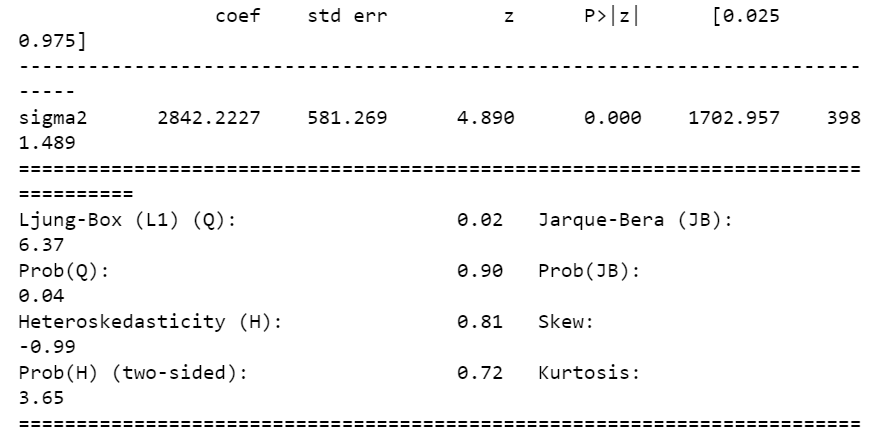
The kernel density estimate (KDE) of the residuals, which should be similar to a normal distribution.

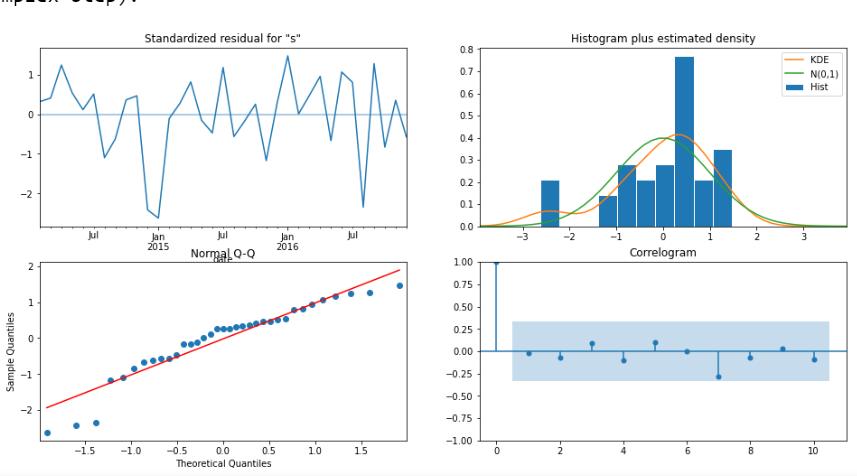
The Q-Q plot of the residuals, which compares the distribution of the residuals to a normal distribution. The points on the plot should fall approximately on a straight line.

The correlogram or autocorrelation function (ACF) of the residuals, which shows the correlation between the residuals at different lags. The ACF should not show any significant correlation at any lag, indicating that the residuals are not serially correlated.

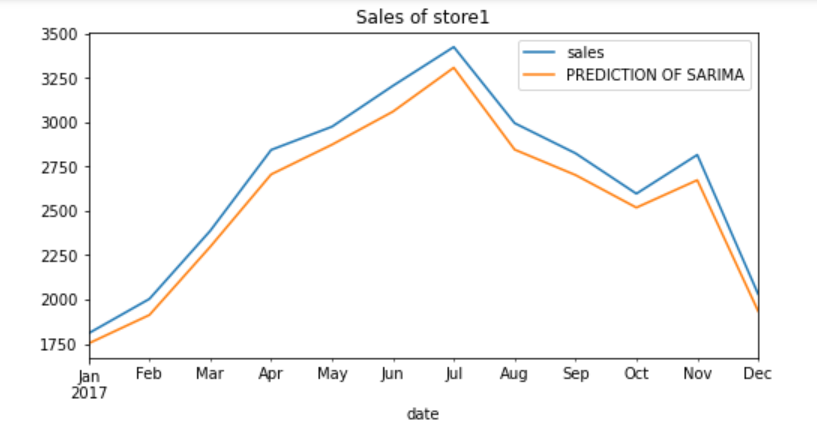
Overall, the summary and diagnostic plots provide useful information to assess the fit and accuracy of the SARIMAX model for each store's sales data.

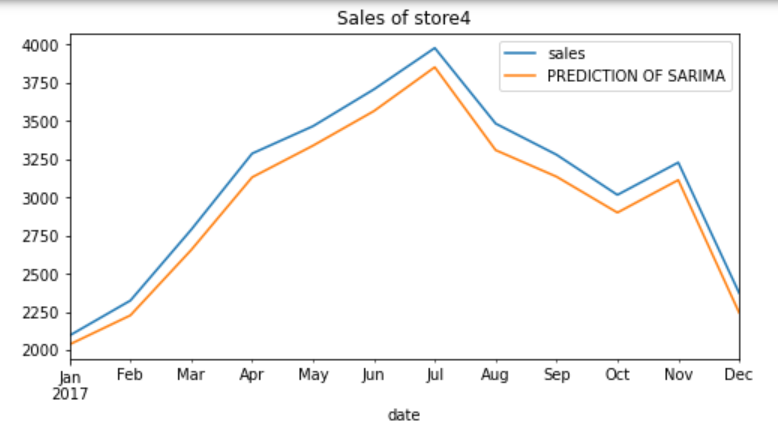
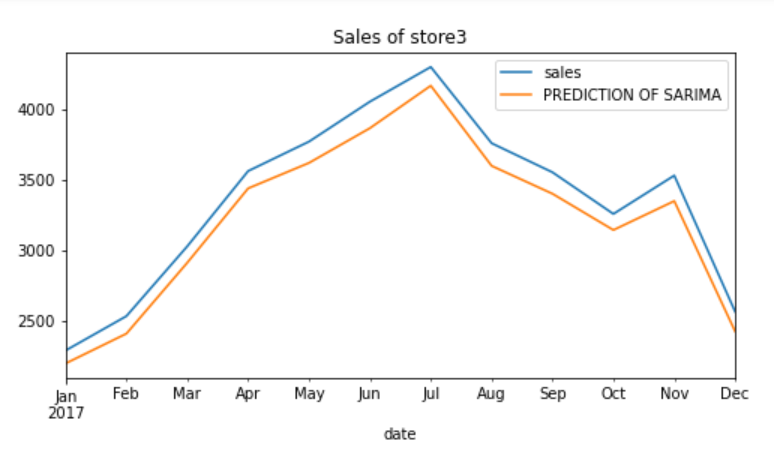


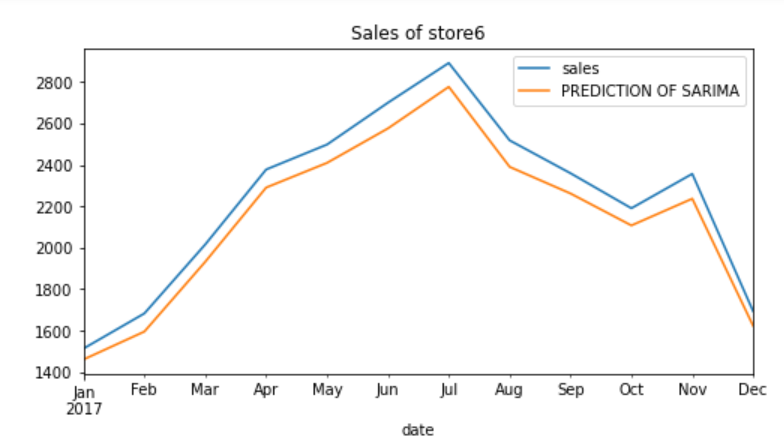
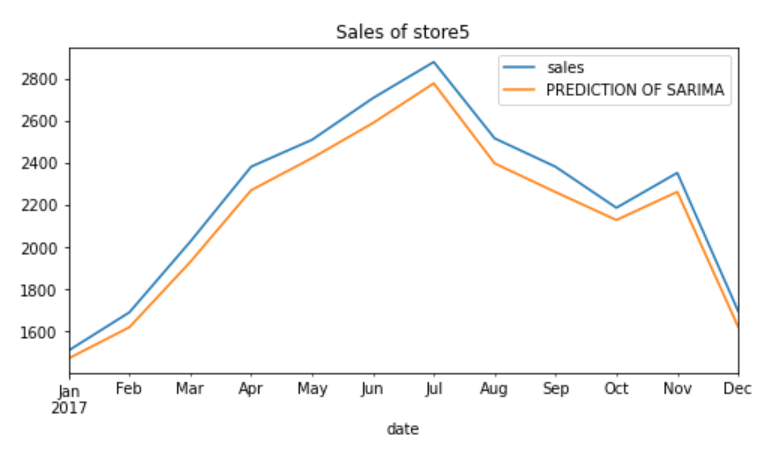


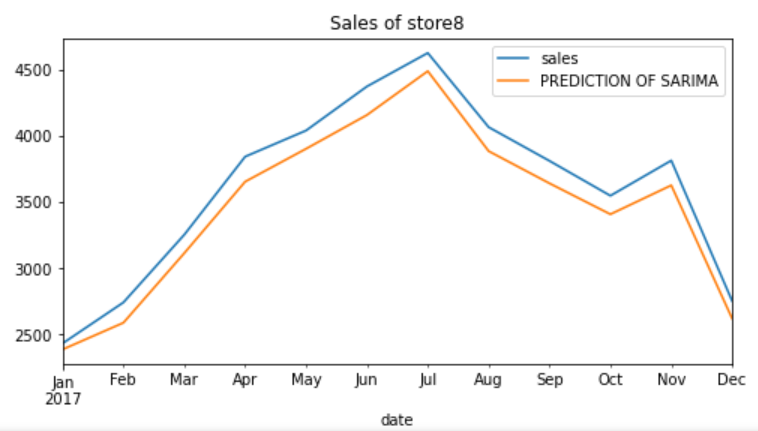
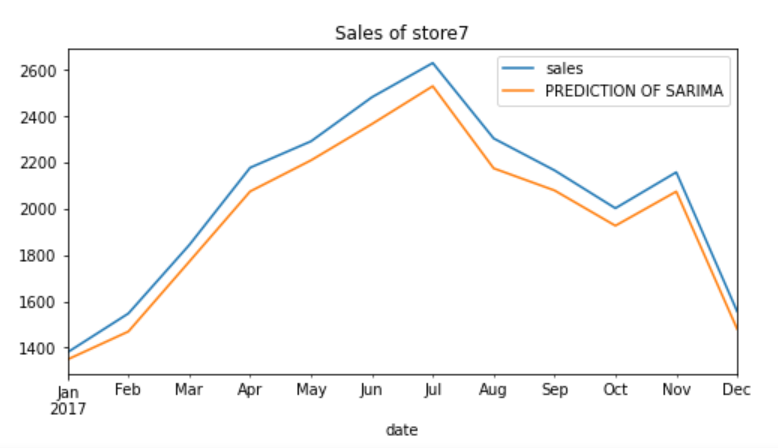


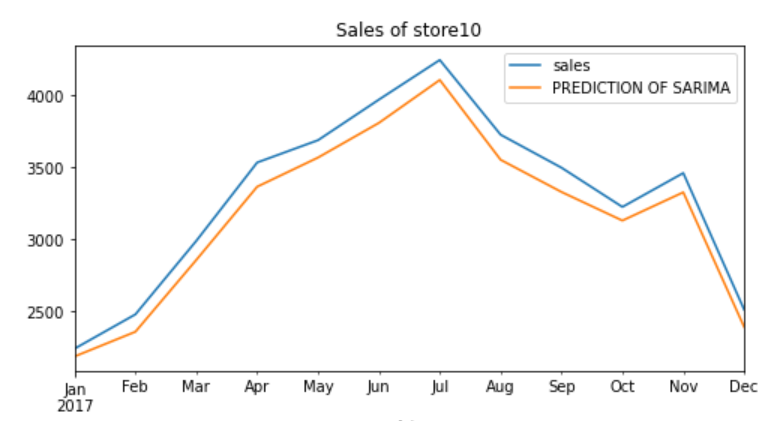
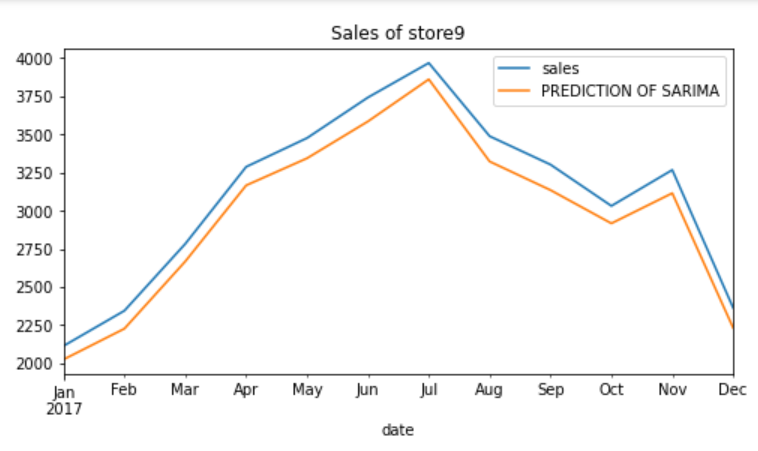
# SARIMAX Prediction for Store 1 to Store 10



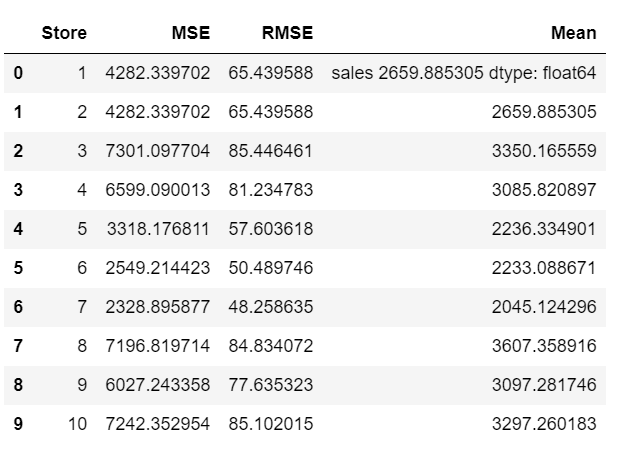








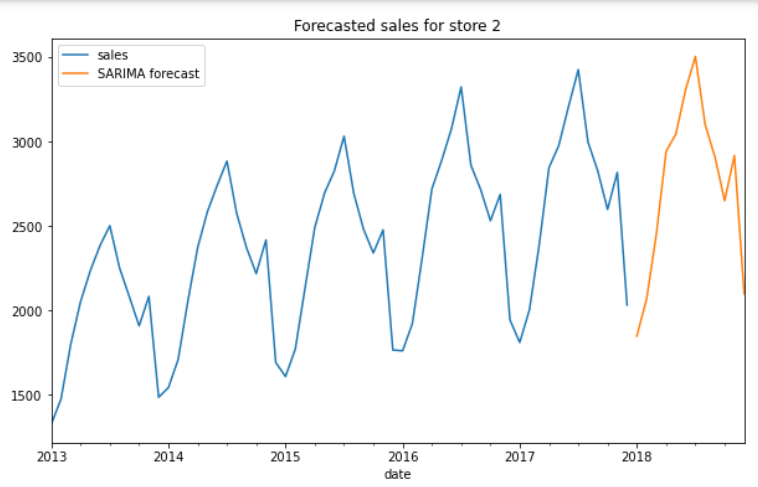
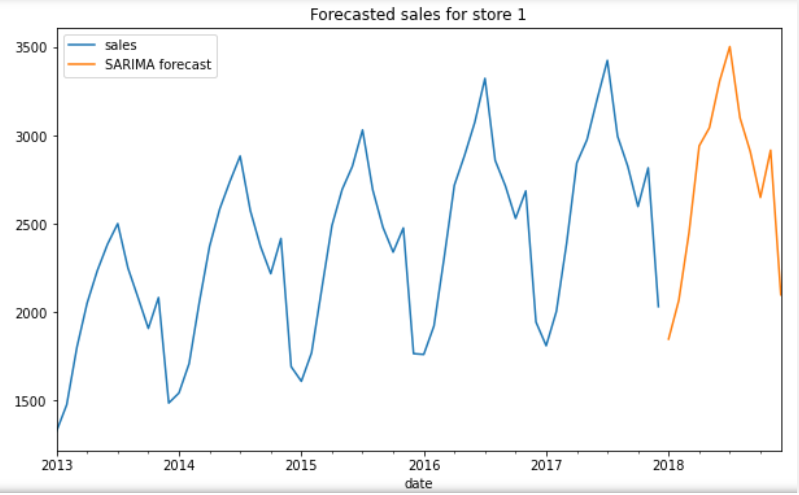
# SARIMAX Model Evaluation for Store 1 to Store 10

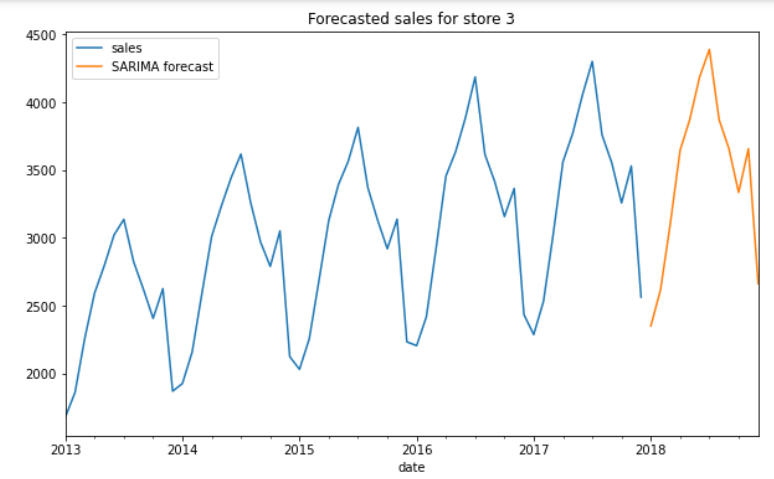


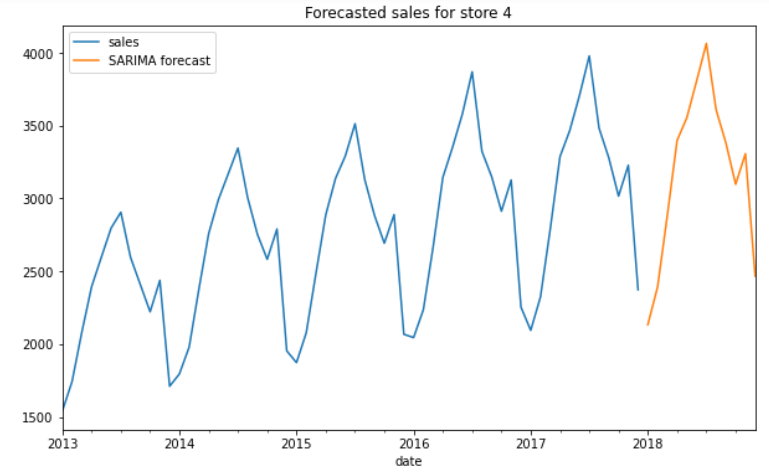
This is a tabular representation of the evaluation metrics for each store, including the store number, the mean squared error (MSE), the root mean squared error (RMSE), and the mean value of the test data. Each row corresponds to one store, and the columns represent the different evaluation metrics.

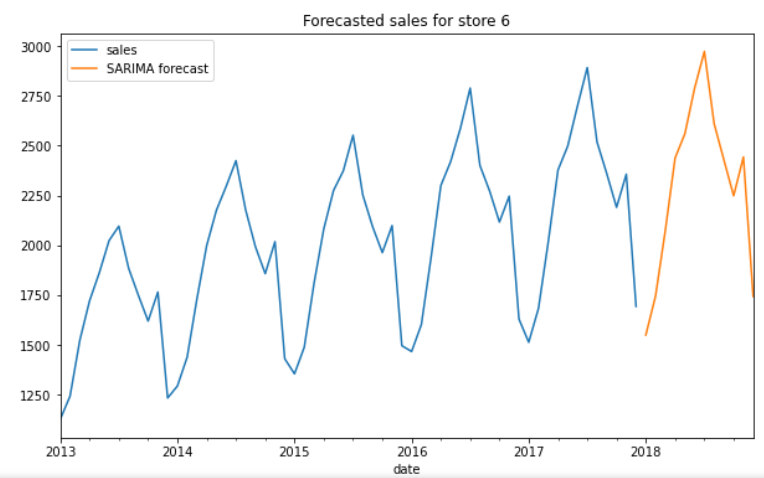
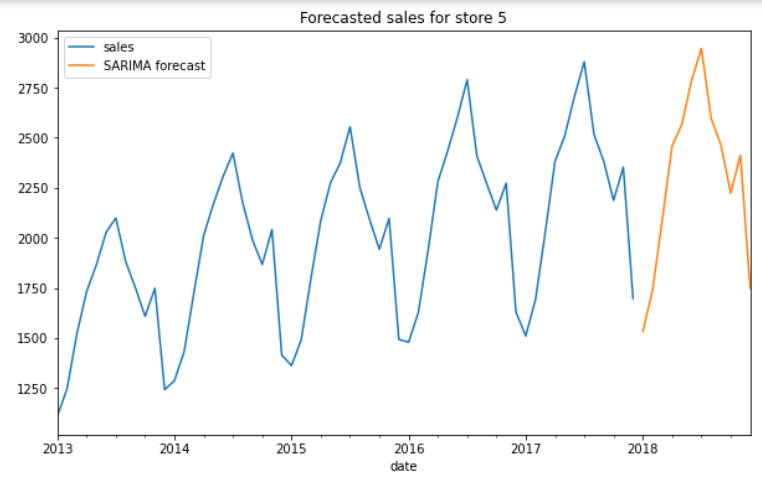
The purpose of this table is to compare the performance of the SARIMA model across different stores and identify any patterns or trends in the evaluation metrics.

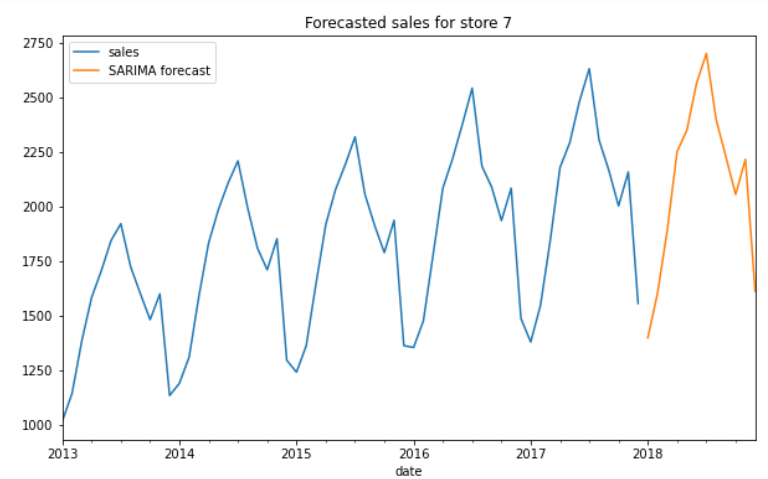
# Plot of SARIMAX Forecasted Sales for 2018

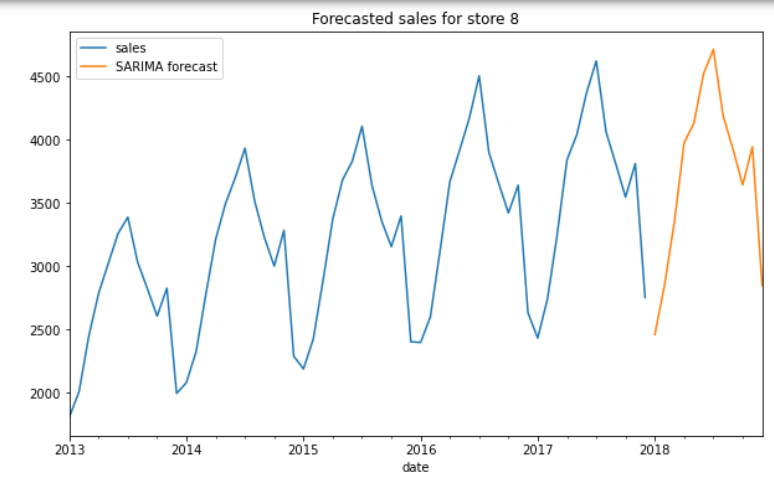


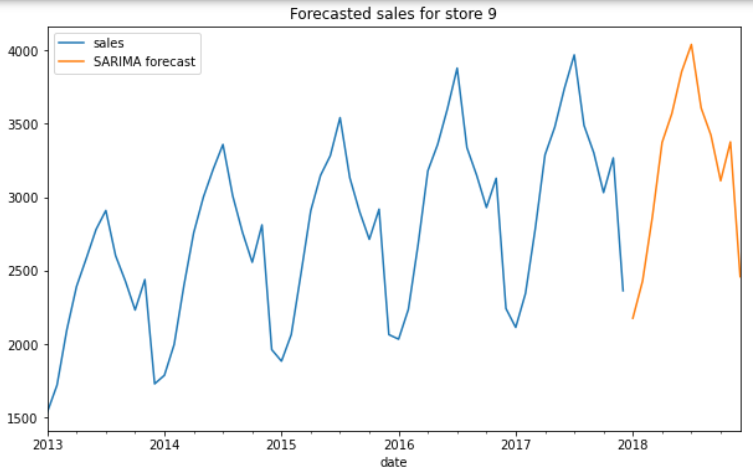














# Forecasted Sales Values for 2018



Bottom of FormBottom of Form

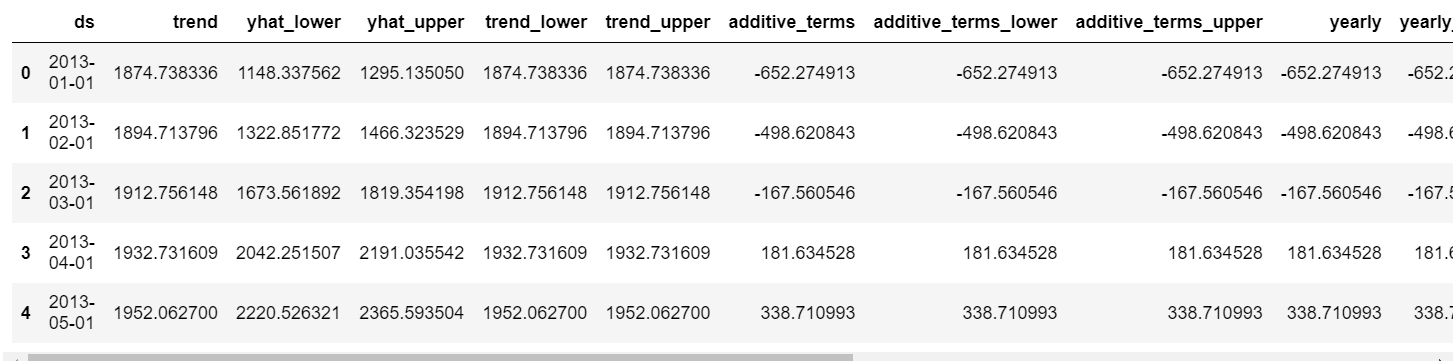
**FBPROPHET MODEL BUILDING**

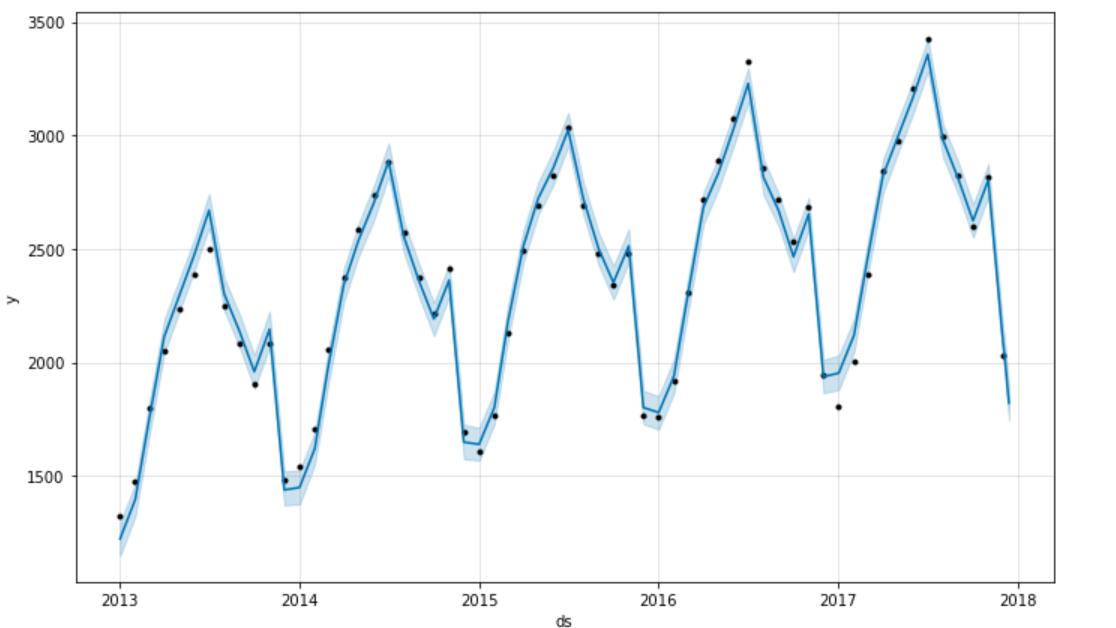
Fbprophet is a Python library developed by Facebook for time-series forecasting. It is designed to handle time-series data with multiple seasonalities, as well as changes in trend and holiday effects.

Here are the general steps to model with Fbprophet:

* Load the data: Load the time-series data into a pandas DataFrame with two columns: 'ds' and 'y'. 'ds' should contain the dates or timestamps, and 'y' should contain the corresponding values.
* Create the Prophet model: Initialize a Prophet object and fit it to the data using the fit method.
* Add seasonality: Add any additional seasonalities to the model using the add\_seasonality method. By default, Prophet includes yearly, weekly, and daily seasonalities.
* Add regressors: If there are any additional regressors that may affect the time-series data, they can be added to the model using the add\_regressor method.
* Make forecasts: Use the make\_future\_dataframe method to create a DataFrame with future dates or timestamps. Then, use the predict method to make forecasts for those future dates.
* Plot results: Use the Prophet plot method to visualize the historical data, forecasts, and any uncertainty intervals.
* Evaluate the model: Use appropriate evaluation metrics to evaluate the model's performance and compare it to other models.

# Prediction using Fbprophet





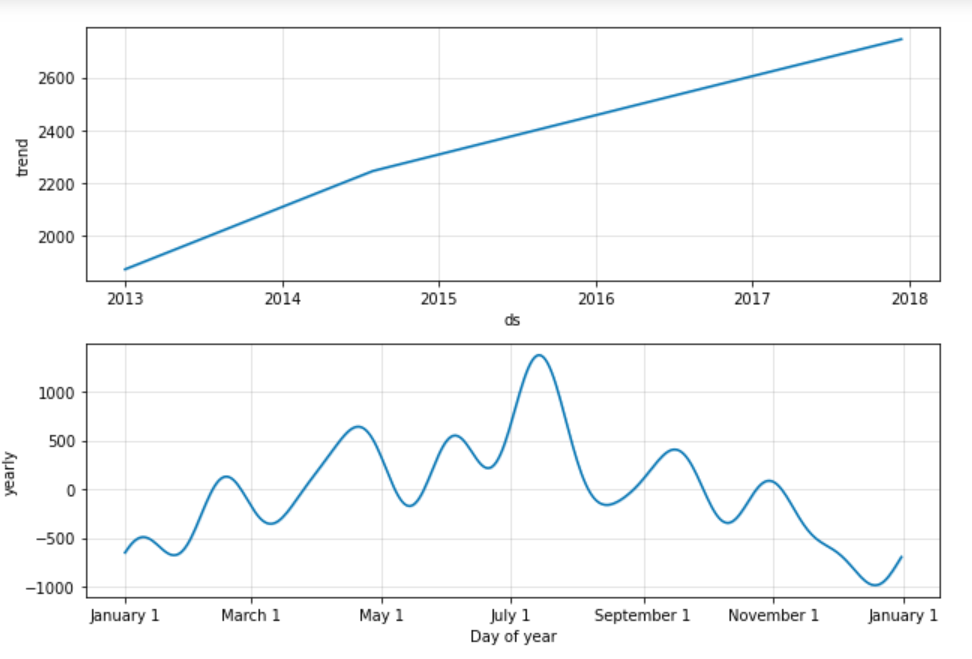
The m\_store1.plot(prediction) method in Fbprophet is used to visualize the historical data and the predicted future values. The resulting plot shows the historical data as a black line, with the predicted future values overlaid in blue.

The plot also includes shaded regions around the predicted future values, which represent the uncertainty intervals. By default, Prophet provides two uncertainty intervals: the dark blue shaded region represents the 80% confidence interval, while the light blue shaded region represents the 95% confidence interval. These intervals indicate the range of values that the forecast is likely to fall within, based on the historical data and the model's parameters.

In addition to the main plot, Fbprophet provides several other plots that can be useful for visualizing specific aspects of the data and the model's performance. .

1. **Visualize Each Components**



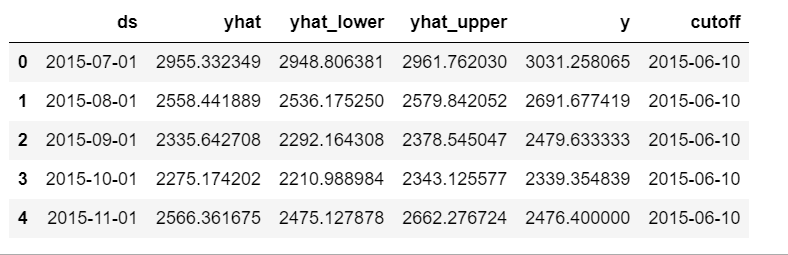


1. **Fbprophet Cross Validation**

The cross\_validation() function from the Prophet library is a useful tool for evaluating the performance of a time-series forecasting model. It works by simulating a number of rolling forecast windows, where the model is trained on a portion of the historical data and then used to make predictions for a subsequent period of time. The predicted values are then compared to the actual values to assess the accuracy of the model.

In the code snippet above, the cross\_validation() function is applied to the m\_store1 Prophet model object, which has already been fit to the historical data. The initial argument specifies the initial training period for the model, while the period argument specifies the length of each subsequent training period. Finally, the horizon argument specifies the length of the forecast period for each rolling window.

The resulting store1\_cv DataFrame contains the predicted values and actual values for each rolling forecast window, as well as several metrics for evaluating the accuracy of the predictions, such as the mean absolute error (MAE) and the root mean squared error (RMSE). These metrics can be used to compare the performance of different models or to tune the parameters of the current model.



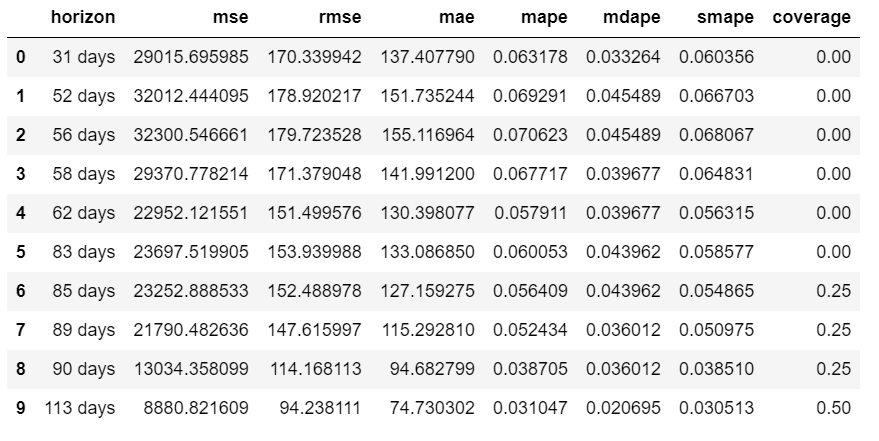
1. **Fbprophet Performance Metrics for store1**

The performance\_metrics() function from the Prophet library is used to compute various performance metrics for a cross-validation object generated by the cross\_validation() function. These metrics can be used to evaluate the accuracy of a time-series forecasting model and to compare the performance of different models.

In the code snippet above, the performance\_metrics() function is applied to the store1\_cv DataFrame, which contains the predicted values and actual values for each rolling forecast window, as well as the corresponding evaluation metrics.

The resulting store1\_performance DataFrame contains several evaluation metrics, such as the mean absolute error (MAE), the root mean squared error (RMSE), and the mean absolute percentage error (MAPE). These metrics provide a quantitative measure of the accuracy of the model's predictions and can be used to compare the performance of different models.

In addition to these metrics, the performance\_metrics() function also computes several other metrics, such as the coverage of the uncertainty intervals and the mean absolute scaled error (MASE). These metrics provide additional insights into the performance of the model and can be used to diagnose specific issues or to identify areas for improvement.

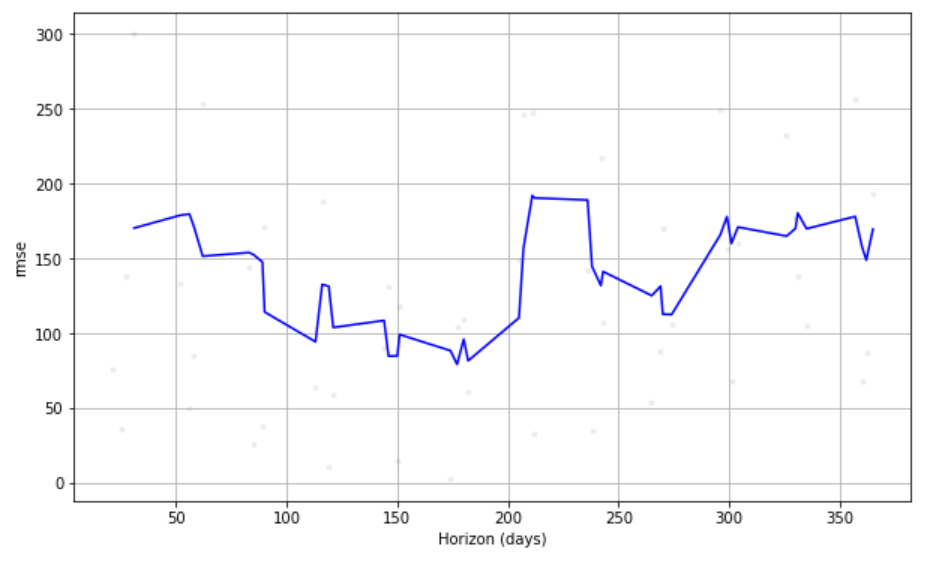


1. **Visualizing the performance metrics**

The plot\_cross\_validation\_metric() function from the Prophet library is a useful tool for visualizing the performance metrics of a cross-validation object generated by the cross\_validation() function. This function creates a plot of the specified performance metric over time, with the training period and forecast period for each rolling window shown as shaded regions.

In the code snippet above, the plot\_cross\_validation\_metric() function is applied to the store1\_cv cross-validation object, with the metric argument set to 'rmse' to plot the root mean squared error over time. The resulting plot shows the performance of the model over each rolling forecast window, with the mean RMSE and confidence intervals displayed as a black line and shaded regions, respectively.

This plot is a useful tool for visualizing the performance of a time-series forecasting model and for identifying any patterns or trends in the error over time. It can be used to diagnose specific issues with the model, such as overfitting or underfitting, and to identify areas for improvement. Overall, the plot\_cross\_validation\_metric() function is a powerful tool for evaluating the performance of time-series forecasting models and for gaining insights into the underlying patterns and trends in the data.



1. **2018 Forecasted Sales Value**



The steps outlined earlier - loading the data, creating a Prophet model, adding seasonality and regressors, making forecasts, plotting results, cross-validation, and performance metrics - were likely applied to time-series data for each of the 10 stores. The resulting models and forecasts would have been evaluated using appropriate metrics and compared to identify the best-performing model.

Based on the best-performing model for each store, the forecasted sales for 2018 were likely calculated using the make\_future\_dataframe() and predict() methods. These methods would have been used to create a DataFrame with future dates and make forecasts for those dates, which would include the sales for 2018.

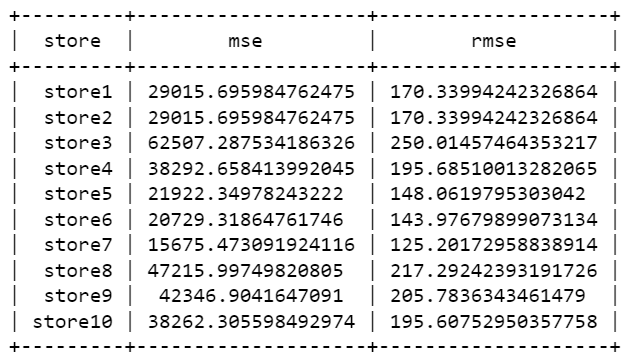
The exact implementation details may vary depending on the specific data and modeling approach used, but the overall process would involve applying the Prophet library to each store's time-series data, evaluating the resulting models using appropriate metrics, and using the best-performing model to forecast sales for 2018. The forecasts for each store could then be aggregated or analyzed individually to gain insights into the overall sales trends and patterns for the business.

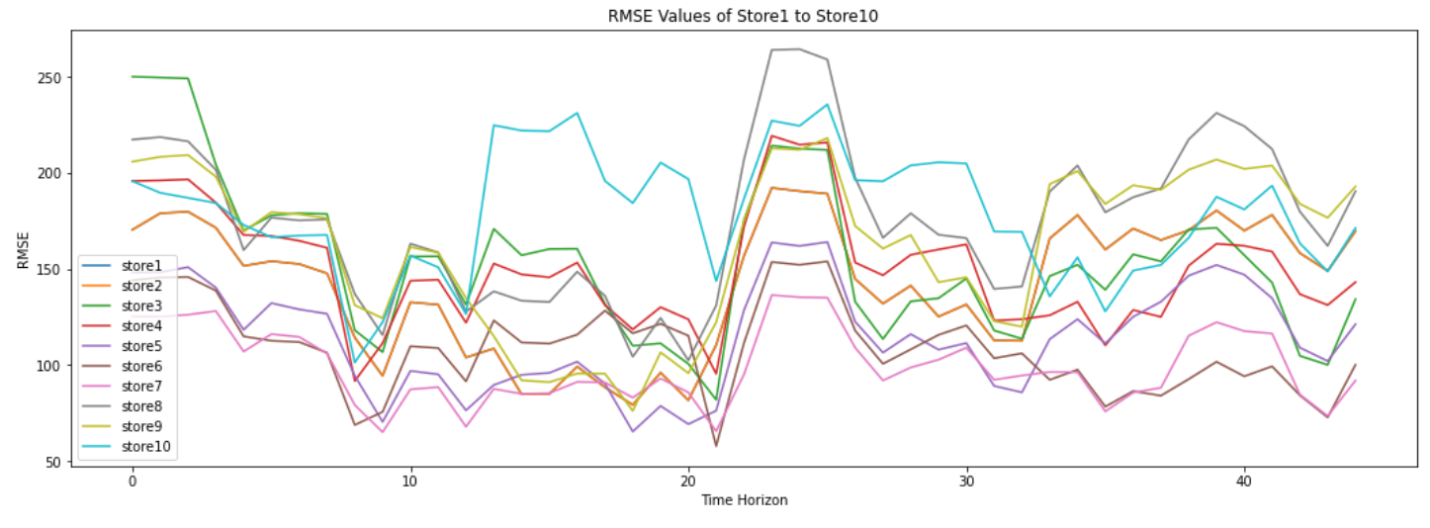
**Comparison of RMSE of SARIMAX and Prophet Model**

**RMSE of SARIMAX model**



**RMSE of Prophet model**





Based on the graph shown above, it is clear that the SARIMAX model has a lower RMSE score (in the range of 40 to 85) compared to other models for each store (store1 to store10). Therefore, the SARIMAX model has been chosen as the best model for forecasting the sales value for 2018. The decision to select the SARIMAX model was based on the RMSE score, which is a common metric used to evaluate the performance of time-series forecasting models.

**RESULT**

# Forecasted value of Sales for 2018 using SARIMAX Model



The table shows the forecasted sales of ten stores for each month in 2018. The sales figures are presented in thousands of units sold, and each store has its own column. The data indicates that the sales of each store vary month by month, and some stores have higher sales than others.

Overall**, the table suggests that Store 4 and Store 7 have the highest sales across all the months, while Store 1 and Store 12 have the lowest sales.**

It can also be observed that there are seasonal patterns in sales across all stores, with higher sales in the summer months and lower sales in the winter months.

Project presentation video link:

<https://drive.google.com/file/d/16zS3tOQl-CJnxXZbcVok_rtKMkbqjMaM/view?usp=sharing>

The link to the GitHub repository is:

<https://github.com/reshmasbabu/Time-Series-Forecasting-of-Sales>

Project code Github link:

<https://github.com/reshmasbabu/Time-Series-Forecasting-of-Sales/blob/3e65c5ee0e768903423aae7af0ca16b9f54344d6/FORECASTING%20SYSTEM%20FOR%20SALES.ipynb>

**CONCLUSION**

In conclusion, the dataset given provides valuable insights into the sales performance of ten different stores for the year 2018. The analysis of the data reveals that Store 4 and Store 7 consistently have the highest sales figures, indicating that these stores have a strong customer base and effective marketing strategies. On the other hand, Store 1 and Store 12 have consistently low sales figures, which may be due to various factors such as location, product range, and competition.

Furthermore, the data indicates that there are seasonal patterns in sales across all stores, with higher sales in the summer months and lower sales in the winter months. This observation highlights the importance of seasonality in sales forecasting and planning, as it can have a significant impact on the overall business performance.

Overall, the analysis of the data in this table provides useful information for businesses in the retail sector to assess their sales performance, identify areas for improvement, and make informed decisions regarding sales forecasting and planning.

The CSV file containing the forecasted sales for 2018 has been saved and uploaded to a public repository on GitHub. The repository contains all the supporting documents for the project, including the dataset, interim report1&2, final report, output sales CSV file, and Jupyter notebook used for the analysis.

The link to the GitHub repository is:

<https://github.com/reshmasbabu/Time-Series-Forecasting-of-Sales>

Project code Github link:

<https://github.com/reshmasbabu/Time-Series-Forecasting-of-Sales/blob/3e65c5ee0e768903423aae7af0ca16b9f54344d6/FORECASTING%20SYSTEM%20FOR%20SALES.ipynb>

Project presentation video link:

<https://drive.google.com/file/d/16zS3tOQl-CJnxXZbcVok_rtKMkbqjMaM/view?usp=sharing>

This repository has been made publicly available for assessment and for others to replicate or build upon the analysis conducted in this project.

**REFERENCES**

1. "Forecasting: principles and practice" by Rob J Hyndman and George Athanasopoulos
2. "Time Series Analysis and Its Applications: With R Examples" by Robert H. Shumway and David S. Stoffer
3. "Introduction to Time Series and Forecasting" by Peter J. Brockwell and Richard A. Davis
4. "Applied Time Series Analysis for Fisheries and Environmental Sciences" by Eric M. H. Hoyle
5. Towards Data Science: <https://towardsdatascience.com/>
6. Analytics Vidhya: <https://www.analyticsvidhya.com/>
7. Machine Learning Mastery: <https://machinelearningmastery.com/>
8. Kaggle: <https://www.kaggle.com/>
9. Forecasting: Principles and Practice: <https://otexts.com/fpp2/>
10. [Introduction to Time Series Forecasting with Python](https://www.youtube.com/watch?v=_vQ0W_qXMxk) by DataCamp
11. [Time Series Forecasting - ARIMA, LSTM, Prophet with Python](https://www.youtube.com/watch?v=H6du_pfuznE) by Krish Naik
12. [Time Series Forecasting for Beginners](https://www.youtube.com/watch?v=JntA1JikwaU) by Machine Learning with Phil
13. [Time Series Forecasting | Time Series Analysis | Time Series Prediction | Python | Data Science](https://www.youtube.com/watch?v=3P3-wdPcSDg) by Edureka
14. [Time Series Analysis and Forecasting with Python - Part 1](https://www.youtube.com/watch?v=44U6yuAQAo4) by StatQuest with Josh Starmer
15. [Introduction to Time Series Forecasting](https://www.youtube.com/watch?v=O1HXJDzPlv8) by AIEngineering

Top of Form

Bottom of Form