INTERNSHIP: PROJECT REPORT

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

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."

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.

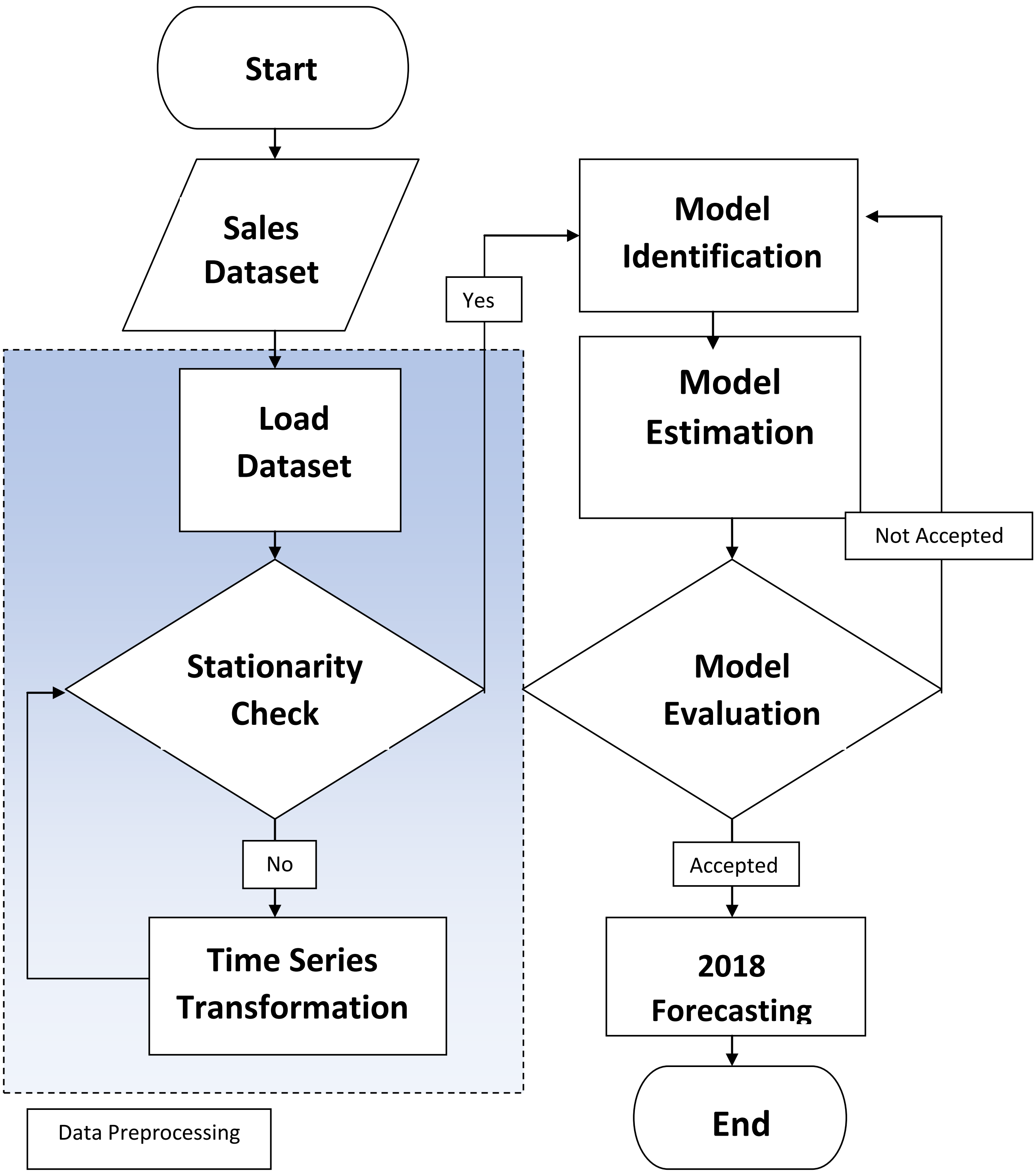
Some common techniques for time series forecasting include:

- 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



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.

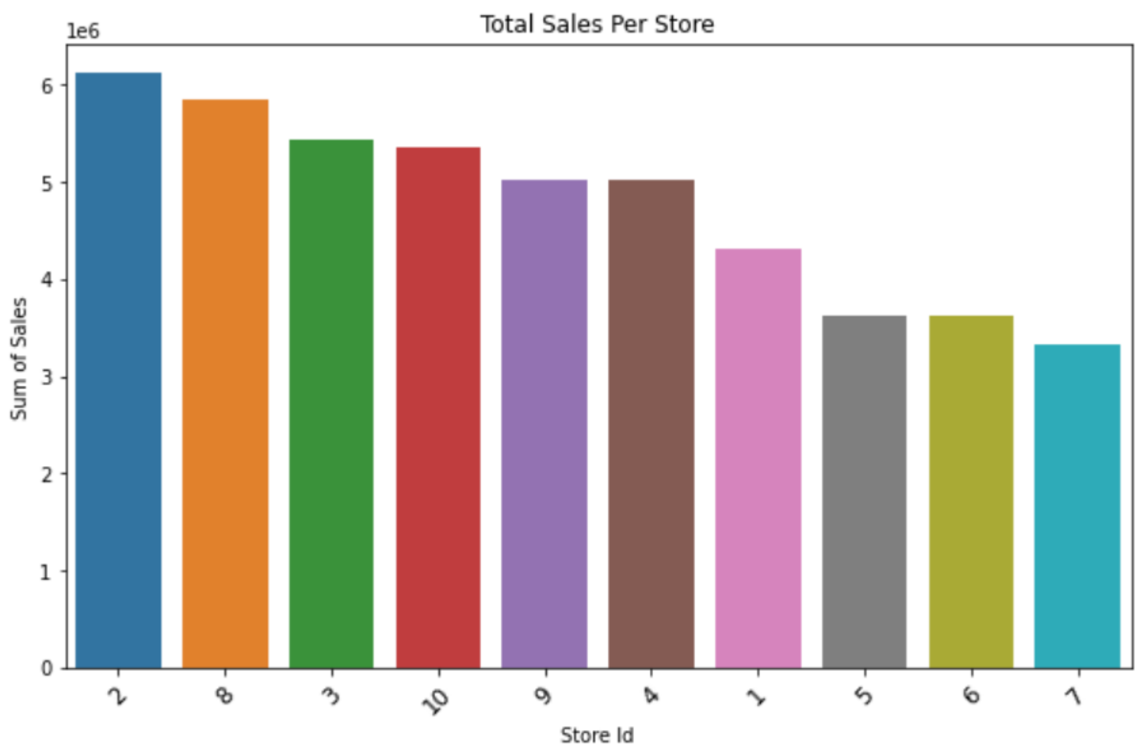


Figure: Total sales per store

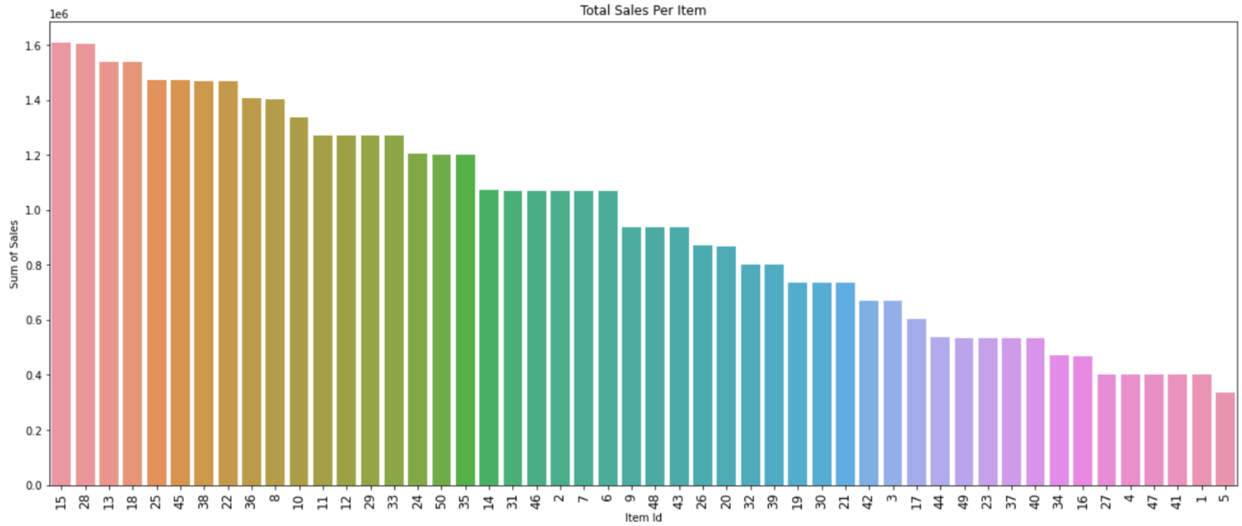


Figure: Total sales per item

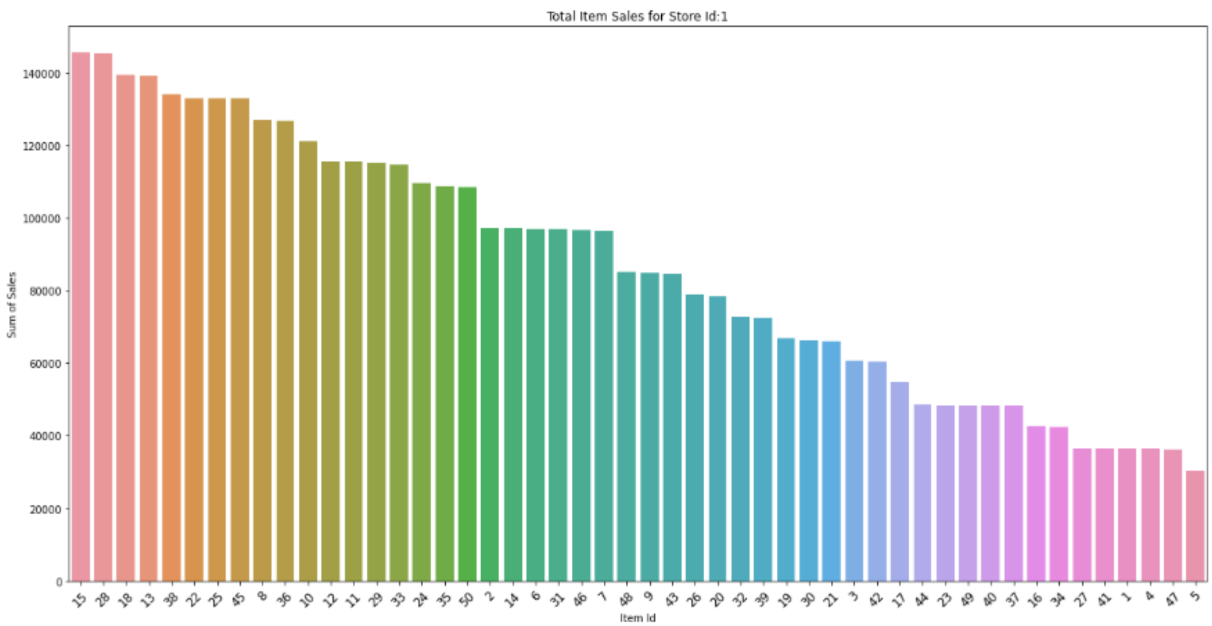


Figure: Total item sales for store 1

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.

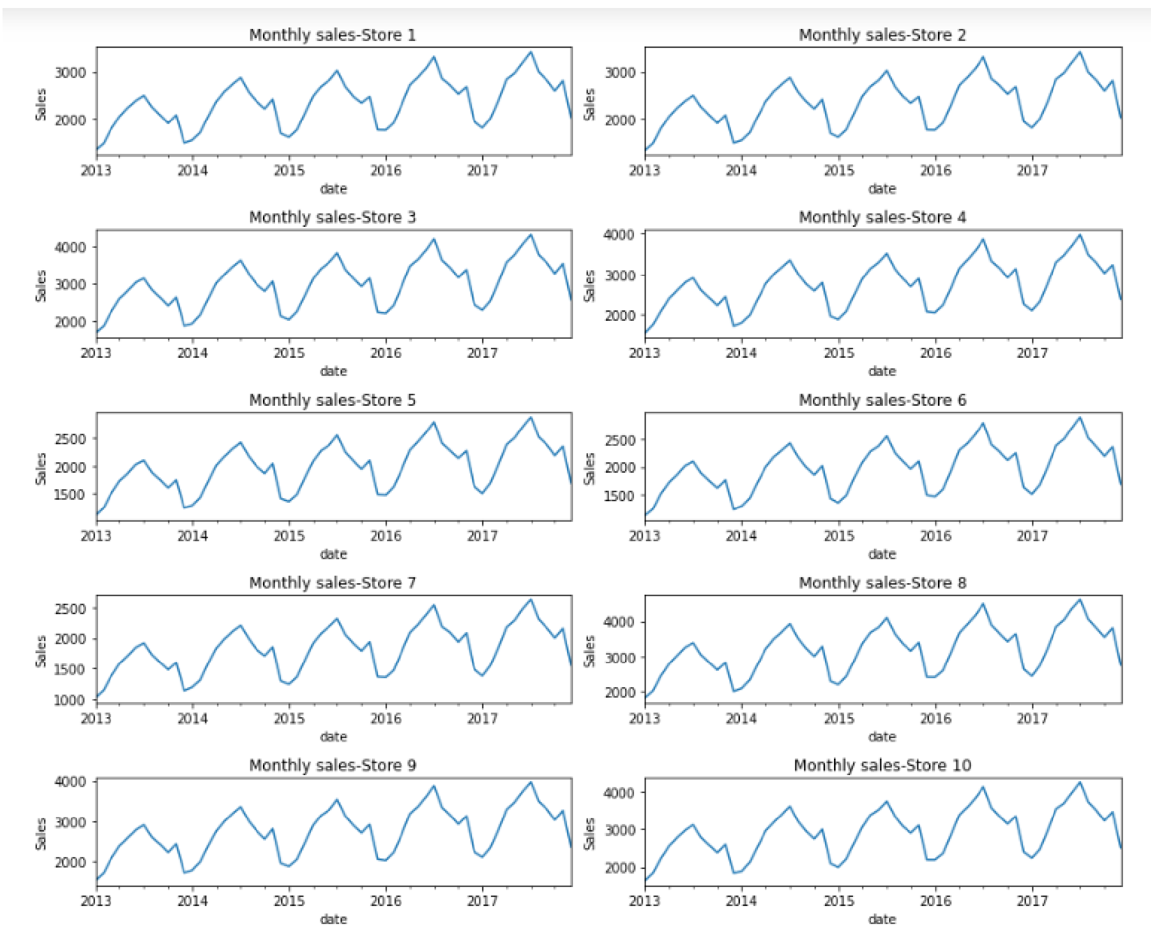


Figure: Monthly sales per store

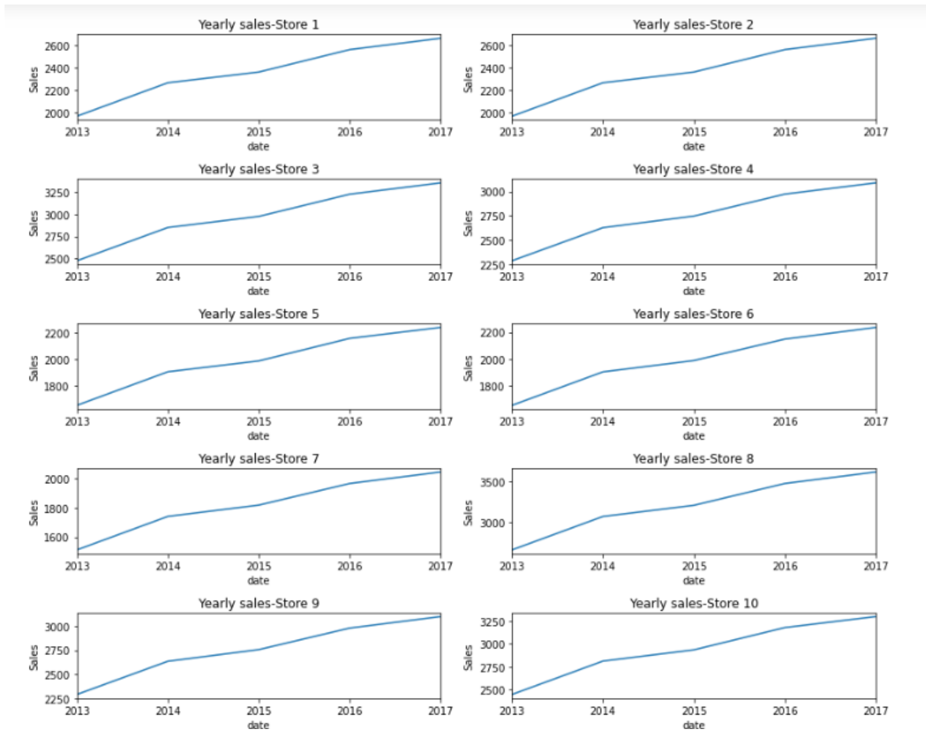


Figure: Yearly sales per store

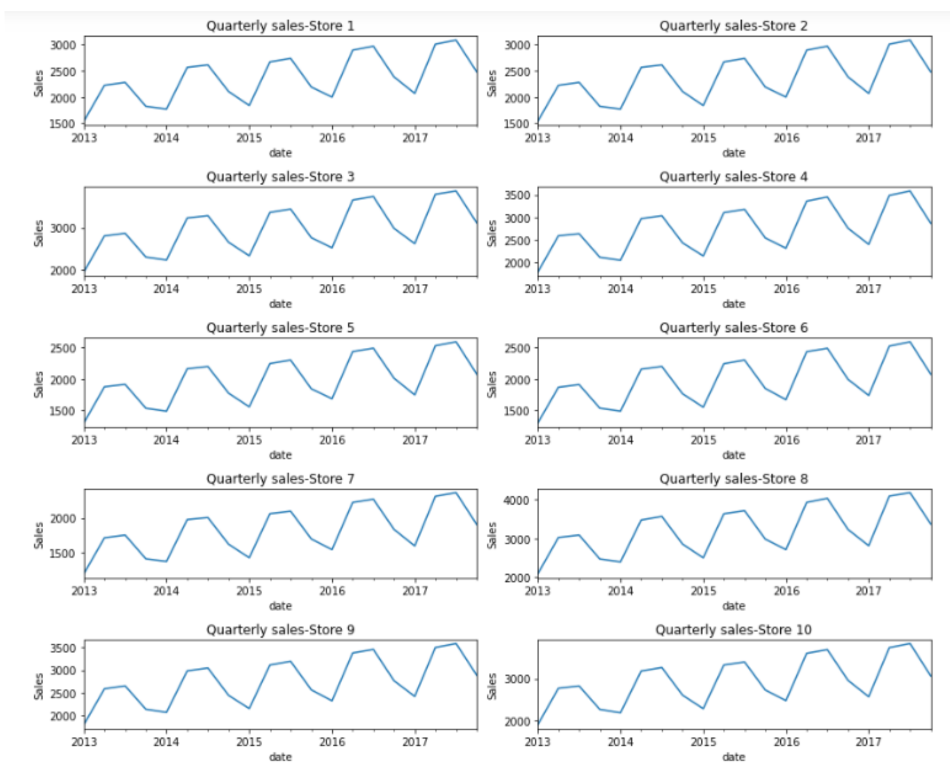
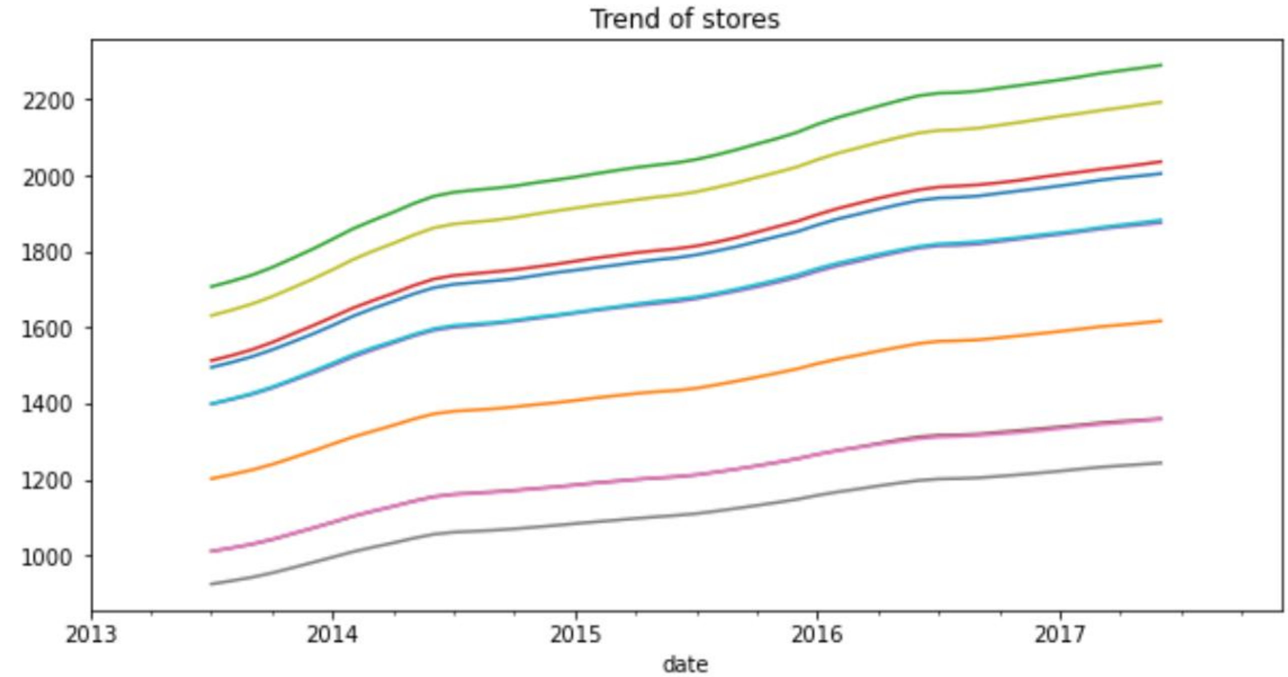


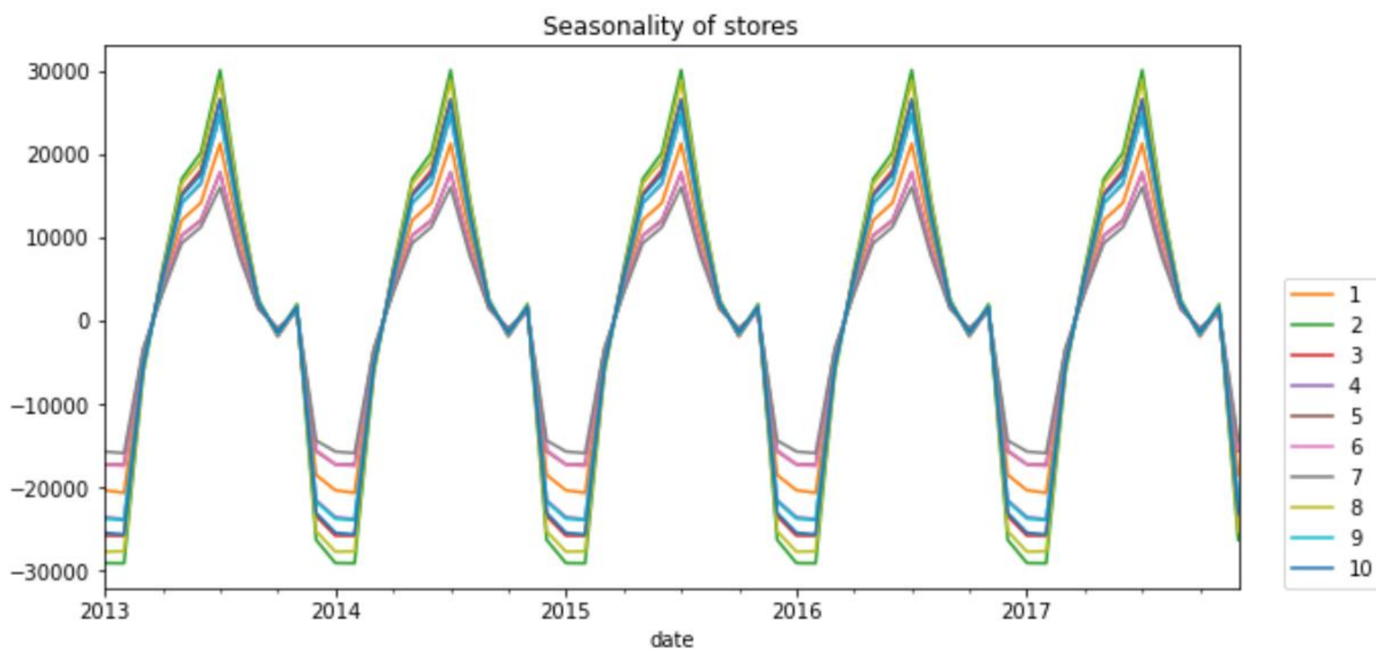
Figure: Quarterly sales per store

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 any recurring 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.

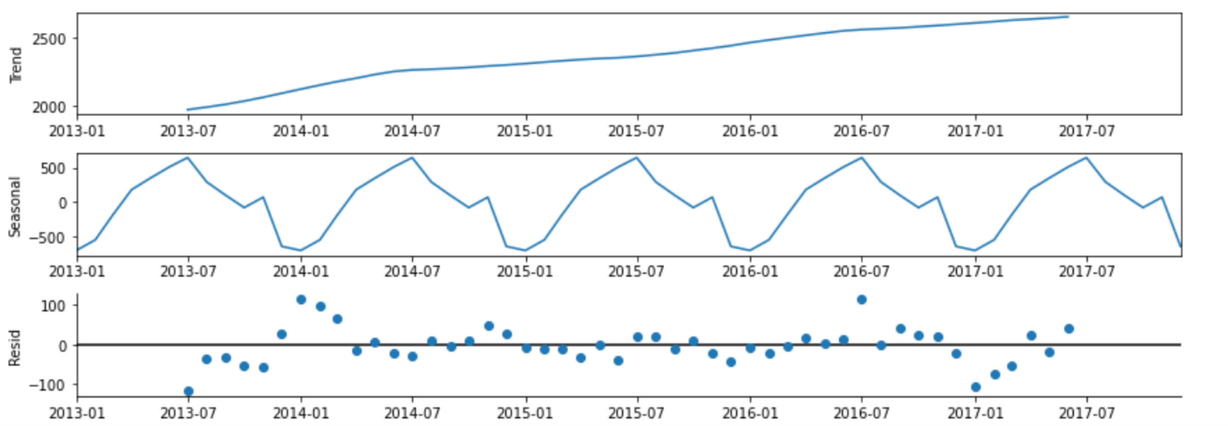

Trend of stores
2200
2000
1800
1600
1400
1200
1000
2013
2014
2015
2016
2017
date


Seasonality of stores
30000
20000
10000
0
−10000
−20000
−30000
2013
2014
2015
2016
2017
date
1
2
3
4
5
6
7
8
9
10

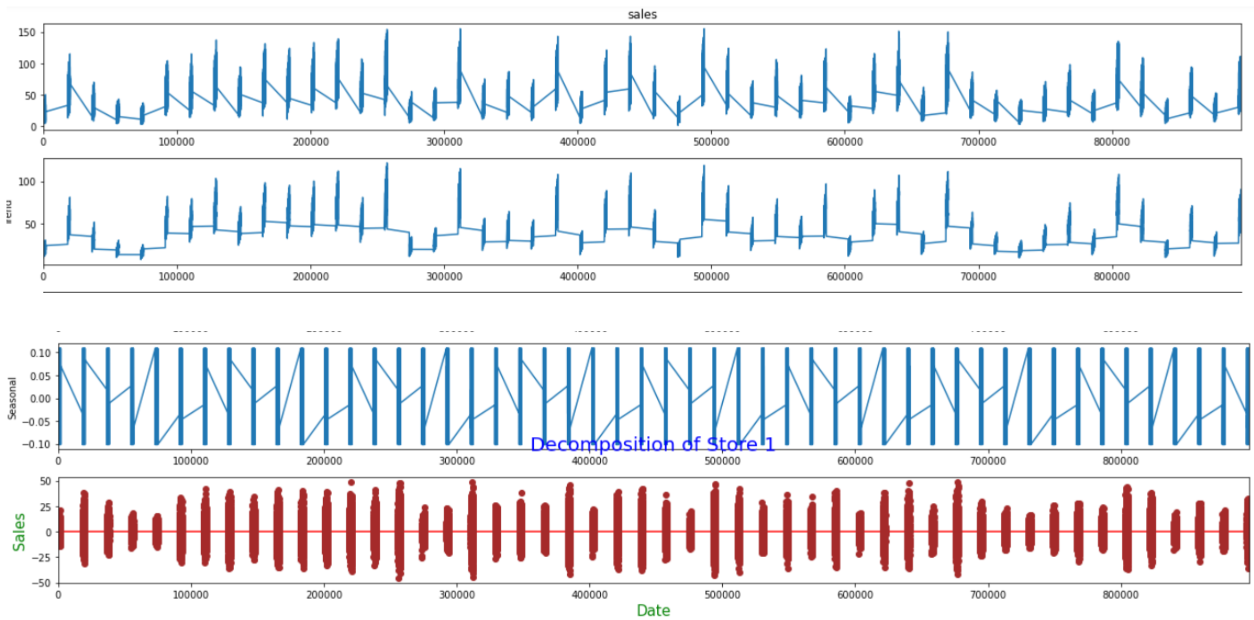
ETS (ERROR, TREND, SEASONALITY) DECOMPOSITION

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.

```
Augmented Dickey-Fuller Test:
ADF test statistic      -5.165600
p-value                  0.000010
# lags used             11.000000
# observations          48.000000
critical value (1%)     -3.574589
critical value (5%)     -2.923954
critical value (10%)    -2.600039
--------------------------------------------
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

```
Augmented Dickey-Fuller Test:
ADF test statistic      -5.165600
p-value                  0.000010
# lags used             11.000000
# observations          48.000000
critical value (1%)     -3.574589
critical value (5%)     -2.923954
critical value (10%)    -2.600039
--------------------------------------------
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -5.209288
p-value                   0.000008
# lags used              11.000000
# observations           48.000000
critical value (1%)      -3.574589
critical value (5%)      -2.923954
critical value (10%)     -2.600039
--------------------------------------------
--------------
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -5.511807
p-value                   0.000002
# lags used              11.000000
# observations           48.000000
critical value (1%)      -3.574589
critical value (5%)      -2.923954
critical value (10%)     -2.600039
--------------------------------------------
--------------
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -5.148333
p-value                   0.000011
# lags used              11.000000
# observations           48.000000
critical value (1%)      -3.574589
critical value (5%)      -2.923954
critical value (10%)     -2.600039
--------------------------------------------
--------------
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -5.501111
p-value                   0.000002
# lags used              11.000000
# observations           48.000000
critical value (1%)      -3.574589
critical value (5%)      -2.923954
critical value (10%)     -2.600039
--------------------------------------------
--------------
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -5.576794
p-value                   0.000001
# lags used              11.000000
# observations           48.000000
critical value (1%)      -3.574589
critical value (5%)      -2.923954
critical value (10%)     -2.600039
--------------------------------------------
--------------
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -5.323105
p-value                   0.000005
# lags used              11.000000
# observations           48.000000
critical value (1%)      -3.574589
critical value (5%)      -2.923954
critical value (10%)     -2.600039
--------------------------------------------
--------------
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

```
Augmented Dickey-Fuller Test:
ADF test statistic      -5.528050
p-value                  0.000002
# lags used             11.000000
# observations          48.000000
critical value (1%)     -3.574589
critical value (5%)     -2.923954
critical value (10%)    -2.600039
------------------------------------------
--------------
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

```
Augmented Dickey-Fuller Test:
ADF test statistic      -5.275954
p-value                  0.000006
# lags used             11.000000
# observations          48.000000
critical value (1%)     -3.574589
critical value (5%)     -2.923954
critical value (10%)    -2.600039
------------------------------------------
--------------
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

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.

	date	sales
0	2013-01-01	13696
1	2013-01-02	13678
2	2013-01-03	14488
3	2013-01-04	15677
4	2013-01-05	16237

	year_month	sales
0	2013-01	454904
1	2013-02	459417
2	2013-03	617382
3	2013-04	682274
4	2013-05	763242

	year	sales
0	2013	7941243
1	2014	9135482
2	2015	9536887
3	2016	10357160
4	2017	10733740

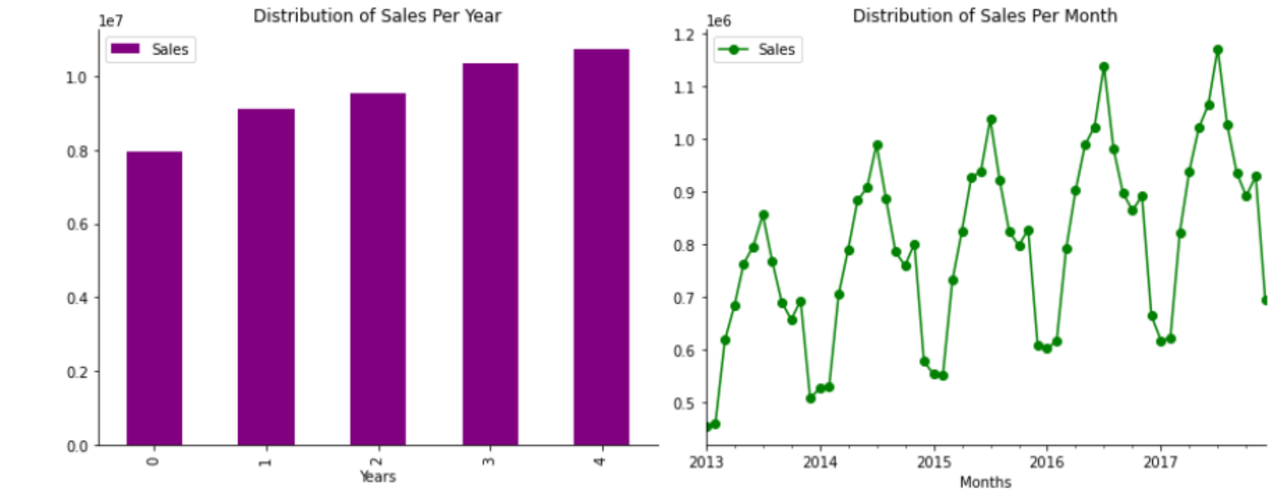
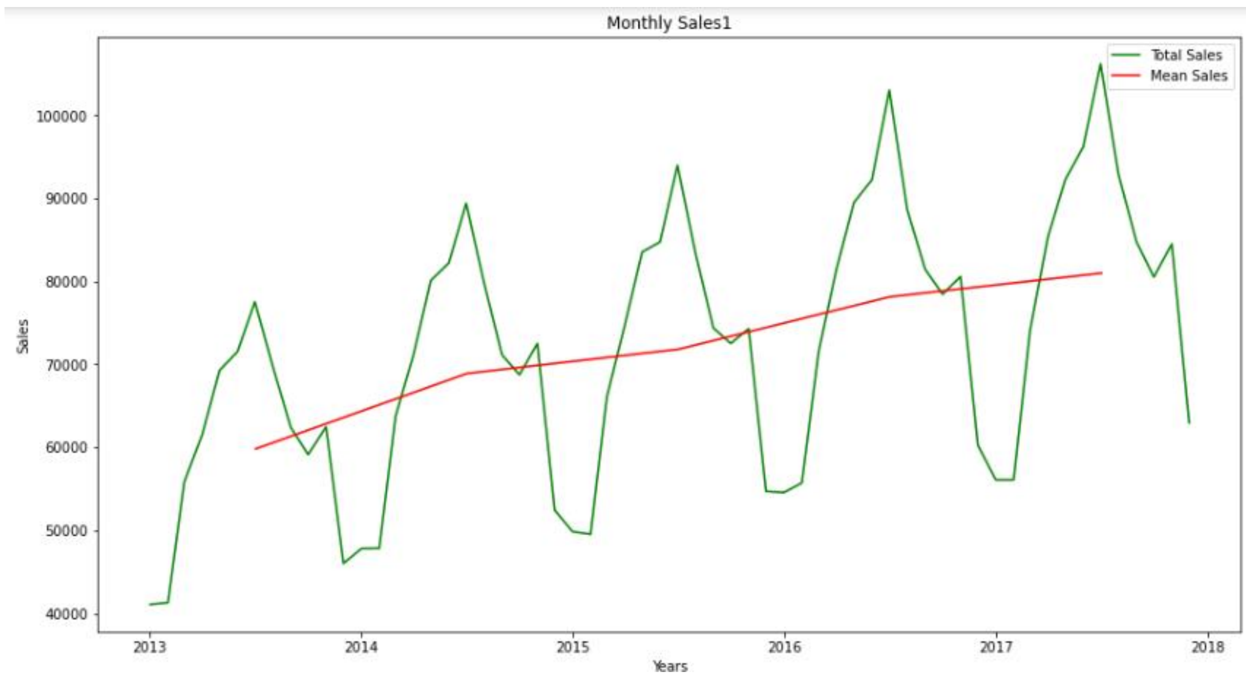


Figure: Daliy sales from 2013 to 2017



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.

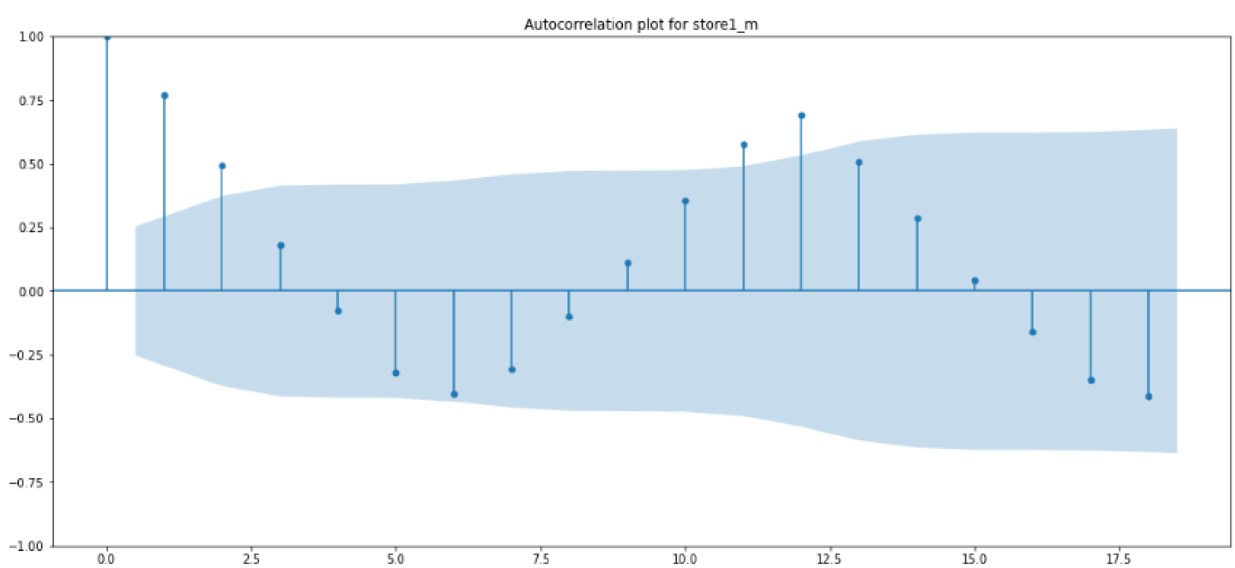
ARIMA MODEL BUILDING

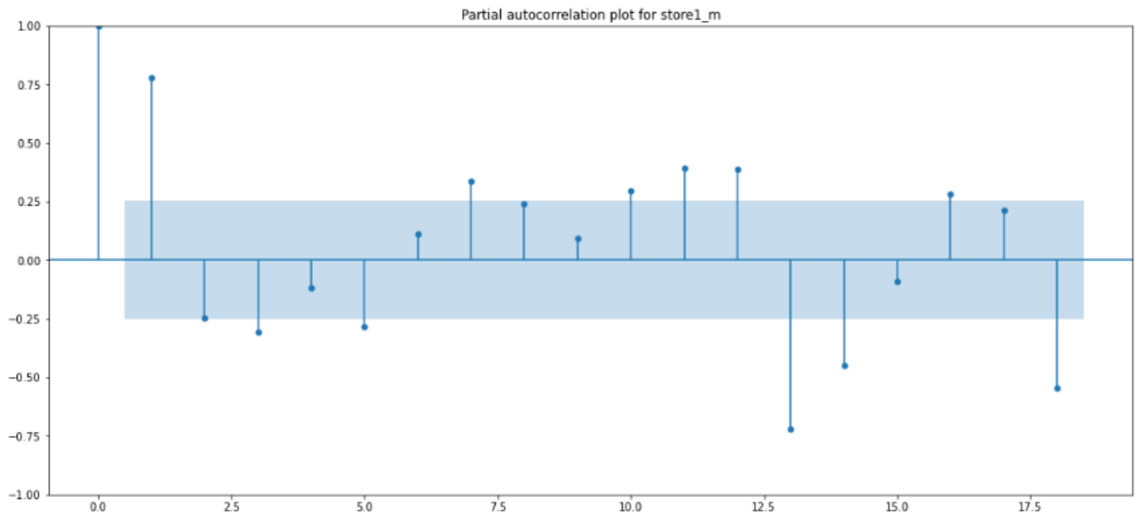
a) Finding Autocorrelation and Partial autocorrelation plots for all stores

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.





b) Splitting into Train and Test Data

```
store1_train=store1_m.iloc[:48]
store1_test=store1_m.iloc[48:]
```

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.

c) 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.

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,1,1)[12]             : AIC=inf, Time=1.12 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=509.051, Time=0.05 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=512.111, Time=0.17 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=512.275, Time=0.29 sec
 ARIMA(0,1,0)(1,1,0)[12]             : AIC=510.737, Time=0.17 sec
 ARIMA(0,1,0)(0,1,1)[12]             : AIC=510.853, Time=0.19 sec
 ARIMA(0,1,0)(1,1,1)[12]             : AIC=inf, Time=0.53 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=510.569, Time=0.17 sec
 ARIMA(0,1,1)(0,1,0)[12]             : AIC=510.533, Time=0.15 sec
 ARIMA(1,1,1)(0,1,0)[12]             : AIC=512.514, Time=0.25 sec
 ARIMA(0,1,0)(0,1,0)[12] intercept   : AIC=510.926, Time=0.08 sec

Best model:  ARIMA(0,1,0)(0,1,0)[12]
Total fit time: 3.398 seconds
```

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.

d) 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.

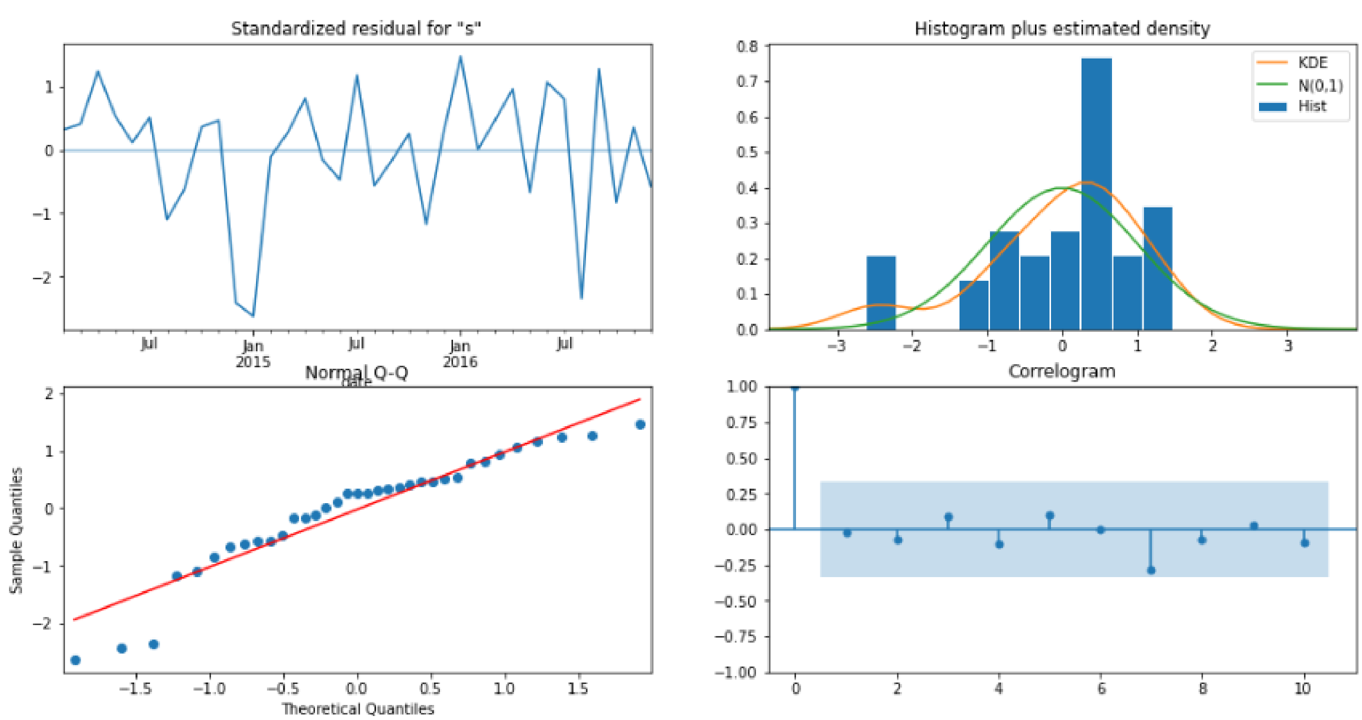
The Q-Q plot of the residuals, which compares the distribution of the residuals to a normal distribution. The points on the plot should fall approximately on a straight line.

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.

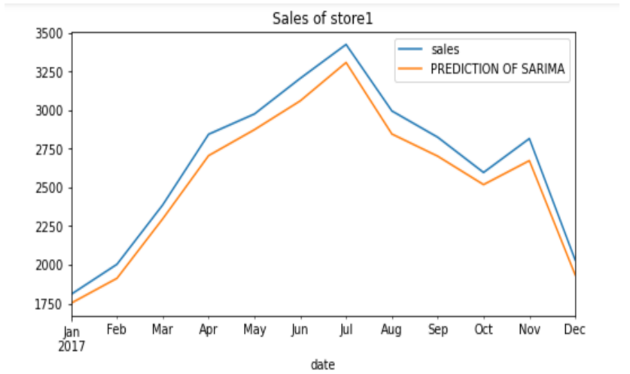
Overall, the summary and diagnostic plots provide useful information to assess the fit and accuracy of the SARIMAX model for each store's sales data.

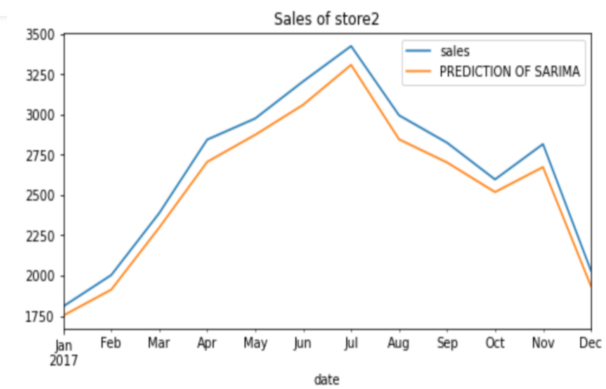
```
Store 1:
                                     SARIMAX Results
==========================================================================
=================
Dep. Variable:                                sales   No. Observations:
48
Model:             SARIMAX(0, 1, 0)x(0, 1, 0, 12)   Log Likelihood
-188.830
Date:                              Wed, 22 Feb 2023   AIC
379.660
Time:                                      10:50:48   BIC
381.215
Sample:                                  01-01-2013   HQIC
380.197
                                       - 12-01-2016
Covariance Type:                                opg
==========================================================================
```

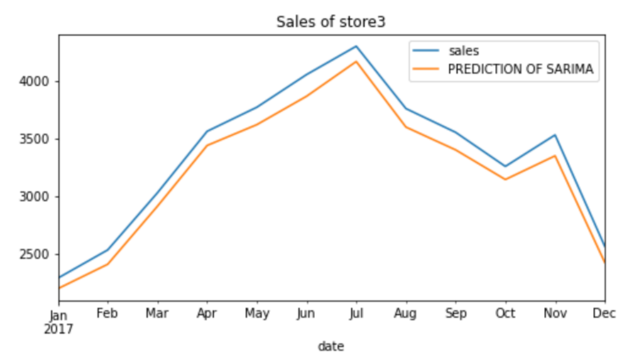
```
                 coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------
-----
sigma2      2842.2227    581.269      4.890      0.000    1702.957     398
1.489
==========================================================================
==========
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):
6.37
Prob(Q):                              0.90   Prob(JB):
0.04
Heteroskedasticity (H):               0.81   Skew:
-0.99
Prob(H) (two-sided):                  0.72   Kurtosis:
3.65
==========================================================================
```

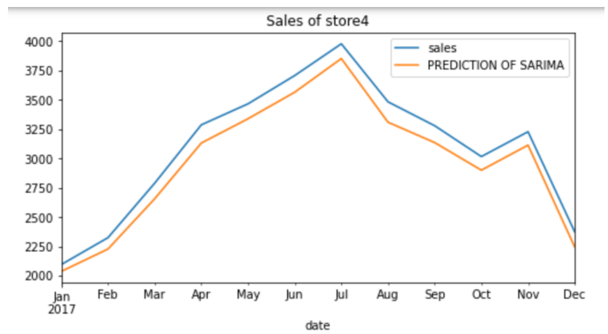


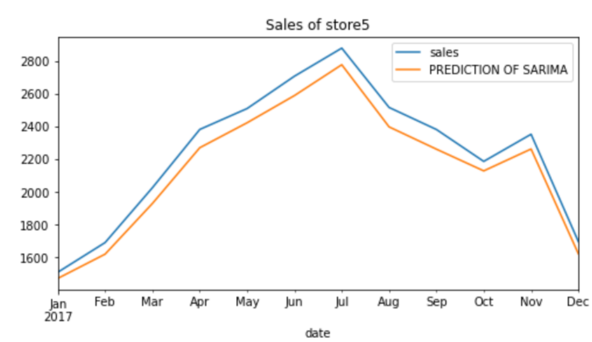
e) SARIMAX Prediction for Store 1 to Store 10

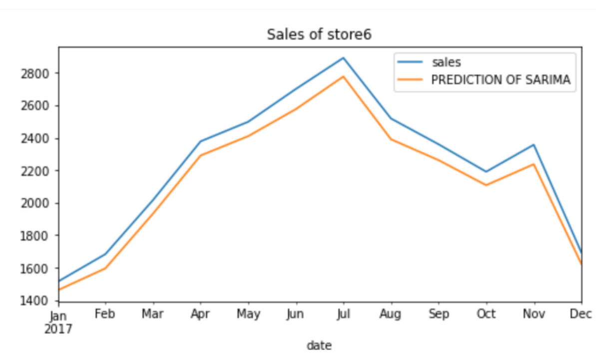


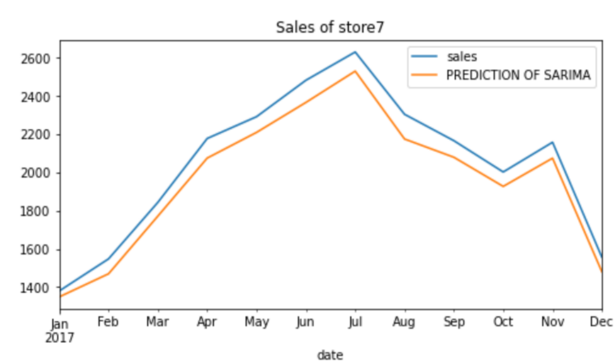


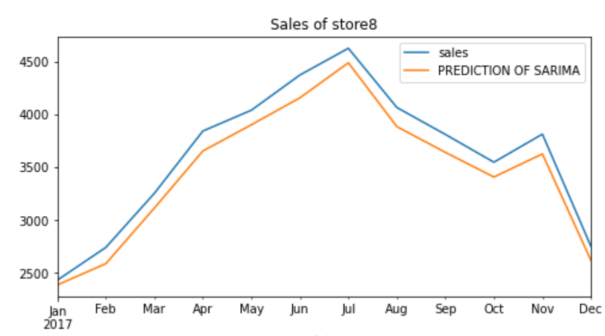


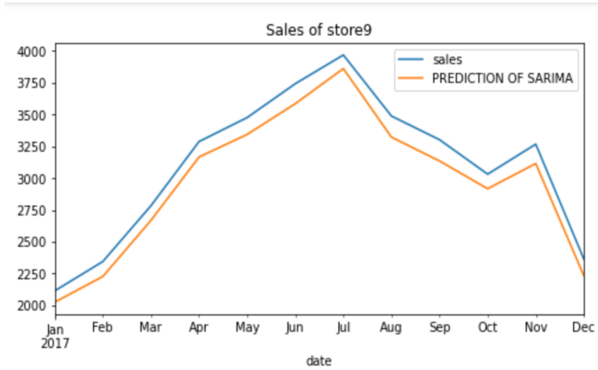


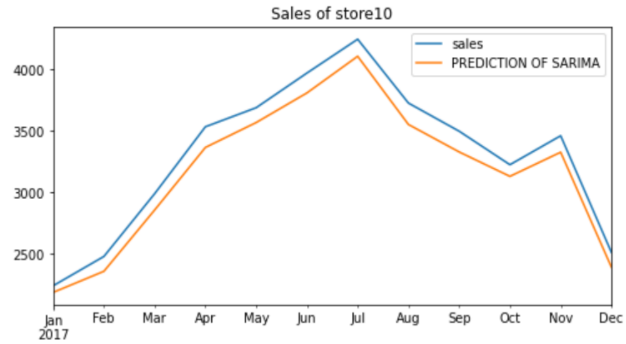












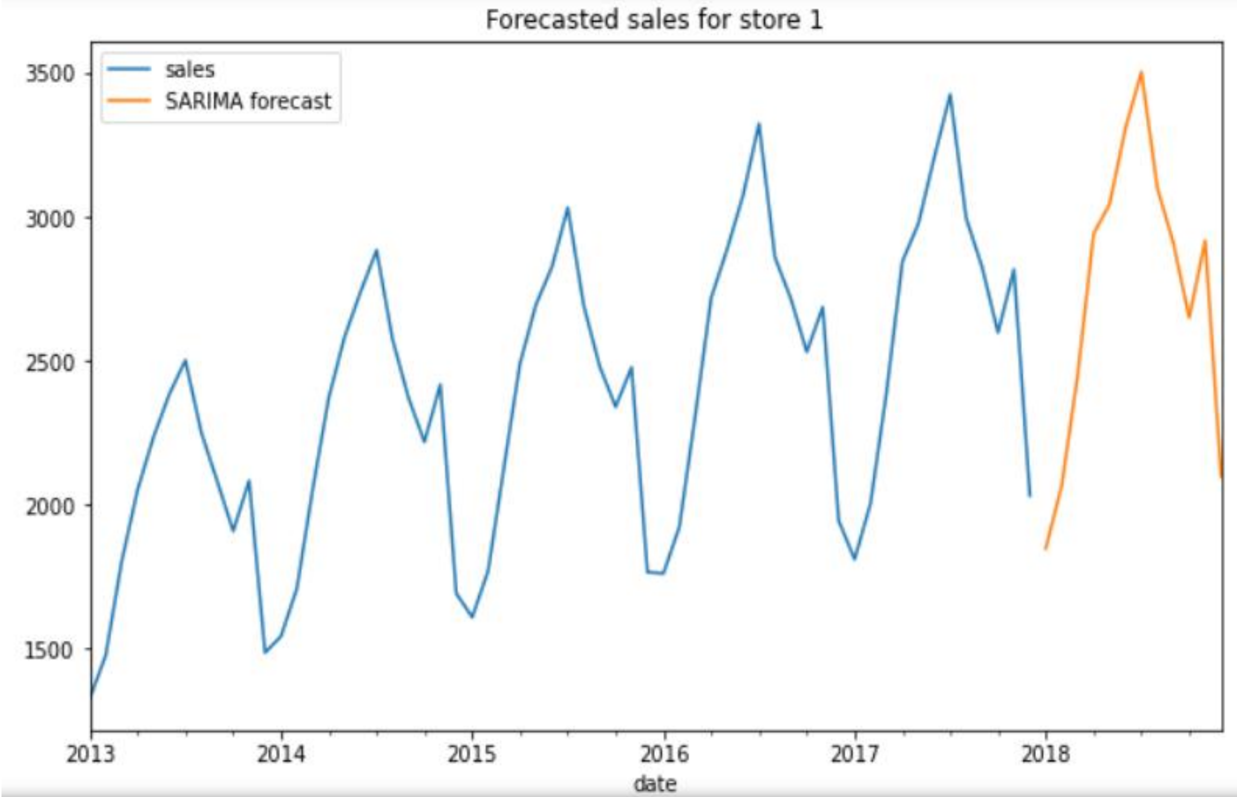
f) SARIMAX Model Evaluation for Store 1 to Store 10

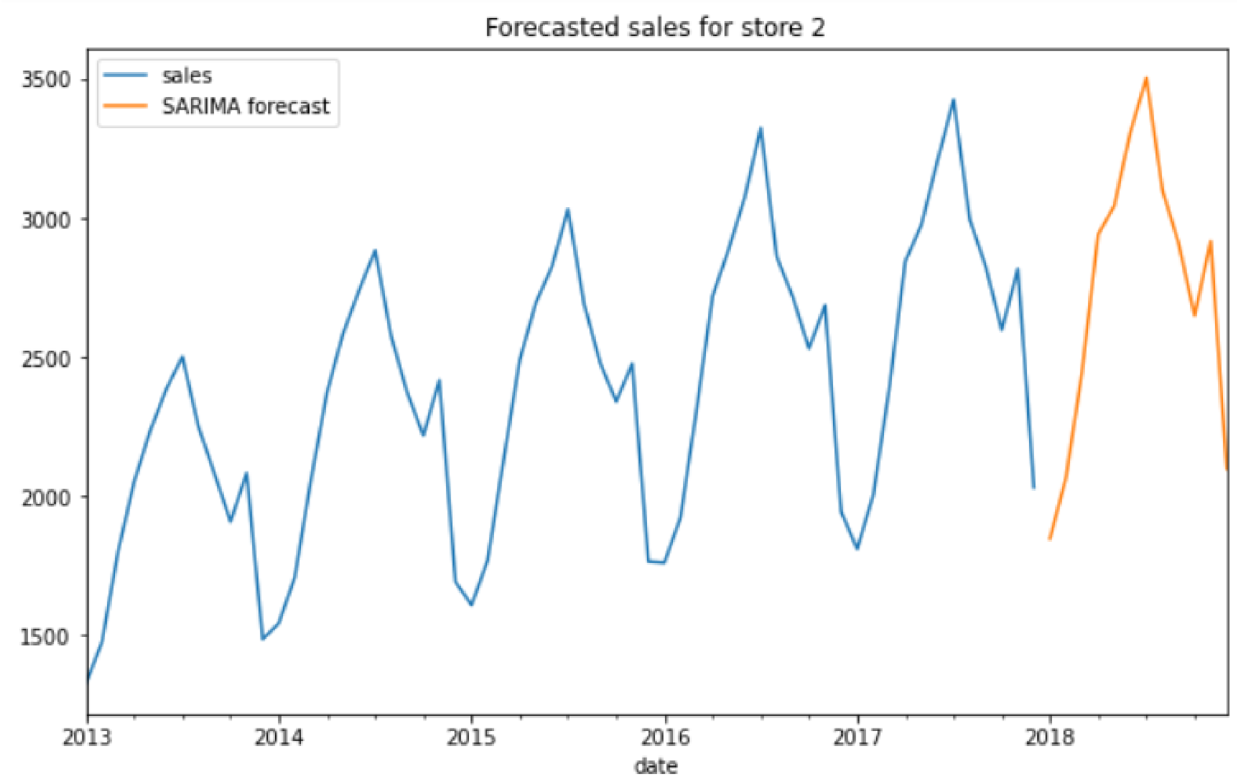
	Store	MSE	RMSE	Mean
0	1	4282.339702	65.439588	sales 2659.885305 dtype: float64
1	2	4282.339702	65.439588	2659.885305
2	3	7301.097704	85.446461	3350.165559
3	4	6599.090013	81.234783	3085.820897
4	5	3318.176811	57.603618	2236.334901
5	6	2549.214423	50.489746	2233.088671
6	7	2328.895877	48.258635	2045.124296
7	8	7196.819714	84.834072	3607.358916
8	9	6027.243358	77.635323	3097.281746
9	10	7242.352954	85.102015	3297.260183

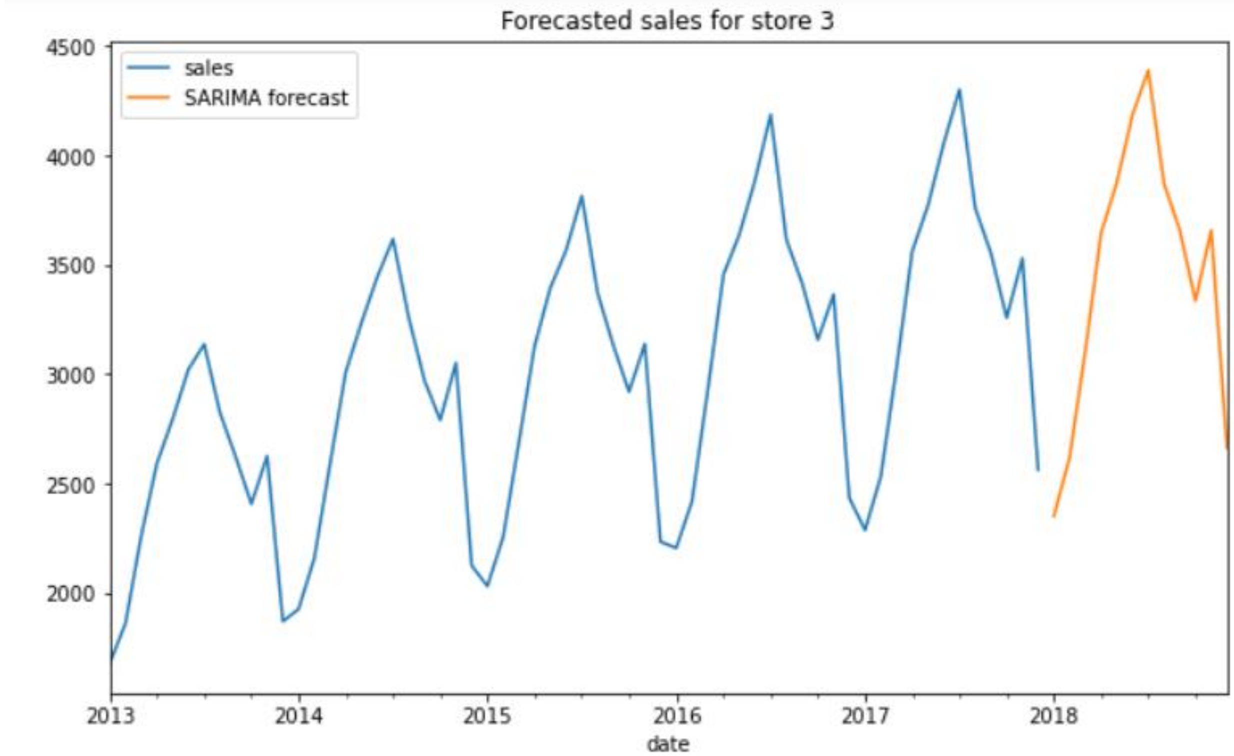
This is a tabular representation of the evaluation metrics for each store, including the store number, the mean squared error (MSE), the root mean squared error (RMSE), and the mean value of the test data. Each row corresponds to one store, and the columns represent the different evaluation metrics.

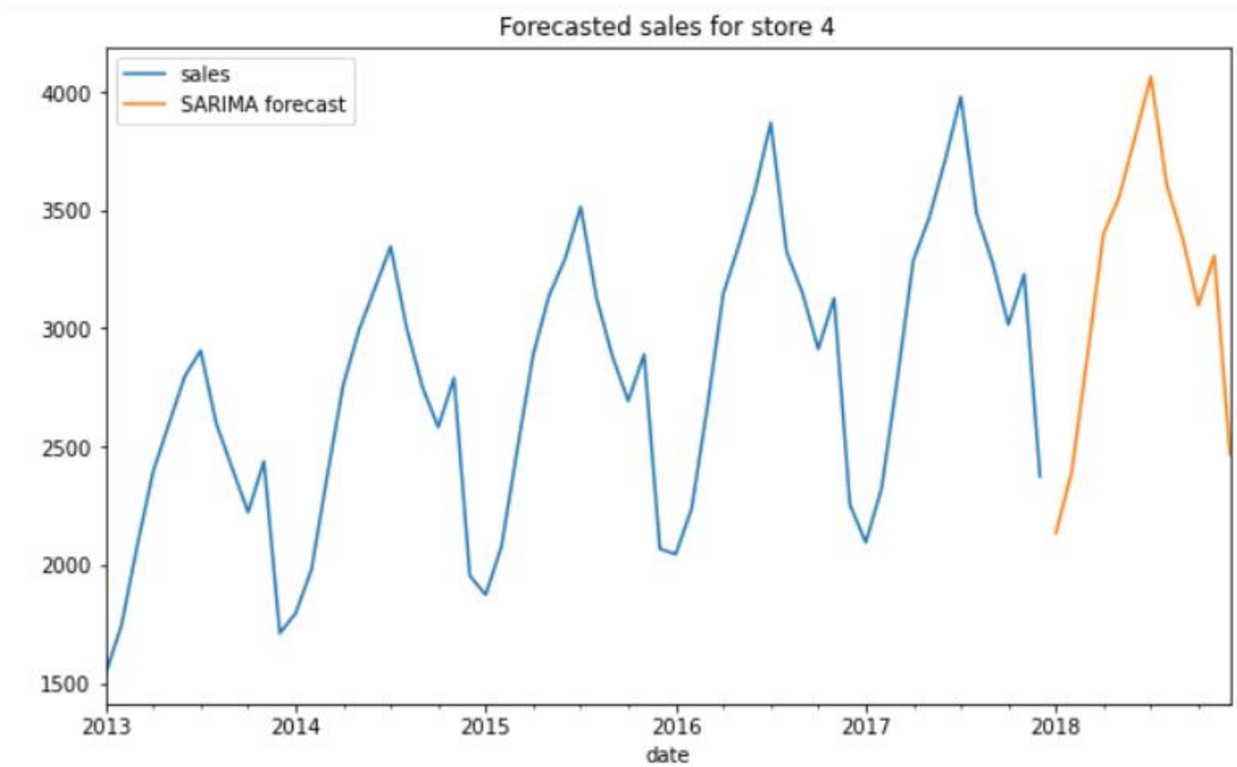
The purpose of this table is to compare the performance of the SARIMA model across different stores and identify any patterns or trends in the evaluation metrics.

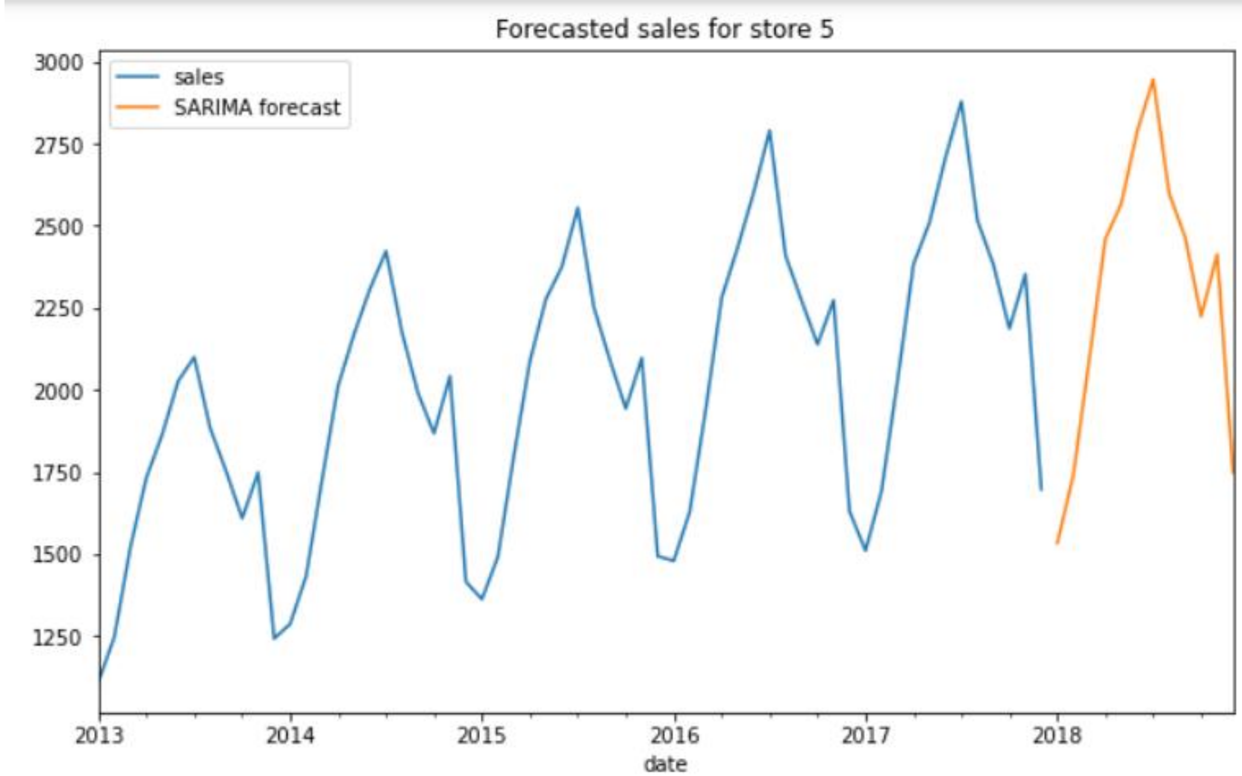
g) Plot of SARIMAX Forecasted Sales for 2018

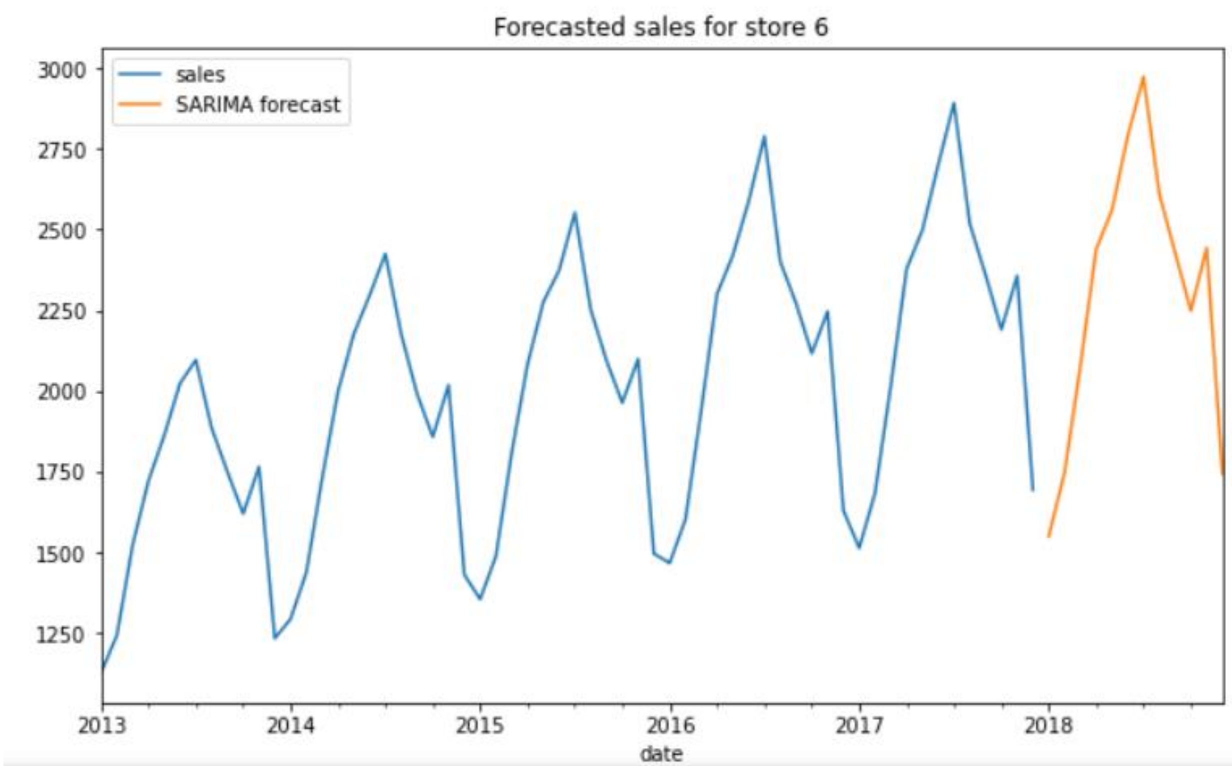


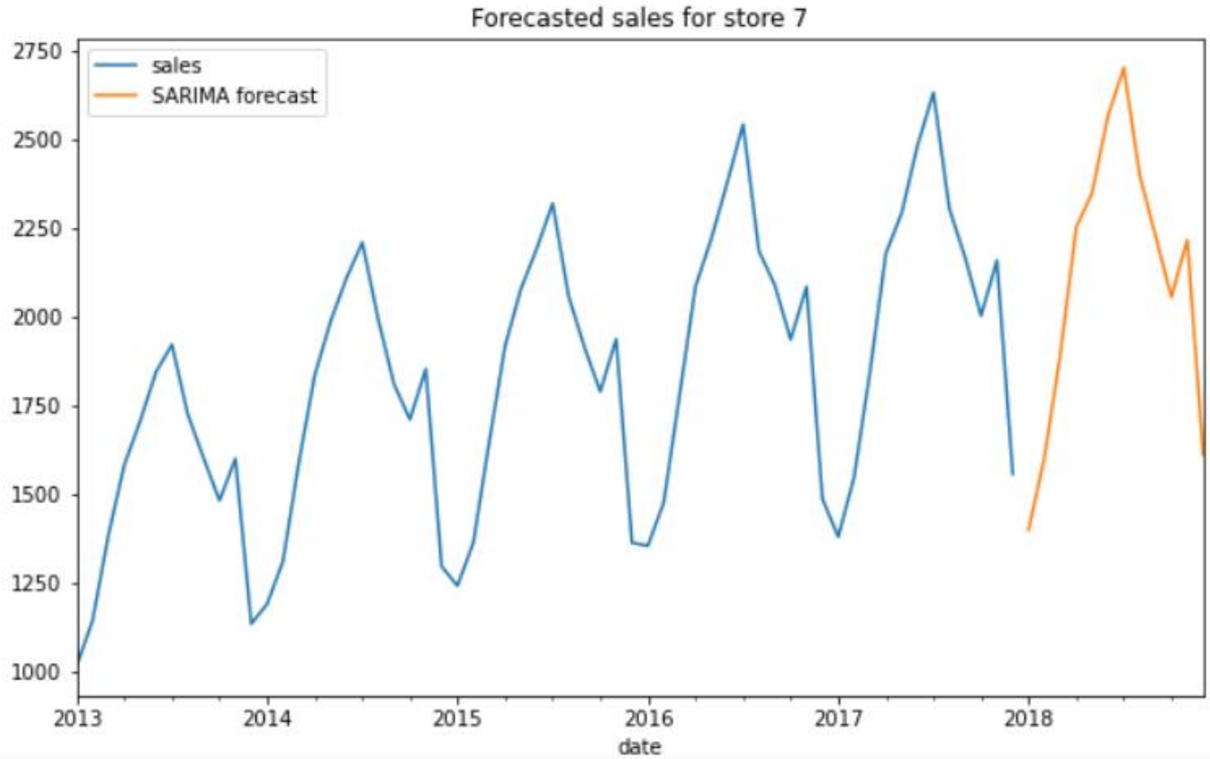


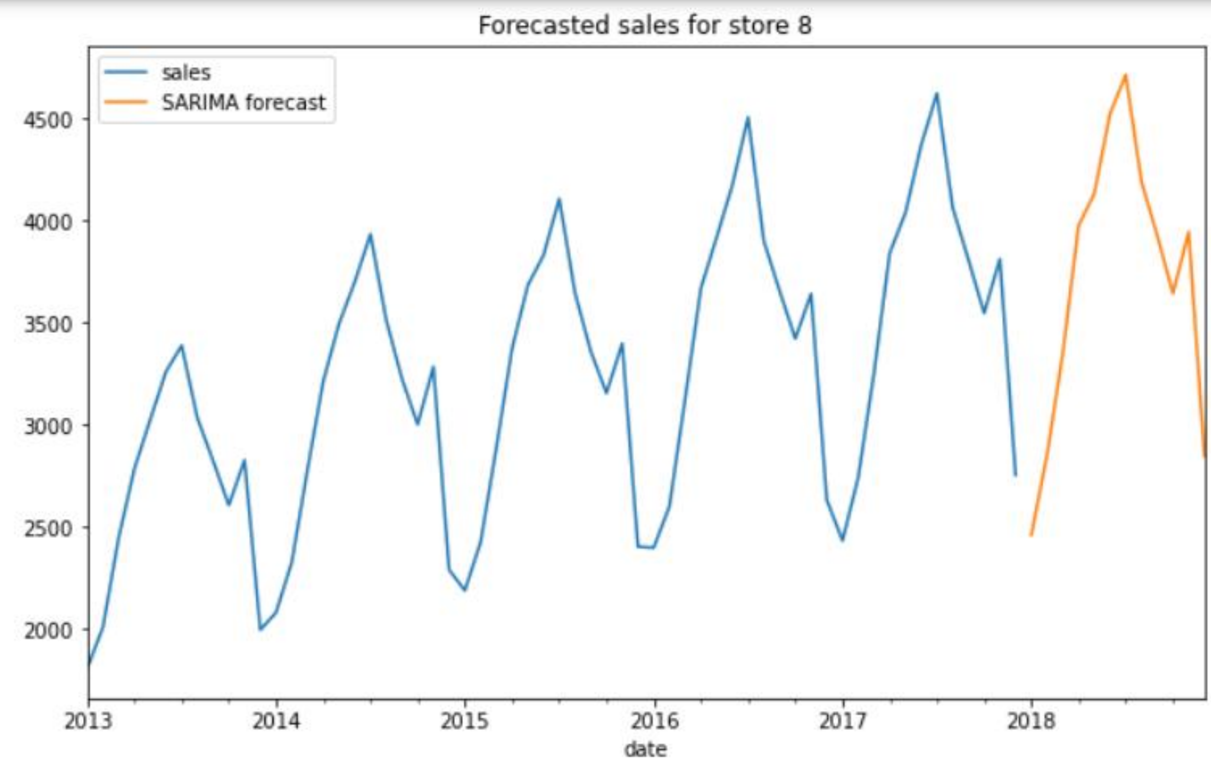

Forecasted sales for store 3
sales
SARIMA forecast
4500
4000
3500
3000
2500
2000
2013
2014
2015
2016
2017
2018
date

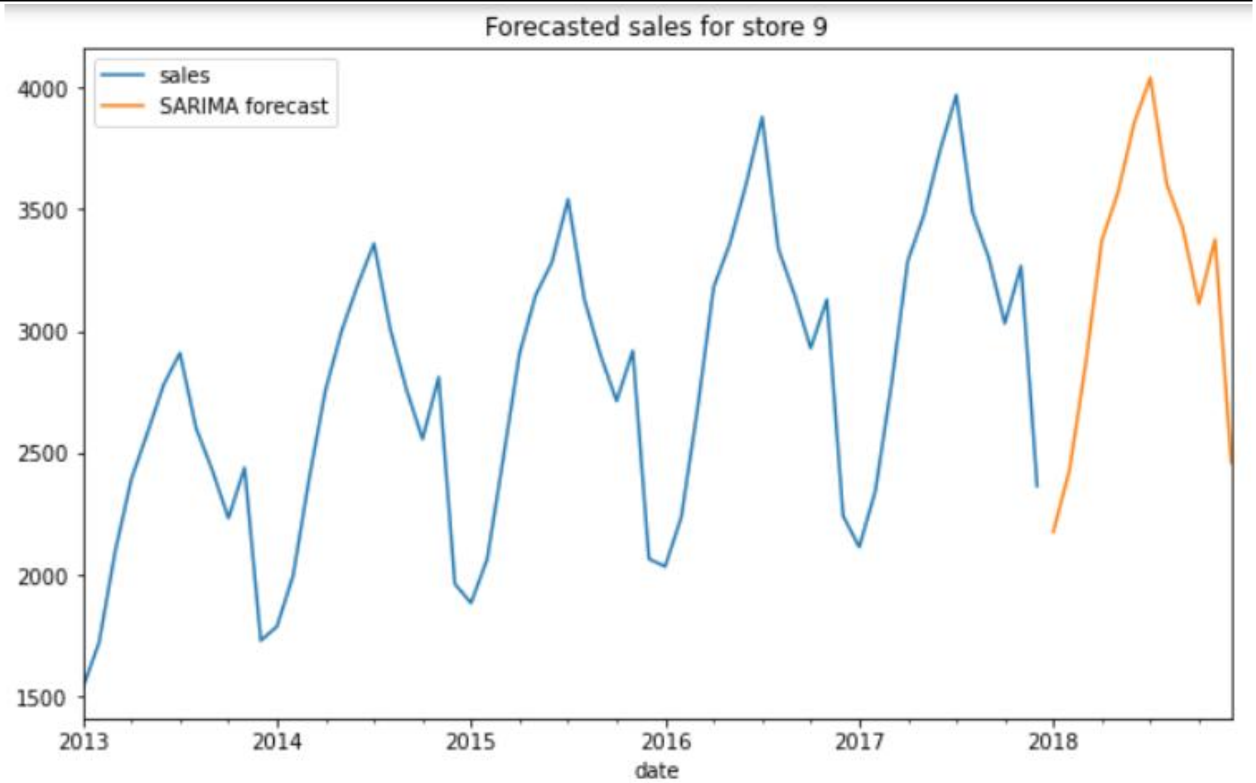

Forecasted sales for store 4
sales
SARIMA forecast
4000
3500
3000
2500
2000
1500
2013
2014
2015
2016
2017
2018
date

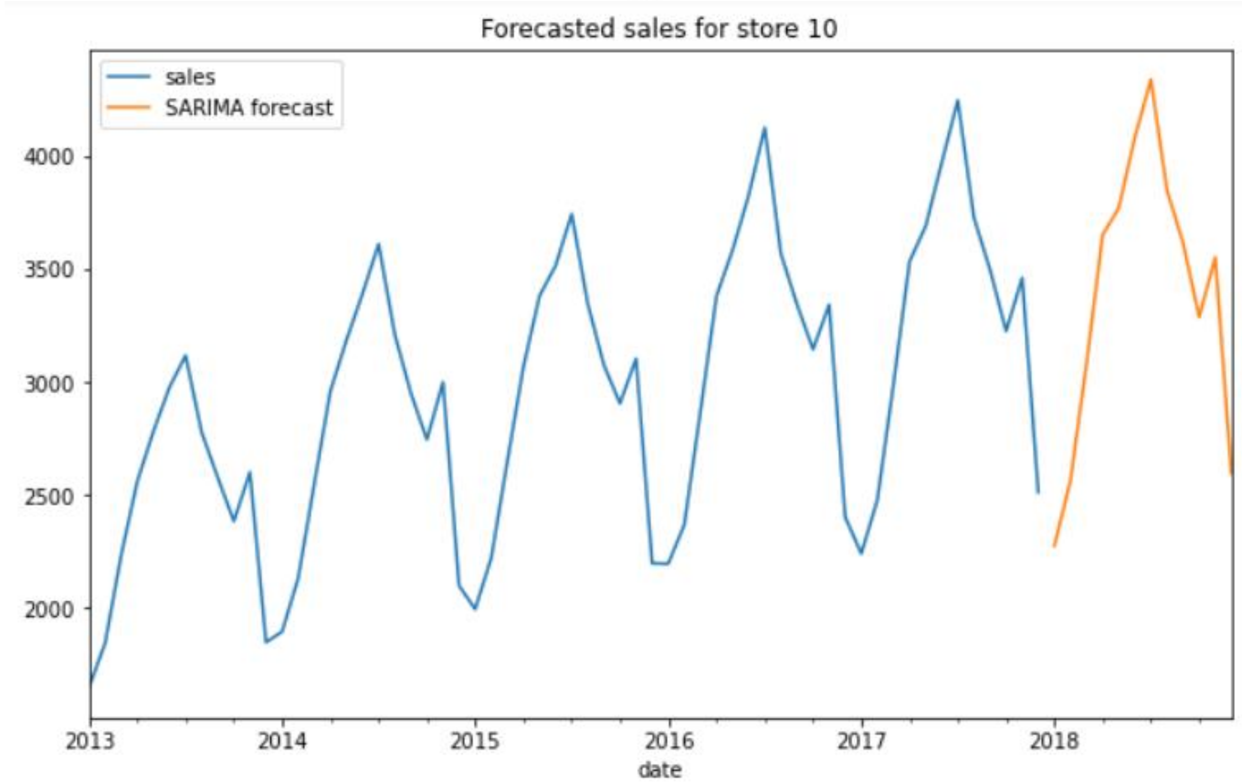

Forecasted sales for store 5
sales
SARIMA forecast
3000
2750
2500
2250
2000
1750
1500
1250
2013
2014
2015
2016
2017
2018
date


Forecasted sales for store 6
sales
SARIMA forecast
3000
2750
2500
2250
2000
1750
1500
1250
2013
2014
2015
2016
2017
2018
date


Forecasted sales for store 7
sales
SARIMA forecast
2750
2500
2250
2000
1750
1500
1250
1000
2013
2014
2015
2016
2017
2018
date


Forecasted sales for store 8
sales
SARIMA forecast
4500
4000
3500
3000
2500
2000
2013
2014
2015
2016
2017
2018
date


Forecasted sales for store 9
sales
SARIMA forecast
4000
3500
3000
2500
2000
1500
2013
2014
2015
2016
2017
2018
date


Forecasted sales for store 10
sales
SARIMA forecast
4000
3500
3000
2500
2000
2013
2014
2015
2016
2017
2018
date

h) Forecasted Sales Values for 2018

	Store 1	Store 2	Store 3	Store 4	Store 5	Store 6	Store 7	Store 8	Store 9	Store 10
2018-01-01	1846.933867	1846.933867	2349.623412	2133.651430	1533.389582	1549.745152	1398.740020	2460.539367	2176.140095	2273.577440
2018-02-01	2066.312281	2066.312281	2618.502518	2392.478598	1739.015716	1746.300146	1603.528952	2852.548030	2429.164446	2562.546405
2018-03-01	2449.334413	2449.334413	3107.266963	2884.172741	2092.192165	2075.107472	1893.861169	3350.131768	2864.335100	3080.840860
2018-04-01	2941.081107	2941.081107	3645.313948	3398.548742	2458.175966	2438.132472	2251.259583	3974.934274	3372.397539	3650.979028
2018-05-01	3043.552240	3043.552240	3875.728897	3555.059342	2566.353962	2559.229646	2348.961095	4131.186740	3569.066138	3769.012829
2018-06-01	3307.238828	3307.238828	4187.392672	3807.223689	2786.891999	2788.449842	2565.574069	4522.535535	3853.558461	4081.725995
2018-07-01	3503.420805	3503.420805	4390.154461	4064.423361	2944.948176	2972.557326	2700.737184	4713.524407	4039.393600	4338.812030
2018-08-01	3099.811353	3099.811353	3871.433395	3608.635475	2597.135572	2610.382497	2398.528193	4190.178377	3607.264631	3847.954228
2018-09-01	2909.890021	2909.890021	3658.772502	3381.655590	2464.233084	2428.439825	2227.210367	3927.810668	3422.854210	3615.237578
2018-10-01	2648.825139	2648.825139	3334.534291	3097.435832	2223.101530	2248.418503	2054.983438	3643.471107	3111.338678	3286.661388
2018-11-01	2916.745721	2916.745721	3658.577319	3307.967766	2412.531260	2443.095026	2216.284747	3944.488303	3376.093800	3551.675014
2018-12-01	2097.873927	2097.873927	2662.389496	2467.066138	1747.634622	1743.224750	1610.802161	2847.871280	2459.148901	2595.382466

FBPROPHET MODEL BUILDING

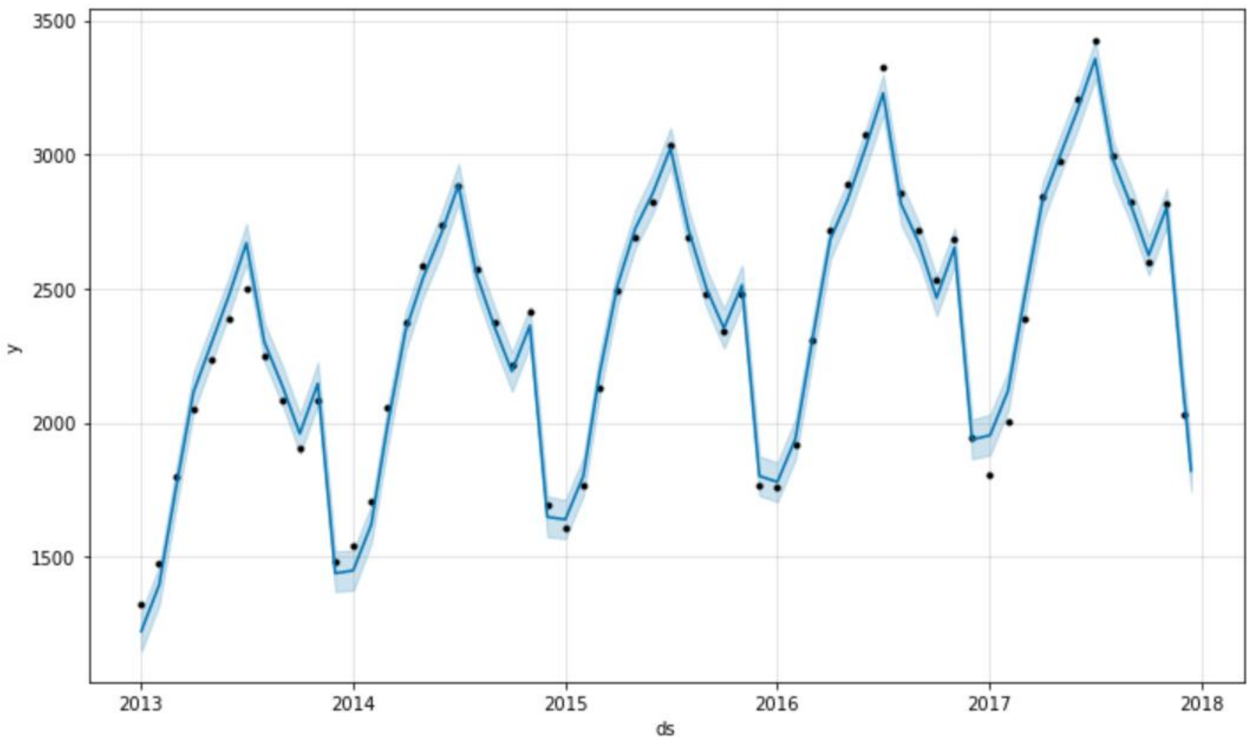
Fbprophet is a Python library developed by Facebook for time-series forecasting. It is designed to handle time-series data with multiple seasonalities, as well as changes in trend and holiday effects.

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.

a) Prediction using Fbprophet

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	yearly	yearly_
0	2013-01-01	1874.738336	1148.337562	1295.135050	1874.738336	1874.738336	-652.274913	-652.274913	-652.274913	-652.274913	-652.2
1	2013-02-01	1894.713796	1322.851772	1466.323529	1894.713796	1894.713796	-498.620843	-498.620843	-498.620843	-498.620843	-498.6
2	2013-03-01	1912.756148	1673.561892	1819.354198	1912.756148	1912.756148	-167.560546	-167.560546	-167.560546	-167.560546	-167.5
3	2013-04-01	1932.731609	2042.251507	2191.035542	1932.731609	1932.731609	181.634528	181.634528	181.634528	181.634528	181.6
4	2013-05-01	1952.062700	2220.526321	2365.593504	1952.062700	1952.062700	338.710993	338.710993	338.710993	338.710993	338.7

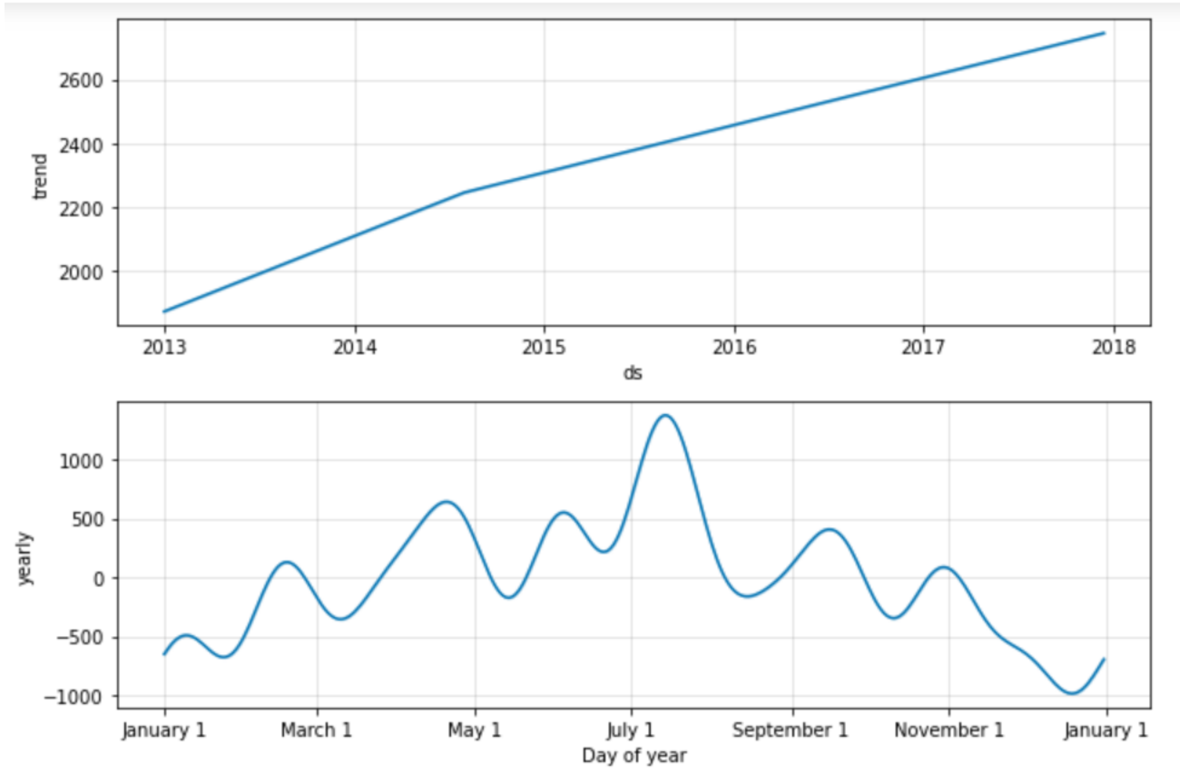


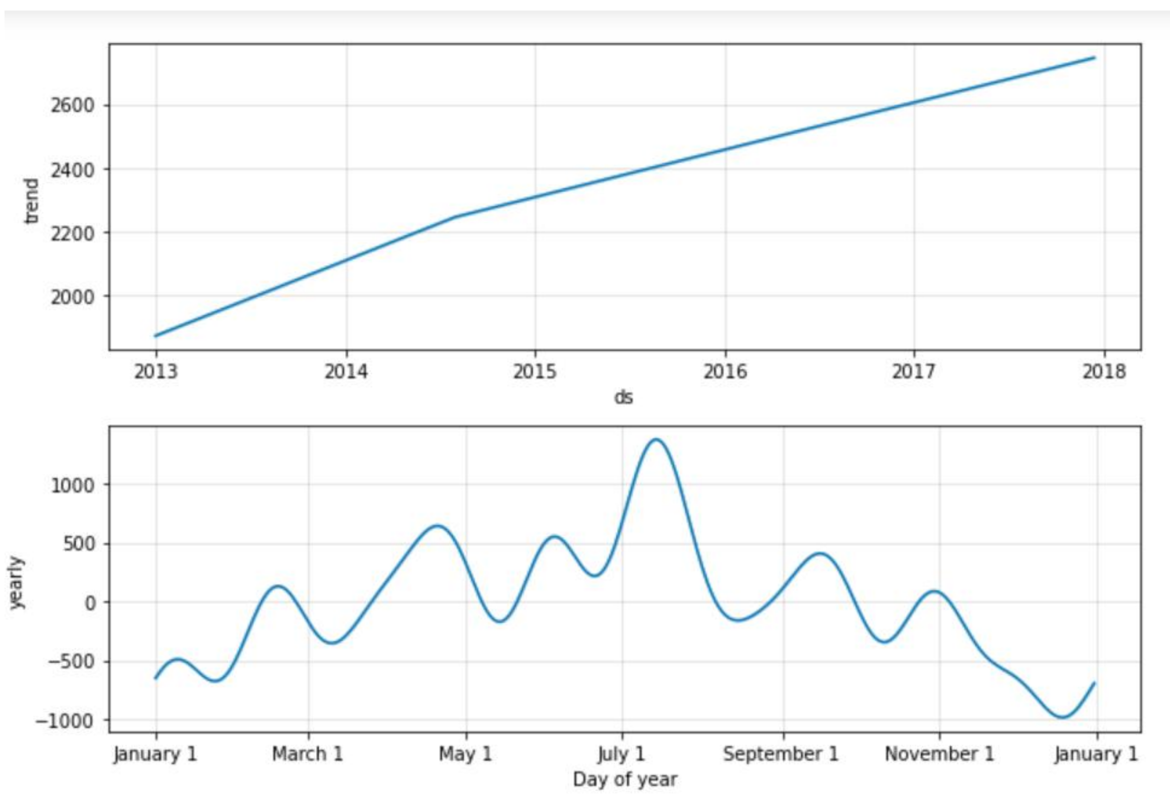
The m_store1.plot(prediction) method in Fbprophet is used to visualize the historical data and the predicted future values. The resulting plot shows the historical data as a black line, with the predicted future values overlaid in blue.

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. .

b) Visualize Each Components





c) 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.

	ds	yhat	yhat_lower	yhat_upper	y	cutoff
0	2015-07-01	2955.332349	2948.806381	2961.762030	3031.258065	2015-06-10
1	2015-08-01	2558.441889	2536.175250	2579.842052	2691.677419	2015-06-10
2	2015-09-01	2335.642708	2292.164308	2378.545047	2479.633333	2015-06-10
3	2015-10-01	2275.174202	2210.988984	2343.125577	2339.354839	2015-06-10
4	2015-11-01	2566.361675	2475.127878	2662.276724	2476.400000	2015-06-10

d) 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.

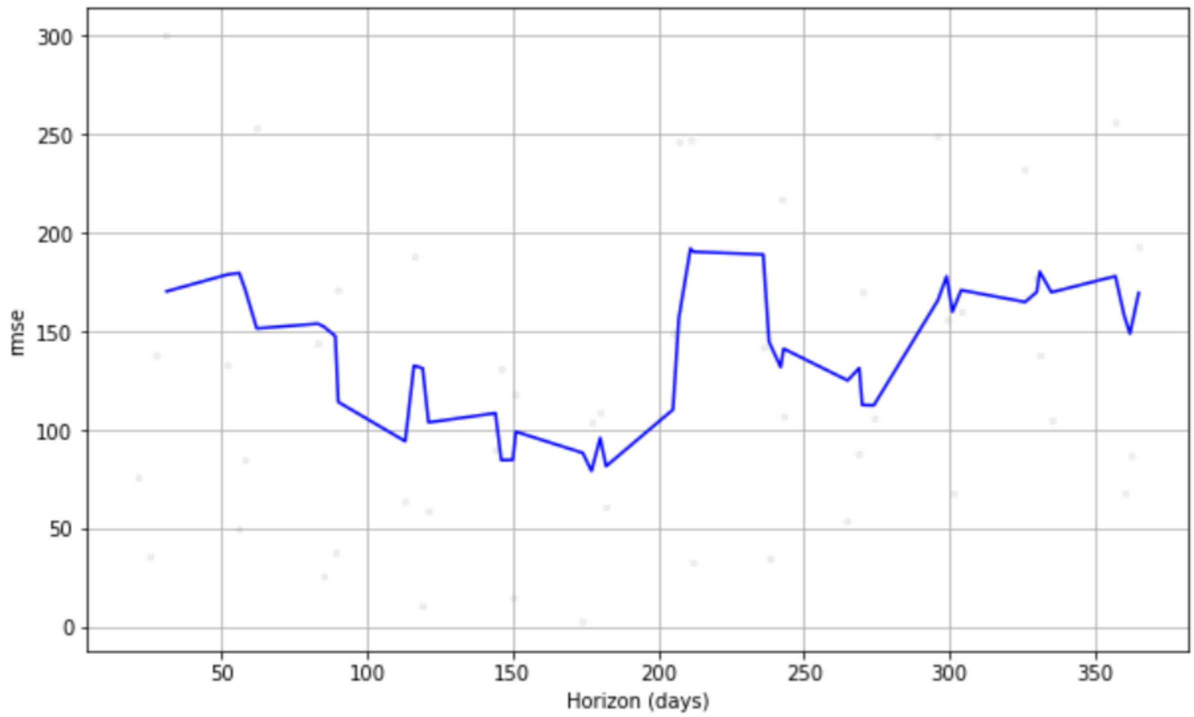
	horizon	mse	rmse	mae	mape	mdape	smape	coverage
0	31 days	29015.695985	170.339942	137.407790	0.063178	0.033264	0.060356	0.00
1	52 days	32012.444095	178.920217	151.735244	0.069291	0.045489	0.066703	0.00
2	56 days	32300.546661	179.723528	155.116964	0.070623	0.045489	0.068067	0.00
3	58 days	29370.778214	171.379048	141.991200	0.067717	0.039677	0.064831	0.00
4	62 days	22952.121551	151.499576	130.398077	0.057911	0.039677	0.056315	0.00
5	83 days	23697.519905	153.939988	133.086850	0.060053	0.043962	0.058577	0.00
6	85 days	23252.888533	152.488978	127.159275	0.056409	0.043962	0.054865	0.25
7	89 days	21790.482636	147.615997	115.292810	0.052434	0.036012	0.050975	0.25
8	90 days	13034.358099	114.168113	94.682799	0.038705	0.036012	0.038510	0.25
9	113 days	8880.821609	94.238111	74.730302	0.031047	0.020695	0.030513	0.50

e) 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.



f) 2018 Forecasted Sales Value

```
2018 Forecasted Sales Value of Store 1
:
            ds         yhat
0   2018-01-01  2091.547855
1   2018-02-01  2255.423505
2   2018-03-01  2617.357195
3   2018-04-01  2962.877799
4   2018-05-01  3152.142099
5   2018-06-01  3308.128818
6   2018-07-01  3484.785702
7   2018-08-01  3144.630513
8   2018-09-01  2948.974749
9   2018-10-01  2782.495023
10  2018-11-01  2952.904169
11  2018-12-01  2239.529640
```

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.

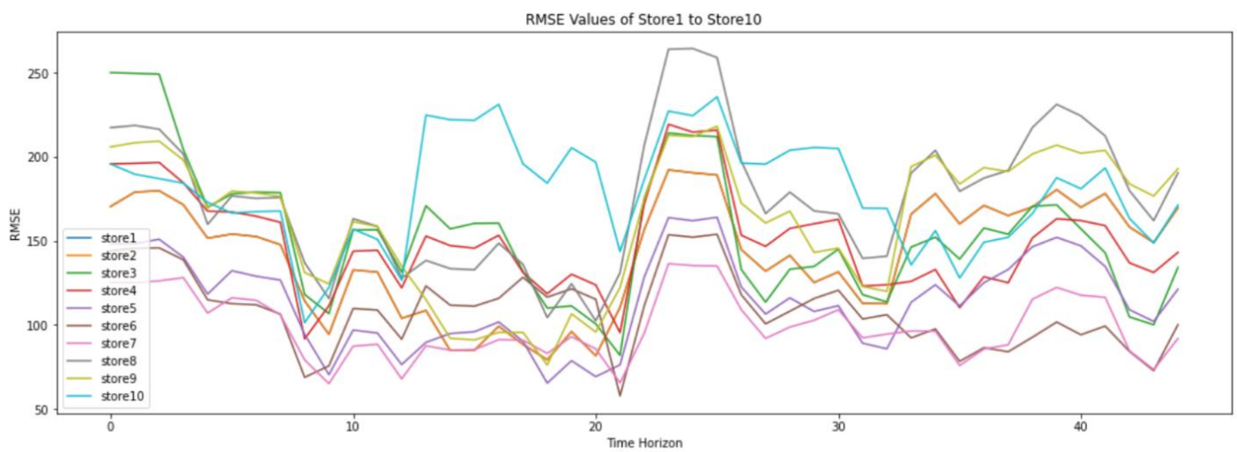
Comparison of RMSE of SARIMAX and Prophet Model

RMSE of SARIMAX model

	Store	MSE	RMSE	Mean
0	1	4282.339702	65.439588	sales 2659.885305 dtype: float64
1	2	4282.339702	65.439588	2659.885305
2	3	7301.097704	85.446461	3350.165559
3	4	6599.090013	81.234783	3085.820897
4	5	3318.176811	57.603618	2236.334901
5	6	2549.214423	50.489746	2233.088671
6	7	2328.895877	48.258635	2045.124296
7	8	7196.819714	84.834072	3607.358916
8	9	6027.243358	77.635323	3097.281746
9	10	7242.352954	85.102015	3297.260183

RMSE of Prophet model

store	mse	rmse
store1	29015.695984762475	170.3399424326864
store2	29015.695984762475	170.3399424326864
store3	62507.287534186326	250.01457464353217
store4	38292.658413992045	195.68510013282065
store5	21922.34978243222	148.0619795303042
store6	20729.31864761746	143.97679899073134
store7	15675.473091924116	125.20172958838914
store8	47215.99749820805	217.29242393191726
store9	42346.9041647091	205.7836343461479
store10	38262.305598492974	195.60752950357758



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

	Store 1	Store 2	Store 3	Store 4	Store 5	Store 6	Store 7	Store 8	Store 9	Store 10
2018-01-01	1846.933867	1846.933867	2349.623412	2133.651430	1533.389582	1549.745152	1398.740020	2460.539367	2176.140095	2273.577440
2018-02-01	2066.312281	2066.312281	2618.502518	2392.478598	1739.015716	1746.300146	1603.528952	2852.548030	2429.164446	2562.546405
2018-03-01	2449.334413	2449.334413	3107.266963	2884.172741	2092.192165	2075.107472	1893.861169	3350.131768	2864.335100	3080.840860
2018-04-01	2941.081107	2941.081107	3645.313948	3398.548742	2458.175966	2438.132472	2251.259583	3974.934274	3372.397539	3650.979028
2018-05-01	3043.552240	3043.552240	3875.728897	3555.059342	2566.353962	2559.229646	2348.961095	4131.186740	3569.066138	3769.012829
2018-06-01	3307.238828	3307.238828	4187.392672	3807.223689	2786.891999	2788.449842	2565.574069	4522.535535	3853.558461	4081.725995
2018-07-01	3503.420805	3503.420805	4390.154461	4064.423361	2944.948176	2972.557326	2700.737184	4713.524407	4039.393600	4338.812030
2018-08-01	3099.811353	3099.811353	3871.433395	3608.635475	2597.135572	2610.382497	2398.528193	4190.178377	3607.264631	3847.954228
2018-09-01	2909.890021	2909.890021	3658.772502	3381.655590	2464.233084	2428.439825	2227.210367	3927.810668	3422.854210	3615.237578
2018-10-01	2648.825139	2648.825139	3334.534291	3097.435832	2223.101530	2248.418503	2054.983438	3643.471107	3111.338678	3286.661388
2018-11-01	2916.745721	2916.745721	3658.577319	3307.967766	2412.531260	2443.095026	2216.284747	3944.488303	3376.093800	3551.675014
2018-12-01	2097.873927	2097.873927	2662.389496	2467.066138	1747.634622	1743.224750	1610.802161	2847.871280	2459.148901	2595.382466

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 by DataCamp
11. Time Series Forecasting - ARIMA, LSTM, Prophet with Python by Krish Naik
12. Time Series Forecasting for Beginners by Machine Learning with Phil
13. Time Series Forecasting | Time Series Analysis | Time Series Prediction | Python | Data Science by Edureka
14. Time Series Analysis and Forecasting with Python - Part 1 by StatQuest with Josh Starmer
15. Introduction to Time Series Forecasting by AIEngineering