

**Capstone Project**

Future Sales Prediction



**Submitted by**

**SUBASREE.M.R**

**Table of Contents**

1. Problem Statement

2. Project Objective

3. Data Description

4. Data Pre-processing Steps and Inspiration

5. Choosing the Algorithm for the Project

6. Motivation and Reasons For Choosing the Algorithm

7. Model Building

8. Model Evaluation and Techniques

9. Inferences from the Same

10. Future Possibilities of the Project

11. Conclusion

12. References

**Chapter 1 - Problem Statement**

* A retail store that has multiple outlets across the country are facing issues in managing the inventory - to match the demand with respect to supply. We need to come up with useful insights using the data and make prediction models to forecast the sales for the next 12 weeks.
* We have the sales data for 45 different wallmart stores and we are trying to predict the weekly sales of different stores based on variables such as store number, date, temperature, fuel price, unemployment rate etc.

**Chapter 2- Project Objective**

* Understand the dataset and features
* Use suitable Data Preprocessing and Feature Selection/Engineering Methods
* Fine tune the model and hyper parameters and Finalise the Model
* Make the model deployment-ready by giving User-Input provision

**Chapter 3- Data Description**

The Dataset used in this project is WALLMART dataset**.**

This dataset contains data for the weekly sales prediction for one of the leading retail store, Walmart. There are sales data available for 45 stores of Walmart. The columns in the dataset includes the following

* **Store** - the store number
* **Date** - the week of sales
* **Weekly\_Sales** - sales for the given store
* **Holiday\_Flag** - whether the week is a special holiday week 1 – Holiday week 0 – Non-holiday week
* **Temperature** - Temperature on the day of sale
* **Fuel\_Price** - Cost of fuel in the region
* **CPI** – Prevailing consumer price index
* **Unemployment** - Prevailing unemployment rate

Here our dependent variable or the column which we are trying to predict is **Weekly sales** and rest are independent variables.

**Introduction**

****

#### Predicting future sales for a company is one of the most important aspects of strategic planning.[¶](https://www.kaggle.com/code/anushkaml/walmart-time-series-sales-forecasting#Predicting-future-sales-for-a-company-is-one-of-the-most-important-aspects-of-strategic-planning.)

In this kernel, we will wanted analyse in depth how internal and external factors of one of the biggest companies in the US can affect their Weekly Sales in the future.

This module contains complete analysis of data , includes time series analysis , identifies the best performing stores , performs sales prediction with the help of multiple linear regression.

The data collected ranges from 2010 to 2012, where 45 Walmart stores across the country were included in this analysis. It is important to note that we also have external data available like CPI, Unemployment Rate and Fuel Prices in the region of each store which, hopefully, help us to make a more detailed analysis.

#### In Retail Industry and chain of stores one of the biggest issue they face are supply chain management. The component of supply chain management (SCM) involved with determining how best to fulfil the requirements created from the Demand Plan.

It's objective is to balance supply and demand in a manner that achieves the financial and service objectives of the enterprise.

If we look into the case of a retail chain stores one of the basic case is to know the demand of products that are sold in the store. If the decision making authority know what’s the demand of each products for a week or month, they would be able to plan the supply chain accordingly. If that is possible this would save a lot of money for them because they don't have to overstock or can plan their Logistics accordingly.

#### **Sales:**

-Date: The date of the week where this observation was taken.

-Weekly\_Sales: The sales recorded during that Week.

-Store: The store which observation in recorded 1–45

-Holiday\_Flag: Boolean value representing a holiday week or not.

#### **Features:**

-Temperature: Temperature of the region during that week.

-Fuel\_Price: Fuel Price in that region during that week.

-CPI: Consumer Price Index during that week.

-Unemployment: The unemployment rate during that week in the region of the store.

**TYPES OF ALGORITHIM USED**

**Time Series algorithms:**

Time series forecasting is used when you want to make scientific predictions based on historical time-stamped data. [It involves building models through historical analysis and using them to make observations and drive future strategic decision-making](https://www.tableau.com/learn/articles/time-series-forecasting). [Time series forecasting has a range of applications in various industries such as weather forecasting, economic forecasting, healthcare forecasting, engineering forecasting, finance forecasting, retail forecasting, business forecasting, environmental studies forecasting, social studies forecasting and more](https://www.tableau.com/learn/articles/time-series-forecasting). Since we are going to predict the future sales which is based on the time stamp this project uses Time series algorithms.

**What is Time Series**

Any data recorded with some fixed interval of time is called as time series data. This fixed interval can be hourly, daily, monthly or yearly. e.g. hourly temp reading, daily changing fuel prices, monthly electricity bill, annul company profit report etc. In time series data, time will always be independent variable and there can be one or many dependent variable.

Sales forecasting time series with shampoo sales for every month will look like this,



Reference Pic only

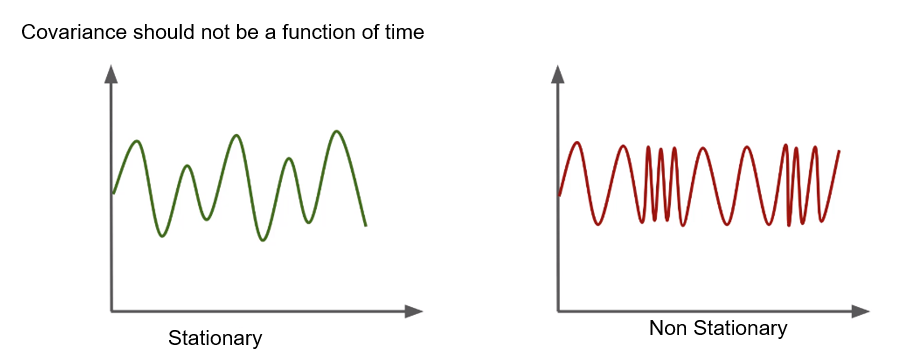
In above example since there is only one variable dependent on time so its called as univariate time series. If there are multiple dependent variables, then its called as multivariate time series.

Objective of time series analysis is to understand how change in time affect the dependent variables and accordingly predict values for future time intervals.

Stationarity Data

For accurate analysis and forecasting trend and seasonality is removed from the time series and converted it into stationary series. Stationary data refers to the **time series data that mean and variance do not vary across time.** The data is considered non-stationary if there is a **strong trend or seasonality** observed from the data.

The graph given below shows the difference between stationary and non stationary



Test for Stationarity

Easy way is to look at the plot and look for any obvious trend or seasonality.

There are various statistical tests to check for stationarity in time series data. [Two commonly used tests are the Augmented Dickey-Fuller (ADF) test and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test](https://www.analyticsvidhya.com/blog/2021/06/statistical-tests-to-check-stationarity-in-time-series-part-1/). The ADF test checks for the presence of a unit root in the time series, which would indicate that it is non-stationary. [The KPSS test, on the other hand, checks for the presence of a trend in the time series](https://help.xlstat.com/6697-unit-root-dickey-fuller-and-stationarity-tests-time). These tests can help determine whether a time series is stationary or not.

**Augmented Dickey-Fuller test(ADF)**

Augmented Dickey-Fuller Test or ADF test is one of the most popular statistical methods to determine whether a time series data is stationary. In Python, we can directly use the [adfuller package](https://www.statsmodels.org/stable/generated/statsmodels.tsa.stattools.adfuller.html" \t "_blank) from statsmodels to see whether the data is stationary at different significance levels. Let us use the ADF test on our step count data to see whether it is stationary or not.

ADF test will return 'p-value' and 'Test Statistics' output values.

* **p-value > 0.05**: non-stationary.
* **p-value <= 0.05**: stationary.
* **Test statistics**: More negative this value more likely we have stationary series. Also, this value should be smaller than critical values(1%, 5%, 10%). For e.g. If test statistic is smaller than the 5% critical values, then we can say with 95% confidence that this is a stationary series

**Components of Time series**

The reasons or forces that change the attributes of a time series are known as the Components of Time Series.

The following are the components of time series −

* **Trend**
* **Seasonal Variations**
* **Cyclical Variations**
* **Random or Irregular Movements**

#### **Trend**

Trend shows a common tendency of data. It may move upward or increase or go downward or decrease over a certain, long period of time. The trend is a stable and long-term general tendency of movement of the data.

#### **Seasonal Variations**

Seasonal variations are changes in time series that occur in the short term, usually within less than 12 months. They usually show the same pattern of upward or downward growth in the 12-month period of the time series. These variations are often recorded as hourly, daily, weekly, quarterly, and monthly schedules.

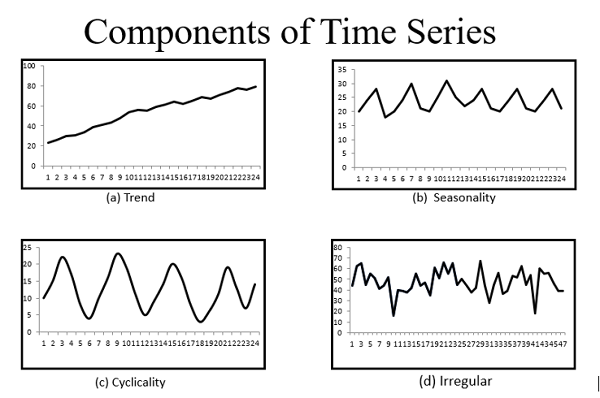
#### **Cyclical Variations**

Variations in time series that occur themselves for the span of more than a year are called Cyclical Variations. Such oscillatory movements of time serious often have a duration of more than a year. One complete period of operation is called either a cycle or a ‘Business Cycle’.

#### **Random or Irregular Movements**

There is another kind of movement that can be seen in the case of time series. It is pure Irregular and Random Movement. As the name suggests, no hypothesis or trend can be used to suggest irregular or random movements in a time series. These outcomes are unforeseen, erratic, unpredictable, and uncontrollable in nature.

Earthquakes, war, famine, and floods are some examples of random time series components.



Reference pic only

**Time Series Analysis**

Time series analysis is a specific way of analyzing a sequence of data points collected over an interval of time. As name suggest its analysis of the time series data to identify the patterns in it.

#### **Auto-Correlation Function (ACF)**

ACF is used to indicate how similar a value is within a given time series and the previous value. (OR) It measures the degree of the similarity between a given time series and the lagged version of that time series at the various intervals we observed.

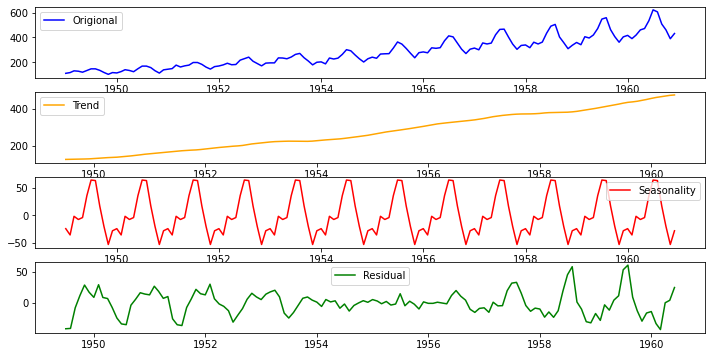
Python Statsmodels library calculates autocorrelation. This is used to identify a set of trends in the given dataset and the influence of former observed values on the currently observed values.

#### **Partial Auto-Correlation (PACF)**

PACF is similar to Auto-Correlation Function and is a little challenging to understand. It always shows the correlation of the sequence with itself with some number of time units per sequence order in which only the direct effect has been shown, and all other intermediary effects are removed from the given time series.

**Time series Decomposition**

  Decomposition can be useful for better understanding the underlying patterns in a time series and for improving forecasting accuracy. Time series decomposition helps to deconstruct the time series into several component like trend and seasonality for better visualization of its characteristics. Using time-series decomposition makes it easier to quickly identify a changing mean or variation in the data.



Reference pic only

**ARIMA MODEL**

* **ARIMA** stands for **Autoregressive Integrated Moving Average Model**. It belongs to a class of models that explains a given time series based on its own past values -i.e.- its own lags and the lagged forecast errors. The equation can be used to forecast future values. Any ‘non-seasonal’ time series that exhibits patterns and is not a random white noise can be modeled with ARIMA models.
* So, **ARIMA**, short for **AutoRegressive Integrated Moving Average**, is a forecasting algorithm based on the idea that the information in the past values of the time series can alone be used to predict the future values.
* **ARIMA Models** are specified by three order parameters: (p, d, q),

where,

* + p is the order of the AR term
  + q is the order of the MA term
  + d is the number of differencing required to make the time series stationary
* **AR(p) Autoregression** – a regression model that utilizes the dependent relationship between a current observation and observations over a previous period. An auto regressive (AR(p)) component refers to the use of past values in the regression equation for the time series.
* **I(d) Integration** – uses differencing of observations (subtracting an observation from observation at the previous time step) in order to make the time series stationary. Differencing involves the subtraction of the current values of a series with its previous values d number of times.
* **MA(q) Moving Average** – a model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations. A moving average component depicts the error of the model as a combination of previous error terms. The order q represents the number of terms to be included in the model.

**Types of ARIMA Model**

* **ARIMA** : Non-seasonal Autoregressive Integrated Moving Averages
* **SARIMA**: Seasonal ARIMA
* **SARIMAX**: Seasonal ARIMA with exogenous variables

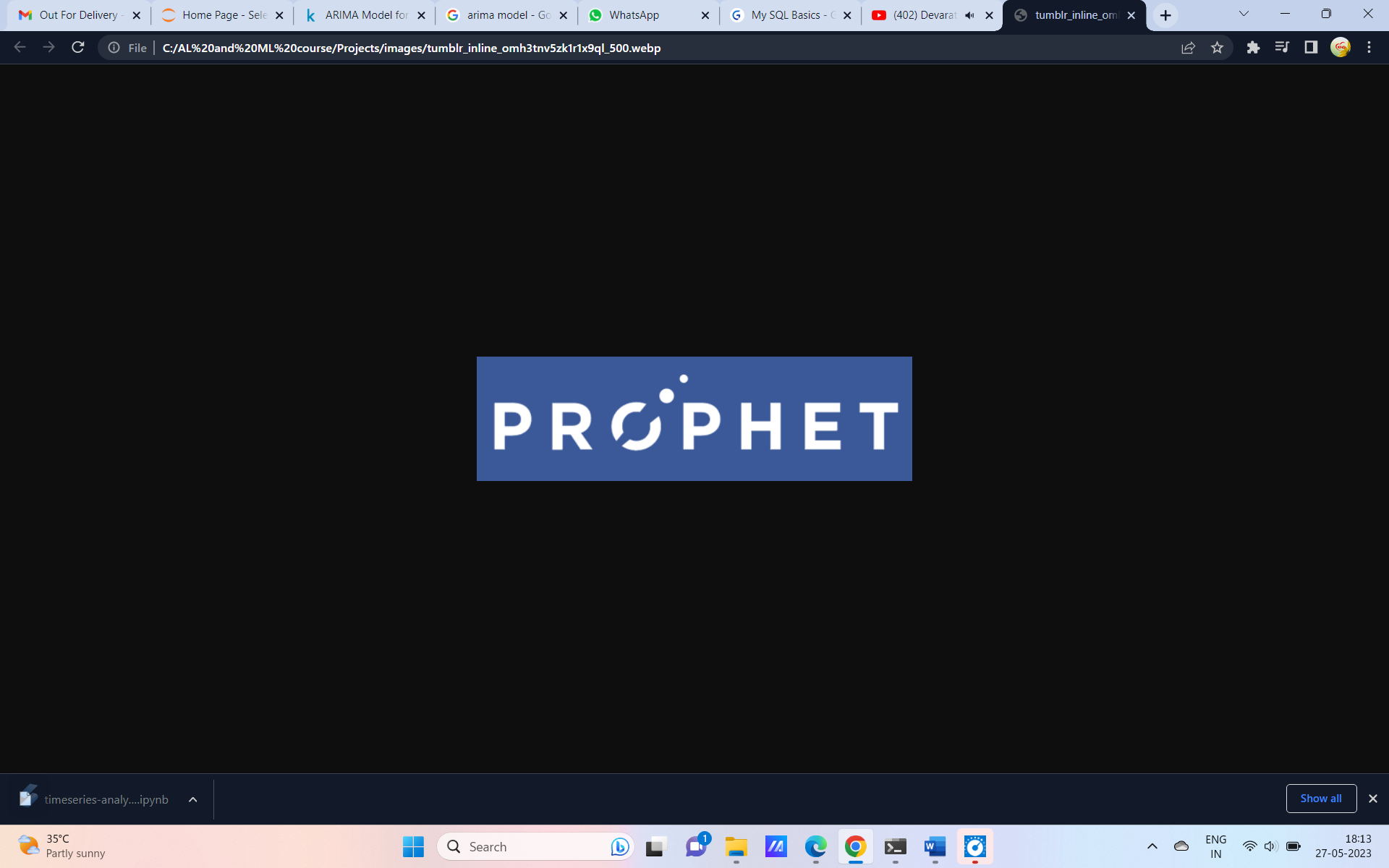
If a time series, has seasonal patterns, then we need to add seasonal terms and it becomes SARIMA, short for **Seasonal ARIMA**.

**SARIMAX:**

SARIMAX stands for **Seasonal AutoRegressive Integrated Moving Average with exogenous variables** model. [It is a time series forecasting model that can handle seasonality and external variables](https://365datascience.com/tutorials/python-tutorials/sarimax/). [The SARIMAX model is an extension of the SARIMA model, which allows for the addition of external variables](https://link.springer.com/chapter/10.1007/978-1-4842-7150-6_8). [The full name of the model would be Seasonal Autoregressive Integrated Moving Average Exogenous model](https://365datascience.com/tutorials/python-tutorials/sarimax/).

The SARIMAX model requires 4 additional orders compared to the ARIMAX model. The first 3 of these 4 orders are just seasonal versions of the ARIMA orders. [In other words, we have a seasonal autoregressive order denoted by upper-case P, an order of seasonal integration denoted by upper-case D, and a seasonal moving average order signified by upper-case Q](https://365datascience.com/tutorials/python-tutorials/sarimax/).

**PROPHET MODEL:**



Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

Prophet is open source software released by Facebook’s Core Data Science team. It is available for download on CRAN and PyPI.

* So, Prophet is the facebooks’ open source tool for making time series predictions.
* Prophet decomposes time series data into trend, seasonality and holiday effect.
* **Trend** models non periodic changes in the time series data.
* **Seasonality** is caused due to the periodic changes like daily, weekly, or yearly seasonality.
* **Holiday effect** which occur on irregular schedules over a day or a period of days.
* **Error terms** is what is not explained by the model.

**Advantages of Prophet**

Prophet has several advantages associated with it. These are given below:-

* **1. Accurate and fast** - Prophet is accurate and fast. It is used in many applications across Facebook for producing reliable forecasts for planning and goal setting.
* **2. Fully automatic** - Prophet is fully automatic. We will get a reasonable forecast on messy data with no manual effort.
* **3. Tunable forecasts** - Prophet produces adjustable forecasts. It includes many possibilities for users to tweak and adjust forecasts. We can use human-interpretable parameters to improve the forecast by adding our domain knowledge.
* **4. Handles seasonal variations well** - Prophet accommodates seasonality with multiple periods.
* **5. Robust to outliers** - It is robust to outliers. It handles outliers by removing them.
* **6. Robust to missing data** - Prophet is resilient to missing data.

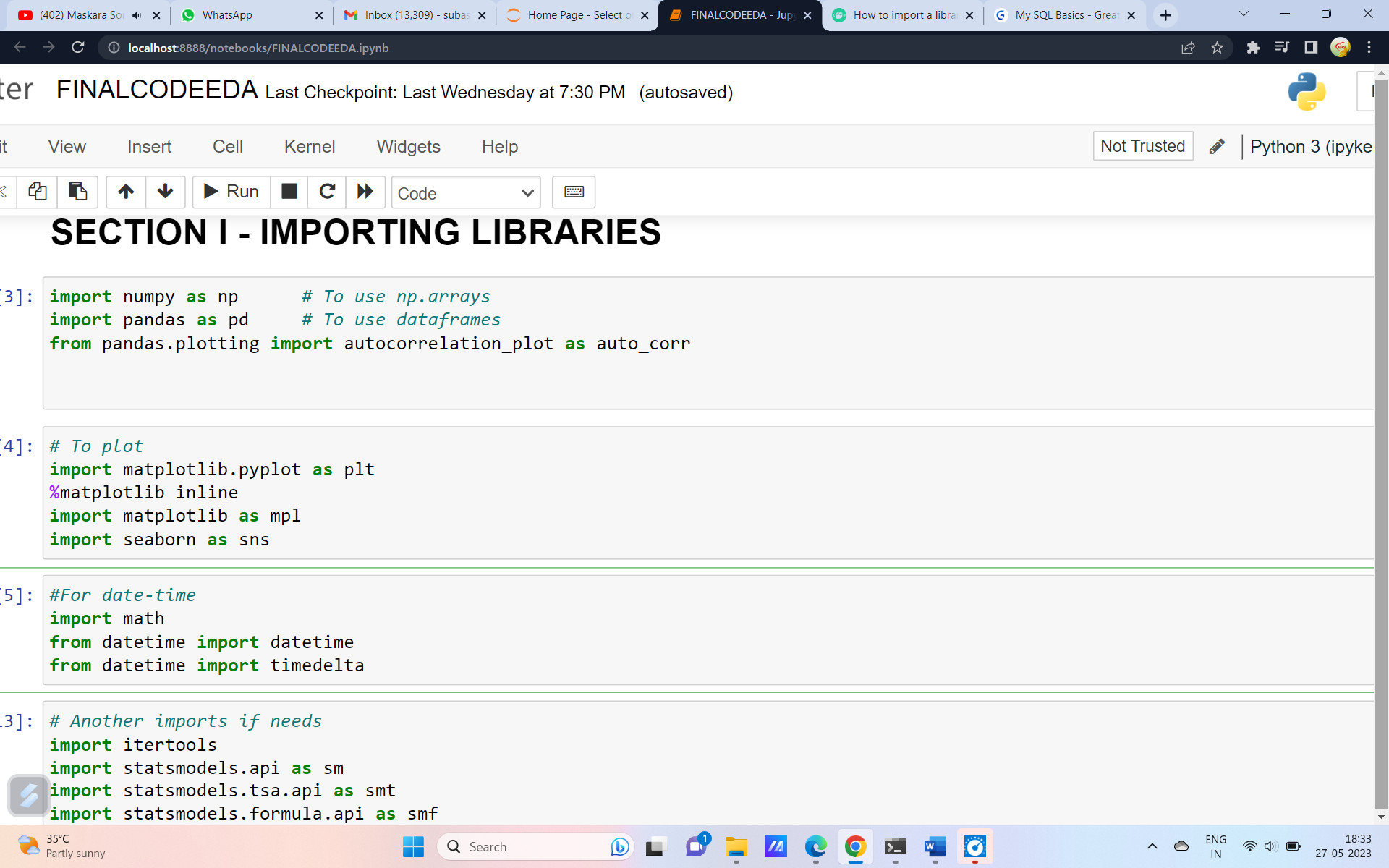
Prophet also imposes the strict condition that the input columns must be named as **ds (the time column)** and **y (the metric column)**.

**Chapter -4 Data Pre-processing Steps**

In this section we are going to start working on the dataset given. The flow of the coding is based on the flow given below.

**Section – I -Importing Libraries**

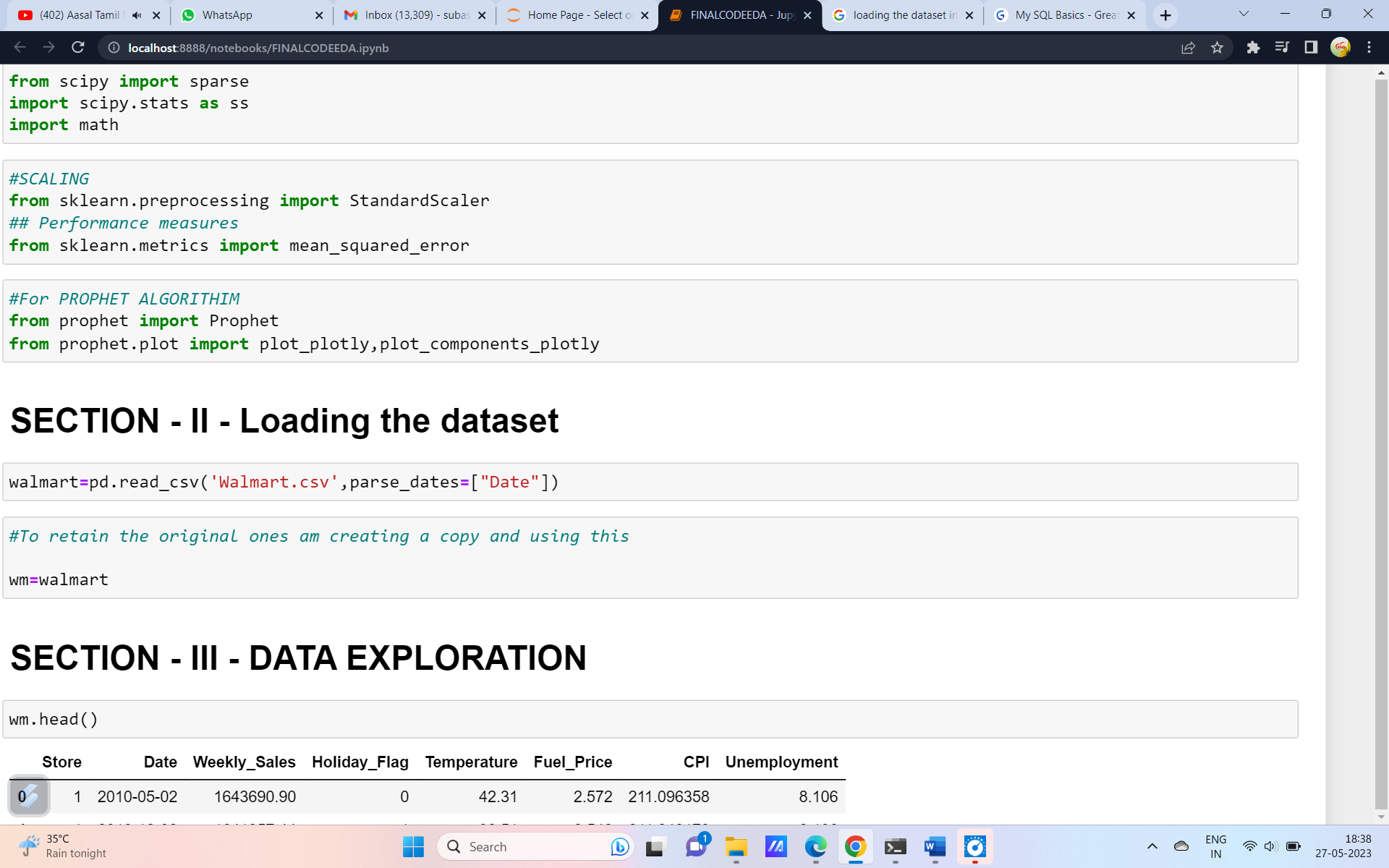
In Python, libraries are used to refer to a collection of modules that are used repeatedly in various programs without the need of writing them from scratch.. In Python, you can import libraries using the import statement. After importing the library, you can use its functions and methods by calling them with the library’s alias.



**Fig.1**

**Section II Loading the dataset**

In Python, you can load a dataset from various sources such as a CSV file, an Excel file, or a database. One common way to load a dataset is to use the pandas library. For example, to load a CSV file into a pandas DataFrame, you can use the read\_csv() function.



**Fig 2.**

In this code the data set is imported under the dataframe name Walmart. To retain the original dataset a copy of another dataframe is created under the name wm. Since the dataset deals with the date parse\_dates[‘Date’] is given for the date column.

Section III - Data Exploration

Data exploration is the process of analyzing and visualizing a dataset to better understand its characteristics and uncover any underlying patterns or relationships. In Python, there are several libraries that can be used for data exploration, including pandas, matplotlib, and seaborn.

The pandas library provides several functions for exploring data, such as the describe() function which generates summary statistics for numerical columns, and the value\_counts() function which counts the frequency of unique values in a categorical column.

For visualizing data, you can use the matplotlib and seaborn libraries to create various types of plots such as histograms, scatter plots, and box plots.

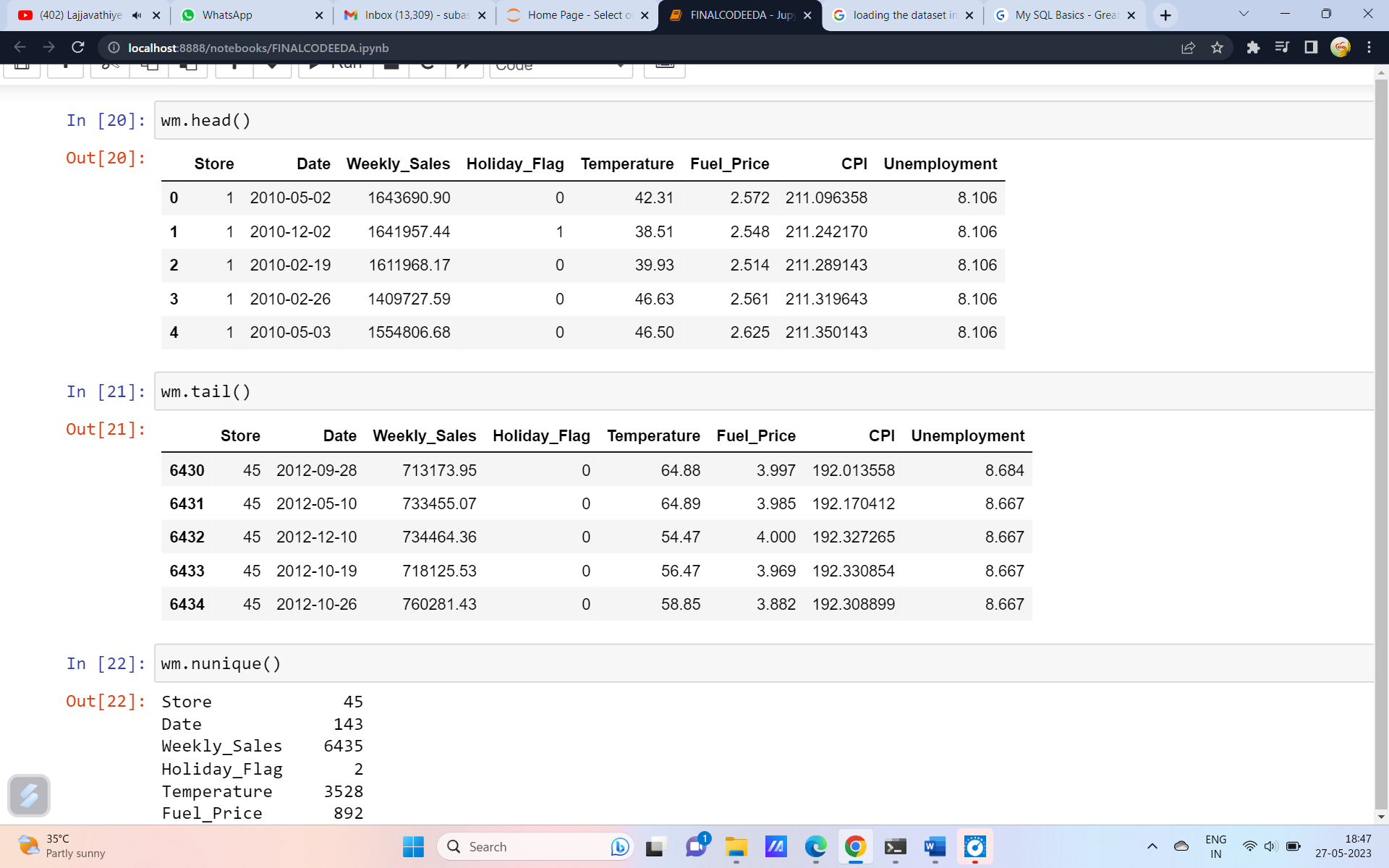


Fig-3

The dataset contains 6434 entries with 8 columns. There are no null values present.Thre is no need to deal with null values or NAN values.

# **SECTION - 4 - Data Visualization**

Data visualization is the process of representing data in a graphical or pictorial format to help communicate information clearly and effectively. In Python, there are several libraries that can be used for data visualization, including matplotlib, seaborn, and plotly. In this section a detailed EDA is carried out.

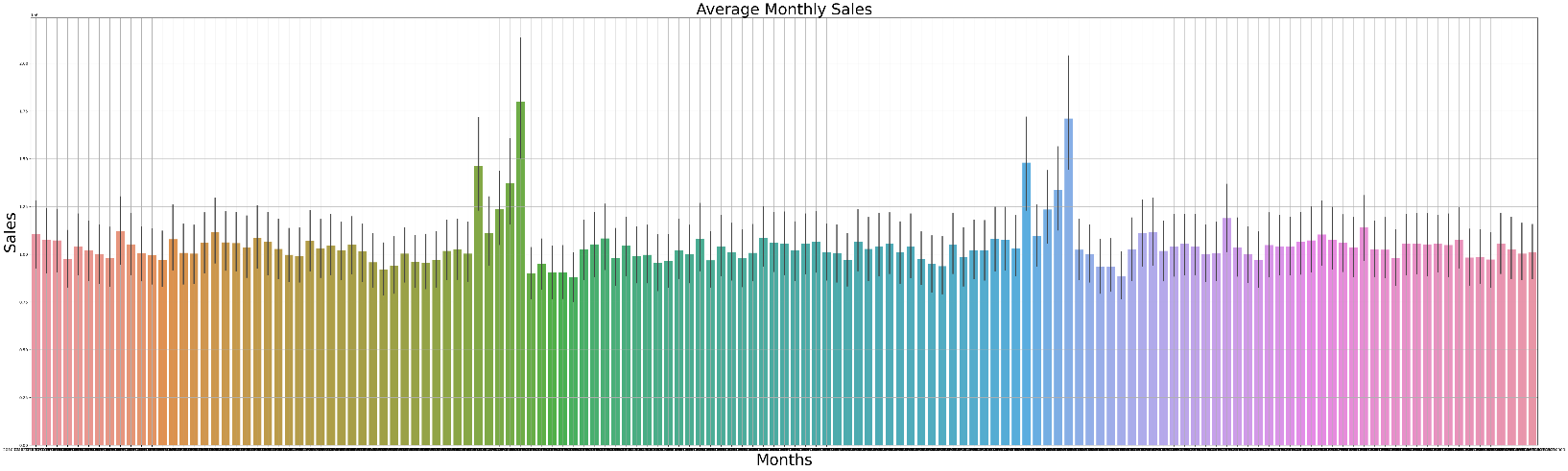


Fig.4 Average Monthly sales

The fig.4 gives the details of the Average monthly sales of every stores. The graph shows there is a peak in sales at the end of the year which is during the christmas and the new year.

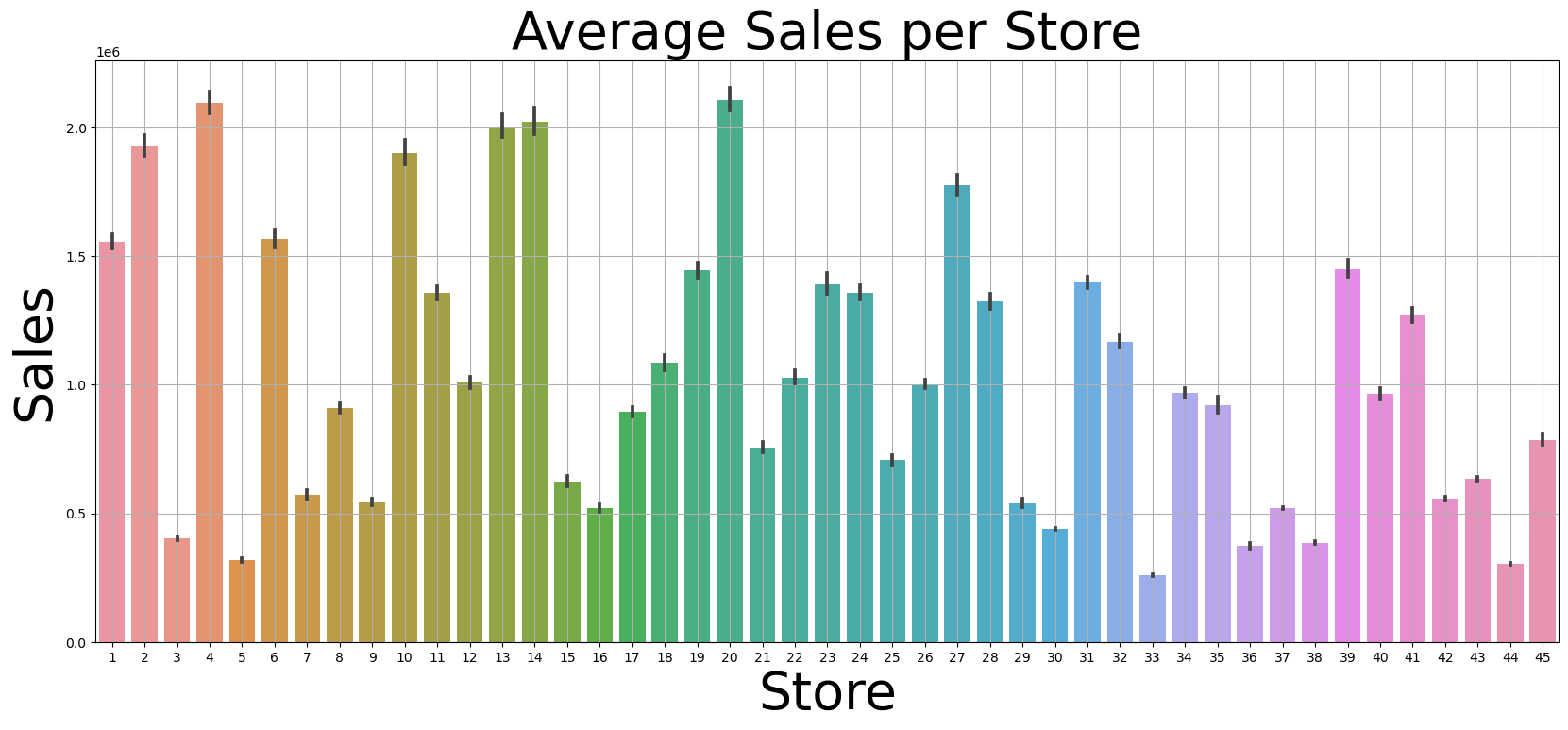


Fig.5 Average sales per store

Fig 5 shows the Average sales of all the 45 stores individually. From the graph we can see the stores 2,4,10,13,14,20 have high sales compared to other stores.

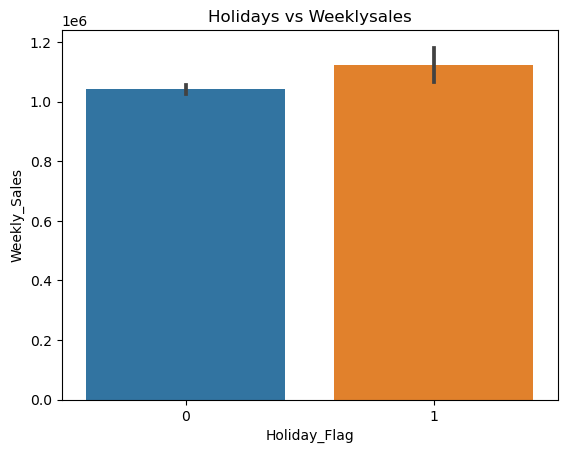


Fig.6 Effect of holidays

Fig.6 shows the effect of sales during holidays. From the graph it is clear that the sales are higher during holidays**.**

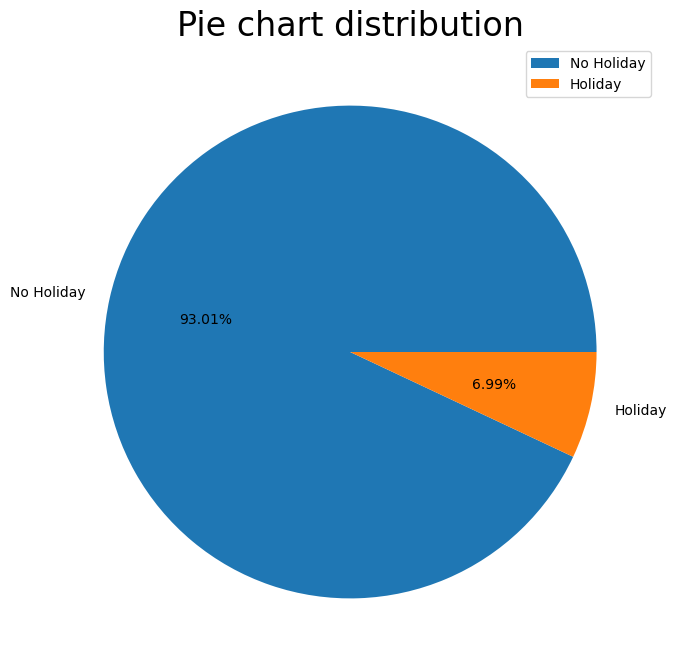


Fig.7 Distribution of Holidays in data

Fig.7 shows the percentage of Holidays and Non Holidays in the dataset. Which shows the dataset contains 93.01% of days are working days and remaining 6.99% is the holiday.

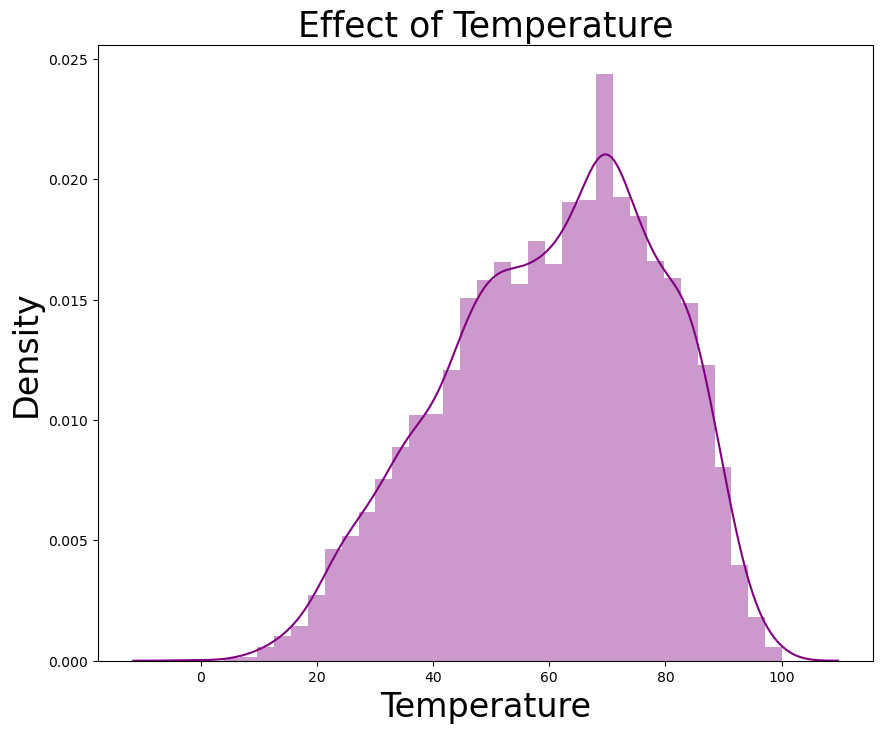


Fig.8 Effect of temperature

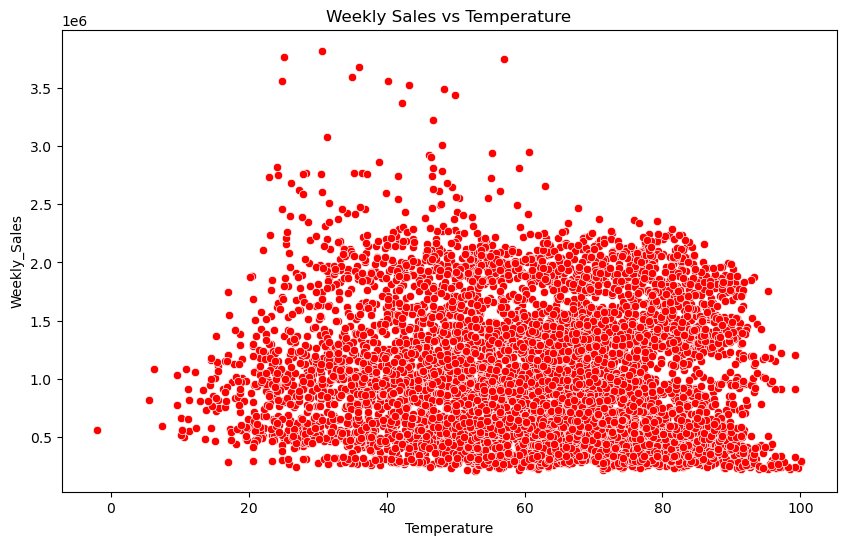


Fig.9 Effect of Temperature on weekly sales

Fig.8 and 9 shows the effect of temperature in the sales. Temperature change has the effect in the weekly sales.

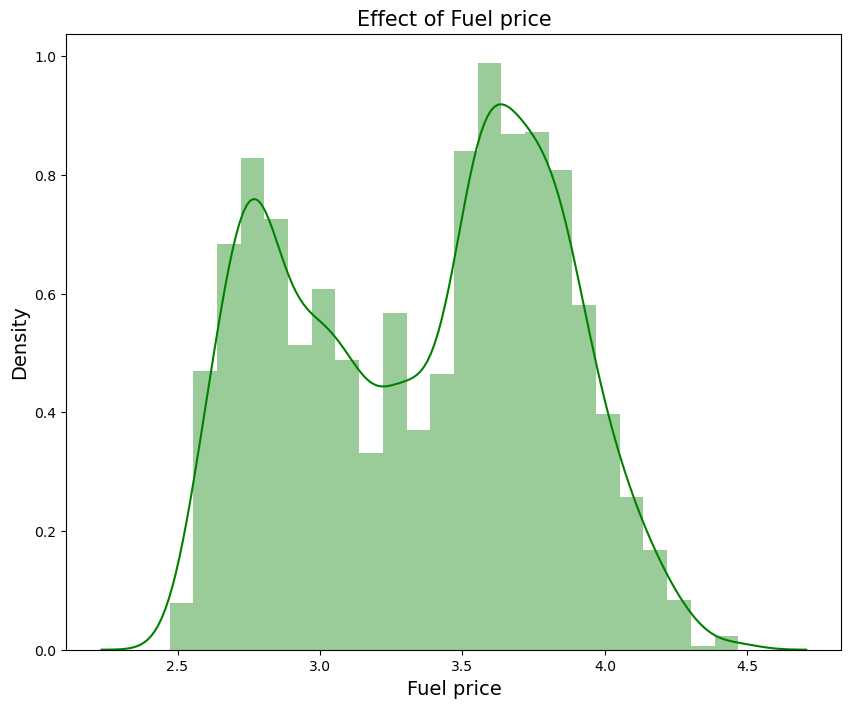


Fig.10 Effect of Fuel price

Fig.10 shows the effect of Fuel price in sales. We can see that when the fuel price increases there is a decrease in sales.

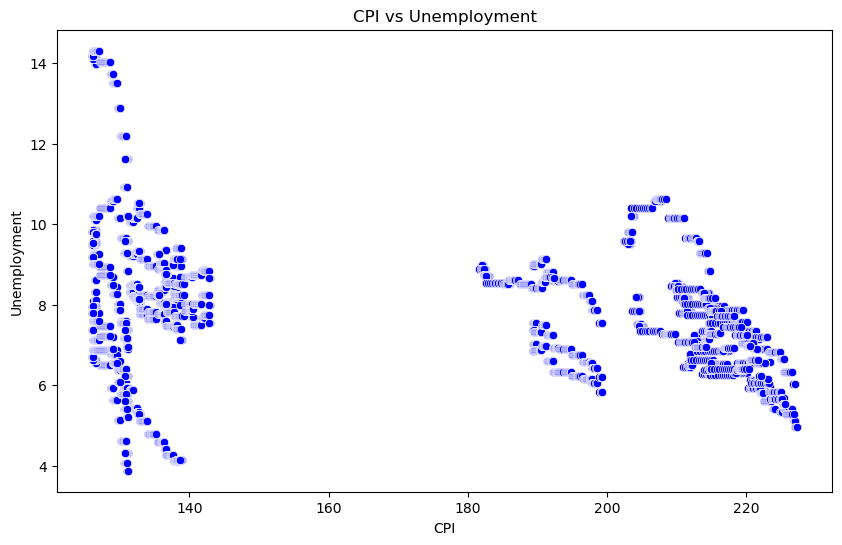


Fig.11 CPI vs Unemployment

Fig.11 shows the CPI vs Unemployment CPI:

CPI is the Consumer Price Index during that week. And Unemployment is the unemployment rate during that week in the region of the store

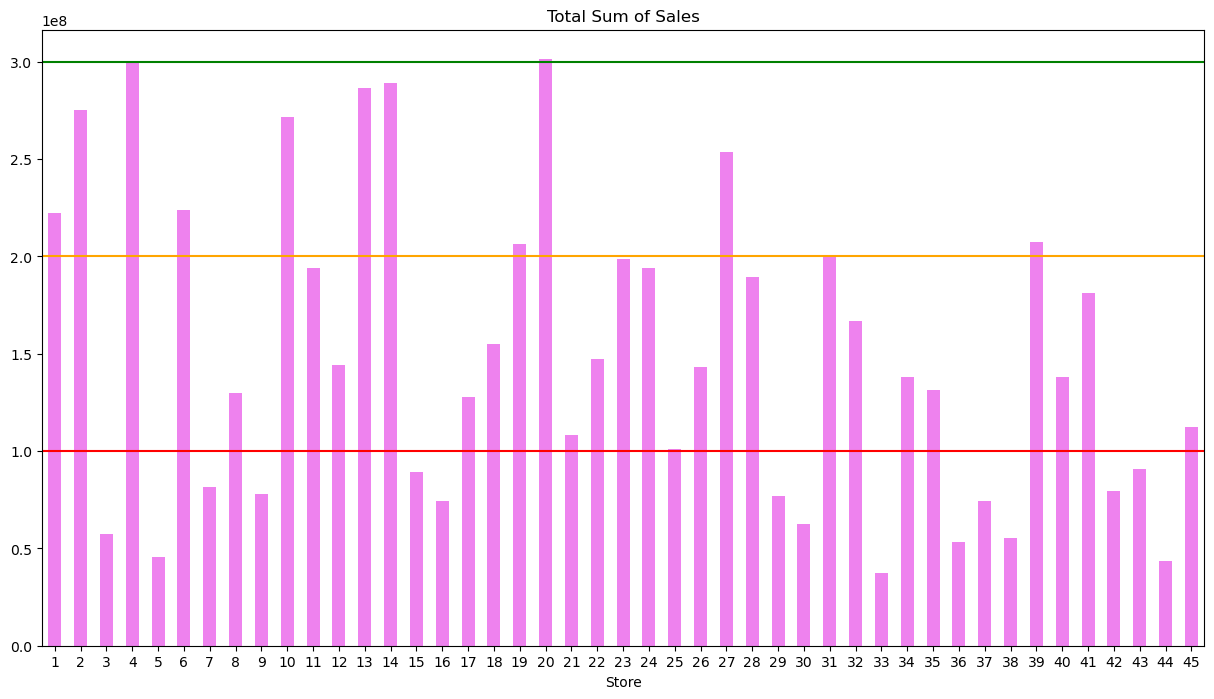


Fig 12. Total sum of sales of all stores

Fig.12 shows the total sum of the sales of every store. They are categorized into 3 types. Stores which have total sales of 1M,2M,3M. Green line shows 3M sales. Store number 4 and 20 have the total sales of

3M. There are about 11 stores which have more than 2M sum of sales.Store number 3,4,33,44 have the lowest sum of sales.

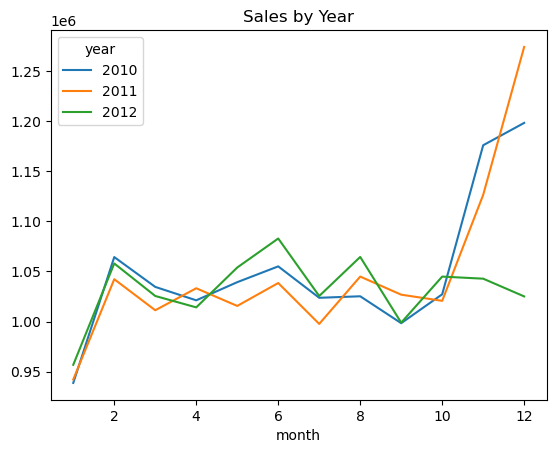


Fig.13 Sales by year

Fig.13 gives the sales by the year. The dataset has the sales of the year 2010,2011 and 2012. The graph shows the similar trend in all the years. Clearly there is a peak in the sales after the month of October and the sales reaches the peak during the December which is the Christmas time. We will analyse it more.

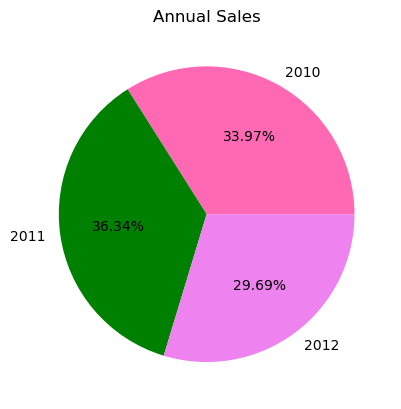


Fig 14. Anuual sales

Fig.14 shows the anuual sales for the year 2010,2011,2012. 2010 has the sales of about 33.97%. In the year 2011 the stores has achieved the sales of about 36.34%. 2012 shows only 29.69% since the data is upto October month of that year.

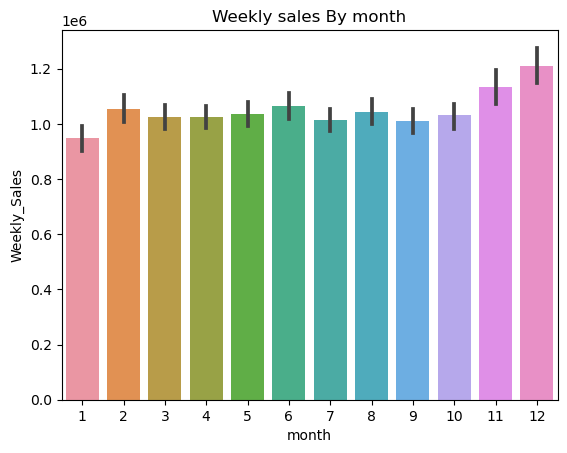


Fig.15 Weekly sales by Month

The fig.15 shows the weekly sales of every month in all the 3 years. the sales is high during the month of November and December.

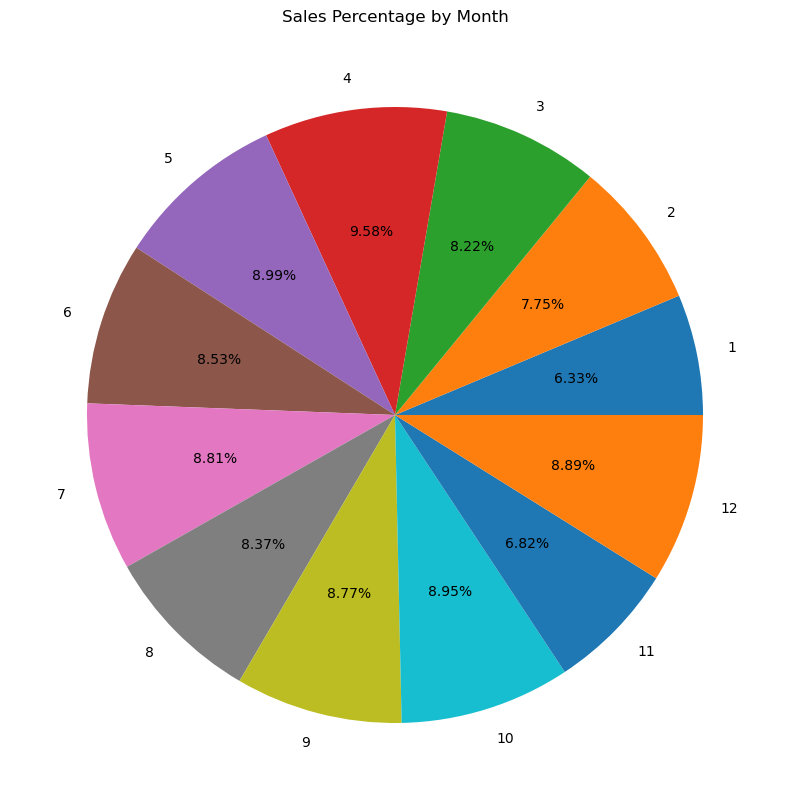


Fig.16 Sales percentage of every month

Fig.16 shows the sum of sales percentage of every month. April and December months have the highest percentage in the sales.

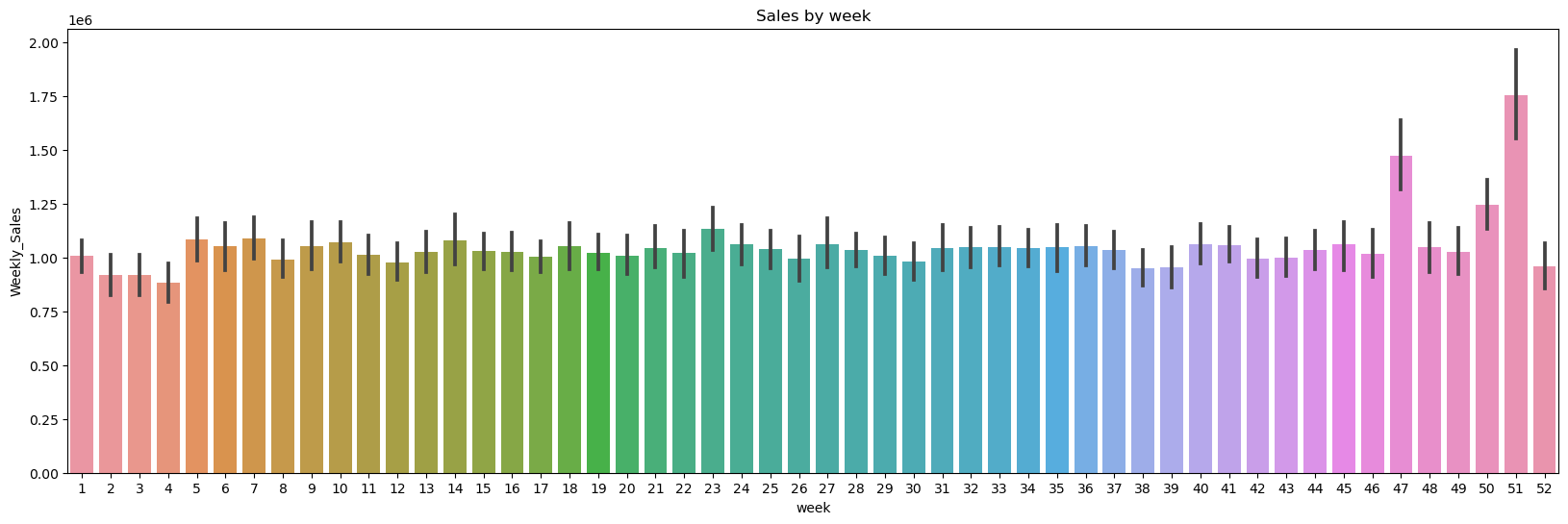


Fig.17 Sales by Week

Fig.17 shows the sales during all the weeks of all the 3 years.

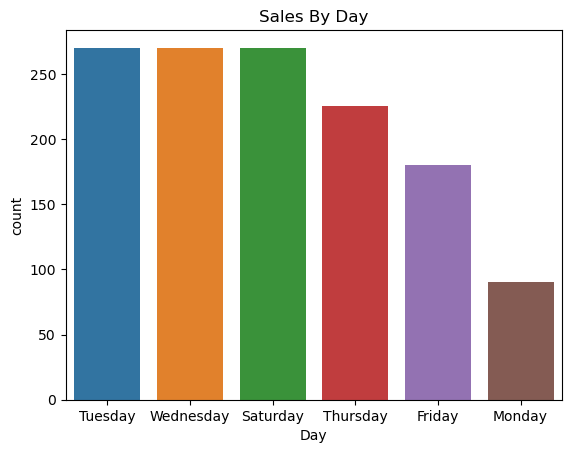


Fig.18 Sales by day’s of the week

Fig.18 shows the sales achieved in every day in a week. In this usually the sales is high during the Tuesday, Wednesday, and Saturday. Monday has the lowest sales.

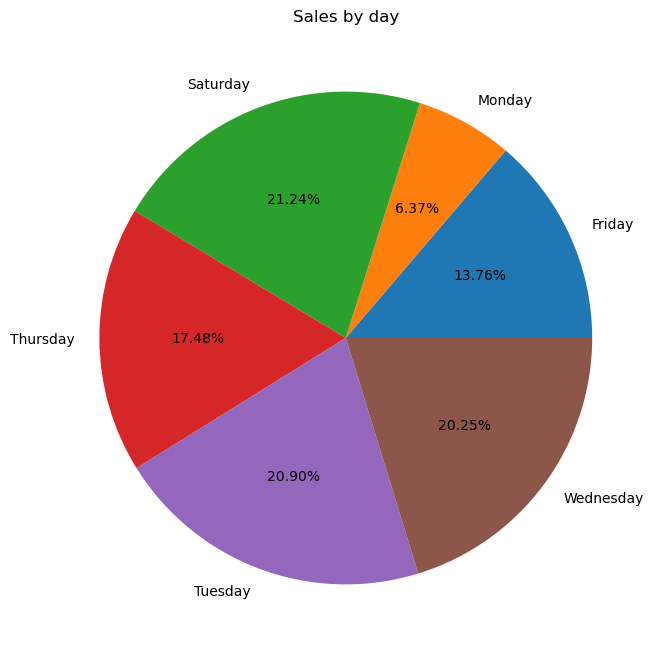


Fig.19 Sales by day’s of the week

Fig.19 shows the sales of each and every day of the week. As seen

Saturday,Tuesday and Wednesday have the high sum of sales and

Monday has a very low sales.

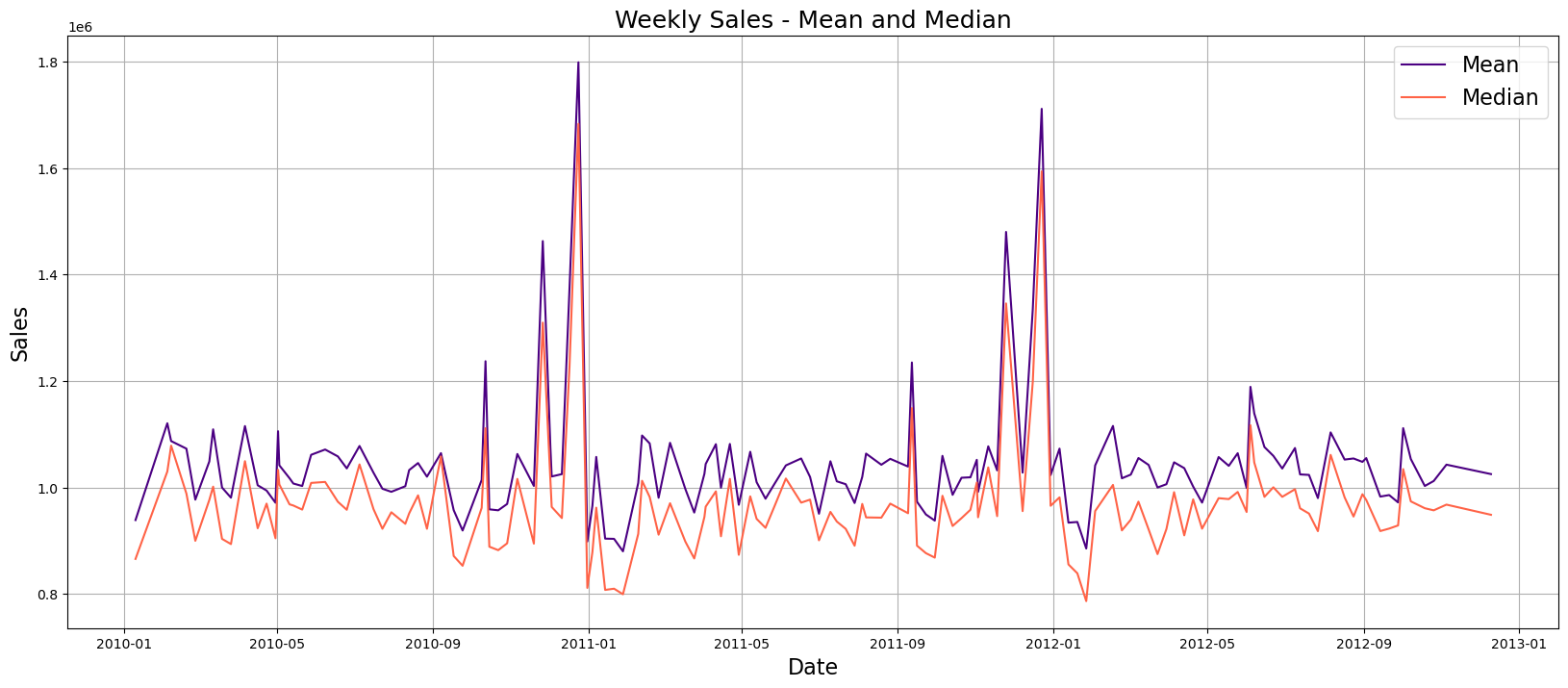


Fig.20 Mean and Median

Fig 20 shows the mean and median of the weekly sales in every year. Clearly we can see that the sales are high during the year end. And there is a decrease in sales after mid of January. The pattern is a repeating one. So there is a yearly trend.

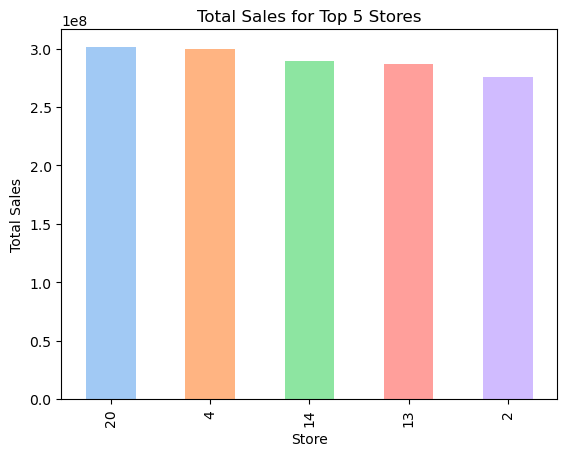


Fig 21. Top 5 stores which has the highest sales

Fig.21 shows the top 5 stores which has the highest sales. Store number 20,4,14,13,2 have the highest number of sales.

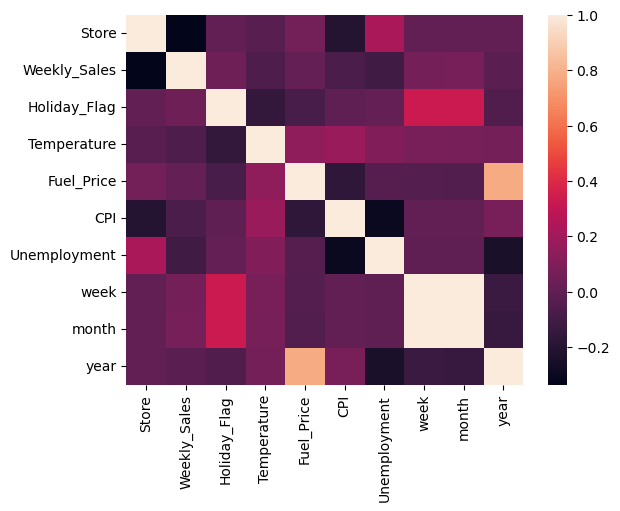


Fig.22 Correlation Matrix

Fig 22 shows the correlation matrix for each and every columns.

Discounts are correlated and higher unemployment means lower Consumer Price Index. More interestingly, it appears that higher department numbers have higher sales. Maybe because they are newer? Also, larger stores generate more sales, discounts generally generate higher sales values and larger unemployment result in a bit fewer sales. Unfortunately, there appears to be little relationship between holidays, temperatures or fuelprices with our weekly sales.

CHAPTER -5.

Choosing the Algorithm for the Project

The given problem is a time series problem as discussed in previous section the models are built using ARIMA,SARIMAX,and PROPHET. We will compare the results to see which model performs good and the final model is built using that Algorithim.

Before going into Model building we will discuss some of the time series components for the data set to check the stationarity,and all.

**Augmented Dickey-Fuller test(ADF)**

Augmented Dickey-Fuller Test or ADF test is one of the most popular statistical methods to determine whether a time series data is stationary. In Python, we can directly use the [adfuller package](https://www.statsmodels.org/stable/generated/statsmodels.tsa.stattools.adfuller.html" \t "_blank) from statsmodels to see whether the data is stationary at different significance levels. Let us use the ADF test on our step count data to see whether it is stationary or not.

from statsmodels.tsa.stattools import adfuller

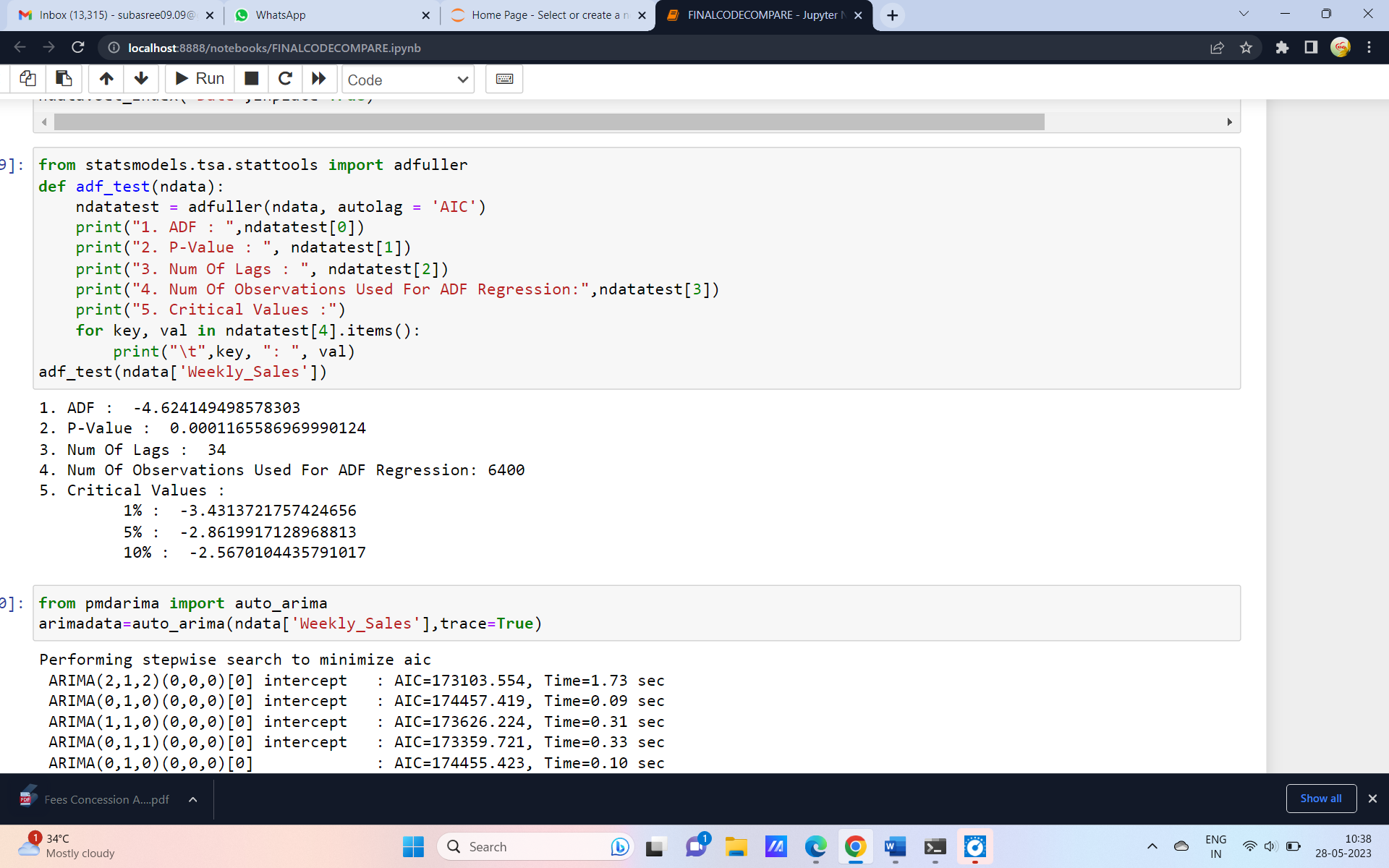


Fig 23 ADF test

From the fig 23 we can observe that **p-value <= 0.05**

so the data given is stationery. Since the data is stationary we will proceed further.

**Autocorrelation Plot and Partial autocorrelation plot:**

Autocorrelation plots are commonly used in time series analysis to analyze the properties of the data and to check for patterns such as seasonality or trends

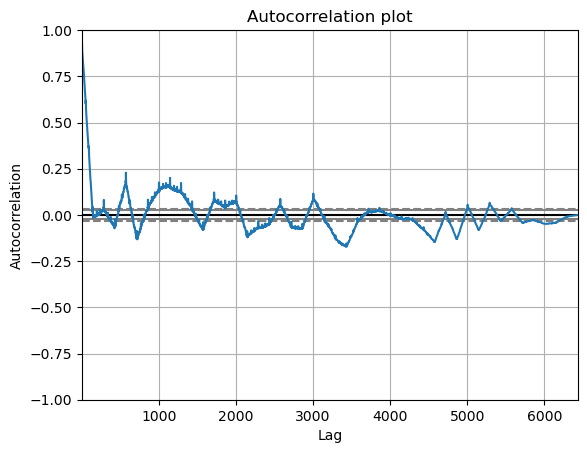


Fig.24 Autocorrelation plot

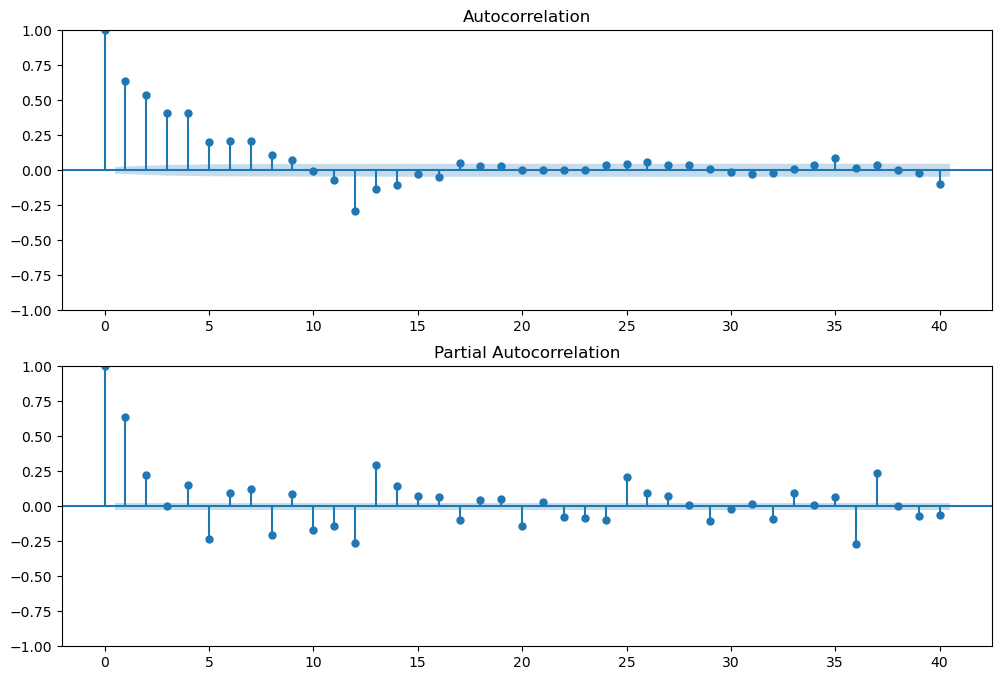


Fig 25 ACF and PACF plot

ACF plot is used to identify the lags and PACF plot is used to find the order of the data.

**ARIMA MODEL :**

The model building starts from this part. First is the ARIMA model.

from pmdarima import auto\_arima

arimadata=auto\_arima(ndata['Weekly\_Sales'],trace=True)

In the Arima the above part of the code can be used to find the best fit order of p,d,q values. The above code has given the output of (4,1,5) so we will use these values while building out model.

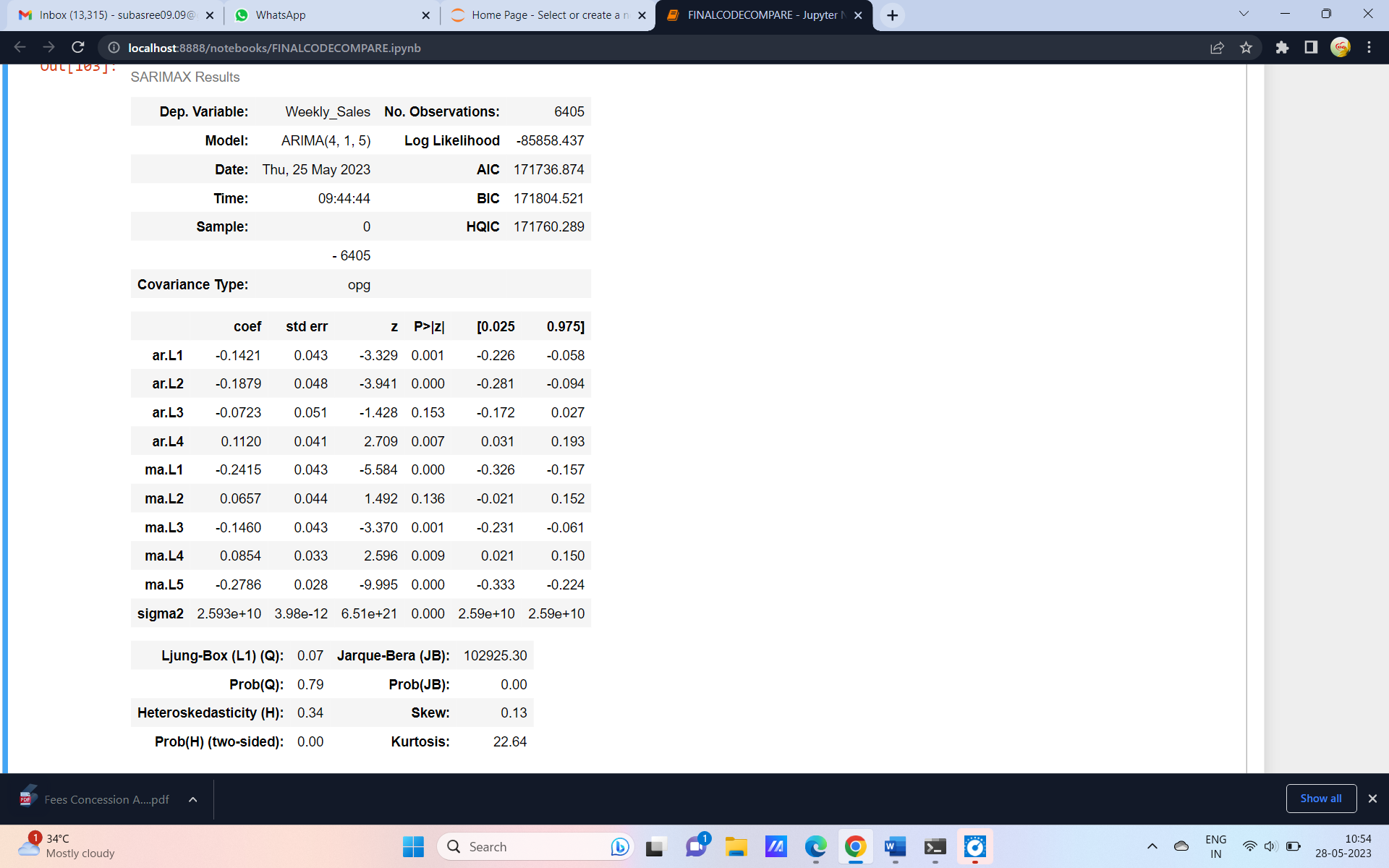


Fig 26 P values by the model.

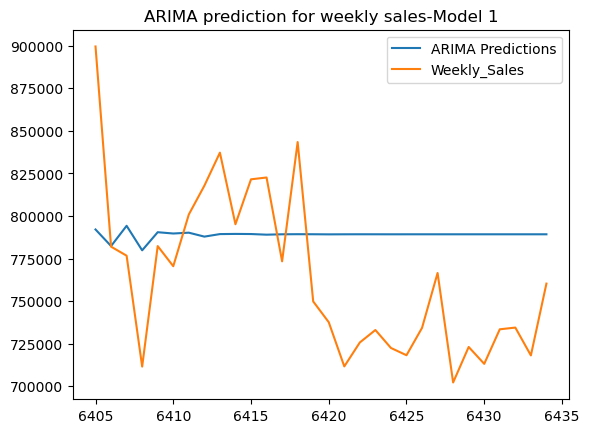


Fig.27 ARIMA model 1 results

From the fig.27 the Model 1 does not give the best results so we do some hyper parameter tuning and built the model again.

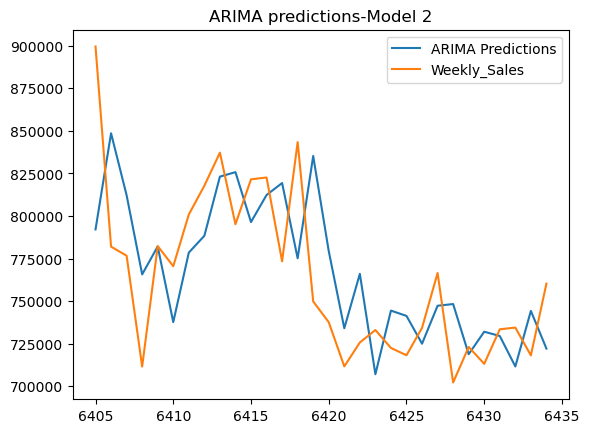


Fig.28 Arima Predictions by model 2

Fig.28 shows the second model gives the better results and the predictions were very near to the actual sales.

Now lets do the forecast for the next 12 weeks.

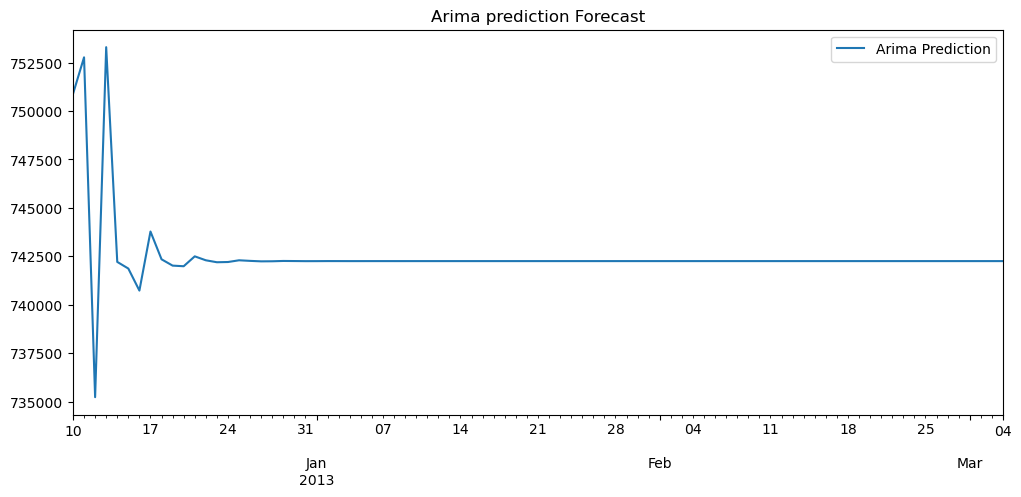


Fig 29 Arima Forecast

From the fig 29 it is clearly seen that the Arima model did not predict well after December month. This may be due to several reasons. It could be due to the nature of the data, the model parameters, or the way the model is being used. since forecasted line is flat, it could indicate that the model is not able to capture the patterns in the data. And another reason could be that the data is unpredictable by ARIMA. We will go for next mode SARIMAX

**SARIMAX:**

SARIMAX stands for Seasonal Auto-Regressive Integrated Moving Average with exogenous factors. It is an updated version of the ARIMA model that can handle seasonality and external effects

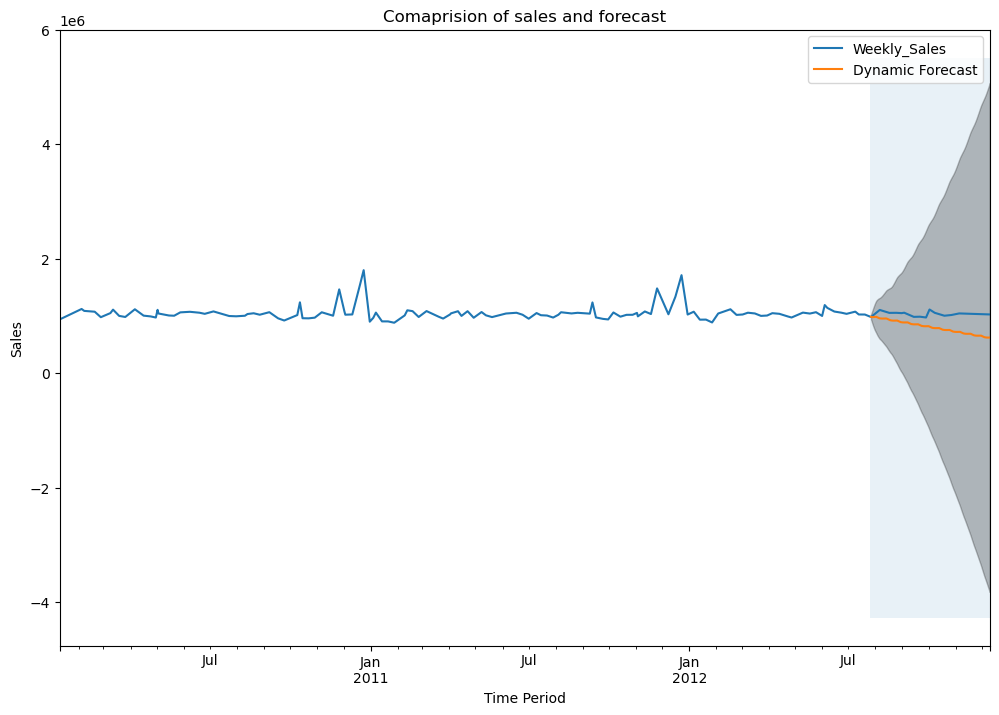


Fig.30 One step ahead forecast

The forecast has been done fore already present data and Sarimax was not able to capture that properly and the model was not able to forecast the sales. We will try further.

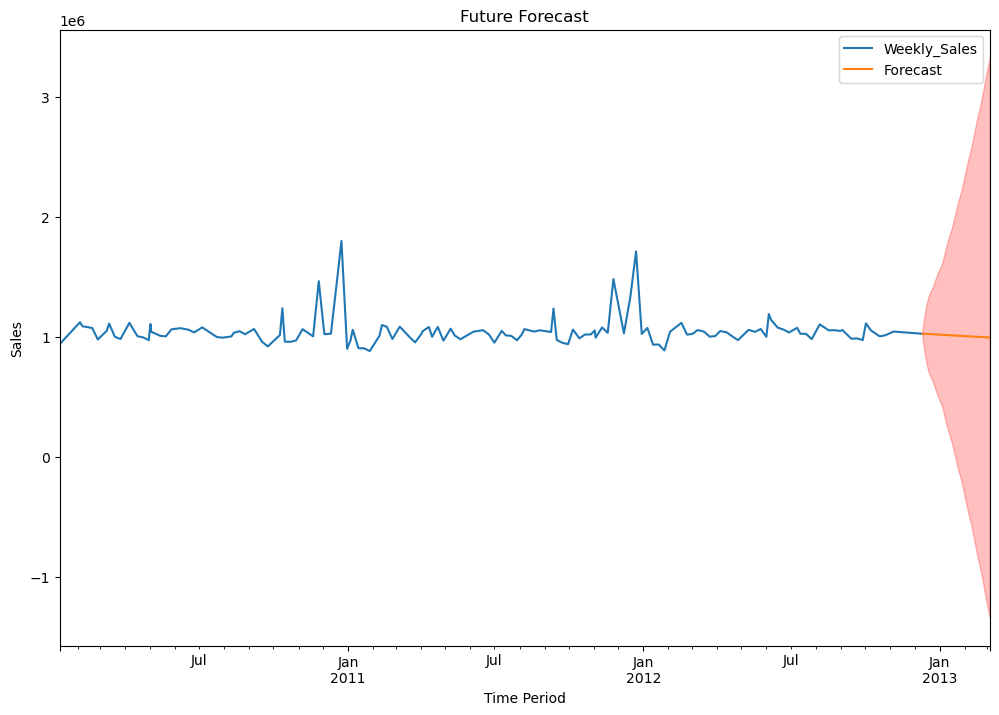


Fig. 31 Future forecast for 12 weeks

Fig.31 shows the forecast for future 12 weeks. The Sarimax was also not able to predict well. The line is flat and the model is not able to forecast the sales. So we will proceed with the Prophet model.

**PROPHET:**

In this section the Model is built using Prophet. When the model is fitted for the predictions which are already present we got the following output.

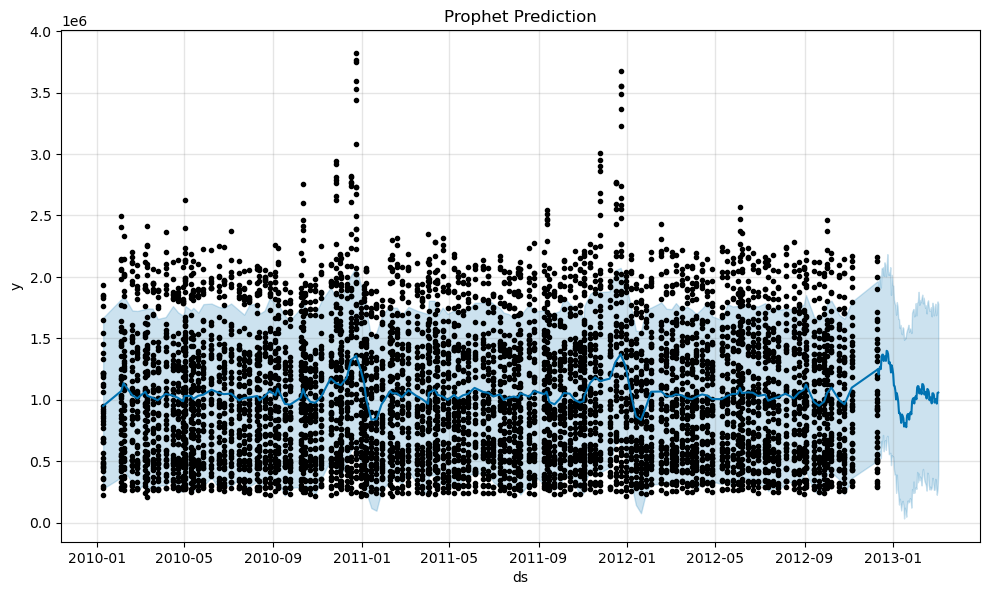


Fig.32 Weekly sales Prediction and forecast

Fig 32 shows the prophet was able to predict the sales accordingly .

It was also able to catch the trend in the data.



Fig.33 Model components

Fig 33 shows the various model components which shows the trend is during the end of year .the weekly and yearly components were also shown.

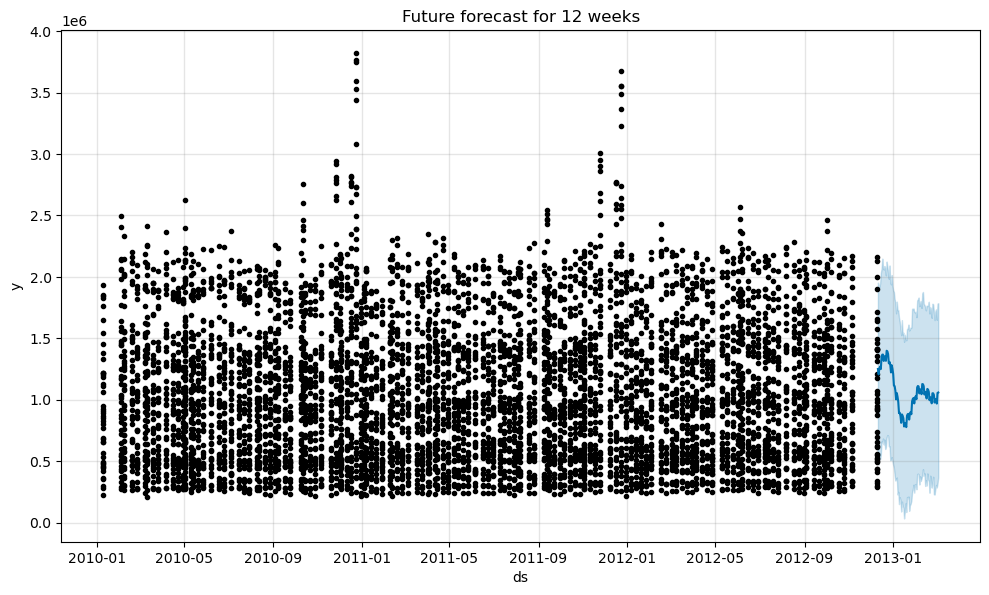


Fig 34 Future forecast for 12 weeks



Fig. 35 Future sales alone

Fig 34 and 35 shows the future forecast for 12 weeks. From the fig 35 it is clearly seen that the sales rise from December 16 and there is a fall of sales after 1 st week of new year and at the end of the January the sales will be increasing again . The model is clearly able to predict the trend present in the data. The pattern we got is very much similar to the previous year trends also. So in general we can conclude the Prophet model works well for this data.

The reasons why prophet is working well than the traditional models is Prophet is a forecasting tool has some advantages over traditional time series models such as ARIMA and SARIMAX. One advantage of Prophet is that it can handle data with missing values and large outliers, which can be challenging for traditional time series models. Additionally, Prophet is designed to be more flexible and easier to use than traditional time series models, making it more accessible to non-experts. However, it is important to note that the performance of any forecasting model depends on the specific characteristics of the data being analyzed, and there may be cases where ARIMA or SARIMAX models perform better than Prophet.

So it is clear that the prophet model gives the better results . So we are going to select PROPHET and build a model using that. The next sections deals with that.

Chapter 7. Model Building

The previous section we have decided to build a model using prophet. In this section we will discuss how the model is built. Since the problem statement is to predict the sales in each and every store so we are going to analyse each store. The model gives the sales prediction and forecast for all the 45 stores individually.

from prophet import Prophet

from prophet.plot import plot\_plotly,plot\_components\_plotly

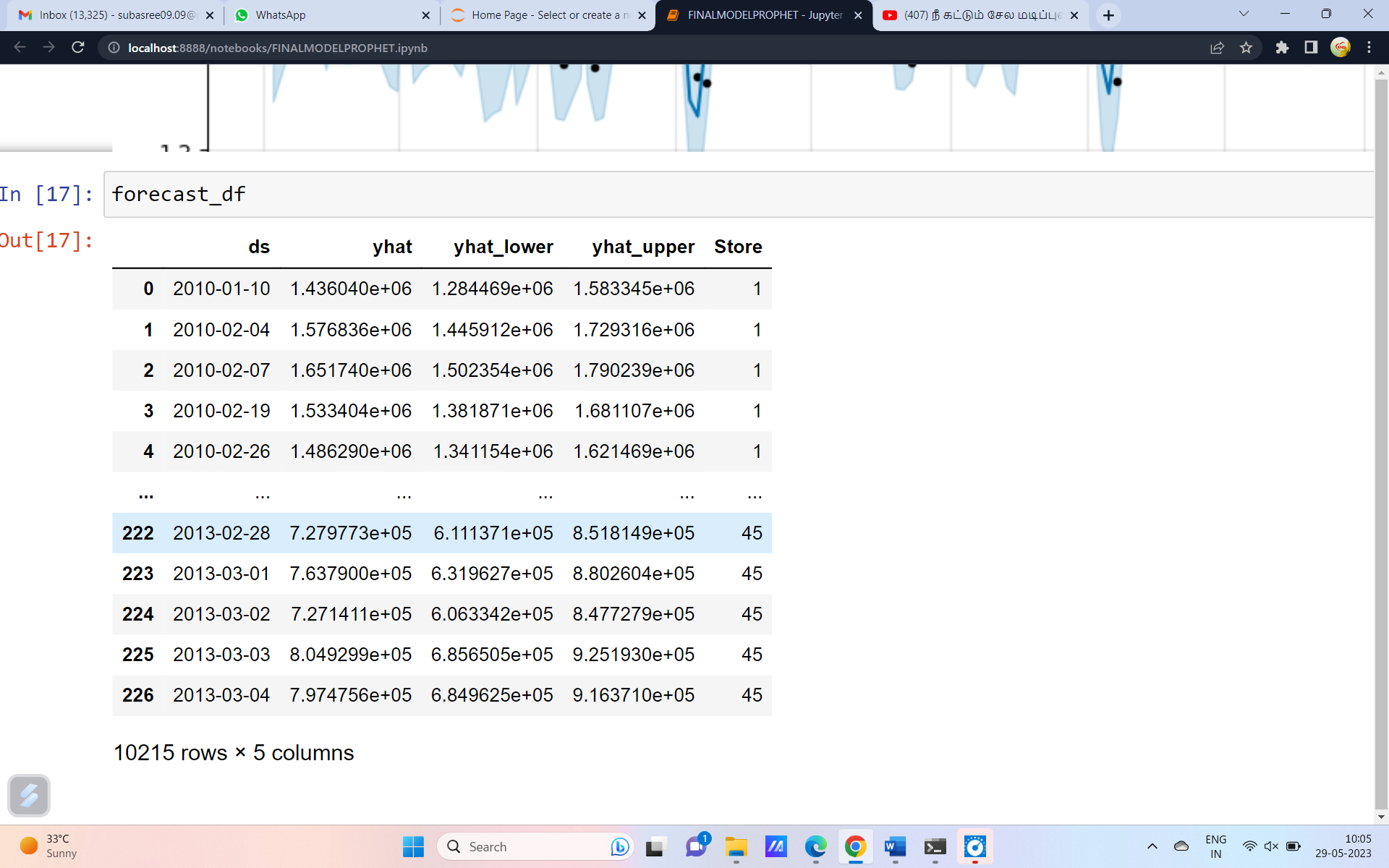
using the above library prophet is imported.

Prophet also imposes the strict condition that the input columns must be named as **ds (the time column)** and **y (the target column)**. So, we must rename the columns in our dataframe.

In order to obtain forecasts of our time series, we must provide Prophet with a new DataFrame containing a ds column that holds the dates for which we want predictions.

Prophet returns a large DataFrame with many interesting columns, but we subset our output to the columns most relevant to forecasting. These are:

* **ds**: the datestamp of the forecasted value
* **yhat**: the forecasted value of our metric (in Statistics, yhat is a notation traditionally used to represent the predicted values of a value y)
* **yhat\_lower**: the lower bound of our forecasts
* **yhat\_upper**: the upper bound of our forecasts
* A variation in values from the output presented is to be expected as Prophet relies on **Markov chain Monte Carlo (MCMC)** methods to generate its forecasts.
* MCMC is a stochastic process, so values will be slightly different each time.



Prophet also provides a convenient function to quickly plot the results of our forecasts as follows.Prophet plots the observed values of our time series (the black dots), the forecasted values (blue line) and the uncertainty intervals of our forecasts (the blue shaded regions).

The below table shows the output for all the 45 stores

|  |  |
| --- | --- |
| **Store number** | **Forecast** |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |
| 21 |  |
| 22 |  |
| 23 |  |
| 24 |  |
| 25 |  |
| 26 |  |
| 27 |  |
| 28 |  |
| 29 |  |
| 30 |  |
| 31 |  |
| 32 |  |
| 33 |  |
| 34 |  |
| 35 |  |
| 36 |  |
| 37 |  |
| 38 |  |
| 39 |  |
| 40 |  |
| 41 |  |
| 42 |  |
| 43 |  |
| 44 |  |
| 45 |  |

**Chapter 8. Model Evaluation and Techniques**

The Model is built in such a way that sales can be forecasted for the store number of which the user input is given. Model is deployment-ready by giving User-Input provision.

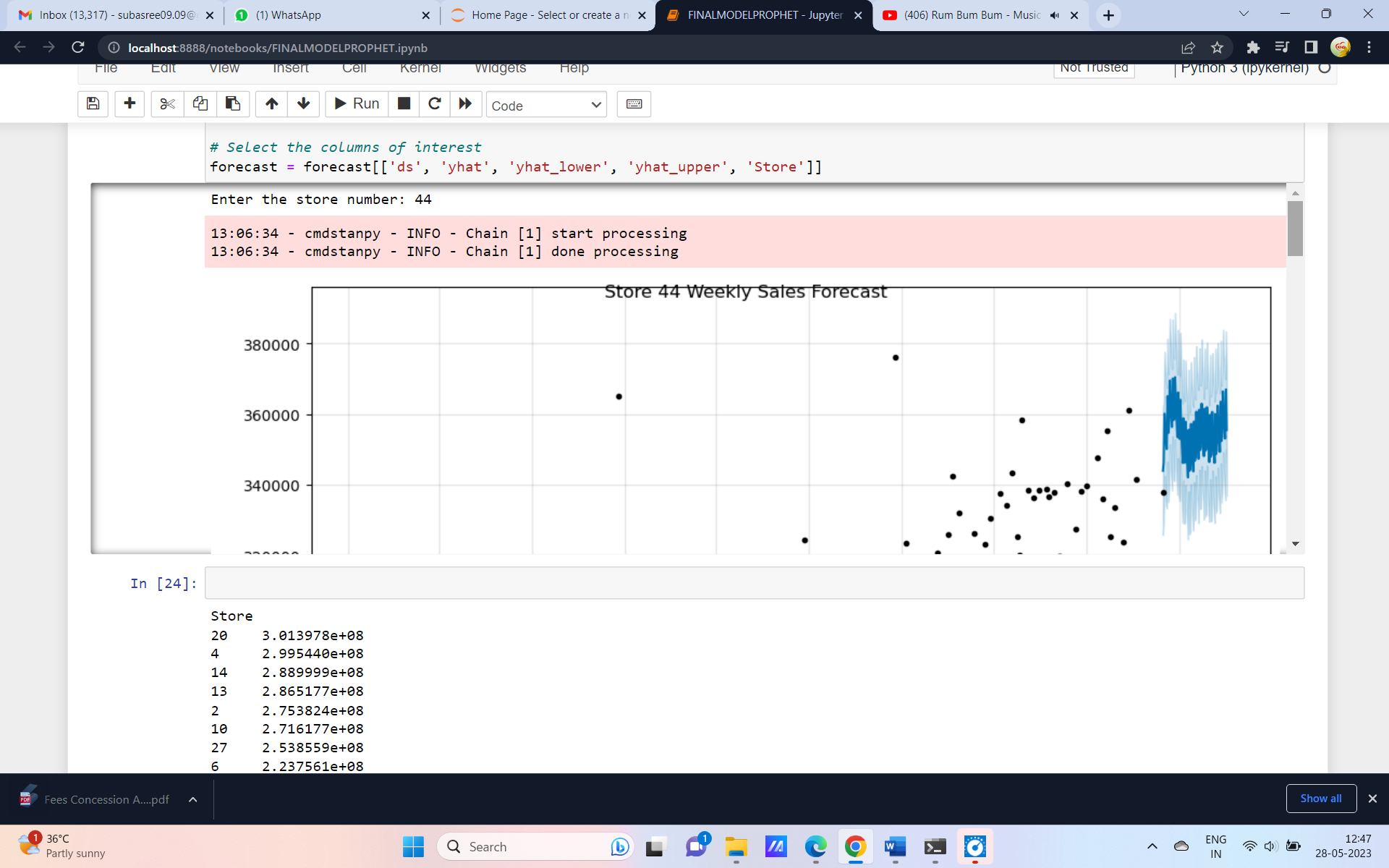


Fig. 36 User input provision

This model also provides the provision to compare 2 or more stores at the same time. Which can be the input given by user.



Fig 37 sales comparision for different stores

The future sales of the different stores can also be compared by getting the input.

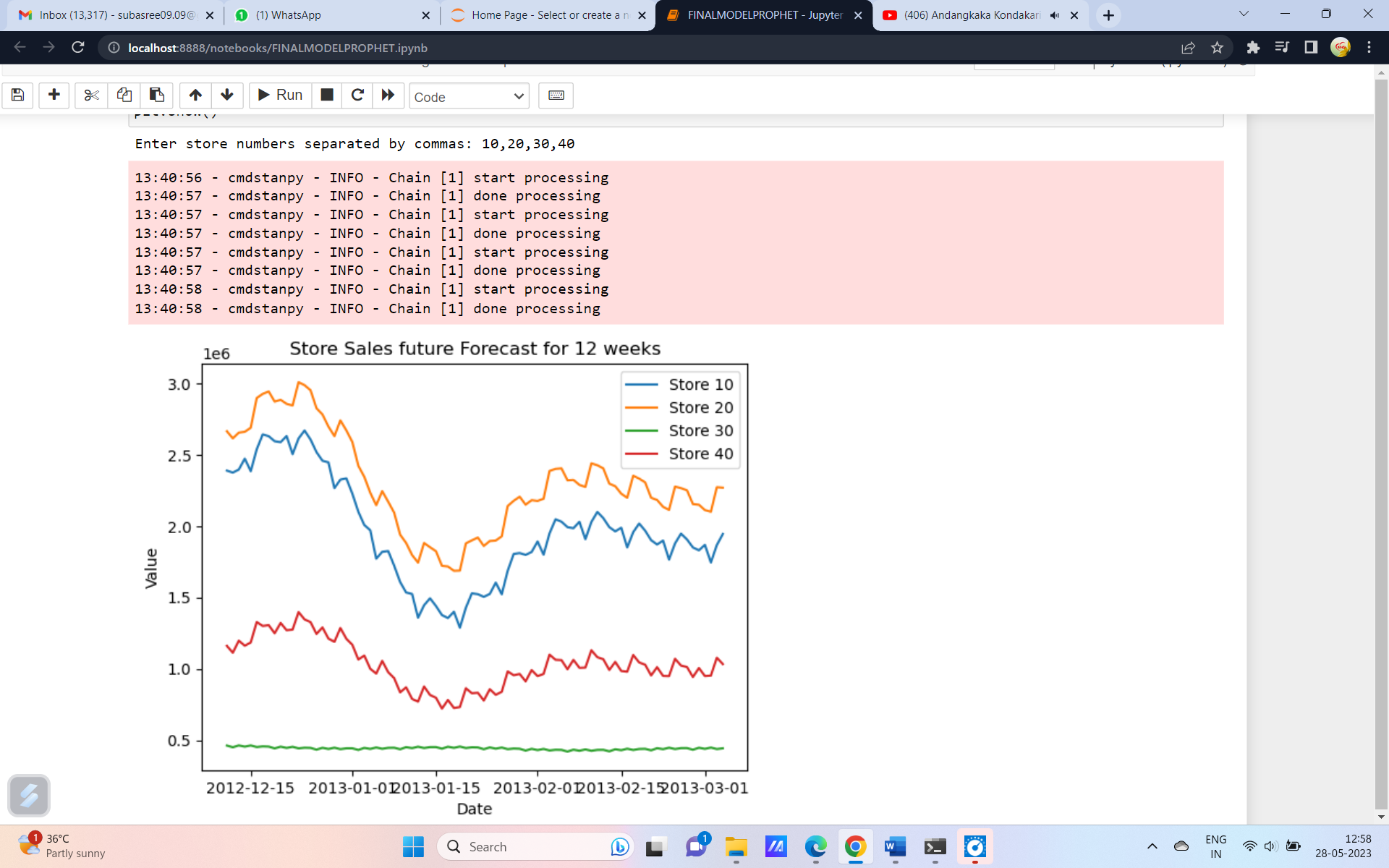


Fig.38 Sales forecast for stores

The sales forecast for the top 5 stores and the last 5 stores is also given which is easy to identify the top and the last selling stores.

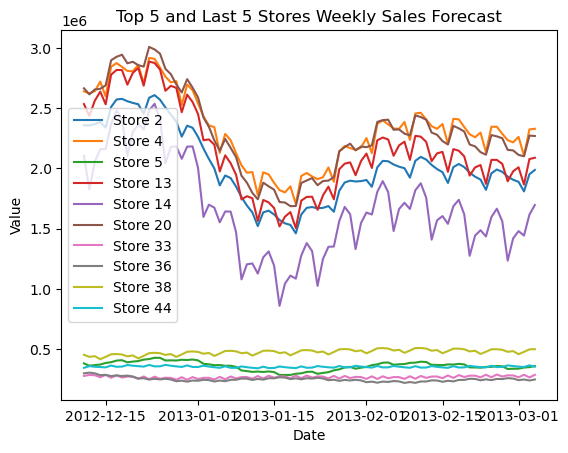


Fig 39 Sales forecast for top 5 and last 5 stores

**Chapter 9. Inferences and Recommendations**

From the results we got by the model here are some of the recommendations that can be done

1. **Each store has a unique prediction power**. They can be separately analyzed to get prediction for each individual store
2. The **Sales are very high during November and December and go down in January**. So its better to employee more staff as casual employee in November and December
3. The predicted sales data can be used to analyse the sales pattern and accordingly **adjust the staff in the store.**
4. The **low selling stores should look forward to increasing their size and capacity to store more items and consumer products.**
5. **Special discount coupons can be distributed during low selling periods to attract more customers**
6. Sales are likely to fluctuate during holidays. **Special offers can be given during festive season** accompanied with suitable marketing to keep the sales high during holidays as well

**Chapter 10 Future Possibilities of the Project**

The future of machine learning is exceptionally exciting. Machine learning is already being used in a wide range of industries, including healthcare, search engines, digital marketing, and education. [.](https://dzone.com/articles/the-future-of-machine-learning-trends-to-watch) The current model is deployed using a time series algorithim. The same can be modelled using Neural Networks. Neural networks can be used to forecast future sales of a dataset. Neural networks are a type of machine learning algorithm that can learn to make predictions based on past data. They can be trained on historical sales data to learn the underlying patterns and relationships between the input variables and the target variable (sales). Once trained, the neural network can be used to make predictions about future sales.

**Chapter 11 – Conclusion**

The project undertook the sales forecast for a retail store which has 45 stores. Some of the important findings and recommendations were given which includes,

* 1. The sales will reach a peak during the month of November and December and sales decreases during the month of June.
  2. To improve the sales some recommendations were given.

Thereby concluding the project.

Chapter 12 – REFERENCES

1. <https://facebook.github.io/prophet/>
2. <https://facebook.github.io/prophet/docs/quick_start.html>
3. <https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/>
4. <https://towardsdatascience.com/stationarity-assumption-in-time-series-data-67ec93d0f2f>
5. <https://www.statsmodels.org/stable/generated/statsmodels.tsa.stattools.adfuller.html>. (ADF test package in Python)
6. Kaggle.com/ARIMA Model for Time Series Forecasting by Prashant banarjee

CODE

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

import warnings

warnings.filterwarnings('ignore')

import datetime

wd=pd.read\_csv('Walmart.csv',parse\_dates=["Date"])

from prophet import Prophet

from prophet.plot import plot\_plotly,plot\_components\_plotly

grouped\_data=wd.groupby('Store')

wd

wd['Date']=pd.to\_datetime(wd['Date'],format='%d-%m-%Y')

data1=wd.sort\_values(['Store','Date'],ascending=[True,True])

data1.head()

wd=data1

wd=wd.reset\_index()

wd=wd.drop(['index'],axis=1)

wd

import os

forecast\_list=[]

for store,wmr in grouped\_data:

prophet\_data=wmr[['Date','Weekly\_Sales']]

prophet\_data.columns=['ds','y']

model=Prophet()

model.fit(prophet\_data)

future=model.make\_future\_dataframe(periods=84,freq='d')

forecast=model.predict(future)

fig=model.plot(forecast)

fig.suptitle(f'Store {store} Weekly Sales Forecast')

folder\_path = 'C:\\AL and ML course\\Projects\\code images\\Prophet\\main'

if not os.path.exists(folder\_path):

os.makedirs(folder\_path)

plt.savefig(f'{folder\_path}\\Store\_{store}\_Weekly\_Sales\_Forecast.png')

forecasting\_plot=model.plot\_components(forecast)

folder\_path = 'C:\\AL and ML course\\Projects\\code images\\Prophet'

if not os.path.exists(folder\_path):

os.makedirs(folder\_path)

plt.savefig(f'{folder\_path}\\Store\_{store}\_Weekly\_Sales\_Forecast.png')

plt.show()

forecast['Store']=store

forecast\_list.append(forecast[['ds','yhat','yhat\_lower','yhat\_upper','Store']])

forecast\_df=pd.concat(forecast\_list)

le(f'Store {store} Weekly Sales Forecast')

plt.savefig(f'{folder\_path}\\Store\_{store}\_Full\_Weekly\_Sales\_Forecast.png')

#Plot components

forecasting\_plot = model.plot\_components(forecast)

future\_forecast = forecast[forecast['ds'] > last\_date]

folder\_path = 'C:\\AL and ML course\\Projects\\code images\\try'

if not os.path.exists(folder\_path):

os.makedirs(folder\_path)

# Plot the future forecasts

plt.figure()

plt.plot(future\_forecast['ds'], future\_forecast['yhat'], label='forecast')

plt.fill\_between(future\_forecast['ds'],

future\_forecast['yhat\_lower'],

future\_forecast['yhat\_upper'], color='k', alpha=.2)

# Add title, labels and legend

plt.title(f'Store {store} Weekly Sales Forecast')

plt.xlabel('Date')

plt.ylabel('Value')

plt.legend()

# Save the filtered plot

plt.savefig(f'{folder\_path}\\Store\_{store}\_Filtered\_Weekly\_Sales\_Forecast.png')

# Show the plots

plt.show()

# Add the store number to the forecast data

forecast['Store'] = store

# Select the columns of interest

forecast = forecast[['ds', 'yhat', 'yhat\_lower', 'yhat\_upper', 'Store']]

top\_5\_stores = wd.groupby('Store')['Weekly\_Sales'].sum().sort\_values(ascending=False).head(5).index

top\_5\_stores\_data = wd[wd['Store'].isin(top\_5\_stores)]

for store, wmr in top\_5\_stores\_data.groupby('Store'):

prophet\_data = wmr[['Date', 'Weekly\_Sales']]

prophet\_data.columns = ['ds', 'y']

model = Prophet()

model.fit(prophet\_data)

future = model.make\_future\_dataframe(periods=84, freq='d')

forecast = model.predict(future)

# Filter the forecast data to include only future dates

last\_date = prophet\_data['ds'].max()

future\_forecast = forecast[forecast['ds'] > last\_date]

# Plot the filtered forecast for the store

plt.plot(future\_forecast['ds'], future\_forecast['yhat'], label=f'Store {store}')

# Add title, labels and legend

plt.title('Top 5 Stores Weekly Sales Forecast')

plt.xlabel('Date')

plt.ylabel('Value')

plt.legend()

# Show the plot

plt.show()

last\_5\_stores = wd.groupby('Store')['Weekly\_Sales'].sum().sort\_values(ascending=True).head(5).index

last\_5\_stores\_data = wd[wd['Store'].isin(last\_5\_stores)]

for store, wmr in last\_5\_stores\_data.groupby('Store'):

prophet\_data = wmr[['Date', 'Weekly\_Sales']]

prophet\_data.columns = ['ds', 'y']

model = Prophet()

model.fit(prophet\_data)

future = model.make\_future\_dataframe(periods=84, freq='d')

forecast = model.predict(future)

# Filter the forecast data to include only future dates

last\_date = prophet\_data['ds'].max()

future\_forecast = forecast[forecast['ds'] > last\_date]

# Plot the filtered forecast for the store

plt.plot(future\_forecast['ds'], future\_forecast['yhat'], label=f'Store {store}')

# Add title, labels and legend

plt.title('last 5 Stores Weekly Sales Forecast')

plt.xlabel('Date')

plt.ylabel('Value')

plt.legend()

# Show the plot

plt.show()

# Assuming your data is stored in a DataFrame called `df`

store\_numbers = input('Enter store numbers separated by commas: ').split(',')

store\_numbers = [int(store) for store in store\_numbers]

stores\_data = wd[wd['Store'].isin(store\_numbers)]

for store, wmr in stores\_data.groupby('Store'):

# Plot the sales of the store

plt.plot(wmr['Date'], wmr['Weekly\_Sales'], label=f'Store {store}')

# Add title, labels and legend

plt.title('Store Sales Comparison')

plt.xlabel('Date')

plt.ylabel('Value')

plt.legend()

# Show the plot

plt.show()

# Assuming your data is stored in a DataFrame called `df`

store\_numbers = input('Enter store numbers separated by commas: ').split(',')

store\_numbers = [int(store) for store in store\_numbers]

stores\_data = wd[wd['Store'].isin(store\_numbers)]

for store, wmr in stores\_data.groupby('Store'):

prophet\_data = wmr[['Date', 'Weekly\_Sales']]

prophet\_data.columns = ['ds', 'y']

model = Prophet()

model.fit(prophet\_data)

future = model.make\_future\_dataframe(periods=84, freq='d')

forecast = model.predict(future)

# Filter the forecast data to include only future dates

last\_date = prophet\_data['ds'].max()

future\_forecast = forecast[forecast['ds'] > last\_date]

# Plot the filtered forecast for the store

plt.plot(future\_forecast['ds'], future\_forecast['yhat'], label=f'Store {store}')

# Add title, labels and legend

plt.title('Store Sales future Forecast for 12 weeks')

plt.xlabel('Date')

plt.ylabel('Value')

plt.legend()

# Show the plot

plt.show()

# Assuming your data is stored in a DataFrame called `df`

top\_stores = wd.groupby('Store')['Weekly\_Sales'].sum().sort\_values(ascending=False)

# Set the figure size

plt.figure(figsize=(10, 10))

# Create a bar plot of the total sales for the top 10 stores

top\_stores.plot(kind='bar', title='Top Stores order ')

# Add labels

plt.xlabel('Store number',fontsize=10)

plt.ylabel('Total Sales',fontsize=10)

# Show the plot

plt.show()