Title- *Forecasting of Weekly Sales for Walmart.*

Anik Samaddar

Intellipaat (*Advanced Certification Data Science & Artificial Intelligence*)

Index

Topic **Page No**

Abstract: **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.**

Feature Information

The dataset contains of 8 columns.

Feature Name	Description
Store	Store number
Date	Week of Sales
Weekly_Sales	Sales for the given store in that week
Holiday_Flag	If it is a holiday week
Temperature	Temperature on the day of the sale
Fuel_Price	Cost of the fuel in the region
CPI	Consumer Price Index
Unemployment	Unemployment Rate

Information of features

Store- This feature contains the respective wallmart store numbers. (datatype=int64)

Date- This feature contains the dates which has interval of one week. (datatype=Object)

Weekly_Sales- This Feature contains data of sales in weekly format for that respective date. (datatype=float64)

Holiday_flag- This feature gives information about the week is a holiday week or not. (datatype=int64)

Temperature- This feature gives the temperature of the respective day. (datatype=float64)

Fuel_price- This feature gives information about the fuel_price on that region. (datatype=float64)

CPI- This feature gives information about the consumer price index.

Unemployment-This feature gives information about the unemployment rate for a store region.

Import list needed for the Project

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
import numpy as np
from datetime import datetime
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
from sklearn.metrics import *
from statsmodels.tsa.arima_model import ARIMA
```

Data Visualisation and Exploration

Data Cleaning &Preprocessing :- At first the data is read and store in a variable df. There is no Null Values in the dataset and is also same conclusion holds for duplicate and spaces as well.

Date- All the features of the dataset were into int and float format apart from "Date" column

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Store         6435 non-null   int64  
 1   Date          6435 non-null   object 
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64  
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
```

The following feature is transformed into datetime format for future work.

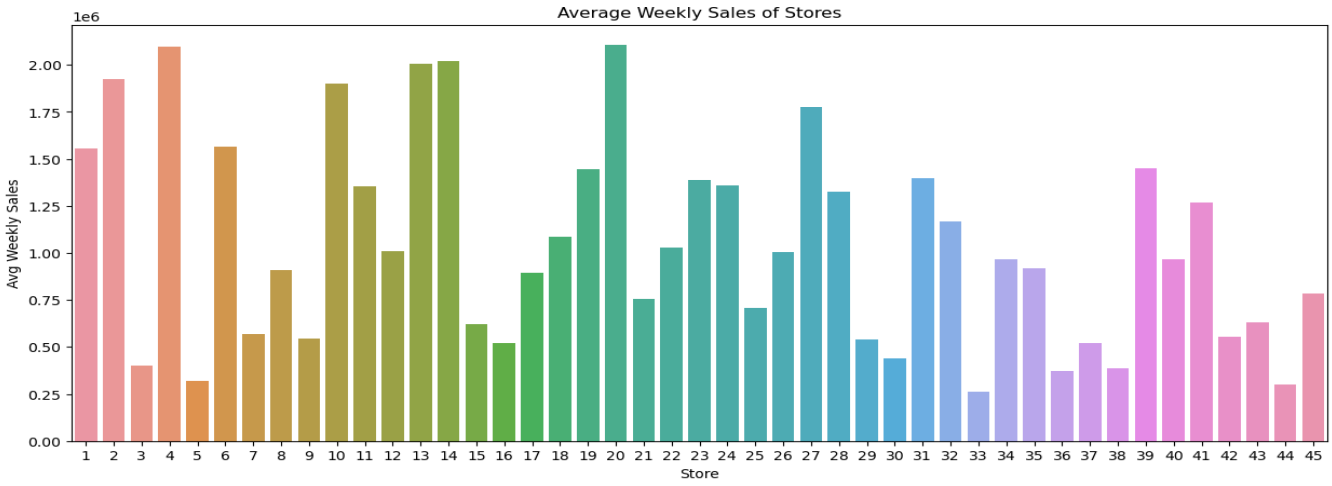
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   Store         6435 non-null   int64         
 1   Weekly_Sales  6435 non-null   float64       
 2   Holiday_Flag  6435 non-null   int64         
 3   Temperature   6435 non-null   float64       
 4   Fuel_Price    6435 non-null   float64       
 5   CPI           6435 non-null   float64       
 6   Unemployment  6435 non-null   float64       
 7   Year          6435 non-null   object        
 8   Month         6435 non-null   object        
 9   Day           6435 non-null   object        
 10  date          6435 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(5), int64(2), object(3)
memory usage: 553.1+ KB
```

Top Performing Stores:- In this section we find the top performing stores according to their historical data. The goal is to find out the stores which have higher sales than others. A group by operation is used to achieve this goal.

Below the result id shown.

Code-avg_sales=df.groupby("Store")["Weekly_Sales"].mean().round(2)

```
plt.figure(figsize=(15,6))
sns.barplot(x=avg_sales.index,y=avg_sales.values)
plt.title("Average Weekly Sales of Stores")
plt.xlabel("Store")
plt.ylabel("Avg Weekly Sales")
plt.show()
```



Top 5 best stores according to average sales are store no- 4,20,2,13,14

Worst 5 stores according to average sales are store no- 33,5,44,3,36

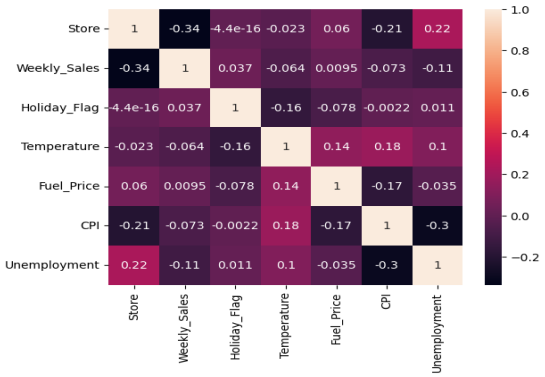
The difference between Top and Worst store

Code- avg_sales.max()-avg_sales.min()

The difference of average weekly Sales between best and worst store is 187815.1

Correlation between the features – To find the correlation among the features a heatmap is plotted. Since our target feature is "**Weekly_Sales**" the relation of that with respect to other features is to be acutely analysed.

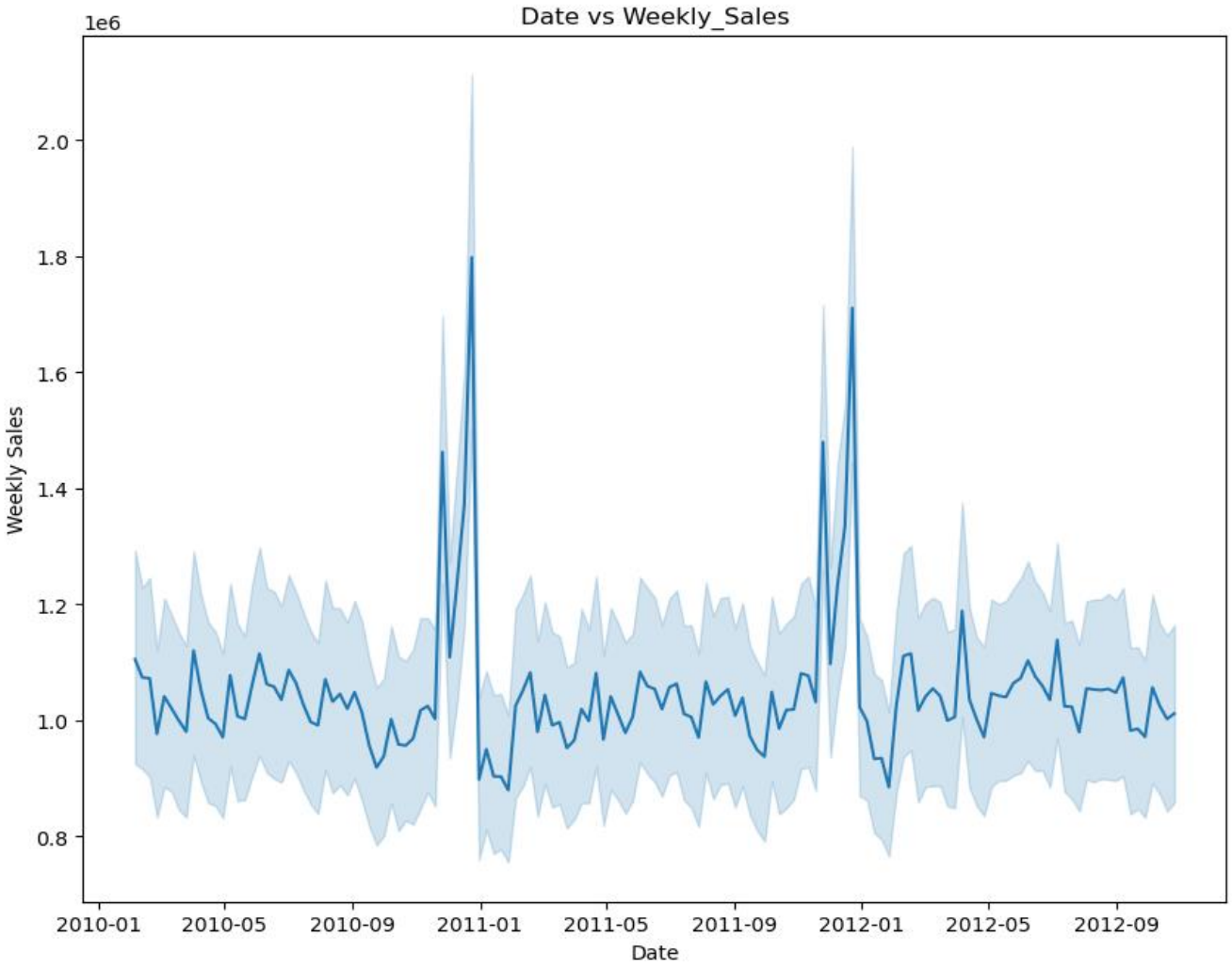
Code- sns.heatmap(df.corr(),annot=True)



Seasonal Trend of Weekly Sales :- To analyse the seasonality of the weekly sales feature a line plot has been used.

It has been analysed from the plot that during the year ending and at the time of the next year the sales values go up.

Code-
```
plt.figure(figsize=(10,8))
sns.lineplot(x="date",y="Weekly_Sales",data=df)
plt.title("Date vs Weekly_Sales")
plt.xlabel("Date")
plt.ylabel("Weekly Sales")
plt.show()
```

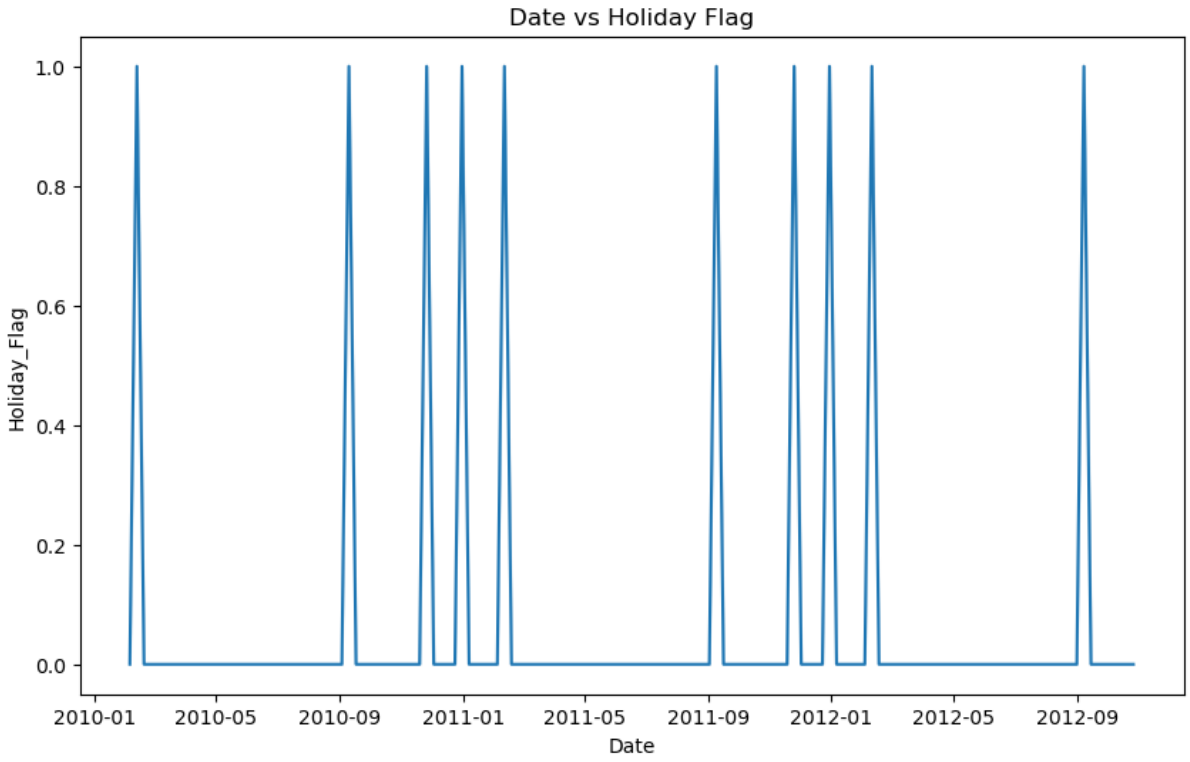


Impact of Holiday Week on Weekly Sales :- To understand the impact the impact of holiday on weekly sales a line plot similar to above has been plotted.

It is concluded from the graph that during the dec month and in the Jan month the “Holiday_flag” are high, so this time of the year due to holiday season sales have high spikes in the plot.

Code-

```
plt.figure(figsize=(10,6))
sns.lineplot(x="date",y="Holiday_Flag",data=df)
plt.title("Date vs Holiday Flag")
plt.xlabel("Date")
plt.ylabel("Holiday_Flag")
plt.show()
```



Impact on Unemployment with Weekly Sales **:-** To Analyse the impact of Unemployment on weekly sales. A group by operation is utilised to understand the relation, and the values are sorted to determine the store which has been affected the most.

A lineplot is used to visualise the result.

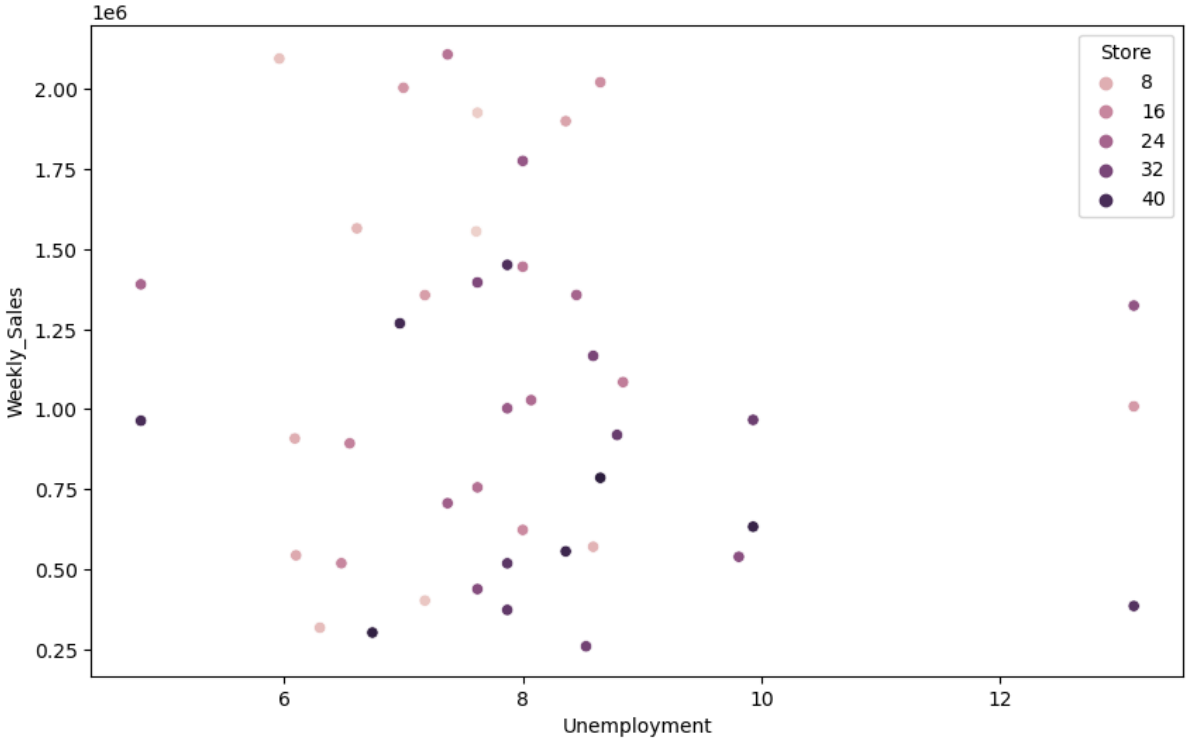
Code-

```
unemp_sales=df.groupby("Store").agg({"Weekly_Sales":"mean","Unemployment":"mean"}).round(2)
unemp_sales=unemp_sales.sort_values(by="Weekly_Sales")
plt.figure(figsize=(10,6))
```

sns.scatterplot(x="Unemployment",y="Weekly_Sales",hue="Store",data=unemp_sales)

Store	Weekly_Sales	Unemployment
33	259861.69	8.53
44	302748.87	6.74
5	318011.81	6.30
36	373511.99	7.87
38	385731.65	13.12

These are the five stores which are affected by unemployment.



The plot shows that the data points are scattered implying the fact that there is no good amount of relation between two features for the whole data.

But the result is also similar when EDA is done on the data frame of store no -33 as the sales were lowest here with respect to the unemployment in the above code.

A line plot is utilised to analyse the visualisation. As seen that in Jan Unemployment rate was high still the sales were high and for unemployment rate less than that the sales dipped and when the unemployment rate was lowest again the sales spiked.

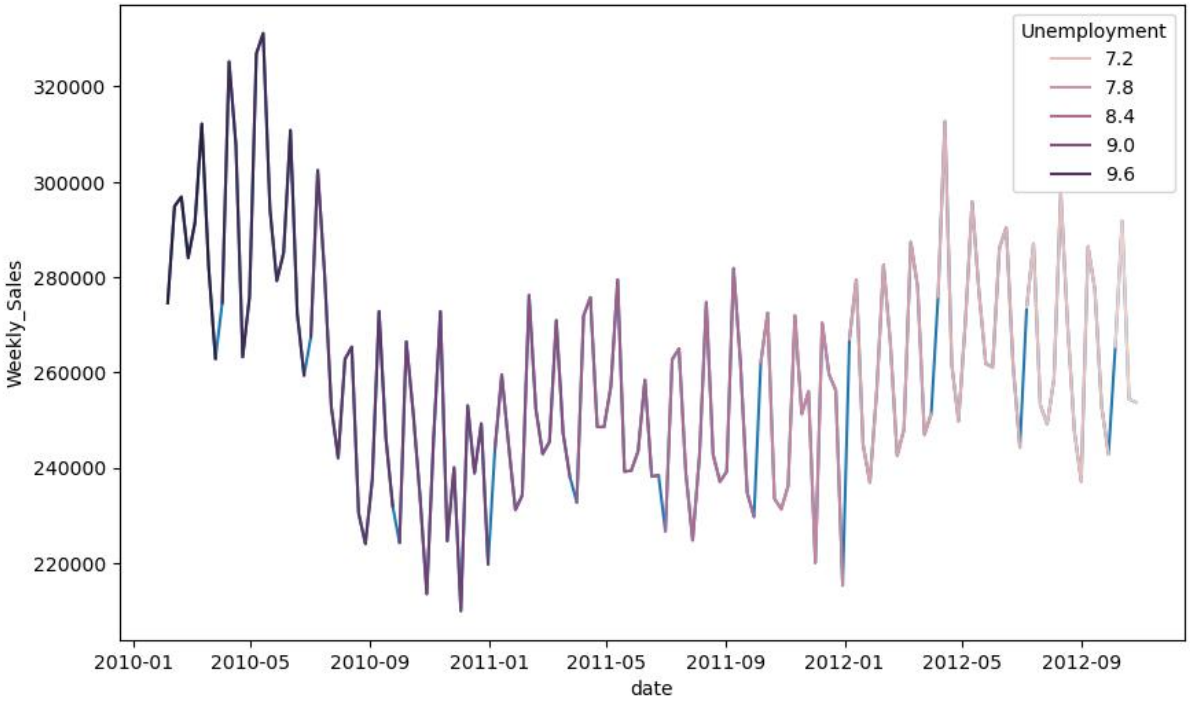
So, it can be concluded that the unemployment rate has no relation with sales for this data.

Code- dv=df[df["Store"]==33]

plt.figure(figsize=(10,6))

sns.lineplot(x="date",y="Weekly_Sales",data=dv)

sns.lineplot(x="date",y="Weekly_Sales",hue="Unemployment",data=dv)



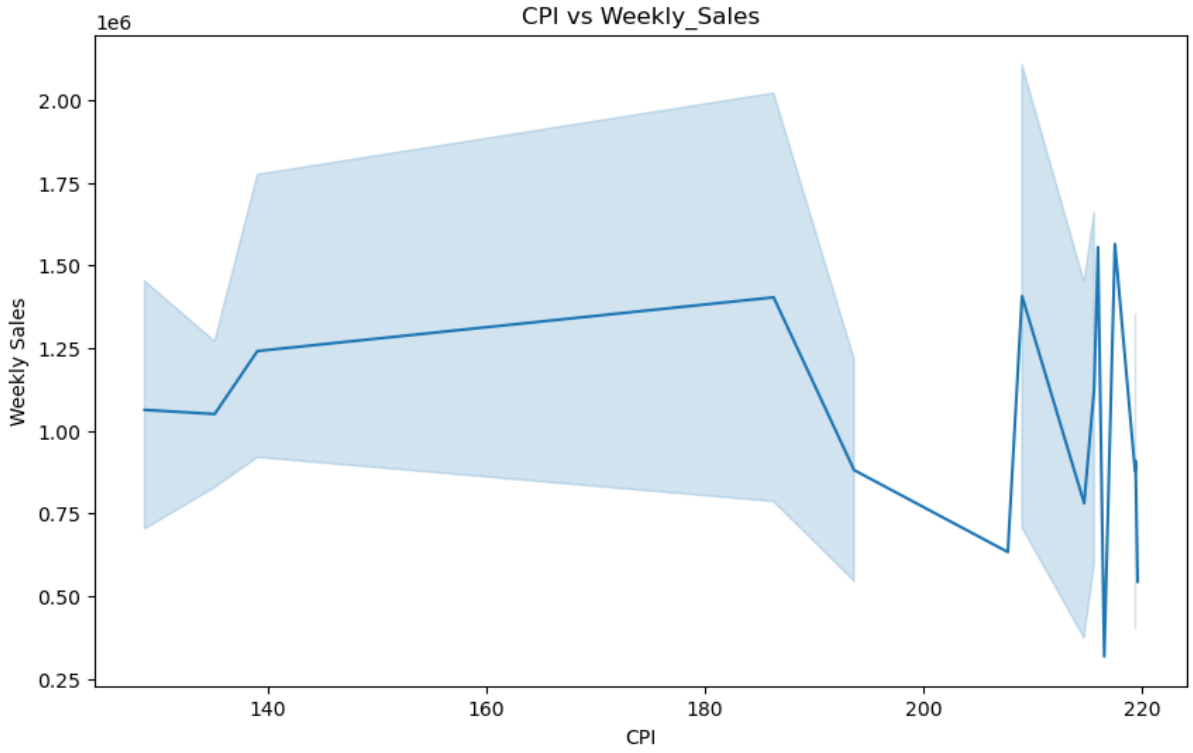
Impact of CPI on Weekly Sales :- For analysing the following relation the same group by method is used then the result is visualised with a line plot.

Code-CPI_SALES=df.groupby("Store").agg({"Weekly_Sales":"mean","CPI":"mean"}).round(2)

Store	Weekly_Sales	CPI
33	259861.69	128.68
44	302748.87	128.68
5	318011.81	216.57
36	373511.99	214.73
38	385731.65	128.68

Store-33 affects the most as it the worst performing store

```
plt.figure(figsize=(10,6))
sns.lineplot(x="CPI",y="Weekly_Sales",data=CPI_SALES)
plt.title("CPI vs Weekly_Sales")
plt.xlabel("CPI")
plt.ylabel("Weekly Sales")
plt.show()
```



According to the visualisation there is no relation between these two features as weekly sales fluctuates with respect to the Cpi.

Impact Of Temperature on Weekly sales :- A line plot is utilised to see the visualisation of temperature and sales.

During the temperature range 20-40 degree Celsius there is some spikes in sales values. But in conclusion it can be said that the temperature as no effect on sales.

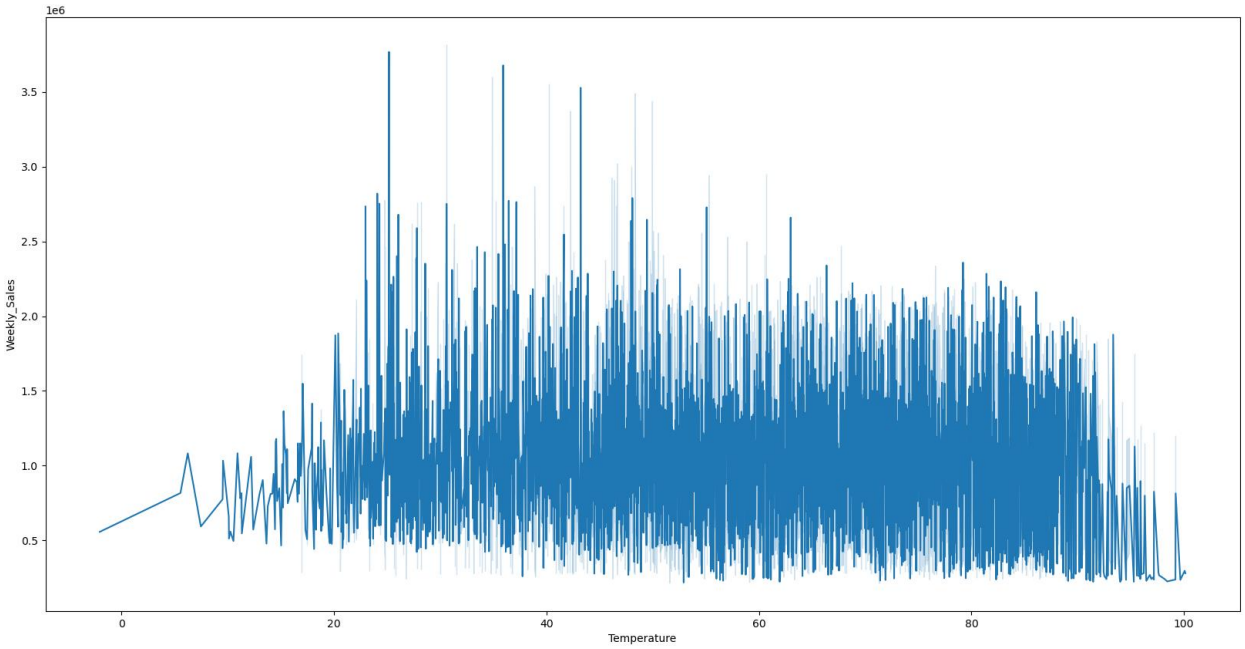
Code - plt.figure(figsize=(20,10))

sns.lineplot(x="Temperature",y="Weekly_Sales",data=df)

plt.xlabel("Temperature")

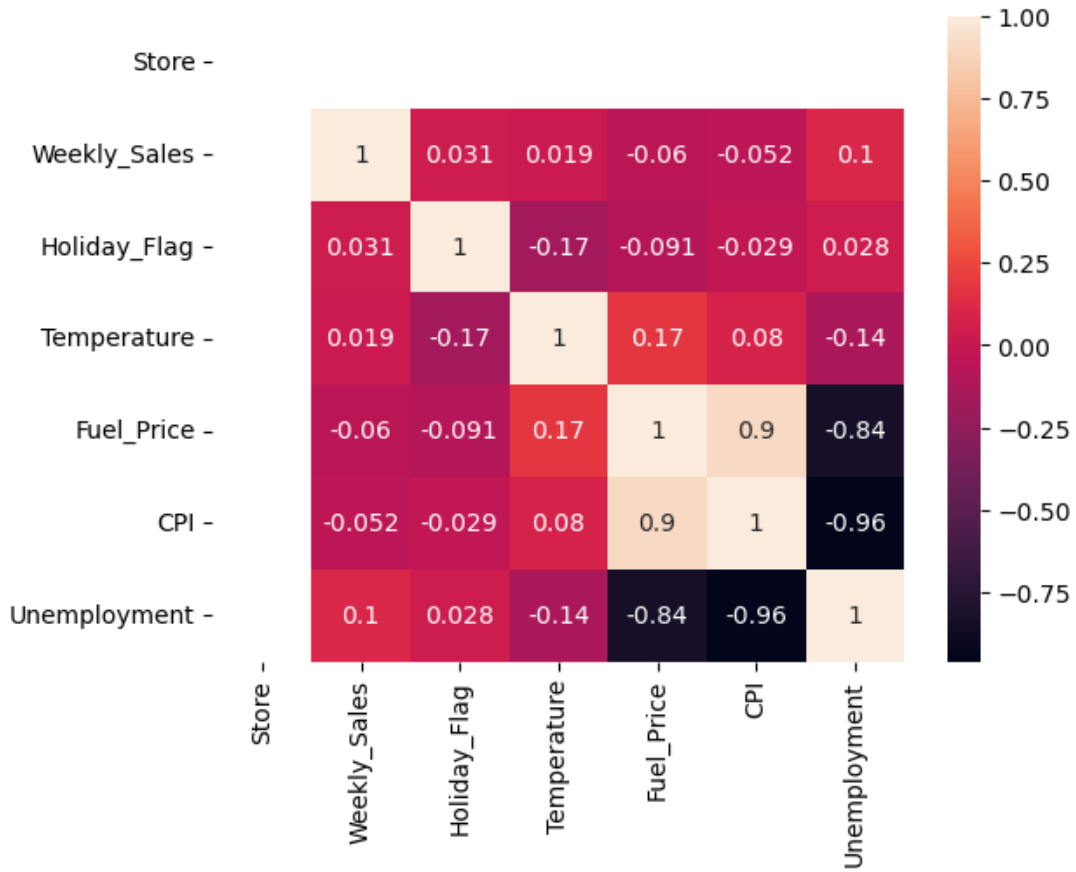
plt.ylabel("Weekly_Sales")

plt.show()



EDA on the Worst Performing Store :- To understand the reasons for the bad performance in terms of sales , The data regarding the store-33 is analysed. The data is read in dv dataframe.

1.Heat map is plotted to see the corelation.

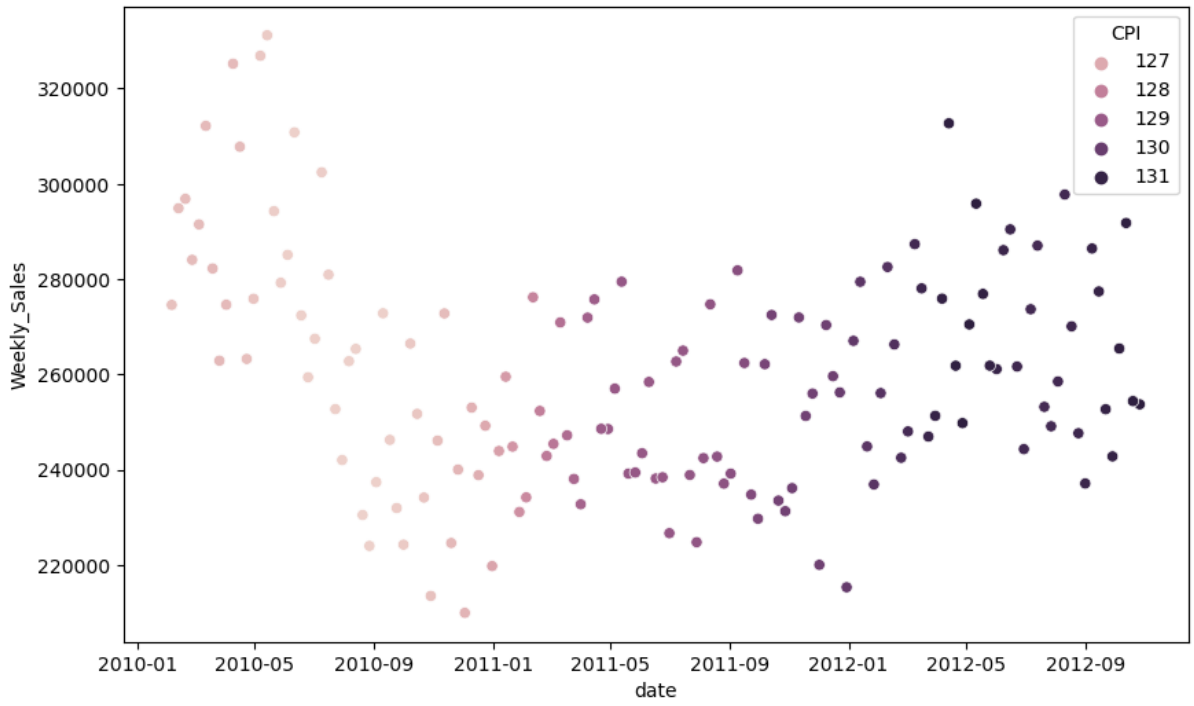


2. Now from the above heatmap it has been observed that the Impact of CPI and the Fuel Price should be analysed for each store.

A Scatter plot is used to understand the relation of these two features with respect to each store

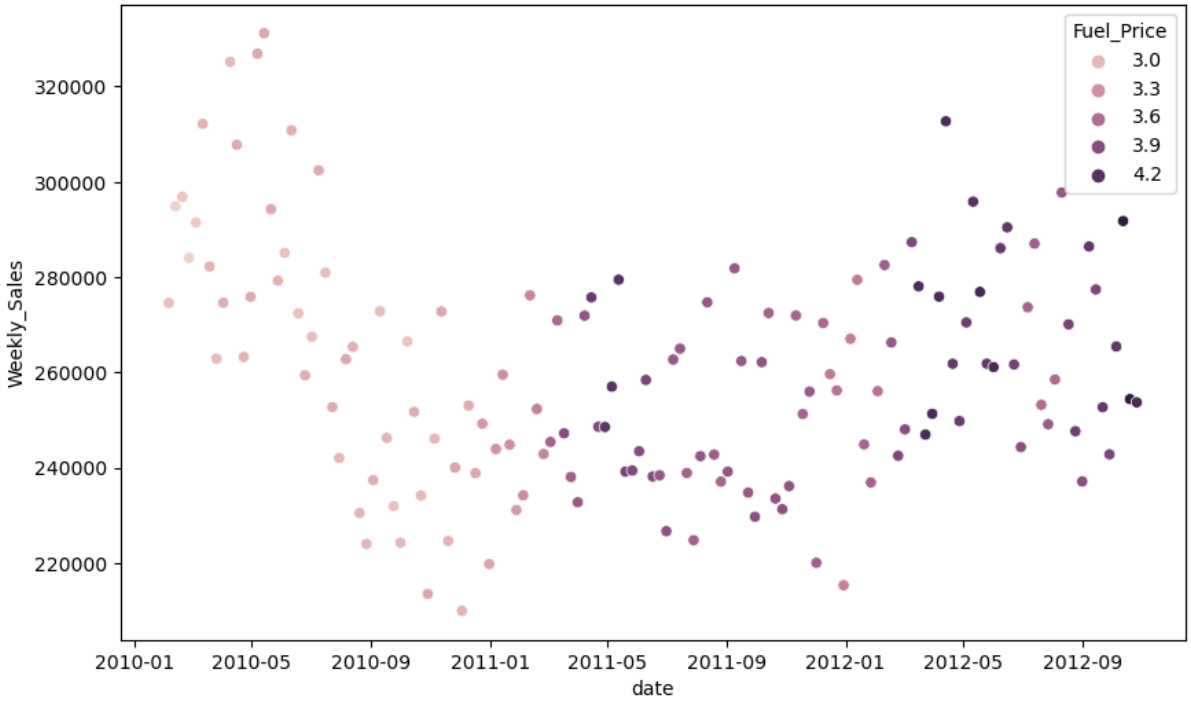
Impact of CPI :- It has been observed that on the January month when the sales were high at first the CPI values were minimum, so this could be a reason for high sales in this month and after that the more the CPI values are increased there is seen the sales values decrease.

Code-
```
dv=df[df["Store"]==33]

plt.figure(figsize=(10,6))

sns.scatterplot(x="date",y="Weekly_Sales",hue="Fuel_Price",data=dv)
```



Impact of Fuel Price :- The analysing results are same as off CPI here in January month the fuel price was very low and the data shows a spike here and for the higher fuel price there is some dips in sales value.

Code-
```
plt.figure(figsize=(10,6))

sns.scatterplot(x="date",y="Weekly_Sales",hue="Fuel_Price",data=dv)
```



So, it can be concluded that CPI and Fuel Price play an import role for less performing stores

Model Building :-In case of Model Building two approach has been used,

1- **Manual Forecasting**-In this approach a dataframe is made for the targeted store, which has to be forecasted, then different time series methods applied and the result is analysed.

2- **Automatic Forecasting –** In this method two function is made first one checks that the data is static or not and how many times the differencing should be needed with acf and pacf plots , and the second function will give the MSE and RMSE of each method which will show which method is applicable for that store

Manual forecasting for store no-1:-

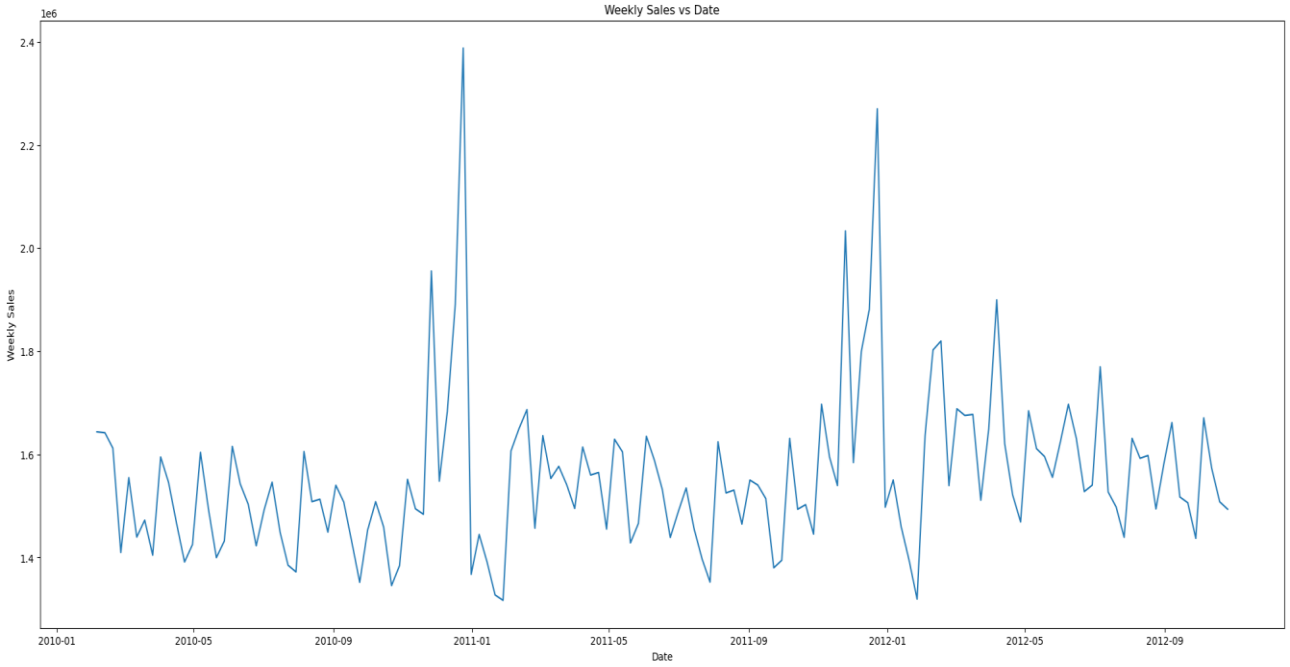
Data Preprocessing

A data frame is created of store no -1 then only dates and weekly sales feature is extracted and data column is used as an index.

code- dl=df[df["Store"]==1]

dl=dl[["date","Weekly_Sales"]]

```
dl.drop("date", axis=1, inplace=True)
plt.figure(figsize=(25,10))
plt.plot(dl)
plt.xlabel("Date")
plt.ylabel("Weekly Sales")
plt.title("Weekly Sales vs Date")
plt.show()
```



Augmented Dickyfuller Test

If the data is changing with respect to time that is said to be dynamic, for time series forecasting it's mandatory to make data static for applying the model and forecasting. Augmented DickyFuller test is used to see the data is static or not. I

Here a function is made which will check the data is static or not, if not differencing should be done to make it static.

Code-

```
def adf_test(series):
    result=adfuller(series)
    pval = result[1]
    print(pval)

    if pval < 0.05:
        print("The Data is Static")
    else:
        print("The Data is Not Static")
```

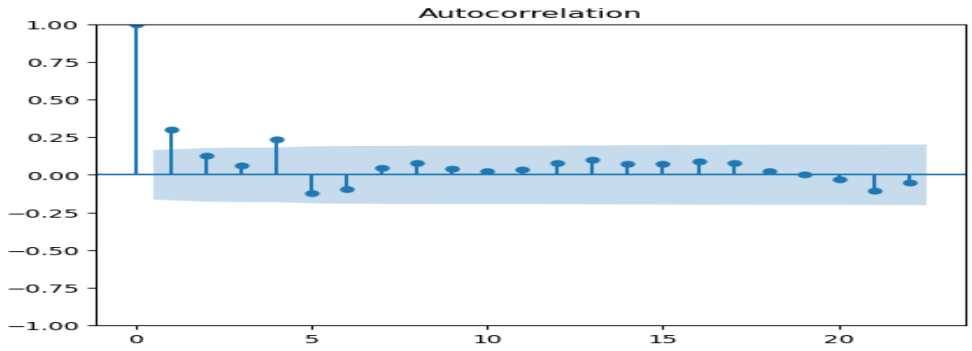
Result- 1.3877788330759535e-05
The Data is Static

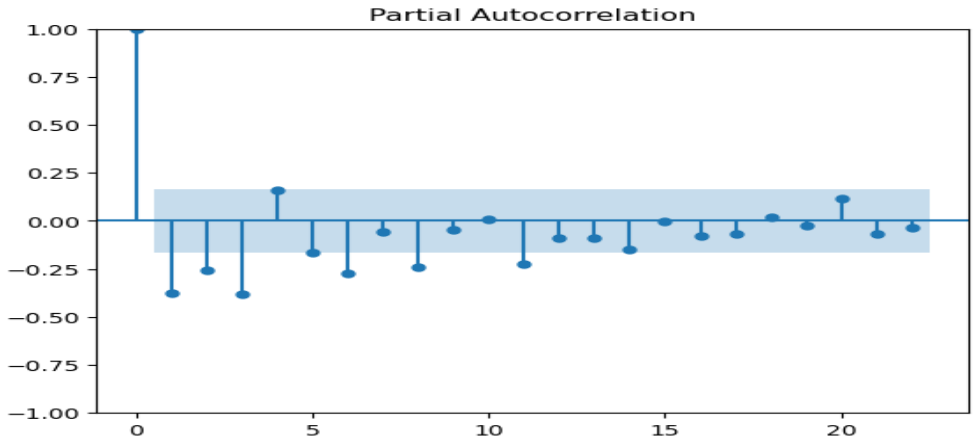
ACF & PACF plots

ACF and PACF plot is visualised and from the plots p(auto regressive lags), d(differencing) and q(moving average) values can be found.

Code-acf=plot_acf(dl["Weekly_Sales"])

Pacf=plot_pacf(dl["Weekly_Sales"])





Train, Test data , Model Fitting :- Splitting of train and test data is done here indexing technique is used and different model is fitted to the data. Here Arima and Sarima is used for forecasting

Code- train_size = int(0.8 * len(dl["Weekly_Sales"]))

train_data =dl["Weekly_Sales"][:train_size]

test_data = dl["Weekly_Sales"][train_size:]

Forecasting and Error :-The forecasting values are seen and plotted to see the result, Here in this data since there is seasonality the sarima model and Holt Winter model works good has less error value than the arima model. Here mean squared and Root mean squared values are used to compare the results.

From the acf and pacf graph p,d,q is chosen as 5,0,4 and due to it has sale hike in year ending months seasonality is taken as s=52

The PMDarima, Arima , Sarima and Holt Winter model is applied and the error values are-

Model	MSE	RMSE
Auto_Arima	6.356485e+09	79727.566545
ARIMA	6.531520e+09	80817.817822
SARIMAX	5.186982e+09	72020.707888
HoltWinters	5.648063e+09	75153.599139

Conclusion It has been concluded from the above result that SARIMA and Holt Winter model giving lower error MSE and RMSE values than Arima and Auto Arima model for this following dataset.

Automatic Forecasting:- In this approach two function is created,

Analyze Static Function

In this approach two function is made, the first one checks the data is static or not with Augmented DickyFuller test and gives the differencing order with ACF and PACF plots.

```
def analyze_static(store_id, df, diffs=3):
    num=0
    stored_data = df[df["Store"] == store_id]
    result = adfuller(stored_data["Weekly_Sales"])
    print(f"P-value is {result[1]}")
    if result[1] < 0.05:
        print("The data is stationary")
        d = 0
        D=0
        diffed_sales = stored_data["Weekly_Sales"]
    else:
        print("Differencing needed")
        d = num
        D = num

        while result[1] > 0.05 and num < diffs:
            diffed_sales = stored_data["Weekly_Sales"].diff()
            diffed_sales.dropna(inplace=True)
            result = adfuller(diffed_sales)
            num += 1
            print(f"The number of differencing is {num}")


        plot_acf(diffed_sales)
        plot_pacf(diffed_sales)


    return diffed_sales, d, D
```

Analyze Store Function

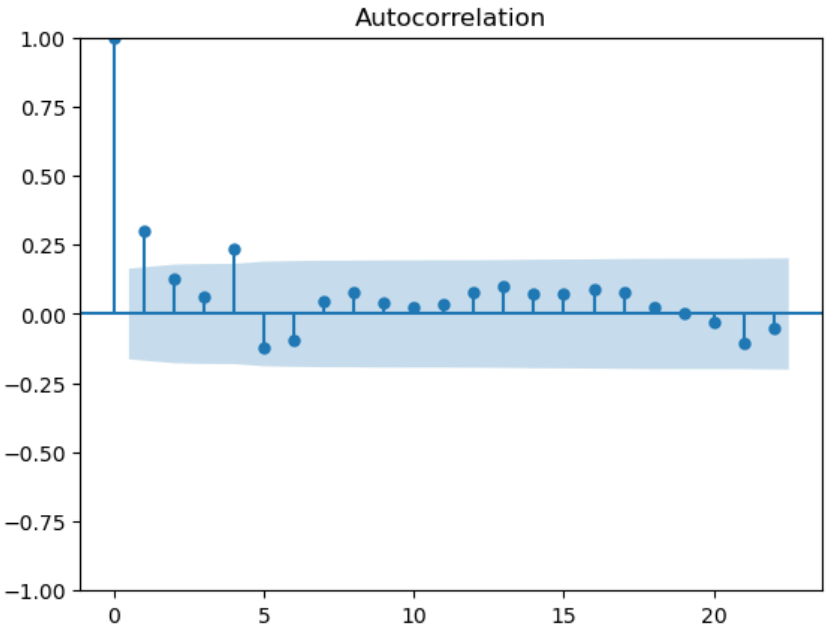
The data from analyzed static function is passed to here, and p,d,q values are chosen from the visualisation of analyze static function,

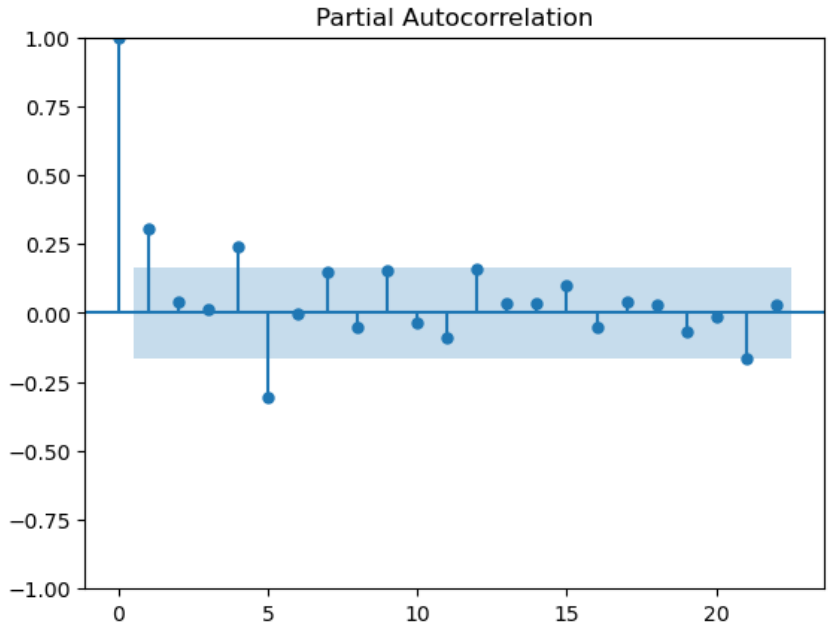
This function provides the MSE and RMSE values for the same store which was passed to analyze static function. As seen in the manual forecasting Sarima and Holt Winter is performing well for this data, here Sarima and HoltWinter MSE and RMSE values are calculated.

```
def analyze_store(store_id, df, p=5, q=4, P=5, Q=4, S=52):
    stored_data = df[df["Store"] == store_id]
    diffed_sales, d,D = analyze_static(store_id, df, diffs=3)

    train_size = int(0.8 * len(diffed_sales))
    train_data = diffed_sales[:train_size]
    test_data = diffed_sales[train_size:]

    model_holt = ExponentialSmoothing(train_data, seasonal='add', seasonal_periods=S,trend="add")
    model_holt_fit = model_holt.fit()
    predict_hw = model_holt_fit.forecast(len(test_data))

    mse_hw = mean_squared_error(test_data, predict_hw)
    rmse_hw = mse_hw ** 0.5
    print(f"The MSE for Holt Winter for {store_id}: {mse_hw}")
    print(f"The RMSE for Holt Winter for {store_id}: {rmse_hw}")


    # Perform SARIMA modeling
    model_sarimax = SARIMAX(train_data, order=(p, d, q), seasonal_order=(P, D, Q, S))
    model_sarimax_fit = model_sarimax.fit()
    pred_sarimax = model_sarimax_fit.forecast(len(test_data))

    mse_sarimax = mean_squared_error(test_data, pred_sarimax)
    rmse_sarimax = mse_sarimax ** 0.5
    print(f"The MSE for SARIMA for {store_id}: {mse_sarimax}")
    print(f"The RMSE for SARIMA for {store_id}: {rmse_sarimax}")
```

Results:- For Store No -1 the results are shown.

```
P-value is 1.3877788330759535e-05
The data is stationary
```





Conclusion:- It is seen from the result that the error values for Sarima and Holt-Winter is same as the manual forecasting part**.**
So a selected store id can be passed through these two function to see the data is static or not and also to understand the errors, from this an idea will generate that which model suits best for the data then forecasting by that model can be done manually to fine tune it even more by bartering the hyper parameters.

Future Scope:-Different models can be also incorporated to the automation part and referring the errors the best model could be found.

Also with the help of looping the different error values for each p,d,q values can be compared to find the optimal numbers, this process can eliminate the visual inspection of ACF and PACF plot part.

Reference:-

1.Time Series Forecasting using SARIMA (Python) Can Ozdogar

(https://medium.com/@ozdogar/time-series-forecasting-using-sarima-python-8db28f1d8cfc)

2. https://en.wikipedia.org/wiki/Time_series

3. https://en.wikipedia.org/wiki/Exponential_smoothing

4. https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average

5. https://en.wikipedia.org/wiki/Mean_squared_error

6. https://en.wikipedia.org/wiki/Augmented_Dickey%E2%80%93Fuller_test