WALMART SALES FORECASTING AND ANALYSIS

Table of Contents

1. Problem Statement

Predicting future sales for a company is one of the most important aspects of strategic planning. We want to analyze in depth how internal and external factors of one of the biggest companies in the US can affect their 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 time series analysis.

2. Project Objective

We want to analyze in depth how internal and external factors of one of the biggest companies in the US can affect their Weekly Sales in the future. Utilizing predictive modeling techniques, we want to forecast the sales for each store for the next 12 weeks.

3. Dataset Overview

This data set contains information about the stores, date, weekly_sales, temperature, unemployment, CPI, Holiday_Flag, and Fuel Price. The data collected ranges from 2010 to 2012, where 45 Walmart stores across the country were included in this analysis.

Stores:

Store: The store number. Range from 1–45.

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.

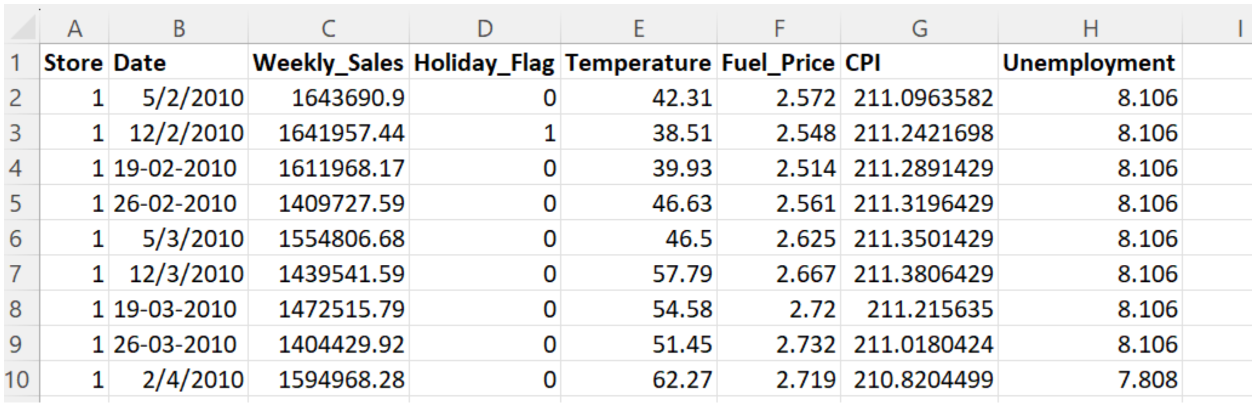
Unemployment: The unemployment rate during that week in the region of the store.

Sales:

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

Weekly_Sales: The sales recorded during that Week.

Holiday_Flag: A Boolean value representing a holiday week or not.

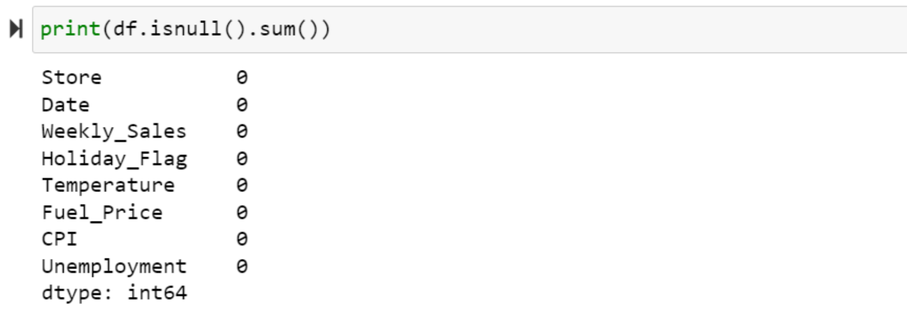


	A	B	C	D	E	F	G	H	I
1	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	
2	1	5/2/2010	1643690.9	0	42.31	2.572	211.0963582	8.106	
3	1	12/2/2010	1641957.44	1	38.51	2.548	211.2421698	8.106	
4	1	19-02-2010	1611968.17	0	39.93	2.514	211.2891429	8.106	
5	1	26-02-2010	1409727.59	0	46.63	2.561	211.3196429	8.106	
6	1	5/3/2010	1554806.68	0	46.5	2.625	211.3501429	8.106	
7	1	12/3/2010	1439541.59	0	57.79	2.667	211.3806429	8.106	
8	1	19-03-2010	1472515.79	0	54.58	2.72	211.215635	8.106	
9	1	26-03-2010	1404429.92	0	51.45	2.732	211.0180424	8.106	
10	1	2/4/2010	1594968.28	0	62.27	2.719	210.8204499	7.808	

4. <u>Data Preprocessing</u>

We required some important attributes for our analysis. We checked for the missing values or null values.

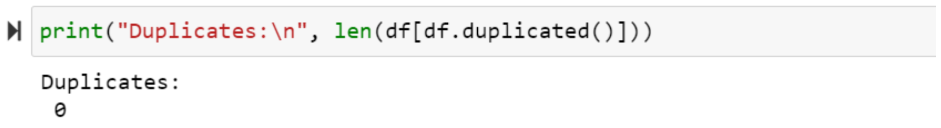
1. Checking for null values



```
print(df.isnull().sum())
```

```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
dtype: int64
```

2. Checking for duplicates



```
print("Duplicates:\n", len(df[df.duplicated()]))
```

```
Duplicates:
 0
```

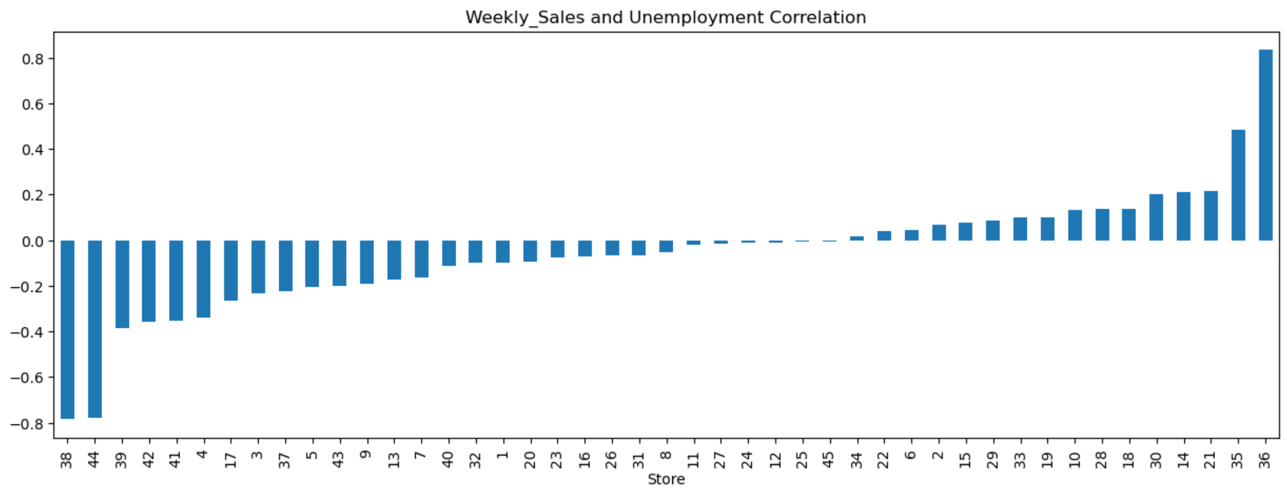
5. Exploratory Data Analysis

We initialized our dataset as a Panda Data Frame and created visualization graphs using Matplotlib package. Following are the variation graphs for each parameter:

A) Weekly Sales Vs Unemployment Rate

```
print('''\nQ.1\na) If the weekly sales are affected by the unemployment rate,
if yes - which stores are suffering the most?''')
df_grp_store = df.groupby("Store")
corr_ws_u = df_grp_store.apply(lambda group: group["Weekly_Sales"].corr(group["Unemployment"]))
corr_ws_u_asc = corr_ws_u.sort_values(ascending=True)
mas = [i for i in corr_ws_u_asc.head(3).index]

fig1a, ax1a = plt.subplots(figsize=(15, 5))
ax1a = corr_ws_u_asc.plot.bar()
ax1a.set_title("Weekly_Sales and Unemployment Correlation")
plt.show()
```

```
print(f'''{len(corr_ws_u_asc[corr_ws_u_asc > 0])} stores show +ve correlation:
{[i for i in corr_ws_u_asc[corr_ws_u_asc > 0].index]}.
{len(corr_ws_u_asc[corr_ws_u_asc < 0])} stores show -ve correlation:
{[i for i in corr_ws_u_asc[corr_ws_u_asc < 0].index]}.
The 3 most affected stores suffering from increasing unemployment are {mas}.''')
```

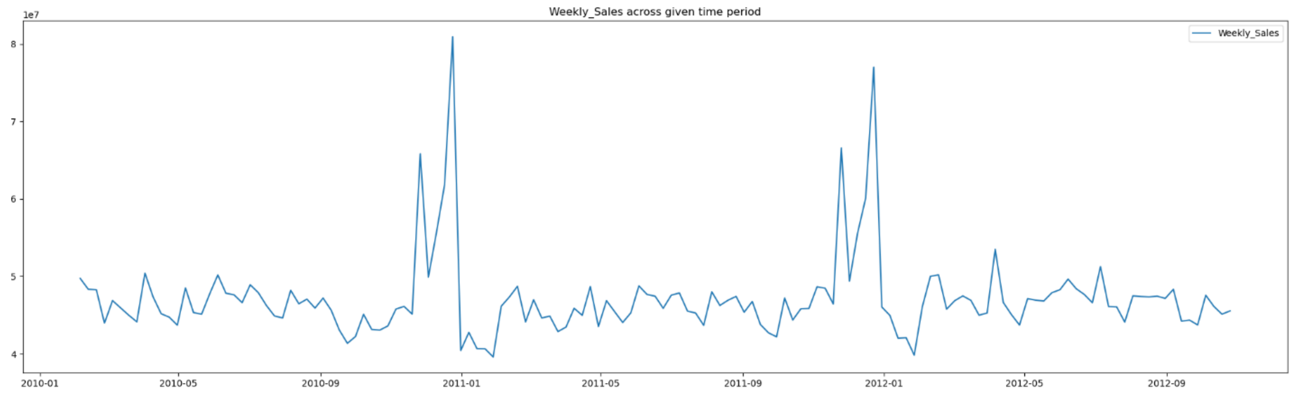


Conclusion:

1) The weekly sales of each store is effected by the Unemployment Rate
2) The most affected stores due to the employment rate are Stores 38, 44, 39.

B) Weekly Sales Vs Time Period

```python
print('''\nb) If the weekly sales show a seasonal trend, when and what could be
the reason?''')
df_grp_date = df.groupby("Date")
agg_weekly_sales = df_grp_date.agg(Weekly_Sales = ("Weekly_Sales", sum))

fig1b, ax1b = plt.subplots(figsize=(25, 7))
ax1b.plot(agg_weekly_sales.index, agg_weekly_sales["Weekly_Sales"], label="Weekly_Sales")
ax1b.set_title("Weekly_Sales across given time period")
ax1b.legend()
plt.show()
```



Conclusion:

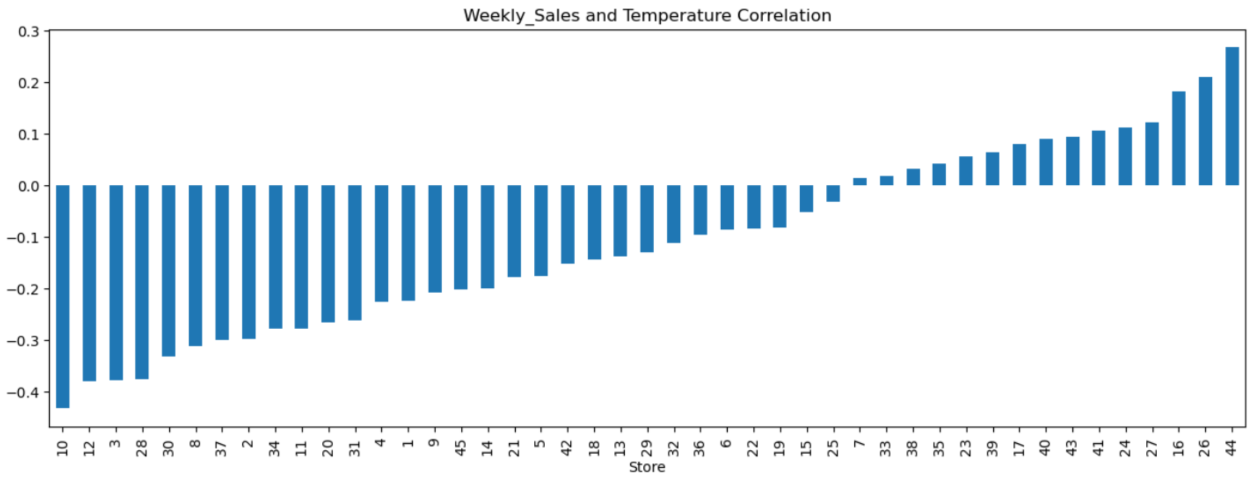
1) The weekly sales have a seasonal trend as shown in the graph.
2) There is a weekly sales rise in the months of Oct, Nov and reach its peak in Dec, then drops.
3) The reason can be due to - Festive season of Diwali, Christmas, and related promotional offers.

C) Weekly Sales Vs Temperature

```python
print('''\nc) Does temperature affect the weekly sales in any manner?''')
corr_ws_t = df_grp_store.apply(lambda group: group["Weekly_Sales"].corr(group["Temperature"]))
corr_ws_t_asc = corr_ws_t.sort_values(ascending=True)

fig1c, ax1c = plt.subplots(figsize=(15,5))
ax1c = corr_ws_t_asc.plot.bar()
ax1c.set_title("Weekly_Sales and Temperature Correlation")
plt.show()
```

```python
print(f'''{len(corr_ws_t_asc[corr_ws_t_asc > 0])} stores show +ve correlation:
{[i for i in corr_ws_t_asc[corr_ws_t_asc > 0].index]}.
{len(corr_ws_t_asc[corr_ws_t_asc < 0])} stores show -ve correlation:
{[i for i in corr_ws_t_asc[corr_ws_t_asc < 0].index]}.''')
```

```
15 stores show +ve correlation:
[7, 33, 38, 35, 23, 39, 17, 40, 43, 41, 24, 27, 16, 26, 44].
30 stores show -ve correlation:
[10, 12, 3, 28, 30, 8, 37, 2, 34, 11, 20, 31, 4, 1, 9, 45, 14, 21, 5, 42, 18, 13, 29, 32, 36, 6, 22, 19, 15, 25].
```

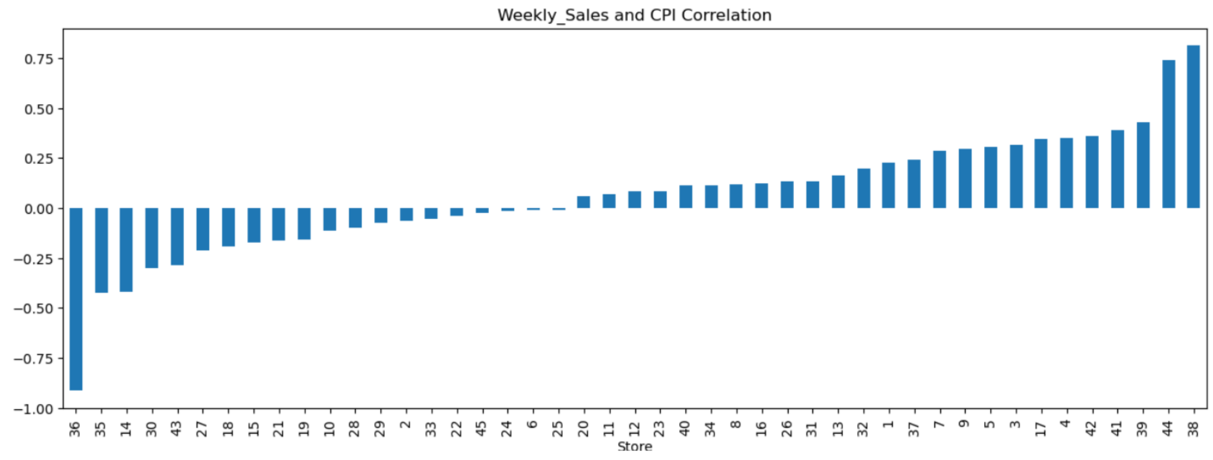


Conclusion:

1) The Temperature has an impact on the Weekly Sales as shown in the graph above.
2) Out of 45 stores, 15 stores show positive correlation, and 30 stores show negative correlation.

D) Weekly Sales Vs Consumer Price Index (CPI)

```
print('''\nd) How is the Consumer Price index affecting the weekly sales of
various stores?''')
corr_ws_c = df_grp_store.apply(lambda group: group["Weekly_Sales"].corr(group["CPI"]))
corr_ws_c_asc = corr_ws_c.sort_values(ascending=True)

fig1d, ax1d = plt.subplots(figsize=(15,5))
ax1d = corr_ws_c_asc.plot.bar()
ax1d.set_title("Weekly_Sales and CPI Correlation")
plt.show()
```



Conclusion:

1) The Consumer price index shows an impact over the Weekly Sales
2) The store 36, 35 and 14 show the negative impact on the weekly Sales and the stores 39, 44 and 38 shows the positive impact on the weekly sales.

E) Top Performing Stores according to the Historical Data

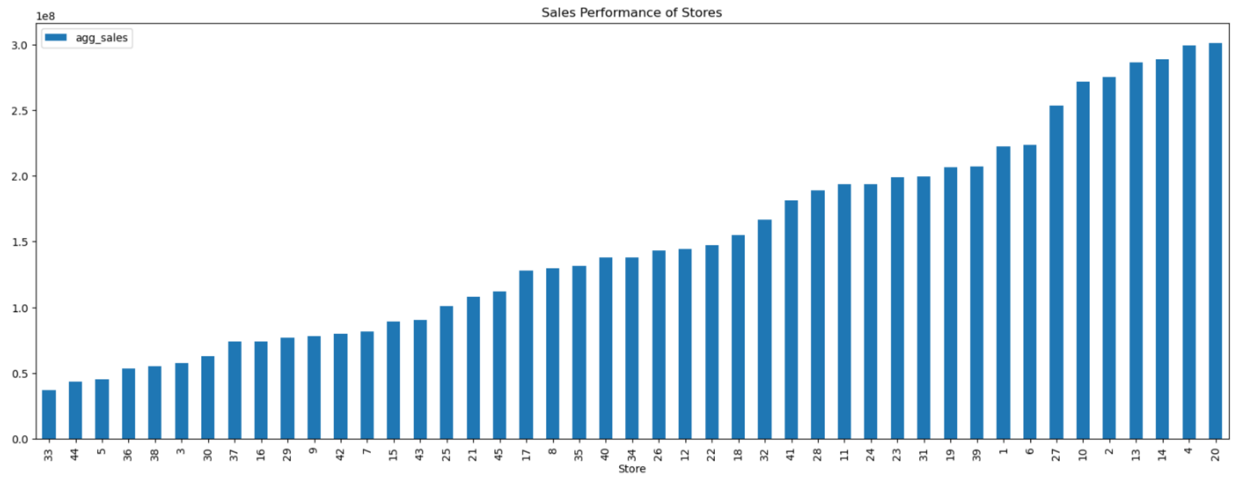
```python
print('''\ne) Top performing stores according to the historical data.''')
agg_store_sales = df_grp_store.agg(agg_sales = ("Weekly_Sales", sum))
agg_store_sales_asc = agg_store_sales.sort_values(by="agg_sales", ascending=True)
mps = [i for i in agg_store_sales_asc.tail(10).index]

ax1e = agg_store_sales_asc.plot.bar(figsize=(20,7))
ax1e.set_title("Sales Performance of Stores")
plt.show()
```

```python
print(f'''Top 10 performing stores in the decreasing order of performance:
{mps[::-1]}''')
```

```
Top 10 performing stores in the decreasing order of performance:
[20, 4, 14, 13, 2, 10, 27, 6, 1, 39]
```

```python
print('''\nf) The worst performing store, and how significant is the difference
between the highest and lowest performing stores.''')
hps = agg_store_sales_asc.tail(1).index.item()
wps = agg_store_sales_asc.head(1).index.item()
print(f"The worst performing store: Store {wps}.")
diff_hl = round(agg_store_sales_asc["agg_sales"].max() - agg_store_sales_asc["agg_sales"].min())
print(f"Difference between the highest and lowest performing store: {diff_hl}")
```

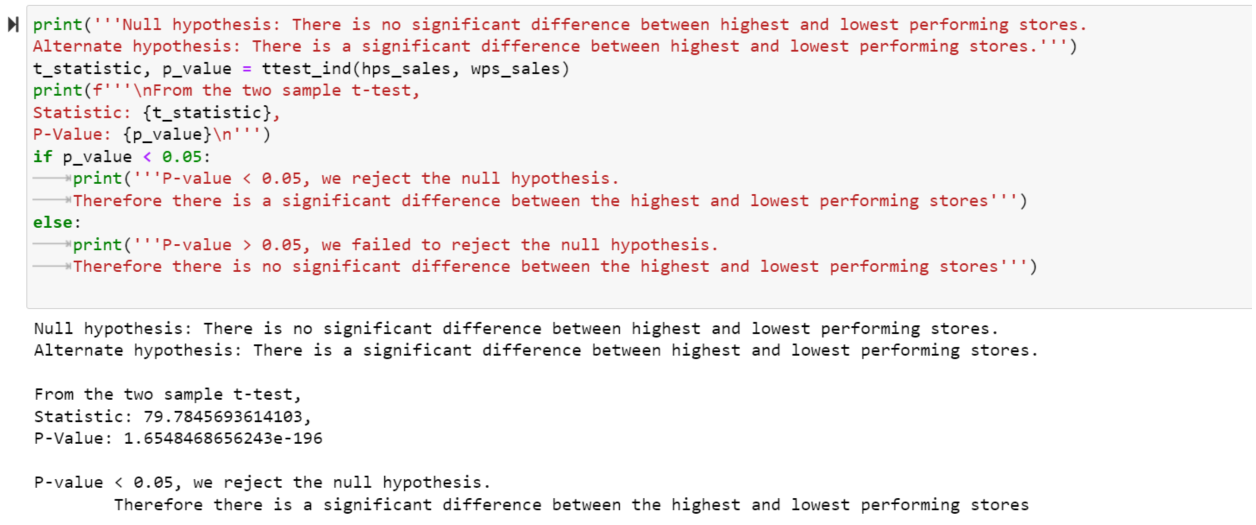
```
f) The worst performing store, and how significant is the difference
between the highest and lowest performing stores.
The worst performing store: Store 33.
Difference between the highest and lowest performing store: 264237570
```



Conclusion:

1) The store 20, 4 and 14 are the highest performing stores according to the Historical data.
2) Store '33' is the worst performing store and the difference between the lowest and highest performing stores sales is USD 264237570.

We performed the two-sample T-test to determine, if there's a significant difference between highest and lowest performing stores.

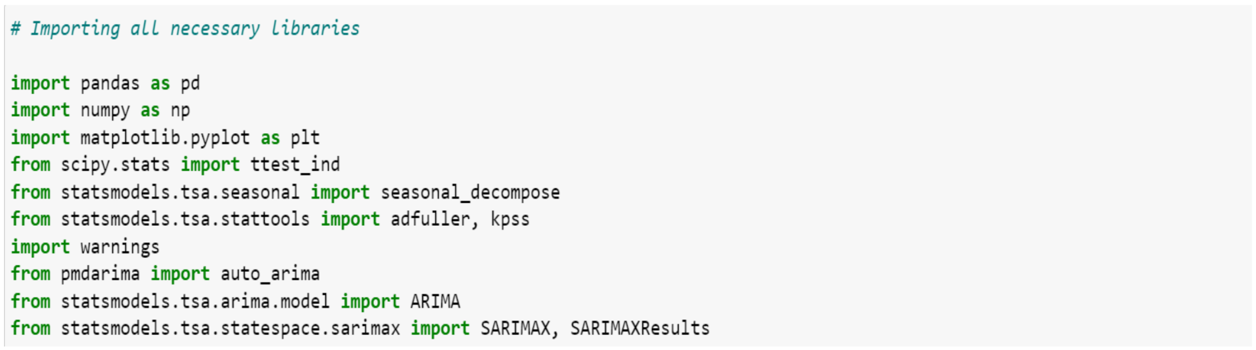


```
print('''Null hypothesis: There is no significant difference between highest and lowest performing stores.
Alternate hypothesis: There is a significant difference between highest and lowest performing stores.''')
t_statistic, p_value = ttest_ind(hps_sales, wps_sales)
print(f'''\nFrom the two sample t-test,
Statistic: {t_statistic},
P-Value: {p_value}\n''')
if p_value < 0.05:
    print('''P-value < 0.05, we reject the null hypothesis.
    Therefore there is a significant difference between the highest and lowest performing stores''')
else:
    print('''P-value > 0.05, we failed to reject the null hypothesis.
    Therefore there is no significant difference between the highest and lowest performing stores''')
```

```
Null hypothesis: There is no significant difference between highest and lowest performing stores.
Alternate hypothesis: There is a significant difference between highest and lowest performing stores.

From the two sample t-test,
Statistic: 79.7845693614103,
P-Value: 1.6548468656243e-196

P-value < 0.05, we reject the null hypothesis.
        Therefore there is a significant difference between the highest and lowest performing stores
```

6. <u>Choosing the Algorithm</u>

We will perform a detailed Time series analysis to gather useful insights and forecast the future sales for each store. We will use one of the most used methods for time-series forecasting, known as ARIMA and SARIMAX.



```
# Importing all necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import ttest_ind
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller, kpss
import warnings
from pmdarima import auto_arima
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX, SARIMAXResults
```

Based on our analysis, we have observed that the weekly sales have a seasonal trend. Our goal is to achieve stationary data that is not subject to seasonality. That means the statistical properties of the data series, such as mean, variance and autocorrelation, are constant over time.

There are 2 ways to test the stationarity of time series:

a) Rolling Mean: A rolling analysis of a time series model is often used to assess the model's stability over time. The window is rolled (slid across the data) on a weekly basis, in which the average is taken on a weekly basis. Rolling Statistics is a visualization test, where we can compare the original data with the rolled data and check if the data is stationary or not.

b) Augmented Dicky -Fuller test: This test provides us the statistical data such as p-value to understand whether we can reject the null hypothesis. The null hypothesis is that data is not stationary, and the alternative hypothesis says that data is stationary. If the p-value is less than the critical value (say 0.5), we will reject the null hypothesis and say that data is stationary.

7. Motivation and Reasons for Choosing the Algorithm

We used different predictive modelling techniques such as ARIMA and SARIMAX as we want to forecast the future sales for each store. The ARIMA model predicts a given time series based on its own past values.

a) Autoregressive Integrated Moving Average (ARIMA)

We have used one of the most used methods for time-series forecasting, known as ARIMA.

ARIMA models are denoted by ARIMA (p, d, q).

p, d, and q represent seasonality, trend, and noise in data respectively.

We performed the Time series decomposition that helps to deconstruct the time series into several components 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. The seasonal decomposition shows the trend to be removed, but the seasonality might still be present in the time series.

b) Seasonal Autoregressive Integrated Moving Average (SARIMA)

It is a statistical technique used for forecasting time series data, which is a series of observations recorded at regular intervals over time.

SARIMA works by modeling the relationships between past and present values of a time series and identifying patterns in the data. Specifically, SARIMA uses a combination of autoregression (AR) and moving average (MA) models, as well as differencing, to capture the patterns and seasonality in the data.

8. Assumptions

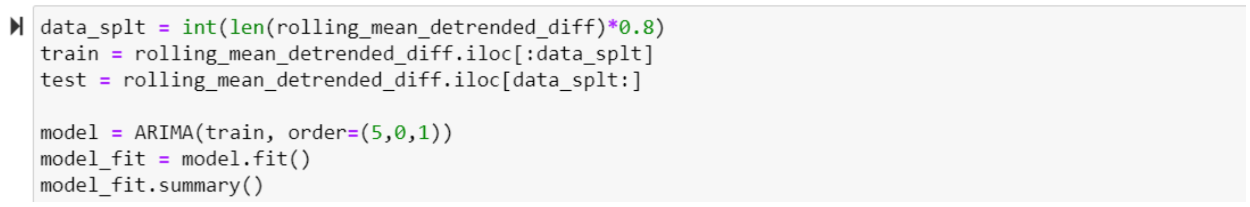
To use time series forecasting models, we need to ensure that our time series data is stationary i.e., constant mean, constant variance, and constant covariance with time.

To test the stationarity of time series, we used Rolling Mean. It's a rolling analysis of a time series model that is often used to assess the model's stability over time. The window is rolled (slid across the data) on a weekly basis, in which the average is taken on a weekly basis. Rolling Statistics is a visualization test, where we can compare the original data with the rolled data and check if the data is stationary or not.

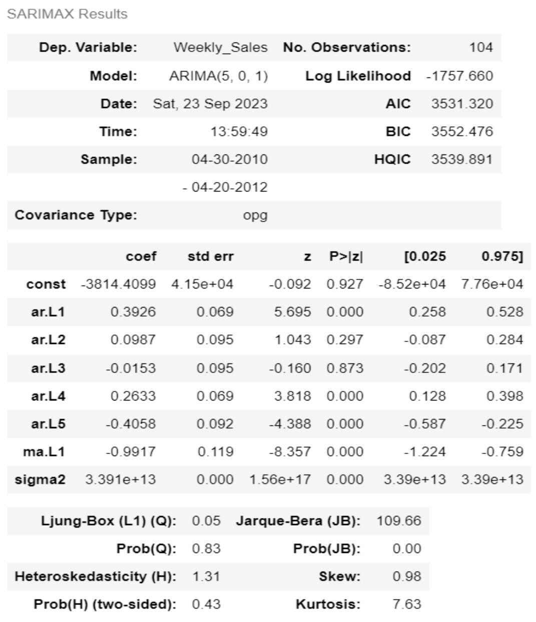
9) Model Evaluation and Techniques

When evaluating a model, we split our data into a training and a test set. While the training set is used to train the model and determine the optimal hyperparameters, the test set is used to evaluate it.

We can pick the model with the best performance and validate the model performance by splitting the data into a train and a test set.



```
data_splt = int(len(rolling_mean_detrended_diff)*0.8)
train = rolling_mean_detrended_diff.iloc[:data_splt]
test = rolling_mean_detrended_diff.iloc[data_splt:]

model = ARIMA(train, order=(5,0,1))
model_fit = model.fit()
model_fit.summary()
```



SARIMAX Results

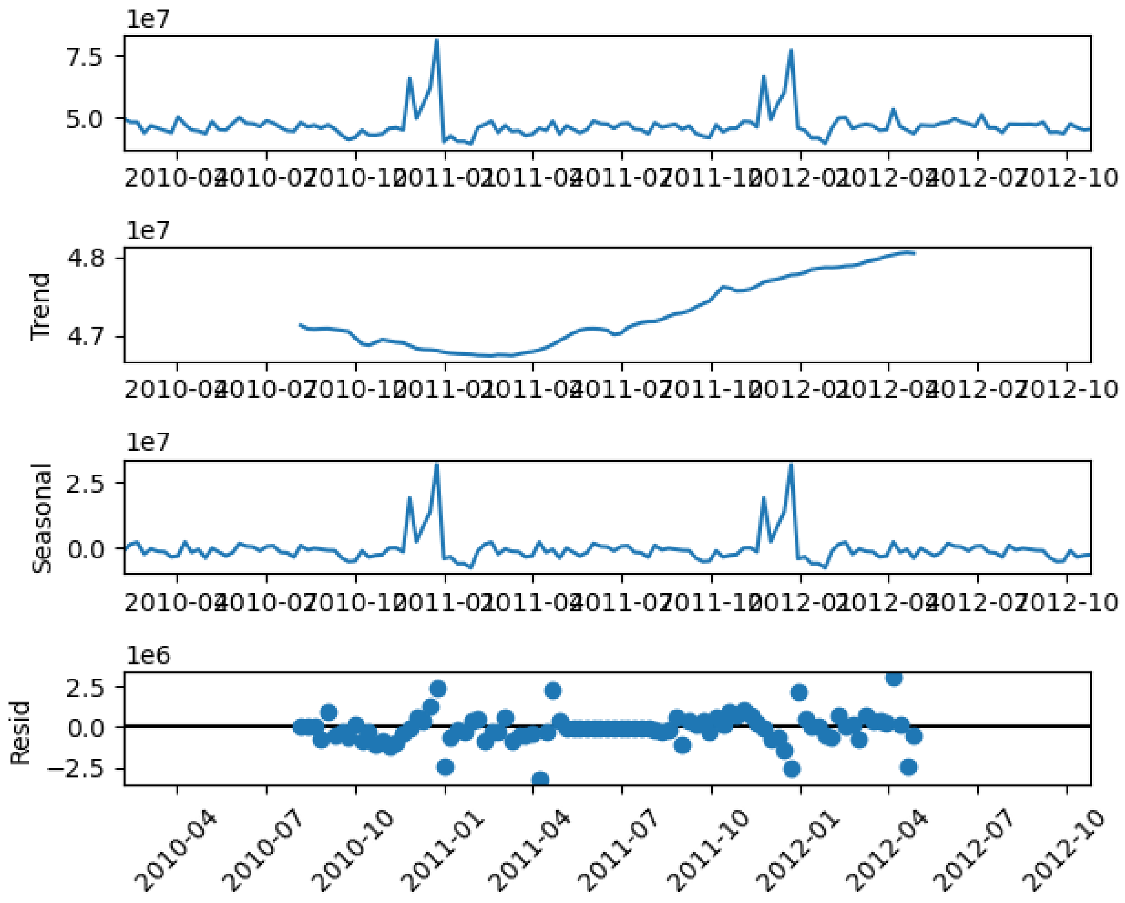
Dep. Variable:	Weekly_Sales	**No. Observations:**	104
Model:	ARIMA(5, 0, 1)	**Log Likelihood**	-1757.660
Date:	Sat, 23 Sep 2023	**AIC**	3531.320
Time:	13:59:49	**BIC**	3552.476
Sample:	04-30-2010	**HQIC**	3539.891
	- 04-20-2012		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-3814.4099	4.15e+04	-0.092	0.927	-8.52e+04	7.76e+04
ar.L1	0.3926	0.069	5.695	0.000	0.258	0.528
ar.L2	0.0987	0.095	1.043	0.297	-0.087	0.284
ar.L3	-0.0153	0.095	-0.160	0.873	-0.202	0.171
ar.L4	0.2633	0.069	3.818	0.000	0.128	0.398
ar.L5	-0.4058	0.092	-4.388	0.000	-0.587	-0.225
ma.L1	-0.9917	0.119	-8.357	0.000	-1.224	-0.759
sigma2	3.391e+13	0.000	1.56e+17	0.000	3.39e+13	3.39e+13

Ljung-Box (L1) (Q):	0.05	**Jarque-Bera (JB):**	109.66
Prob(Q):	0.83	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.31	**Skew:**	0.98
Prob(H) (two-sided):	0.43	**Kurtosis:**	7.63

When evaluating a time series machine learning model, we used different predictive modeling techniques to check the Seasonal variation and Trends in the data.

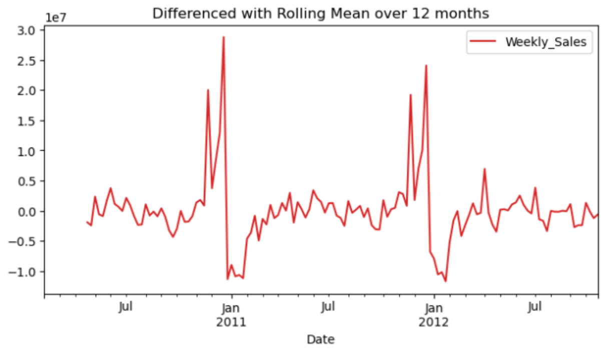
```
print('''\nQ.2 Use predictive modeling techniques to forecast the sales for each
store for the next 12 weeks.''')
data = agg_weekly_sales.copy()

print("\nCheck seasonal variation and trends.")
decompose_result = seasonal_decompose(data)
decompose_result.plot()
# plt.xticks(rotation=45)
plt.show()
```

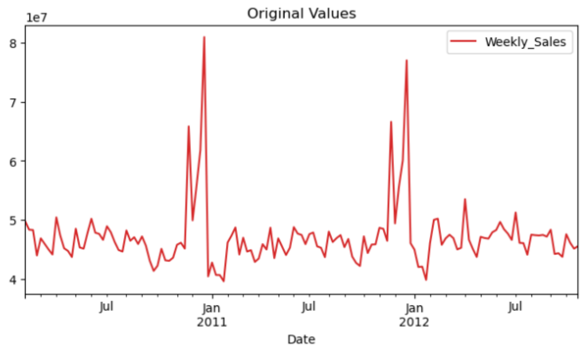


Removing Trend and Seasonality

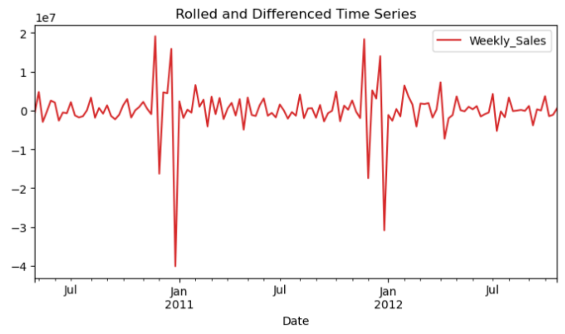
We create a new time series, by subtracting the rolling mean with the original data.

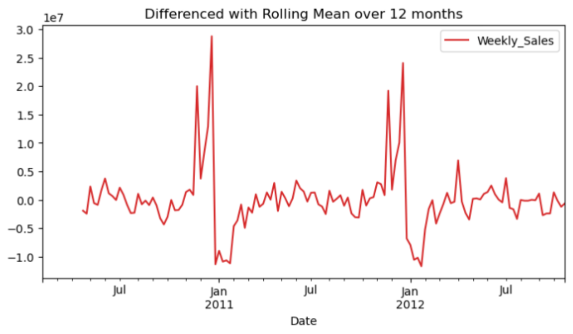
```
rolling_mean = data.rolling(window=12).mean()
rolling_mean_detrended = data - rolling_mean
rolling_mean_detrended_diff = rolling_mean_detrended - rolling_mean_detrended.shift()
rolling_mean_detrended_diff = rolling_mean_detrended_diff.dropna()


ax1 = plt.subplot(121)
# plt.figure(figsize=(12,4))
rolling_mean_detrended.plot(color="tab:red",
                           title= "Differenced with Rolling Mean over 12 months",
                            ax=ax1)

ax2 = plt.subplot(122)
# plt.figure(figsize=(18,4))
data.plot(figsize=(18,4),color="tab:red",
          title="Original Values",
          ax=ax2)
```





```
ax1 = plt.subplot(121)
# plt.figure(figsize=(12,4))
rolling_mean_detrended_diff.plot(color="tab:red",
                                title= "Rolled and Differenced Time Series",
                                 ax=ax1)

ax2 = plt.subplot(122)
# plt.figure(figsize=(18,4))
rolling_mean_detrended.plot(figsize=(18,4),color="tab:red",
          title="Differenced with Rolling Mean over 12 months",
          ax=ax2)
```





```python
adf_result = adfuller(rolling_mean_detrended_diff["Weekly_Sales"])
if adf_result[1] < 0.05:
    print(f"\nP-Value: {adf_result[1]}\nadfuller result: Go\n")
else:
    print(f"\nP-Value: {adf_result[1]}\nadfuller result: NoGo\n")
```

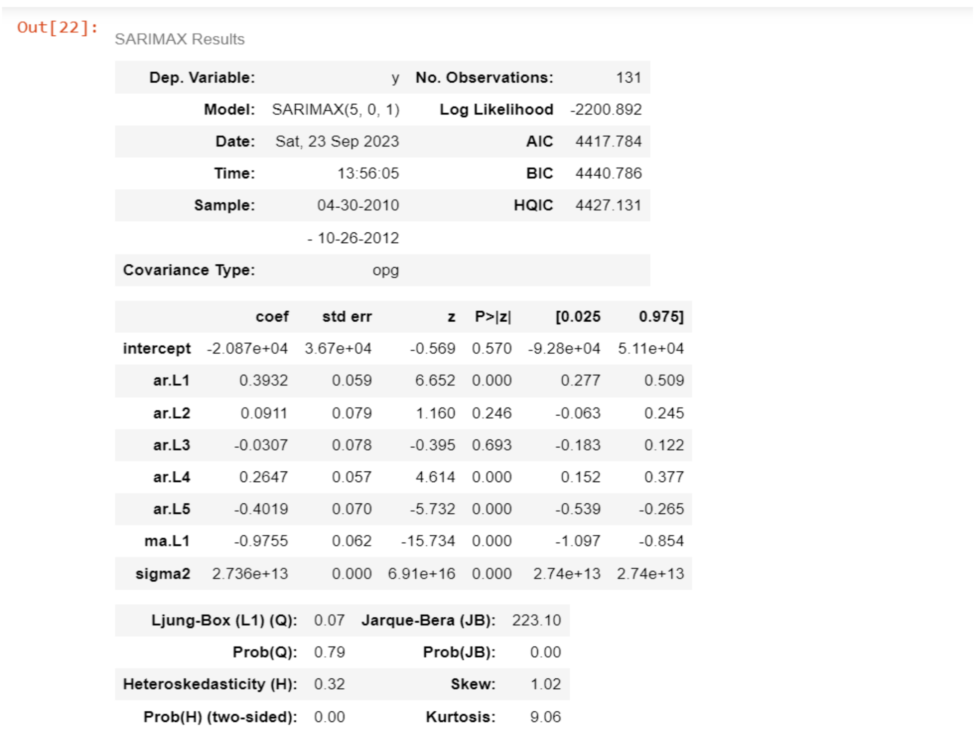
```
P-Value: 7.925658104125754e-08
adfuller result: Go
```

Hyperparameter tuning for ARIMA:

To choose the best combination of the above parameters, we'll use a grid search. The best combination of parameters will give the lowest Akaike information criterion (AIC) score. AIC tells us the quality of statistical models for a given set of data.

```python
order = auto_arima(rolling_mean_detrended_diff["Weekly_Sales"], trace = True)
order.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=4435.173, Time=0.23 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=4473.974, Time=0.03 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=4452.977, Time=0.03 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=4445.558, Time=0.05 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=4471.974, Time=0.02 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=4447.486, Time=0.11 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.20 sec
 ARIMA(3,0,2)(0,0,0)[0] intercept   : AIC=4432.008, Time=0.16 sec
 ARIMA(3,0,1)(0,0,0)[0] intercept   : AIC=4438.941, Time=0.16 sec
 ARIMA(4,0,2)(0,0,0)[0] intercept   : AIC=4430.873, Time=0.34 sec
 ARIMA(4,0,1)(0,0,0)[0] intercept   : AIC=4437.666, Time=0.13 sec
 ARIMA(5,0,2)(0,0,0)[0] intercept   : AIC=4420.755, Time=0.66 sec
 ARIMA(5,0,1)(0,0,0)[0] intercept   : AIC=4417.784, Time=0.28 sec
 ARIMA(5,0,0)(0,0,0)[0] intercept   : AIC=4434.790, Time=0.26 sec
 ARIMA(4,0,0)(0,0,0)[0] intercept   : AIC=4437.083, Time=0.15 sec
 ARIMA(5,0,1)(0,0,0)[0]             : AIC=inf, Time=0.27 sec

Best model:  ARIMA(5,0,1)(0,0,0)[0] intercept
Total fit time: 3.095 seconds
```



Out[22]:

SARIMAX Results

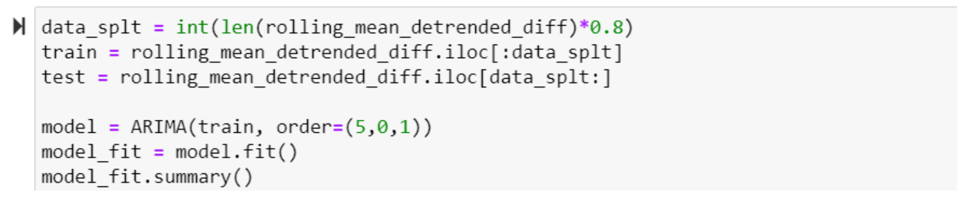
Dep. Variable:	y	No. Observations:	131
Model:	SARIMAX(5, 0, 1)	Log Likelihood	-2200.892
Date:	Sat, 23 Sep 2023	AIC	4417.784
Time:	13:56:05	BIC	4440.786
Sample:	04-30-2010	HQIC	4427.131
	- 10-26-2012		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-2.087e+04	3.67e+04	-0.569	0.570	-9.28e+04	5.11e+04
ar.L1	0.3932	0.059	6.652	0.000	0.277	0.509
ar.L2	0.0911	0.079	1.160	0.246	-0.063	0.245
ar.L3	-0.0307	0.078	-0.395	0.693	-0.183	0.122
ar.L4	0.2647	0.057	4.614	0.000	0.152	0.377
ar.L5	-0.4019	0.070	-5.732	0.000	-0.539	-0.265
ma.L1	-0.9755	0.062	-15.734	0.000	-1.097	-0.854
sigma2	2.736e+13	0.000	6.91e+16	0.000	2.74e+13	2.74e+13

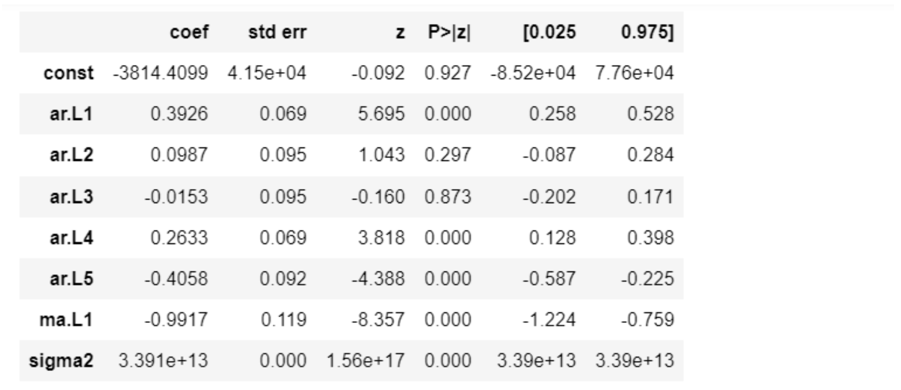
Ljung-Box (L1) (Q):	0.07	Jarque-Bera (JB):	223.10
Prob(Q):	0.79	Prob(JB):	0.00
Heteroskedasticity (H):	0.32	Skew:	1.02
Prob(H) (two-sided):	0.00	Kurtosis:	9.06

The above iteration suggested that SARIMAX (5, 0, 1) **x** (0, 0, 0) is the best parameter combination with the lowest AIC: 4417.784.

Fitting the ARIMA model



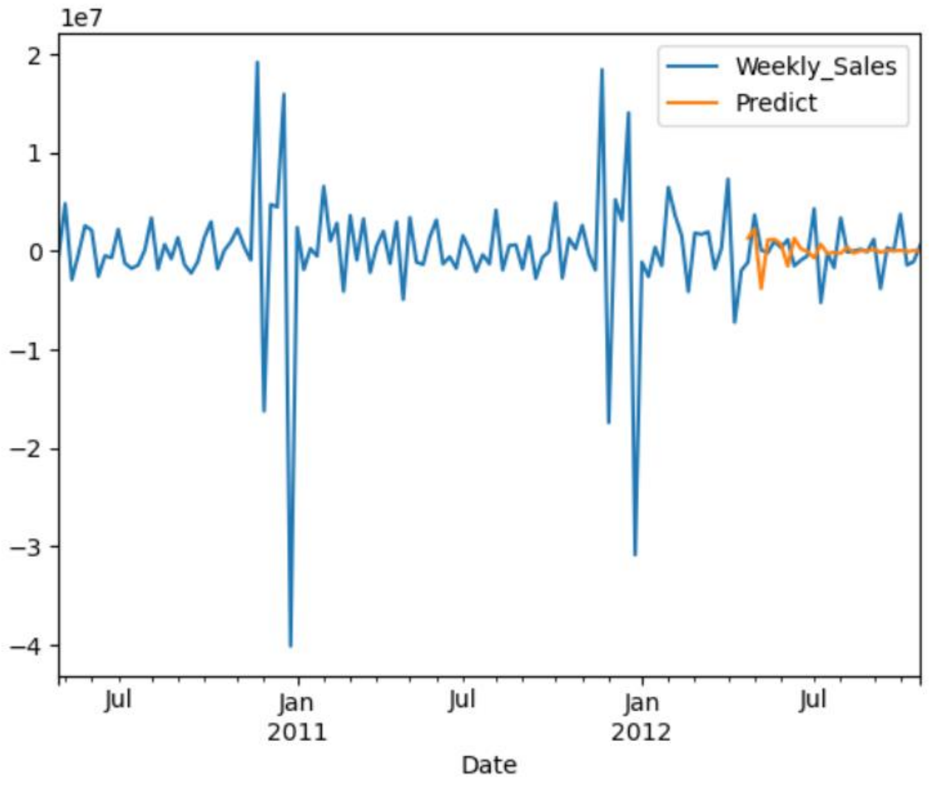
```
data_splt = int(len(rolling_mean_detrended_diff)*0.8)
train = rolling_mean_detrended_diff.iloc[:data_splt]
test = rolling_mean_detrended_diff.iloc[data_splt:]

model = ARIMA(train, order=(5,0,1))
model_fit = model.fit()
model_fit.summary()
```



	coef	std err	z	P>|z|	[0.025	0.975]
const	-3814.4099	4.15e+04	-0.092	0.927	-8.52e+04	7.76e+04
ar.L1	0.3926	0.069	5.695	0.000	0.258	0.528
ar.L2	0.0987	0.095	1.043	0.297	-0.087	0.284
ar.L3	-0.0153	0.095	-0.160	0.873	-0.202	0.171
ar.L4	0.2633	0.069	3.818	0.000	0.128	0.398
ar.L5	-0.4058	0.092	-4.388	0.000	-0.587	-0.225
ma.L1	-0.9917	0.119	-8.357	0.000	-1.224	-0.759
sigma2	3.391e+13	0.000	1.56e+17	0.000	3.39e+13	3.39e+13

Model Prediction

```
rolling_mean_detrended_diff["Predict"] = model_fit.predict(start=len(train), end=len(train)+len(test)-1, dynamic=True)
rolling_mean_detrended_diff[["Weekly_Sales", "Predict"]].plot()
```



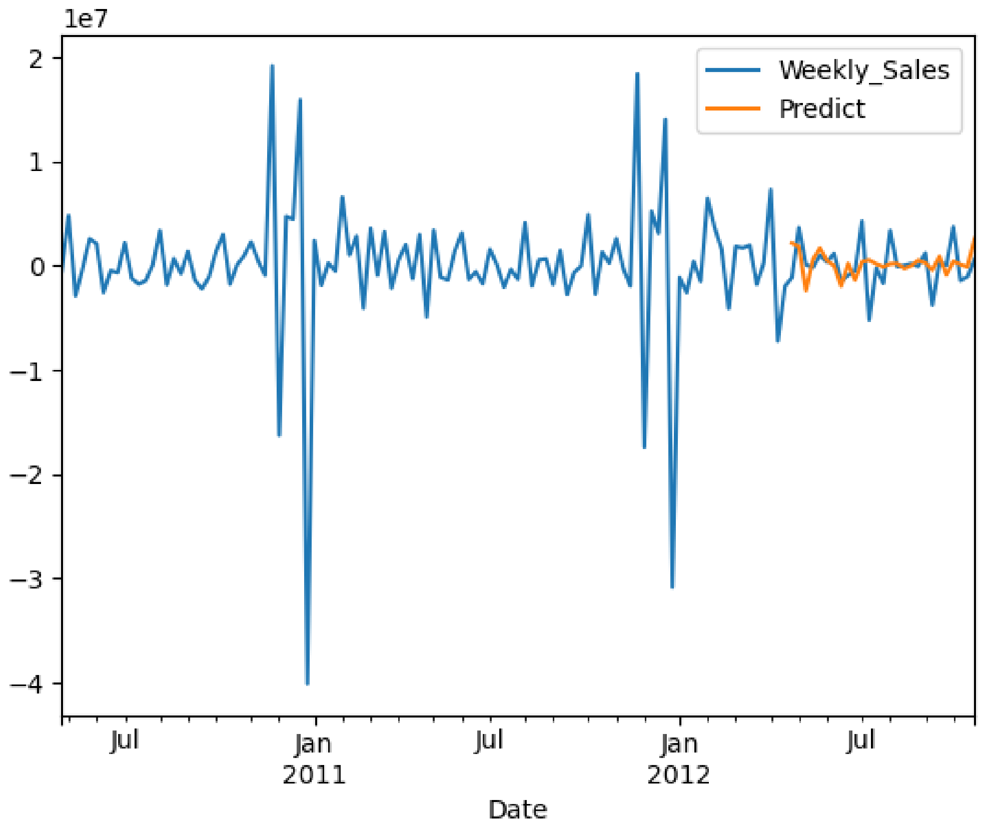
From the above graph, the predictions are a bit off the actual values from the test set. Therefore, we can move to the seasonal ARIMA model for our forecasting.

In the seasonal ARIMA model, we have specified the seasonal order as well. The seasonal order remains the same as the ARIMA order, and we added the periodic order in the seasonal order according to the periodicity.

```
from statsmodels.tsa.statespace.sarimax import SARIMAX, SARIMAXResults

model = SARIMAX(train, order=(5,0,1), seasonal_order=(5,0,1,12), enforce_stationarity=False)
model_fit = model.fit()
```

Here, we can see the predicted values on the test set are more accurate than the ARIMA model. Therefore, we have successfully created a Time series forecast model.

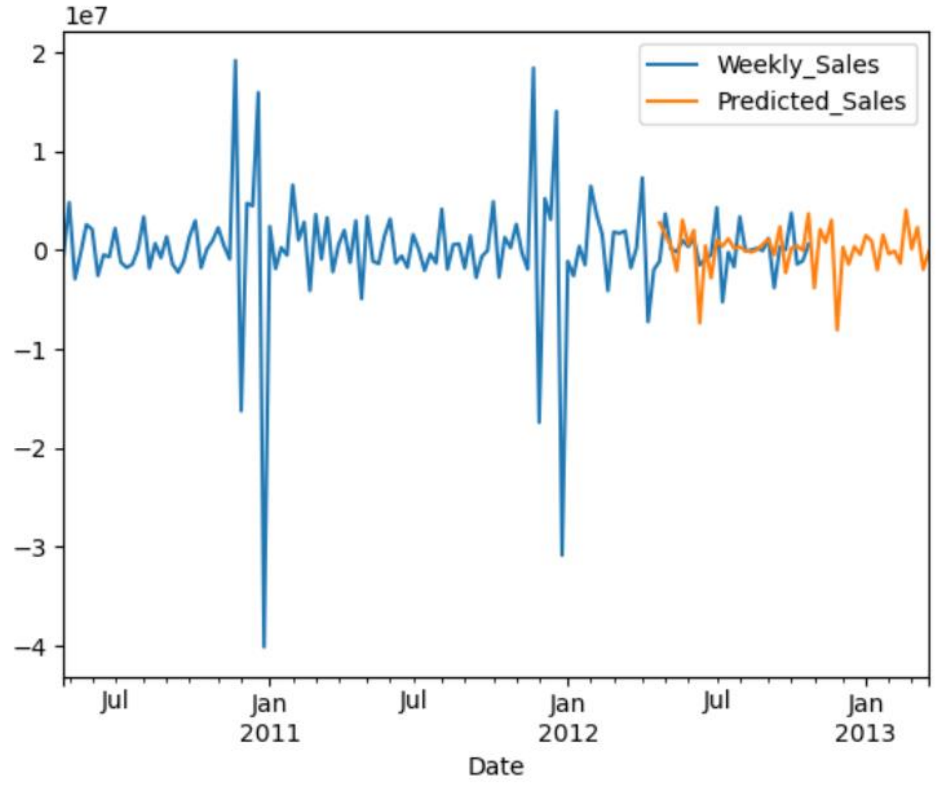
```
print("\nCheck against test data.")
rolling_mean_detrended_diff["Predict"] = model_fit.predict(start=len(train), end=len(train)+len(test)-1, dynamic=True)
ax2b = rolling_mean_detrended_diff[["Weekly_Sales", "Predict"]].plot()
plt.show()
```



10) Inferences

```
print("\nInferences")
forecast = model_fit.forecast(steps=48)
ax2c = rolling_mean_detrended_diff["Weekly_Sales"].plot()
ax2c = forecast.plot(label="Predicted_Sales")
plt.legend(loc="best")
plt.show()
```

```
Inferences
```



We have successfully trained the model on the transformed time series. Now we have a forecast model that can predict the sales in the coming weeks. With real-world data, the predictions will almost never be 100% in sync with the data, but we can see that the predictions stay in a reasonable range for off-holiday season sales. We can use the forecast information to optimize staffing and other resources.

11) Conclusion

This project analysis will help Walmart to make well-informed decisions to boost their sales in future and would help them to get the answers about the following:

1) How the Weekly Sales vary by Unemployment Rate
2) How the Weekly Sales vary by Temperature
3) How the Weekly Sales vary by CPI
4) Does Weekly Sales show a Seasonal Trend
5) Who is the best and worst performing Stores based on the historical data
6) Gather useful insights and forecast the future sales for each store.

12) Future Possibilities of the Project

1) To check into the store that has poor weekly sales and check for other factors as well that impact on their sales.
2) To further improve the predictive model using the ensembling method to combine models and come with a better model.
3) Take the data to Department level and predict the Department level sales which would help to solve the inventory management issues and supply chain management.

13) References

1) https://medium.com/analytics-vidhya/predicting-sales-time-series-analysis-forecasting-with-python-b81d3e8ff03f
2) https://www.kaggle.com/code/anushkaml/walmart-time-series-sales-forecasting/notebook
3) https://www.kaggle.com/code/lakshminarashimman/walmart-sales-prediction-and-forecasting