RETAIL ANALYSIS WITH WALMART DATA

WRITE-UP

Introduction:

Walmart Inc. is an American multinational retail corporation that operates a chain of hypermarkets (also called supercenters), discount department stores, and grocery stores from the United States, headquartered in Bentonville, Arkansas.

Project Scope:

- store has maximum sales
- store has maximum standard deviation i.e., the sales vary a lot. Also, find out the coefficient of mean to standard deviation
- store/s has good quarterly growth rate in Q3'2012
- Some holidays have a negative impact on sales. Find out holidays which have higher sales than the mean sales in non-holiday season for all stores together
- Provide a monthly and semester view of sales in units and give insights
- Linear Regression – Utilize variables like date and restructure dates as 1 for 5 Feb 2010 (starting from the earliest date in order). Hypothesize if CPI, unemployment, and fuel price have any impact on sales
- Change dates into days by creating new variable

Dataset Description

This is the historical data that covers sales from 2010-02-05 to 2012-11-01, in the file Walmart_Store_sales. Within this file you will find the following fields:

- 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

Data cleaning:

1) Handling Null values:

The data was checked for null values and it contained no null values.

```
In [4]: sales.isnull().sum()

Out[4]: Store           0
        Date            0
        Weekly_Sales    0
        Holiday_Flag    0
        Temperature     0
        Fuel_Price      0
        CPI             0
        Unemployment    0
        dtype: int64
```

2) Datatypes:

The datatype of all the variables were checked and the column “Date” was converted to “datetime” datatype

```
In [3]: sales['Date']= pd.to_datetime(sales['Date'])
        sales.info()

        <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   datetime64[ns]
         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: datetime64[ns](1), float64(5), int64(2)
        memory usage: 402.3 KB
```

3) Column Addition:

Derived columns were added to the current dataset for analysis purpose. Columns like “Day, Month, Year” were derived from “Date” column.

```
In [5]: sales['Day']= pd.DatetimeIndex(sales['Date']).day
        sales['Month']= pd.DatetimeIndex(sales['Date']).month
        sales['Year']= pd.DatetimeIndex(sales['Date']).year
        sales.head()
```

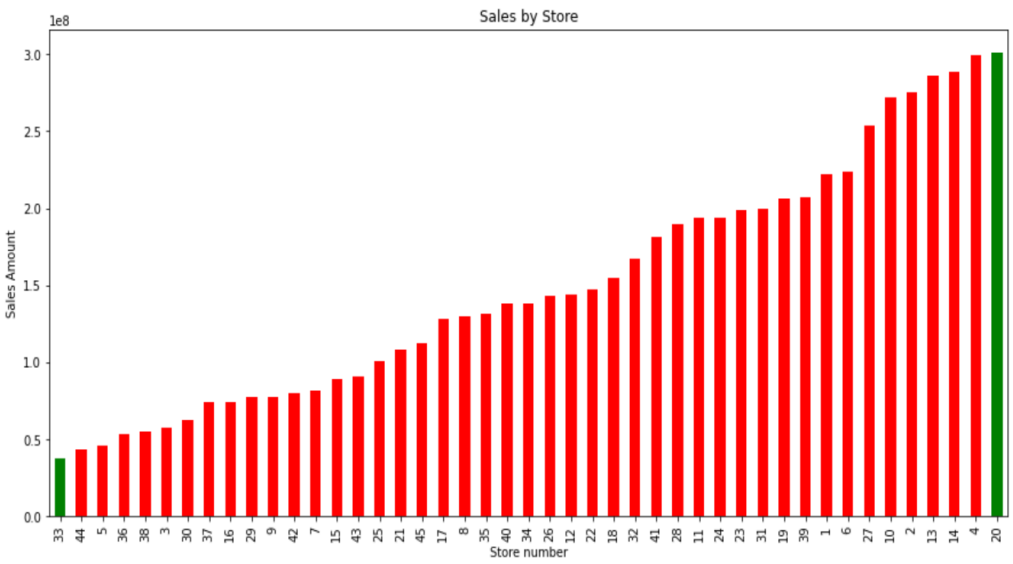
Out[5]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Day	Month	Year
0	1	2010-05-02	1643690.90	0	42.31	2.572	211.096358	8.106	2	5	2010
1	1	2010-12-02	1641957.44	1	38.51	2.548	211.242170	8.106	2	12	2010
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106	19	2	2010
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106	26	2	2010
4	1	2010-05-03	1554806.68	0	46.50	2.625	211.350143	8.106	3	5	2010

DATA ANALYSIS:

1) Store with maximum sales:

- Store and their weekly sales data was stored separately in an array. That array was used to find the total sales of each store by grouping data based on the store variable. The calculated value was then sorted and stored
- A bar chart highlighting the store with maximum and minimum sales was then created using the above data



2) Store with maximum Standard Deviation:

A simple standard deviation built-in function (std()) was used to calculate the std for all stores. The store having maximum standard deviation is “Store No.14”

```
In [8]: #Store having maximum standard deviation

        sales_std = sales.groupby('Store')['Weekly_Sales'].std().sort_values(ascending=False)
        print("The Store with maximum std deviation is"+ str(sales_std.head(1)))

        The Store with maximum std deviation isStore
        14    317569.949476
        Name: Weekly_Sales, dtype: float64
```

3) Stores with growth rate in Q3’2012

The data was filtered based on the date to get the Q3’2012 sales value of all the stores. Then the total sales value of each store for that particular period was calculated.

The top 5 stores having high growth rate was displayed.

```
In [11]: #Store having good quarterly growth rate in Q3'2012

         sales_Q3 = sales[(sales['Date']>'2012-07-01')&
                          (sales['Date']< '2012-09-30')].groupby('Store')['Weekly_Sales'].sum().sort_values(ascending=False)
         sales_Q3.head()

Out[11]: Store
         4     25652119.35
         20    24665938.11
         13    24319994.35
         2     22396867.61
         10    21169356.45
         Name: Weekly_Sales, dtype: float64
```

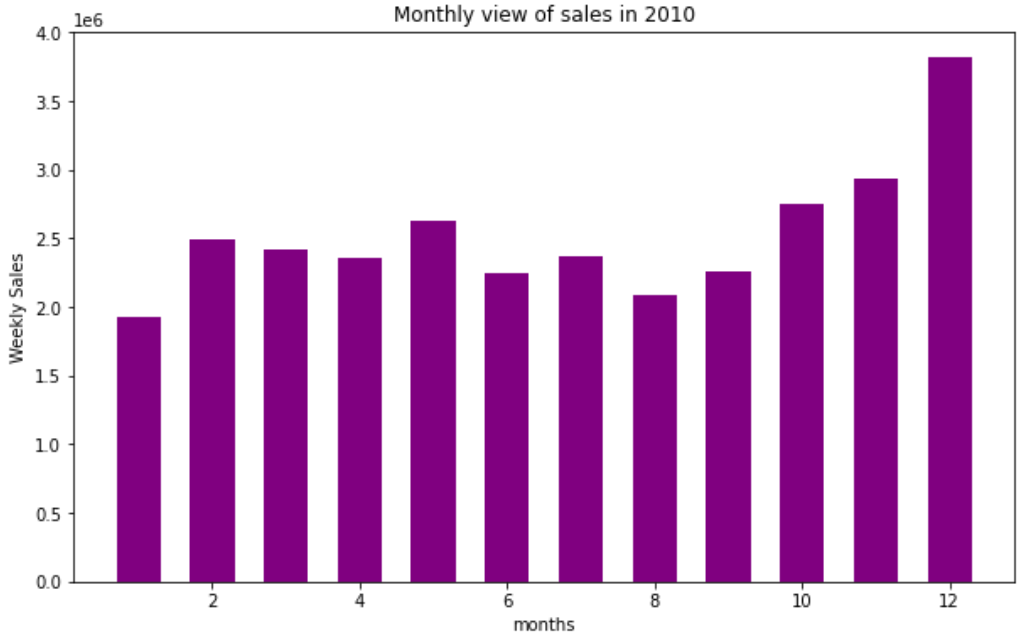
4) Holidays with higher sales:

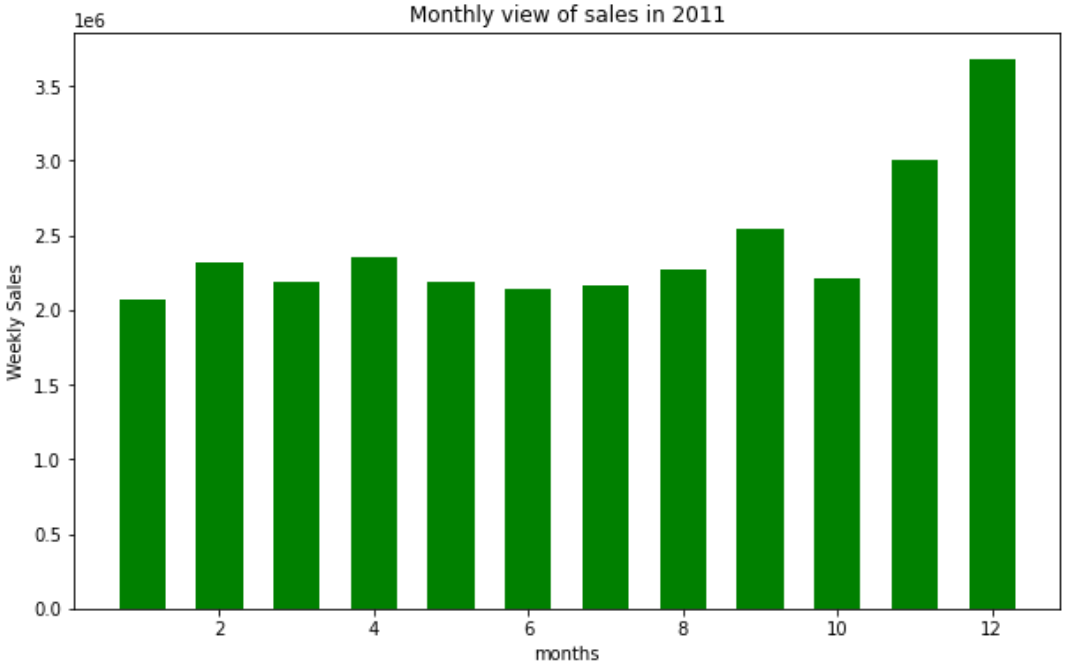
- To check whether the holidays have negative impact on sales, a comparison between holiday and non-holiday sales is to be done.
- First, the total holiday sales of all stores were calculated date wise using holiday flag as filter
- Second, the mean sales of all stores were calculated date wise using the same holiday flag as filter
- Using for loop each holiday sales was compared with each non-holiday sales and the higher ones were displayed

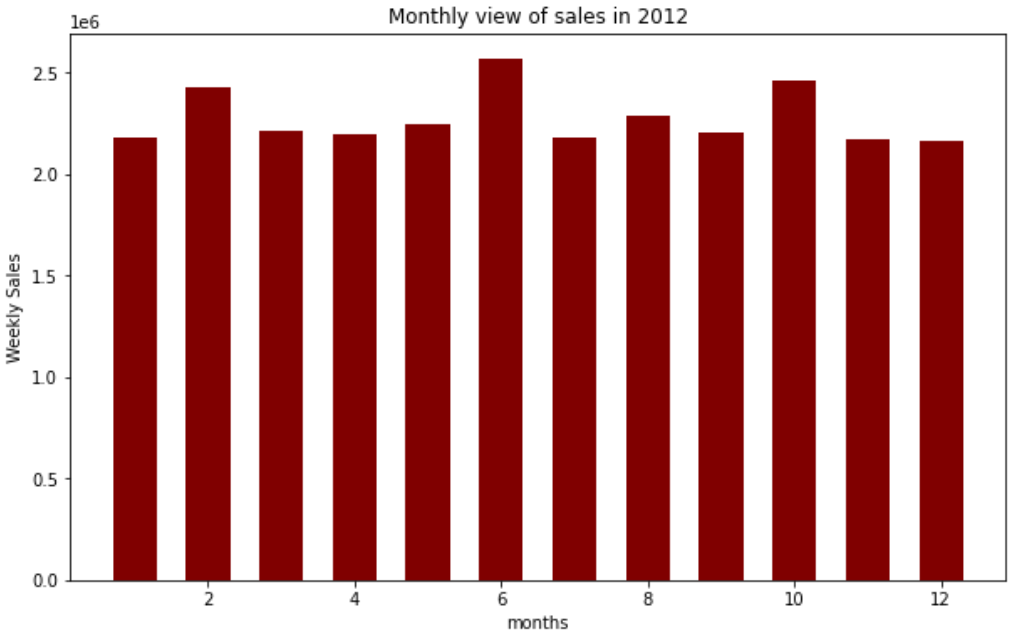
```
In [15]: print("Holidays whose sales are higher than non-holiday sales")
         for X in holidayweek_sale_sum.itertuples():
             for X1 in mean_non_holiday.itertuples():
                 if X.Weekly_Sales > X1.Weekly_Sales:
                     print("On Date {} , sale {}"
                           .format(X.Date,X.Weekly_Sales))
                     break;

         Holidays whose sales are higher than non-holiday sales
         On Date 2010-10-09 00:00:00 , sale 45634397.839999996
         On Date 2010-11-26 00:00:00 , sale 65821003.24
         On Date 2010-12-02 00:00:00 , sale 48336677.63
         On Date 2010-12-31 00:00:00 , sale 40432519.0
         On Date 2011-09-09 00:00:00 , sale 46763227.53
         On Date 2011-11-02 00:00:00 , sale 47336192.79
         On Date 2011-11-25 00:00:00 , sale 66593605.26
         On Date 2011-12-30 00:00:00 , sale 46042461.04
         On Date 2012-07-09 00:00:00 , sale 48330059.31
         On Date 2012-10-02 00:00:00 , sale 50009407.92
```

5) Monthly Sales

Monthly view of sales is plotted using bar chart. The derived columns “Month” and “Year” was used as filter to find the total sales of all stores for a particular month and in a particular year.





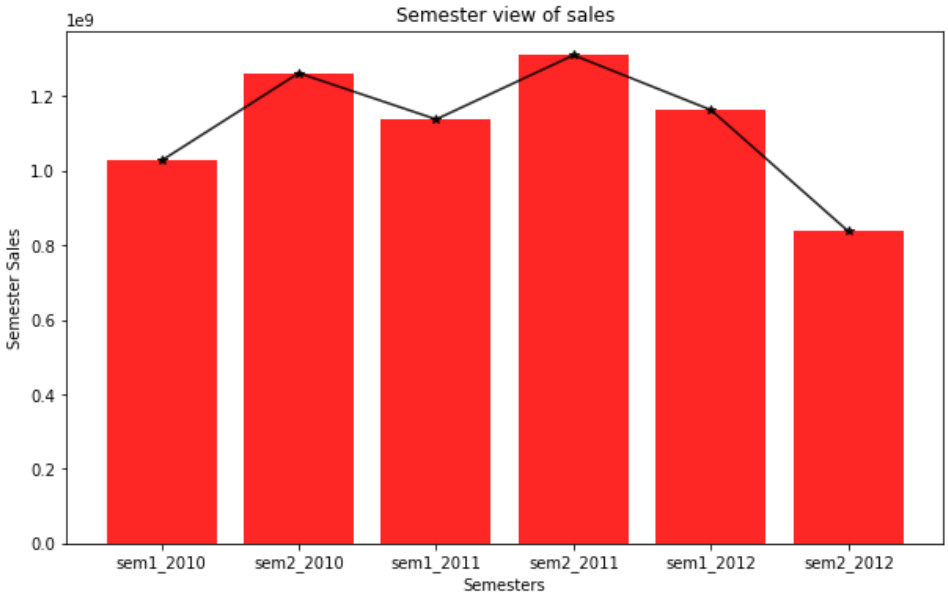


Insights:

The sale value is found to be even through out the year in all months with small variations. It is observed that year-end sales have given a rise in total sale value during the month of November and December

6) Semester Sales:

In semester wise sales, there is a rise and fall pattern observed i.e. the sale increases every second semester and decreases in first semester. It is showed by plotting the data using both bar and line chart together in a single frame.



Statistical Model

Linear Regression – Store 1

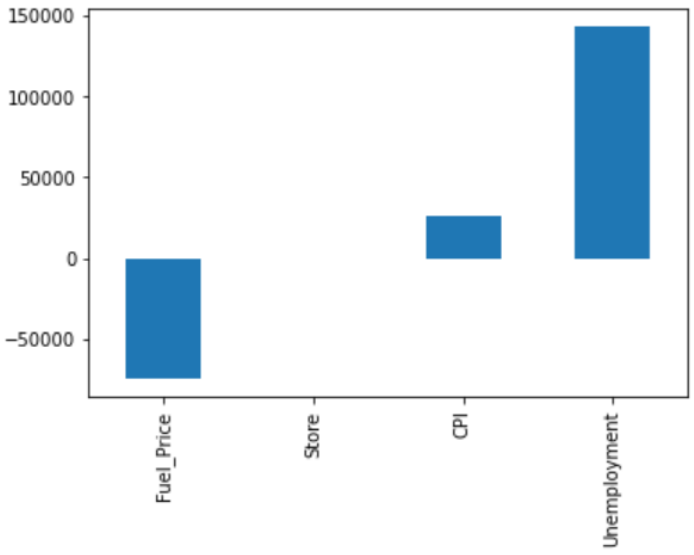
- To predict the demand for store -1 with “CPI, Unemployment, fuel Price” as independent variables and weekly sales as dependent/ target variable
- Train and Test data was split in the ratio of 70:30
- Below screen shot shows the metrics of linear regression where the accuracy was only 11%

```
In [24]: print('Accuracy:',reg_all.score(X_train, y_train)*100)
         print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
         print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
         print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

         Accuracy: 11.202300833725253
         Mean Absolute Error: 88238.04791792025
         Mean Squared Error: 13528318683.05324
         Root Mean Squared Error: 116311.30075385298
```

Relationship b/w feature and target variables:

It is observed from the below relationship chart that the variables “CPI and Unemployment” are having positive impact on sales where “Fuel price” is having negative relation with the sales

```
In [23]: relation=pd.Series(reg_all.coef_,x.columns).sort_values()
         relation.plot(kind="bar")

Out[23]: <AxesSubplot:>
```



Polynomial Regression:

- To predict the demand for store -1 with “CPI, Unemployment, fuel Price” as independent variables and weekly sales as dependent/ target variable
- The degree used here is 3
- Below screen shot shows the metrics of Polynomial regression where the accuracy is 85%

```
from math import sqrt
from sklearn. metrics import r2_score
from sklearn.metrics import mean_squared_error

#Root mean square error value and score(Accuracy) calculation
r_squared = r2_score(y_target, Ypipehat)
print('R-squared :', r_squared)
print('RMSE:' , sqrt(mean_squared_error(y_target,Ypipehat)))
```

```
Pipeline(steps=[('scale', StandardScaler()),
                ('polynomial', PolynomialFeatures(include_bias=False)),
                ('model', LinearRegression())])
[[1.000000e+00 1.671584e+06]
 [1.000000e+00 1.568736e+06]
 [1.000000e+00 1.635136e+06]
 [1.000000e+00 1.435456e+06]]
(143, 4)
(143, 35)
R-squared : 0.8509254518051932
RMSE: 60013.50032850766
```

The best model to predict demand is Polynomial regression

Changing Dates into Days:

A new column named “Days” was added to convert the date to days of the week using to_datetime function

In [99]:
```
#change of dates to days
sales['Days'] = pd.to_datetime(sales['Date']).dt.day_name()
sales
```

Out[99]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Day	Month	Year	Days
0	1	2010-05-02	1643690.90	0	42.31	2.572	211.096358	8.106	2	5	2010	Sunday
1	1	2010-12-02	1641957.44	1	38.51	2.548	211.242170	8.106	2	12	2010	Thursday
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106	19	2	2010	Friday
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106	26	2	2010	Friday
4	1	2010-05-03	1554806.68	0	46.50	2.625	211.350143	8.106	3	5	2010	Monday
...	...	...	...	...	...	...	...	...	...	...	...	...
6430	45	2012-09-28	713173.95	0	64.88	3.997	192.013558	8.684	28	9	2012	Friday
6431	45	2012-05-10	733455.07	0	64.89	3.985	192.170412	8.667	10	5	2012	Thursday
6432	45	2012-12-10	734464.36	0	54.47	4.000	192.327265	8.667	10	12	2012	Monday
6433	45	2012-10-19	718125.53	0	56.47	3.969	192.330854	8.667	19	10	2012	Friday
6434	45	2012-10-26	760281.43	0	58.85	3.882	192.308899	8.667	26	10	2012	Friday

6435 rows × 12 columns

Conclusion:

Thus, in this project “Retail analysis with Walmart Data” a comprehensive analysis is made.

- Data cleaning done by checking for null values and the variables were checked for correct datatypes
- General sales analysis was performed by finding the store with maximum sales and maximum standard deviation
- The impact of holidays on the sales was analysed
- Statistical analysis to predict the demand, using Linear and polynomial regression was done