Retail Analysis with Walmart Data

Part-1

Business Understanding:

Walmart is an American retail corporation that operates a chain of hypermarkets, discount department stores, and grocery stores.

Here we analyzing and building model for 45 stores of Walmart.

Retail Analysis with Walmart Data

- Walmart runs several promotional markdown events throughout the year. These markdowns precede prominent holidays, the four largest of all, which are the Super Bowl, Labour Day, Thanksgiving, and Christmas. The weeks including these holidays are weighted five times higher in the evaluation than non-holiday weeks. Part of the challenge presented by this competition is modeling the effects of markdowns on these holiday weeks in the absence of complete/ideal historical data. Historical sales data for 45 Walmart stores located in different regions are available.

In this project we focused to answer the following questions:

A .Basic Statistics tasks

1. Which store has maximum sales
2. Which store has maximum standard deviation i.e., the sales vary a lot. Also, find out the coefficient of mean to standard deviation
3. Which store/s has good quarterly growth rate in Q3'2012
4. 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
5. Provide a monthly and semester view of sales in units and give insights

B. Statistical Model

1. For Store 1 – Build prediction models to forecast demand *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.

Data Understanding

In the file Walmart_Store_sales, there are sales data available for 45 stores This is the historical data that covers sales from 2010-02-05 to 2012-11-01

The data contains these features:

- 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

(1) Import required libraries and dataset—

```
# Import necessary libraries:
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import dates
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')
```

```python
# Load dataset
data=pd.read_csv('Walmart_Store_sales.csv')
data
```

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

6435 rows × 8 columns

2) Changing the data type of the 'Date' column —

We are changing the data type of the 'Date' column because it is an object type.

```python
# Convert date to datetime format and show dataset information
data['Date']=pd.to_datetime(data['Date'])
data.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
```

Here, the dataset does not have any null values. So, we are ready to proceed with basic statistical tasks.

(A) Statistical Tasks —

1. Which store has maximum sales?

In order to find out the maximum sales, I have created a new variable called ‘total_sales’. Then groupby stores and find the sum of the weekly sales of each store. This will give me the maximum sales.

```
total_sales=data.groupby('Store')['Weekly_Sales'].sum().round().sort_values(ascending=0)
```

```
# Maximum Sales

pd.DataFrame(total_sales).head()
```

	Weekly_Sales
Store	
20	301397792.0
4	299543953.0
14	288999911.0
13	286517704.0
2	275382441.0

Store-20 has the maximum sales of $301397792.0

2. Which store has maximum standard deviation i.e., the sales vary a lot. Also, find out the coefficient of mean to standard deviation.

To find out the maximum standard deviation, create a new variable and then group it by stores and find the standard deviation.

Maximum standard deviation

```
std_sales = data.groupby('Store')['Weekly_Sales'].std().round(3).sort_values(ascending=0)
```

```
pd.DataFrame(std_sales).head()
```

	Weekly_Sales
Store	
14	317569.949
10	302262.063
20	275900.563
4	266201.442
13	265506.996

Store - 14 has a maximum standard deviation = $317569.949

Coefficient of mean to standard deviation

Coefficient of mean to standard deviation

```
store14 = data[data.Store == 14].Weekly_Sales
```

```
store14
```

```
1859    2623469.95
1860    1704218.84
1861    2204556.70
1862    2095591.63
1863    2237544.75
           ...
1997    1522512.20
1998    1687592.16
1999    1639585.61
2000    1590274.72
2001    1704357.62
Name: Weekly_Sales, Length: 143, dtype: float64
```

```
Coefficient_of_variantion=store14.std()/store14.mean()*100
Coefficient_of_variantion.round(2)
```

```
15.71
```

Coefficient of mean to Standard Deviation = 15.71%

3. Which store/s has a good quarterly growth rate in Q3'2012?

First, we will find the Q2 sales and then Q3 sales, take out the difference and then find the growth rate.

```
q2_sales= data[((data['Date']>="2012-04-01")&(data['Date']<="2012-06-30"))].groupby('Store')['Weekly_Sales'].sum().round()
q3_sales= data[((data['Date']>="2012-07-01")&(data['Date']<="2012-09-30"))].groupby('Store')['Weekly_Sales'].sum().round()
```

```
pd.DataFrame({'Q2 Sales':q2_sales,
              'Q3 Sales':q3_sales,
              'Difference':(q3_sales-q2_sales),
              'Growth Rate':(q3_sales-q2_sales)/q3_sales*100}).sort_values(by=['Growth Rate'],ascending=0).head()
```

Store	Q2 Sales	Q3 Sales	Difference	Growth Rate
16	6626133.0	6441311.0	-184822.0	-2.869323
7	7613594.0	7322394.0	-291200.0	-3.976841
35	10753571.0	10252123.0	-501448.0	-4.891163
26	13218290.0	12417575.0	-800715.0	-6.448240
39	20191586.0	18899955.0	-1291631.0	-6.834043

Q2 sales has always higher than Q3 sales so No store shown quaterly growth rate in Q3'2012, although store 16 has maximum growth rate as compared to others.

4. 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?

We have 4 Holiday Events,

(1) Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13, (2) Labour Day: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13, (3) Thanksgiving: 26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13,

(4) Christmas: 31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13. Now calculate the holiday event sales of each of the events and then find the non-holiday sales.

Holiday Event Sales:

```
#Holiday Events:

Super_Bowl = ['12-02-10', '11-02-11', '10-02-12', '8-02-13']
Labour_Day = ['10-09-10', '9-09-11', '7-09-12', '6-09-13']
Thanksgiving = ['26-11-10', '25-11-11', '23-11-12', '29-11-13']
Christmas = ['31-12-10', '30-12-11', '28-12-12', '27-12-13']
```

```
# Calculating holiday events sales
Super_Bowl_sales = data.loc[data.Date.isin(Super_Bowl)]['Weekly_Sales'].mean().round(2)
Labour_Day_sales = data.loc[data.Date.isin(Labour_Day)]['Weekly_Sales'].mean().round(2)
Thanksgiving_sales = data.loc[data.Date.isin(Thanksgiving)]['Weekly_Sales'].mean().round(2)
Christmas_sales = data.loc[data.Date.isin(Christmas)]['Weekly_Sales'].mean().round(2)
```

```
Super_Bowl_sales, Labour_Day_sales, Thanksgiving_sales, Christmas_sales
```

```
(1079127.99, 1042427.29, 1471273.43, 960833.11)
```

Non-holiday Sales and Comparison:

```
#Non-holiday Sales:
non_holiday_sales = data[(data['Holiday_Flag']== 0)]['Weekly_Sales'].mean().round(2)
non_holiday_sales
```

```
1041256.38
```

```
result = pd.DataFrame([{'Super Bowl Sales':Super_Bowl_sales,
              'Labour Day Sales':Labour_Day_sales,
              'Thanksgiving Sales':Thanksgiving_sales,
              'Christmas Sales':Christmas_sales,
              'Non Holiday Sales':non_holiday_sales}]).T

result
```

	0
Super Bowl Sales	1079127.99
Labour Day Sales	1042427.29
Thanksgiving Sales	1471273.43
Christmas Sales	960833.11
Non Holiday Sales	1041256.38

Here Thanksgiving has the highest sales (1,471,273.43) than non-holiday sales (1,041,256.38).

5. Provide a monthly and semester view of sales in units and give insights:

Plotting a month-wise bar graph for weekly sales to get an idea about which month has the maximum sales and then will plot a year-wise bar graph for weekly sales to know which year has the highest weekly sales. Also will plot the semester-wise bar graph for weekly sales to get some insights about the semester's weekly sales.

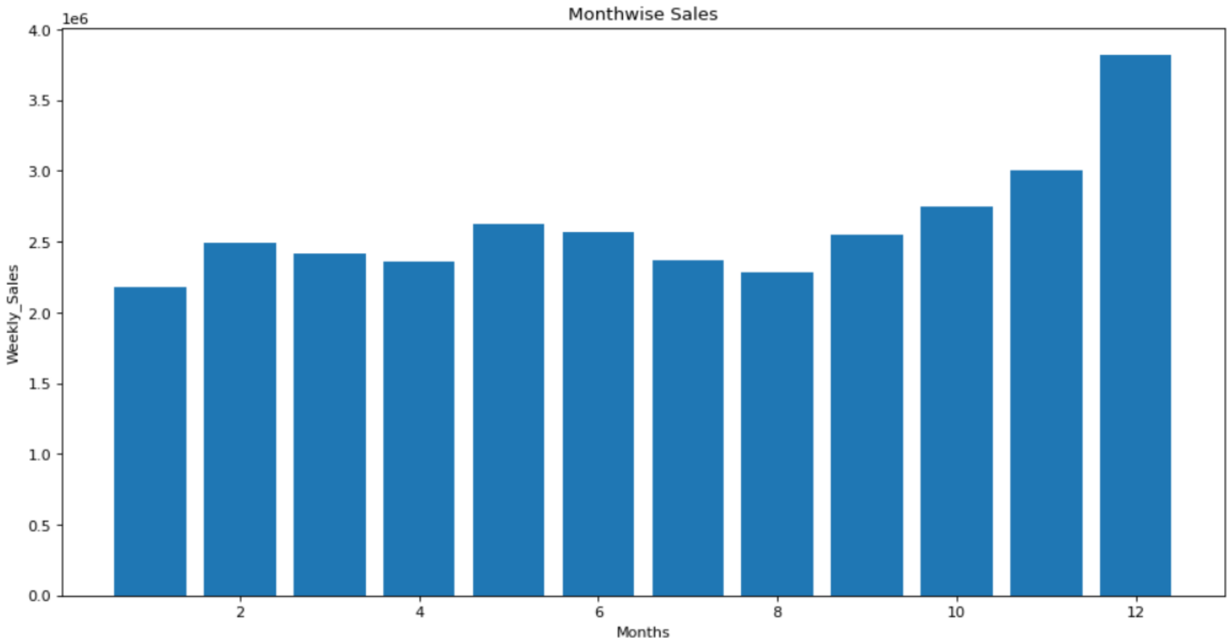
Month wise sales:

```
# Monthwise Sales

plt.figure(figsize=(14,7), dpi=80)
plt.bar(data['Month'], data['Weekly_Sales'])
plt.xlabel('Months')
plt.ylabel('Weekly_Sales')
plt.title('Monthwise Sales')
```

Y

```
Text(0.5, 1.0, 'Monthwise Sales')
```

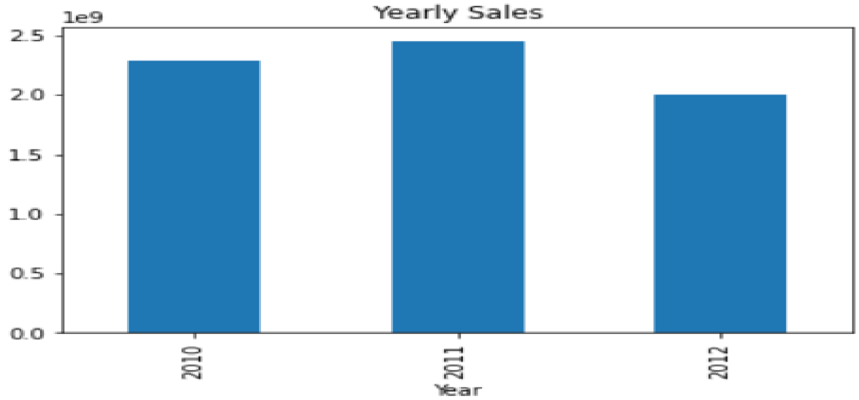


Yearly sales:

```
# Yearly Sales

plt.figure(figsize=(10,7), dpi=80)
data.groupby('Year')[['Weekly_Sales']].sum().plot(kind='bar', legend=False)
plt.title('Yearly Sales')
```

```
Text(0.5, 1.0, 'Yearly Sales')
<Figure size 800x560 with 0 Axes>
```

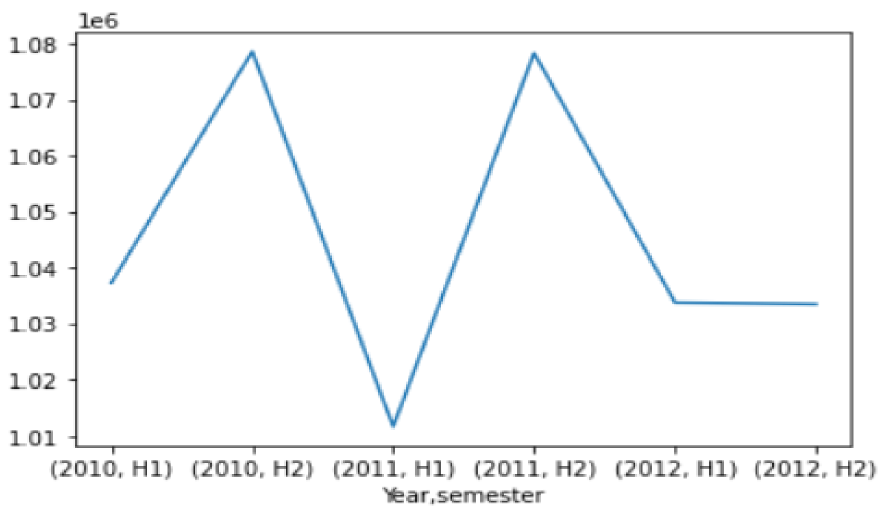


Semester wise sales:

```
#Semesterwise Sales
data ['semester'] = np.where(data['Date'].dt.month.le(6),'H1','H2')
```

```
data.groupby(['Year','semester'])['Weekly_Sales'].mean().plot()
```

```
<AxesSubplot: xlabel='Year,semester'>
```



Insights:

1) Year 2010 has the highest sales and 2012 has the lowest sales.

(2) December month has the highest weekly sales.

(3) Semester 2 has the highest weekly sales.

(4) Year 2011 has the highest weekly sales.

Part-2 - **Model Building**

Build prediction models to forecast demand (Modeling)

(1) Statistical Model

```
import statsmodels.formula.api as sm
model=sm.ols('Weekly_Sales~CPI+Unemployment+Fuel_Price+Temperature+Holiday_Flag',data=data).fit()
model.summary()
```

OLS Regression Results

Dep. Variable:	Weekly_Sales	**R-squared:**	0.025
Model:	OLS	**Adj. R-squared:**	0.025
Method:	Least Squares	**F-statistic:**	33.57
Date:	Wed, 17 Apr 2024	**Prob (F-statistic):**	5.93e-34
Time:	13:31:23	**Log-Likelihood:**	-94269.
No. Observations:	6435	**AIC:**	1.886e+05
Df Residuals:	6429	**BIC:**	1.886e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	1.727e+06	7.98e+04	21.646	0.000	1.57e+06	1.88e+06
CPI	-1598.8717	195.127	-8.194	0.000	-1981.385	-1216.358
Unemployment	-4.155e+04	3972.660	-10.460	0.000	-4.93e+04	-3.38e+04
Fuel_Price	-1.017e+04	1.58e+04	-0.645	0.519	-4.11e+04	2.07e+04
Temperature	-724.1715	400.461	-1.808	0.071	-1509.207	60.864
Holiday_Flag	7.489e+04	2.76e+04	2.710	0.007	2.07e+04	1.29e+05

Omnibus:	365.109	**Durbin-Watson:**	0.114
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	430.212
Skew:	0.633	**Prob(JB):**	3.81e-94
Kurtosis:	3.007	**Cond. No.**	2.16e+03

The statistical model does not give a good accuracy and a lot of data manipulation needs to be done to get a good accuracy.

Defining dependent and independent variables. Here, store, fuel price, CPI, unemployment, day, month, and year are the independent variables and weekly sales is the dependent variable. Now, it's time to train the model. Import train_test_split from sklearn.model_selection and train 80% of the data and test on the rest 20% of the data.

Train Test Split and Standardization:

```python
#Define independent and dependent variable
# Select features and target
x=data[['Store','Fuel_Price','CPI','Unemployment','Day','Month','Year']]
y=data['Weekly_Sales']
```

```python
from sklearn.model_selection import train_test_split
# Split data to train and test (0.80:0.20)
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2)
```

```python
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

(2) Linear Regression model

```python
# Import sklearn
from sklearn import metrics
from sklearn.linear_model import LinearRegression

print('Linear Regression \n')
print()
reg = LinearRegression()
reg.fit(x_train, y_train)
y_pred = reg.predict(x_test)
print('Accuracy:',reg.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)))
```

```
Linear Regression


Accuracy: 14.489679918130216
Mean Absolute Error: 424454.0765194912
Mean Squared Error: 265015737709.25616
Root Mean Squared Error: 514796.7926369163
```

3. Random Forest Regressor Model

```
# Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
print('Random Forest Regressor:')
print()
rfr = RandomForestRegressor(n_estimators = 400,max_depth=15,n_jobs=5)
rfr.fit(x_train,y_train)
y_pred=rfr.predict(x_test)
print('Accuracy:',rfr.score(x_test, y_test)*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)))
```

```
Random Forest Regressor:

Accuracy: 95.6738770422647
Mean Absolute Error: 66415.10584058112
Mean Squared Error: 13397009304.230692
Root Mean Squared Error: 115745.45046882272
```

Here, we evaluated 3 algorithms to predict weekly sales. Statistical model and linear regression showed low accuracy, while Random Forest Regression achieved almost 95% accuracy. Thus, Random Forest Regression is the best model for forecasting weekly sales.

Change dates into days by creating new variable.

```
#Change dates into days by creating new variable.
data['day'] = pd.to_datetime(data['Date']).dt.day_name()
data.head()
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Day	Month	Year	semester	exp_day	day
0	1	2010-05-02	1643690.90	0	42.31	2.572	211.096358	8.106	2	5	2010	H1	117	Sunday
1	1	2010-12-02	1641957.44	1	38.51	2.548	211.242170	8.106	2	12	2010	H2	331	Thursday
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106	19	2	2010	H1	45	Friday
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106	26	2	2010	H1	52	Friday
4	1	2010-05-03	1554806.68	0	46.50	2.625	211.350143	8.106	3	5	2010	H1	118	Monday

```
data['day'].head(10)
```

```
0      Sunday
1    Thursday
2      Friday
3      Friday
4      Monday
5      Friday
6      Friday
7      Friday
8    Thursday
9    Saturday
Name: day, dtype: object
```