Retail Analysis with Walmart Data

Abstract

One of the leading retail stores in the US, Walmart, would like to predict the sales and demand accurately. There are certain events which impact sales on each day. Using the data of 45 stores provided, build a forecast model which gives best accuracy.

Presented by: Ankita Agarwal

Problem Statement:

One of the leading retail stores in the US, Walmart, would like to predict the sales and demand accurately. There are certain events and holidays which impact sales on each day. There are sales data available for 45 stores of Walmart. The business is facing a challenge due to unforeseen demands and runs out of stock sometimes, due to the inappropriate machine learning algorithm. An ideal ML algorithm will predict demand at different points of time covering seasonality and ingest factors like economic conditions including CPI, Unemployment Index, etc.

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.

Detailed description of the given dataset:

This is the historical data which covers sales from 2010-02-05 to 2012-11-01 with 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

Unemployment: Prevailing unemployment rate

Holiday Events: Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13 Labour Day: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13 Thanksgiving: 26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13 Christmas: 31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13

To Analyze:

Basic Statistics tasks: -

1. Which store has maximum sales.
2. Which store has a 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 a 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 a non-holiday season for all stores together.
5. Provide a monthly and semester view of sales in units and give insights.

Statistical Model: -

For Store 1 – Build prediction models to forecast demand. Select the model which gives best accuracy.

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.

Analysis and Interpretations:

Basic Statistics tasks: -

1. Which store has maximum sales.

We aggregate the Weekly Sales by Store as shown below.

```
#Maximum Sales - groupby Store and sum the sales

df1 = df.groupby(['Store']).agg({'Weekly_Sales': 'sum'})
df1["%"] = df1.apply(lambda x:  100*x / x.sum()).applymap('{:.2f}%'.format)
df1.head()
```

Store	Weekly_Sales	%
1	2.224028e+08	3.30%
2	2.753824e+08	4.09%
3	5.758674e+07	0.85%
4	2.995440e+08	4.45%
5	4.547569e+07	0.67%

```
max_storeSales = df1.max()['Weekly_Sales']
max_storeSales
```

```
301397792.46000004
```

```
print(df1[df1.Weekly_Sales == df1.Weekly_Sales.max()])
```

```
       Weekly_Sales      %
Store
20     3.013978e+08  4.47%
```

Interpretation:

As can be clearly seen Store 20 has the maximum sales of 301397792.46 in the given time period.

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

In order to calculate the standard deviation and Coefficient of mean to standard deviation. We calculate the SD using the groupby function and aggregating the Weekly Sales using the std call. We then calculate the mean and SD of Weekly Sales to get CoV.

```
#Maximum Standard Deviation

df2 = df.groupby(['Store']).agg({'Weekly_Sales': 'std'})
max_storeSales = df2.max()['Weekly_Sales']
max_storeSales
print(df2[df2.Weekly_Sales == df2.Weekly_Sales.max()])
```

```
        Weekly_Sales
Store
14     317569.949476
```

```
#Coefficient of Variation - the coefficient of mean to standard deviation
#Coefficient of Variation = Standard deviation / mean

SD = df1.std()['Weekly_Sales']
Mean = df1.mean()['Weekly_Sales']
CoV = "{:.2%}".format(SD/Mean)
print(CoV)
```

```
52.21%
```

Interpretation:

It can clearly be seen that Store 14 has the maximum deviation from mean of sales for the given time period. Also, the coefficient of variation is 52.2% i.e., the standard deviation is 52.2% of the mean.

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

In order to calculate Growth Rate of Q3 in 2012 over Q2, we first aggregated the weekly sales grouping it by "yr_qr". "yr_qr" variable was created at the initial stages of the project using datetime library. The we take the subset of data with only Q2 and Q3 of 2012 and find the growth rate using the formula: -

Growth = (2012_Q3/2012_Q2)-1

```
#good quarterly growth rate in Q3'2012

#Maximum Sales - groupby Store and sum the sales

df3 = df.groupby(['Store','yr_qr']).agg({'Weekly_Sales': 'sum'})
df3.sort_values("yr_qr", axis = 0, ascending = True,
                inplace = True, na_position ='last')

max_QtrSales = df3.max()['Weekly_Sales']
print(df3[df3.Weekly_Sales == df3.Weekly_Sales.max()])
```

```
               Weekly_Sales
Store yr_qr
20    2010_Q4   32573122.65
```

```
qtrs=['2012_Q2','2012_Q3']
sol_df=df[df.yr_qr.isin(qtrs)]
sol_df.head()

df4=pd.DataFrame(sol_df.groupby(['Store','yr_qr'])['Weekly_Sales'].sum())
df4.reset_index(inplace=True)

# Reshaping the data frame from long to wide format
df5=df4.pivot(index='Store', columns='yr_qr', values='Weekly_Sales')

df5['Growth'] = (df5['2012_Q3']/df5['2012_Q2'])-1
df5.head()
```

yr_qr	2012_Q2	2012_Q3	Growth
Store			
1	20978760.12	20253947.78	-0.034550
2	25083604.88	24303354.86	-0.031106
3	5620316.49	5298005.47	-0.057347
4	28454363.67	27796792.46	-0.023110
5	4466363.69	4163790.99	-0.067745

```
Store_max_growth = df5.max()['Growth']
print(df5[df5.Growth == df5.Growth.max()])
```

```
yr_qr     2012_Q2     2012_Q3    Growth
Store
7      7290859.27  8262787.39  0.133308
```

Interpretation:

Store 20 has the max sales of 32573123 in Q4'2010. However, as per the problem statement, the quarterly growth rate in Q3'2012 is the highest in Store 7 with a quarterly growth rate of 0.133.

4. Some holidays have a negative impact on sales. Find out holidays which have higher sales than the mean sales in a non-holiday season for all stores together.

Firstly, we create two subsets, one each for a holiday and non-holiday season. We then find the aggregate mean grouping Year and month. We then merge both the dataframe and find the difference between the Sales with respect to Year and Month and check

which dates had higher sales on a holiday as compared to non-holiday days in the same month.

```
#holidays which have higher sales than the mean sales - Creating subsets and finding means

holiday_df=df.loc[df['Holiday_Flag']==1]
Nonholiday_df=df.loc[df['Holiday_Flag']==0]

Nonholiday_df=pd.DataFrame(Nonholiday_df.groupby(['Year','Month'])['Weekly_Sales'].mean())
holiday_df=pd.DataFrame(holiday_df.groupby(['Year','Month'])['Weekly_Sales'].mean())
```

```
# holidays which have higher sales than the mean sales - merge and difference
Holiday_sales_impact= pd.merge(Nonholiday_df,
                 holiday_df,
                 on=['Year', 'Month'],
                 how='inner')

Holiday_sales_impact.rename(columns={'Weekly_Sales_x':'Average_Sales_Non-Holiday', 'Weekly_Sales_y':'Average_Sales_Holiday'} , inplace

Holiday_sales_impact['Difference'] = Holiday_sales_impact["Average_Sales_Holiday"] > Holiday_sales_impact["Average_Sales_Non-Holiday"]

Holiday_sales_impact
```

Year	Month	Average_Sales_Non-Holiday	Average_Sales_Holiday	Difference
2010	Dec	1.379600e+06	8.985004e+05	False
	Feb	1.051824e+06	1.074148e+06	True
	Nov	1.015055e+06	1.462689e+06	True
	Sep	9.750630e+05	1.014098e+06	True
2011	Dec	1.344642e+06	1.023166e+06	False
	Feb	1.029594e+06	1.051915e+06	True
	Nov	1.063472e+06	1.479858e+06	True
	Sep	9.671362e+05	1.039183e+06	True
2012	Feb	1.052253e+06	1.111320e+06	True
	Sep	9.801147e+05	1.074001e+06	True

Interpretation:

Fairly easy to conclude here. The Dates or Holidays associated with it that have higher sales as compared to the mean sales on a Non-Holiday season are 25-11-2011 and 26-11-2010 (both Thanksgiving).

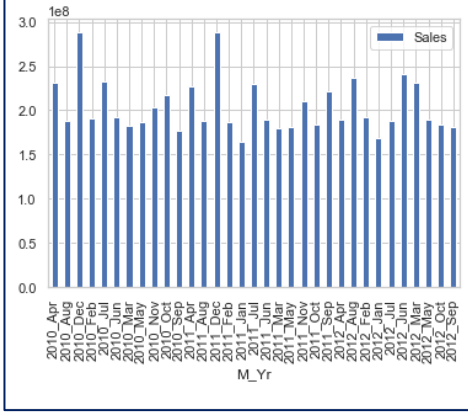
5. Provide a monthly and semester view of sales in units and give insights.

Date was converted to Month, Semester and Year variables at the initial stages using the datetime library. We then aggregate these variables and sum together the weekly sales to get the monthly and semester views. We shall also create visualizations for the same.

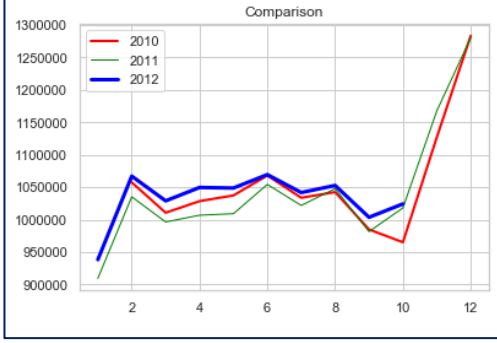
```
#monthly and semester view of sales in units and give insights

#Montly_Sales_Trend
Monthly_Sales = df.groupby(['Year','Month']).agg(['sum'])['Weekly_Sales'].reset_index().rename(columns={'sum':'Sales'})
Monthly_Sales['M_Yr'] = Monthly_Sales['Year'].astype(str) + '_' + Monthly_Sales['Month'].astype(str)
Monthly_Sales.head()
Monthly_Sales.plot.bar(x='M_Yr', y='Sales')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xf83937adc8>
```




```
plt.plot(x.loc[2010, :], label = "2010", color = "red", linewidth = 2)
plt.plot(x.loc[2011, :], label = "2011", color = "green", linewidth = 1)
plt.plot(x.loc[2012, :], label = "2012", color = "blue", linewidth=3)
plt.legend(loc = "best")
plt.title("Comparison")
plt.show()
```




```
max_MonthlySales = Monthly_Sales.max()['Sales']
print(Monthly_Sales[Monthly_Sales.Sales == Monthly_Sales.Sales.max()])
```

```
   Year Month         Sales      M_Yr
2  2010   Dec  2.887605e+08  2010_Dec
```

```
#Montly_Sales_Trend - Store wise

Monthly_Sales = df.groupby(['Store','Year','Month']).agg(['sum'])['Weekly_Sales'].reset_index().rename(columns={'sum':'Sales'}
Monthly_Sales['M_Yr'] = Monthly_Sales['Year'].astype(str) + '_' + Monthly_Sales['Month'].astype(str)
max_MonthlySalesStore = Monthly_Sales.max()['Sales']
print(Monthly_Sales[Monthly_Sales.Sales == Monthly_Sales.Sales.max()])
```

```
     Store  Year Month        Sales      M_Yr
629     20  2010   Dec  13553791.64  2010_Dec
```

Interpretation:

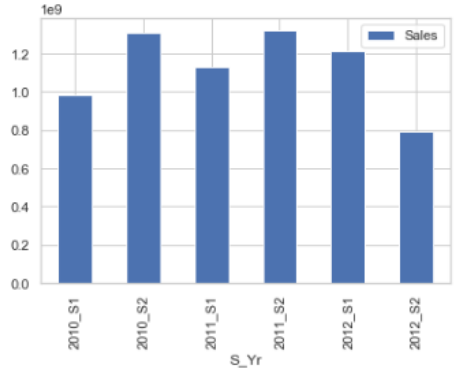
Monthly view - The maximum sales of 288760533 is in the month of Dec 2010. Store 20 has the maximum sales of 13553792 in the month on Dec 2010 when compared to all store sales monthly data.

```
#semester field
conditions = [
    (df['Quarter'] == 1) | (df['Quarter'] == 2) ,
    (df['Quarter'] == 3) | (df['Quarter'] == 4)
]
choices = [1, 2]
df['Semester'] = np.select(conditions, choices)

#Semester_Sales_Trend
Semester_Sales = df.groupby(['Year','Semester']).agg(['sum'])['Weekly_Sales'].reset_index().rename(columns={'sum
Semester_Sales['S_Yr'] = Semester_Sales['Year'].astype(str) + '_' + 'S' + Semester_Sales['Semester'].astype(str)

Semester_Sales.plot.bar(x='S_Yr', y='Sales')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xf83a4c8688>
```




```
max_SemesterSales = Semester_Sales.max()['Sales']
print(Semester_Sales[Semester_Sales.Sales == Semester_Sales.Sales.max()])
```

```
   Year  Semester         Sales     S_Yr
3  2011         2  1.320860e+09  2011_S2
```

```
#Semester_Sales_Trend - Storewise

Semester_Sales = df.groupby(['Store','Year','Semester']).agg(['sum'])['Weekly_Sales'].reset_index().rename(columns={'sum':'Sal
Semester_Sales['S_Yr'] = Semester_Sales['Year'].astype(str) + '_' + 'S' + Semester_Sales['Semester'].astype(str)
max_SemesterSalesStore = Semester_Sales.max()['Sales']
print(Semester_Sales[Semester_Sales.Sales == Semester_Sales.Sales.max()])
```

```
    Store  Year  Semester        Sales     S_Yr
21      4  2011         2  60366595.85  2011_S2
```

Semester view - The maximum sales of 1320860210 is in semester 2 of 2011. Store 4 has the maximum sales of 60366596 in S2 of 2011 when compared to all store sales semester data.

Statistical Model: -

For Store 1 – Build prediction models to forecast demand

Create new variables Day and WeekNum and a subset for Store 1 from the data provided.

6. 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.

To analyze if there is a relation between CPI, unemployment, fuel price and Weekly Sales, we first check the correlation of these variables with Weekly Sales. Furthermore, we split the data into train and test data and perform OLS regression on the train data and check accuracy using the test.

```
# B. Statistical Model - For Store 1 - Build prediction models to forecast demand
#Change dates into days by creating new variable.

df['Day'] = df.Date.apply(lambda x: x.day)
df['WeekNum'] = pd.cut(df.Day, bins = [1,7,14,21,28, 31], labels = [1,2,3,4,5], include_lowest=True)
df.drop(['Date','yr_qr','Month'], axis = 1, inplace = True)
df = df.astype(np.float64)
df.head()
```

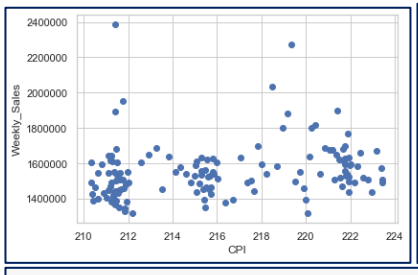
```
#Creating a subset for Store 1
Store1_df=df.loc[df['Store']==1]
Store1_df=Store1_df[['Weekly_Sales','Day','Month1','WeekNum','Year', 'Fuel_Price', 'CPI', 'Unemployment','Holiday_Flag', 'Temp
```

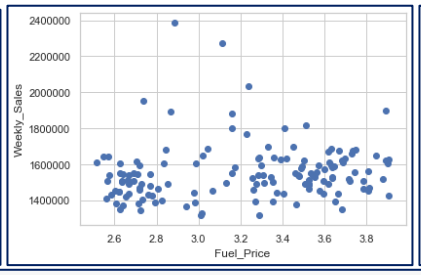
```
Store1_df.corr()
```

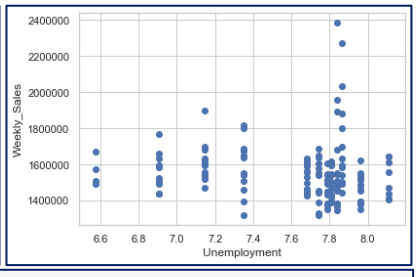
	Weekly_Sales	Day	Month1	WeekNum	Year	Fuel_Price	CPI	Unemployment	Holiday_Flag	Temperature
Weekly_Sales	1.000000	-0.271685	0.202188	-0.249095	0.152396	0.124592	0.225408	-0.097955	0.194905	-0.222701
Day	-0.271685	1.000000	0.015192	0.975592	0.006406	0.030806	0.033588	-0.018342	0.044526	0.051077
Month1	0.202188	0.015192	1.000000	0.047315	-0.194465	-0.101256	0.050952	0.040821	0.122996	0.246417
WeekNum	-0.249095	0.975592	0.047315	1.000000	-0.010680	0.033077	0.024007	0.015458	0.045056	0.081582
Year	0.152396	0.006406	-0.194465	-0.010680	1.000000	0.809769	0.948141	-0.798149	-0.056783	0.068843
Fuel_Price	0.124592	0.030806	-0.101256	0.033077	0.809769	1.000000	0.755259	-0.513944	-0.085903	0.228493
CPI	0.225408	0.033588	0.050952	0.024007	0.948141	0.755259	1.000000	-0.813471	-0.028919	0.118503
Unemployment	-0.097955	-0.018342	0.040821	0.015458	-0.798149	-0.513944	-0.813471	1.000000	0.082949	-0.180695
Holiday_Flag	0.194905	0.044526	0.122996	0.045056	-0.056783	-0.085903	-0.028919	0.082949	1.000000	-0.200543
Temperature	-0.222701	0.051077	0.246417	0.081582	0.068843	0.228493	0.118503	-0.180695	-0.200543	1.000000

```
def scatter(Store1_df, column):
    plt.figure()
    plt.scatter(Store1_df[column] , Store1_df['Weekly_Sales'])
    plt.ylabel('Weekly_Sales')
    plt.xlabel(column)

scatter(Store1_df, 'Fuel_Price')
scatter(Store1_df, 'CPI')
scatter(Store1_df, 'Unemployment')
```



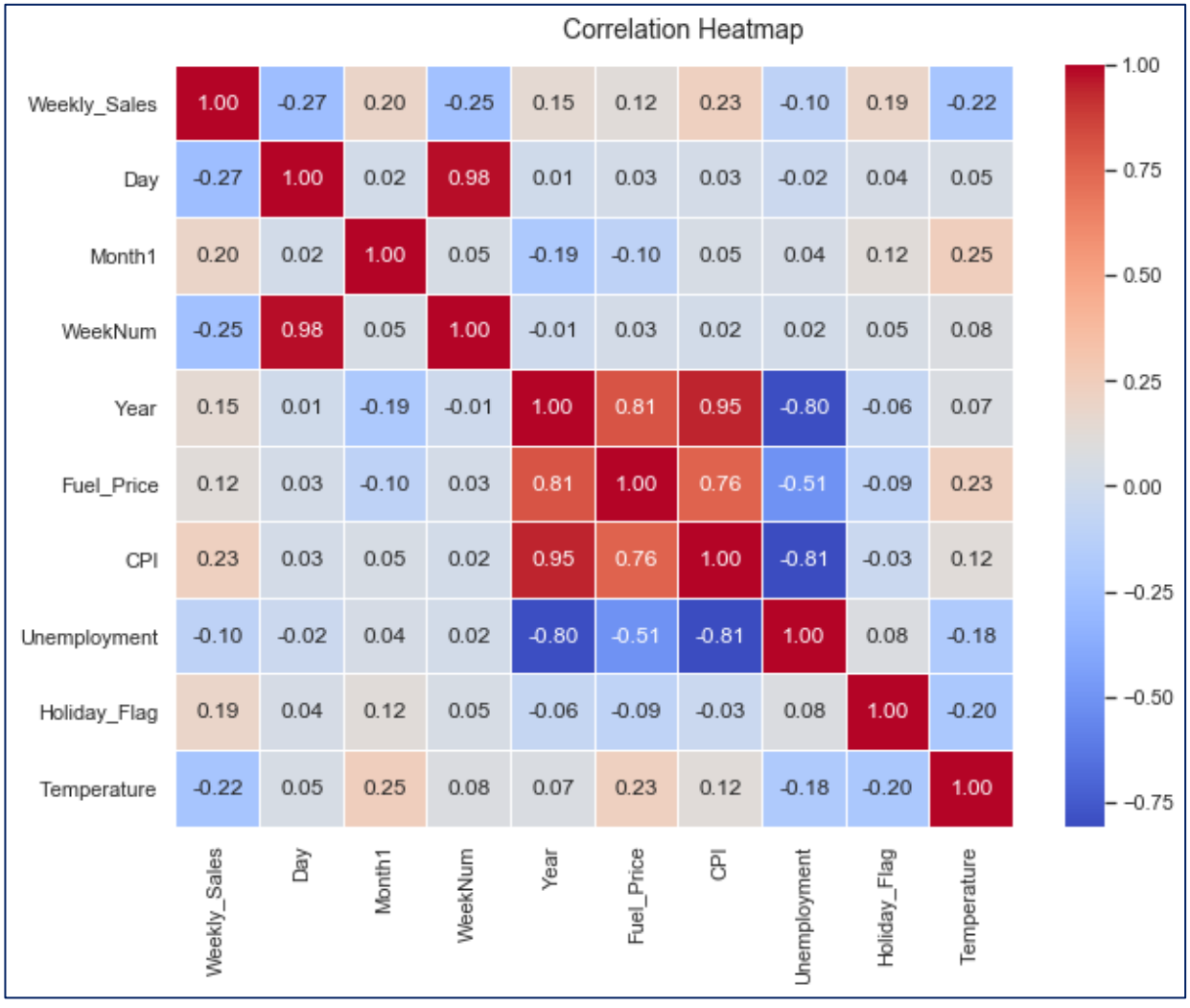





```
# Correlation Matrix Heatmap

f, ax = plt.subplots(figsize=(10, 7))
corr = Store1_df.corr()

hm = sns.heatmap(round(corr,2), annot=True, ax=ax, cmap="coolwarm",fmt='.2f',
                 linewidths=.05)
f.subplots_adjust(top=0.93)
t= f.suptitle('Correlation Heatmap', fontsize=14)
```



```
features=Store1_df[['Weekly_Sales','Day','Month1','Year', 'CPI', 'Unemployment','Holiday_Flag', 'Temperature']]
features.head()
```

	Weekly_Sales	Day	Month1	Year	CPI	Unemployment	Holiday_Flag	Temperature
0	1643690.90	5.0	2.0	2010.0	211.096358	8.106	0.0	42.31
1	1641957.44	12.0	2.0	2010.0	211.242170	8.106	1.0	38.51
2	1611968.17	19.0	2.0	2010.0	211.289143	8.106	0.0	39.93
3	1409727.59	26.0	2.0	2010.0	211.319643	8.106	0.0	46.63
4	1554806.68	5.0	3.0	2010.0	211.350143	8.106	0.0	46.50

```
#Splitting data into train and test

train,test = train_test_split(features,test_size=0.2,random_state=39)
```

```
#Linear Regression - Hypothesize if CPI, unemployment, and fuel price have any impact on sales.

lm = smf.ols(formula='Weekly_Sales ~ Temperature + Holiday_Flag  + CPI ' , data=train).fit()
lm.summary()
```

OLS Regression Results

Dep. Variable:	Weekly_Sales	**R-squared:**	0.196
Model:	OLS	**Adj. R-squared:**	0.174
Method:	Least Squares	**F-statistic:**	8.935
Date:	Sat, 06 Jun 2020	**Prob (F-statistic):**	2.39e-05
Time:	11:33:57	**Log-Likelihood:**	-1503.6
No. Observations:	114	**AIC:**	3015.
Df Residuals:	110	**BIC:**	3026.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.474e+05	6.32e+05	-0.707	0.481	-1.7e+06	8.06e+05
Temperature	-2779.9958	915.429	-3.037	0.003	-4594.161	-965.831
Holiday_Flag	1.064e+05	4.96e+04	2.148	0.034	8220.541	2.05e+05
CPI	1.009e+04	2930.164	3.442	0.001	4279.643	1.59e+04

Omnibus:	43.875	**Durbin-Watson:**	1.907
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	139.782
Skew:	1.350	**Prob(JB):**	4.43e-31
Kurtosis:	7.705	**Cond. No.**	1.16e+04

Interpretation:

As we can see that the R-squared value and Adjusted R-squared value is less than 20%, this means that less than 20% of the variation in the dependent variable is explained by the independent variables. The model suggests that the independent variables are not significant to the dependent variable. From the above correlation matrix, heatmap and the OLS regression model, we can say that there is a no relation between CPI, unemployment, fuel price and Weekly Sales.

7. The model which gives best accuracy –

To get the best accurate models, we check the data using Linear Regression, Decision Tree and Random Forest. However, Linear regression model is used here to predict the weekly sales for the test model.

```
##1.Linear Regression

clf = LinearRegression()

clf.fit(X_train, Y_train)

y_pred_lr=clf.predict(X_test)

acc_lr=round( clf.score(X_train, Y_train) * 100, 2)

print ("Accuracy:%i %% \n"%acc_lr)
```

```
Accuracy:100 %
```

```
##2. Random Forest

clf = RandomForestRegressor(n_estimators=100)

clf.fit(X_train, Y_train)

y_pred_rf=clf.predict(X_test)

acc_rf= round(clf.score(X_train, Y_train) * 100, 2)

print ("Accuracy: %i %% \n"%acc_rf)
```

```
Accuracy: 99 %
```

```
##3. Decision Tree

clf=DecisionTreeRegressor()

clf.fit(X_train, Y_train)

y_pred_dt= clf.predict(X_test)

acc_dt = round( clf.score(X_train, Y_train) * 100, 2)

print ("Accuracy: %i %% \n"%acc_dt)
```

```
Accuracy: 100 %
```

```
##Comparing Models - Let's compare the accuracy score of all the regression models used above.

models = pd.DataFrame({
    'Model': ['Linear Regression','Random Forest','Decision Tree'],

    'Score': [acc_lr, acc_rf,acc_dt]
    })

models.sort_values(by='Score', ascending=False)
```

	Model	Score
0	Linear Regression	100.00
2	Decision Tree	100.00
1	Random Forest	99.41

```
#Linear Regression

linear_reg = LinearRegression()
linear_reg.fit(X_train,Y_train)

print("Intercept: ", linear_reg.intercept_)
print("Coefficient: " , linear_reg.coef_)
```

```
Intercept:  -1.862645149230957e-09
Coefficient:  [ 1.00000000e+00  1.03982846e-13 -8.81859399e-14  9.10074316e-13]
```

```
y_pred = linear_reg.predict(X_test)
print(y_pred)
```

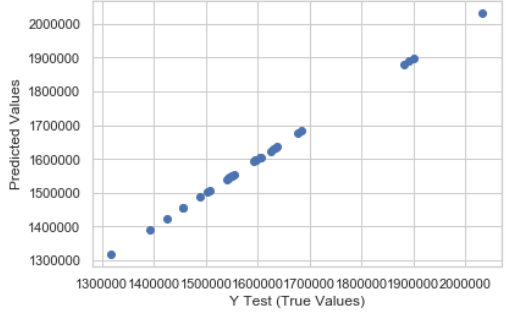
```
[1684519.99 1503284.06 1550229.22 1881176.67 1636263.41 1425100.71
 1592409.97 1629391.28 1604775.58 1542561.09 1316899.31 1624383.75
 2033320.66 1597868.05 1550369.92 1455090.69 1508237.76 1635078.41
 1540421.49 1488538.09 1391256.12 1595901.87 1677472.78 1899676.88
 1456800.28 1891034.93 1545418.53 1554806.68 1605491.78]
```

```
mse = metrics.mean_squared_error(Y_test,y_pred)
rmse = np.sqrt(mse)
print('%.2f'%rmse)
```

```
0.00
```

```
plt.scatter(Y_test,y_pred)
plt.xlabel('Y Test (True Values)')
plt.ylabel('Predicted Values')
```

```
Text(0, 0.5, 'Predicted Values')
```




```
print('MAE:', '%.2f'%metrics.mean_absolute_error(Y_test,y_pred))
print('MSE:', '%.2f'%metrics.mean_squared_error(Y_test,y_pred))
print('RMSE:', '%.2f'%np.sqrt(metrics.mean_squared_error(Y_test,y_pred)))
```

```
MAE: 0.00
MSE: 0.00
RMSE: 0.00
```

Interpretation:

We can clearly see that linear regression is the best model for the given set of Data with RSME, MAE and MAE equal to 0.

Programming Codes:

Adobe Acrobat Document

----------The End----------