Walmart_Store_Project

April 23, 2024

[3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.metrics import accuracy_score
import sklearn
import seaborn as sns
import statsmodels.formula.api as smf
```

[4]:
```
data = pd.read_csv("Walmart_Store_sales.csv")
```

[5]:
```
data.head()
```

[5]:

	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

[6]:
```
data.tail()
```

[6]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
6430	45	28-09-2012	713173.95	0	64.88	3.997		
6431	45	05-10-2012	733455.07	0	64.89	3.985		
6432	45	12-10-2012	734464.36	0	54.47	4.000		
6433	45	19-10-2012	718125.53	0	56.47	3.969		
6434	45	26-10-2012	760281.43	0	58.85	3.882		

```
6430  192.013558         8.684
6431  192.170412         8.667
6432  192.327265         8.667
6433  192.330854         8.667
6434  192.308899         8.667
```

[7]:
```
#EDA (Exploratory Data analysis | data Audit)
data.shape
```

[7]:
```
(6435, 8)
```

[8]:
```
data.columns
```

[8]:
```
Index(['Store', 'Date', 'Weekly_Sales', 'Holiday_Flag', 'Temperature',
       'Fuel_Price', 'CPI', 'Unemployment'],
      dtype='object')
```

[9]:
```
data
```

[9]:
```
      Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0         1  05-02-2010    1643690.90             0        42.31       2.572
1         1  12-02-2010    1641957.44             1        38.51       2.548
2         1  19-02-2010    1611968.17             0        39.93       2.514
3         1  26-02-2010    1409727.59             0        46.63       2.561
4         1  05-03-2010    1554806.68             0        46.50       2.625
…     …          …             …            …          …          …
6430     45  28-09-2012     713173.95             0        64.88       3.997
6431     45  05-10-2012     733455.07             0        64.89       3.985
6432     45  12-10-2012     734464.36             0        54.47       4.000
6433     45  19-10-2012     718125.53             0        56.47       3.969
6434     45  26-10-2012     760281.43             0        58.85       3.882

             CPI  Unemployment
0     211.096358         8.106
1     211.242170         8.106
2     211.289143         8.106
3     211.319643         8.106
4     211.350143         8.106
…          …            …
6430  192.013558         8.684
6431  192.170412         8.667
6432  192.327265         8.667
6433  192.330854         8.667
6434  192.308899         8.667

[6435 rows x 8 columns]
```

[10]:
```
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   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
```

[11]:
```
data.describe().T
```

[11]:

	count	mean	std	min	25%	\
Store	6435.0	2.300000e+01	12.988182	1.000	12.000	
Weekly_Sales	6435.0	1.046965e+06	564366.622054	209986.250	553350.105	
Holiday_Flag	6435.0	6.993007e-02	0.255049	0.000	0.000	
Temperature	6435.0	6.066378e+01	18.444933	-2.060	47.460	
Fuel_Price	6435.0	3.358607e+00	0.459020	2.472	2.933	
CPI	6435.0	1.715784e+02	39.356712	126.064	131.735	
Unemployment	6435.0	7.999151e+00	1.875885	3.879	6.891	

	50%	75%	max
Store	23.000000	3.400000e+01	4.500000e+01
Weekly_Sales	960746.040000	1.420159e+06	3.818686e+06
Holiday_Flag	0.000000	0.000000e+00	1.000000e+00
Temperature	62.670000	7.494000e+01	1.001400e+02
Fuel_Price	3.445000	3.735000e+00	4.468000e+00
CPI	182.616521	2.127433e+02	2.272328e+02
Unemployment	7.874000	8.622000e+00	1.431300e+01

[13]:
```
#Removing Dte column inorder to apply function, to get detailed info.
data1 = data[["Store", "Weekly_Sales", "Holiday_Flag",␣
 ↪"Temperature","Fuel_Price","CPI","Unemployment"]]
data1.head()
```

[13]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	\
0	1	1643690.90	0	42.31	2.572	211.096358	
1	1	1641957.44	1	38.51	2.548	211.242170	
2	1	1611968.17	0	39.93	2.514	211.289143	

```
3      1    1409727.59             0        46.63       2.561  211.319643
4      1    1554806.68             0        46.50       2.625  211.350143

   Unemployment
0         8.106
1         8.106
2         8.106
3         8.106
4         8.106
```

[14]:
```
def var_summary(x):
    return pd.Series([x.count(), x.isnull().sum(), x.sum(), x.mean(), x.
 ↪median(),
                      x.std(), x.var(), x.min(), x.quantile(0.01), x.quantile(0.
 ↪05),
                      x.quantile(0.10), x.quantile(0.25), x.quantile(0.50),
                      x.quantile(0.75), x.quantile(0.90), x.quantile(0.95),
                      x.quantile(0.99), x.max()],
                     index=['N', 'NMISS', 'SUM', 'MEAN', 'MEDIAN', 'STD', 'VAR',
                            'MIN', 'P1', 'P5', 'P10', 'P25', 'P50', 'P75',
                            'P90', 'P95', 'P99', 'MAX'])
```

[15]:
```
data1.apply(lambda x:var_summary(x)).T
```

[15]:

	N	NMISS	SUM	MEAN	MEDIAN	\
Store	6435.0	0.0	1.480050e+05	2.300000e+01	23.000000	
Weekly_Sales	6435.0	0.0	6.737219e+09	1.046965e+06	960746.040000	
Holiday_Flag	6435.0	0.0	4.500000e+02	6.993007e-02	0.000000	
Temperature	6435.0	0.0	3.903714e+05	6.066378e+01	62.670000	
Fuel_Price	6435.0	0.0	2.161264e+04	3.358607e+00	3.445000	
CPI	6435.0	0.0	1.104107e+06	1.715784e+02	182.616521	
Unemployment	6435.0	0.0	5.147454e+04	7.999151e+00	7.874000	

	STD	VAR	MIN	P1	\
Store	12.988182	1.686929e+02	1.000	1.000000	
Weekly_Sales	564366.622054	3.185097e+11	209986.250	253103.068600	
Holiday_Flag	0.255049	6.504996e-02	0.000	0.000000	
Temperature	18.444933	3.402155e+02	-2.060	18.523600	
Fuel_Price	0.459020	2.106991e-01	2.472	2.565000	
CPI	39.356712	1.548951e+03	126.064	126.106903	
Unemployment	1.875885	3.518944e+00	3.879	4.156000	

	P5	P10	P25	P50	\
Store	3.00000	5.000000	12.000	23.000000	
Weekly_Sales	308426.68900	384125.462000	553350.105	960746.040000	
Holiday_Flag	0.00000	0.000000	0.000	0.000000	
Temperature	27.73000	34.560000	47.460	62.670000	

```
Fuel_Price          2.64200       2.720000       2.933       3.445000
CPI               126.49129     128.512193     131.735     182.616521
Unemployment        5.32600       6.061000       6.891       7.874000

                         P75           P90           P95           P99  \
Store           3.400000e+01  4.100000e+01  4.300000e+01  4.500000e+01
Weekly_Sales    1.420159e+06  1.887626e+06  2.049179e+06  2.404035e+06
Holiday_Flag    0.000000e+00  0.000000e+00  1.000000e+00  1.000000e+00
Temperature     7.494000e+01  8.399200e+01  8.766300e+01  9.319000e+01
Fuel_Price      3.735000e+00  3.916000e+00  4.029000e+00  4.203000e+00
CPI             2.127433e+02  2.195341e+02  2.219267e+02  2.254702e+02
Unemployment    8.622000e+00  9.863000e+00  1.218700e+01  1.418000e+01

                         MAX
Store           4.500000e+01
Weekly_Sales    3.818686e+06
Holiday_Flag    1.000000e+00
Temperature     1.001400e+02
Fuel_Price      4.468000e+00
CPI             2.272328e+02
Unemployment    1.431300e+01
```

[16]:
```
#Checking missing values in weekly sales
#Then fill them with mean
data1['Weekly_Sales'] = data1['Weekly_Sales'].fillna(data1['Weekly_Sales'].
 ↪mean())


data1
```

```
/tmp/ipykernel_236/2044636604.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data1['Weekly_Sales'] =
data1['Weekly_Sales'].fillna(data1['Weekly_Sales'].mean())
```

[16]:
```
      Store  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price         CPI  \
0         1    1643690.90             0        42.31       2.572  211.096358
1         1    1641957.44             1        38.51       2.548  211.242170
2         1    1611968.17             0        39.93       2.514  211.289143
3         1    1409727.59             0        46.63       2.561  211.319643
4         1    1554806.68             0        46.50       2.625  211.350143
…     …            …            …           …          …          …
6430     45     713173.95             0        64.88       3.997  192.013558
```

```
6431     45     733455.07            0        64.89       3.985  192.170412
6432     45     734464.36            0        54.47       4.000  192.327265
6433     45     718125.53            0        56.47       3.969  192.330854
6434     45     760281.43            0        58.85       3.882  192.308899

      Unemployment
0            8.106
1            8.106
2            8.106
3            8.106
4            8.106
…            …
6430         8.684
6431         8.667
6432         8.667
6433         8.667
6434         8.667

[6435 rows x 7 columns]
```

1.Which store has maximum sales

[17]:
```
#Which store maximum sales
Maxsales_store=data1.groupby("Store").Weekly_Sales.sum()
Maxsales_store
print("store {} has max number of sales of amount {}".format(Maxsales_store.
  ↪idxmax(),Maxsales_store.max()) )
```

```
store 20 has max number of sales of amount 301397792.46
```

2.Store that has maximum standard deviation i.e, the sales vary a lot also, find out the coefficient of mean to standard deviation

[18]:
```
#Which store has maximum standard deviation
Maxstd_store=data.groupby("Store").Weekly_Sales.std()
Maxstd_store
print("store {} has max standard deviation {}".format(Maxstd_store.
  ↪idxmax(),Maxstd_store.max()) )
```

```
store 14 has max standard deviation 317569.9494755081
```

[19]:
```
#Coefficient of mean to standard deviation
Maxstd_store1=data.groupby("Store").Weekly_Sales.agg(['mean','std'])
Maxstd_store1
```

[19]:
```
              mean            std
Store
1      1.555264e+06  155980.767761
2      1.925751e+06  237683.694682
```

3	4.027044e+05	46319.631557
4	2.094713e+06	266201.442297
5	3.180118e+05	37737.965745
6	1.564728e+06	212525.855862
7	5.706173e+05	112585.469220
8	9.087495e+05	106280.829881
9	5.439806e+05	69028.666585
10	1.899425e+06	302262.062504
11	1.356383e+06	165833.887863
12	1.009002e+06	139166.871880
13	2.003620e+06	265506.995776
14	2.020978e+06	317569.949476
15	6.233125e+05	120538.652043
16	5.192477e+05	85769.680133
17	8.935814e+05	112162.936087
18	1.084718e+06	176641.510839
19	1.444999e+06	191722.638730
20	2.107677e+06	275900.562742
21	7.560691e+05	128752.812853
22	1.028501e+06	161251.350631
23	1.389864e+06	249788.038068
24	1.356755e+06	167745.677567
25	7.067215e+05	112976.788600
26	1.002912e+06	110431.288141
27	1.775216e+06	239930.135688
28	1.323522e+06	181758.967539
29	5.394514e+05	99120.136596
30	4.385796e+05	22809.665590
31	1.395901e+06	125855.942933
32	1.166568e+06	138017.252087
33	2.598617e+05	24132.927322
34	9.667816e+05	104630.164676
35	9.197250e+05	211243.457791
36	3.735120e+05	60725.173579
37	5.189003e+05	21837.461190
38	3.857317e+05	42768.169450
39	1.450668e+06	217466.454833
40	9.641280e+05	119002.112858
41	1.268125e+06	187907.162766
42	5.564039e+05	50262.925530
43	6.333247e+05	40598.413260
44	3.027489e+05	24762.832015
45	7.859814e+05	130168.526635

3. store with a good quarterly growth rate in Q3'2012

[20]:
```
#Which store/s has good quarterly growth rate in Q3'2012
data['Date'] = pd.to_datetime(data['Date'])
data['Date']
data
dtq3 = data[(data['Date'] >= pd.to_datetime('07-01-2012')) & (data['Date'] <=␣
 ↪pd.to_datetime('09-30-2012'))]
dtq3
```

```
/tmp/ipykernel_236/1281302458.py:2: UserWarning: Parsing dates in DD/MM/YYYY
format when dayfirst=False (the default) was specified. This may lead to
inconsistently parsed dates! Specify a format to ensure consistent parsing.
  data['Date'] = pd.to_datetime(data['Date'])
```

[20]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	\
109	1	2012-09-03	1675431.16	0	58.76	3.669	
122	1	2012-08-06	1697230.96	0	78.30	3.452	
127	1	2012-07-13	1527014.04	0	77.12	3.256	
128	1	2012-07-20	1497954.76	0	80.42	3.311	
129	1	2012-07-27	1439123.71	0	82.66	3.407	
…	…	…	…	…	…	…	
6426	45	2012-08-31	734297.87	0	75.09	3.867	
6427	45	2012-07-09	766512.66	1	75.70	3.911	
6428	45	2012-09-14	702238.27	0	67.87	3.948	
6429	45	2012-09-21	723086.20	0	65.32	4.038	
6430	45	2012-09-28	713173.95	0	64.88	3.997	

	CPI	Unemployment
109	221.059189	7.348
122	221.749484	7.143
127	221.924158	6.908
128	221.932727	6.908
129	221.941295	6.908
…	…	…
6426	191.461281	8.684
6427	191.577676	8.684
6428	191.699850	8.684
6429	191.856704	8.684
6430	192.013558	8.684

[540 rows x 8 columns]

[21]:
```
growthrateQ3=dtq3.groupby("Store").Weekly_Sales.sum()
print("Store Number {} has Good Quartely Growth in Q3'2012 {}".
 ↪format(growthrateQ3.idxmax(),growthrateQ3.max()))
```

```
Store Number 4 has Good Quartely Growth in Q3'2012 25652119.35
```

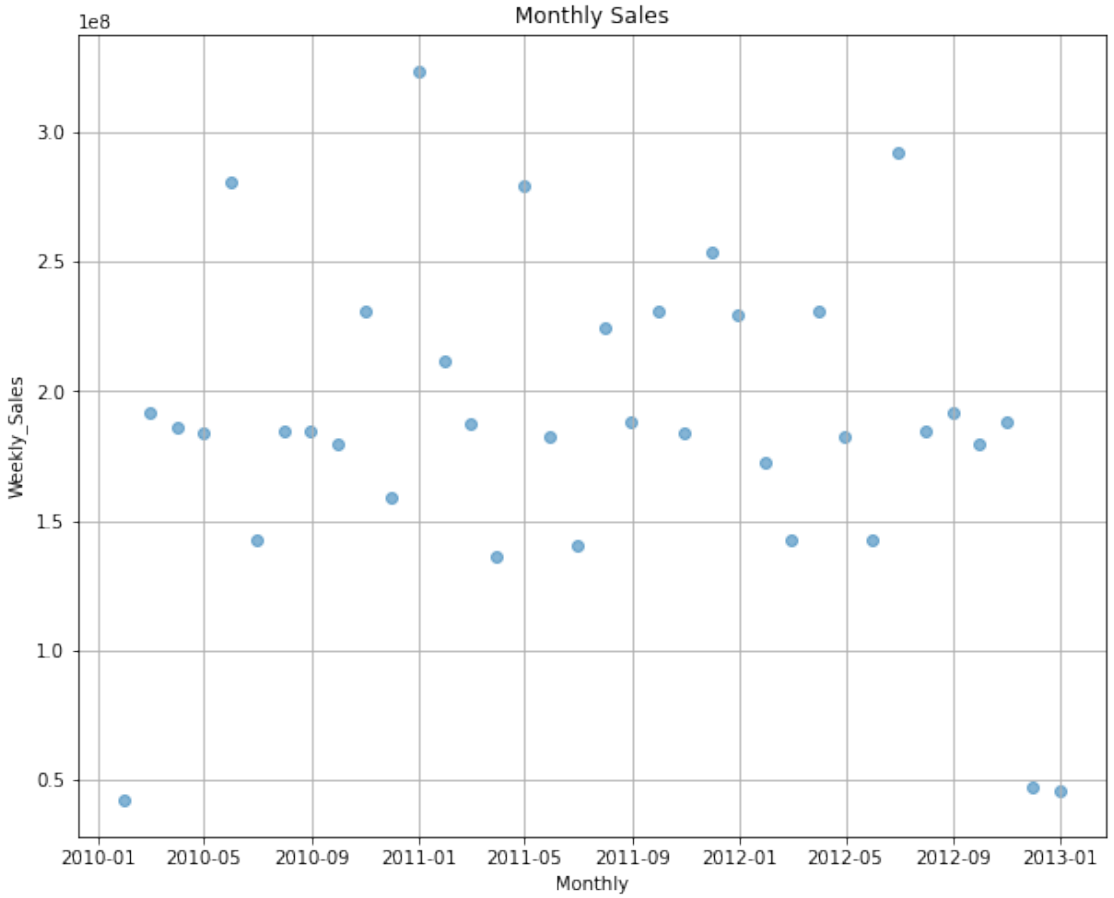
4.Holiday with the higher sales than the mean sales in non-holiday season for all stores

[22]:
```
# Define holiday dates
christmas1 = pd.Timestamp(2010, 12, 31)
christmas2 = pd.Timestamp(2011, 12, 30)
christmas3 = pd.Timestamp(2012, 12, 28)
christmas4 = pd.Timestamp(2013, 12, 27)
thanksgiving1 = pd.Timestamp(2010, 11, 26)
thanksgiving2 = pd.Timestamp(2011, 11, 25)
thanksgiving3 = pd.Timestamp(2012, 11, 23)
thanksgiving4 = pd.Timestamp(2013, 11, 29)
labourDay1 = pd.Timestamp(2010, 2, 10)
labourDay2 = pd.Timestamp(2011, 2, 9)
labourDay3 = pd.Timestamp(2012, 2, 7)
labourDay4 = pd.Timestamp(2013, 2, 6)
superBowl1 = pd.Timestamp(2010, 9, 12)
superBowl2 = pd.Timestamp(2011, 9, 11)
superBowl3 = pd.Timestamp(2012, 9, 10)
superBowl4 = pd.Timestamp(2013, 9, 8)
# Calculate the mean sales during the holidays
christmas_mean_sales = data[(data['Date'] == christmas1) | (data['Date'] ==␣
 ↪christmas2) | (data['Date'] == christmas3) | (data['Date'] ==␣
 ↪christmas4)]['Weekly_Sales'].mean()
thanksgiving_mean_sales = data[(data['Date'] == thanksgiving1) | (data['Date']␣
 ↪== thanksgiving2) | (data['Date'] == thanksgiving3) | (data['Date'] ==␣
 ↪thanksgiving4)]['Weekly_Sales'].mean()
labourDay_mean_sales = data[(data['Date'] == labourDay1) | (data['Date'] ==␣
 ↪labourDay2) | (data['Date'] == labourDay3) | (data['Date'] ==␣
 ↪labourDay4)]['Weekly_Sales'].mean()
superBowl_mean_sales = data[(data['Date'] == superBowl1) | (data['Date'] ==␣
 ↪superBowl2) | (data['Date'] == superBowl3) | (data['Date'] ==␣
 ↪superBowl4)]['Weekly_Sales'].mean()

# Calculate the mean sales during non-holiday seasons
non_holiday_mean_sales = data[data['Holiday_Flag'] == 0]['Weekly_Sales'].mean()

# Find holidays with higher sales than the mean sales in the non-holiday season
List_of_mean_sales = {
    'Christmas_mean_sales': round(christmas_mean_sales, 2),
    'Thanksgiving_mean_sales': round(thanksgiving_mean_sales, 2),
    'LabourDay_mean_sales': round(labourDay_mean_sales, 2),
    'SuperBowl_mean_sales': round(superBowl_mean_sales, 2) if␣
 ↪superBowl_mean_sales > non_holiday_mean_sales else None,
    'Non holiday weekly sales': round(non_holiday_mean_sales, 2)
}
List_of_mean_sales
```

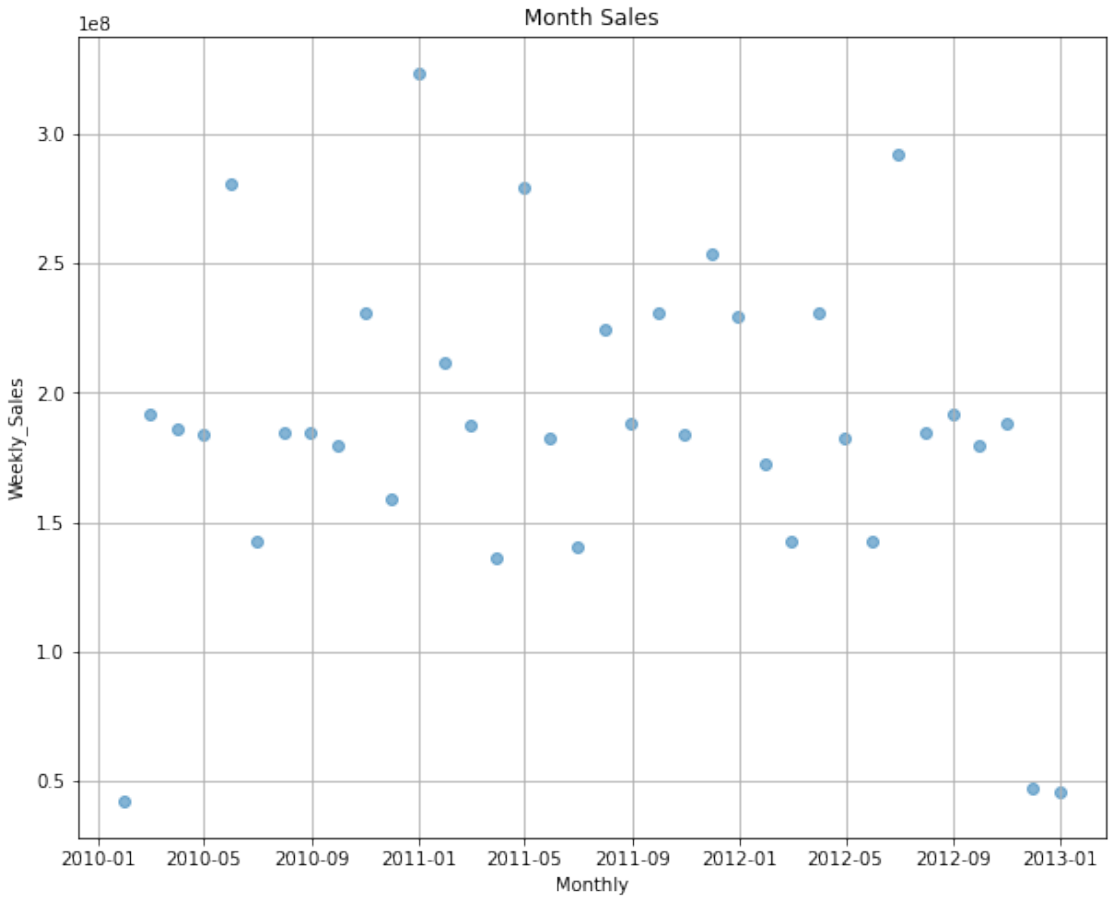
[22]:
```
{'Christmas_mean_sales': 960833.11,
 'Thanksgiving_mean_sales': 1471273.43,
 'LabourDay_mean_sales': 1008369.41,
 'SuperBowl_mean_sales': None,
 'Non holiday weekly sales': 1041256.38}
```

PROVIDE A MONTHLY AND SEMESTER VIEW OF SALES IN UNITS AND GIVE INSIGHTS

[23]:
```
#Monthly:
monthly = data.groupby(pd.Grouper(key='Date', freq='1M')).sum()
monthly
monthly=monthly.reset_index()
monthly
plt.figure(figsize=(10,8))
plt.scatter(monthly.Date,monthly.Weekly_Sales,alpha=0.55)
plt.title('Monthly Sales')
plt.xlabel('Monthly')
plt.ylabel('Weekly_Sales')
plt.grid(True)
plt.show()
```



[24]:
```
#semester sales
Semester = data.groupby(pd.Grouper(key='Date', freq='6M')).sum()
Semester = Semester.reset_index()
Semester
plt.figure(figsize=(10,8))
plt.scatter(monthly.Date,monthly.Weekly_Sales,alpha=0.55)
plt.title('Month Sales')
plt.xlabel('Monthly')
plt.ylabel('Weekly_Sales')
plt.grid(True)
plt.show()
```



We can see from semester sales graph that at beginning of lst semester of 2010 and the 1st semester of 2013 sales are lowest

OUTPUT OR RESULT OF BASIC STATISTIC TASKS:

1.Store 20 has max no. of amount 301397792.46

2. Store 14 has max standard deviation of 317569.9494.

3.Store 4 has good quaeterly growth rate Q3 2012=25652119.35

4. Thanksgiving days has higher sales when compared to non-holiday.

5. from the monthly sales graph the highest sum of sales is recorded in between jan-2011 to march-2011

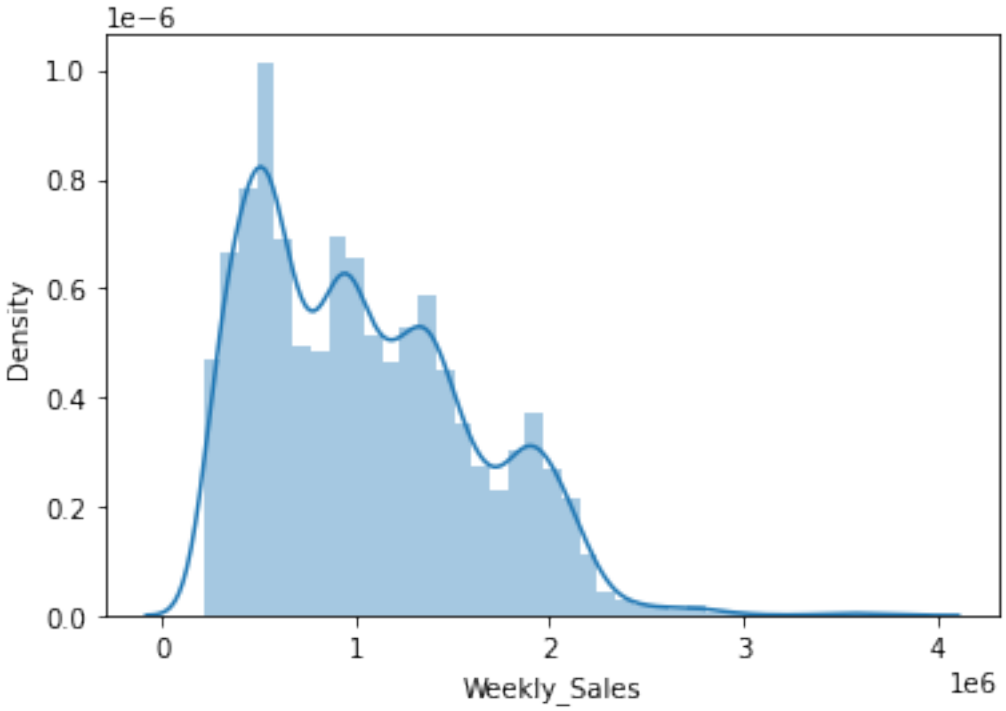
Statistical Model

-Linear Regression

[25]:
```
sns.distplot( data.Weekly_Sales ) #To check the normality of y[Sales]
```

```
/tmp/ipykernel_236/3932051586.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot( data.Weekly_Sales ) #To check the normality of y[Sales]
```

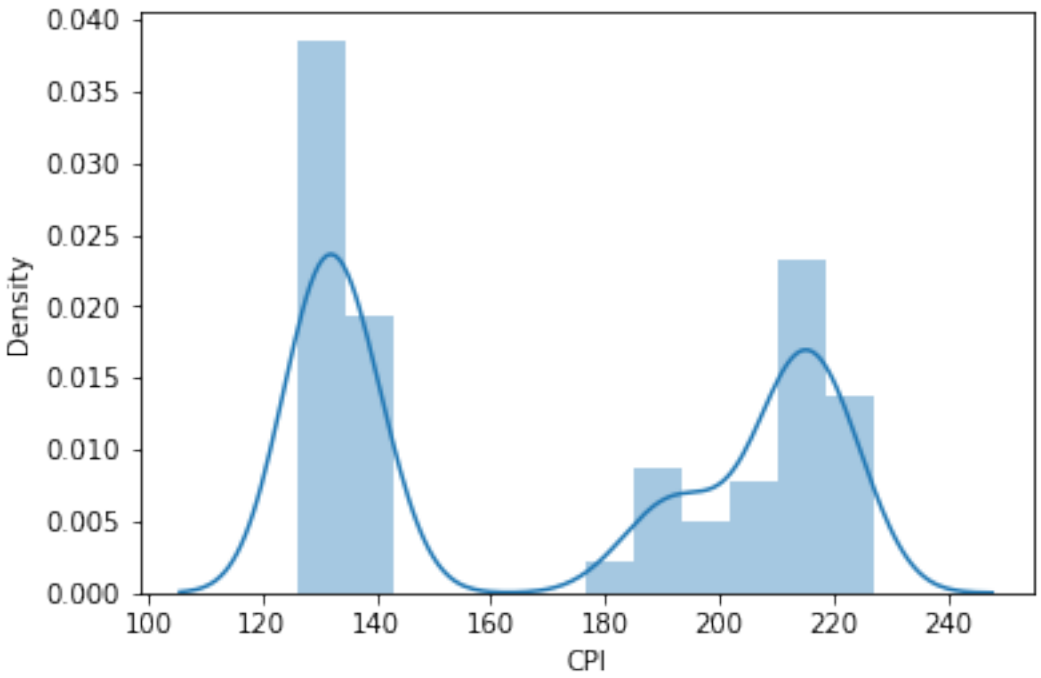
[25]: `<AxesSubplot: xlabel='Weekly_Sales', ylabel='Density'>`



[26]:
```
#checking the normality of CPI
sns.distplot( data.CPI )
```

```
/tmp/ipykernel_236/3251223310.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot( data.CPI )
```

[26]: <AxesSubplot: xlabel='CPI', ylabel='Density'>

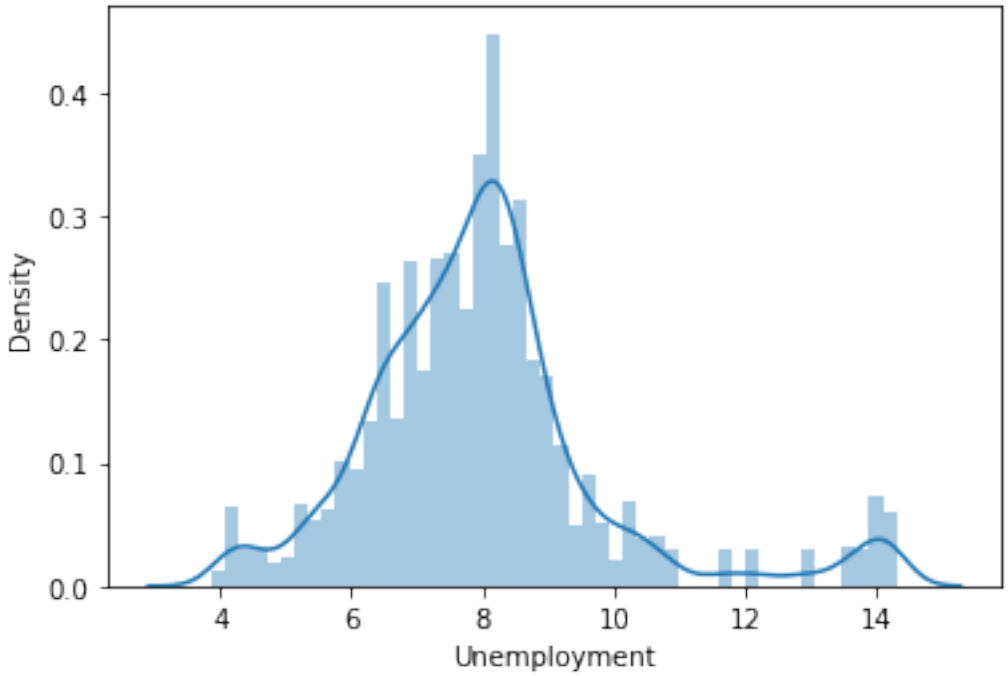


[27]:
```
#Checking the normality of unemployment
sns.distplot( data.Unemployment )
```

```
/tmp/ipykernel_236/2312161371.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```

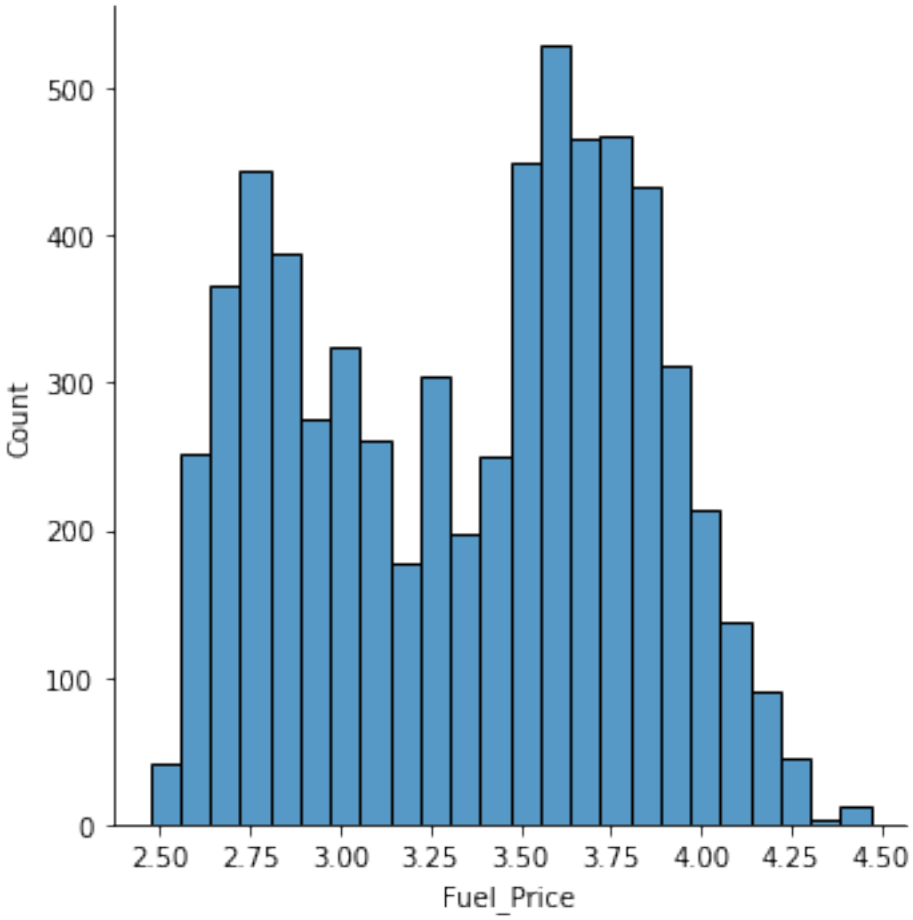
```
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot( data.Unemployment )
```

[27]: <AxesSubplot: xlabel='Unemployment', ylabel='Density'>



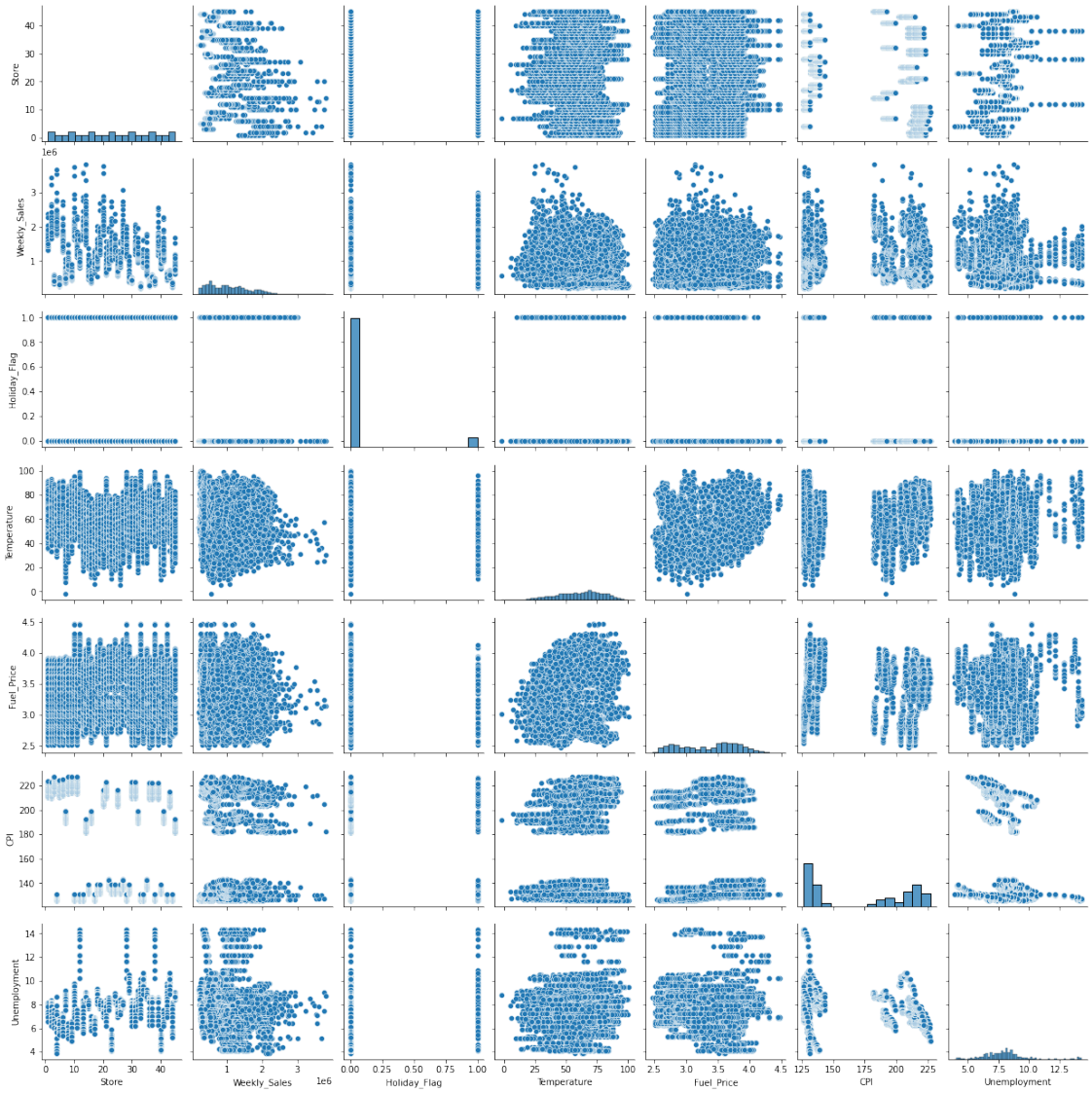
[28]:
```
#Checking normality in Fuel_Price
sns.displot( data.Fuel_Price )
```

[28]: <seaborn.axisgrid.FacetGrid at 0x7fecfd58dcc0>



[29]:
```
#Visualizing the pairwise correlation
sns.pairplot( data )
```

[29]: <seaborn.axisgrid.PairGrid at 0x7fecc1aa92d0>



[30]:
```
#Finding the correlation the given data
data.corr()
```

```
/tmp/ipykernel_236/1800032011.py:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  data.corr()
```

[30]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	\
Store	1.000000e+00	-0.335332	-4.386841e-16	-0.022659	
Weekly_Sales	-3.353320e-01	1.000000	3.689097e-02	-0.063810	
Holiday_Flag	-4.386841e-16	0.036891	1.000000e+00	-0.155091	

```
Temperature  -2.265908e-02     -0.063810 -1.550913e-01     1.000000
Fuel_Price    6.002295e-02      0.009464 -7.834652e-02     0.144982
CPI          -2.094919e-01     -0.072634 -2.162091e-03     0.176888
Unemployment  2.235313e-01     -0.106176  1.096028e-02     0.101158

              Fuel_Price       CPI  Unemployment
Store           0.060023 -0.209492      0.223531
Weekly_Sales    0.009464 -0.072634     -0.106176
Holiday_Flag   -0.078347 -0.002162      0.010960
Temperature     0.144982  0.176888      0.101158
Fuel_Price      1.000000 -0.170642     -0.034684
CPI            -0.170642  1.000000     -0.302020
Unemployment   -0.034684 -0.302020      1.000000
```

BUILDING REGRESSION MODEL

[31]:
```
#Building a model
data4=data[["Weekly_Sales","Fuel_Price","CPI","Unemployment"]]
lm=smf.ols('Weekly_Sales ~ Fuel_Price+CPI+Unemployment', data4).fit()
lm.summary()
```

[31]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           Weekly_Sales   R-squared:                       0.024
Model:                            OLS   Adj. R-squared:                  0.023
Method:                 Least Squares   F-statistic:                     51.75
Date:                Tue, 23 Apr 2024   Prob (F-statistic):           4.81e-33
Time:                        07:19:50   Log-Likelihood:                -94275.
No. Observations:                6435   AIC:                         1.886e+05
Df Residuals:                    6431   BIC:                         1.886e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     1.746e+06   7.96e+04     21.938      0.000    1.59e+06     1.9e+06
Fuel_Price   -1.927e+04   1.54e+04     -1.248      0.212   -4.95e+04     1.1e+04
CPI          -1696.8760    188.793     -8.988      0.000   -2066.973   -1326.779
Unemployment -4.286e+04   3905.197    -10.975      0.000   -5.05e+04   -3.52e+04
==============================================================================
Omnibus:                      370.117   Durbin-Watson:                   0.112
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              436.792
Skew:                           0.638   Prob(JB):                     1.42e-95
Kurtosis:                       3.051   Cond. No.                     2.04e+03
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.04e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[32]:
```
lml=smf.ols('Weekly_Sales ~CPI+Unemployment', data4).fit()
lml.summary()
```

[32]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           Weekly_Sales   R-squared:                       0.023
Model:                            OLS   Adj. R-squared:                  0.023
Method:                 Least Squares   F-statistic:                     76.84
Date:                Tue, 23 Apr 2024   Prob (F-statistic):           1.05e-33
Time:                        07:20:14   Log-Likelihood:                -94276.
No. Observations:                6435   AIC:                         1.886e+05
Df Residuals:                    6432   BIC:                         1.886e+05
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      1.67e+06   5.12e+04     32.588      0.000    1.57e+06    1.77e+06
CPI          -1652.0937    185.358     -8.913      0.000   -2015.457   -1288.730
Unemployment -4.241e+04   3888.879    -10.906      0.000      -5e+04   -3.48e+04
==============================================================================
Omnibus:                      372.804   Durbin-Watson:                   0.112
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              440.398
Skew:                           0.640   Prob(JB):                     2.34e-96
Kurtosis:                       3.060   Cond. No.                     1.30e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.3e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[41]:
```
#Checking f values,model parameter ,confidence level and p-value
lml.f_pvalue
```

[41]:
```
1.0480758575885128e-33
```

[42]:
```
#Getting model parameters
lml.params
```

[42]:
```
Intercept       1.669688e+06
CPI            -1.652094e+03
Unemployment   -4.241192e+04
dtype: float64
```

[43]:
```
#Parameters at 95% confidence intervals
lml.conf_int()
```

[43]:
```
                            0             1
Intercept     1.569249e+06  1.770127e+06
CPI          -2.015457e+03 -1.288730e+03
Unemployment -5.003542e+04 -3.478843e+04
```

[41]:
```
#verifying parameter
lml.pvalues
```

[41]:
```
Intercept       9.627130e-216
CPI              6.375053e-19
Unemployment     1.875484e-27
dtype: float64
```

[44]:
```
#Evaluating model accuracy
lml.rsquared
```

[44]:
```
0.023336073889825504
```

[33]:
```
#Making predictions
ltmpredic = lml.predict(data4)
ltmpredic
```

[33]:
```
0       977145.827686
1       976904.933259
2       976827.329295
3       976776.940437
4       976726.551579
            …
6430    984158.310986
6431    984620.176498
6432    984361.039156
6433    984355.110122
6434    984391.382335
Length: 6435, dtype: float64
```

[34]:
```
#Calculate RMSE
from sklearn import metrics
mse = metrics.mean_squared_error( data4.Weekly_Sales, ltmpredic )
mse
rmse = np.sqrt( mse )
rmse
```

[34]: 557699.3609143773

CONCLUSION

Different between R-squared and Adj R squared value is less with improved F-statistic value.

-Change dates into days by creating new variable.

[37]:
```
data['Day'] = pd.to_datetime(data['Date']).dt.day_name()
data.head()
```

[37]:

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

RESULT OF STATISTICAL MODELS:

1.Parameters estimated are considered to be significant if p-value is less than 0.05

This shows intercept and cpi and unempployment are both significant parameters.and parameters estimated can be accepted.

So, linears model is Sales=B0+B1 CPI+B2 UNEMPLOYMENT

2.Inserted one more VARIABLE named DAY in DATASET.