Reliance _7-11_Sales_prediction_of_sales_Linear_Regression

June 7, 2023

1 Description

The sales prediction project employed linear regression to estimate sales based on several parameters. The initial step was importing the dataset and the necessary libraries for the project. Exploratory data analysis was performed to get insight from the data. This involved analysing category columns, visualising the sales distribution, and examining variations and outliers in numerical columns. As feature engineering techniques, one-hot encoding of categorical variables, feature extraction from dates, and data type correction were all applied. After outliers were identified and removed, the dataset was split into training and testing sets. Multiple linear regression was used to forecast sales, and recursive feature engineering improved the model's performance. The undertaking proved is capable of accurately and successfully hitting revenue projections by 90% on training data and 89% to test data.

==

1.Imported the required libraries.

2.Imported the dataset.

3.Explored the dataset by viewing its contents.

4.Analyzed the sales distribution.

5.Corrected the data type of the date column and extracted months, years, and weeks from the date.

6.Identified categorical and numericalccolumns and created bar graphs to analyze sales.

7.Created box plots to identify outliers in all numerical columns.

9.Created a plot to find the sum and mean of weekly sales with respect to working days, holidays, month and store..

10.Performed one-hot encoding and data scaling on categorical variables.

11.Removed outliers from the dataset, resulting in a removal of 7.94% of the data.

12.Split the dataset into training and testing datasets in an 80:20 ratio and stored the dependent and independent variables in x and y, respectively

13.Standardized the test dataset and transformed it using the training data.

14.Conducted recursive feature engineering to determine the number of features to be removed for maximum R-squared.

15.Plotted a scatter plot to compare the predicted values with the test values using multiple linear regression.
==

2 Insights

2.0.1 Targeted Marketing and Promotions:-

- We may spot patterns and trends that lead to increased sales by examining sales data by shop, month, and working day/holiday. In order to increase sales, use this information to create targeted marketing campaigns and specials for certain months or days. As an example, invest greater resources to marketing initiatives throughout certain months if sales are typically higher during those months.

2.0.2 Inventory management:-

- We may improve your inventory management by understanding the sales distribution and anomalies. We can make sure that there is enough stock available to fulfil customer demand by determining peak sales periods. In addition, by looking at outliers, we may spot any odd sales cycles and change the stock levels accordingly. Also be able to enhance efficiency and save money by reducing stockouts and surplus inventory.

2.0.3 Monitoring sales by shop:-

- This may help us pinpoint the best-performing locations as well as understand the elements that make them successful. Apply their plans and recommendations to underperforming stores by using this information to learn from them. Find any outliers or abnormalities in the sales for certain stores, and then look into the reasons why. We will be able to spread profitable ideas throughout all of your locations.

2.0.4 Focus on Key Drivers:-

- By doing a Recursive feature analysis, you may determine the variables that have the most impact on sales. Put your efforts on improving these key drivers. Consider dedicating more funds to enhancing particular product categories or promotions, for instance, if the study reveals that they have a significant impact on sales.

2.0.5 Sales Forecasting:-

- The project's sales prediction model can be a useful tool for making future forecasts. Continually improve and enhance the model using new information and user input. You can make wise decisions regarding manufacturing, staff, and resource allocation with the help of accurate sales forecasting.

==

1.Imported the required libraries.

[104]:
```
import math
import numpy as np
import pandas as pd
import seaborn as sns


from statsmodels.formula import api
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from IPython.display import display


from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

2.Imported the dataset.

[105]:
```
df=pd.read_csv('D:/DS/resume projects/7-eleven/seven_eleven_sales.csv')
```

[106]:
```
#making new DataFrame for data wrangling
dforg=df
```

3.Overviewing data

[107]:
```
df.head()
```

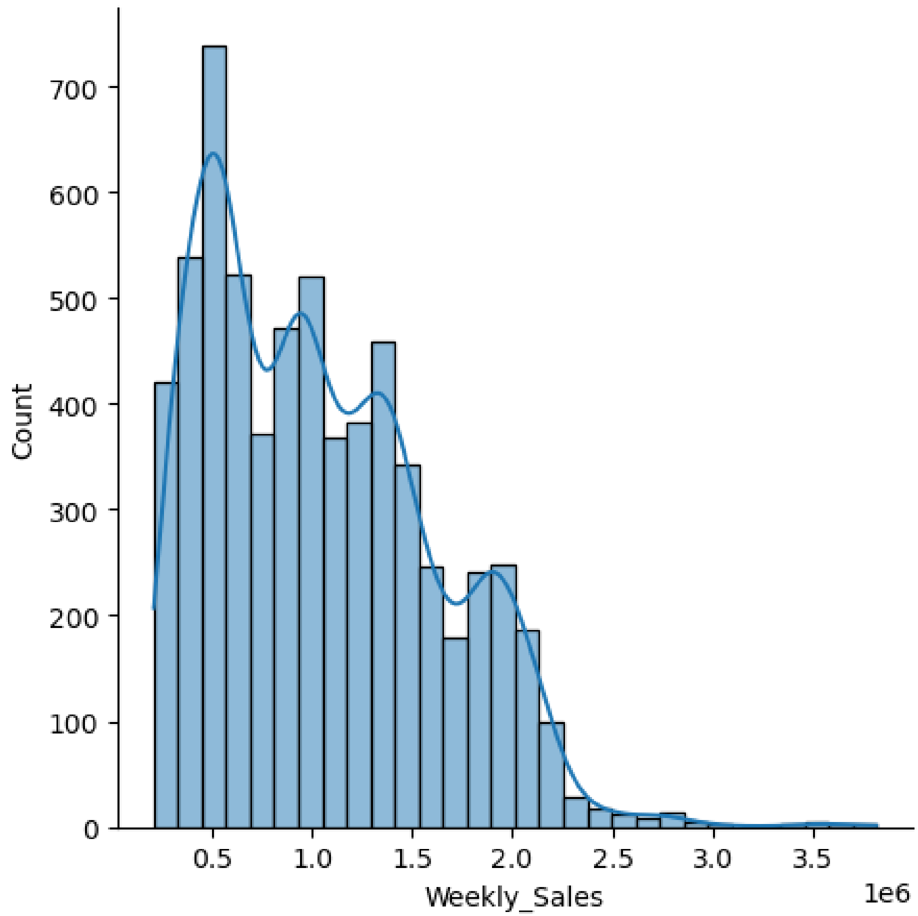
[107]:
```
   Store        Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0      1  05-02-2010    1643690.90             0        42.31       2.572
1      1  12-02-2010    1641957.44             1        38.51       2.548
2      1  19-02-2010    1611968.17             0        39.93       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

          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
```

4.Analyzed the sales distribution.

[108]:
```python
sns.displot(df['Weekly_Sales'],kde=True,bins=30)
```

[108]: <seaborn.axisgrid.FacetGrid at 0x13a33d08370>



5.Corrected the data type of the date column and extracted months, years, and weeks from the date.

[109]:
```python
df['Date']=pd.to_datetime(df['Date'])
df['month']=df['Date'].dt.month
df['Year']=df['Date'].dt.year
df['Week']=df['Date'].dt.weekday
df=df.drop(['Date'],axis=1)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Store         6435 non-null   int64  
 1   Weekly_Sales  6435 non-null   float64
 2   Holiday_Flag  6435 non-null   int64  
 3   Temperature   6435 non-null   float64
 4   Fuel_Price    6435 non-null   float64
 5   CPI           6435 non-null   float64
 6   Unemployment  6435 non-null   float64
 7   month         6435 non-null   int64  
 8   Year          6435 non-null   int64  
 9   Week          6435 non-null   int64  
dtypes: float64(5), int64(5)
memory usage: 502.9 KB
```

[110]:
```python
df.nunique().sort_values()
```

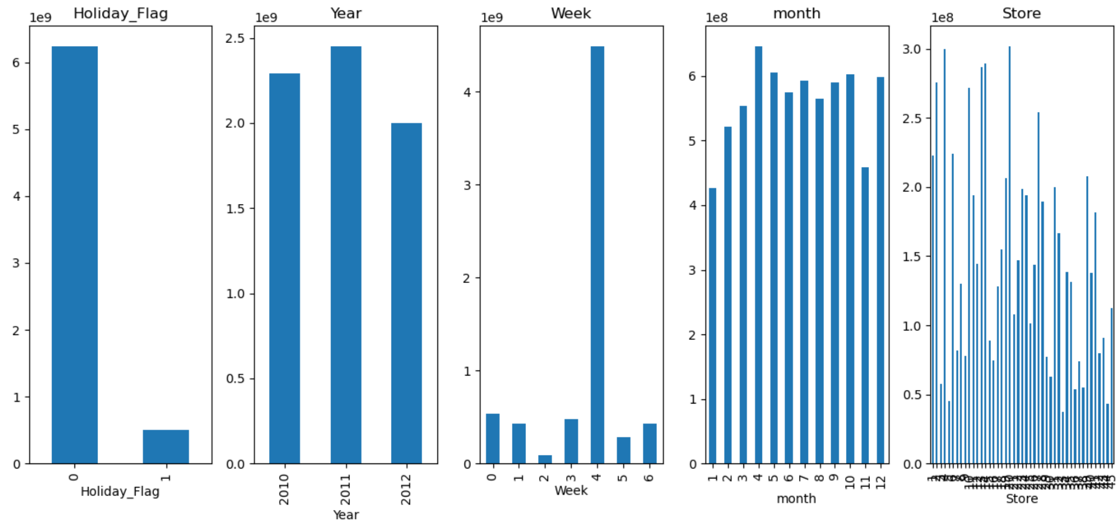
[110]:
```
Holiday_Flag       2
Year               3
Week               7
month             12
Store             45
Unemployment     349
Fuel_Price       892
CPI             2145
Temperature     3528
Weekly_Sales    6435
dtype: int64
```

6.Identified categorical and numericalccolumns and created bar graphs to analyze sales.

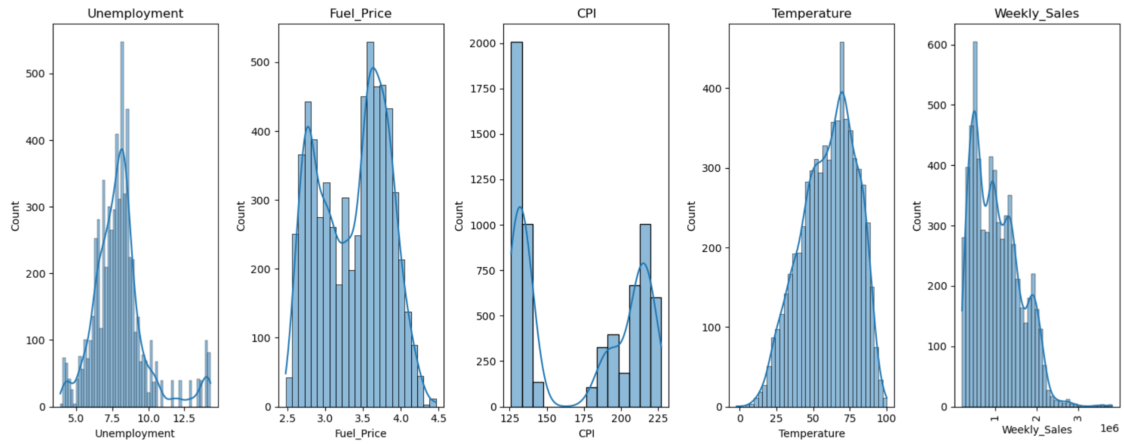
[111]:
```python
catcol=[]
numcol=[]

s=df.nunique().sort_values()

for i in s.index:
  if s[i]<=45: catcol.append(i)
  else: numcol.append(i)
```

[112]:
```python
plt.figure(figsize=(15,6))
for k,i in enumerate(catcol):
  plt.subplot(1,6,k+1)
  df.groupby(i)['Weekly_Sales'].sum().plot(kind='bar')
  plt.title(i)
plt.tight_layout()
```

```
plt.xticks(rotation=90)
plt.show()
```

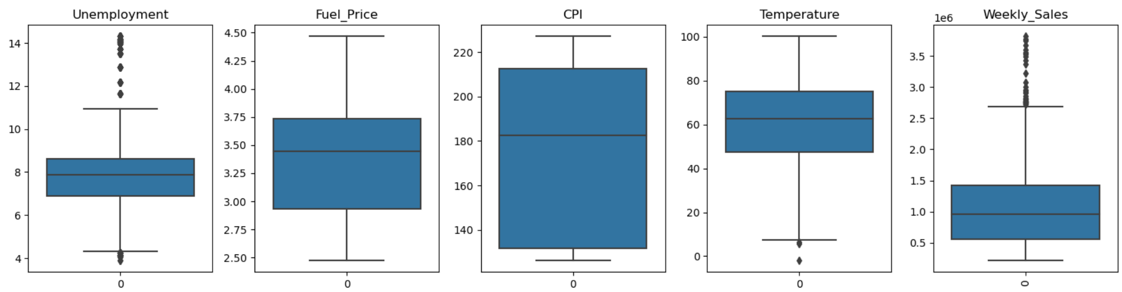


[113]:
```
plt.figure(figsize=(15,6))
for k,i in enumerate(numcol):
  plt.subplot(1,5,k+1)
  sns.histplot(df[i],kde=True)
  plt.title(i)
plt.tight_layout()
plt.xticks(rotation=90)
plt.show()
```



7.Created box plots to identify outliers in all numerical columns.

[114]:
```
plt.figure(figsize=(15,4))
for k,i in enumerate(numcol):
  plt.subplot(1,5,k+1)
  sns.boxplot(df[i])
  plt.title(i)
plt.tight_layout()
plt.xticks(rotation=90)
plt.show()
```



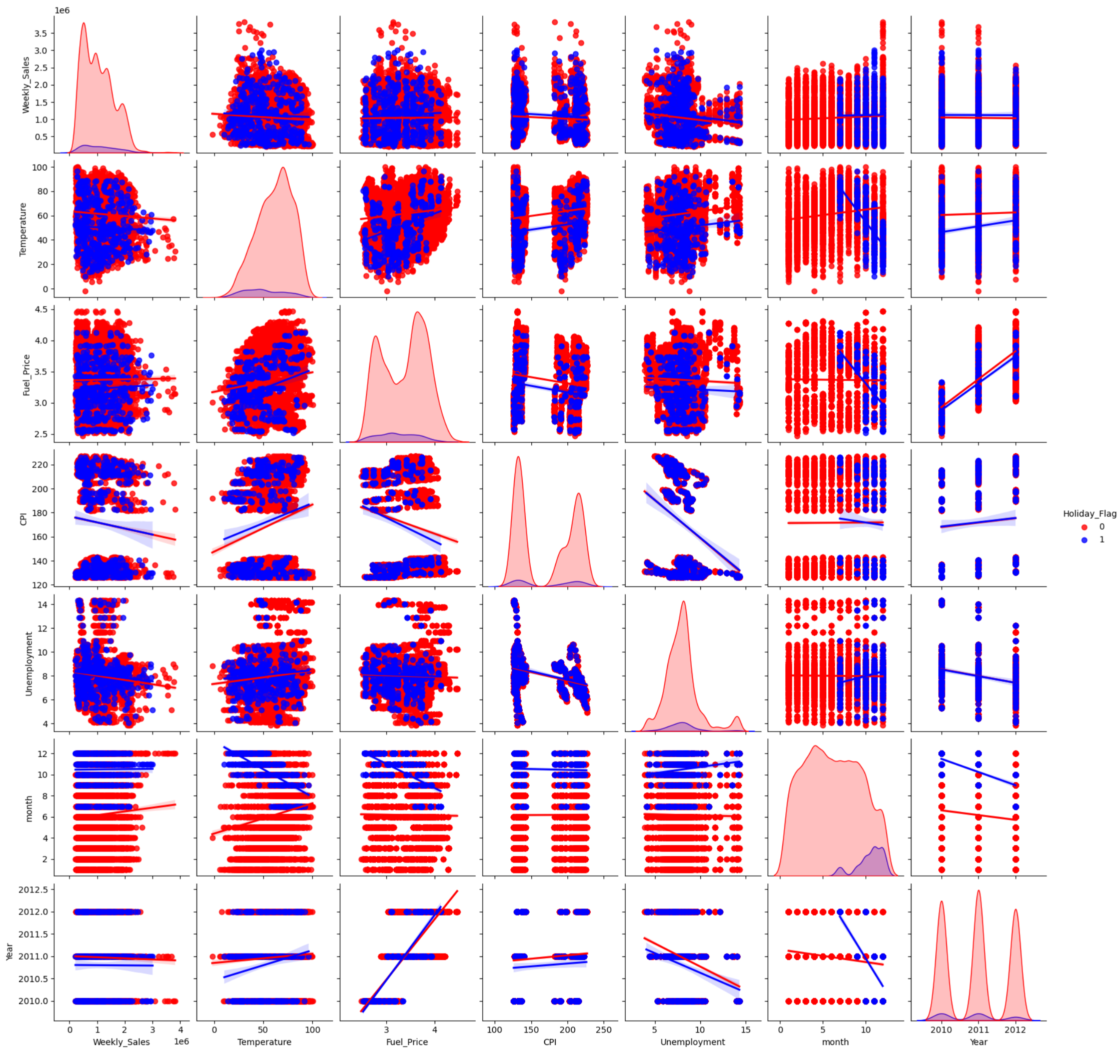
8.Plotted a pair plot to observe the relationships between different fields.

[115]:
```
dfm=df
dfm=dfm.drop(['Store','Week',],axis=1)
dfm.columns
```

[115]:
```
Index(['Weekly_Sales', 'Holiday_Flag', 'Temperature', 'Fuel_Price', 'CPI',
       'Unemployment', 'month', 'Year'],
      dtype='object')
```

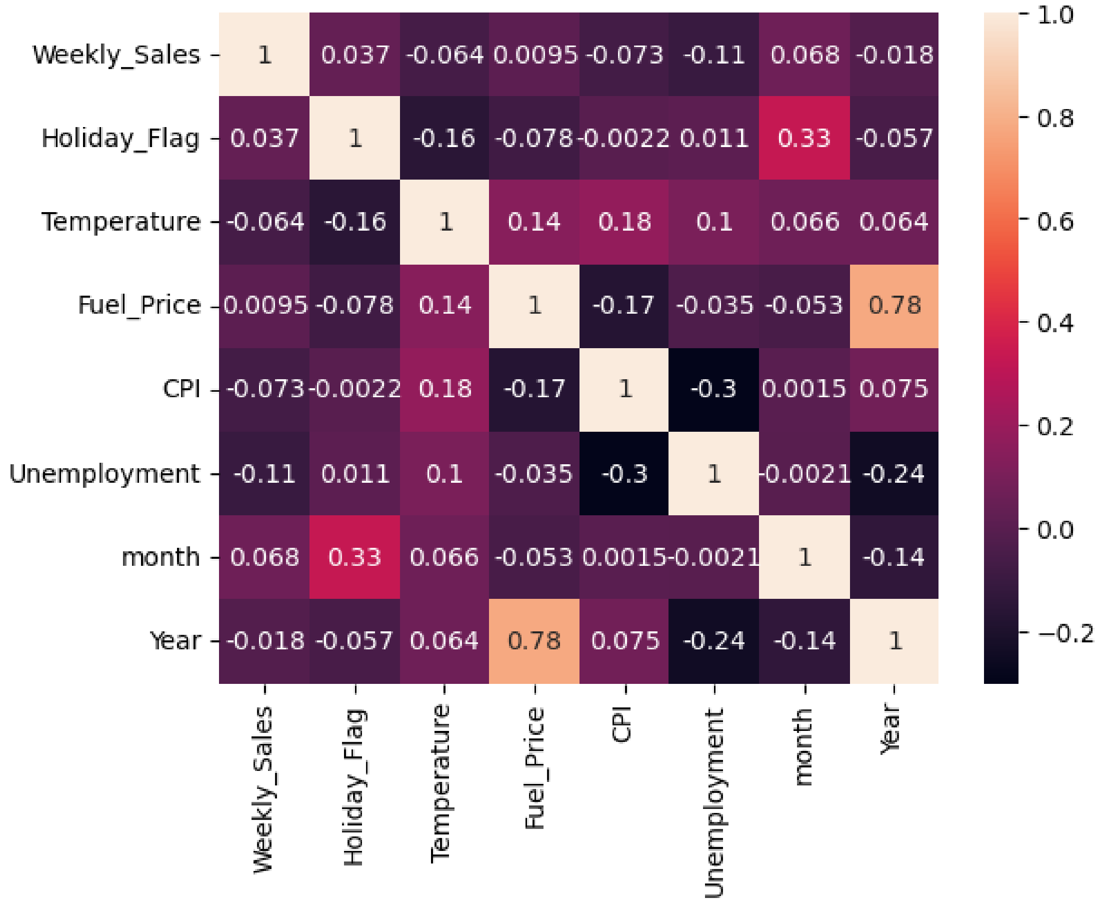
[116]:
```
sns.pairplot(dfm,hue='Holiday_Flag',palette=['red','blue'],kind='reg')
```

[116]:
```
<seaborn.axisgrid.PairGrid at 0x13a4b365780>
```



[117]:
```
sns.heatmap(dfm.corr(),annot=True)
```

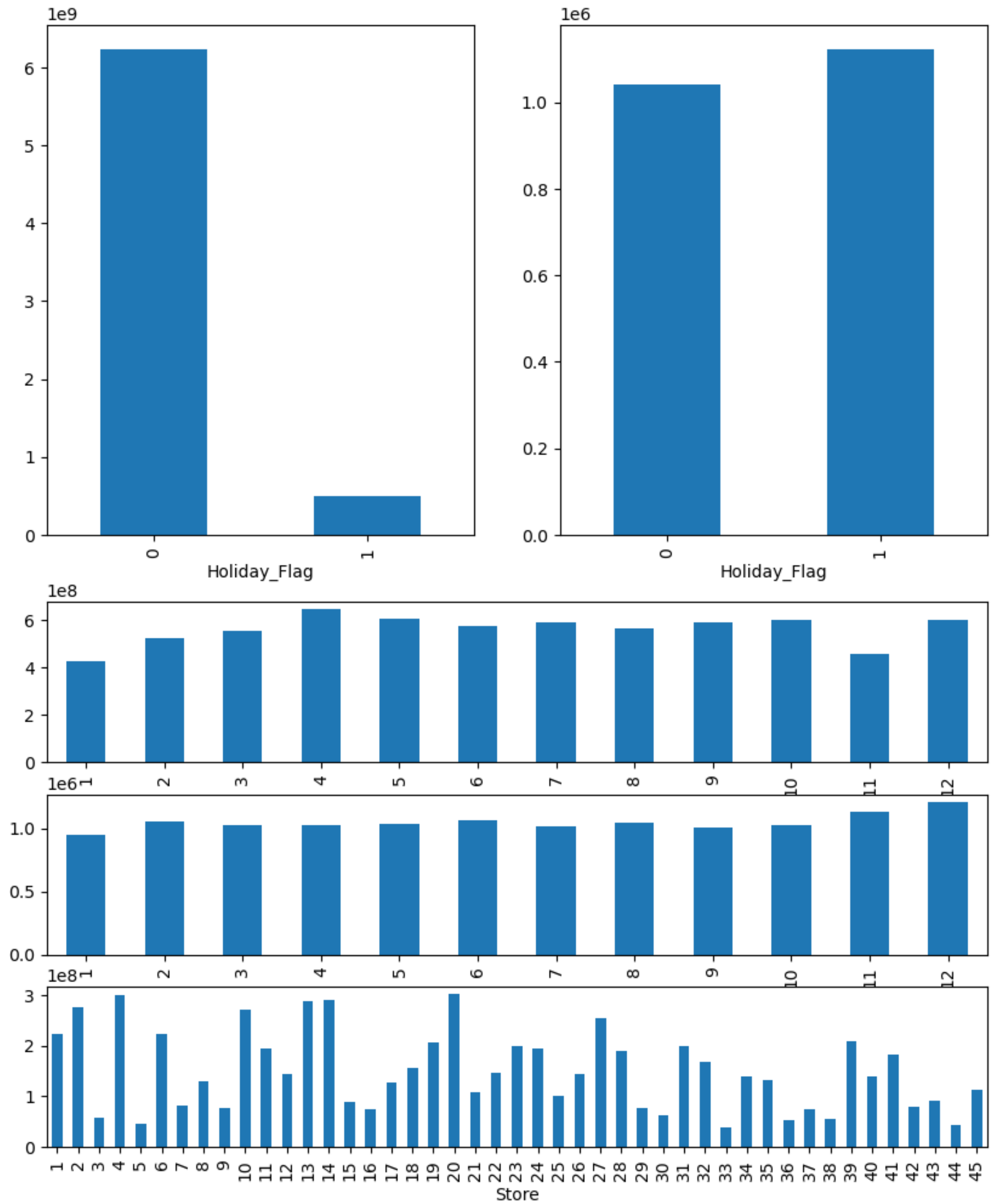
[117]: <Axes: >



9.Created a plot to find the sum and mean of weekly sales with respect to working days, holidays, month and store.

[118]:
```
plt.figure(figsize=[10,12])
plt.subplot(221)
df.groupby('Holiday_Flag')['Weekly_Sales'].sum().plot(kind='bar')

plt.subplot(222)
df.groupby('Holiday_Flag')['Weekly_Sales'].mean().plot(kind='bar')

plt.subplot(614)
df.groupby('month')['Weekly_Sales'].sum().plot(kind='bar')

plt.subplot(615)
df.groupby('month')['Weekly_Sales'].mean().plot(kind='bar')

plt.subplot(616)
df.groupby('Store')['Weekly_Sales'].sum().plot(kind='bar')
```

[118]: <Axes: xlabel='Store'>



10.Performed one-hot encoding and data scaling on categorical variables.

[119]:
```
# one hot ecoding

dfht=pd.DataFrame({})
```

```python
for i in catcol:
  dfht=pd.concat([dfht,pd.
 ↪get_dummies(df[i],drop_first=False,prefix=str(i))],axis=1)
dfht

dfumm=pd.concat((df,dfht),axis=1)
dfumm
```

[119]:

```
      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  month  Year  Week  …  Store_36  Store_37  Store_38  \
0            8.106      5  2010     6  …         0         0         0
1            8.106     12  2010     3  …         0         0         0
2            8.106      2  2010     4  …         0         0         0
3            8.106      2  2010     4  …         0         0         0
4            8.106      5  2010     0  …         0         0         0
…              …     …     …    …  …       …       …       …
6430         8.684      9  2012     4  …         0         0         0
6431         8.667      5  2012     3  …         0         0         0
6432         8.667     12  2012     0  …         0         0         0
6433         8.667     10  2012     4  …         0         0         0
6434         8.667     10  2012     4  …         0         0         0

      Store_39  Store_40  Store_41  Store_42  Store_43  Store_44  Store_45
0            0         0         0         0         0         0         0
1            0         0         0         0         0         0         0
2            0         0         0         0         0         0         0
3            0         0         0         0         0         0         0
4            0         0         0         0         0         0         0
…          …       …       …       …       …       …       …
6430         0         0         0         0         0         0         1
6431         0         0         0         0         0         0         1
6432         0         0         0         0         0         0         1
6433         0         0         0         0         0         0         1
6434         0         0         0         0         0         0         1
```

```
[6435 rows x 79 columns]
```

11.Removed outliers from the dataset, resulting in a removal of 7.94% of the data.

[120]:
```
#remove outlier
for i in numcol:
  q1=dfumm[i].quantile(0.25)
  q3=dfumm[i].quantile(0.75)
  iqr=q3-q1
  h=q3+(1.5*iqr)
  l=q1-(1.5*iqr)
  dfumm=dfumm[(dfumm[i]>=l) & (dfumm[i]<=h)]
dfumm
```

[120]:
```
      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  month  Year  Week  …  Store_36  Store_37  Store_38  \
0            8.106      5  2010     6  …         0         0         0
1            8.106     12  2010     3  …         0         0         0
2            8.106      2  2010     4  …         0         0         0
3            8.106      2  2010     4  …         0         0         0
4            8.106      5  2010     0  …         0         0         0
…              …    …    …   …  …     …       …
6430         8.684      9  2012     4  …         0         0         0
6431         8.667      5  2012     3  …         0         0         0
6432         8.667     12  2012     0  …         0         0         0
6433         8.667     10  2012     4  …         0         0         0
6434         8.667     10  2012     4  …         0         0         0

      Store_39  Store_40  Store_41  Store_42  Store_43  Store_44  Store_45
0            0         0         0         0         0         0         0
1            0         0         0         0         0         0         0
2            0         0         0         0         0         0         0
3            0         0         0         0         0         0         0
4            0         0         0         0         0         0         0
…          …       …       …       …       …       …       …
6430         0         0         0         0         0         0         1
```

6431	0	0	0	0	0	0	1
6432	0	0	0	0	0	0	1
6433	0	0	0	0	0	0	1
6434	0	0	0	0	0	0	1

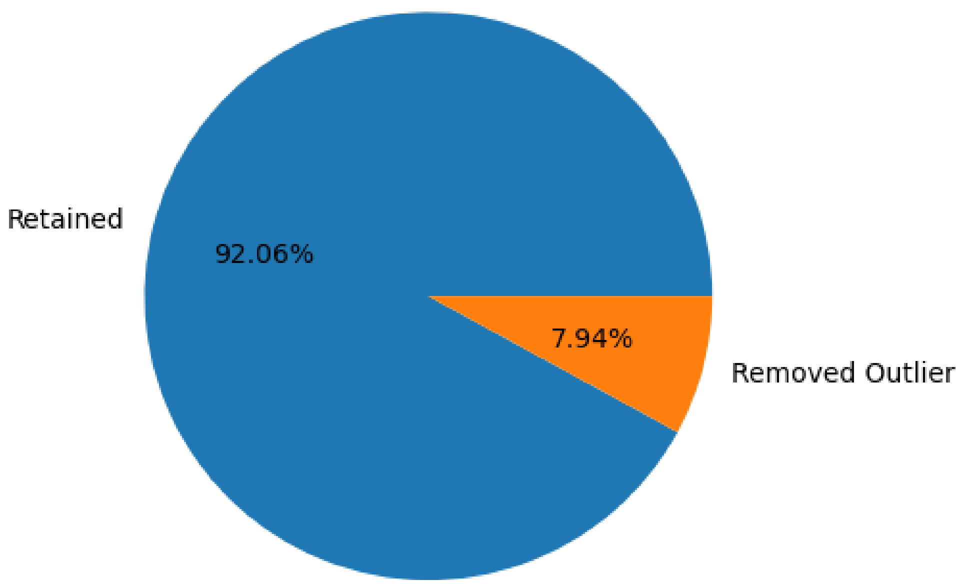
[5924 rows x 79 columns]

[121]:
```
len(dfumm)
```

[121]: 5924

[122]:
```
plt.pie([len(dfumm),len(df)-len(dfumm)],labels=['Retained','Removed␣
↪Outlier'],radius=1,autopct='%1.2f%%')
```

[122]:
```
([<matplotlib.patches.Wedge at 0x13a4e923b80>,
  <matplotlib.patches.Wedge at 0x13a4e923a90>],
 [Text(-1.06594714254286, 0.2715818280090032, 'Retained'),
  Text(1.0659471425428602, -0.2715818280090029, 'Removed Outlier')],
 [Text(-0.5814257141142872, 0.14813554255036537, '92.06%'),
  Text(0.5814257141142873, -0.14813554255036518, '7.94%')])
```



12.Split the dataset into training and testing datasets in an 80:20 ratio and stored the dependent and independent variables in x and y, respectively.

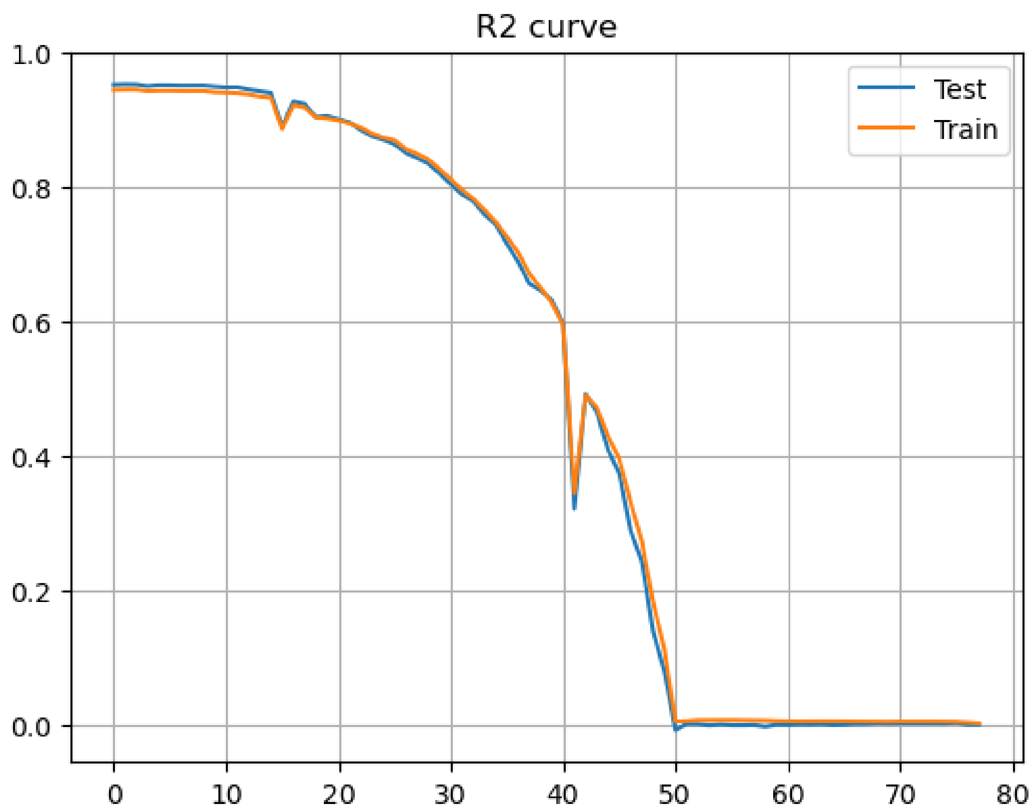
[123]:
```
#split 80/20
x=dfumm.drop(['Weekly_Sales'],axis=1)
y=dfumm['Weekly_Sales']
xn,xs,yn,ys=train_test_split(x,y,train_size=0.8)
```

13.Standardized the test dataset and then transformed training data.

[124]:
```
#Standardization of test and train
std=StandardScaler()
xstdn=std.fit_transform(xn)
xstds=std.transform(xs)
xstdn=pd.DataFrame(xstdn,columns=x.columns)
xstds=pd.DataFrame(xstds,columns=x.columns)
```

14.Conducted recursive feature engineering to determine the number of features to be removed for maximum accuracy.

[125]:
```
#rfe
ln=[]
ls=[]
col=len(x.columns)
for i in range(col):
    lr=LinearRegression()
    rfe=RFE(lr,n_features_to_select=xstdn.shape[1]-i)
    rfe=rfe.fit(xstdn,yn)

    LR=LinearRegression()
    cn=xstdn.loc[:,rfe.support_]
    cs=xstds.loc[:,rfe.support_]
    LR.fit(cn,yn)
    ypn=LR.predict(cn)
    yps=LR.predict(cs)

    ls.append(r2_score(ys,yps))
    ln.append(r2_score(yn,ypn))

plt.plot(ls,label='Test')
plt.plot(ln,label='Train')
plt.title('R2 curve')
plt.legend()
plt.grid()
plt.show()
```



[126]:
```
lr=LinearRegression()
rfe=RFE(lr,n_features_to_select=xstdn.shape[1]-20)
rfe=rfe.fit(xstdn,yn)

LR=LinearRegression()
cn=xstdn.loc[:,rfe.support_]
cs=xstds.loc[:,rfe.support_]
LR.fit(cn,yn)
ypn=LR.predict(cn)
yps=LR.predict(cs)

print(r2_score(ys,yps))
print(r2_score(yn,ypn))
```

```
0.9000366195818327
0.8983948763726114
```

[127]:
```
xrfen=xstdn.loc[:,rfe.support_]
xrfes=xstds.loc[:,rfe.support_]
```

2.1 Focus on Key Drivers

[128]:
```
xrfes
```

[128]:
```
         Store  Holiday_Flag     month      Year      Week  Holiday_Flag_0  \
0     1.621262     -0.269550 -1.664382  0.060107  0.296758        0.269550
1     0.855527     -0.269550 -0.758015  0.060107  0.296758        0.269550
2     0.319512      3.709884  0.148352  1.320484 -2.514504       -3.709884
3     1.008674     -0.269550 -0.153771 -1.200271  0.296758        0.269550
4    -0.139930     -0.269550 -0.455893 -1.200271  0.296758        0.269550
…          …           …         …         …         …              …
1180  0.166365     -0.269550  1.054719 -1.200271 -1.811689        0.269550
1181  0.319512     -0.269550  1.054719  1.320484  0.296758        0.269550
1182 -1.671401     -0.269550 -0.758015  0.060107  0.296758        0.269550
1183  0.166365     -0.269550  0.148352 -1.200271  0.296758        0.269550
1184 -1.058813     -0.269550  0.148352 -1.200271  0.296758        0.269550

      Holiday_Flag_1  Year_2010  Year_2011  Year_2012  …  Store_22  \
0          -0.269550  -0.717315   1.309870  -0.642276  … -0.157703
1          -0.269550  -0.717315   1.309870  -0.642276  … -0.157703
2           3.709884  -0.717315  -0.763435   1.556963  … -0.157703
3          -0.269550   1.394087  -0.763435  -0.642276  … -0.157703
4          -0.269550   1.394087  -0.763435  -0.642276  … -0.157703
…                …          …          …         …   …       …
1180       -0.269550   1.394087  -0.763435  -0.642276  … -0.157703
1181       -0.269550  -0.717315  -0.763435   1.556963  … -0.157703
1182       -0.269550  -0.717315   1.309870  -0.642276  … -0.157703
1183       -0.269550   1.394087  -0.763435  -0.642276  … -0.157703
1184       -0.269550   1.394087  -0.763435  -0.642276  … -0.157703

      Store_25  Store_26  Store_29  Store_30  Store_33  Store_36  Store_39  \
0    -0.152712  -0.16187  -0.16187 -0.158404 -0.161182 -0.152712 -0.155582
1    -0.152712  -0.16187  -0.16187 -0.158404 -0.161182 -0.152712 -0.155582
2    -0.152712  -0.16187  -0.16187 -0.158404 -0.161182 -0.152712 -0.155582
3    -0.152712  -0.16187  -0.16187 -0.158404 -0.161182  6.548254 -0.155582
4    -0.152712  -0.16187  -0.16187 -0.158404 -0.161182 -0.152712 -0.155582
…          …        …        …         …         …         …         …
1180  6.548254  -0.16187  -0.16187 -0.158404 -0.161182 -0.152712 -0.155582
1181 -0.152712  -0.16187  -0.16187 -0.158404 -0.161182 -0.152712 -0.155582
1182 -0.152712  -0.16187  -0.16187 -0.158404 -0.161182 -0.152712 -0.155582
1183  6.548254  -0.16187  -0.16187 -0.158404 -0.161182 -0.152712 -0.155582
1184 -0.152712  -0.16187  -0.16187 -0.158404 -0.161182 -0.152712 -0.155582

      Store_41  Store_45
0    -0.156999 -0.158404
1    -0.156999 -0.158404
2    -0.156999 -0.158404
```

```
3    -0.156999 -0.158404
4    -0.156999 -0.158404
…         …         …
1180 -0.156999 -0.158404
1181 -0.156999 -0.158404
1182 -0.156999 -0.158404
1183 -0.156999 -0.158404
1184 -0.156999 -0.158404

[1185 rows x 58 columns]
```

15.Plotted a scatter plot to compare the predicted values with the test values using multiple linear regression on reduced data after RFE.

[129]:
```
mlr=LinearRegression().fit(xrfen,yn)
print('coefficents:-\n',mlr.coef_)
print('intercept:-\n',mlr.intercept_)
```

```
coefficents:-
 [-7.27594166e+05 -9.95634575e+17 -1.78649156e+18  1.72992043e+18
  2.15032513e+18 -7.86133510e+17  2.09501065e+17  2.04663117e+17
 -8.43207620e+17 -1.78633553e+18  3.61888483e+18  2.92152285e+18
  1.23510945e+18  2.38203661e+18  3.57210254e+18  1.18776448e+18
  1.12585858e+18  3.99918130e+17  5.56726298e+17  7.33671637e+17
  9.45569563e+17  1.06201411e+18  1.19557954e+18  1.39348229e+18
  1.50903015e+18  1.67379492e+18  1.88095925e+18  1.73240898e+18
  1.97059016e+18 -1.98880000e+05 -1.37128000e+05 -3.59496000e+05
 -8.75040000e+04 -3.48888000e+05 -1.55360000e+05 -3.00624000e+05
 -2.36544000e+05 -2.75712000e+05 -7.61280000e+04 -1.47856000e+05
 -6.77760000e+04 -3.40960000e+04 -2.15456000e+05 -2.26288000e+05
 -1.70224000e+05 -1.29248000e+05 -6.00320000e+04 -1.51440000e+05
 -1.01376000e+05 -1.20672000e+05 -7.25280000e+04 -1.17520000e+05
 -1.22160000e+05 -1.28064000e+05 -7.73120000e+04  1.06128000e+05
  9.86880000e+04  5.89120000e+04]
intercept:-
 1175504.553373823
```

[130]:
```
plt.scatter(ys,yps)
plt.plot([min(ys),max(ys)],[min(ys),max(ys)],'r--',label='Test line')
plt.plot([min(yps),max(yps)],[min(yps),max(yps)],'y--',label='Predict Line')
plt.xlabel('Y-Test')
plt.ylabel('Y-Predict')
plt.title('Test Vs Predict')
plt.legend()

print('train r2_score',r2_score(yn,ypn))
print('test r2_score',r2_score(ys,yps))
```

```
train r2_score 0.8983948763726114
```

```
test r2_score 0.9000366195818327
```