Appendix

Varsha Komalla Neha Cemerla Sujith Kondreddy Praneeth Reddy Kothwal

Project Title: Walmart Sales

Dataset: The dataset "Walmart Store Sales" is retrieved from Kaggle.com. This dataset consists of 6436 rows which represent different invoices and 8 columns.

Head():

This function is displaying the first n rows present in the input data frame and is returning the first or last parts of a vector, matrix, table, data frame or function.

```
> head(Walmart_1_)
# A tibble: 6 × 8
  Store Date       Weekly_Sales Holiday_Flag Temperature Fuel_Pr…¹   CPI Unemp…²
  <dbl> <chr>             <dbl>        <dbl>       <dbl>     <dbl> <dbl>   <dbl>
1     1 40300          1643691.            0        42.3      2.57  211.    8.11
2     1 40514          1641957.            1        38.5      2.55  211.    8.11
3     1 19-02-2010     1611968.            0        39.9      2.51  211.    8.11
4     1 26-02-2010     1409728.            0        46.6      2.56  211.    8.11
5     1 40301          1554807.            0        46.5      2.62  211.    8.11
6     1 40515          1439542.            0        57.8      2.67  211.    8.11
# … with abbreviated variable names ¹Fuel_Price, ²Unemployment
> |
```

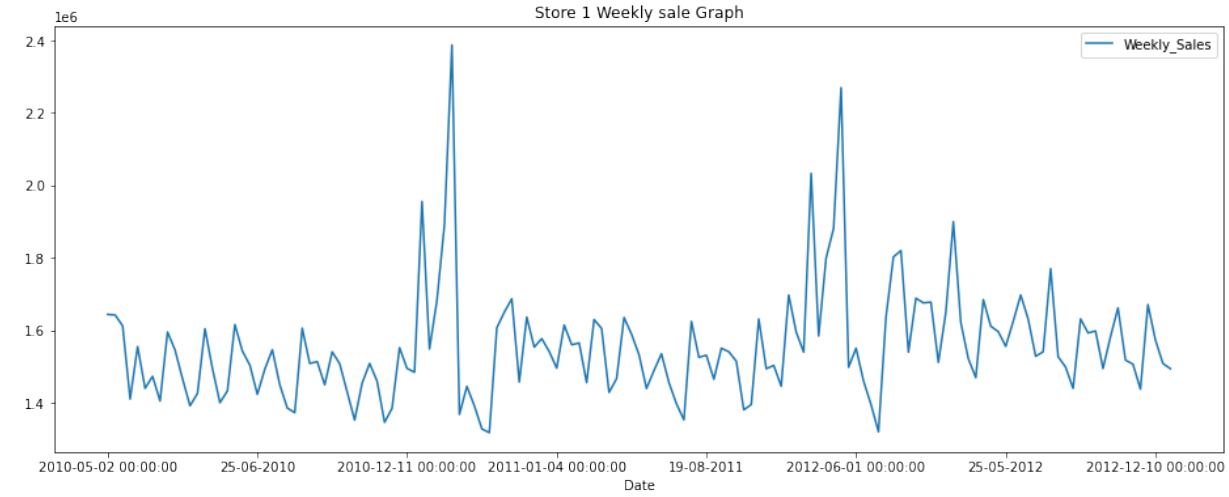
Summary():

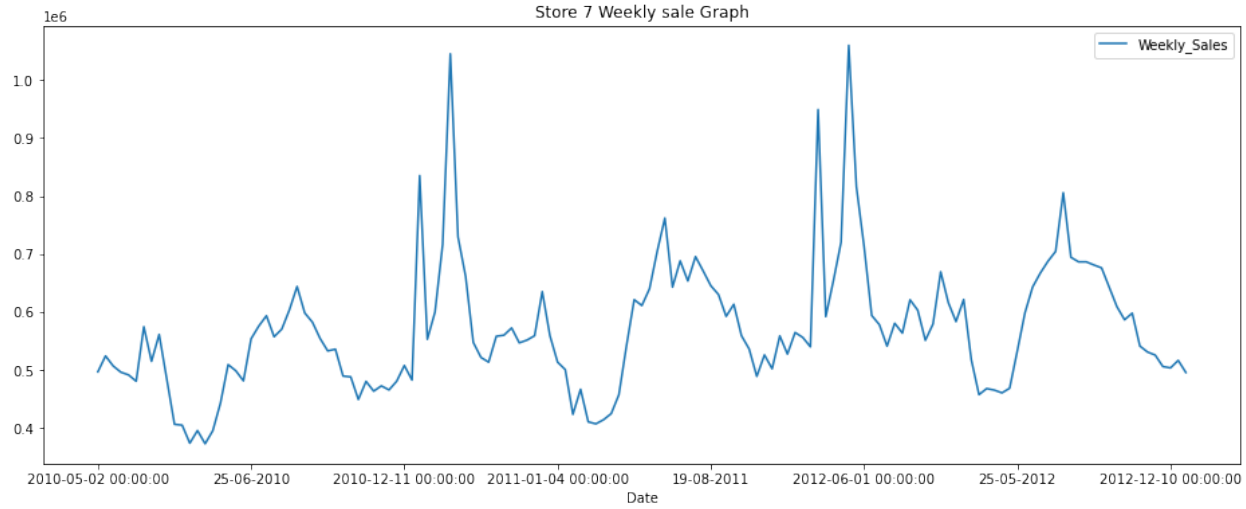
This function is letting us know the short analysis of the data such as data type of the column, minimum, maximum, quartiles, mean, median, etc. of the date.

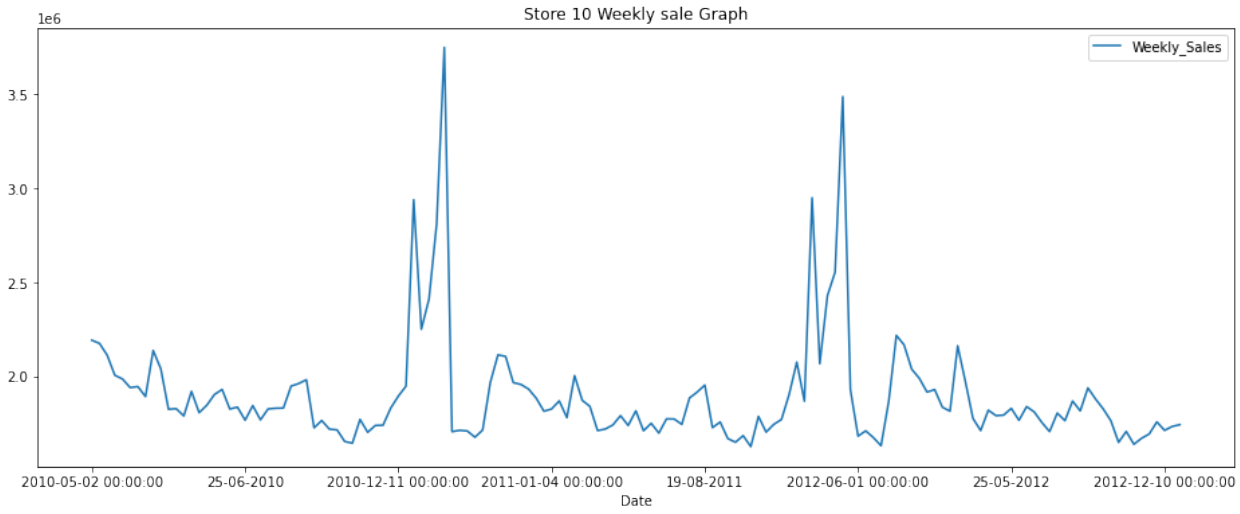
```
> summary(Walmart_1_)
     Store          Date            Weekly_Sales      Holiday_Flag
 Min.   : 1   Length:6435        Min.   : 209986   Min.   :0.00000
 1st Qu.:12   Class :character   1st Qu.: 553350   1st Qu.:0.00000
 Median :23   Mode  :character   Median : 960746   Median :0.00000
 Mean   :23                      Mean   :1046965   Mean   :0.06993
 3rd Qu.:34                      3rd Qu.:1420159   3rd Qu.:0.00000
 Max.   :45                      Max.   :3818686   Max.   :1.00000
  Temperature      Fuel_Price         CPI         Unemployment
 Min.   : -2.06   Min.   :2.472   Min.   :126.1   Min.   : 3.879
 1st Qu.: 47.46   1st Qu.:2.933   1st Qu.:131.7   1st Qu.: 6.891
 Median : 62.67   Median :3.445   Median :182.6   Median : 7.874
 Mean   : 60.66   Mean   :3.359   Mean   :171.6   Mean   : 7.999
 3rd Qu.: 74.94   3rd Qu.:3.735   3rd Qu.:212.7   3rd Qu.: 8.622
 Max.   :100.14   Max.   :4.468   Max.   :227.2   Max.   :14.313
> |
```

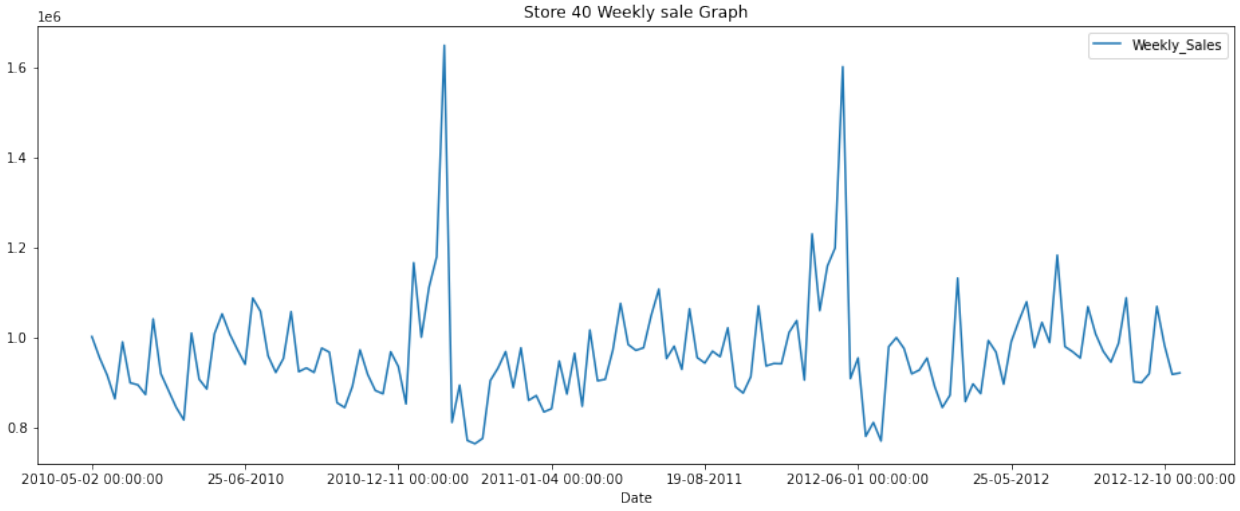
Weekly analysis of Sales:

```
for x in range(45):
    data[data["Store"]==x+1].plot(y="Weekly_Sales",x="Date",title="Store " + str(x+1) + " Weekly sale Graph",figsize=(16,6))
```







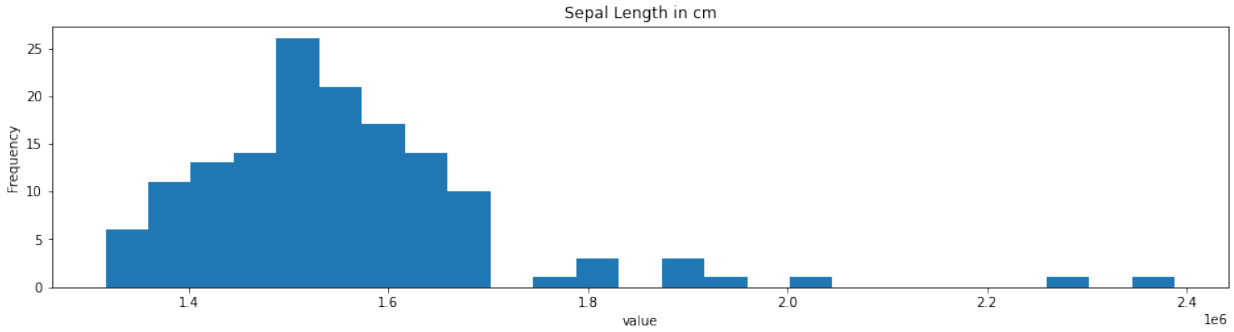


The above shows weekly sales of the store 1, 7, 10, 40 over multiple years. On the x-axis it shows the date and y-axis show the sales. From 2010-12-11 to 2011-01-04 maximum sales can be seen. At the start of 2010 the sales then they started to increase till mid and then at the end they stated to decrease.

Frequency of Sales:

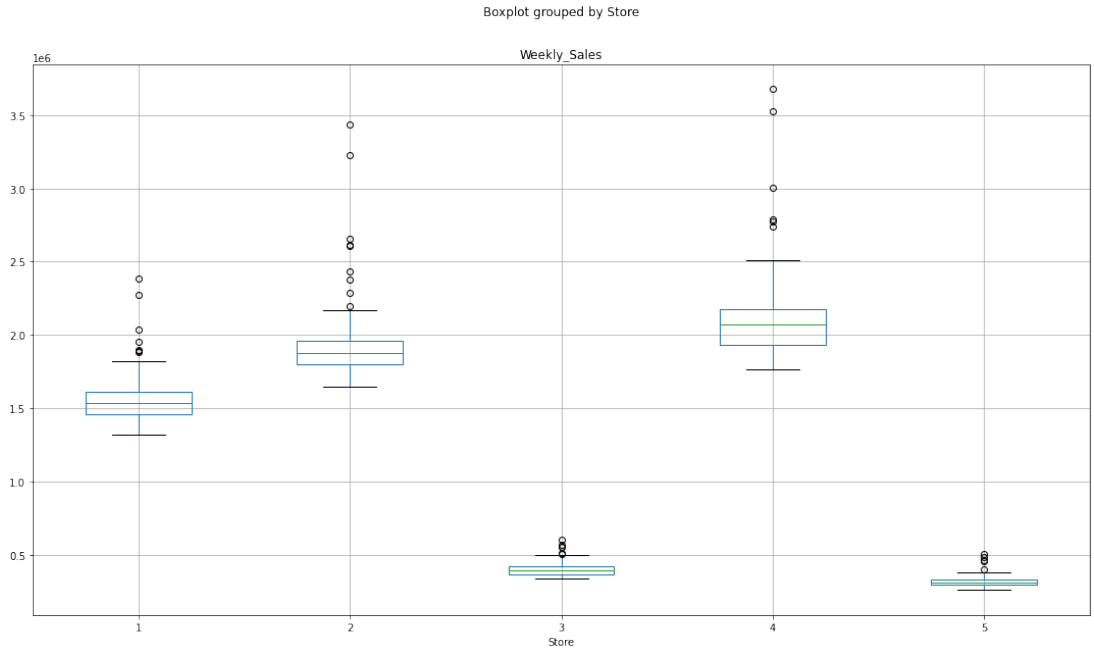
```
In [36]:
fig,ax=plt.subplots((2),figsize=(16,8))
ax[0].set_title("Weekly Sale of Store")
ax[0].set_ylabel("Frequency")
ax[0].set_xlabel("value")
ax[0].hist(data[data["Store"]==1]["Weekly_Sales"].values,bins=25)
```

```
Out[36]: (array([ 6., 11., 13., 14., 26., 21., 17., 14., 10.,  0.,  1.,  3.,  0.,
                  3.,  1.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  1.]),
          array([1316899.31  , 1359741.3456, 1402583.3812, 1445425.4168,
                 1488267.4524, 1531109.488 , 1573951.5236, 1616793.5592,
                 1659635.5948, 1702477.6304, 1745319.666 , 1788161.7016,
                 1831003.7372, 1873845.7728, 1916687.8084, 1959529.844 ,
                 2002371.8796, 2045213.9152, 2088055.9508, 2130897.9864,
                 2173740.022 , 2216582.0576, 2259424.0932, 2302266.1288,
                 2345108.1644, 2387950.2   ]),
          <BarContainer object of 25 artists>)
```

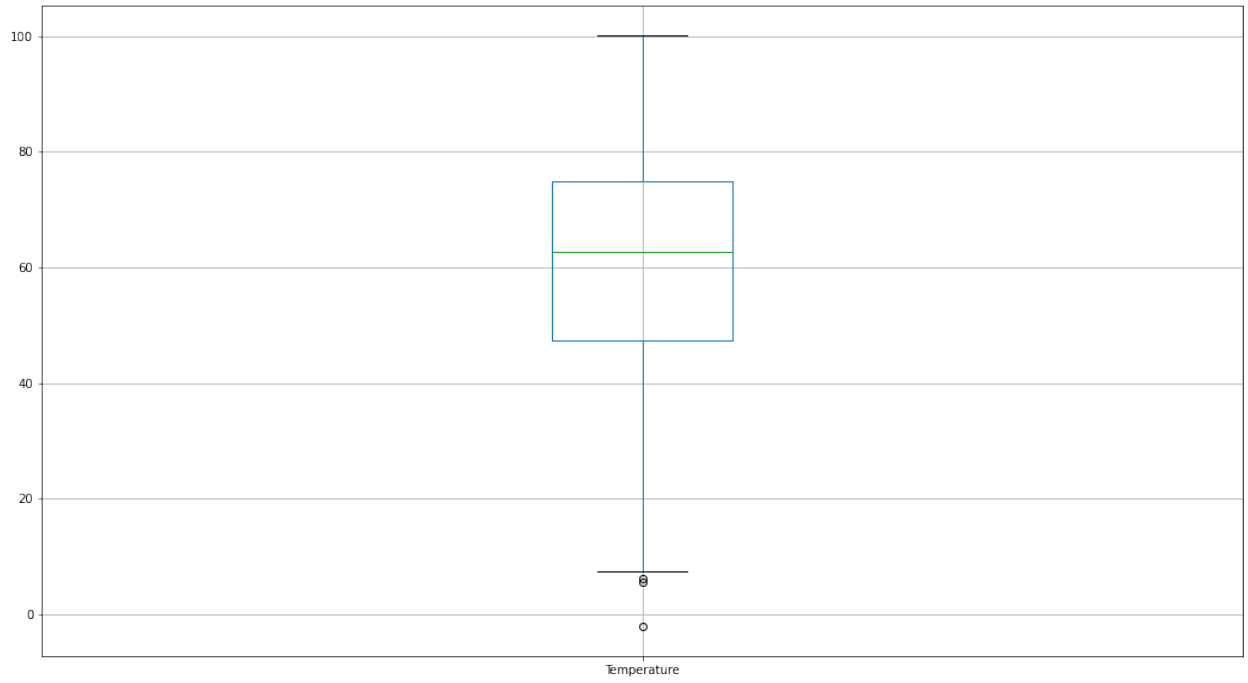


The figure above has the values at the x-axis and y-axis shows the frequency of the sales. The figure shows the weekly sales of the store with frequency as sales.

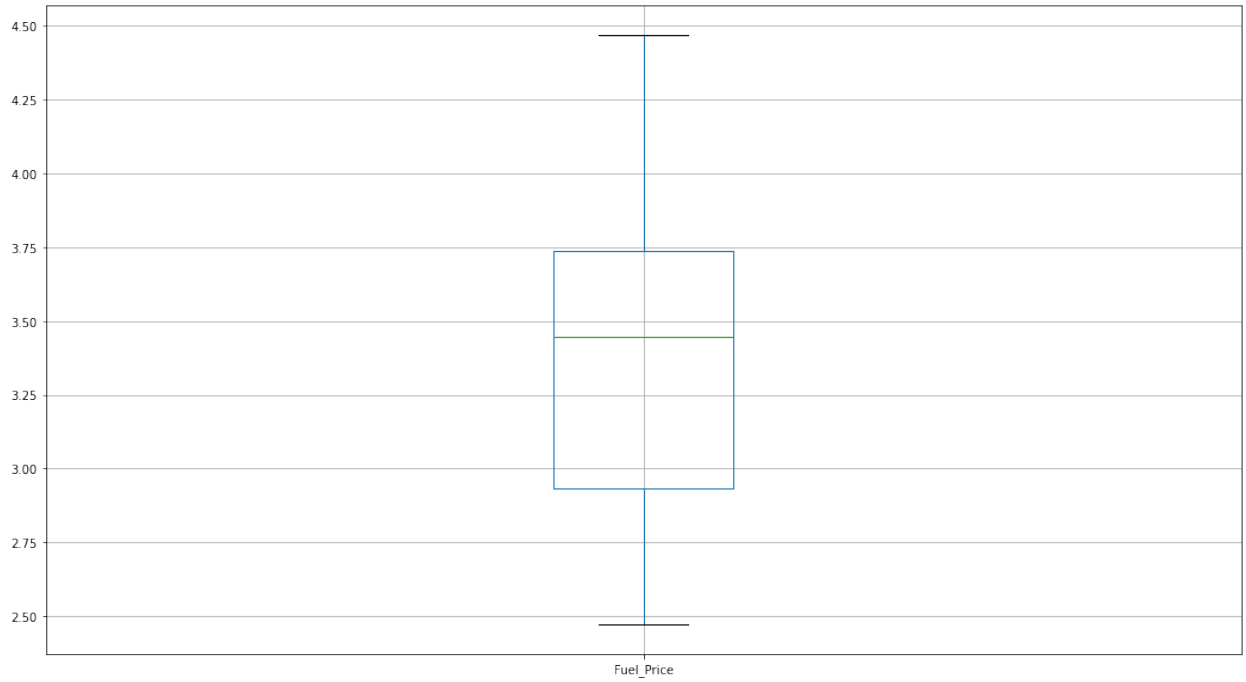
Weekly sales in Boxplot:



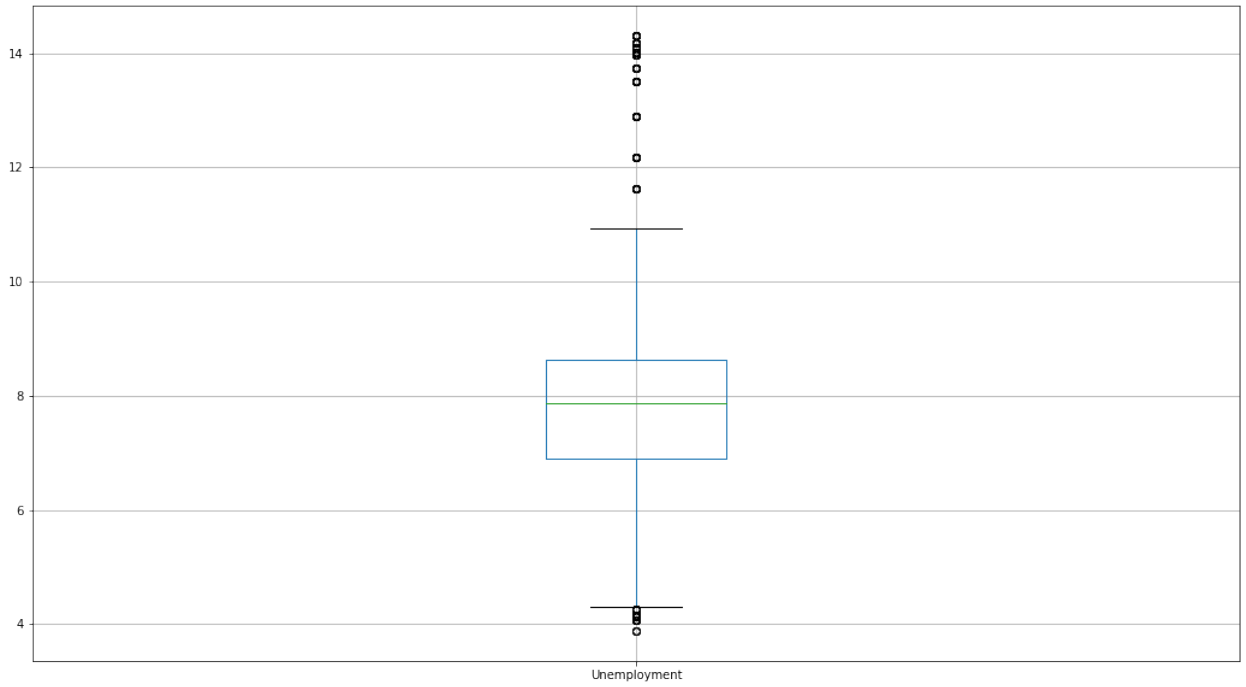
The above boxplot shows the weekly sales with mean and average values. The sales of different stores can be seen with the mean values. Store 4 has the highest average, but it has the greatest number of dispersed data which shows outliers. Stores 3 and 4 have the least average of sales and a smaller number of outliers.



The figure above shows the plot of Temperature with the mean value of nearly 65 and the upper limit of temperature is 100 with the lowest values of temperature starting with 0.



The figure above shows the plot of Fuel Price with the mean value of nearly 3.40 and the upper limit of temperature is 4.48 with the lowest values of temperature starting with 2.4 and there are no outliers.

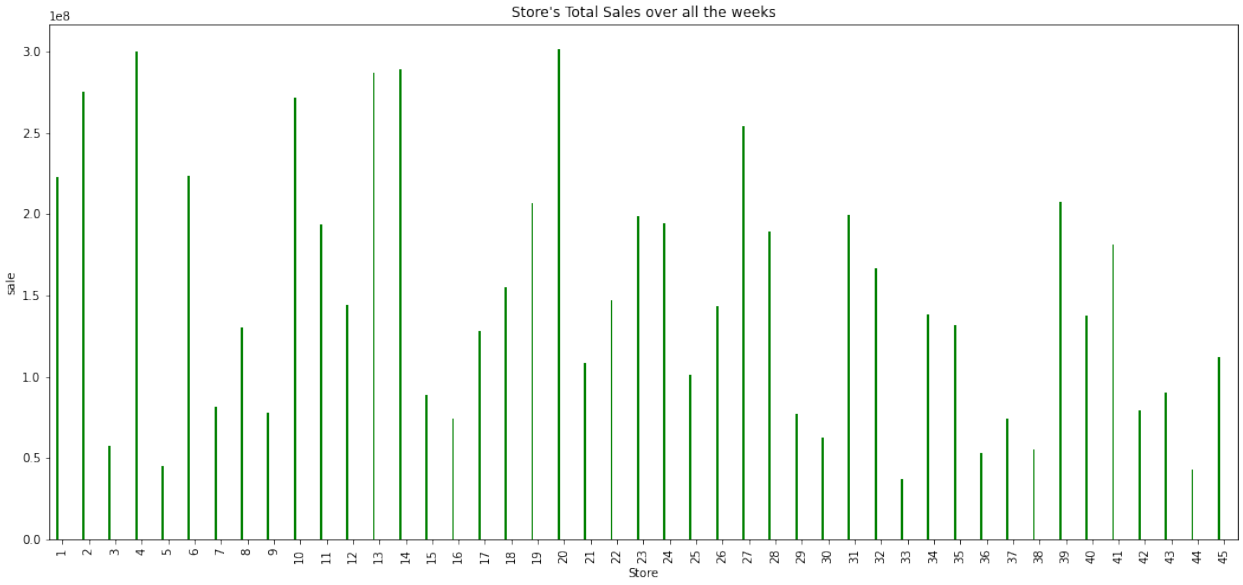


The figure above shows the plot of Unemployment with the mean value of nearly 7 and the upper limit of unemployment is 10.5 with the lowest values of temperature starting with 4.5 and there are are a great number of outliers above upper range means the number of unemployment increases at certain stores greater than others.

Total Sales analysis over all the weeks:

```
In [101]: store_sale=data.groupby(by="Store").sum()#.sort_values(by="Weekly_Sales",ascending=False)

In [102]: store_sale.plot.bar(figsize=(18,8),legend=False,title="Store's Total Sales over all the weeks",color='g',ylabel="sale")
```

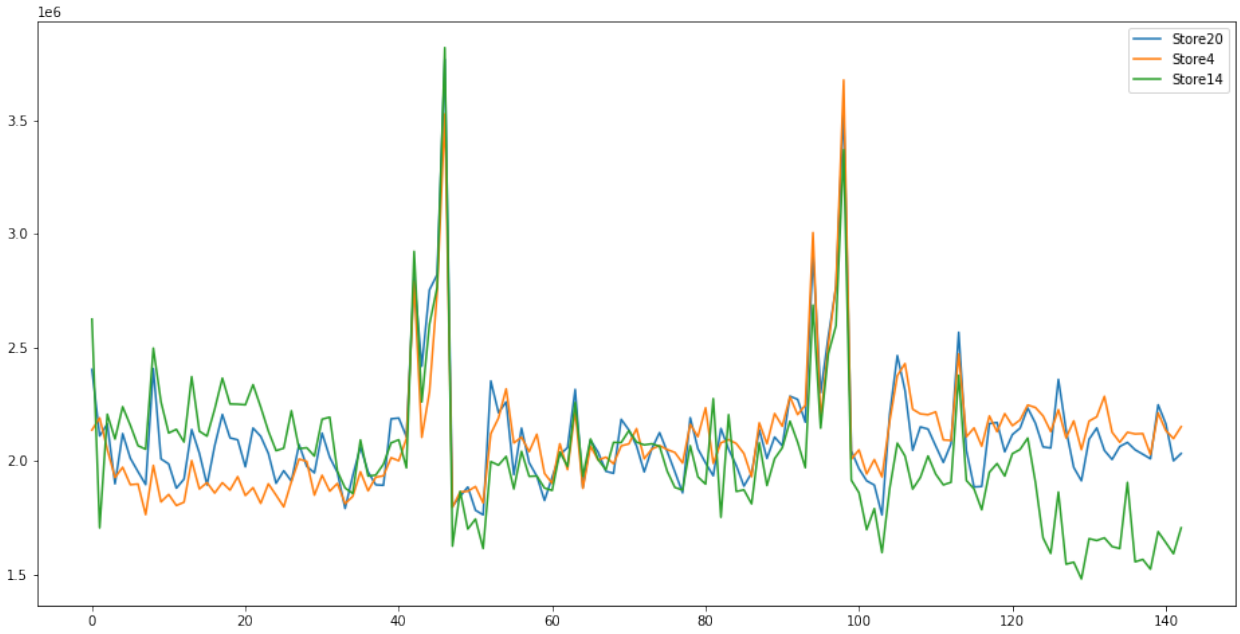


The graph above shows the stores sales over all the weeks by plotting the sales of every week with the store. X-axis has the store, and the y-axis has the sales. Store 4 and 20 at weeks shows the highest number of sales.

```
store_sale=data.groupby(by="Store").sum().sort_values(by="Weekly_Sales",ascending=False)
fig,ax=plt.subplots(figsize=(16,8))
#ax.title="Top 3 Store's Sales trend over the weeks"
#ax.xlabel="Revenue/Sale"
#ax.ylabel="Week Number"

for x in range(3):
    idx=store_sale.index[x]
    ax.plot(range(data[data["Store"]==idx].shape[0]),data[data["Store"]==idx]["Weekly_Sales"],label="Store"+str(idx))

plt.legend()
plt.show()
```

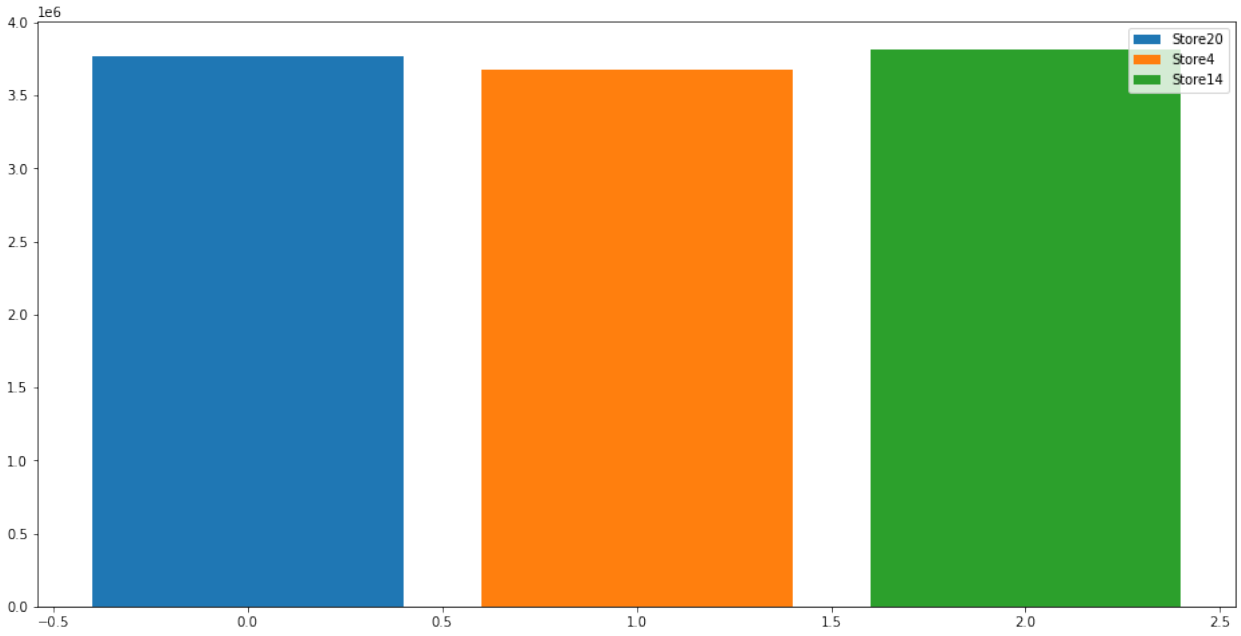


The graph above shows the data of store 20,4 and 14 with the sales over the weeks. The store in green which has store 14 has the highest number of sales even it started with the greater sales than others. In comparison store 4 has less number of sales then store 14 and 20.

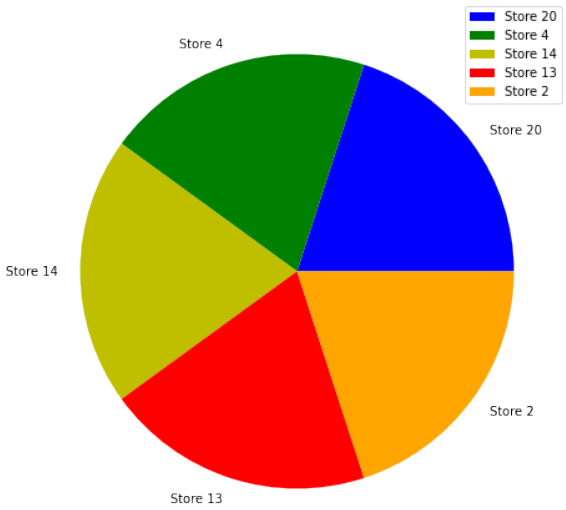
```
color=['b','g','y']
store_sale=data.groupby(by="Store").sum().sort_values(by="Weekly_Sales",ascending=False)
fig,ax=plt.subplots(figsize=(16,8))

for x in range(3):
    idx=store_sale.index[x]
    ax.bar(x,data[data["Store"]==idx]["Weekly_Sales"],label="Store"+str(idx))

plt.legend()
plt.show()
```

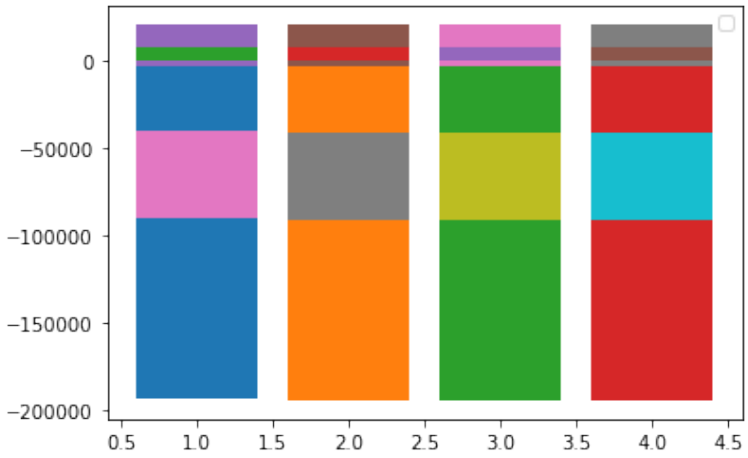


The bar plot above depicts the sales and it is visible that the store 20 and store 14 have nearly equal number of sales but store 4 has less number of sales than these stores.



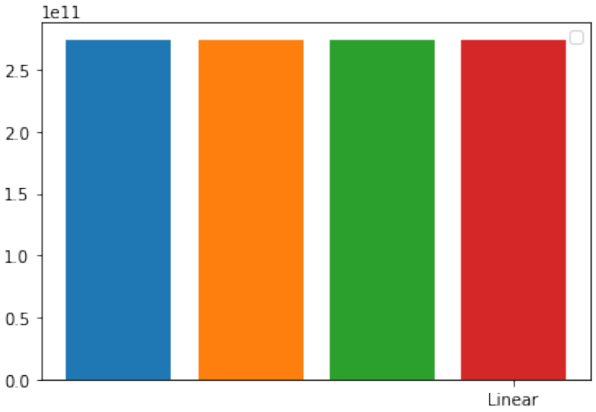
The pie chart above shows the weekly sales of multiple stores with the proportions. And from it we can judge that store 20, 4, 14, 13 and 2 have approximately similar sales.

Mean Squared Error Coefficient of all models:



The coefficients of multiple statistical methods were calculated and then plotted in the above graph which just shows the range of values of coefficients where they fall after training or learning.

Comparison of MSE of all above 4 models using K fold cross validation:



The plot above shows the mean error of the 4 regression models in 5-fold which were EN, Rigid, Lasso and Rigid regression and all of them had the same affect as in they have same value of mean error.

Mean Squared Error values:

In [79]:
```
print("MSE of Elastic Net Model= ", np.mean(EN_Err))
print("MSE of Ridge Model= ", np.mean(Ridge_Err))
print("MSE of Lasso Model= ", np.mean(Lasso_Err))
print("MSE of Linear Regression Model= ", np.mean(Linear_Err))
```

```
MSE of Elastic Net Model=  274299314954.4134
MSE of Ridge Model=  274301145325.88608
MSE of Lasso Model=  274301146656.86865
MSE of Linear Regression Model=  274301146628.2364
```

In [80]:
```
EN_model.coef_
```

Out[80]:
```
array([-193634.38020704,   20361.48853183,  -21568.74417073,
         8190.25190034,  -90575.94052946,  -40752.67855507,
        -2941.7586287 ])
```

In [81]:
```
R_model.coef_
```

Out[81]:
```
array([-194710.70053969,   20464.40453851,  -21521.11023867,
         8143.14134403,  -91317.46317499,  -40931.94808043,
        -2923.43360308])
```

In [82]:
```
L_model.coef_
```

Out[82]:
```
array([-194711.11163758,   20464.43002314,  -21521.08398206,
         8143.07715519,  -91317.74908981,  -40932.00009323,
        -2923.37744014])
```

In [*]:
```
all_cofs=np.array([EN_model.coef_,R_model.coef_,L_model.coef_,LR_model.coef_])
```

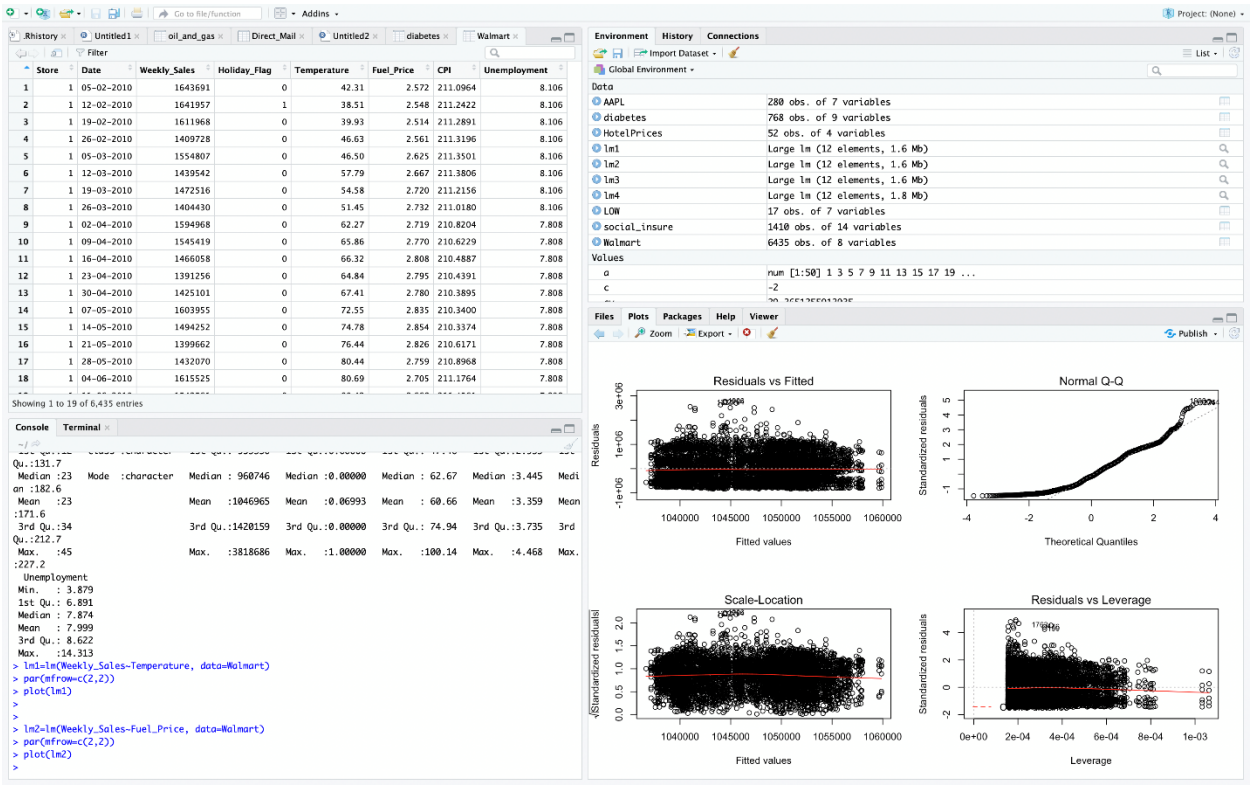
In [87]:
```
all_cofs
```

Out[87]:
```
array([[-193634.38020704,   20361.48853183,  -21568.74417073,
         8190.25190034,  -90575.94052946,  -40752.67855507,
        -2941.7586287 ],
       [-194710.70053969,   20464.40453851,  -21521.11023867,
         8143.14134403,  -91317.46317499,  -40931.94808043,
        -2923.43360308],
       [-194711.11163758,   20464.43002314,  -21521.08398206,
         8143.07715519,  -91317.74908981,  -40932.00009323,
        -2923.37744014],
       [-194711.1212715 ,   20464.44468249,  -21521.09107965,
         8143.12184465,  -91317.75380964,  -40932.0181478 ,
        -2923.42552858]])
```

Linear Regression:

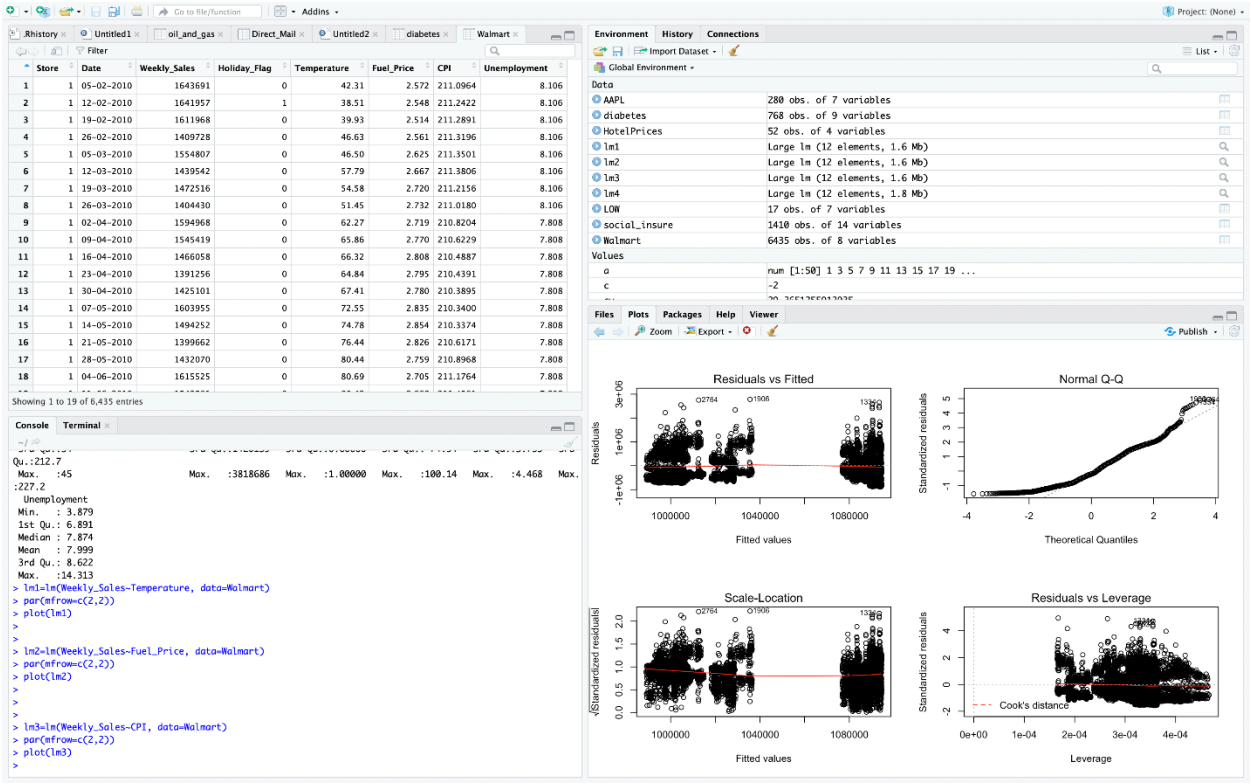
Model 1:

Weekly Sales vs Temperature



Model 2:

Weekly Sales vs Fuel Price




```
%pylab inline

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data=pd.read_excel("Walmart.xlsx")

data

for x in range(45):

    data[data["Store"]==x+1].plot(y="Weekly_Sales",x="Date",title="Store " + str(x+1) + "
        Weekly sale Graph",figsize=(16,6))
```

```
data[data["Store"]==1]

fig,ax=plt.subplots((2),figsize=(16,8))

ax[0].set_title("Weekly Sale of Store")

ax[0].set_ylabel("Frequency")

ax[0].set_xlabel("value")

ax[0].hist(data[data["Store"]==1]["Weekly_Sales"].values,bins=25)

for x in range(1,46,5):

    data[np.logical_and(np.array(data["Store"]>=x),
         np.array(data["Store"]<x+5))].boxplot(column="Weekly_Sales",by="Store",
         figsize=(18,10))

#np.logical_and(np.array(data["Store"]>=1), np.array(data["Store"]<=5))

data.boxplot(column="Temperature", figsize=(18,10))

data.boxplot(column="Fuel_Price", figsize=(18,10))

data.boxplot(column="CPI", figsize=(18,10))

data.boxplot(column="Unemployment", figsize=(18,10))

store_sale=data.groupby(by="Store").sum()#.sort_values(by="Weekly_Sales",ascending=False)

store_sale.plot.bar(figsize=(18,8),legend=False,title="Store's Total Sales over all the
         weeks",color='g',ylabel="sale")

store_sale=data.groupby(by="Store").sum().sort_values(by="Weekly_Sales",ascending=False)

store_sale=data.groupby(by="Store").sum().sort_values(by="Weekly_Sales",ascending=False)

fig,ax=plt.subplots(figsize=(16,8))

#ax.title="Top 3 Store's Sales trend over the weeks"

#ax.xlabel="Revenue/Sale"
```

```
#ax.ylabel="Week Number"
for x in range(3):
    idx=store_sale.index[x]
ax.plot(range(data[data["Store"]==idx].shape[0]),data[data["Store"]==idx]["Weekly_Sales"],lab
        el="Store"+str(idx))
plt.legend()
plt.show()
#data[np.logical_and(np.array(data["Store"]>=1), np.array(data["Store"]<5))]
#data[data["Store"]==1]
color=['b','g','y']
store_sale=data.groupby(by="Store").sum().sort_values(by="Weekly_Sales",ascending=False)
fig,ax=plt.subplots(figsize=(16,8))
for x in range(3):
    idx=store_sale.index[x]
    ax.bar(x,data[data["Store"]==idx]["Weekly_Sales"],label="Store"+str(idx))
plt.legend()
plt.show()
color=['b','g','y','r','orange']
store_sale=data.groupby(by="Store").sum().sort_values(by="Weekly_Sales",ascending=False)
fig,ax=plt.subplots(figsize=(16,8))
sales=[]
storeids=[]
for x in range(5):
```

```
    storeids.append("Store "+str(store_sale.index[x]))
    sales.append(data[data["Store"]==idx]["Weekly_Sales"].sum())
ax.pie(sales,labels=storeids,colors=color)
fig.show()
plt.legend()
plt.show()
data.describe()
data.head()
import datetime as dt
data['Date'] = pd.to_datetime(data['Date'])
data['OrdDate']=data['Date'].apply(lambda x: x.toordinal())
data.head()
from sklearn.preprocessing import PolynomialFeatures
X=np.array(data.drop(columns=['Date','Weekly_Sales']))
# polynomial_features= PolynomialFeatures(degree=2)
# X = polynomial_features.fit_transform(X)
Y=np.array(data['Weekly_Sales'])
from sklearn.preprocessing import StandardScaler
norm_model=StandardScaler()
X_norm=norm_model.fit_transform(X)
from sklearn.linear_model import ElasticNet,Ridge,Lasso,LinearRegression
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error
```

```
kf = KFold(n_splits=5,shuffle=True)
kf.get_n_splits(X)
EN_Err=[]
Ridge_Err=[]
Lasso_Err=[]
Linear_Err=[}
for i, (train_index, test_index) in enumerate(kf.split(X)):
    train_X=X_norm[train_index]
    train_Y=Y[train_index]
    test_X=X_norm[test_index]
    test_Y=Y[test_index]
    EN_model=ElasticNet(alpha=0.01)
    EN_model.fit(train_X,train_Y)
    pred_y_EN=EN_model.predict(test_X)
    EN_Err.append(mean_squared_error(pred_y_EN,test_Y))
    R_model=Ridge(alpha=0.01)
    R_model.fit(train_X,train_Y)
    pred_y_R=R_model.predict(test_X)
    Ridge_Err.append(mean_squared_error(pred_y_R,test_Y))

    L_model=Lasso(alpha=0.01)
    L_model.fit(train_X,train_Y)
    pred_y_L=L_model.predict(test_X)
```

```
    Lasso_Err.append(mean_squared_error(pred_y_L,test_Y))
    LR_model=LinearRegression()
    LR_model.fit(train_X,train_Y)
    pred_y_LR=LR_model.predict(test_X)
    Linear_Err.append(mean_squared_error(pred_y_LR,test_Y))
print("MSE of Elastic Net Model= ", np.mean(EN_Err))
print("MSE of Ridge Model= ", np.mean(Ridge_Err))
print("MSE of Lasso Model= ", np.mean(Lasso_Err))
print("MSE of Linear Regression Model= ", np.mean(Linear_Err))
EN_model.coef_
R_model.coef_
L_model.coef_
LR_model.coef_
all_cofs=np.array([EN_model.coef_,R_model.coef_,L_model.coef_,LR_model.coef_])
all_cofs
ax=plt.subplot()
for x in range(7):
    ax.bar(1,all_cofs[0][x])
    ax.bar(2,all_cofs[1][x])
    ax.bar(3,all_cofs[2][x])
    ax.bar(4,all_cofs[3][x])
plt.legend()
plt.show()
```

```
ax=plt.subplot()
plt.title="Mean Error of all models in 5 folds"
plt.bar(1,np.mean(EN_Err),tick_label="EN")
plt.bar(2,np.mean(Ridge_Err),tick_label="Ridge")
plt.bar(3,np.mean(Lasso_Err),tick_label="Lasso")
plt.bar(4,np.mean(Linear_Err),tick_label="Linear")
plt.legend()
plt.show()
```