In [ ]:
```
#Project 4 :Walmart sales analysis
```

In [701…
```
#importing neccessary files
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import io
```

In [702…
```
#setting up the directory
%cd "C:\Users\Public\Python\Project 4\"
```

```
C:\Users\Public\Python\Project 4
```

In [703…
```
#importing the data into a dataframe sales
sales = pd.read_csv("Walmart_Store_sales.csv")
```

In [704…
```
#checking the datatype of the columns
sales.dtypes
```

Out[704…
```
Store             int64
Date             object
Weekly_Sales    float64
Holiday_Flag      int64
Temperature     float64
Fuel_Price      float64
CPI             float64
Unemployment    float64
dtype: object
```

In [705…
```
#got the description of the data
sales.describe()
```

Out[705…

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
count	6435.000000	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000
mean	23.000000	1.046965e+06	0.069930	60.663782	3.358607	171.578394	7.999151
std	12.988182	5.643666e+05	0.255049	18.444933	0.459020	39.356712	1.875885
min	1.000000	2.099862e+05	0.000000	-2.060000	2.472000	126.064000	3.879000
25%	12.000000	5.533501e+05	0.000000	47.460000	2.933000	131.735000	6.891000
50%	23.000000	9.607460e+05	0.000000	62.670000	3.445000	182.616521	7.874000
75%	34.000000	1.420159e+06	0.000000	74.940000	3.735000	212.743293	8.622000
max	45.000000	3.818686e+06	1.000000	100.140000	4.468000	227.232807	14.313000

In [706…
```
# 1.Which store has maximum sales
```

```
sales.Weekly_Sales.groupby(sales.Store).max().sort_values(ascending=False).iloc[:1]
```

Out[706…
```
Store
14    3818686.45
Name: Weekly_Sales, dtype: float64
```

In [707…
```
# 2. a) Which store has maximum standard deviation
maxstd =sales.Weekly_Sales.groupby(sales.Store).std().sort_values(ascending=False).iloc
print(maxstd)
```

```
Store
14    317569.949476
Name: Weekly_Sales, dtype: float64
```

In [708…
```
# b) Find out the coefficient of mean to standard deviation

from scipy.stats import variation
variation(sales.Weekly_Sales, axis = None)
```

Out[708…
```
0.5390083097474861
```

In [709…
```
# 3) Which store/s has good quarterly growth rate in Q3'2012
# extracting the data based on quarter and year
sales1= sales[['Date','Store','Weekly_Sales']]
sales1.Date= pd.to_datetime(sales1.Date)

sales1['Quarter']=sales1.Date.dt.quarter
sales1['Year'] = sales1.Date.dt.year
s1= sales1[sales1.Quarter==3]
s1= s1[s1.Year==2012]
s1.Weekly_Sales.groupby([sales1.Store,sales1.Quarter]).mean().sort_values(ascending=Fal
```

```
C:\Users\Linnet Ann Verghese\anaconda3\lib\site-packages\pandas\core\generic.py:5494: Se
ttingWithCopyWarning:
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
  self[name] = value
<ipython-input-709-66aa41f457c4>:5: 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
  sales1['Quarter']=sales1.Date.dt.quarter
<ipython-input-709-66aa41f457c4>:6: 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
  sales1['Year'] = sales1.Date.dt.year
```

Out[709…
```
Store  Quarter
4      3          2.137677e+06
Name: Weekly_Sales, dtype: float64
```

In [778…
```
#Answer is store 4 with weekly sales as 2.137677e+06
```

In [710…
```
# 4) Find out holidays which have higher sales than the mean sales in non-holiday seaso
#grouping the data based on Holiday_Flag
sales.Weekly_Sales.groupby(sales.Holiday_Flag).mean()
```

Out[710…
```
Holiday_Flag
0    1.041256e+06
1    1.122888e+06
Name: Weekly_Sales, dtype: float64
```

In [711…
```
sales.Holiday_Flag.value_counts()
```

Out[711…
```
0    5985
1     450
Name: Holiday_Flag, dtype: int64
```

In [712…
```
#extracting non holiday data
salesNonHoliday = sales[sales.Holiday_Flag==0]
```

In [713…
```
#finding the mean of the sales for non-holiday sales.
NonHolidaySales = salesNonHoliday.Weekly_Sales.mean()
NonHolidaySales
```

Out[713…
```
1041256.3802088564
```

In [714…
```
#finding the mean of the sales for Superbowl
SuperbowlDate1= sales[sales.Date=='12-02-2010' ]
SuperbowlDate2= sales[sales.Date=='11-02-2011' ]
SuperbowlDate3= sales[sales.Date=='10-02-2012']
SuperbowlDate4= sales[sales.Date=='08-02-2013']
SuperbowlDate = pd.concat([SuperbowlDate1,SuperbowlDate2,SuperbowlDate3,SuperbowlDate4]
superBowlSales= SuperbowlDate.Weekly_Sales.mean()
superBowlSales
```

Out[714…
```
1079127.9877037038
```

In [715…
```
#finding the mean of the sales for LaborDay
LaborDayDate1= sales[sales.Date=='10-09-2010' ]
LaborDayDate2= sales[sales.Date=='09-09-2011' ]
LaborDayDate3= sales[sales.Date=='07-09-2012' ]
LaborDayDate4= sales[sales.Date=='06-09-2013' ]
LaborDayDate = pd.concat([LaborDayDate1,LaborDayDate2,LaborDayDate3,LaborDayDate4],axis
LaborDaySales = LaborDayDate.Weekly_Sales.mean()
LaborDaySales
```

Out[715…
```
1042427.2939259262
```

In [716…
```
#finding the mean of the sales for Thanksgiving
ThanksgivingDate1 = sales[sales.Date=='26-11-2010' ]
ThanksgivingDate2 = sales[sales.Date=='25-11-2011' ]
ThanksgivingDate3 = sales[sales.Date=='23-11-2012' ]
```

```
ThanksgivingDate4 = sales[sales.Date=='29-11-20103' ]
ThanksgivingDate  = pd.concat([ThanksgivingDate1,ThanksgivingDate2,ThanksgivingDate3,Th
ThanksgivingSales = ThanksgivingDate.Weekly_Sales.mean()
ThanksgivingSales
```

Out[716… 1471273.4277777777

In [717…
```
#finding the mean of the sales for Christmas
ChristmasDate1 = sales[sales.Date=='31-12-2010' ]
ChristmasDate2 = sales[sales.Date=='30-12-2011' ]
ChristmasDate3 = sales[sales.Date=='28-12-2012' ]
ChristmasDate4 = sales[sales.Date=='27-12-2013' ]
ChristmasDate  = pd.concat([ChristmasDate1,ChristmasDate2,ChristmasDate3,ChristmasDate4
ChristmasSales = ChristmasDate.Weekly_Sales.mean()
ChristmasSales
```

Out[717… 960833.1115555555

In [718…
```
#comparing each of the holiday sales with non-holiday sales to get the final result
if (NonHolidaySales<ChristmasSales) :
    print("Christmas")
if (NonHolidaySales<ThanksgivingSales) :
    print("Thanksgiving")
if (NonHolidaySales<LaborDaySales) :
    print("Labor Day")
if (NonHolidaySales<superBowlSales) :
    print("Super Bowl")
```

```
Thanksgiving
Labor Day
Super Bowl
```

In [719…
```
# 5)Provide a monthly and semester view of sales in units and give insights
#grouping data based on month
sale= sales[['Date','Weekly_Sales']]
sale.Date= pd.to_datetime(sale.Date)
monthlysale =  sale.resample('m', on='Date').sum()
monthlysale
```

```
C:\Users\Linnet Ann Verghese\anaconda3\lib\site-packages\pandas\core\generic.py:5494: Se
ttingWithCopyWarning:
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
  self[name] = value
```

Out[719…

Date	Weekly_Sales
2010-01-31	4.223988e+07
2010-02-28	1.915869e+08
2010-03-31	1.862262e+08
2010-04-30	1.838118e+08

Date	Weekly_Sales
2010-05-31	2.806119e+08
2010-06-30	1.424361e+08
2010-07-31	1.842664e+08
2010-08-31	1.845381e+08
2010-09-30	1.797041e+08
2010-10-31	2.311201e+08
2010-11-30	1.587731e+08
2010-12-31	3.235716e+08
2011-01-31	2.119657e+08
2011-02-28	1.876092e+08
2011-03-31	1.365205e+08
2011-04-30	2.789693e+08
2011-05-31	1.828017e+08
2011-06-30	1.401936e+08
2011-07-31	2.244611e+08
2011-08-31	1.880810e+08
2011-09-30	2.310323e+08
2011-10-31	1.837193e+08
2011-11-30	2.534703e+08
2011-12-31	2.293760e+08
2012-01-31	1.722207e+08
2012-02-29	1.428296e+08
2012-03-31	2.307397e+08
2012-04-30	1.825428e+08
2012-05-31	1.422830e+08
2012-06-30	2.923883e+08
2012-07-31	1.845865e+08
2012-08-31	1.916126e+08
2012-09-30	1.797959e+08
2012-10-31	1.880794e+08
2012-11-30	4.692588e+07
2012-12-31	4.612851e+07

In [720…
```
#sorted monthly data based on sales
monthlysale.sort_values(by = 'Weekly_Sales', ascending=False)
```

Out[720…

Date	Weekly_Sales
2010-12-31	3.235716e+08
2012-06-30	2.923883e+08
2010-05-31	2.806119e+08
2011-04-30	2.789693e+08
2011-11-30	2.534703e+08
2010-10-31	2.311201e+08
2011-09-30	2.310323e+08
2012-03-31	2.307397e+08
2011-12-31	2.293760e+08
2011-07-31	2.244611e+08
2011-01-31	2.119657e+08
2012-08-31	1.916126e+08
2010-02-28	1.915869e+08
2011-08-31	1.880810e+08
2012-10-31	1.880794e+08
2011-02-28	1.876092e+08
2010-03-31	1.862262e+08
2012-07-31	1.845865e+08
2010-08-31	1.845381e+08
2010-07-31	1.842664e+08
2010-04-30	1.838118e+08
2011-10-31	1.837193e+08
2011-05-31	1.828017e+08
2012-04-30	1.825428e+08
2012-09-30	1.797959e+08
2010-09-30	1.797041e+08
2012-01-31	1.722207e+08
2010-11-30	1.587731e+08
2012-02-29	1.428296e+08
2010-06-30	1.424361e+08
2012-05-31	1.422830e+08

Date	Weekly_Sales
2011-06-30	1.401936e+08
2011-03-31	1.365205e+08
2012-11-30	4.692588e+07
2012-12-31	4.612851e+07
2010-01-31	4.223988e+07

In [721…
```
monthlysale.sort_values(by = 'Weekly_Sales', ascending=False).iloc[:1] #highest sale wa
```

Out[721…

Date	Weekly_Sales
2010-12-31	3.235716e+08

In [722…
```
monthlysale.sort_values(by = 'Weekly_Sales', ascending=False).iloc[-1:] #lowest sale wa
```

Out[722…

Date	Weekly_Sales
2010-01-31	42239875.87

In [723…
```
#grouping data based on semester
semestersale=  sale.resample('q',closed='left',  on='Date').sum()
semestersale
```

Out[723…

Date	Weekly_Sales
2010-03-31	4.200530e+08
2010-06-30	6.068598e+08
2010-09-30	5.485085e+08
2010-12-31	6.730324e+08
2011-03-31	5.765279e+08
2011-06-30	6.019646e+08
2011-09-30	6.013786e+08
2011-12-31	7.087615e+08
2012-03-31	5.457900e+08
2012-06-30	6.172141e+08
2012-09-30	5.559950e+08

Date	Weekly_Sales
2012-12-31	2.811338e+08

In [724…
```
#sorted semester data based on sales
semestersale.sort_values(by = 'Weekly_Sales', ascending=False)
```

Out[724…

Date	Weekly_Sales
2011-12-31	7.087615e+08
2010-12-31	6.730324e+08
2012-06-30	6.172141e+08
2010-06-30	6.068598e+08
2011-06-30	6.019646e+08
2011-09-30	6.013786e+08
2011-03-31	5.765279e+08
2012-09-30	5.559950e+08
2010-09-30	5.485085e+08
2012-03-31	5.457900e+08
2010-03-31	4.200530e+08
2012-12-31	2.811338e+08

In [725…
```
semestersale.sort_values(by = 'Weekly_Sales', ascending=False).iloc[:1]
#highest sale was in the semester ending Dec 2011 of 7.087615e+08
```

Out[725…

Date	Weekly_Sales
2011-12-31	7.087615e+08

In [726…
```
semestersale.sort_values(by = 'Weekly_Sales', ascending=False).iloc[-1:]
 #Lowest sale was in the semester ending Dec 2012 of 2.811338e+08
```

Out[726…

Date	Weekly_Sales
2012-12-31	2.811338e+08

In [727…
```
# 6 a)Utilize variables like date and restructure dates as 1 for 5 Feb 2010 (starting f
#extracting store1's data
```

```
sales.Date= pd.to_datetime(sales.Date)
newSales = sales[sales.Store==1]
newSales = newSales.sort_values(by = 'Date', ascending=True)
newSales
```

Out[727…

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
34	1	2010-01-10	1453329.50	0	71.89	2.603	211.671989	7.838
8	1	2010-02-04	1594968.28	0	62.27	2.719	210.820450	7.808
21	1	2010-02-07	1492418.14	0	80.91	2.669	211.223533	7.787
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106
…	…	…	…	…	…	…	…	…
131	1	2012-10-08	1592409.97	0	85.05	3.494	221.958433	6.908
141	1	2012-10-19	1508068.77	0	67.97	3.594	223.425723	6.573
142	1	2012-10-26	1493659.74	0	69.16	3.506	223.444251	6.573
118	1	2012-11-05	1611096.05	0	73.77	3.688	221.725663	7.143
140	1	2012-12-10	1573072.81	0	62.99	3.601	223.381296	6.573

143 rows × 8 columns

In [728…

```
#extracting date alone and label encoding it
Datecolumn = newSales[['Date']]
```

In [729…

```
from sklearn.preprocessing import LabelEncoder
```

In [730…

```
le = LabelEncoder()
```

In [731…

```
newdate= Datecolumn.apply(le.fit_transform)
```

In [732…

```
newdate
```

Out[732…

	Date
34	0

	Date
8	1
21	2
2	3
3	4
...	...
131	138
141	139
142	140
118	141
140	142

143 rows × 1 columns

In [734...
```
#dropping unnecessary columns
newSales= newSales.drop(['Date','Holiday_Flag','Temperature','Store'],axis=1)
newSales
```

Out[734...

	Weekly_Sales	Fuel_Price	CPI	Unemployment
34	1453329.50	2.603	211.671989	7.838
8	1594968.28	2.719	210.820450	7.808
21	1492418.14	2.669	211.223533	7.787
2	1611968.17	2.514	211.289143	8.106
3	1409727.59	2.561	211.319643	8.106
...	...	...	...	...
131	1592409.97	3.494	221.958433	6.908
141	1508068.77	3.594	223.425723	6.573
142	1493659.74	3.506	223.444251	6.573
118	1611096.05	3.688	221.725663	7.143
140	1573072.81	3.601	223.381296	6.573

143 rows × 4 columns

In [735...
```
#adding the label encoded date back into the cleaned dataframe
cleanedSales = pd.concat([NewSales,newdate],axis=1)
```

In [738...
```
#6b)Linear Regression using the Date
from sklearn.linear_model import LinearRegression
```

In [739…
```
reg=LinearRegression()
```

In [740…
```
#setting up the depenedent(y) and independent variable(X)
y=cleanedSales.Weekly_Sales
X=cleanedSales.drop(['Weekly_Sales'],axis=1)
```

In [741…
```
regmodel=reg.fit(X,y)
```

In [742…
```
regmodel.score(X,y) #data is underfitting
```

Out[742… 0.1755151618042894

In [743…
```
#Linear regression without date
Saleswithoutdate=cleanedSales.drop(['Date'],axis=1)
```

In [744…
```
#Hypothesize if CPI, unemployment, and fuel price have any impact on sales.
y1=Saleswithoutdate.Weekly_Sales
X1=Saleswithoutdate.drop(['Weekly_Sales'],axis=1)
```

In [745…
```
regmodel1=reg.fit(X1,y1)
```

In [746…
```
regmodel1.score(X1,y1) #data is underfitting
```

Out[746… 0.17446353549276172

In [747…
```
regmodelpredict= regmodel1.predict(X1)
```

In [749…
```
from sklearn.tree import DecisionTreeRegressor
```

In [750…
```
tree = DecisionTreeRegressor()
```

In [751…
```
treemodel = tree.fit(X1,y1)
```

In [752…
```
treemodel.score(X1,y1) #data is overfitting
```

Out[752… 1.0

In [753…
```
from sklearn.neighbors import KNeighborsRegressor
```

In [754…
```
from sklearn.model_selection import GridSearchCV
```

In [755…
```
parameters = {'n_neighbors':range(1,50),'weights':['uniform','distance']}
```

In [756…
```
gridsearch = GridSearchCV(KNeighborsRegressor(),parameters)
#grid search will build knn models
```

In [757…
```
gridsearchmodel =gridsearch.fit(X1,y1)
```

In [758…
```
gridsearchmodel.best_params_
```

Out[758… {'n_neighbors': 24, 'weights': 'uniform'}

In [771…
```
knnreg = KNeighborsRegressor(n_neighbors=5, metric = 'euclidean')
```

In [772…
```
knnregmodel = knnreg.fit(X1,y1)
```

In [773…
```
knnregmodel.score(X1,y1) #r score is better with n_neighbors=5 than 24 but its still un
```

Out[773… 0.3577233858056815

In [774…
```
knnregpredict = knnregmodel.predict(X1)
```

In [775…
```
knnresidual = y-knnregpredict
```

In [776…
```
np.sqrt(np.mean(knnresidual**2))
```

Out[776… 124568.50771167203

In [ ]:
```
# Change dates into days by creating new variable.
```

In [777…
```
sales.head()
```

Out[777…

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	2010-05-02	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	2010-12-02	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
4	1	2010-05-03	1554806.68	0	46.50	2.625	211.350143	8.106

In [684…
```
sales1= sales
sales1.Date= pd.to_datetime(sales1.Date)
```

In [685…
```
#finding the Day from the date
sales1['Day']= sales1['Date'].dt.day
```

In [686…
```
sales1.head() #showing the 1st  5 rows
```

Out[686…

	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	2
1	1	2010-12-02	1641957.44	1	38.51	2.548	211.242170	8.106	2
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.106	19
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.106	26
4	1	2010-05-03	1554806.68	0	46.50	2.625	211.350143	8.106	3

In [687…
```
sales1.drop(['Date'],axis=1) #dropping the date.
```

Out[687…

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

6435 rows × 8 columns

In [ ]: