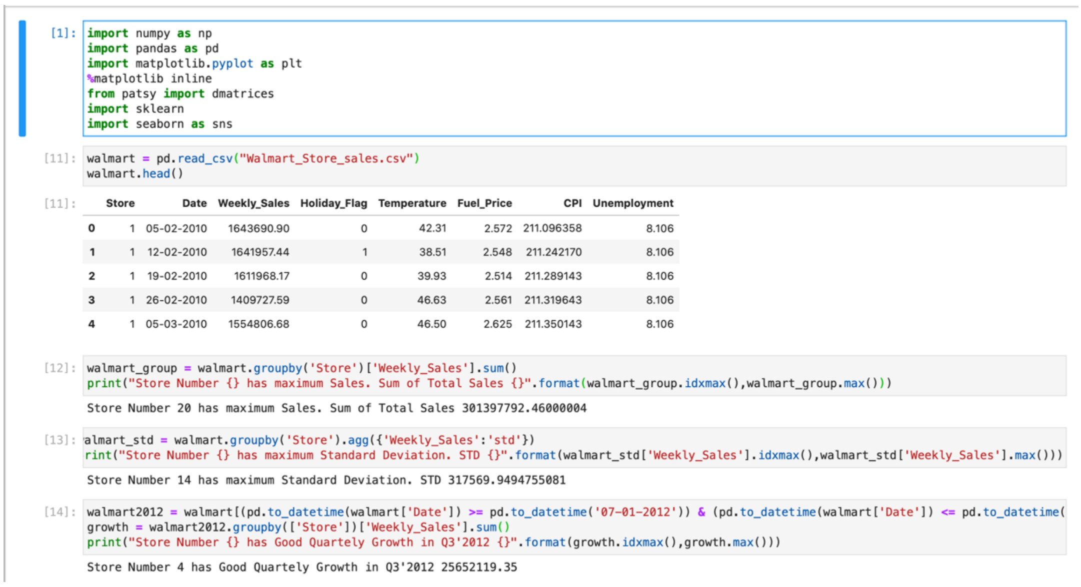
Simplilearn
Data Science with Python
Project Assessment - Retail Analysis with Walmart Data
Screenshots

NAME: Krithika Balasubramanian Email id: krithika1798@gmail.com

Basic Statistical Tasks

- Which store has maximum sales
- Which store has maximum standard deviation i.e., the sales vary a lot. Also, find out the coefficient of mean to standard deviation
- Which store/s has good quarterly growth rate in Q3’2012



```
[1]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     %matplotlib inline
     from patsy import dmatrices
     import sklearn
     import seaborn as sns

[11]: walmart = pd.read_csv("Walmart_Store_sales.csv")
      walmart.head()
```

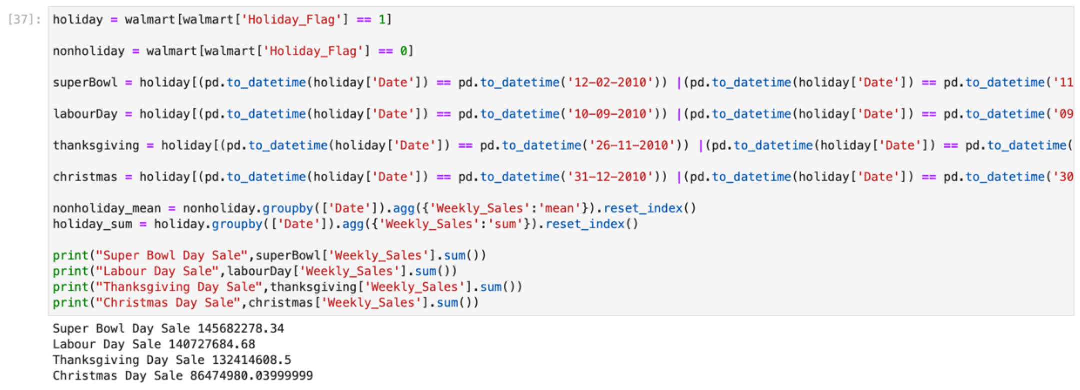
[11]:

	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

```
[12]: walmart_group = walmart.groupby('Store')['Weekly_Sales'].sum()
      print("Store Number {} has maximum Sales. Sum of Total Sales {}".format(walmart_group.idxmax(),walmart_group.max()))

      Store Number 20 has maximum Sales. Sum of Total Sales 301397792.46000004

[13]: almart_std = walmart.groupby('Store').agg({'Weekly_Sales':'std'})
      rint("Store Number {} has maximum Standard Deviation. STD {}".format(walmart_std['Weekly_Sales'].idxmax(),walmart_std['Weekly_Sales'].max()))

      Store Number 14 has maximum Standard Deviation. STD 317569.9494755081

[14]: walmart2012 = walmart[(pd.to_datetime(walmart['Date']) >= pd.to_datetime('07-01-2012')) & (pd.to_datetime(walmart['Date']) <= pd.to_datetime(
      growth = walmart2012.groupby(['Store'])['Weekly_Sales'].sum()
      print("Store Number {} has Good Quartely Growth in Q3'2012 {}".format(growth.idxmax(),growth.max()))

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

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

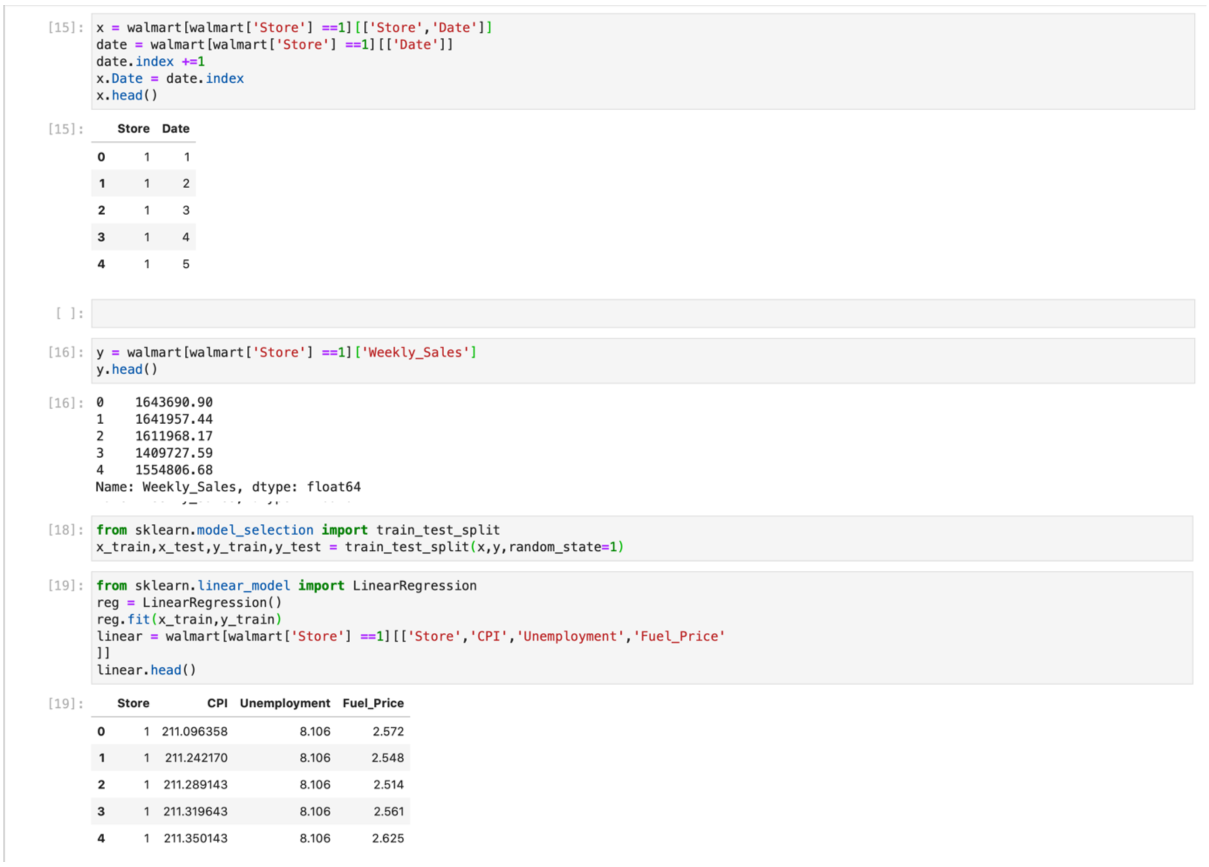
- Provide a monthly and semester view of sales in units and give insights



```
[37]: holiday = walmart[walmart['Holiday_Flag'] == 1]

      nonholiday = walmart[walmart['Holiday_Flag'] == 0]

      superBowl = holiday[(pd.to_datetime(holiday['Date']) == pd.to_datetime('12-02-2010')) |(pd.to_datetime(holiday['Date']) == pd.to_datetime('11

      labourDay = holiday[(pd.to_datetime(holiday['Date']) == pd.to_datetime('10-09-2010')) |(pd.to_datetime(holiday['Date']) == pd.to_datetime('09

      thanksgiving = holiday[(pd.to_datetime(holiday['Date']) == pd.to_datetime('26-11-2010')) |(pd.to_datetime(holiday['Date']) == pd.to_datetime(

      christmas = holiday[(pd.to_datetime(holiday['Date']) == pd.to_datetime('31-12-2010')) |(pd.to_datetime(holiday['Date']) == pd.to_datetime('30

      nonholiday_mean = nonholiday.groupby(['Date']).agg({'Weekly_Sales':'mean'}).reset_index()
      holiday_sum = holiday.groupby(['Date']).agg({'Weekly_Sales':'sum'}).reset_index()

      print("Super Bowl Day Sale",superBowl['Weekly_Sales'].sum())
      print("Labour Day Sale",labourDay['Weekly_Sales'].sum())
      print("Thanksgiving Day Sale",thanksgiving['Weekly_Sales'].sum())
      print("Christmas Day Sale",christmas['Weekly_Sales'].sum())

      Super Bowl Day Sale 145682278.34
      Labour Day Sale 140727684.68
      Thanksgiving Day Sale 132414608.5
      Christmas Day Sale 86474980.03999999
```

For Store 1 – Build prediction models to forecast demand

- 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.



[15]:
```
x = walmart[walmart['Store'] ==1][['Store','Date']]
date = walmart[walmart['Store'] ==1][['Date']]
date.index +=1
x.Date = date.index
x.head()
```

[15]:

	Store	Date
0	1	1
1	1	2
2	1	3
3	1	4
4	1	5

[ ]:

[16]:
```
y = walmart[walmart['Store'] ==1]['Weekly_Sales']
y.head()
```

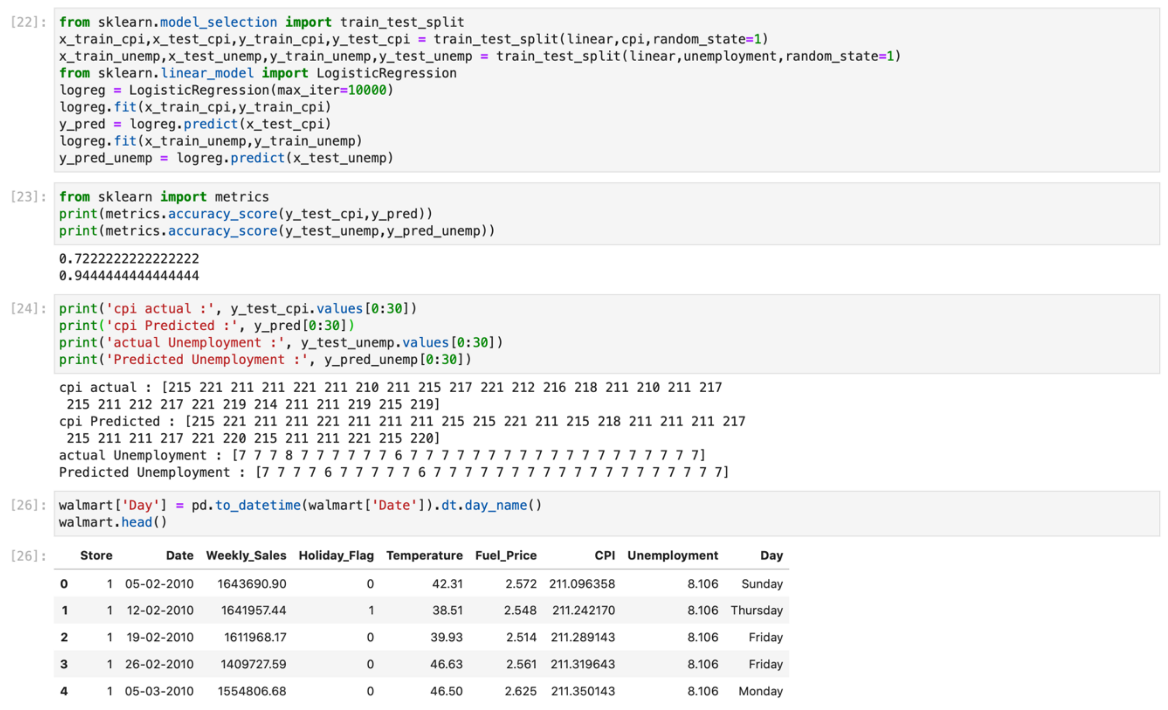
[16]:
```
0    1643690.90
1    1641957.44
2    1611968.17
3    1409727.59
4    1554806.68
Name: Weekly_Sales, dtype: float64
```

[18]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=1)
```

[19]:
```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(x_train,y_train)
linear = walmart[walmart['Store'] ==1][['Store','CPI','Unemployment','Fuel_Price'
]]
linear.head()
```

[19]:

	Store	CPI	Unemployment	Fuel_Price
0	1	211.096358	8.106	2.572
1	1	211.242170	8.106	2.548
2	1	211.289143	8.106	2.514
3	1	211.319643	8.106	2.561
4	1	211.350143	8.106	2.625



[22]:
```
from sklearn.model_selection import train_test_split
x_train_cpi,x_test_cpi,y_train_cpi,y_test_cpi = train_test_split(linear,cpi,random_state=1)
x_train_unemp,x_test_unemp,y_train_unemp,y_test_unemp = train_test_split(linear,unemployment,random_state=1)
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(max_iter=10000)
logreg.fit(x_train_cpi,y_train_cpi)
y_pred = logreg.predict(x_test_cpi)
logreg.fit(x_train_unemp,y_train_unemp)
y_pred_unemp = logreg.predict(x_test_unemp)
```

[23]:
```
from sklearn import metrics
print(metrics.accuracy_score(y_test_cpi,y_pred))
print(metrics.accuracy_score(y_test_unemp,y_pred_unemp))
```

```
0.7222222222222222
0.9444444444444444
```

[24]:
```
print('cpi actual :', y_test_cpi.values[0:30])
print('cpi Predicted :', y_pred[0:30])
print('actual Unemployment :', y_test_unemp.values[0:30])
print('Predicted Unemployment :', y_pred_unemp[0:30])
```

```
cpi actual : [215 221 211 211 221 211 210 211 215 217 221 212 216 218 211 210 211 217
 215 211 212 217 221 219 214 211 211 219 215 219]
cpi Predicted : [215 221 211 211 221 211 211 211 215 215 221 211 215 218 211 211 211 217
 215 211 211 217 221 220 215 211 211 221 215 220]
actual Unemployment : [7 7 7 8 7 7 7 7 7 7 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7]
Predicted Unemployment : [7 7 7 7 6 7 7 7 7 7 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7]
```

[26]:
```
walmart['Day'] = pd.to_datetime(walmart['Date']).dt.day_name()
walmart.head()
```

[26]:

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