Untitled13

October 10, 2021

[131]:
```
import pandas as pd
import numpy as np

from datetime import datetime as date

import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

from sklearn.linear_model import LinearRegression
```

[46]:
```
data = pd.read_csv('Walmart_Store_sales.csv')
```

[47]:
```
data
```

[47]:

	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		
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		

```
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]
```

[48]:
```
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
```

[49]:
```
data.describe()
```

[49]:

	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		
75%	34.000000	1.420159e+06	0.000000	74.940000	3.735000		
max	45.000000	3.818686e+06	1.000000	100.140000	4.468000		

```
50%     182.616521      7.874000
75%     212.743293      8.622000
max     227.232807     14.313000
```

[50]:
```
data.duplicated().sum()
```

[50]:
```
0
```

0.0.1 Store with Maximum Sales

[51]:
```
stores_sales = data.groupby('Store').sum()['Weekly_Sales']
top_shop = stores_sales.sort_values(ascending = False).index[0]
print('Store no. {} has Maximum Sales'.format(top_shop))
```

```
Store no. 20 has Maximum Sales
```

0.0.2 Store with Maximum Standard Deviation

[52]:
```
std_store = data.groupby('Store').std()['Weekly_Sales']
max_std = std_store.sort_values(ascending = False)
print('Store no. {} has Maximum Standard Deviation of {}'.format(max_std.
 ↪index[0],max_std.max()))
```

```
Store no. 14 has Maximum Standard Deviation of 317569.9494755081
```

0.0.3 Store/s has good Quarterly Growth Rate in Q3'2012

[77]:
```
data['Date'] = pd.to_datetime(data['Date'])

Q3_date_from = pd.Timestamp(date(2012,7,1))
Q3_date_to = pd.Timestamp(date(2012,9,30))
Q2_date_from = pd.Timestamp(date(2012,4,1))
Q2_date_to = pd.Timestamp(date(2012,6,30))

#Collecting the data of Q3 and Q2 from original dataset.
Q2data=data[(data['Date'] > Q2_date_from) & (data['Date'] < Q2_date_to)]
Q3data=data[(data['Date'] > Q3_date_from) & (data['Date'] < Q3_date_to)]

# finding the sum weekly sales of each store in Q2
Q2 = pd.DataFrame(Q2data.groupby('Store')['Weekly_Sales'].sum())
Q2.reset_index(inplace=True)
Q2.rename(columns={'Weekly_Sales': 'Q2_Weekly_Sales'},inplace=True)

# finding the sum weekly sales of each store in Q3
```

```
Q3 = pd.DataFrame(Q3data.groupby('Store')['Weekly_Sales'].sum())
Q3.reset_index(inplace = True)
Q3.rename(columns = {'Weekly_Sales': 'Q3_Weekly_Sales'},inplace=True)

# mergeing Q2 and Q3 data on Store as a common column
Q3_Growth= Q2.merge(Q3,how='inner',on='Store')

# Growth rate formula is defined as the ratio of difference in present value to␣
 ↪past value by past value whole multiplied with 100
# (since it is in percentage)
# ((Present value - Past value )/Past value )*100
# Calculating Growth rate of each Store and collecting it into a dataframe
Q3_Growth['Growth_Rate'] =(Q3_Growth['Q3_Weekly_Sales'] -␣
 ↪Q3_Growth['Q2_Weekly_Sales'])/Q3_Growth['Q2_Weekly_Sales']
Q3_Growth['Growth_Rate'] =round(Q3_Growth['Growth_Rate'],2)
Q3_Growth.sort_values('Growth_Rate',ascending=False).head(1)
```

[77]:

	Store	Q2_Weekly_Sales	Q3_Weekly_Sales	Growth_Rate
15	16	6626133.44	6441311.11	-0.03

0.0.4 Holidays which have higher sales than the Mean Sales in Non-Holiday Season for all Stores together.

[117]:
```
superbowl_sales = data[data["Date"] == "12-02-2010"]['Weekly_Sales'].sum() +␣
 ↪data[data["Date"] == "11-02-2011"]['Weekly_Sales'].sum() + data[data["Date"]␣
 ↪== "10-02-2012"]['Weekly_Sales'].sum()
labour_day_sales = data[data["Date"] == '10-09-2010']['Weekly_Sales'].sum() +␣
 ↪data[data["Date"] == '09-09-2011']['Weekly_Sales'].sum() + data[data["Date"]␣
 ↪== '07-09-2012']['Weekly_Sales'].sum()
thanksgiving_sales = data[data["Date"] == '26-11-2010']['Weekly_Sales'].sum() +␣
 ↪data[data["Date"] == '25-11-2011']['Weekly_Sales'].sum() + data[data["Date"]␣
 ↪== '23-11-2012']['Weekly_Sales'].sum()
christmas_sales = data[data["Date"] == '31-12-2010']['Weekly_Sales'].sum() +␣
 ↪data[data["Date"]== '30-12-2011']['Weekly_Sales'].sum() + data[data["Date"]␣
 ↪== '28-12-2012']['Weekly_Sales'].sum()
```

[118]:
```
print(superbowl_sales,labour_day_sales,thanksgiving_sales,christmas_sales)
```

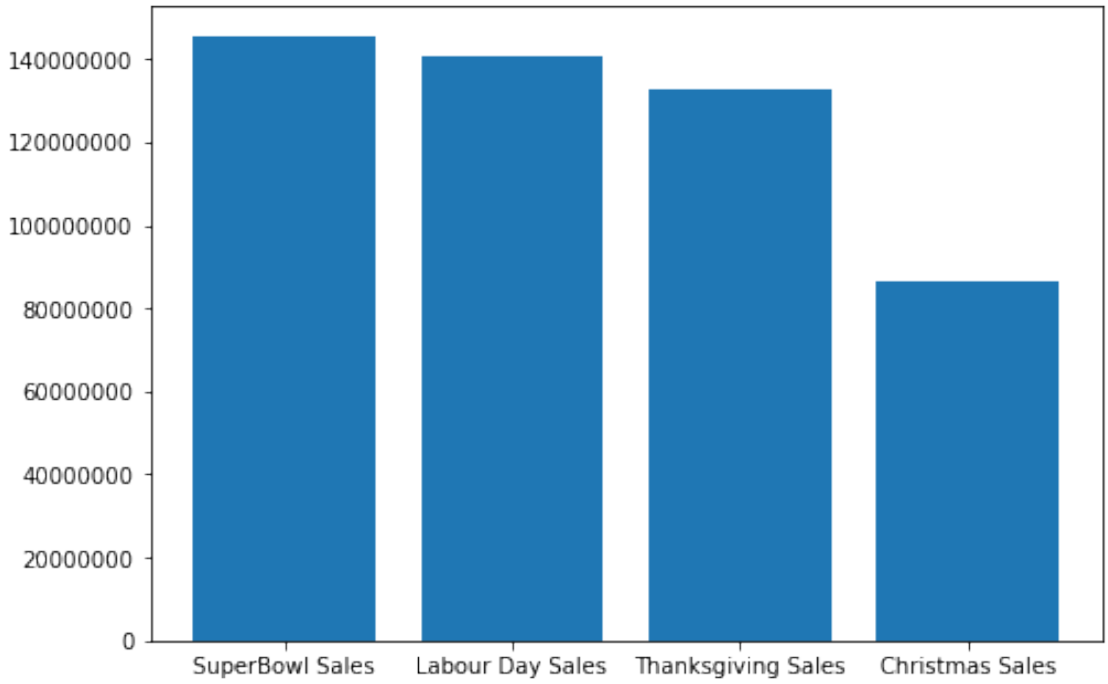
```
145682278.34000003 140727684.68 132414608.5 86474980.04
```

[111]:
```
(data['Weekly_Sales'][data.Holiday_Flag == 0]).mean()
```

[111]:
```
1041256.3802088564
```

[119]:
```
fig = plt.figure()
holidays = ["SuperBowl Sales","Labour Day Sales","Thanksgiving␣
 ↪Sales","Christmas Sales"]
sales = [superbowl_sales,labour_day_sales,thanksgiving_sales,christmas_sales]
ax = fig.add_axes([0,0,1,1])
ax.ticklabel_format(style = 'plain')
ax.bar(holidays,sales)

plt.show()
```



Christmas Sales are under average

[124]:
```
data
data['Date'] = pd.to_datetime(data['Date'])
monthly_sales = data.groupby([data['Date'],data['Store']])['Weekly_Sales'].
 ↪agg(sum).reset_index()
monthly_sales['Year'] = monthly_sales['Date'].dt.year
monthly_sales['Month'] = monthly_sales['Date'].dt.month
monthly_sales['Quater'] = monthly_sales['Date'].dt.quarter
```

[132]:
```
for month in range(1,13):
    df_copy=monthly_sales[monthly_sales["Month"]==month]
```

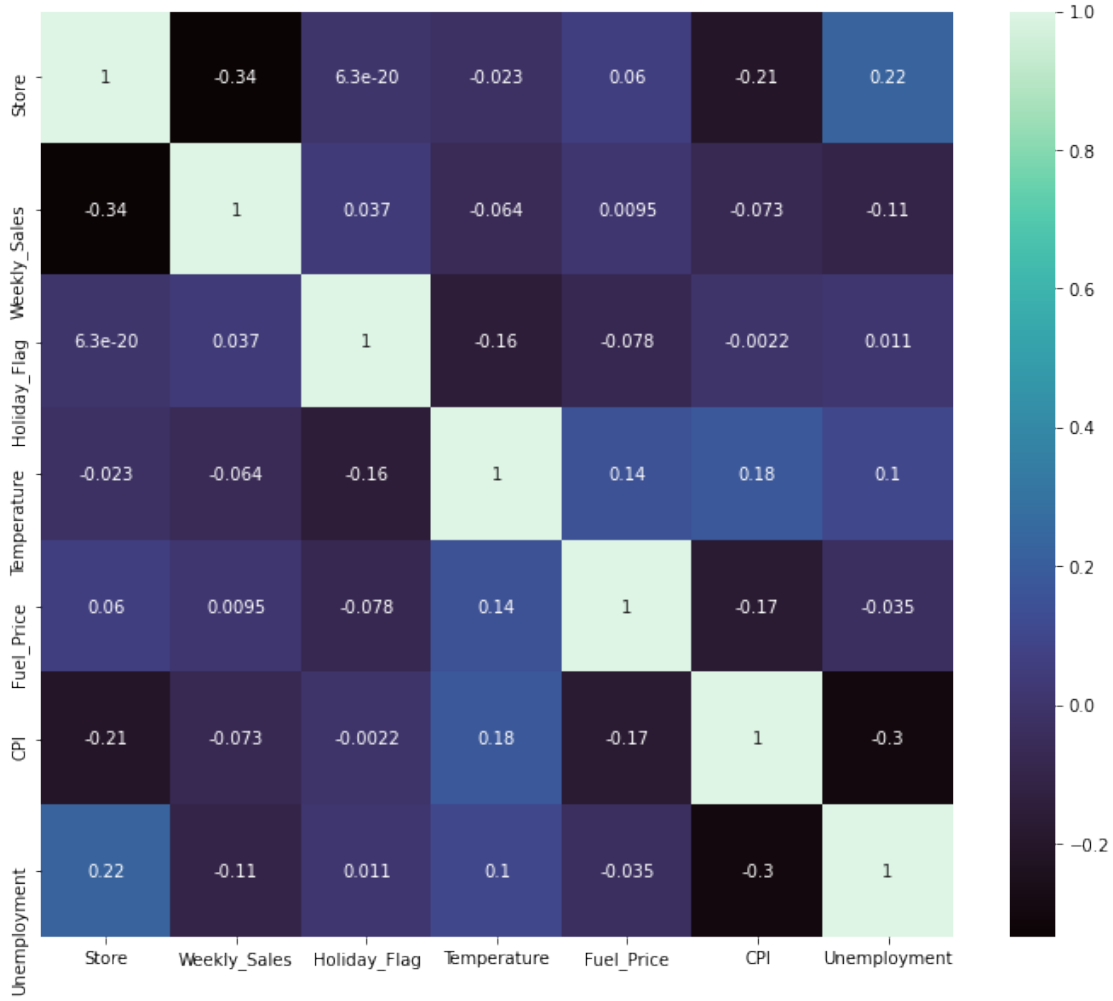
```
    fig = px.bar(df_copy, x="Store",␣
↪y="Weekly_Sales",color='Weekly_Sales',width=500, height=400,title="Month␣
↪"+str(month)+" sales by store",
                 labels={
                     "Weekly_Sales": "Sales",
                     "sepal_width": "Store number",

                 },)
    fig.show()
```

[142]:
```
correl = data.corr()
correl

plt.figure(figsize=(12, 10))
sns.heatmap(correl, cmap = 'mako', annot = True)
```

[142]: <AxesSubplot:>



[143]:
```
y = pd.get_dummies(data['Store']) # OneHot-Encoding the store
data = data.drop('Store', axis = 1)

data = data.join(y)
```

[146]:
```
y = data.Weekly_Sales
x = data.drop(['Weekly_Sales','Date'], axis = 1)

ss = StandardScaler()
scaled_temp = ss.fit_transform(data[['Temperature']])
data['Temperature'] = scaled_temp

scaled_fuel = ss.fit_transform(data[['Fuel_Price']])
data['Fuel_Price'] = scaled_fuel

scaled_unemployment = ss.fit_transform(data[['Unemployment']])
data['Unemployment'] = scaled_unemployment
```

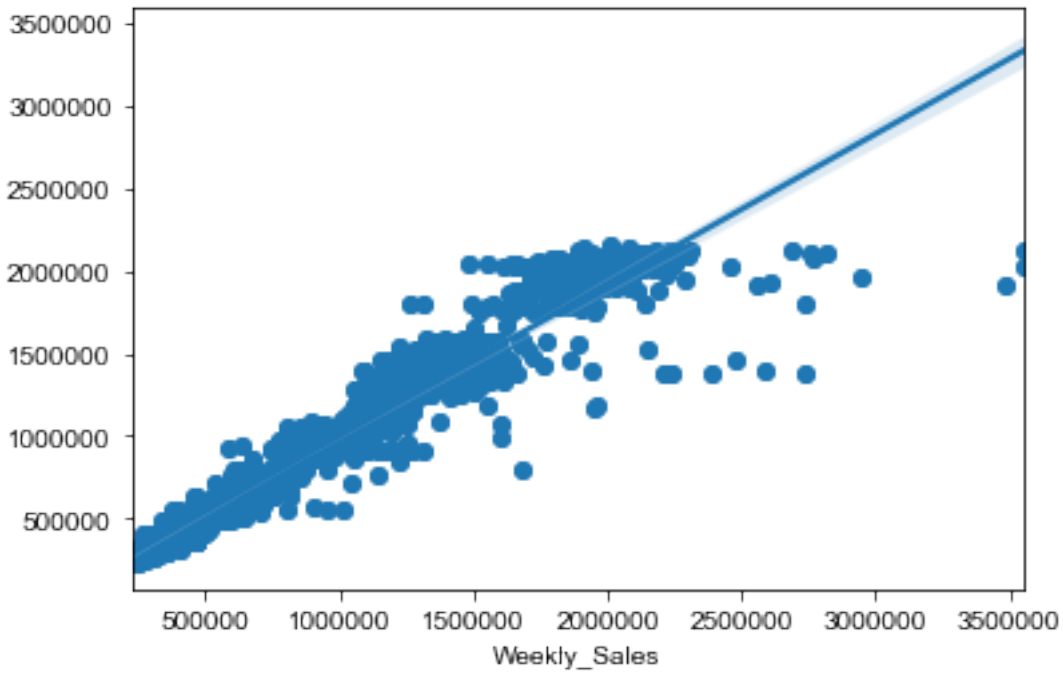
[147]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size = 0.2,␣
 ↪random_state = 1)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(5148, 50)
(5148,)
(1287, 50)
(1287,)
```

[148]:
```
lm = LinearRegression()
lm.fit(x_train,y_train)

y_pred = lm.predict(x_test)
```

[150]:
```
plt.scatter(y_test,y_pred)
plt.ticklabel_format(style = 'plain')
sns.set(rc={'figure.figsize':(18,12)})
sns.regplot(x=y_test, y=y_pred);
```



[152]:
```
score = r2_score(y_test,y_pred)
```

[156]:
```
print("r2 score is ",score) # accuracy
print("mean_sqrd_error is ",mean_squared_error(y_test,y_pred))
print("root_mean_squared error of is ",np.
 ↪sqrt(mean_squared_error(y_test,y_pred)))
```

```
r2 score is  0.9135089173328298
mean_sqrd_error is  28573532531.85688
root_mean_squared error of is  169037.07443001043
```

[ ]:
```

```