In [1]:

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings #  for avoid unwanted warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
os.getcwd()
```

Out[2]:

```
'C:\\Users\\Preksha\\Simplilearn\\Data Science with Python\\Walmart'
```

In [3]:

```
dataset = pd.read_csv('Walmart_Store_sales.csv')
```

In [4]:

```
dataset.head()
```

Out[4]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployme
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.1
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.1
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.1
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.1
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.1

In [5]:

```
dataset.shape
```

Out[5]:

```
(6435, 8)
```

In [6]:

```
dataset.size
```

Out[6]:

```
51480
```

In [7]:

```
dataset.columns
```

Out[7]:

```
Index(['Store', 'Date', 'Weekly_Sales', 'Holiday_Flag', 'Temperature',
       'Fuel_Price', 'CPI', 'Unemployment'],
      dtype='object')
```

In [8]:

```
dataset.dtypes
```

Out[8]:

```
Store             int64
Date             object
Weekly_Sales    float64
Holiday_Flag      int64
Temperature     float64
Fuel_Price      float64
CPI             float64
Unemployment    float64
dtype: object
```

In [9]:

```
dataset.corr()
```

Out[9]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI
Store	1.000000e+00	-0.335332	6.250842e-20	-0.022659	0.060023	-0.209492
Weekly_Sales	-3.353320e-01	1.000000	3.689097e-02	-0.063810	0.009464	-0.072634
Holiday_Flag	6.250842e-20	0.036891	1.000000e+00	-0.155091	-0.078347	-0.002162
Temperature	-2.265908e-02	-0.063810	-1.550913e-01	1.000000	0.144982	0.176888
Fuel_Price	6.002295e-02	0.009464	-7.834652e-02	0.144982	1.000000	-0.170642
CPI	-2.094919e-01	-0.072634	-2.162091e-03	0.176888	-0.170642	1.000000
Unemployment	2.235313e-01	-0.106176	1.096028e-02	0.101158	-0.034684	-0.302020

In [10]:

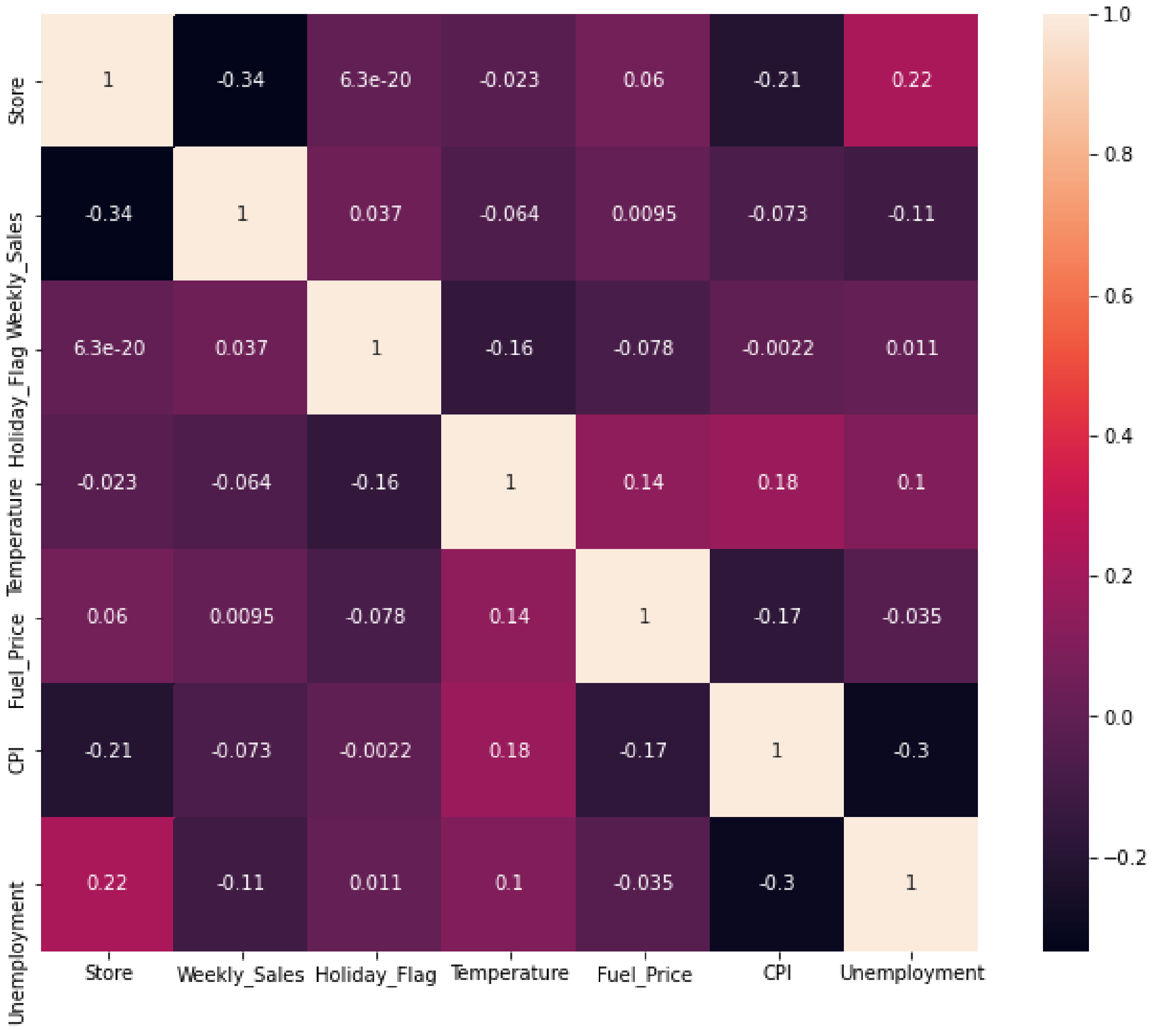
```
dataset.describe()
```

Out[10]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Uner
count	6435.000000	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	64
mean	23.000000	1.046965e+06	0.069930	60.663782	3.358607	171.578394	
std	12.988182	5.643666e+05	0.255049	18.444933	0.459020	39.356712	
min	1.000000	2.099862e+05	0.000000	-2.060000	2.472000	126.064000	
25%	12.000000	5.533501e+05	0.000000	47.460000	2.933000	131.735000	
50%	23.000000	9.607460e+05	0.000000	62.670000	3.445000	182.616521	
75%	34.000000	1.420159e+06	0.000000	74.940000	3.735000	212.743293	
max	45.000000	3.818686e+06	1.000000	100.140000	4.468000	227.232807	

In [11]:

```
f, ax = plt.subplots(figsize = (12,9))
sns.heatmap(data = dataset.corr(), square = True, annot = True);
```



In [12]:

```
dataset.isna().sum()
```

Out[12]:

```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
dtype: int64
```

Which store has maximum sales

In [13]:

```
dataset['Weekly_Sales'].max()
```

Out[13]:

```
3818686.45
```

In [14]:

```
max_sales = dataset[dataset['Weekly_Sales'] == 3818686.45]
max_sales
```

Out[14]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemploy
1905	14	24-12-2010	3818686.45	0	30.59	3.141	182.54459	

Which 10 stores have maximum sales

In [15]:

```
top_sales = dataset.sort_values('Weekly_Sales', ascending = False)
```

In [16]:

```
top_sales.head(10)
```

Out[16]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemplo
1905	14	24-12-2010	3818686.45	0	30.59	3.141	182.544590	
2763	20	24-12-2010	3766687.43	0	25.17	3.141	204.637673	
1333	10	24-12-2010	3749057.69	0	57.06	3.236	126.983581	
527	4	23-12-2011	3676388.98	0	35.92	3.103	129.984548	
1762	13	24-12-2010	3595903.20	0	34.90	2.846	126.983581	
1814	13	23-12-2011	3556766.03	0	24.76	3.186	129.984548	
2815	20	23-12-2011	3555371.03	0	40.19	3.389	212.236040	
475	4	24-12-2010	3526713.39	0	43.21	2.887	126.983581	
1385	10	23-12-2011	3487986.89	0	48.36	3.541	129.984548	
189	2	24-12-2010	3436007.68	0	49.97	2.886	211.064660	

Which store has maximum standard deviation i.e., the sales vary a lot.

In [17]:

```
stores = dataset.groupby('Store')['Weekly_Sales'].std().sort_values(ascending
stores.head()
```

Out[17]:

```
Store
14    317569.949476
10    302262.062504
20    275900.562742
4     266201.442297
13    265506.995776
Name: Weekly_Sales, dtype: float64
```

AS SEEN HERE STORE 14 HAS MAXIMUM STANDARD DEVIATION AMONGST ALL.

In [18]:

```
import datetime as dt
```

In [19]:

```
dataset['Date'] = pd.to_datetime(dataset['Date'])
```

In [20]:

```
dataset['Year'] = dataset['Date'].dt.year
dataset['Month'] = dataset['Date'].dt.month
dataset['Quarter'] = dataset['Date'].dt.quarter
dataset['Day'] = dataset['Date'].dt.day
```

In [21]:

```
dataset.head()
```

Out[21]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemploym
0	1	2010-05-02	1643690.90	0	42.31	2.572	211.096358	8.
1	1	2010-12-02	1641957.44	1	38.51	2.548	211.242170	8.
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	8.
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	8.
4	1	2010-05-03	1554806.68	0	46.50	2.625	211.350143	8.

Which store/s has good quarterly growth rate in Q3’2012

In [22]:
```
Q3_sales = dataset[(dataset['Quarter'] == 3) & (dataset['Year'] == 2012)]
Q3_sales
```

Out[22]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Une
109	1	2012-09-03	1675431.16	0	58.76	3.669	221.059189	
122	1	2012-08-06	1697230.96	0	78.30	3.452	221.749484	
127	1	2012-07-13	1527014.04	0	77.12	3.256	221.924158	
128	1	2012-07-20	1497954.76	0	80.42	3.311	221.932727	
129	1	2012-07-27	1439123.71	0	82.66	3.407	221.941295	
...	...	...	...	...	...	...	...	
6426	45	2012-08-31	734297.87	0	75.09	3.867	191.461281	

In [23]:
```
Q3_sales = Q3_sales.groupby('Store')['Weekly_Sales'].sum().sort_values(ascenc
Q3_sales.head()
```

Out[23]:
```
Store
4     25652119.35
20    24665938.11
13    24319994.35
2     22396867.61
10    21169356.45
Name: Weekly_Sales, dtype: float64
```

THE TOP 5 STORES WITH GOOD QUARTERLY GROWTH RATES ARE LISTED ABOVE

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

In [24]:
```
super_bowl_sales = dataset[(dataset['Date']== '12-02-2010') | (dataset['Date'
                           (dataset['Date']== '10-02-2012') | (dataset['Date'
super_bowl_sales
```

Out[24]:
```
1079127.9877037033
```

In [25]:

```
labour_day_sales = dataset[(dataset['Date']== '10-09-2010') | (dataset['Date'
                          (dataset['Date']== '07-09-2012') | (dataset['Date']=
labour_day_sales
```

Out[25]:

```
1042427.2939259257
```

In [26]:

```
thanks_giving_sales =  dataset[(dataset['Date']== '26-11-2010') | (dataset['D
                          (dataset['Date']== '23-11-2012') | (dataset['Date']=
thanks_giving_sales
```

Out[26]:

```
1471273.427777778
```

In [27]:

```
christmas_sales = dataset[(dataset['Date']== '31-12-2010') | (dataset['Date']
                          (dataset['Date']== '28-12-2012') | (dataset['Date']=
christmas_sales
```

Out[27]:

```
898500.4222222222
```

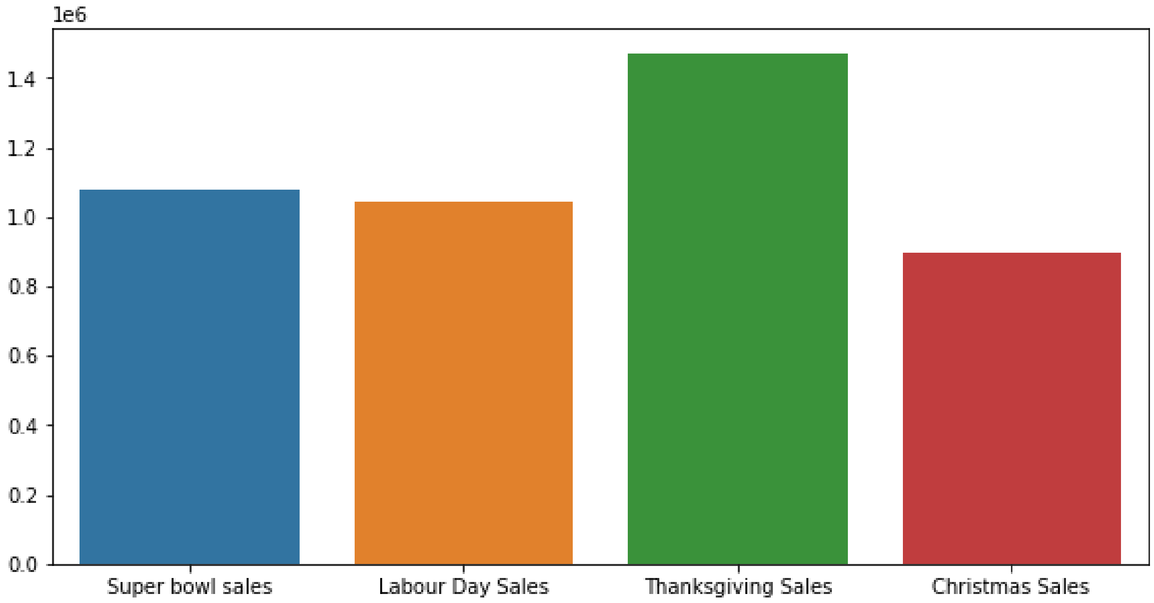
In [28]:

```
print('super_bowl_sale : ',super_bowl_sales)
print('labour_day_sale : ',labour_day_sales)
print('thanks_giving_sale : ',thanks_giving_sales)
print('christmas : ',christmas_sales)
```

```
super_bowl_sale :  1079127.9877037033
labour_day_sale :  1042427.2939259257
thanks_giving_sale :  1471273.427777778
christmas :  898500.4222222222
```

In [29]:

```
plt.figure(figsize=(10,5))
sns.barplot(x=list(['Super bowl sales','Labour Day Sales','Thanksgiving Sales
            y=list([super_bowl_sales,labour_day_sales,thanks_giving_sales,chr
```



In [30]:

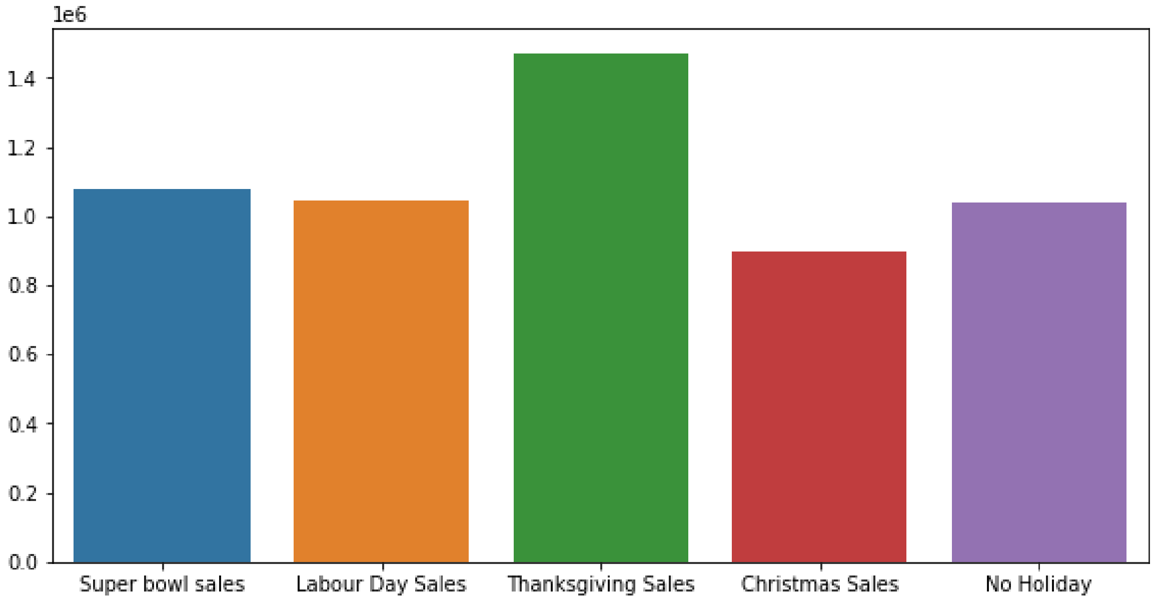
```
no_holiday_sales = dataset[dataset['Holiday_Flag'] == 0]['Weekly_Sales'].sum(
In [31]: print('Sales on days there was no holiday' , no_holiday_sales)
```

```
Sales on days there was no holiday 6231919435.55
```

```
In [32]: no_holiday_sales_mean = dataset[dataset['Holiday_Flag'] == 0]['Weekly_Sales']
         no_holiday_sales_mean
```

```
Out[32]: 1041256.3802088564
```

```
In [33]: plt.figure(figsize=(10,5))
         sns.barplot(x=list(['Super bowl sales','Labour Day Sales','Thanksgiving Sales
                     y=list([super_bowl_sales,labour_day_sales,thanks_giving_sales,chr
```



FROM THE PLOTTED BAR GRAPH IT IS OBSERVED THAT CHRISTMAS HAS THE HIGHEST WEEKLY SALES RATE AS COMPARED TO ANY OTHER HOLIDAY OR NON-HOLIDAY WEEKS.

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

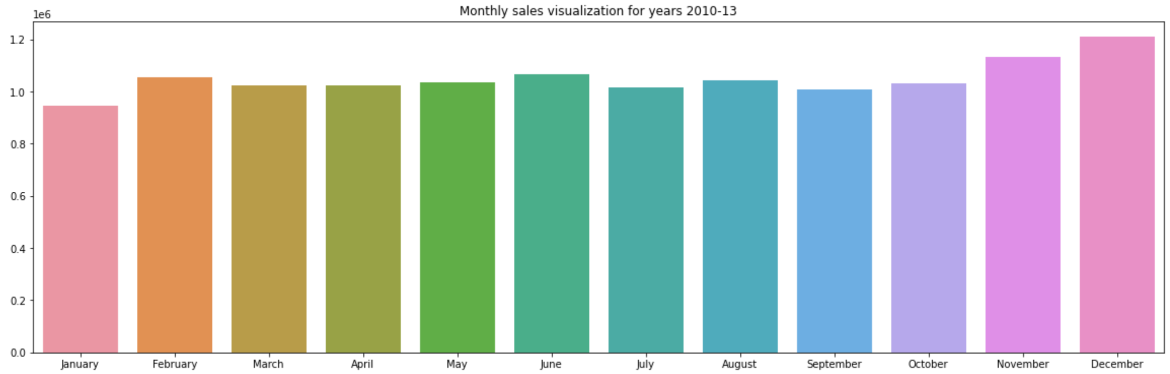
```
In [34]: jan = dataset[dataset['Month'] == 1]['Weekly_Sales'].mean()
         feb = dataset[dataset['Month'] == 2]['Weekly_Sales'].mean()
         mar = dataset[dataset['Month'] == 3]['Weekly_Sales'].mean()
         apr = dataset[dataset['Month'] == 4]['Weekly_Sales'].mean()
         may = dataset[dataset['Month'] == 5]['Weekly_Sales'].mean()
         june = dataset[dataset['Month'] == 6]['Weekly_Sales'].mean()
         july = dataset[dataset['Month'] == 7]['Weekly_Sales'].mean()
         aug = dataset[dataset['Month'] == 8]['Weekly_Sales'].mean()
         sept = dataset[dataset['Month'] == 9]['Weekly_Sales'].mean()
         octo = dataset[dataset['Month'] == 10]['Weekly_Sales'].mean()
         nov = dataset[dataset['Month'] == 11]['Weekly_Sales'].mean()
         dec = dataset[dataset['Month'] == 12]['Weekly_Sales'].mean()
```

In [35]:
```
jan,feb
```

Out[35]:
```
(947613.9223333331, 1054597.3391717167)
```

MONTHLY SALES GRAPH

In [36]:
```
plt.figure(figsize=(20,6))
sns.barplot(x=list(['January','February','March','April','May','June','July',
                             ,'December']), y=list([jan,feb,mar,apr,may,june,july,aug,
                                                  dec])).set(Title ='Monthly sales v
```



In [37]:
```
IT IS VERY CLEAR FROM THE DATA THAT DECEMBER HAS THE HIGHEST SALES OF ALL MON
```

```
  File "<ipython-input-37-d9ca1ea1a38d>", line 1
    IT IS VERY CLEAR FROM THE DATA THAT DECEMBER HAS THE HIGHEST SALES OF A
LL MONTHS ROUND THE YEAR.
       ^
SyntaxError: invalid syntax
```

SEMESTER SALES GRAPH

In [ ]:
```
semester1 = dataset[(dataset['Quarter'] == 1) | (dataset['Quarter'] == 2)]['W
semester2 = dataset[(dataset['Quarter'] == 3) | (dataset['Quarter'] == 4)]['W
```

In [ ]:
```
plt.figure(figsize=(10,5))
sns.barplot(x=list(['Semester 1', 'Semester 2']), y=list([semester1, semester
           ).set(Title='Semester Sales Visualization Year 2010-13');
```

In [ ]:
```
THE TWO SEMESTERS DONOT HAVE A VERY LARGE DIFFERENCE IN SALES, THE 2ND SEMEST
```

In [ ]:
```
dataset
```

In [ ]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [ ]:

```
scale = MinMaxScaler()
```

In [ ]:

```
dataset['Temperature'] = scale.fit_transform(dataset[['Temperature']])
dataset['Fuel_Price'] = scale.fit_transform(dataset[['Fuel_Price']])
dataset['Unemployment'] = scale.fit_transform(dataset[['Unemployment']])
```

In [ ]:

```
dataset
```

In [ ]:

```
dataset = dataset.sort_values('Date', ascending = True)
```

In [ ]:

```
dataset
```

Change dates into days by creating new variable

In [ ]:

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

In [ ]:

```
coder = LabelEncoder()
```

In [ ]:

```
dataset['Date_new'] = coder.fit_transform(dataset['Date'])
```

In [ ]:

```
dataset.head()
```

In [ ]:

```
y = pd.get_dummies(dataset["Store"])
dataset = dataset.drop('Store',axis = 1)
dataset = dataset.join(y)
```

In [ ]:

```
dataset.columns
```

In [ ]:

```
X = dataset.drop(columns=['Weekly_Sales','Date','Year','Month','Quarter','Day
y = dataset['Weekly_Sales']
```

In [ ]:

```
from sklearn.model_selection import train_test_split
```

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 100)
X_train.shape,X_test.shape, y_train.shape, y_test.shape
```

Linear Regression

In [ ]:

```
from sklearn.linear_model import LinearRegression
```

In [ ]:
```
model = LinearRegression()
```

In [ ]:
```
model.fit(X_train, y_train)
```

In [ ]:
```
y_pred = model.predict(X_test)
```

In [ ]:
```
from sklearn.metrics import r2_score, mean_squared_error, accuracy_score
```

In [ ]:
```
y_pred
```

In [ ]:
```
y_test
```

In [ ]:
```
model.score(X_test, y_test)
```

In [ ]:
```
mean_squared_error(y_pred, y_test)
```

In [ ]:
```
r2 = r2_score(y_test, y_pred)
```

In [ ]:
```
accuracy = r2*100
accuracy
```

THE ACCURACY FOR THE MODEL IS 91.03% WHICH IS PRETTY HIGH