Lab - 3

Name : Arul Kumar ARK

Roll No. : 225229103

```
                                        Fuel Amount Prediction using Linear Regression
```

Step : 1

```
Prepare Your Dataset
```

Step : 2

In [2]:
```
import pandas as pd
```

In [3]:
```
fuel=pd.read_csv('fuel_data.csv')
fuel
```

Out[3]:

	drivenKM	fuelAmount
0	390.00	3600.0
1	403.00	3705.0
2	396.50	3471.0
3	383.50	3250.5
4	321.10	3263.7
5	391.30	3445.2
6	386.10	3679.0
7	371.80	3744.5
8	404.30	3809.0
9	392.20	3905.0
10	386.43	3874.0
11	395.20	3910.0
12	381.00	4020.7
13	372.00	3622.0
14	397.00	3450.5
15	407.00	4179.0
16	372.40	3454.2
17	375.60	3883.8
18	399.00	4235.9

In [6]:
```
fuel.head(10)
```

Out[6]:

	drivenKM	fuelAmount
0	390.0	3600.0
1	403.0	3705.0
2	396.5	3471.0
3	383.5	3250.5
4	321.1	3263.7
5	391.3	3445.2
6	386.1	3679.0
7	371.8	3744.5
8	404.3	3809.0
9	392.2	3905.0

In [7]:

```
fuel.shape
```

Out[7]:

```
(19, 2)
```

In [8]:

```
fuel.columns
```

Out[8]:

```
Index(['drivenKM', 'fuelAmount'], dtype='object')
```

In [9]:

```
type(fuel)
```

Out[9]:

```
pandas.core.frame.DataFrame
```

In [10]:

```
fuel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19 entries, 0 to 18
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   drivenKM    19 non-null     float64
 1   fuelAmount  19 non-null     float64
dtypes: float64(2)
memory usage: 432.0 bytes
```

Step : 3

In [11]:

```
fuel.isnull()
```

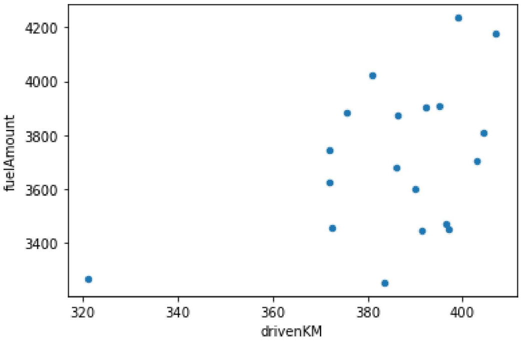
Out[11]:

	drivenKM	fuelAmount
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
5	False	False
6	False	False
7	False	False
8	False	False
9	False	False
10	False	False
11	False	False
12	False	False
13	False	False
14	False	False
15	False	False
16	False	False
17	False	False
18	False	False

Step : 4

In [25]:

```
import matplotlib.pyplot as plt
fuel.plot(kind="scatter",x="drivenKM",y="fuelAmount")
print(plt.show())
```



```
None
```

Step : 5 & Step : 6

In [15]:

```
X = fuel[["drivenKM"]]
```

In [16]:

```
print(X)
type(X)
```

```
    drivenKM
0     390.00
1     403.00
2     396.50
3     383.50
4     321.10
5     391.30
6     386.10
7     371.80
8     404.30
9     392.20
10    386.43
11    395.20
12    381.00
13    372.00
14    397.00
15    407.00
16    372.40
17    375.60
18    399.00
```

Out[16]:

```
pandas.core.frame.DataFrame
```

In [17]:

```
y = fuel.fuelAmount
print(y)
type(y)
```

```
0     3600.0
1     3705.0
2     3471.0
3     3250.5
4     3263.7
5     3445.2
6     3679.0
7     3744.5
8     3809.0
9     3905.0
10    3874.0
11    3910.0
12    4020.7
13    3622.0
14    3450.5
15    4179.0
16    3454.2
17    3883.8
18    4235.9
Name: fuelAmount, dtype: float64
```

Out[17]:

```
pandas.core.series.Series
```

Step : 7

In [19]:

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

In [20]:

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

In [34]:

```
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [35]:

```
X_train.shape
```

Out[35]:

```
(15, 1)
```

In [36]:

```
X_test.shape
```

Out[36]:

```
(4, 1)
```

In [37]:

```
y_train.shape
```

Out[37]:

```
(15,)
```

In [38]:

```
y_test.shape
```

Out[38]:

```
(4,)
```

```
                        Part - I Linear Regression Model
```

Step : 8

In [39]:

```
reg=LinearRegression()
reg.fit(X_train,y_train)
```

Out[39]:

```
LinearRegression()
```

Step : 9

In [40]:

```
pred_800_KM=reg.predict([[800]])
print("Deisel price for 800KM:",pred_800_KM)
```

```
Deisel price for 800KM: [6905.64571567]

C:\Users\arulk\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but Linear
Regression was fitted with feature names
  warnings.warn(
```

Step : 10

In [41]:

```
y_pred=reg.predict(X_test)
y_pred
```

Out[41]:

```
array([3775.81615646, 3785.74000628, 3815.51155575, 3875.05465468])
```

Step : 11

In [42]:

```
import sklearn.metrics as metrics
MSE=metrics.mean_squared_error(y_test,y_pred)
R2=metrics.r2_score(y_test,y_pred)
print("MSE: ",MSE.astype('int'))
```

```
MSE:  46181
```

In [33]:

```
print("R2: ",R2)
```

```
R2:  -0.4409983890088389
```

In [43]:

```
print("coefficient:",reg.coef_)
```

```
coefficient: [7.63373063]
```

```
In [44]: print("Intercept:",reg.intercept_)

Intercept: 798.6612098962887
```

```
Part - II Linear Regression Model With Scaling Using StanderScaler
```

Step : 12 & Step : 13

```
In [50]: from sklearn.preprocessing import StandardScaler
         s=StandardScaler()
         sd_X_train=s.fit_transform(X_train)
         sd_X_train

Out[50]: array([[ 1.0601947 ],
                [-0.5322439 ],
                [ 0.02186483],
                [-0.55221178],
                [ 1.19497791],
                [-0.37250084],
                [ 0.670821  ],
                [ 0.45616627],
                [ 0.79562026],
                [-3.09312478],
                [-0.10293443],
                [-0.56219572],
                [ 0.16812957],
                [ 0.69578085],
                [ 0.15165606]])
```

```
In [51]: sd_X_test=s.transform(X_test)
         sd_X_test

Out[51]: array([[0.34634292],
                [0.41123853],
                [0.60592538],
                [0.99529908]])
```

```
In [52]: reg.fit(sd_X_train,y_train)

Out[52]: LinearRegression()
```

```
In [53]: sd_y_pred=reg.predict(sd_X_test)
         sd_y_pred

Out[53]: array([3775.81615646, 3785.74000628, 3815.51155575, 3875.05465468])
```

Step : 14

```
In [54]: sd_MSE=metrics.mean_squared_error(y_test,sd_y_pred)
         sd_R2=metrics.r2_score(y_test,sd_y_pred)
```

```
In [55]: print("Mean Squared Error: ",sd_MSE)

Mean Squared Error:  46181.36710639172
```

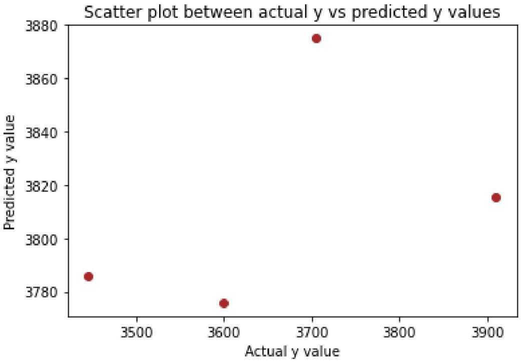
```
In [56]: print("R2 Error: ",sd_R2)

R2 Error:  -0.6180990161577082
```

Step : 15

In [57]:

```
plt.scatter(y_test,y_pred,color='Brown',marker='o')
plt.title("Scatter plot between actual y vs predicted y values")
plt.xlabel('Actual y value')
plt.ylabel('Predicted y value')
plt.show()
```



```
Part - III Linear Regression Model With Scaling Using StanderScaler Using MinMaxScaler and Comparison with
KNeighborsRegressor and SCDRegressor
```

Step : 16

In [29]:

```
from sklearn.preprocessing import MinMaxScaler
mm=MinMaxScaler()
MinMax_X_train=mm.fit_transform(X_train)
MinMax_X_test=mm.transform(X_test)

reg.fit(MinMax_X_train,y_train)
MinMax_y_pred=reg.predict(MinMax_X_test)
print("Predictions of MinMaxScaler:",MinMax_y_pred)

MinMax_MSE=metrics.mean_squared_error(y_test,MinMax_y_pred)
MinMax_R2=metrics.r2_score(y_test,MinMax_y_pred)
print("MinMaxScaler MSE: ",MinMax_MSE)
print("MinMaxScaler R2: ",MinMax_R2)
```

```
Predictions of MinMaxScaler: [3775.81615646 3785.74000628 3815.51155575 3875.05465468]
MinMaxScaler MSE:  46181.3671063917
MinMaxScaler R2:  -0.6180990161577073
```

Step : 17

In [30]:

```
from sklearn.neighbors import KNeighborsRegressor
knr=KNeighborsRegressor()
knr.fit(X_train,y_train)
knr_y_pred=knr.predict(X_test)
print("Predictions of KNeighborsRegressor:",knr_y_pred)
knr_mse=metrics.mean_squared_error(y_test,knr_y_pred)
knr_r2=metrics.r2_score(y_test,knr_y_pred)
print("KNR MSE: ",knr_mse)
print("KNR R2: ",knr_r2)
```

```
Predictions of KNeighborsRegressor: [3635.9   3675.9   3787.28 3829.08]
KNR MSE:  21241.836200000045
KNR R2:  0.2557302563733307
```

Step : 18

In [31]:

```
from sklearn.linear_model import SGDRegressor
sgd=SGDRegressor()
sgd.fit(X_train, y_train)
sgd_y_pred=sgd.predict(X_test)
print("Predictions of SGDRegressor:",sgd_y_pred)
sgd_mse=metrics.mean_squared_error(y_test, sgd_y_pred)
sgd_r2=metrics.r2_score(y_test,sgd_y_pred)
print("SGD MSE:",sgd_mse)
print("SGD R2:",sgd_r2)
```

```
Predictions of SGDRegressor: [-2.21183835e+14 -2.21921118e+14 -2.24132969e+14 -2.28556671e+14]
SGD MSE: 5.016125282148493e+28
SGD R2: -1.7575459308663273e+24
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-packages\sklearn\linear_model\stochastic_gradien
t.py:128: FutureWarning: max_iter and tol parameters have been added in <class 'sklearn.linear_model.stochastic_gradient.SGD
Regressor'> in 0.19. If both are left unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults
to max_iter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

Step : 19

In [32]:

```
data_mse = {'lr_mse':[46181.36710639155],'std_mse':[46181.36710639172],'minmax_mse':[46181.3671063917],'knr_mse':[21241.83620

def best_model(data_mse):
# Calculating the lowest MSE
    mse_min = min(data_mse.values())
# Storing the lowest MSE in result
    result = [key for key in data_mse if data_mse[key] == mse_min]
    Model_name = []
    if result == ['lr_mse']:
        a = 'LinearRegression'
        Model_name.append(a)
    elif result == ['std_mse']:
        b = 'StandardScaler'
        Model_name.append(b)
    elif result == ['minmax_mse']:
        c = 'MinMaxScaler'
        Model_name.append(c)
    elif result == ['knr_mse']:
        d = 'KNeighborsRegressor'
        Model_name.append(d)
    elif result == ['sgd_mse']:
        e = 'SGDRegressor'
        Model_name.append(e)
# Printing the result
    print("The best model with the lowest MSE to be selected is", Model_name)

best_model(data_mse)
```

```
The best model with the lowest MSE to be selected is ['KNeighborsRegressor']
```