Lab3. Fuel Amount Prediction using Linear Regression

Name:P.Asha Belcilda

Rollno:225229104

In [1]:

```
import pandas as pd
```

Step2.Import dataset

In [3]:

```
df=pd.read_csv("fuel_data.csv")
df
```

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 [4]:

```
#head
df.head()
```

Out[4]:

	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

In [5]:

```
#shape
df.shape
```

Out[5]:

```
(19, 2)
```

In [6]:

```
#columns
df.columns
```

Out[6]:

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

In [7]:

```
#type
type(df)
```

Out[7]:

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

In [8]:

```
#info
df.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
```

Step3.Preprocessing

In [97]:

```
df.isnull()
```

Out[97]:

	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 Visualize Relationships.

In [ ]:

```
import numpy as np
import seaborn as sns
```

In [ ]:

```
sns.relplot(data=df,x='drivenKM',y='fuelAmount')
```

Step5.Prepare X matrix and y matrix

In [48]:

```
x=df[['drivenKM']]
y=df[['fuelAmount']]
```

In [49]:
```
x
```

Out[49]:

	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

In [50]:
```
y
```

Out[50]:

	fuelAmount
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

Step6.Examine X and y.

In [51]:
```
type(x)
```

Out[51]:
```
pandas.core.frame.DataFrame
```

In [52]:
```
type(y)
```

Out[52]:
```
pandas.core.frame.DataFrame
```

Step7.Split dataset

In [53]:
```
from sklearn.model_selection import train_test_split
```

In [54]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

In [55]:
```
x_train
```

Out[55]:
```
    drivenKM
 8    404.30
16    372.40
 3    383.50
13    372.00
15    407.00
17    375.60
 2    396.50
 9    392.20
18    399.00
 4    321.10
12    381.00
 7    371.80
10    386.43
14    397.00
 6    386.10
```

In [56]:
```
x_train.shape
```

Out[56]:
```
(15, 1)
```

In [57]:
```
y_train
```

Out[57]:

	fuelAmount
8	3809.0
16	3454.2
3	3250.5
13	3622.0
15	4179.0
17	3883.8
2	3471.0
9	3905.0
18	4235.9
4	3263.7
12	4020.7
7	3744.5
10	3874.0
14	3450.5
6	3679.0

In [58]:
```
y_train.shape
```

Out[58]:
```
(15, 1)
```

In [59]:
```
x_test
```

Out[59]:

	drivenKM
0	390.0
5	391.3
11	395.2
1	403.0

x_test.shape

In [60]:
```
y_test
```

Out[60]:

	fuelAmount
0	3600.0
5	3445.2
11	3910.0
1	3705.0

In [61]:
```
y_test.shape
```

Out[61]:
```
(4, 1)
```

Part-I Linear Regression Baseline Model

Step8.Build Model

In [62]:
```
from sklearn.linear_model import LinearRegression
im=LinearRegression()
im.fit(x_train,y_train)
```

Out[62]:
```
LinearRegression()
```

Step9.Predict price for 800 KM

In [63]:
```

price=[[800]]
im.predict(price)
```

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

Out[63]:
```
array([[6905.64571567]])
```

Step10.Predict on entire dataset.

In [64]:
```
y_data=im.predict(x_test)
y_data
```

Out[64]:
```
array([[3775.81615646],
       [3785.74000628],
       [3815.51155575],
       [3875.05465468]])
```

Step11.Print Mean Squared Error and R2 Error

In [65]:
```
from sklearn.metrics import mean_squared_error
```

In [66]:
```
mean_squared_error(y_test,y_data)
```

Out[66]:
```
46181.36710639155
```

In [67]:
```
im.coef_
```

Out[67]:
```
array([[7.63373063]])
```

In [68]:
```
im.intercept_
```

Out[68]:
```
array([798.6612099])
```

Part - II Linear Regression With Scaling Using StandardScaler

Step12.Normalize X_train and X_test values.

In [69]:
```
from sklearn.preprocessing import StandardScaler
```

In [70]:
```
scaler=StandardScaler()
```

In [71]:
```
norm_x_train=scaler.fit_transform(x_train)
norm_x_train
```

Out[71]:
```
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 [72]:
```
norm_x_train.shape
```

Out[72]:
```
(15, 1)
```

In [73]:
```
norm_x_test=scaler.transform(x_test)
norm_x_test
```

Out[73]:
```
array([[0.34634292],
       [0.41123853],
       [0.60592538],
       [0.99529908]])
```

Step13.Build LR model

In [74]:
```
from sklearn.linear_model import LinearRegression
```

In [76]:
```
LR_model=LinearRegression()
LR_model.fit(scaled_x_train,y_train)
y_predict=LR_model.predict(norm_x_test)
y_predict
```

Out[76]:
```
array([[3775.81615646],
       [3785.74000628],
       [3815.51155575],
       [3875.05465468]])
```

In [77]:
```
from sklearn.metrics import mean_squared_error
```

In [78]:
```
LR=mean_squared_error(y_test,y_predict)
LR
```

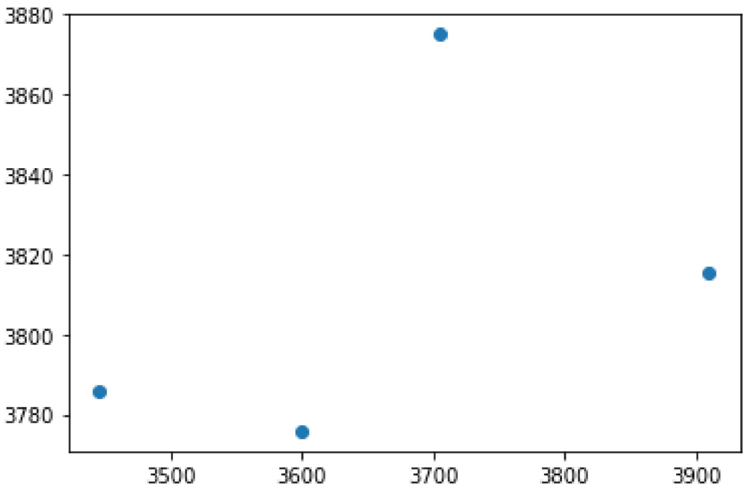
Out[78]:
```
46181.36710639172
```

Step15.Plot scatter plot.

In [80]:
```
import matplotlib.pyplot as plt
```

In [81]:
```
plt.scatter(y_test,y_predict)
```

Out[81]:
```
<matplotlib.collections.PathCollection at 0x15a55349190>
```



Part-III.Linear Regression with Scaling using MinMaxScaler and Comparison with KNeighborsRegressor and SGDRegressor.

Step16.Repeat with MinmaxScaler.

In [82]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [84]:
```
mms=MinMaxScaler()
mms.fit(scaled_x_train,y_train)
mms_pe=mms.transform(norm_x_test)
mms_pe
```

Out[84]:
```
array([[0.80209546],
       [0.81722934],
       [0.86263097],
       [0.95343423]])
```

```
In [85]: MMS=mean_squared_error(y_test,mms_pe)
         MMS

Out[85]: 13454828.539572451
```

Step17.Compare KNN Regressor

```
In [86]: from sklearn.neighbors import KNeighborsRegressor
```

```
In [88]: neigh = KNeighborsRegressor()
         neigh.fit(scaled_x_train,y_train)
         re=neigh.predict(norm_x_test)
```

```
In [89]: KN=mean_squared_error(y_test,re)
         KN

Out[89]: 21241.836200000045
```

Step18.Compare SGD Regressor.

```
In [90]: from sklearn.linear_model import SGDRegressor
```

```
In [92]: sgd=SGDRegressor()
         sgd.fit(scaled_x_train,y_train)
         sgd_pre=sgd.predict(norm_x_test)
         sgd_pre

         C:\Users\ashac\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: Dat
         aConversionWarning: A column-vector y was passed when a 1d array was expected.
         Please change the shape of y to (n_samples, ), for example using ravel().
           y = column_or_1d(y, warn=True)

Out[92]: array([3775.25666053, 3785.17567077, 3814.93270149, 3874.44676293])
```

```
In [93]: SGDR1=mean_squared_error(y_test,sgd_pre)
         SGDR1

Out[93]: 46012.087622021594
```

Step19.Select best model.

```
In [94]: table=pd.DataFrame([SGDR1,KN,LR])
```

```
In [95]: table['Algorithm']=['SGDR','KN',"LR"]
In [96]: table.set_index('Algorithm')
```

```
Out[96]:
                        0
           Algorithm
               SGDR  46012.087622
                 KN  21241.836200
                 LR  46181.367106
```