Fuel Amount Prediction using Linear Regression

Name : MURALI KUMAR R

Roll no : 225229120

Import dataset:

In [72]:
```
#step2:

import pandas as pd
```

In [73]:
```
data=pd.read_csv('fuel_data.csv')
data
```

Out[73]:

	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 [3]:
```
data.head()
```

Out[3]:

	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 [74]:
```
data.shape
```

Out[74]:
```
(19, 2)
```

In [77]:
```
data.shape[0]
```

Out[77]:
```
19
```

In [6]:
```
type(data)
```

Out[6]:
```
pandas.core.frame.DataFrame
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19 entries, 0 to 18
Data columns (total 2 columns):
drivenKM      19 non-null float64
fuelAmount    19 non-null float64
dtypes: float64(2)
memory usage: 384.0 bytes
```

Proprocessing

In [8]:

```
#step3:

data.isnull()
```

Out[8]:

	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

Visualize Relationships

In [23]:

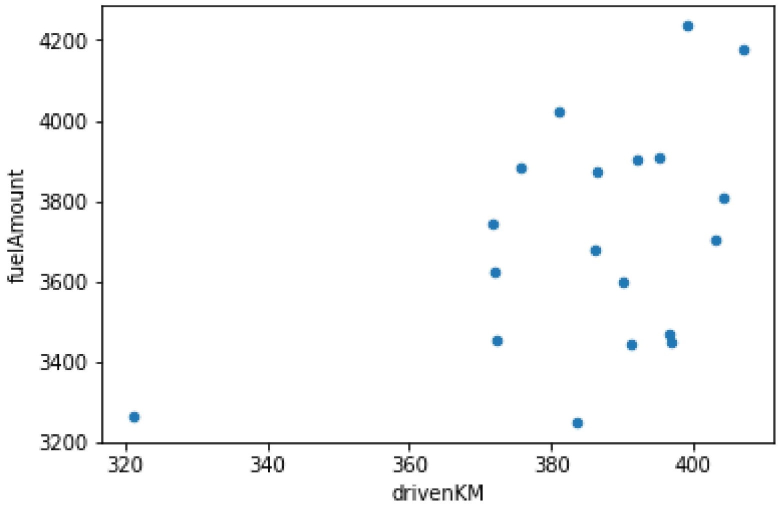
```
#step4:

import matplotlib.pyplot as plt

df=pd.read_csv("fuel_data.csv")
df.plot(kind='scatter',x='drivenKM', y='fuelAmount')
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x27ed34461d0>
```



Prepare x matrix and y vector

In [87]:

```
#step5:

X=pd.DataFrame(df)
cols=[1]
X=X[X.columns [cols]]
```

In [88]:

```
y=df['fuelAmount'].values
```

Examine X and y

In [89]:

```
#step6:

X
```

Out[89]:

	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

In [90]:

```
type(X)
```

Out[90]:

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

In [91]:

```
print(y)
```

```
[3600.  3705.  3471.  3250.5 3263.7 3445.2 3679.  3744.5 3809.  3905.
 3874.  3910.  4020.7 3622.  3450.5 4179.  3454.2 3883.8 4235.9]
```

In [92]:

```
type(y)
```

Out[92]:

```
numpy.ndarray
```

Split dataset

```
In [93]: #step7:

         from sklearn.linear_model import LinearRegression
         from sklearn.model_selection import train_test_split

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

         X_train.shape

Out[93]: (15, 1)
```

```
In [94]: X_test.shape

Out[94]: (4, 1)
```

```
In [95]: y_train.shape

Out[95]: (15,)
```

```
In [96]: y_test.shape

Out[96]: (4,)
```

Part - I : Linear Regression Baseline Model

Build Model

```
In [97]: #step8:

         model = LinearRegression()
```

```
In [98]: model.fit(X_train, y_train)

Out[98]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
In [104]: pred_800_KM=model.predict([[800]])
          print("Deisel price for 800KM:",pred_800_KM[0])

          Deisel price for 800KM: 799.9999999999993
```

Predict on entire dataset

```
In [36]: #step10:

         y_pred=reg.predict(X_test)
```

```
In [41]: y_pred

Out[41]: array([3471. , 4179. , 3263.7, 3454.2])
```

Print Mean Squared Error and R2 Error

In [42]:

```
#step11:

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)
print("R2: ",r2)
print("\n")
print("Model parameters:")
print("coefficient:",reg.coef_)
print("Intercept:",reg.intercept_)
```

```
MSE:  5.169878828456423e-26
R2:  1.0


Model parameters:
coefficient: [-2.41215635e-16  1.00000000e+00]
Intercept: -3.183231456205249e-12
```

Part -II : Linear Reagression with Scaling using StandardScaler

Normalize x_train and x_test Values

In [46]:

```
#step12:

from sklearn.preprocessing import StandardScaler

data=StandardScaler()
data_X_train=data.fit_transform(X_train)
data_X_train
```

Out[46]:

```
array([[-0.24458719, -0.26255795],
       [ 0.14285799, -0.59002998],
       [ 0.36141682,  0.67426076],
       [ 0.83827243, -1.2097397 ],
       [ 0.27200639, -1.23170934],
       [-1.64535055, -0.49883524],
       [ 1.56349034,  0.27632007],
       [-0.21180337,  0.54575907],
       [-1.66521953,  0.00895367],
       [ 0.65945157,  0.69498683],
       [-0.75124628,  1.15386219],
       [ 1.43434194, -0.15478235],
       [ 1.03696226,  2.04591257],
       [-0.50288398, -2.0387828 ],
       [-1.28770884,  0.58638219]])
```

In [47]:
```
data_X_test=data.transform(X_test)
data_X_test
```

Out[47]:
```
array([[ 0.78859997, -1.12476278],
       [ 1.83172162,  1.81004981],
       [-6.70200692, -1.98406595],
       [-1.60561258, -1.1944024 ]])
```

Build LR Model

In [48]:
```
#step13:

from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(data_X_train,y_train)
```

Out[48]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [50]:
```
data_y_pred=model.predict(data_X_test)
data_y_pred
```

Out[50]:
```
array([3471. , 4179. , 3263.7, 3454.2])
```

Print MSE and R2 Error

In [54]:
```
#step14:

data_mse=metrics.mean_squared_error(y_test,data_y_pred)
data_r2=metrics.r2_score(y_test,data_y_pred)
print("SS_MSE: ",data_mse)
print("SS_R2: ",data_r2)
```

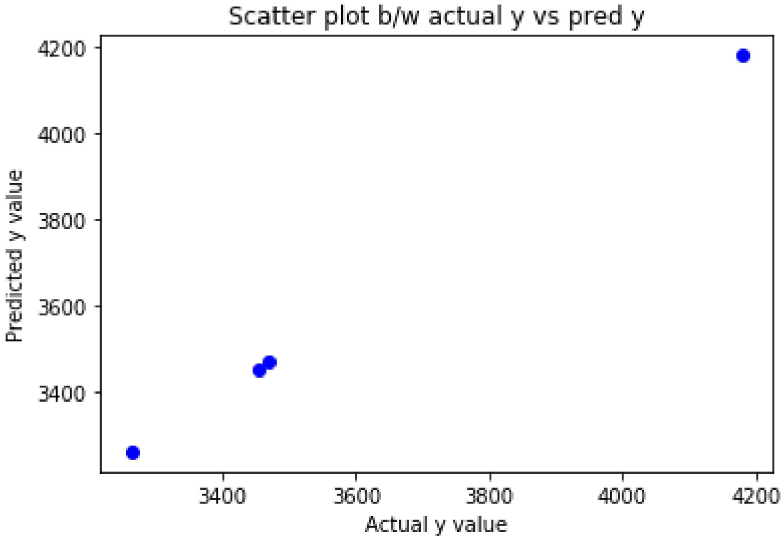
```
SS_MSE:  1.550963648536927e-25
SS_R2:  1.0
```

Plot scatter plot

In [55]:

```
#step15:

import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter(y_test,y_pred,color='Blue',marker='o')
plt.title("Scatter plot b/w actual y vs pred y")
plt.xlabel('Actual y value')
plt.ylabel('Predicted y value')
plt.show()
```



Part - III : Linear Regression with Scalling using MinmaxScaler and Comparison with KNeighborsRegreszor and SGDRegressor

Repeat with MinmaxScaler

In [56]:

```
#step16:

from sklearn.preprocessing import MinMaxScaler
mm=MinMaxScaler()
mm_X_train=mm.fit_transform(X_train)
mm_X_test=mm.transform(X_test)
mm_lr=LinearRegression()
mm_lr.fit(mm_X_train,y_train)
mm_y_pred=mm_lr.predict(mm_X_test)
print("Predictions of scaled data using MinMaxScaler:",mm_y_pred)

mm_mse=metrics.mean_squared_error(y_test,mm_y_pred)
mm_r2=metrics.r2_score(y_test,mm_y_pred)
print("MM_MSE: ",mm_mse)
print("MM_R2: ",mm_r2)
```

```
Predictions of scaled data using MinMaxScaler: [3471.  4179.  3263.7 3454.2]
MM_MSE:  3.618915179919496e-25
MM_R2:  1.0
```

Compare KNN Regressor

In [57]:

```
#step17:

from sklearn.neighbors import KNeighborsRegressor
knr=KNeighborsRegressor()
knr.fit(X_train,y_train)
knr_y_pred=knr.predict(X_test)
print("Predictions of scaled data using 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 scaled data using KNeighborsRegressor: [3559.34 3991.08 3473.64
3473.64]
KNR_MSE:  21892.649800000112
KNR_R2:  0.819806065470086
```

Compare SGD Regressor

In [58]:

```
#step 18:

from sklearn.linear_model import SGDRegressor
sgd=SGDRegressor()
sgd.fit(X_train, y_train)
sgd_y_pred=sgd.predict(X_test)
print("Predictions of scaled data using 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 scaled data using SGDRegressor: [-3.65452966e+16 -4.38218219e+16
 -3.42319994e+16 -3.63123660e+16]
SGD_MSE: 1.4365821215542595e+33
SGD_R2: -1.1824214388072463e+28

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-pac
kages\sklearn\linear_model\stochastic_gradient.py:128: FutureWarning: max_iter
and tol parameters have been added in <class 'sklearn.linear_model.stochastic_g
radient.SGDRegressor'> in 0.19. If both are left unset, they default to max_ite
r=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)
```

Select best model

In [64]:

```
#step19:

data_mse = {'lr_mse':[46181.36710639157],
'ss_mse':[46181.36710639172],
'mm_mse':[46181.36710639165],
'knr_mse':[21241.836200000045],
'sgd_mse':[1.1221718443614637e+29]}

def best_model(data_mse):

    mse_min = min(data_mse.values())

    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 == ['ss_mse']:
        b = 'StandardScaler'
        Model_name.append(b)
    elif result == ['mm_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)

    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']
```

In [ ]: