Lab report 8

Fall 2021

CSE422L Data Analytics Lab

Submitted by: **Ayaz Mehmood**

Registration No.:**18PWCSE1652**

Section: **A**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Student Signature: _____

Submitted to:

Engr. Mian Ibad Ali Shah

Last date of Submission:

16 February 2022

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

TASKS

Build a Multiple Linear Regression model with a good accuracy for the dataset. The target feature should be Selling Price. You can use Statsmodel or Sklearn

Reading a csv file

In [1]:
```
import pandas as pd
from sklearn.linear_model import LinearRegression as LR
```

In [2]:
```
data =  pd.read_csv('Car selling dataset.csv')
data

```

Out[2]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4 kmpl	1248 CC	74 bhp	190Nm@ 2000rpm	5.0
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14 kmpl	1498 CC	103.52 bhp	250Nm@ 1500-2500rpm	5.0
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC	78 bhp	12.7@ 2,700(kgm@ rpm)	5.0
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0 kmpl	1396 CC	90 bhp	22.4 kgm at 1750-2750rpm	5.0
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC	88.2 bhp	11.5@ 4,500(kgm@ rpm)	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8123	Hyundai i20 Magna	2013	320000	110000	Petrol	Individual	Manual	First Owner	18.5 kmpl	1197 CC	82.85 bhp	113.7Nm@ 4000rpm	5.0
8124	Hyundai Verna CRDi SX	2007	135000	119000	Diesel	Individual	Manual	Fourth & Above Owner	16.8 kmpl	1493 CC	110 bhp	24@ 1,900-2,750(kgm@ rpm)	5.0
8125	Maruti Swift Dzire ZDi	2009	382000	120000	Diesel	Individual	Manual	First Owner	19.3 kmpl	1248 CC	73.9 bhp	190Nm@ 2000rpm	5.0
8126	Tata Indigo CR4	2013	290000	25000	Diesel	Individual	Manual	First Owner	23.57 kmpl	1396 CC	70 bhp	140Nm@ 1800-3000rpm	5.0
8127	Tata Indigo CR4	2013	290000	25000	Diesel	Individual	Manual	First Owner	23.57 kmpl	1396 CC	70 bhp	140Nm@ 1800-3000rpm	5.0

Getting Unique values

In [3]:
```
data.fuel.unique()
```

Out[3]:
```
array(['Diesel', 'Petrol', 'LPG', 'CNG'], dtype=object)
```

In [4]:
```

df = data.copy(deep=True)
# Now mapping the categorical values of fuel to numerical
# treat_fuel=data.fuel.map({'Petrol':0,
#                           'Diesel':1,'LPG':2,'CNG':3})
# treat_fuel
df.fuel.replace({'Petrol':0,
                 'Diesel':1,'LPG':2,'CNG':3},inplace=True)
df
```

Changing unique categorical values to integer

In [13]:
```
data.seller_type.unique()
```

Out[13]:
```
array(['Individual', 'Dealer', 'Trustmark Dealer'], dtype=object)
```

In [14]:
```
df['seller_type'].replace({'Individual':0,
                           'Dealer':1,'Trustmark Dealer':2},inplace=True)
df

```

Out[14]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	0	0	Manual	First Owner	23.4 kmpl	1248 CC	74 bhp	190Nm@ 2000rpm	5.0
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	0	0	Manual	Second Owner	21.14 kmpl	1498 CC	103.52 bhp	250Nm@ 1500-2500rpm	5.0

In [16]:
```
df['transmission'].replace({'Manual':0,
                            'Automatic':1},inplace=True)

```

In [17]:
```
data.owner.unique()
```

Out[17]:
```
array(['First Owner', 'Second Owner', 'Third Owner',
       'Fourth & Above Owner', 'Test Drive Car'], dtype=object)
```

In [18]:
```
df['owner'].replace({'First Owner':0,
                     'Second Owner':1,'Third Owner':2,
                    'Fourth & Above Owner':3,
                    'Test Drive Car':4
                    },inplace=True)

```

In [24]:
```
df
```

Out[24]:

In [5]:
```
# seller_type
data.seller_type.unique()
```

Out[5]:
```
array(['Individual', 'Dealer', 'Trustmark Dealer'], dtype=object)
```

In [6]:
```
# treat_seller=data.seller_type.map({'Individual':0,
#                            'Dealer':1,'Trustmark Dealer':2})
# treat_seller.unique()

df.seller_type.replace({'Individual':0,
                        'Dealer':1,'Trustmark Dealer':2},inplace=True)
df

```

Out[6]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	1	0	Manual	First Owner	23.4 kmpl	1248 CC	74 bhp	190Nm@ 2000rpm	5.0
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	1	0	Manual	Second Owner	21.14 kmpl	1498 CC	103.52 bhp	250Nm@ 1500-2500rpm	5.0
2	Honda City 2017-2020 EXi	2006	158000	140000	0	0	Manual	Third Owner	17.7 kmpl	1497 CC	78 bhp	12.7@ 2,700(kgm@ rpm)	5.0
3	Hyundai i20 Sportz Diesel	2010	225000	127000	1	0	Manual	First Owner	23.0 kmpl	1396 CC	90 bhp	22.4 kgm at 1750-2750rpm	5.0
4	Maruti Swift VXI BSIII	2007	130000	120000	0	0	Manual	First Owner	16.1 kmpl	1298 CC	88.2 bhp	11.5@ 4,500(kgm@ rpm)	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...

8128 rows × 13 columns

In [7]:
```
data.transmission.unique()
```

Out[7]:
```
array(['Manual', 'Automatic'], dtype=object)
```

In [8]:
```
df.transmission.replace({'Manual':0,
                         'Automatic':1},inplace=True)

```

In [9]:
```
data.owner.unique()
```

Out[9]:
```
array(['First Owner', 'Second Owner', 'Third Owner',
       'Fourth & Above Owner', 'Test Drive Car'], dtype=object)
```

In [10]:
```
df.owner.replace({'First Owner':0,
                          'Second Owner':1,'Third Owner':2,
                        'Fourth & Above Owner':3,
                        'Test Drive Car':4
                        },inplace=True)

```

In [11]:
```
df
```

Dropping the unit:

In [78]:
```
unit = "kmpl"
# Remove Units from Value List
# Using replace() + strip() + list comprehension
df['mileage']= [sub.replace(unit, "").strip() for sub in df['mileage']]
```

In [79]:
```
unit = "km/kg"
# Remove Units from Value List
# Using replace() + strip() + list comprehension
df['mileage']= [sub.replace(unit, "").strip() for sub in df['mileage']]
```

In [95]:
```
df['mileage']
```

Out[95]:
```
0       23.40
1       21.14
2       17.70
3       23.00
4       16.10
        ...
8123    18.50
8124    16.80
8125    19.30
8126    23.57
```

In [101]:
```
hp = []
for i in df['max_power']:
    if str(i).endswith('bhp'):
        i = i[:-5]
        i = str(i)
        hp.append(i)
    else:
        hp.append(str(i))

```

In [102]:
```
df['max_power'] = hp
```

In [103]:
```
df['max_power']
```

Out[103]:
```
0           7
1       103.5
2           7
3           9
4        88.
        ...
8123     82.8
8124       11
8125      73.
8126        7
8127        7
Name: max_power, Length: 8128, dtype: object
```

In [121]:
```
without_cc = []
for i in df.engine:
    if str(i).endswith('CC'):
        i = i[:-3]
        i = float(i)
        without_cc.append(float(i))
    else:
        without_cc.append(i)
```

In [122]:
```
df['engine']=without_cc
```

In [123]:
```
mean_value=df['engine'].mean()
df['engine'].fillna(mean_value,inplace=True)
```

In [124]:
```
df.seller_type.isnull().values.any()
```

Out[124]: False

In [38]:
```
y=df.selling_price

```

In [39]:
```
df.columns
```

Out[39]:
```
Index(['name', 'year', 'selling_price', 'km_driven', 'fuel', 'seller_type',
       'transmission', 'owner', 'mileage', 'engine', 'max_power', 'torque',
       'seats'],
      dtype='object')
```

In [40]:
```
features = ['year','km_driven','fuel','seller_type','transmission','owner','mileage','engine',
            'max_power','seats']
X=df[features]
```

In [41]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression as MLR
```

In [42]:
```
# Splitting the data
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.4)
```

In [43]:
```
from sklearn import linear_model
```

Model Training and Testing:

In [44]:
```


# training model
mlr=linear_model.LinearRegression()
model=mlr.fit(X_train,y_train)

```

In [45]:
```
predict = model.predict(X_test)
predict
```

Out[45]:
```
array([835258.44407876, 543498.90412374, 640602.17893738, ...,
       471018.9863454 , 610181.53673473, 633946.54680522])
```

In [46]:
```
from sklearn.metrics import r2_score
testing_accuracy= r2_score(y_test, predict)
print('Testing Accuracy: ',testing_accuracy*100, '%')
```

```
Testing Accuracy:  55.83564449363476 %
```