Name : Krish Sukhani

Batch : D

Roll No : 59

TE IT

DWM Linear Regression

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
from google.colab import drive
drive.mount("/content/gdrive")
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive
```

```
cars = pd.read_csv('/content/gdrive/My Drive/datasets/car data.csv',encoding= 'unicode_escape')
```

```
cars.head()
```

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

```
cars.shape
```

```
(301, 9)
```

```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Car_Name       301 non-null    object
 1   Year           301 non-null    int64
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
```

```
 4   Kms_Driven     301 non-null    int64
 5   Fuel_Type      301 non-null    object
 6   Seller_Type    301 non-null    object
 7   Transmission   301 non-null    object
 8   Owner          301 non-null    int64
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

```
cars.describe()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.644115	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

```
cars.isna().sum()
```

```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

Converted Year of purchase to the current age of the car

```
cars['Age'] = 2021 - cars['Year']
cars.drop('Year',axis=1,inplace = True)
```

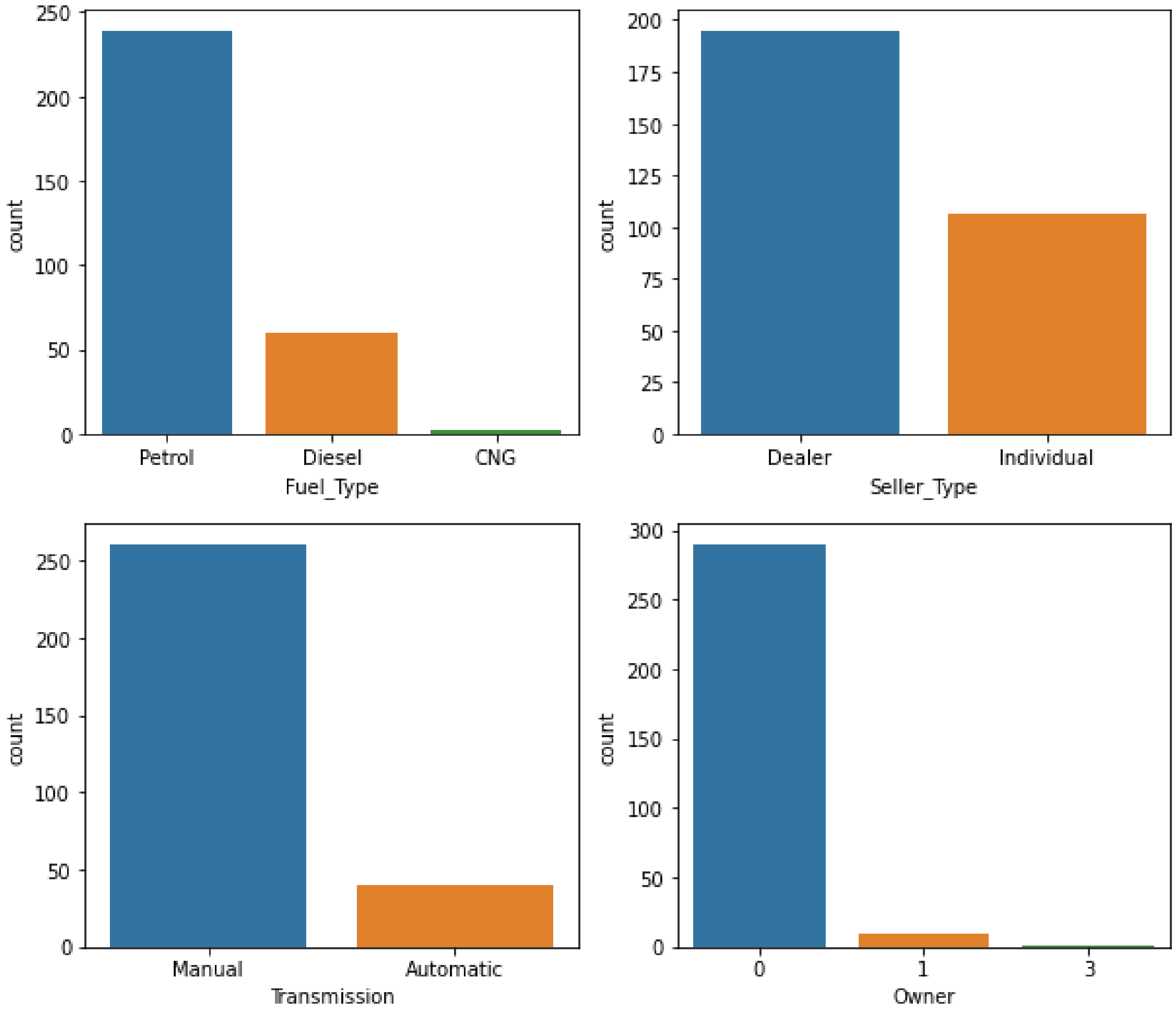
```
cars.head()
```

	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Age
0	ritz	3.35	5.59	27000	Petrol	Dealer	Manual	0	7
1	sx4	4.75	9.54	43000	Diesel	Dealer	Manual	0	8
2	ciaz	7.25	9.85	6900	Petrol	Dealer	Manual	0	4
3	wagon r	2.85	4.15	5200	Petrol	Dealer	Manual	0	10
4	swift	4.60	6.87	42450	Diesel	Dealer	Manual	0	7

```
cat_cols = ['Fuel_Type','Seller_Type','Transmission','Owner']
i=0
while i < 4:
    fig = plt.figure(figsize=[10,4])
    plt.subplot(1,2,1)
    sns.countplot(x=cat_cols[i], data=cars)
    i += 1
```

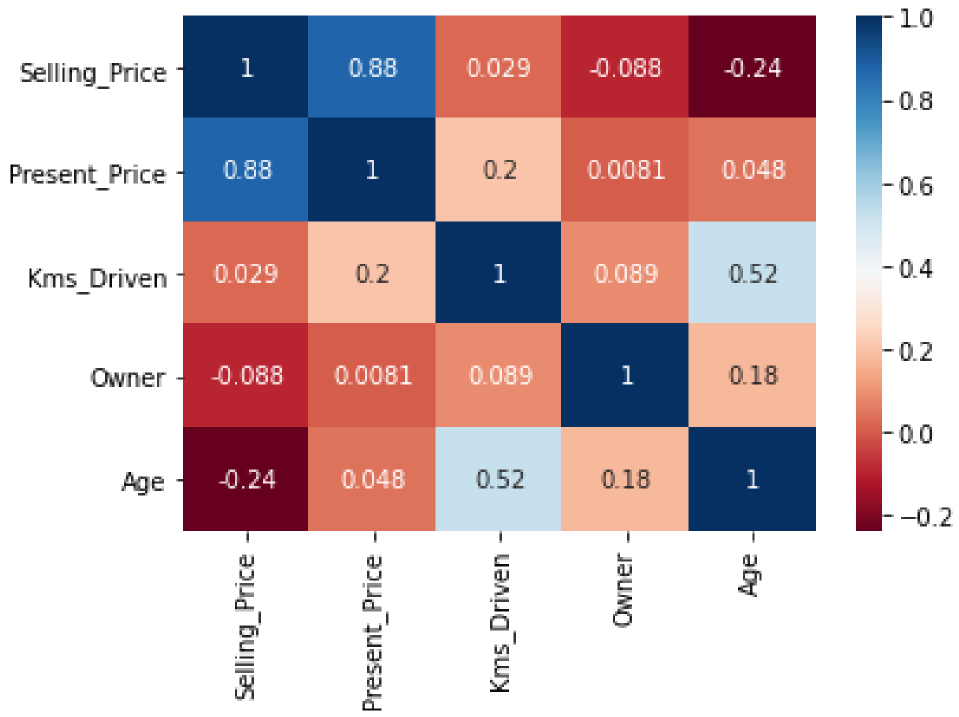
```
    plt.subplot(1,2,2)
    sns.countplot(x=cat_cols[i], data=cars)
    i += 1

    plt.show()
```



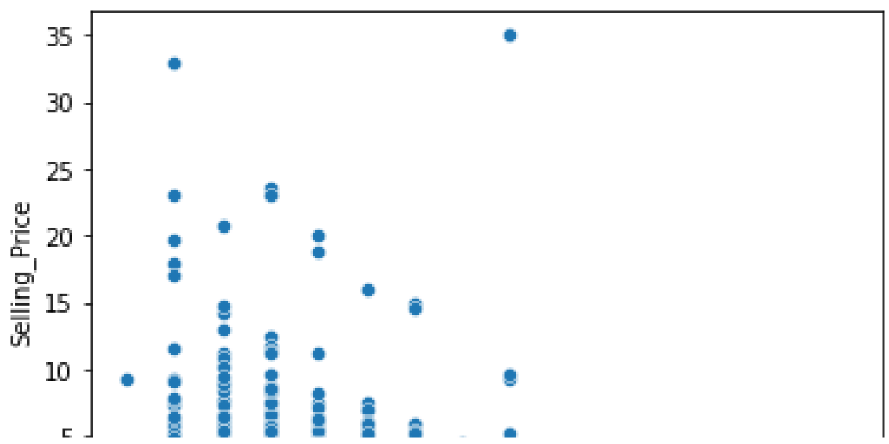
Plotted the count of different attributes

```
sns.heatmap(cars.corr(), annot=True, cmap="RdBu")
plt.show()
```




```
sns.scatterplot(y="Selling_Price", x="Age",data=cars)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f25675776d0>



With this we can infer that as the age of the car increases the selling price goes down

Linear Regression

```
cars1 = pd.DataFrame(zip(cars.Present_Price, cars.Selling_Price), columns=['Present_Price','Selling_Price'])
```

```
cars1
```

	Present_Price	Selling_Price
0	5.59	3.35
1	9.54	4.75
2	9.85	7.25
3	4.15	2.85
4	6.87	4.60
...	...	...
296	11.60	9.50
297	5.90	4.00
298	11.00	3.35
299	12.50	11.50
300	5.90	5.30

301 rows × 2 columns

```
y = cars1['Selling_Price']
X = cars1.drop('Selling_Price',axis=1)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
print("x train: ",X_train.shape)
print("x test: ",X_test.shape)
print("y train: ",y_train.shape)
print("y test: ",y_test.shape)
```

```
x train:  (240, 1)
x test:  (61, 1)
y train:  (240,)
y test:  (61,)
```

```
lr = LinearRegression()
```

```
lr.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

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

```
y_test_col = list()
for i in y_test:
  y_test_col.append(i)
```

```
cars2 = pd.DataFrame(data = (zip(y_test,y_pred)),columns=['Actual','Predicted'])
```

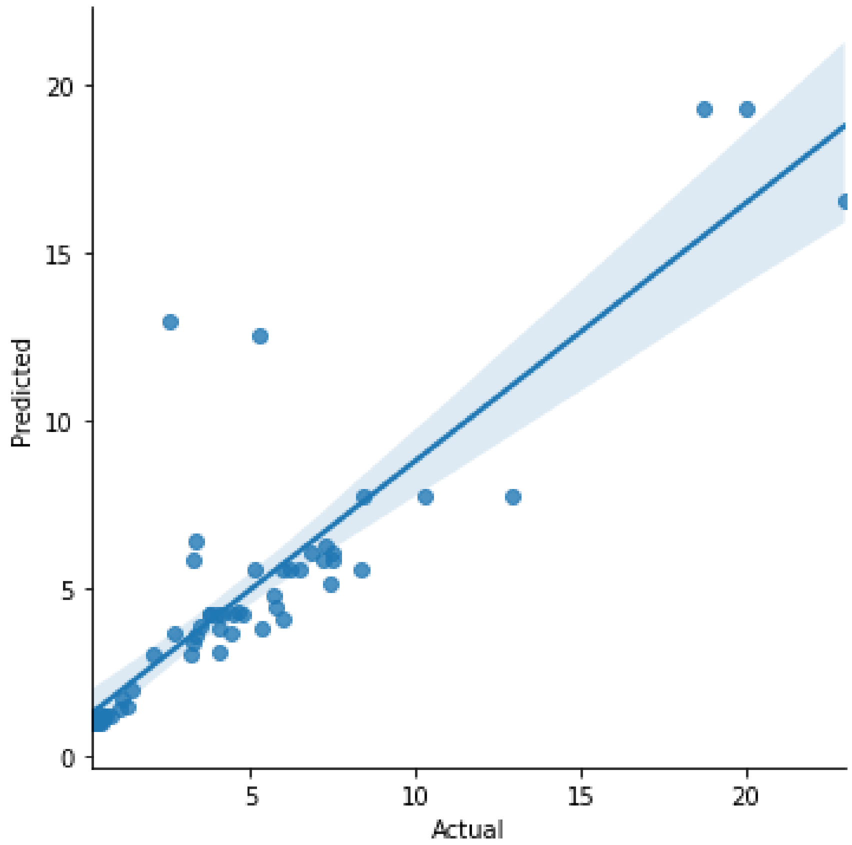
```
cars2
```

	Actual	Predicted
0	7.40	5.129972
1	4.00	3.117986
2	0.50	1.171002
3	3.15	3.030284
4	1.25	1.518715
...	...	...
56	18.75	19.296418
57	0.50	1.198860
58	6.45	5.594276
59	5.65	4.820436
60	0.25	1.013139

61 rows × 2 columns

```
sns.lmplot(x='Actual',y='Predicted',data=cars2)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f2567494750>
```



Plotted the linear regression line and also the data points to check how accurately does it

```
'Mean absolute error = ' + str(mean_absolute_error(y_test, y_pred))
'Mean squared error = ' + str(mean_squared_error(y_test, y_pred))
'R2 score = ' + str(r2_score(y_test, y_pred))
```

```
'R2 score = 0.7734861900562625'
```

Computed the Mean absolute error, Mean Squared error and the R2 Score

```
accuracy = lr.score(X_test,y_test)
print(accuracy*100,'%')
```

```
77.34861900562625 %
```

This is the Accuracy of the model

```
Price = lr.predict([[5]])[0]
```

```
"The selling price of your car will be " + str(Price) + " Lacs"
```

```
'The selling price of your car will be 3.3243433197261587 Lacs'
```

Predicted the selling price of another car by providing the attributes

Multi Linear Regression

```
cars.drop(labels='Car_Name',axis= 1, inplace = True)
```

```
cars = pd.get_dummies(data = cars,drop_first=True)
```

```
cars.head()
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_I
0	3.35	5.59	27000	0	7	0	1	
1	4.75	9.54	43000	0	8	1	0	
2	7.25	9.85	6900	0	4	0	1	
3	2.85	4.15	5200	0	10	0	1	
4	4.60	6.87	42450	0	7	1	0	

```
# Separating target variable and its features
y = cars['Selling_Price']
X = cars.drop('Selling_Price',axis=1)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
print("x train: ",X_train.shape)
print("x test: ",X_test.shape)
print("y train: ",y_train.shape)
print("y test: ",y_test.shape)
```

```
x train:  (240, 8)
x test:  (61, 8)
y train:  (240,)
y test:  (61,)
```

```
lr = LinearRegression()
```

```
lr.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

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

```
y_test_col = list()
for i in y_test:
  y_test_col.append(i)
```

```
cars3 = pd.DataFrame(data = (zip(y_test,y_pred)),columns=['Actual','Predicted'])
```

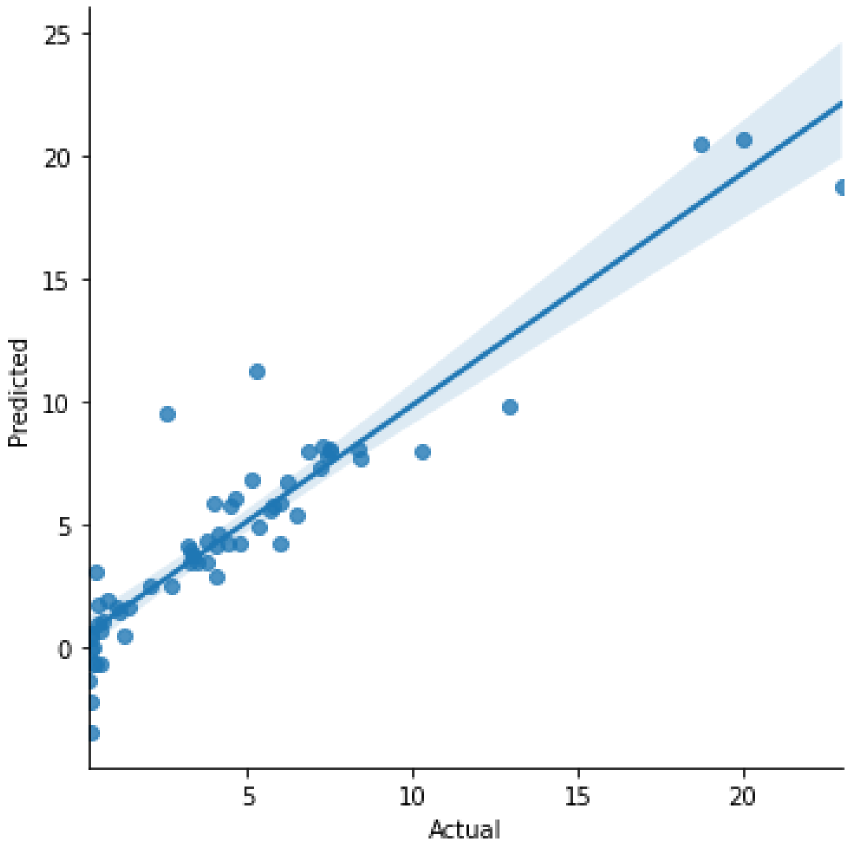
```
cars3
```

	Actual	Predicted
0	7.40	7.862732
1	4.00	2.968287
2	0.50	-0.590305
3	3.15	4.213360
4	1.25	0.483176
...	...	...
56	18.75	20.480622
57	0.50	0.662504
58	6.45	5.400274
59	5.65	5.658562
60	0.25	0.647876

61 rows × 2 columns

```
sns.lmplot(x='Actual',y='Predicted',data=cars3)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f2582b06e10>
```



Plotted the linear regression line and also the data points to check how accurately does it

```
'Mean absolute error = ' + str(mean_absolute_error(y_test, y_pred))
'Mean squared error = ' + str(mean_squared_error(y_test, y_pred))
'R2 score = ' + str(r2_score(y_test, y_pred))
```

```
'R2 score = 0.8625260513315252'
```

Computed the Mean absolute error, Mean Squared error and the R2 Score

```
accuracy = lr.score(X_test,y_test)
print(accuracy*100,'%')
```

```
86.2526051331525 %
```

This is the Accuracy of the model

```
Price = lr.predict([[8,25000,1,6,0,1,1,1]])[0]
```

Predicted the selling price of another car by providing the attributes

```
"The selling price of your car will be " + str(Price) + " Lacs"
```

```
'The selling price of your car will be 4.444588027463719 Lacs'
```

Conclusion : Hence we trained the datase for linear and multi linear regression and predicted the value for both the model by passing the parameters in the model.predict function.