Experiment No. – 3

Student Name: ANIKET KUMAR
UID: 20BCS5306
Branch: CSE
Section/Group: 20BCS_WM-703 / B
Semester: 5th
Date of Performance: 06th Sep, 2022
Subject Name: Machine Learning Lab
Subject Code: 20CSP-317

Aim/Overview of the practical:

Implement Linear Regression.

Apparatus/Simulatorused:

- Jupyter Notebook/GoogleCollab
- Python
- Pand as Library
- Seaborn Library
- Standard Dataset

Code and Output:

```
In [2]: import pandas as pd
        import numpy as np
        import seaborn as sns
```

```
In [3]: cd desktop

C:\Users\HP\desktop
```

```
In [4]: cars_data=pd.read_csv('Toyota.csv',index_col=0,na_values=["??","????"])
```

```
In [5]: cars_data
```

Out[5]:

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23.0	46986.0	Diesel	90.0	1.0	0	2000	three	1165
1	13750	23.0	72937.0	Diesel	90.0	1.0	0	2000	3	1165
2	13950	24.0	41711.0	Diesel	90.0	NaN	0	2000	3	1165
3	14950	26.0	48000.0	Diesel	90.0	0.0	0	2000	3	1165
4	13750	30.0	38500.0	Diesel	90.0	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...
1431	7500	NaN	20544.0	Petrol	86.0	1.0	0	1300	3	1025
1432	10845	72.0	NaN	Petrol	86.0	0.0	0	1300	3	1015
1433	8500	NaN	17016.0	Petrol	86.0	0.0	0	1300	3	1015
1434	7250	70.0	NaN	NaN	86.0	1.0	0	1300	3	1015
1435	6950	76.0	1.0	Petrol	110.0	0.0	0	1600	5	1114

1436 rows × 10 columns

```
In [6]: cars=cars_data.copy()
```

```
In [7]: cars.drop_duplicates(keep='first',inplace=True)
```

```
In [9]: cars
```

Out[9]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500	23.0	46986	Diesel	90	1.0	0	2000	three	1165
1	1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165
2	2	13950	24.0	41711	Diesel	90	NaN	0	2000	3	1165
3	3	14950	26.0	48000	Diesel	90	0.0	0	2000	3	1165
4	4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500	NaN	20544	Petrol	86	1.0	0	1300	3	1025
1432	1432	10845	72.0	??	Petrol	86	0.0	0	1300	3	1015
1433	1433	8500	NaN	17016	Petrol	86	0.0	0	1300	3	1015
1434	1434	7250	70.0	??	NaN	86	1.0	0	1300	3	1015
1435	1435	6950	76.0	1	Petrol	110	0.0	0	1600	5	1114

1436 rows × 11 columns

In [11]:
```
cars_omit.isnull().sum()
```

Out[11]:
```
Price        0
Age          0
KM           0
FuelType     0
HP           0
MetColor     0
Automatic    0
CC           0
Doors        0
Weight       0
dtype: int64
```

In [12]:
```
cars_omit=pd.get_dummies(cars_omit,drop_first=True)
```

In [14]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

In [15]:
```
# =============================================================================
# MODEL BUILDING WITH OMITTED DATA
# =============================================================================

# Separating input and output features
x1 = cars_omit.drop(['Price'], axis='columns', inplace=False)
y1 = cars_omit['Price']
```

In [16]:
```
# Splitting data into test and train
X_train, X_test, y_train, y_test = train_test_split(x1, y1, test_size=0.3, random_state = 3)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(767, 15) (329, 15) (767,) (329,)
```

In [24]:
```
# =============================================================================
# LINEAR REGRESSION WITH OMITTED DATA
# =============================================================================

# Setting intercept as true
lgr=LinearRegression(fit_intercept=True)
```

In [25]:
```
# Model
model_lin1=lgr.fit(X_train,y_train)
```

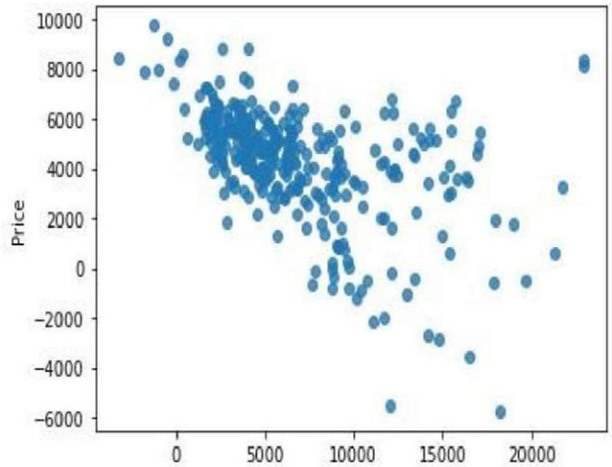
In [26]:
```
# Predicting model on test set
cars_predictions_lin1 = lgr.predict(X_test)
```

In [27]:
```
# Computing MSE and RMSE
lin_mse1 = mean_squared_error(y_test, cars_predictions_lin1)
lin_rmse1 = np.sqrt(lin_mse1)
print(lin_rmse1)
```

```
4844.550770469122
```

In [28]:
```
# R squared value
r2_lin_test1=model_lin1.score(X_test,y_test)
r2_lin_train1=model_lin1.score(X_train,y_train)
print(r2_lin_test1,r2_lin_train1)
```

```
-0.49794311219235876 0.9929299667779579
```



In [30]:
```
# ==============================================================================
# MODEL BUILDING WITH IMPUTED DATA
# ==============================================================================

cars_imputed = cars.apply(lambda x:x.fillna(x.median()) \
                  if x.dtype=='float' else \
                  x.fillna(x.value_counts().index[0]))
cars_imputed.isnull().sum()
```

In [29]:
```
# Regression diagnostics- Residual plot analysis
residuals1=y_test-cars_predictions_lin1
sns.regplot(x=cars_predictions_lin1, y=residuals1, scatter=True,
            fit_reg=False)
residuals1.describe()
```

Out[29]:
```
count     334.000000
mean     4293.567433
std      2247.236266
min     -5747.588323
25%      3470.908983
50%      4638.141910
75%      5645.491231
max      9747.071200
Name: Price, dtype: float64
```

In [31]:
```
# Converting categorical variables to dummy variables
cars_imputed=pd.get_dummies(cars_imputed,drop_first=True)
```

In [33]:
```
# ===============================================================================
# MODEL BUILDING WITH IMPUTED DATA
# ===============================================================================

# Separating input and output feature
x2 = cars_imputed.drop(['Price'], axis='columns', inplace=False)
y2 = cars_imputed['Price']
```

In [34]:
```
# Plotting the variable price
prices = pd.DataFrame({"1. Before":y2, "2. After":np.log(y2)})
prices.hist()
```

In [36]:
```
# ===============================================================================
# LINEAR REGRESSION WITH IMPUTED DATA
# ===============================================================================

# Setting intercept as true
lgr2=LinearRegression(fit_intercept=True)
```

In [37]:
```
# Model
model_lin2=lgr2.fit(X_train1,y_train1)
```

In [38]:
```
# Predicting model on test set
cars_predictions_lin2 = lgr2.predict(X_test1)
```

In [39]:
```
# Computing MSE and RMSE
lin_mse2 = mean_squared_error(y_test1, cars_predictions_lin2)
lin_rmse2 = np.sqrt(lin_mse2)
print(lin_rmse2)
```

```
1946.4094207402252
```

```
In [40]: # R squared value
         r2_lin_test2=model_lin2.score(X_test1,y_test1)
         r2_lin_train2=model_lin2.score(X_train1,y_train1)
         print(r2_lin_test2,r2_lin_train2)

         0.7155594432226939 0.9833549973827853
```

```
In [43]: # Final output

         print("Metrics for models built from data where missing values were omitted")
         print("R squared value for train from Linear Regression=  %s"% r2_lin_train1)
         print("R squared value for test from Linear Regression=  %s"% r2_lin_test1)
         print("RMSE value for test from Linear Regression=  %s"% lin_rmse1)
         print("Metrics for models built from data where missing values were imputed")
         print("R squared value for train from Linear Regression=  %s"% r2_lin_train2)
         print("R squared value for test from Linear Regression=  %s"% r2_lin_test2)
         print("RMSE value for test from Linear Regression=  %s"% lin_rmse2)

         Metrics for models built from data where missing values were omitted
         R squared value for train from Linear Regression=  0.9929299667779579
         R squared value for test from Linear Regression=  -0.49794311219235876
         RMSE value for test from Linear Regression=  4844.550770469122
         Metrics for models built from data where missing values were imputed
         R squared value for train from Linear Regression=  0.9833549973827853
         R squared value for test from Linear Regression=  0.7155594432226939
         RMSE value for test from Linear Regression=  1946.4094207402252
```

```
In [ ]:
```

Evaluation Grid(To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			