Name: Garcia, John Carlos M.

Section: CPE22S3

Date Performed: 04/22/2024

Date Submitted: 04/27/2024

Linear Regression Analysis

Setup

```
 1 import pandas as pd
 2 import numpy as np
 3 import seaborn as sns
 4 import matplotlib.pyplot as plt
 5 from sklearn.model_selection import train_test_split
 6 from sklearn import metrics
 7 from sklearn.linear_model import LinearRegression
 8 import hvplot.pandas
 9 %matplotlib inline
10
11 from sklearn.metrics import r2_score, mean_squared_error
```

```
 1 pip install ucimlrepo
```

```
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.10/dist-packages (0.0
```

```python
from ucimlrepo import fetch_ucirepo

# fetch dataset
automobile = fetch_ucirepo(id=10)

# data (as pandas dataframes)
X = automobile.data.features
y = automobile.data.targets

# metadata
print("Metadata:\n",automobile.metadata)

# variable information
print("\nVariables:\n",automobile.variables)
```

```
Metadata:
 {'uci_id': 10, 'name': 'Automobile', 'repository_url': 'https://archive.ics.uci.edu/

Variables:
                  name     role         type demographic  \
0                price  Feature   Continuous        None
1          highway-mpg  Feature   Continuous        None
2             city-mpg  Feature   Continuous        None
3             peak-rpm  Feature   Continuous        None
4           horsepower  Feature   Continuous        None
5   compression-ratio  Feature   Continuous        None
6               stroke  Feature   Continuous        None
7                 bore  Feature   Continuous        None
8          fuel-system  Feature  Categorical        None
9          engine-size  Feature   Continuous        None
10   num-of-cylinders  Feature      Integer        None
11        engine-type  Feature  Categorical        None
12        curb-weight  Feature   Continuous        None
13             height  Feature   Continuous        None
14              width  Feature   Continuous        None
15             length  Feature   Continuous        None
16         wheel-base  Feature   Continuous        None
17    engine-location  Feature       Binary        None
18       drive-wheels  Feature  Categorical        None
19         body-style  Feature  Categorical        None
20       num-of-doors  Feature      Integer        None
21         aspiration  Feature       Binary        None
22          fuel-type  Feature       Binary        None
23               make  Feature  Categorical        None
24  normalized-losses  Feature   Continuous        None
25          symboling   Target      Integer        None

                                        description units missing_values
0                      continuous from 5118 to 45400  None            yes
1                           continuous from 16 to 54  None             no
2                           continuous from 13 to 49  None             no
3                       continuous from 4150 to 6600  None            yes
4                          continuous from 48 to 288  None            yes
5                            continuous from 7 to 23  None             no
6                       continuous from 2.07 to 4.17  None            yes
```

```
7                        continuous from 2.54 to 3.94  None            yes
8         1bbl, 2bbl, 4bbl, idi, mfi, mpfi, spdi, spfi  None             no
9                             continuous from 61 to 326  None             no
10          eight, five, four, six, three, twelve, two  None             no
11              dohc, dohcv, l, ohc, ohcf, ohcv, rotor  None             no
12                        continuous from 1488 to 4066  None             no
13                        continuous from 47.8 to 59.8  None             no
14                        continuous from 60.3 to 72.3  None             no
15                      continuous from 141.1 to 208.1  None             no
16                          continuous from 86.6 120.9  None             no
17                                       front, rear  None             no
18                                     4wd, fwd, rwd  None             no
19       hardtop, wagon, sedan, hatchback, convertible  None             no
20                                         four, two  None            yes
21                                       std, turbo  None             no
22                                       diesel, gas  None             no
23  alfa-romero, audi, bmw, chevrolet, dodge, hond...  None             no
```

Concatenation

```
1 linear_df = pd.concat([X, y], axis=1) #Combine both dataframes into one for more efficier
```

Exploration

```
1 print("Head:\n",linear_df.head(), "\n\n")
2
3 print("DTypes:\n",linear_df.dtypes, "\n\n")
4
5 print("Description:\n",linear_df.describe())
```

```
Head:
      price  highway-mpg  city-mpg  peak-rpm  horsepower  compression-ratio  \
0  13495.0           27        21    5000.0       111.0                9.0
1  16500.0           27        21    5000.0       111.0                9.0
2  16500.0           26        19    5000.0       154.0                9.0
3  13950.0           30        24    5500.0       102.0               10.0
4  17450.0           22        18    5500.0       115.0                8.0

   stroke  bore fuel-system  engine-size  ...  wheel-base engine-location  \
0    2.68  3.47        mpfi          130  ...        88.6           front
1    2.68  3.47        mpfi          130  ...        88.6           front
2    3.47  2.68        mpfi          152  ...        94.5           front
3    3.40  3.19        mpfi          109  ...        99.8           front
4    3.40  3.19        mpfi          136  ...        99.4           front

   drive-wheels   body-style  num-of-doors  aspiration  fuel-type  \
0           rwd  convertible           2.0         std        gas
1           rwd  convertible           2.0         std        gas
2           rwd    hatchback           2.0         std        gas
```

```
3           fwd        sedan           4.0         std        gas
4           4wd        sedan           4.0         std        gas

          make normalized-losses symboling
0  alfa-romero               NaN         3
1  alfa-romero               NaN         3
2  alfa-romero               NaN         1
3         audi             164.0         2
4         audi             164.0         2

[5 rows x 26 columns]


DTypes:
 price                float64
highway-mpg             int64
city-mpg                int64
peak-rpm              float64
horsepower            float64
compression-ratio     float64
stroke                float64
bore                  float64
fuel-system            object
engine-size             int64
num-of-cylinders        int64
engine-type            object
curb-weight             int64
height                float64
width                 float64
length                float64
wheel-base            float64
engine-location        object
drive-wheels           object
body-style             object
num-of-doors          float64
aspiration             object
fuel-type              object
make                   object
normalized-losses     float64
```

Identification of Missing Values

```
1 print("Nulls:\n",linear_df.isnull().sum())
```

```
Nulls:
 price                4
highway-mpg           0
city-mpg              0
peak-rpm              2
horsepower            2
compression-ratio     0
stroke                4
bore                  4
fuel-system           0
```

```
engine-size           0
num-of-cylinders      0
engine-type           0
curb-weight           0
height                0
width                 0
length                0
wheel-base            0
engine-location       0
drive-wheels          0
body-style            0
num-of-doors          2
aspiration            0
fuel-type             0
make                  0
normalized-losses    41
symboling             0
dtype: int64
```

```
print("Unique values in 'price' column:")
print(linear_df['price'].unique())

print("\n\nUnique values in 'peak-rpm' column:")
print(linear_df['peak-rpm'].unique())

print("\n\nUnique values in 'horsepower' column:")
print(linear_df['horsepower'].unique())

print("\n\nUnique values in 'stroke' column:")
print(linear_df['stroke'].unique())

print("\n\nUnique values in 'bore' column:")
print(linear_df['bore'].unique())

print("\n\nUnique values in 'num-of-doors' column:")
print(linear_df['num-of-doors'].unique())

print("\n\nUnique values in 'normalized-losses' column:")
print(linear_df['normalized-losses'].unique())
```

```
Unique values in 'price' column:
[13495. 16500. 13950. 17450. 15250. 17710. 18920. 23875.    nan 16430.
 16925. 20970. 21105. 24565. 30760. 41315. 36880.  5151.  6295.  6575.
  5572.  6377.  7957.  6229.  6692.  7609.  8558.  8921. 12964.  6479.
  6855.  5399.  6529.  7129.  7295.  7895.  9095.  8845. 10295. 12945.
 10345.  6785. 11048. 32250. 35550. 36000.  5195.  6095.  6795.  6695.
  7395. 10945. 11845. 13645. 15645.  8495. 10595. 10245. 10795. 11245.
 18280. 18344. 25552. 28248. 28176. 31600. 34184. 35056. 40960. 45400.
 16503.  5389.  6189.  6669.  7689.  9959.  8499. 12629. 14869. 14489.
  6989.  8189.  9279.  5499.  7099.  6649.  6849.  7349.  7299.  7799.
  7499.  7999.  8249.  8949.  9549. 13499. 14399. 17199. 19699. 18399.
 11900. 13200. 12440. 13860. 15580. 16900. 16695. 17075. 16630. 17950.
 18150. 12764. 22018. 32528. 34028. 37028.  9295.  9895. 11850. 12170.
```

```
 15040. 15510. 18620.  5118.  7053.  7603.  7126.  7775.  9960.  9233.
 11259.  7463. 10198.  8013. 11694.  5348.  6338.  6488.  6918.  7898.
  8778.  6938.  7198.  7788.  7738.  8358.  9258.  8058.  8238.  9298.
  9538.  8449.  9639.  9989. 11199. 11549. 17669.  8948. 10698.  9988.
 10898. 11248. 16558. 15998. 15690. 15750.  7975.  7995.  8195.  9495.
  9995. 11595.  9980. 13295. 13845. 12290. 12940. 13415. 15985. 16515.
 18420. 18950. 16845. 19045. 21485. 22470. 22625.]


Unique values in 'peak-rpm' column:
[5000. 5500. 5800. 4250. 5400. 5100. 4800. 6000. 4750. 4650. 4200. 4350.
 4500. 5200. 4150. 5600. 5900. 5750.   nan 5250. 4900. 4400. 6600. 5300.]


Unique values in 'horsepower' column:
[111. 154. 102. 115. 110. 140. 160. 101. 121. 182.  48.  70.  68.  88.
 145.  58.  76.  60.  86. 100.  78.  90. 176. 262. 135.  84.  64. 120.
  72. 123. 155. 184. 175. 116.  69.  55.  97. 152. 200.  95. 142. 143.
 207. 288.  nan  73.  82.  94.  62.  56. 112.  92. 161. 156.  52.  85.
 114. 162. 134. 106.]


Unique values in 'stroke' column:
[2.68 3.47 3.4  2.8  3.19 3.39 3.03 3.11 3.23 3.46 3.9  3.41 3.07 3.58
 4.17 2.76 3.15  nan 3.16 3.64 3.1  3.35 3.12 3.86 3.29 3.27 3.52 2.19
 3.21 2.9  2.07 2.36 2.64 3.08 3.5  3.54 2.87]


Unique values in 'bore' column:
[3.47 2.68 3.19 3.13 3.5  3.31 3.62 2.91 3.03 2.97 3.34 3.6  2.92 3.15
 3.43 3.63 3.54 3.08  nan 3.39 3.76 3.58 3.46 3.8  3.78 3.17 3.35 3.59
 2.99 3.33 3.7  3.61 3.94 3.74 2.54 3.05 3.27 3.24 3.01]


Unique values in 'num-of-doors' column:
[ 2.  4. nan]


Unique values in 'normalized-losses' column:
[ nan 164. 158. 192. 188. 121.  98.  81. 118. 148. 110. 145. 137. 101.
  78. 106.  85. 107. 104. 113. 150. 129. 115.  93. 142. 161. 153. 125.
 128. 122. 103. 168. 108. 194. 231. 119. 154.  74. 186.  83. 102.  89.
  87.  77.  91. 134.  65. 197.  90.  94. 256.  95.]
```

Identification of Duplicated Rows

```
1 duplicates = linear_df.duplicated()
2 print("Duplicates:\n\n", duplicates, "\n\n")
3 print("Duplicate Rows:\n\n",linear_df[duplicates])
```

```
Duplicates:

 0      False
```

```
1      False
2      False
3      False
4      False
       ...
200    False
201    False
202    False
203    False
204    False
Length: 205, dtype: bool


Duplicate Rows:

 Empty DataFrame
Columns: [price, highway-mpg, city-mpg, peak-rpm, horsepower, compression-ratio, stroke,
Index: []

[0 rows x 26 columns]
```

Cleaning

Since there's no column with the object type in the columns we are going to use, we don't have to convert the other object columns into integers

Clean Missing Values

As the price column is the only one with missing values, we will just drop those rows, as it can only lead to inconsistent data

```
1 linear_df = linear_df[['price', 'highway-mpg', 'city-mpg']].dropna()
2 print("Nulls:\n",linear_df.isnull().sum())
3 print("\n\nHead:\n",linear_df.head(), "\n\n")
4 print("DTypes:\n",linear_df.dtypes, "\n\n")
5 print("Columns:\n",linear_df.columns, "\n\n")
```

```
Nulls:
 price          0
highway-mpg    0
city-mpg       0
dtype: int64


Head:
      price  highway-mpg  city-mpg
```

```
0  13495.0           27        21
1  16500.0           27        21
2  16500.0           26        19
3  13950.0           30        24
4  17450.0           22        18


DTypes:
 price          float64
highway-mpg      int64
city-mpg         int64
dtype: object


Columns:
 Index(['price', 'highway-mpg', 'city-mpg'], dtype='object')
```

Removing Duplicates

Nothing to remove

Task 1: Predict the price of the automobile based on the miles per gallon of gasoline consumed

X and y arrays

```
1 X = linear_df[['highway-mpg', 'city-mpg']]
2 y = linear_df[['price']]
3 print("X = ", X.shape, "\ny = ", y.shape)
```

```
X =  (201, 2)
y =  (201, 1)
```

Train Test Split

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101
```

```
1 X_train.shape
```

```
(140, 2)
```

```
1 X_test.shape
```

```
(61, 2)
```

```
1 y_train.shape
```

```
(140, 1)
```

```
1 y_test.shape
```

```
(61, 1)
```

Linear Regression

```
1 model = LinearRegression()
```

```
1 model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Model Evaluation

```
1 model.coef_
```

```
array([[-1121.5850521 ,   350.19274032]])
```

```
1 pd.DataFrame(model.coef_.T, X.columns, columns=['Coedicients'])
```

	Coedicients
highway-mpg	-1121.585052
city-mpg	350.192740

Predictions from our Model

```
1 y_pred = model.predict(X_test)
```

Regression Evaluation Metrics

```
MAE = r2_score(y_test, y_pred)
MSE = mean_squared_error(y_test, y_pred)
RMSE = np.sqrt(MSE)

print("R-squared:", MAE)
print("Mean Squared Error:", MSE)
print("Root Mean Squared Error:", RMSE)
```

```
R-squared: 0.4340915662521546
Mean Squared Error: 35085547.01563065
Root Mean Squared Error: 5923.3054129962475
```

```
linear_df['price'].mean()
```

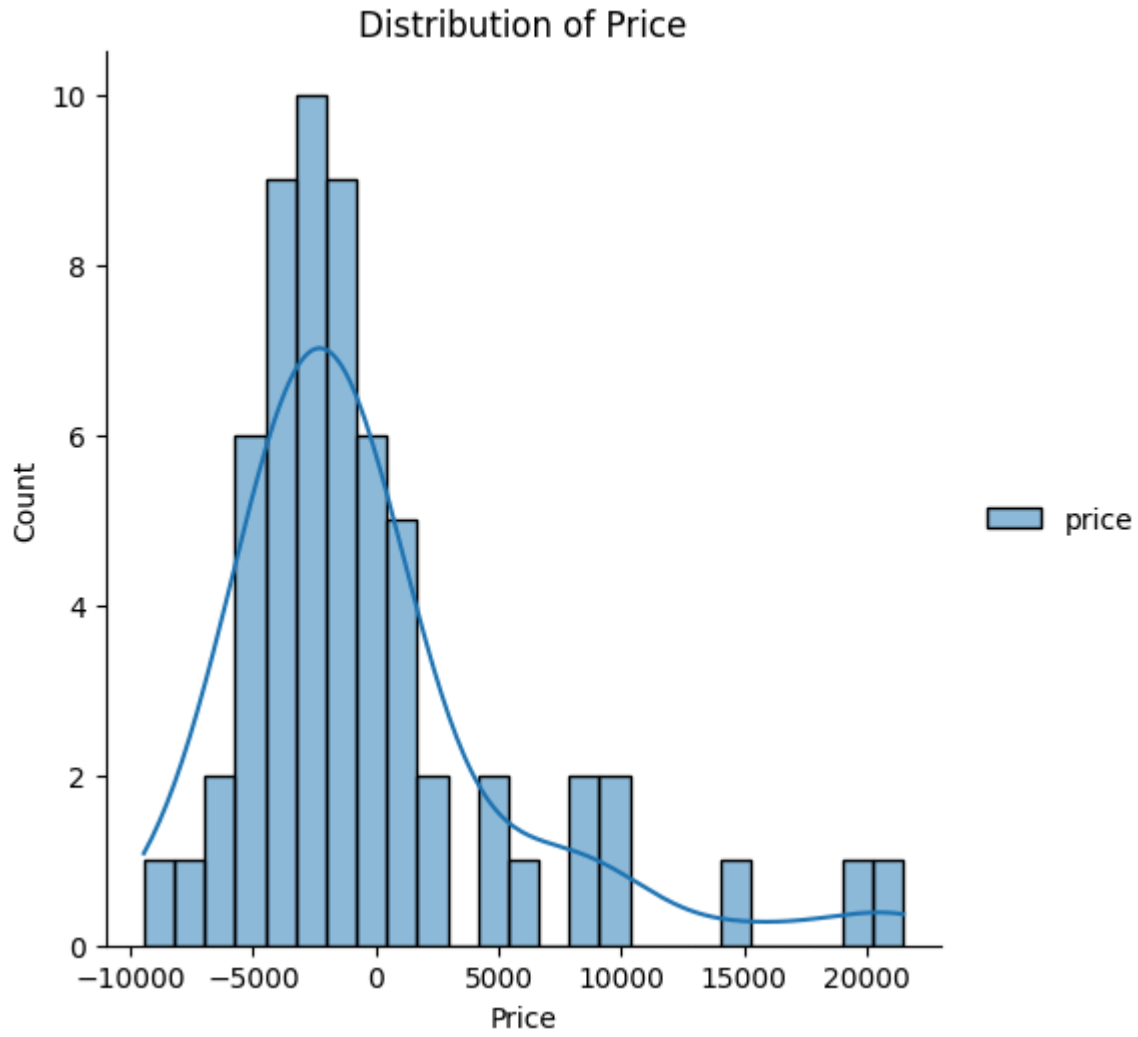
```
13207.129353233831
```

Residuals Histogram

```
test_residual= y_test - y_pred
```

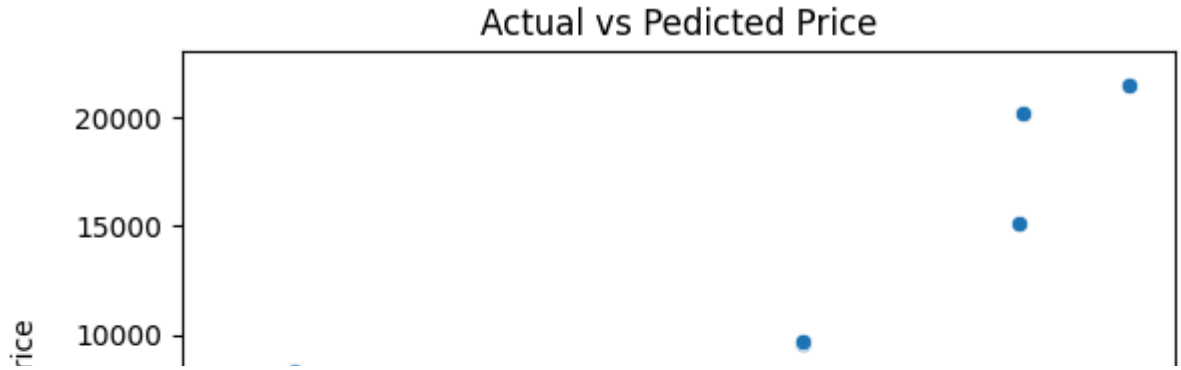
```
sns.displot(test_residual, bins=25, kde=True)
plt.xlabel('Price')
plt.title('Distribution of Price')
```

```
Text(0.5, 1.0, 'Distribution of Price')
```




```
1 sns.scatterplot(x=y_test.values.flatten(), y=test_residual.values.flatten())
2 plt.axhline(y=0, color='r', ls='--')
3 plt.xlabel('Actual Price')
4 plt.ylabel('Predicted Price')
5 plt.title('Actual vs Pedicted Price')
```

```
Text(0.5, 1.0, 'Actual vs Pedicted Price')
```



Conclusions:

1. An R-square value of 0.43 suggests that the miles per gallon of the vehicle in Hihgways and Cities can explain 43% of the automobiles' prices, whereas the remaining 57% can be explained by other factors that I did not include in the analysis.
2. A Mean Square Error of 35085547.02 indicates that there is a large difference to the actual and predicted price.
3. The Root Mean Squared Error of 5923.31 tells us that the predictions have an error of around that value, in terms of price—which would be a prediction error of $5923.31 in this case.