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

data = pd.read_csv(r'/content/CarPrice_Assignment.csv')

print(data.info())
print(data.describe())
print(data.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    object 
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    object 
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
None
            car_ID   symboling   wheelbase   carlength    carwidth
```

```
carheight  \
count  205.000000  205.000000  205.000000  205.000000  205.000000
205.000000
mean   103.000000    0.834146   98.756585  174.049268   65.907805
53.724878
std     59.322565    1.245307    6.021776   12.337289    2.145204
2.443522
min      1.000000   -2.000000   86.600000  141.100000   60.300000
47.800000
25%     52.000000    0.000000   94.500000  166.300000   64.100000
52.000000
50%    103.000000    1.000000   97.000000  173.200000   65.500000
54.100000
75%    154.000000    2.000000  102.400000  183.100000   66.900000
55.500000
max    205.000000    3.000000  120.900000  208.100000   72.300000
59.800000

        curbweight  enginesize   boreratio      stroke
compressionratio  \
count   205.000000  205.000000  205.000000  205.000000
205.000000
mean   2555.565854  126.907317    3.329756    3.255415
10.142537
std     520.680204   41.642693    0.270844    0.313597
3.972040
min    1488.000000   61.000000    2.540000    2.070000
7.000000
25%    2145.000000   97.000000    3.150000    3.110000
8.600000
50%    2414.000000  120.000000    3.310000    3.290000
9.000000
75%    2935.000000  141.000000    3.580000    3.410000
9.400000
max    4066.000000  326.000000    3.940000    4.170000
23.000000

       horsepower      peakrpm     citympg  highwaympg         price
count  205.000000   205.000000  205.000000  205.000000    205.000000
mean   104.117073  5125.121951   25.219512   30.751220  13276.710571
std     39.544167   476.985643    6.542142    6.886443   7988.852332
min     48.000000  4150.000000   13.000000   16.000000   5118.000000
25%     70.000000  4800.000000   19.000000   25.000000   7788.000000
50%     95.000000  5200.000000   24.000000   30.000000  10295.000000
75%    116.000000  5500.000000   30.000000   34.000000  16503.000000
max    288.000000  6600.000000   49.000000   54.000000  45400.000000
car_ID              0
symboling           0
CarName             0
```

```
fueltype             0
aspiration           0
doornumber           0
carbody              0
drivewheel           0
enginelocation       0
wheelbase            0
carlength            0
carwidth             0
carheight            0
curbweight           0
enginetype           0
cylindernumber       0
enginesize           0
fuelsystem           0
boreratio            0
stroke               0
compressionratio     0
horsepower           0
peakrpm              0
citympg              0
highwaympg           0
price                0
dtype: int64
```

```
# One-hot encoding for the 'fueltype'
data = pd.get_dummies(data, columns=['fueltype'], drop_first=True)

# Check the updated DataFrame's columns
print("Columns after one-hot encoding:\n", data.columns)
```

```
Columns after one-hot encoding:
 Index(['car_ID', 'symboling', 'CarName', 'aspiration', 'doornumber',
'carbody',
       'drivewheel', 'enginelocation', 'wheelbase', 'carlength',
'carwidth',
       'carheight', 'curbweight', 'enginetype', 'cylindernumber',
'enginesize',
       'fuelsystem', 'boreratio', 'stroke', 'compressionratio',
'horsepower',
       'peakrpm', 'citympg', 'highwaympg', 'price', 'fueltype_gas'],
      dtype='object')
```

```
# Convert categorical columns using Label Encoding for other
categorical variables
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
data["enginetype"] = label_encoder.fit_transform(data["enginetype"])
data["carbody"] = label_encoder.fit_transform(data["carbody"])
```

```
# Selecting features (independent variables)
# Update feature selection based on the actual column names
X = data[['horsepower', 'enginesize', 'enginetype', 'carbody',
'fueltype_diesel']] if 'fueltype_diesel' in data.columns else
data[['horsepower', 'enginesize', 'enginetype', 'carbody']]

# target variable
Y = data['price']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.3, random_state=42)

model = LinearRegression()
model.fit(X_train, Y_train)
```

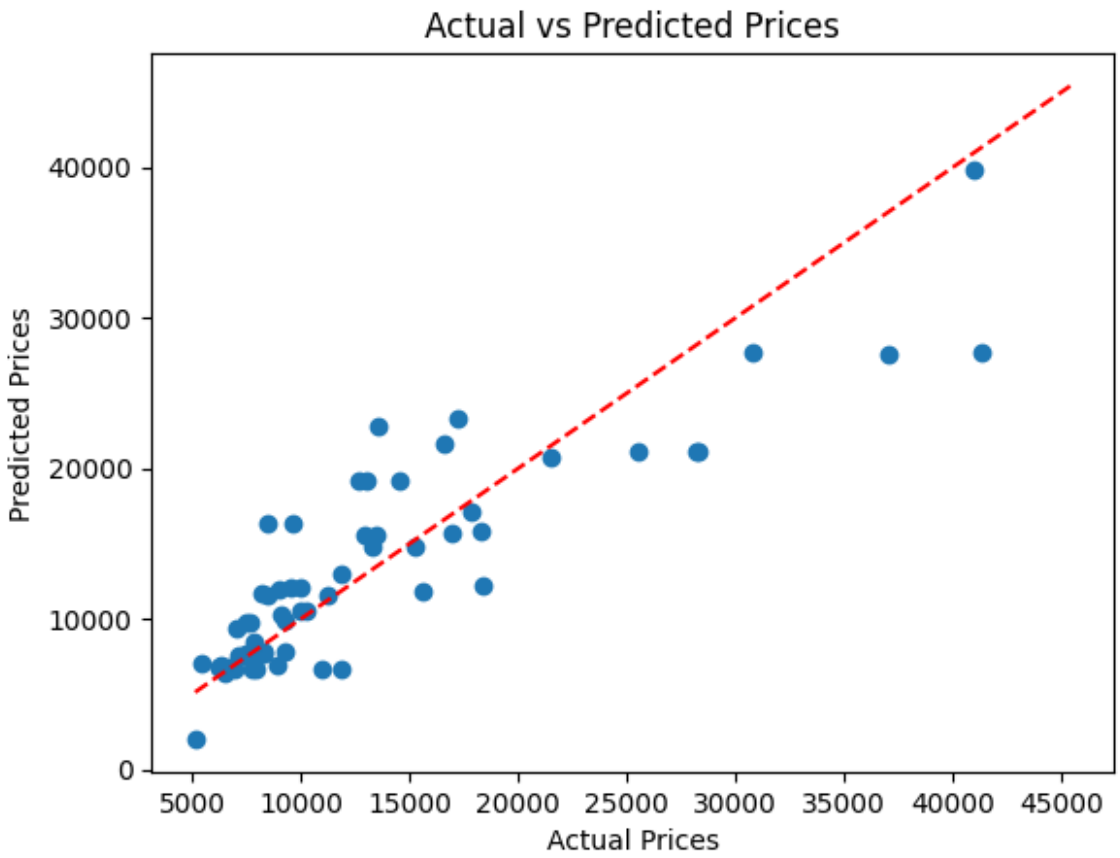
```
LinearRegression()
```

```
y_pred = model.predict(X_test)

print('Mean Squared Error: ', mean_squared_error(Y_test, y_pred))
print('Mean Absolute Error: ', mean_absolute_error(Y_test, y_pred))
print('Root Mean Squared Error: ',
math.sqrt(mean_squared_error(Y_test, y_pred)))
```

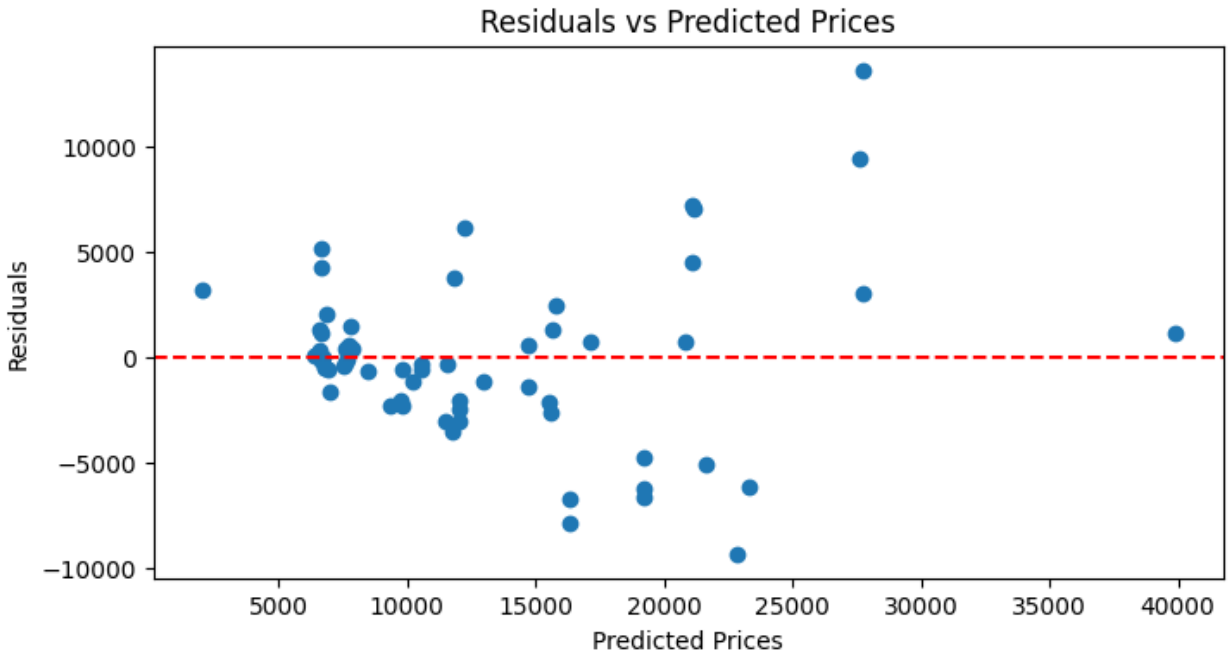
```
Mean Squared Error:  15633073.22984185
Mean Absolute Error:  2749.6020331322816
Root Mean Squared Error:  3953.868135110458
```

```
# Visualizing actual and predicted values
plt.scatter(Y_test, y_pred)
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual vs Predicted Prices')
plt.plot([Y.min(), Y.max()], [Y.min(), Y.max()], color='red',
linestyle='--')  # Diagonal line
plt.show()
```




```
# Calculating residuals
residuals = Y_test - y_pred

# Plotting residuals
plt.figure(figsize=(8, 4))
plt.scatter(y_pred, residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Predicted Prices')
plt.ylabel('Residuals')
plt.title('Residuals vs Predicted Prices')
plt.show()
```


Residuals vs Predicted Prices
Residuals
Predicted Prices
10000
5000
0
−5000
−10000
5000
10000
15000
20000
25000
30000
35000
40000