Linear Regression

Step 1: Importing the Necessary Libraries

First, we'll import the libraries required for data manipulation, visualization, and modeling.

In [2]:

```
import numpy as np
import pandas as pd
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, r2_score
```

Step 2: Creating the DataFrame

Next, we'll create a DataFrame from the provided dataset.

In [ ]:

```
# Loading the dataset
df = pd.read_csv('cars.csv')

# Display the first few rows of the DataFrame to ensure it's loaded correctly
print(df.head())
```

Step 3: Data Preprocessing

We need to convert categorical variables into numerical values using one-hot encoding and ensure all columns are in a suitable format for regression.

In [6]:

```
# One-hot encoding categorical variables
df = pd.get_dummies(df, columns=['Brand', 'Model', 'Fuel_Type', 'Transmission', 'Owner_Type'], drop_first=True)

# Display the DataFrame
print(df.head())
```

```
   Car_ID  Year  Kilometers_Driven  Mileage  Engine  Power  Seats    Price  \
0       1  2018              50000       15    1498    108      5   800000
1       2  2019              40000       17    1597    140      5  1000000
2       3  2017              20000       10    4951    395      4  2500000
3       4  2020              30000       23    1248     74      5   600000
4       5  2016              60000       18    1999    194      5   850000

   Brand_BMW  Brand_Ford  ...  Model_WR-V  Model_X1  Model_X3  Model_X5  \
0          0           0  ...           0         0         0         0
1          0           0  ...           0         0         0         0
2          0           1  ...           0         0         0         0
3          0           0  ...           0         0         0         0
4          0           0  ...           0         0         0         0

   Model_XUV300  Model_Yaris  Fuel_Type_Petrol  Transmission_Manual  \
0             0            0                 1                    1
1             0            0                 1                    0
2             0            0                 1                    0
3             0            0                 0                    1
4             0            0                 0                    0

   Owner_Type_Second  Owner_Type_Third
0                  0                 0
1                  1                 0
2                  0                 0
3                  0                 1
4                  1                 0

[5 rows x 79 columns]
```

Step 4: Defining Features and Target Variable

Select the features (X) and the target variable (y).

In [7]:

```
# Defining the feature matrix and target vector
X = df.drop(columns=['Car_ID', 'Price'])
y = df['Price']
```

Step 5: Splitting the Data

Split the data into training and testing sets.

In [8]:

```
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Print the shapes of the train/test splits
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(70, 77) (30, 77) (70,) (30,)
```

Step 6: Training the Linear Regression Model

Instantiate the Linear Regression model and train it on the training data.

In [9]:

```
# Instantiating the Linear Regression model
model = LinearRegression()

# Training the model
model.fit(X_train, y_train)
```

Out[9]:

```
LinearRegression()
```

Step 7: Making Predictions

Use the trained model to make predictions on the test data.

In [10]:

```
# Making predictions on the test data
y_pred = model.predict(X_test)
```

Step 8: Evaluating the Model

Evaluate the model using metrics like Mean Squared Error (MSE) and R-squared score.

In [11]:

```
# Calculating and printing the MSE and R-squared score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Root Mean Squared Error (RMSE): {np.sqrt(mse)}')
print(f'R-squared Score: {r2}')
```

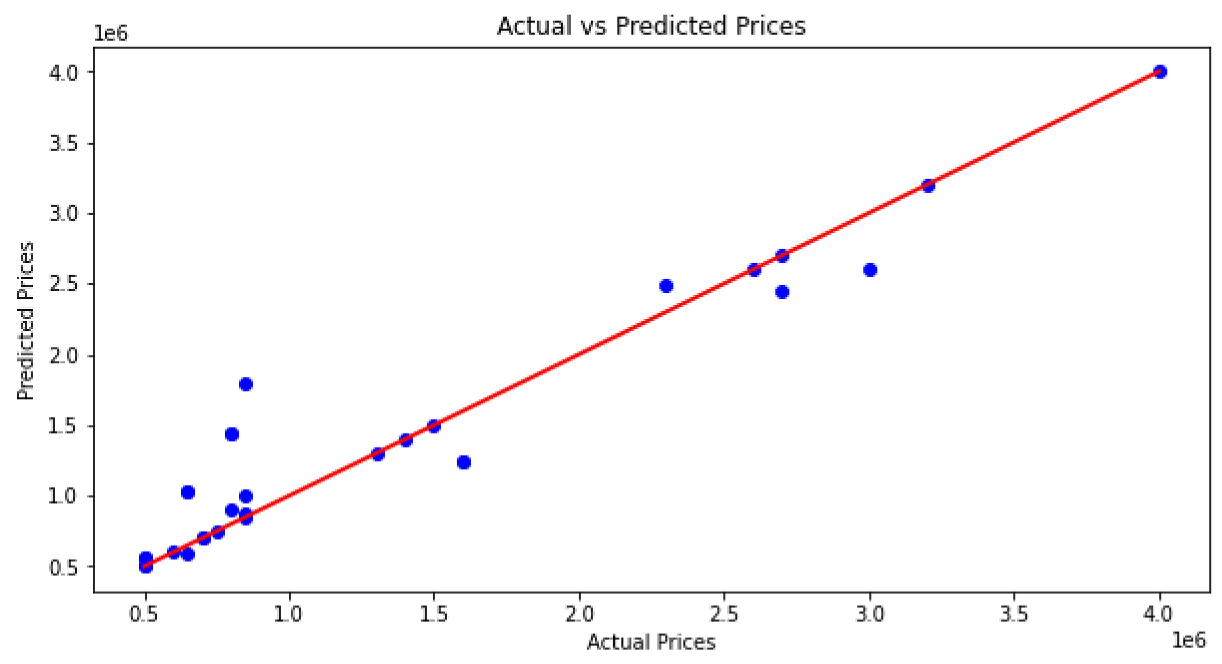
```
Root Mean Squared Error (RMSE): 292186.53264320886
R-squared Score: 0.9067460338468155
```

Step 9: Visualizing the Results

Finally, we can visualize the predicted prices versus the actual prices.

In [12]:

```
# Plotting the actual vs predicted prices
plt.figure(figsize=(10, 5))
plt.scatter(y_test, y_pred, color='blue')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linewidth=2)
plt.title('Actual vs Predicted Prices')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.show()
```



Learning Reflection

Building a Linear Regression Model

This project provided a hands-on experience in constructing a linear regression model for predicting car prices. It underscored the significance of each phase in the machine learning pipeline, offering valuable insights into data preprocessing, model training, evaluation, and visualization.

Data Preprocessing

The initial steps involved loading the dataset and preparing it for analysis. Handling categorical variables through one-hot encoding ensured compatibility with the regression model. Additionally, examining and addressing missing data and outliers were crucial for maintaining data integrity and improving model performance.

Feature Selection and Target Variable

Defining the feature matrix and target vector required careful consideration of which variables to include as predictors and which to designate as the target variable. This step involved selecting relevant features that could potentially influence the target variable, such as car specifications and attributes.

Model Training and Evaluation

Training the linear regression model on the training data allowed for learning the underlying patterns and relationships within the dataset. Evaluating the model's performance using metrics like Root Mean Squared Error (RMSE) and R-squared score provided quantitative measures of its predictive accuracy and goodness of fit.

Interpreting Results

Visualizing the predicted prices against the actual prices provided an intuitive understanding of how well the model generalized to unseen data. The scatter plot highlighted areas of agreement and discrepancy between predicted and observed values, offering insights into potential areas for improvement or further investigation.

Summary

This project illuminated the iterative nature of machine learning, where each step builds upon the previous one to iteratively refine the model's performance. It emphasized the importance of data quality, feature engineering, and rigorous evaluation in constructing robust and reliable predictive models. Moving forward, these insights will inform future endeavors in data analysis and predictive modeling, guiding the development of more sophisticated and accurate machine learning solutions.