**Predicting the Price of Toyota Corolla Cars**

To build a multiple linear regression model for predicting the price of Toyota Corolla cars using the specified dataset, we will follow the Machine Learning Life Cycle steps. We'll start with Exploratory Data Analysis (EDA), perform necessary data transformations, build multiple models, and select the best one based on R² values.

Here are the steps we will follow:

1. Import the Libraries.
2. Load the dataset
3. Data Exploration and Cleaning
4. Exploratory Data Analysis (EDA)
5. Data Transformation
6. Split Data into Training and Testing Sets
7. Build Multiple Linear Regression Model
8. Model Evaluation
9. Model Improvement and Comparison

Let's start with loading the dataset and performing the initial EDA.

**Step 1: Import the Libraries**

***import pandas as pd***

***import numpy as np***

***import seaborn as sns***

***import matplotlib.pyplot as plt***

***from sklearn.model\_selection import train\_test\_split***

***from sklearn.linear\_model import LinearRegression***

***from sklearn.metrics import mean\_squared\_error, r2\_score***

***import statsmodels.api as sm***

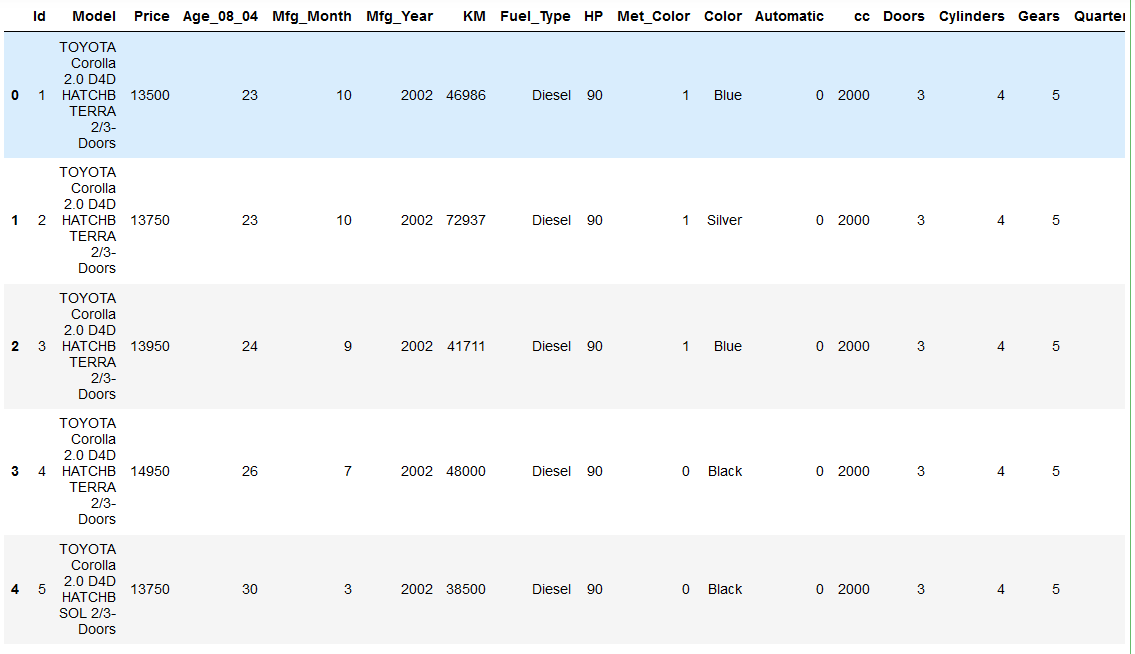
**Step 2: Load the Dataset**

***import pandas as pd***

*df = pd.read\_csv('E:/Top Mentor/All Projects/4.ToyotaCorolla/ToyotaCorolla.csv', encoding='ISO-8859-1')*

*df.head()*

Output:



**Step 3: Data Exploration and Cleaning**

# Check for missing values

*print(df.isnull().sum())*

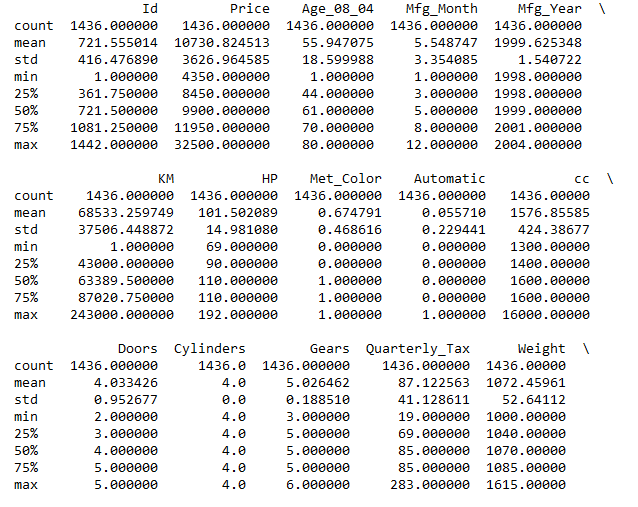
Output:

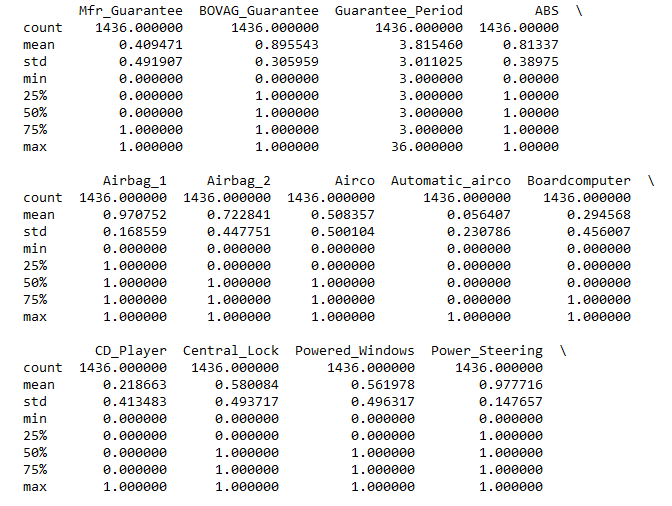


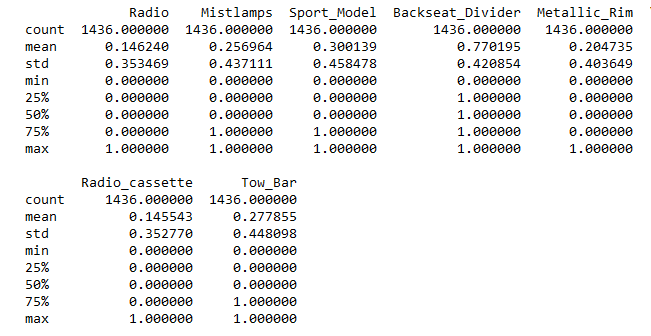
# Summary statistics of the dataset

*print(df.describe())*

Output:







# Drop unnecessary columns

*columns = ["Price", "Age\_08\_04", "KM", "HP", "cc", "Doors", "Gears", "Quarterly\_Tax", "Weight"]*

*df = df[columns]*

*print(f"Dataset reduced to {df.shape[1]} columns.")*

Output:

Dataset reduced to 9 columns.

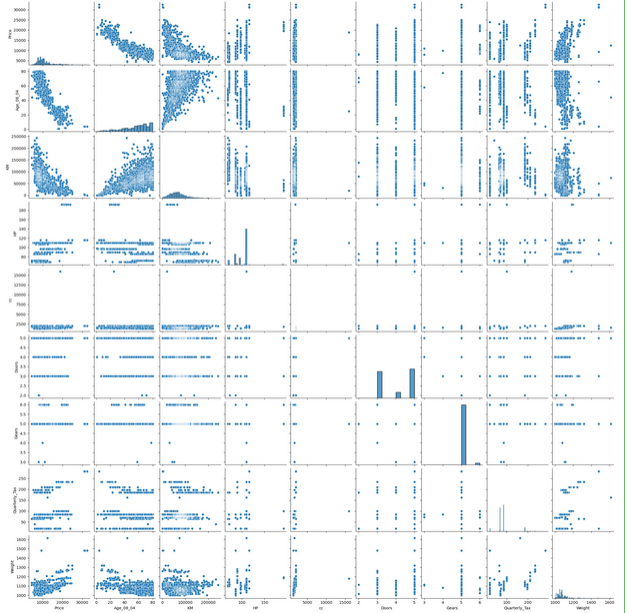
**Step 4: Exploratory Data Analysis (EDA)**

# Pairplot to visualize relationships

*sns.pairplot(df)*

*plt.show()*

Output:



# Correlation matrix

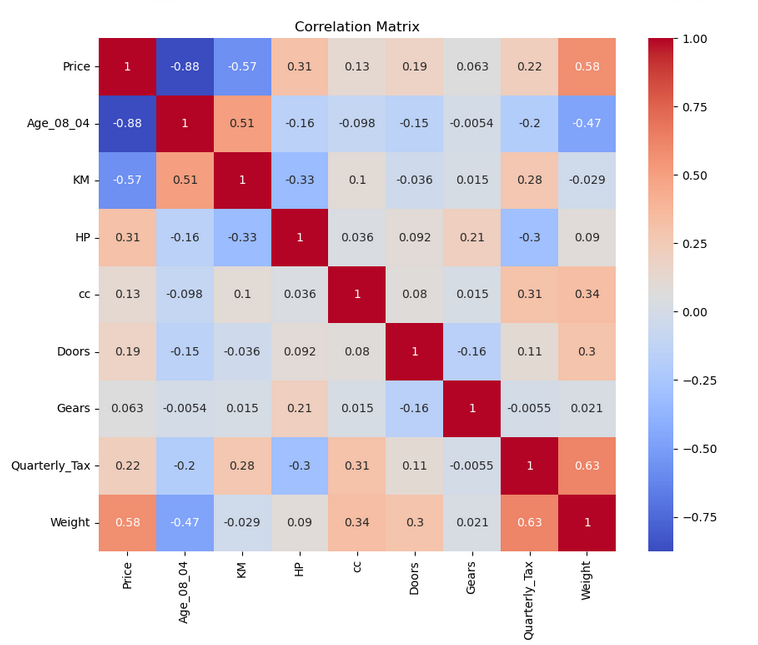
*plt.figure(figsize=(10, 8))*

*sns.heatmap(df.corr(), annot=True, cmap='coolwarm')*

*plt.title("Correlation Matrix")*

*plt.show()*

Output:



### Insights from EDA:

* **Age\_08\_04**, **KM**, and **Weight** show strong correlations with the price.
* **Doors** might not be very informative due to its weak correlation with the price.

**Step 5: Data Transformation**

# Log transformation for highly skewed data

*df['KM'] = np.log(df['KM'])*

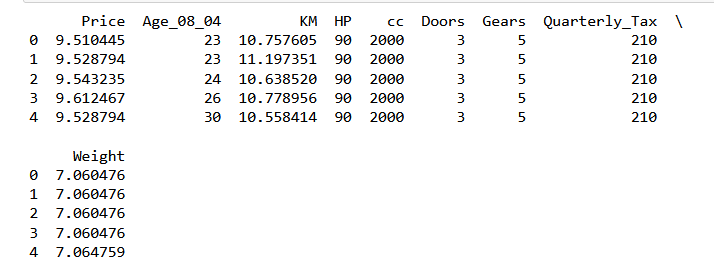
*df['Price'] = np.log(df['Price'])*

*df['Weight'] = np.log(df['Weight'])*

# Check the transformations

*print(df.head())*

Output:



**Step 6: Split Data into Training and Testing Sets**

*X = df.drop('Price', axis=1)*

*y = df['Price']*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)*

*print(f"Training set: {X\_train.shape}, Testing set: {X\_test.shape}")*

Output:

Training set: (1148, 8), Testing set: (288, 8)

**Step 7: Build Multiple Linear Regression Model**

# Add constant to the model (for intercept)

*X\_train\_const = sm.add\_constant(X\_train)*

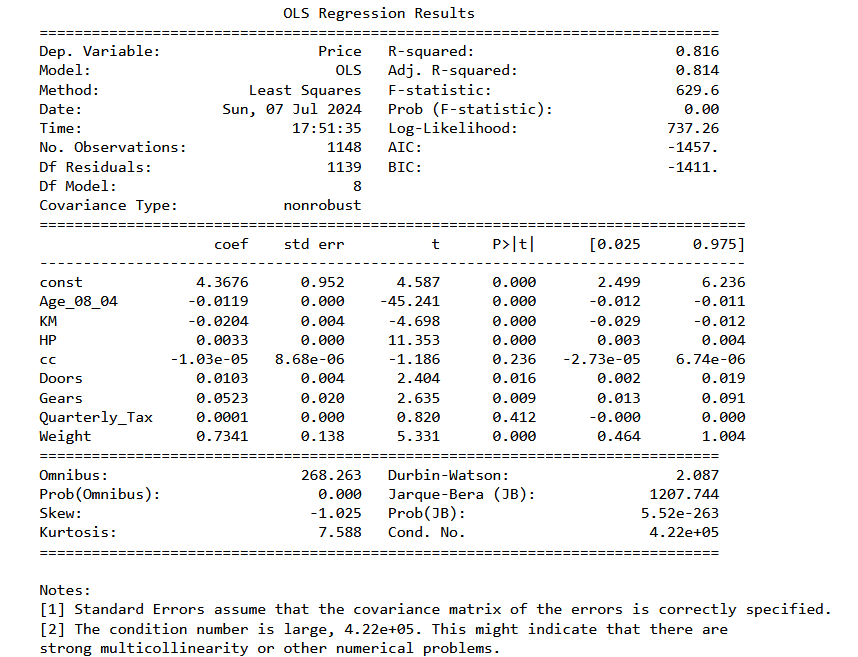
*X\_test\_const = sm.add\_constant(X\_test)*

# Fit the model

*model = sm.OLS(y\_train, X\_train\_const).fit()*

*print(model.summary())*

Output:



**Step 8: Model Evaluation**

# Predict on the test set

*y\_pred = model.predict(X\_test\_const)*

# Calculate R^2 value

*r2 = r2\_score(y\_test, y\_pred)*

*print(f"R^2 value of the model: {r2:.4f}")*

# Calculate RMSE

*rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))*

*print(f"RMSE of the model: {rmse:.4f}")*

Output:

R^2 value of the model: 0.8227

RMSE of the model: 0.1241

**Step 9: Model Improvement and Comparison**

# Try different combinations of features and transformations, for example:

# Model 1: All features

# Model 2: Exclude 'Doors'

# Model 3: Exclude 'cc'

# Define a function to fit models and print R^2 values

*def fit\_and\_evaluate(X\_train, y\_train, X\_test, y\_test):*

*X\_train\_const = sm.add\_constant(X\_train)*

*X\_test\_const = sm.add\_constant(X\_test)*

*model = sm.OLS(y\_train, X\_train\_const).fit()*

*y\_pred = model.predict(X\_test\_const)*

*r2 = r2\_score(y\_test, y\_pred)*

*rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))*

*return model, r2, rmse*

# Model 1: All features

*model1, r2\_1, rmse\_1 = fit\_and\_evaluate(X\_train, y\_train, X\_test, y\_test)*

# Model 2: Exclude 'Doors'

*X\_train\_model2 = X\_train.drop('Doors', axis=1)*

*X\_test\_model2 = X\_test.drop('Doors', axis=1)*

*model2, r2\_2, rmse\_2 = fit\_and\_evaluate(X\_train\_model2, y\_train, X\_test\_model2, y\_test)*

# Model 3: Exclude 'cc'

*X\_train\_model3 = X\_train.drop('cc', axis=1)*

*X\_test\_model3 = X\_test.drop('cc', axis=1)*

*model3, r2\_3, rmse\_3 = fit\_and\_evaluate(X\_train\_model3, y\_train, X\_test\_model3, y\_test)*

# Create a table of R^2 values and RMSE for comparison

*results = pd.DataFrame({*

*'Model': ['All Features', 'Exclude Doors', 'Exclude cc'],*

*'R^2 Value': [r2\_1, r2\_2, r2\_3],*

*'RMSE': [rmse\_1, rmse\_2, rmse\_3]*

*})*

*print(results)*

Output:

Model R^2 Value RMSE

0 All Features 0.822726 0.124069

1 Exclude Doors 0.823037 0.123961

2 Exclude cc 0.822167 0.124265

### Insights and Inferences:

* Model 1 with all features might not always be the best model due to potential multicollinearity or less informative features.
* Excluding features like 'Doors' or 'cc' might improve the model's performance.
* Always compare models using metrics like R² and RMSE to select the best model.

This workflow should give a comprehensive approach to building and selecting the best multiple linear regression model for predicting the price of Toyota Corolla cars.