```
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.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor  # Using Regression Model
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

```

```
file_path = "/content/car_price_dataset.csv"  # File Path
df = pd.read_csv(file_path)

# Display first 5 rows
print("First 5 rows of the dataset:")
print(df.head())

```

```
First 5 rows of the dataset:
         Brand   Model  Year  Engine_Size Fuel_Type    Transmission  Mileage  \
0          Kia     Rio  2020          4.2    Diesel          Manual   289944
1    Chevrolet  Malibu  2012          2.0    Hybrid       Automatic     5356
2     Mercedes     GLA  2020          4.2    Diesel       Automatic   231440
3         Audi      Q5  2023          2.0  Electric          Manual   160971
4   Volkswagen    Golf  2003          2.6    Hybrid  Semi-Automatic   286618

   Doors  Owner_Count  Price
0      3            5   8501
1      2            3  12092
2      4            2  11171
3      2            1  11780
4      3            3   2867
```

```
# Check Dataset Information
print("\nDataset Info:")
print(df.info())

print("\nChecking for Missing Values:")
print(df.isnull().sum())


print("\nDataset Statistics:")
print(df.describe())

```

```

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Brand         10000 non-null  object
 1   Model         10000 non-null  object
 2   Year          10000 non-null  int64
 3   Engine_Size   10000 non-null  float64
 4   Fuel_Type     10000 non-null  object
 5   Transmission  10000 non-null  object
 6   Mileage       10000 non-null  int64
 7   Doors         10000 non-null  int64
 8   Owner_Count   10000 non-null  int64
 9   Price         10000 non-null  int64
dtypes: float64(1), int64(5), object(4)
memory usage: 781.4+ KB
None

Checking for Missing Values:
Brand           0
Model           0
Year            0
Engine_Size     0
Fuel_Type       0
Transmission    0
Mileage         0
Doors           0
Owner_Count     0
Price           0
dtype: int64
```

```
Dataset Statistics:
                Year   Engine_Size        Mileage         Doors   Owner_Count  \
count  10000.000000  10000.000000   10000.000000  10000.000000  10000.000000
mean    2011.543700      3.000560  149239.111800      3.497100      2.991100
std        6.897699      1.149324   86322.348957      1.110097      1.422682
min     2000.000000      1.000000      25.000000      2.000000      1.000000
25%     2006.000000      2.000000   74649.250000      3.000000      2.000000
50%     2012.000000      3.000000  149587.000000      3.000000      3.000000
75%     2017.000000      4.000000  223577.500000      4.000000      4.000000
max     2023.000000      5.000000  299947.000000      5.000000      5.000000

              Price
count  10000.00000
mean    8852.96440
std     3112.59681
min     2000.00000
25%     6646.00000
50%     8858.50000
75%    11086.50000
max    18301.00000
```

```
# Separate numeric and categorical columns
numeric_cols = df.select_dtypes(include=['number']).columns
categorical_cols = df.select_dtypes(include=['object']).columns

# Fill missing values for numeric columns with mean
df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())

# Fill missing values for categorical columns with the most frequent value (mode)
df[categorical_cols] = df[categorical_cols].fillna(df[categorical_cols].mode().iloc[0])

# Check if missing values are handled
print("\nMissing values after filling:")
print(df.isnull().sum())

```

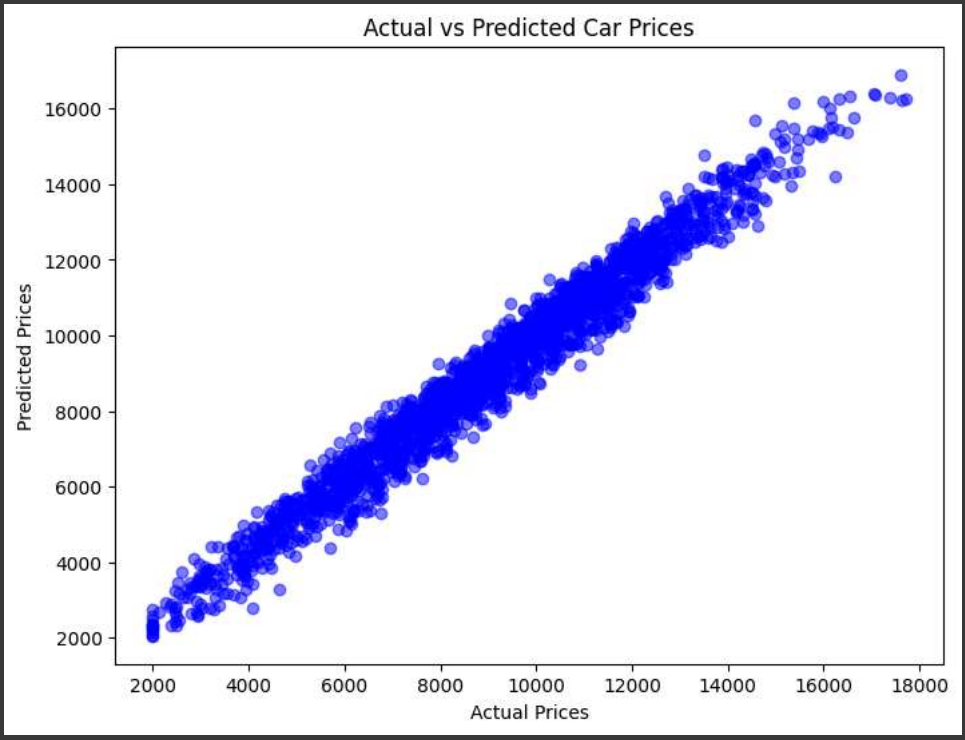
```
Missing values after filling:
Brand           0
Model           0
Year            0
Engine_Size     0
Fuel_Type       0
Transmission    0
Mileage         0
Doors           0
Owner_Count     0
Price           0
dtype: int64
```

```
df = pd.get_dummies(df, drop_first=True)  # Convert categorical variables into numeric
X = df.drop(columns=['Price'])  # Assuming 'price' is the target variable
y = df['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

```

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("\nModel Performance Metrics:")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R² Score: {r2}")

```

```
Model Performance Metrics:
Mean Absolute Error (MAE): 439.79915
Mean Squared Error (MSE): 296942.1043499
Root Mean Squared Error (RMSE): 544.9239436379172
R² Score: 0.9676808679542888
```

```
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, alpha=0.5, color='blue')
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted Car Prices")
plt.show()

```




```
import joblib

# Save Model
joblib.dump(model, 'car_price_model.pkl')

# Load Model
loaded_model = joblib.load('car_price_model.pkl')

```