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.ensemble import RandomForestRegressor

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

# Load the dataset

df = pd.read\_csv('car.csv')

# Display the first 5 rows to understand the data

print("First 5 rows of the dataset:")

print(df.head())

# Get a summary of the dataset (columns, data types, non-null counts)

print("\nDataset Information:")

df.info()

# --- 1. Feature Engineering: Create 'Car\_Age' ---

# The 'Year' column is less informative than the car's actual age.

# Let's create a 'Car\_Age' feature. We'll use 2024 as the current year for calculation.

current\_year = 2024

df['Car\_Age'] = current\_year - df['Year']

# --- 2. Data Cleaning: Drop irrelevant columns ---

# 'Car\_Name' has too many unique values (high cardinality). Including it would make the model

# too complex for this task. 'Year' is now redundant because we have 'Car\_Age'.

df.drop(columns=['Car\_Name', 'Year'], inplace=True)

# --- 3. Convert Categorical Features to Numerical ---

# We use One-Hot Encoding to convert categorical columns into numerical ones.

# This creates new columns for each category (e.g., 'Fuel\_Type\_Petrol', 'Fuel\_Type\_Diesel').

df = pd.get\_dummies(df, columns=['Fuel\_Type', 'Seller\_Type', 'Transmission'], drop\_first=True)

# Display the first 5 rows of the processed data

print("\nProcessed dataset (first 5 rows):")

print(df.head())

# Create a heatmap to visualize correlations

plt.figure(figsize=(12, 8))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Matrix of Car Features')

plt.show()

# --- 1. Define Features (X) and Target (y) ---

X = df.drop('Selling\_Price', axis=1)

y = df['Selling\_Price']

# --- 2. Split Data into Training and Testing sets ---

# 80% for training, 20% for testing

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

# --- 3. Select and Train the Model ---

# We'll use RandomForestRegressor. 'random\_state' ensures our results are reproducible.

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

# Fit the model to the training data

model.fit(X\_train, y\_train)

print("\nModel training completed!")

# --- 1. Make Predictions on the Test Set ---

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

# --- 2. Evaluate with Metrics ---

print("\n--- Model Evaluation ---")

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

print(f"R-squared (R²): {r2:.2f}")

print(f"Mean Absolute Error (MAE): {mean\_absolute\_error(y\_test, y\_pred):.2f}")

print(f"Root Mean Squared Error (RMSE): {np.sqrt(mean\_squared\_error(y\_test, y\_pred)):.2f}")

# --- 3. Visualize Predictions vs Actual Values ---

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred, alpha=0.7)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], color='red', linestyle='--')

plt.title('Actual vs. Predicted Selling Prices')

plt.xlabel('Actual Price (in Lakhs)')

plt.ylabel('Predicted Price (in Lakhs)')

plt.grid(True)

plt.show()