30.Question: Classification and Regression Trees (CART) for Car Price Prediction

You are working for a car dealership, and you want to predict the price of used cars based on

various features such as the car's mileage, age, brand, and engine type. You have collected a dataset

of used cars with their respective prices.

Write a Python program that loads the car dataset and allows the user to input the features of a new

car they want to sell. The program should use the Classification and Regression Trees (CART)

algorithm from scikit-learn to predict the price of the new car based on the input features.

The CART algorithm will create a tree-based model that will split the data into subsets based on the

chosen features and their values, leading to a decision path that eventually predicts the price of the

car. The program should output the predicted price and display the decision path (the sequence of

conditions leading to the prediction) for the new car.

Code:

import pandas as pd

from sklearn.tree import DecisionTreeRegressor, export\_text

from sklearn.preprocessing import LabelEncoder

import os

import sys

# === File path ===

file\_path = r"C:\Users\vara prasad\Downloads\car\_data.csv.xlsx"

# === Load dataset ===

if not os.path.exists(file\_path):

print(f"❌ Error: File not found at {file\_path}")

sys.exit()

try:

df = pd.read\_excel(file\_path)

except Exception as e:

print(f"❌ Error reading Excel file: {e}")

sys.exit()

# === Validate columns ===

required\_cols = ['mileage', 'age', 'brand', 'engine', 'price']

if not all(col in df.columns for col in required\_cols):

print(f"❌ Error: Dataset must contain columns: {required\_cols}")

sys.exit()

# === Encode categorical columns ===

label\_encoders = {}

for col in ['brand', 'engine']:

le = LabelEncoder()

df[col] = le.fit\_transform(df[col])

label\_encoders[col] = le

# === Features and target ===

X = df[['mileage', 'age', 'brand', 'engine']]

y = df['price']

# === Train CART model ===

model = DecisionTreeRegressor(random\_state=42)

model.fit(X, y)

# === User input ===

def get\_user\_input():

mileage = float(input("Enter mileage (in km): "))

age = int(input("Enter age (in years): "))

brand\_options = tuple(label\_encoders['brand'].classes\_)

brand = input(f"Enter brand {brand\_options}: ")

while brand not in brand\_options:

brand = input(f"❌ Invalid brand. Enter again {brand\_options}: ")

engine\_options = tuple(label\_encoders['Engine'].classes\_)

engine = input(f"Enter engine type {engine\_options}: ")

while engine not in engine\_options:

engine = input(f"❌ Invalid engine type. Enter again {engine\_options}: ")

brand\_encoded = label\_encoders['Brand'].transform([brand])[0]

engine\_encoded = label\_encoders['Engine'].transform([engine])[0]

return pd.DataFrame([[mileage, age, brand\_encoded, engine\_encoded]],

columns=['mileage', 'age', 'brand', 'engine'])

# === Predict and explain ===

user\_features = get\_user\_input()

predicted\_price = model.predict(user\_features)[0]

print(f"\n✅ Predicted price: ${predicted\_price:.2f}")

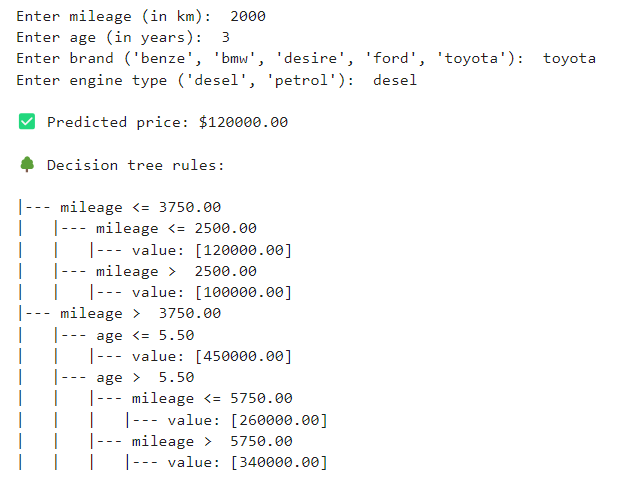
# === Display decision path ===

tree\_rules = export\_text(model, feature\_names=['mileage', 'age', 'brand', 'engine'])

print("\n🌳 Decision tree rules:\n")

print(tree\_rules)

output:



Dataset:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| mileage | age | brand | engine | price |
| 2000 | 3 | toyota | petrol | 120000 |
| 3000 | 8 | ford | petrol | 100000 |
| 4500 | 7 | bmw | petrol | 260000 |
| 7000 | 6 | benze | desel | 340000 |
| 6000 | 5 | desire | desel | 450000 |