EXPERINMENT-30

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
import numpy as np
from sklearn.tree import DecisionTreeRegressor, _tree
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
data = pd.read_csv("cars.csv")
X = data[["mileage", "age", "brand", "engine_type"]]
y = data["price"]
num_cols = ["mileage", "age"]
cat_cols = ["brand", "engine_type"]
preprocessor = ColumnTransformer(
    transformers=[
        ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols)
    ],
    remainder="passthrough"
)
model = Pipeline([
    ("pre", preprocessor),
    ("tree", DecisionTreeRegressor(max_depth=6, random_state=0))
])
model.fit(X, y)
m = float(input("Enter mileage: "))
a = float(input("Enter age: "))
```

```
b = input("Enter brand (Toyota/Honda/BMW/Ford): ")
e = input("Enter engine type (Petrol/Diesel/Hybrid): ")
new_car = pd.DataFrame([{
    "mileage": m,
    "age": a,
    "brand": b,
    "engine_type": e
}])
predicted_price = model.predict(new_car)[0]
print("\nPredicted Price:", round(predicted_price, 2))
def print_decision_path(model, row):
    pre = model.named_steps["pre"]
    tree = model.named_steps["tree"]
    X_transformed = pre.transform(row)
    node_indicator = tree.decision_path(X_transformed)
    leaf_id = tree.apply(X_transformed)
    tree_ = tree.tree_
    feature_names = []
    ohe = pre.named_transformers_["cat"]
    ohe_features = list(ohe.get_feature_names_out(cat_cols))
    feature_names = ohe_features + num_cols
    print("\nDecision Path:")
    node_index = node_indicator.indices
    for node_id in node_index:
        if tree_.feature[node_id] != _tree.TREE_UNDEFINED:
            fname = feature_names[tree_.feature[node_id]]
            threshold = tree_.threshold[node_id]
            val = X_transformed[0, tree_.feature[node_id]]
            direction = "<=" if val <= threshold else ">"
            print(f"Node {node_id}: {fname} ({val:.2f}) {direction} {threshold:.2f}")
        else:
            print(f"Leaf Node {node_id}: Predicted Price = {tree_.value[node_id][0][0]:.2f}")
print_decision_path(model, new_car)
```

Output:

```
[Running] python -u "c:\Users\karan\OneDrive\Desktop\New folder (2)\30.py"
Predicted Price: 262055.84

Decision Path:
Node 0: mileage (45000.00) <= 99722.00
Node 1: brand_BMW (0.00) <= 0.50
Node 2: age (5.00) <= 11.50
Node 3: mileage (45000.00) <= 58113.50
Node 4: age (5.00) <= 7.50
Node 5: mileage (45000.00) > 25943.50
```