```python
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeRegressor, _tree
from sklearn.preprocessing import LabelEncoder

data = pd.DataFrame({
    'mileage': [50000, 30000, 70000, 40000, 60000, 25000],
    'age': [5, 3, 7, 4, 6, 2],
    'brand': ['Toyota', 'BMW', 'Toyota', 'BMW', 'Ford', 'Ford'],
    'engine_type': ['Petrol', 'Diesel', 'Diesel', 'Petrol', 'Petrol', 'Diesel'],
    'price': [15000, 25000, 12000, 23000, 14000, 20000]
})

label_encoders = {}
for col in ['brand', 'engine_type']:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

X = data[['mileage', 'age', 'brand', 'engine_type']]
y = data['price']

model = DecisionTreeRegressor(random_state=42)
model.fit(X, y)

print("\nEnter details of the car you want to sell:")
mileage = float(input("Mileage (in km): "))
age = int(input("Age of the car (in years): "))
brand_input = input("Brand (e.g., Toyota, BMW, Ford): ")
engine_input = input("Engine Type (Petrol/Diesel): ")

if brand_input not in label_encoders['brand'].classes_ or engine_input not in label_encoders['engine_type'].classes_:
    print("Error: Invalid brand or engine type.")
    print("Available brands:", list(label_encoders['brand'].classes_))
    print("Available engine types:", list(label_encoders['engine_type'].classes_))
    exit()

brand_encoded = label_encoders['brand'].transform([brand_input])[0]
engine_encoded = label_encoders['engine_type'].transform([engine_input])[0]

new_car = np.array([[mileage, age, brand_encoded, engine_encoded]])

predicted_price = model.predict(new_car)[0]
print(f"\nPredicted Price: ${predicted_price:,.2f}")

print("\nDecision Path:")
node_indicator = model.decision_path(new_car)
tree = model.tree_
feature_names = ['mileage', 'age', 'brand', 'engine_type']

for node_id in node_indicator.indices:
    if tree.children_left[node_id] == _tree.TREE_LEAF:
        continue
    feature = feature_names[tree.feature[node_id]]
    threshold = tree.threshold[node_id]
    value = new_car[0][tree.feature[node_id]]
    if value <= threshold:
        decision = f"{feature} <= {threshold:.2f}"
    else:
        decision = f"{feature} > {threshold:.2f}"
    print(f" - Node {node id}: {decision}")
```

Output

```
Enter details of the car you want to sell:
Mileage (in km):  256
Age of the car (in years):  9
Brand (e.g., Toyota, BMW, Ford):  BMW
Engine Type (Petrol/Diesel):  Diesel

Predicted Price: $20,000.00

Decision Path:
 - Node 0: mileage <= 45000.00
 - Node 1: mileage <= 27500.00
```