21. Scenario: you are a scientist conducting research on rare elements found in a specific region. Your goal is to estimate the average concentration of a rare element in the region using a random sample of measurements. You will use the NumPy library to perform point estimation and calculate confidence intervals for the population mean.The rare element concentration data is stored in a CSV file named "rare_elements.csv," where each row contains a single measurement of the concentration.

Question: write a Python program that allows the user to input the sample size, confidence level, and desired level of precision.

CODE:

```
import pandas as pd

import numpy as np

import scipy.stats as stats

data = pd.read_csv("rare-elements.csv")

concentrations = data.iloc[:, 0].dropna().values

sample_size = int(input("Enter the sample size: "))

confidence_level = float(input("Enter the confidence level (e.g., 0.95 for 95%): "))

precision = float(input("Enter the desired level of precision: "))

np.random.seed(0)

sample = np.random.choice(concentrations, size=sample_size, replace=False)

sample_mean = np.mean(sample)

std_error = np.std(sample, ddof=1) / np.sqrt(sample_size)

z_score = stats.norm.ppf((1 + confidence_level) / 2)

margin_of_error = z_score * std_error

lower_bound = sample_mean - margin_of_error

upper_bound = sample_mean + margin_of_error

print("\nPoint Estimate (Sample Mean):", round(sample_mean, 4))

print(f"Confidence Interval ({int(confidence_level*100)}%): ({round(lower_bound, 4)},
{round(upper_bound, 4)})")

print("Margin of Error:", round(margin_of_error, 4))

print("Desired Precision Met?" , "Yes" if margin_of_error <= precision else "No")
```

OUTPUT:

```
=== RESTART: C:\Users\Mayu\OneDrive\Documents\Desktop\FOD
Enter the sample size: 5
Enter the confidence level (e.g., 0.95 for 95%): 0.94
Enter the desired level of precision: 0.2

Point Estimate (Sample Mean): 2.48
Confidence Interval (94%): (2.3703, 2.5897)
Margin of Error: 0.1097
Desired Precision Met? Yes
```

22. Scenario: Imagine you are an analyst for a popular online shopping website. Your task is to analyze customer reviews and provide insights on the average rating and customer satisfaction level for a specific product category.

Question: You will use the pandas library to calculate confidence intervals to estimate the true population mean rating.

CODE:

```
import pandas as pd

import numpy as np

from scipy import stats

df = pd.read_csv("customer_review.csv")

ratings = df['rating'].dropna()

mean_rating = ratings.mean()

std_dev = ratings.std(ddof=1)

n = len(ratings)

confidence = 0.95

alpha = 1 - confidence

z_score = stats.norm.ppf(1 - alpha / 2)

margin_of_error = z_score * (std_dev / np.sqrt(n))

ci_lower = mean_rating - margin_of_error

ci_upper = mean_rating + margin_of_error

print(f"Average Rating: {mean_rating:.2f}")

print(f"{int(confidence*100)}% Confidence Interval: ({ci_lower:.2f}, {ci_upper:.2f})")
```

OUTPUT:

```
=== RESTART: C:\Users\Mayu\OneDrive\Documen
Average Rating: 3.18
95% Confidence Interval: (2.31, 4.05)
```

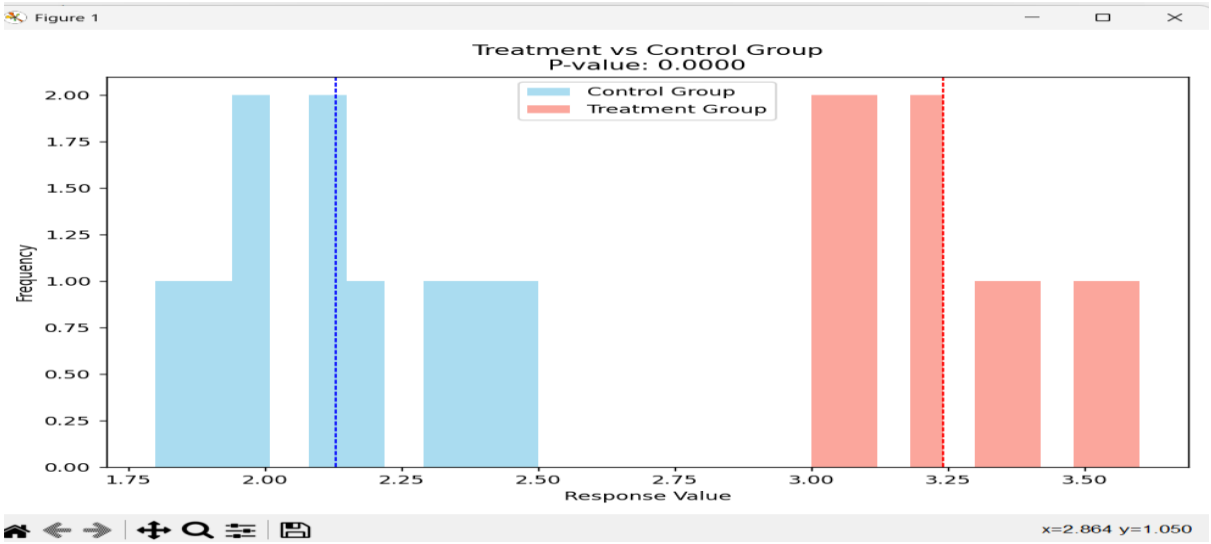
23. You are a researcher working in a medical lab, investigating the effectiveness of a new treatment for a specific disease. You have collected data from a clinical trial with two groups: a control group receiving a placebo, and a treatment group receiving the new drug.Your goal is to analyze the data using hypothesis testing and calculate the p-value to determine if the new treatment has a statistically significant effect compared to the placebo. You will use the matplotlib library to visualize the data and the p-value.

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
control = np.array([2.3, 2.1, 1.8, 2.5, 2.0, 2.2, 1.9, 2.4, 2.1, 2.0])
treatment = np.array([3.1, 3.5, 3.0, 3.4, 3.2, 3.3, 3.1, 3.6, 3.0, 3.2])
t_stat, p_value = stats.ttest_ind(treatment, control)
print(f"P-value: {p_value:.4f}")
plt.figure(figsize=(8, 6))
plt.hist(control, alpha=0.7, label='Control Group', color='skyblue', bins=10)
plt.hist(treatment, alpha=0.7, label='Treatment Group', color='salmon', bins=10)
plt.axvline(np.mean(control), color='blue', linestyle='dashed', linewidth=1)
plt.axvline(np.mean(treatment), color='red', linestyle='dashed', linewidth=1)
plt.title(f'Treatment vs Control Group\nP-value: {p_value:.4f}')
plt.xlabel('Response Value')
plt.ylabel('Frequency')
plt.legend()
plt.tight_layout()
plt.show()
```

OUTPUT:

```
= RESTART: C:\Users\M
P-value: 0.0000
```



24. Question: K-Nearest Neighbors (KNN) Classifier You are working on a classification problem to predict whether a patient has a certain medical condition or not based on their symptoms. You have collected a dataset of patients with labeled data (0 for no condition, 1 for the condition) and various symptom features. Write a Python program that allows the user to input the features of a new patient and the value of k (number of neighbors). The program should use the KNN classifier from the scikit-learn library to predict whether the patient has the medical condition or not based on the input features.

CODE:

```
import numpy as np

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split

from sklearn.datasets import make_classification

X, y = make_classification(n_samples=100, n_features=5, n_classes=2, random_state=42)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

knn = KNeighborsClassifier()

knn.fit(X_train, y_train)

print("Enter the patient's symptoms (5 features): ")

symptoms = [float(input(f"Feature {i+1}: ")) for i in range(5)]

k = int(input("Enter the value of k (number of neighbors): "))

knn.n_neighbors = k

prediction = knn.predict([symptoms])

print(f"The predicted medical condition is: {'Condition present (1)' if prediction[0] == 1 else 'No condition (0)'}")
```

OUTPUT:

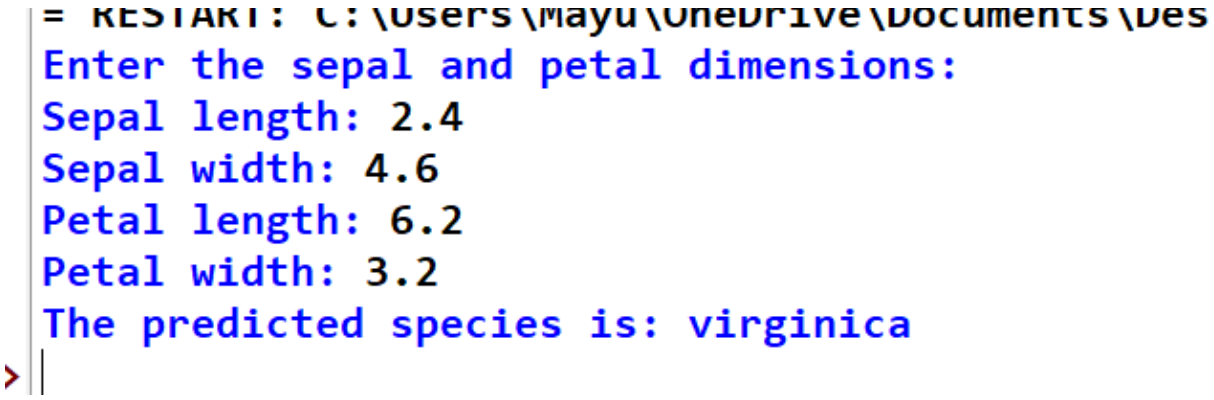
```
Enter the patient's symptoms (5 features):
Feature 1: 0.6
Feature 2: 3.5
Feature 3: 2.5
Feature 4: 1.8
Feature 5: 2.9
Enter the value of k (number of neighbors): 2
The predicted medical condition is: No condition (0)
```

25. Question 2: Decision Tree for Iris Flower Classification You are analyzing the famous Iris flower dataset to classify iris flowers into three species based on their sepal and petal dimensions. You want to use a Decision Tree classifier to accomplish this task. Write a Python program that loads the Iris dataset from scikit-learn, and allows the user to input the sepal length, sepal width, petal length, and petal width of a new flower. The program should then use the Decision Tree classifier to predict the species of the new flower.

CODE:

```
from sklearn.datasets import load_iris

from sklearn.tree import DecisionTreeClassifier

import numpy as np

iris = load_iris()

X, y = iris.data, iris.target

clf = DecisionTreeClassifier()

clf.fit(X, y)

print("Enter the sepal and petal dimensions:")

sepal_length = float(input("Sepal length: "))

sepal_width = float(input("Sepal width: "))

petal_length = float(input("Petal length: "))

petal_width = float(input("Petal width: "))

new_flower = np.array([[sepal_length, sepal_width, petal_length, petal_width]])

prediction = clf.predict(new_flower)

species = iris.target_names[prediction[0]]

print(f"The predicted species is: {species}")
```

OUTPUT:



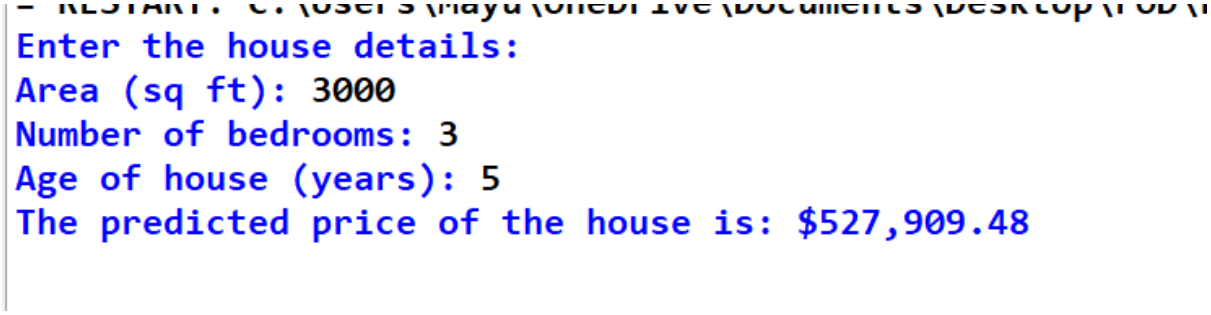
26. Question 3: Linear Regression for Housing Price Prediction You are a real estate analyst trying to predict housing prices based on various features of the houses, such as area, number of bedrooms, and location. You have collected a dataset of houses with their respective prices. Write a Python program that allows the user to input the features (area, number of bedrooms, etc.) of a new house. The program should use linear regression from scikit-learn to predict the price of the new house based on the input features.

CODE:

```
import numpy as np

import pandas as pd

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

data = {

   "area": [1400, 1600, 1700, 1875, 1100, 1550, 2350, 2450, 1425, 1700],

   "bedrooms": [3, 3, 3, 4, 2, 3, 4, 4, 2, 3],

   "age": [10, 15, 20, 18, 5, 7, 3, 6, 4, 8],

   "price": [245000, 312000, 279000, 308000, 199000, 219000, 405000, 410000, 240000, 300000]

}


df = pd.DataFrame(data)

X = df[["area", "bedrooms", "age"]]

y = df["price"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
```

```
model.fit(X_train, y_train)

print("Enter the house details:")

area = float(input("Area (sq ft): "))

bedrooms = int(input("Number of bedrooms: "))

age = int(input("Age of house (years): "))

new_house = pd.DataFrame([[area, bedrooms, age]], columns=["area", "bedrooms", "age"])

predicted_price = model.predict(new_house)

print(f"The predicted price of the house is: ${predicted_price[0]:,.2f}")
```

OUTPUT:



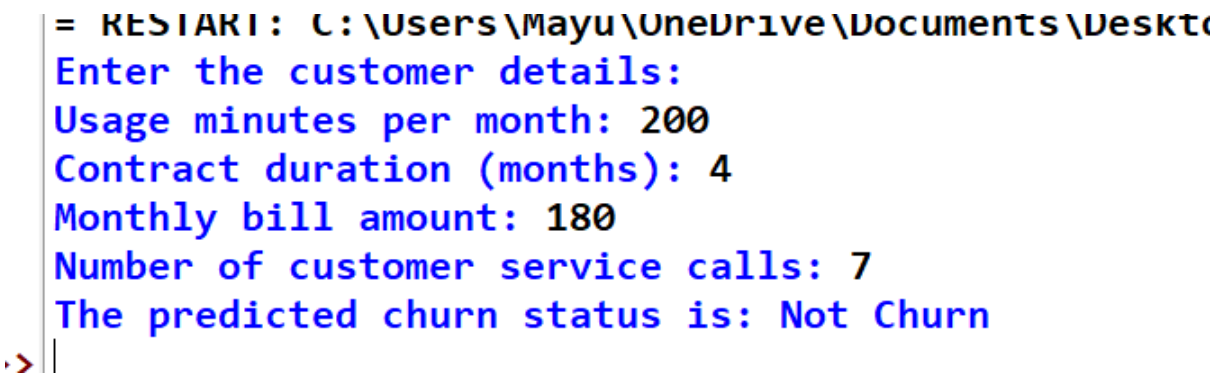
27. Question: Logistic Regression for Customer Churn Prediction You are working for a telecommunications company, and you want to predict whether a customer will churn (leave the company) based on their usage patterns and demographic data. You have collected a dataset of past customers with their churn status (0 for not churned, 1 for churned) and various features. Write a Python program that allows the user to input the features (e.g., usage minutes, contract duration) of a new customer. The program should use logistic regression from scikit-learn to predict whether the new customer will churn or not based on the input features.

CODE:

```
import numpy as np

import pandas as pd

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split

data = {

   "usage_minutes": [200, 450, 150, 600, 120, 350, 500, 300, 700, 400],

   "contract_duration": [12, 24, 6, 36, 3, 18, 30, 9, 48, 15],

   "monthly_bill": [50, 80, 30, 100, 25, 60, 90, 40, 110, 55],
```

```
    "customer_service_calls": [1, 3, 0, 4, 0, 2, 5, 1, 6, 2],
    "churn": [0, 1, 0, 1, 0, 0, 1, 0, 1, 0]
}
df = pd.DataFrame(data)
X = df[["usage_minutes", "contract_duration", "monthly_bill", "customer_service_calls"]]
y = df["churn"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
print("Enter the customer details:")
usage_minutes = float(input("Usage minutes per month: "))
contract_duration = int(input("Contract duration (months): "))
monthly_bill = float(input("Monthly bill amount: "))
customer_service_calls = int(input("Number of customer service calls: "))
new_customer = pd.DataFrame([[usage_minutes, contract_duration, monthly_bill,
customer_service_calls]],
                            columns=["usage_minutes", "contract_duration", "monthly_bill",
"customer_service_calls"])
prediction = model.predict(new_customer)
churn_status = "Churn" if prediction[0] == 1 else "Not Churn"
print(f"The predicted churn status is: {churn_status}")
```

OUTPUT:



28. Question: K-Means Clustering for Customer Segmentation You are working for an e-commerce company and want to segment your customers into distinct groups based on their purchasing behavior. You have collected a dataset of customer data with various

shopping-related features. Write a Python program that allows the user to input the shopping-related features of a new customer. The program should use K-Means clustering from scikit-learn to assign the new customer to one of the existing segments based on the input features.

CODE:

```
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
data = {
    'Annual Spending': [500, 1500, 2500, 800, 2200, 3000],
    'Purchase Frequency': [20, 50, 90, 25, 85, 120],
    'Product Diversity': [5, 15, 25, 6, 20, 30]
}
df = pd.DataFrame(data)
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
num_clusters = 3
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
df['Segment'] = kmeans.fit_predict(scaled_data)
def classify_new_customer():
    print("Enter the shopping features of the new customer:")
    annual_spending = float(input("Annual Spending: "))
    purchase_frequency = float(input("Purchase Frequency: "))
    product_diversity = float(input("Product Diversity: "))
    new_data = pd.DataFrame([[annual_spending, purchase_frequency, product_diversity]],
                            columns=df.columns[:-1])
    new_data_scaled = scaler.transform(new_data)
    segment = kmeans.predict(new_data_scaled)[0]
    print(f"The new customer belongs to Segment {segment}")
```

classify_new_customer()

OUTPUT:

```
= RESTART: C:\Users\Mayu\OneDrive\Documents\Desktop\FOD
Enter the shopping features of the new customer:
Annual Spending: 1800
Purchase Frequency: 45
Product Diversity: 20
The new customer belongs to Segment 2
>
```

29.Question: Evaluation Metrics for Model Performance You have trained a machine learning model on a dataset, and now you want to evaluate its performance using various metrics. Write a Python program that loads a dataset and trained model from scikit-learn. The program should ask the user to input the names of the features and the target variable they want to use for evaluation. The program should then calculate and display common evaluation metrics such as accuracy, precision, recall, and F1-score for the model's predictions on the test data.

CODE:

```
from sklearn.datasets import load_breast_cancer

from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

import pandas as pd

data = load_breast_cancer()

df = pd.DataFrame(data.data, columns=data.feature_names)

df['target'] = data.target

print("\nAvailable features:")

print(df.columns.tolist())

features_input = input("\nEnter feature names separated by commas: ")

target_input = input("Enter target variable name (e.g., 'target'): ")

features = [f.strip() for f in features_input.split(",")]

target = target_input.strip()

X = df[features]

y = df[target]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier(random_state=42)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("\nModel Evaluation Metrics:")

print(f"Accuracy:  {accuracy_score(y_test, y_pred):.4f}")

print(f"Precision: {precision_score(y_test, y_pred):.4f}")

print(f"Recall:    {recall_score(y_test, y_pred):.4f}")

print(f"F1 Score:  {f1_score(y_test, y_pred):.4f}")
```

OUTPUT:

```
Available features:
['mean radius', 'mean texture', 'mean perimeter', 'mean area', 'mea
n smoothness', 'mean compactness', 'mean concavity', 'mean concave
points', 'mean symmetry', 'mean fractal dimension', 'radius error',
'texture error', 'perimeter error', 'area error', 'smoothness error
', 'compactness error', 'concavity error', 'concave points error',
'symmetry error', 'fractal dimension error', 'worst radius', 'worst
texture', 'worst perimeter', 'worst area', 'worst smoothness', 'wor
st compactness', 'worst concavity', 'worst concave points', 'worst
symmetry', 'worst fractal dimension', 'target']

Enter feature names separated by commas: mean radius,radius error,s
ymmetry error
Enter target variable name (e.g., 'target'): target

Model Evaluation Metrics:
Accuracy:  0.9211
Precision: 0.9189
Recall:    0.9577
F1 Score:  0.9379
```

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
from sklearn.model_selection import train_test_split
data = {
   'mileage': [10000, 50000, 30000, 70000, 25000],
   'age': [1, 5, 3, 7, 2],
   'brand': ['Toyota', 'BMW', 'Ford', 'BMW', 'Toyota'],
   'engine_type': ['Petrol', 'Diesel', 'Diesel', 'Petrol', 'Electric'],
   'price': [20000, 15000, 18000, 12000, 22000]
}
df = pd.DataFrame(data)
le_brand = LabelEncoder()
le_engine = LabelEncoder()
df['brand'] = le_brand.fit_transform(df['brand'])
df['engine_type'] = le_engine.fit_transform(df['engine_type'])
X = df[['mileage', 'age', 'brand', 'engine_type']]
y = df['price']
model = DecisionTreeRegressor()
model.fit(X, y)
print("\nEnter the new car details:")
mileage = int(input("Mileage (e.g., 30000): "))
age = int(input("Age in years (e.g., 2): "))
print(f"Available brands: {list(le_brand.classes_)}")
brand_input = input("Brand: ")
brand_encoded = le_brand.transform([brand_input])[0]
print(f"Available engine types: {list(le_engine.classes_)}")
```

```
engine_input = input("Engine Type: ")
engine_encoded = le_engine.transform([engine_input])[0]
new_car = pd.DataFrame([[mileage, age, brand_encoded, engine_encoded]],
                       columns=['mileage', 'age', 'brand', 'engine_type'])
predicted_price = model.predict(new_car)[0]
print(f"\nPredicted Price: ${predicted_price:.2f}")
tree_rules = export_text(model, feature_names=['mileage', 'age', 'brand', 'engine_type'])
print("\nDecision Path:")
print(tree_rules)
```

OUTPUT:

```
Enter the new car details:
Mileage (e.g., 30000): 28000
Age in years (e.g., 2): 4
Available brands: ['BMW', 'Ford', 'Toyota']
Brand: Ford
Available engine types: ['Diesel', 'Electric', 'Petrol']
Engine Type: Diesel

Predicted Price: $18000.00

Decision Path:
|--- mileage <= 40000.00
|   |--- brand <= 1.50
|   |   |--- value: [18000.00]
|   |--- brand >  1.50
|   |   |--- mileage <= 17500.00
|   |   |   |--- value: [20000.00]
|   |   |--- mileage >  17500.00
|   |   |   |--- value: [22000.00]
|--- mileage >  40000.00
|   |--- engine_type <= 1.00
|   |   |--- value: [15000.00]
|   |--- engine_type >  1.00
|   |   |--- value: [12000.00]
```