BMS Institute of Technology and Management

(An Autonomous Institution, Affiliated to VTU, Belagavi)

Department of Master of Computer Applications

(Accredited by NBA, New Delhi)

Alternate Assessment Tool (AAT)

USN	1BY22MC027	Student Name	Manjunath Raju S G
Course Code	22MCA301	Course Title	Machine Learning
Semester	3rd	Academic Year	2023-2024
Date of Submission	30/01/2024	Number of Pages Submitted	03

AAT Question or Topic or Problem Statement

Implement Decision Tree Algorithm (ID3) using Python for the given Dataset

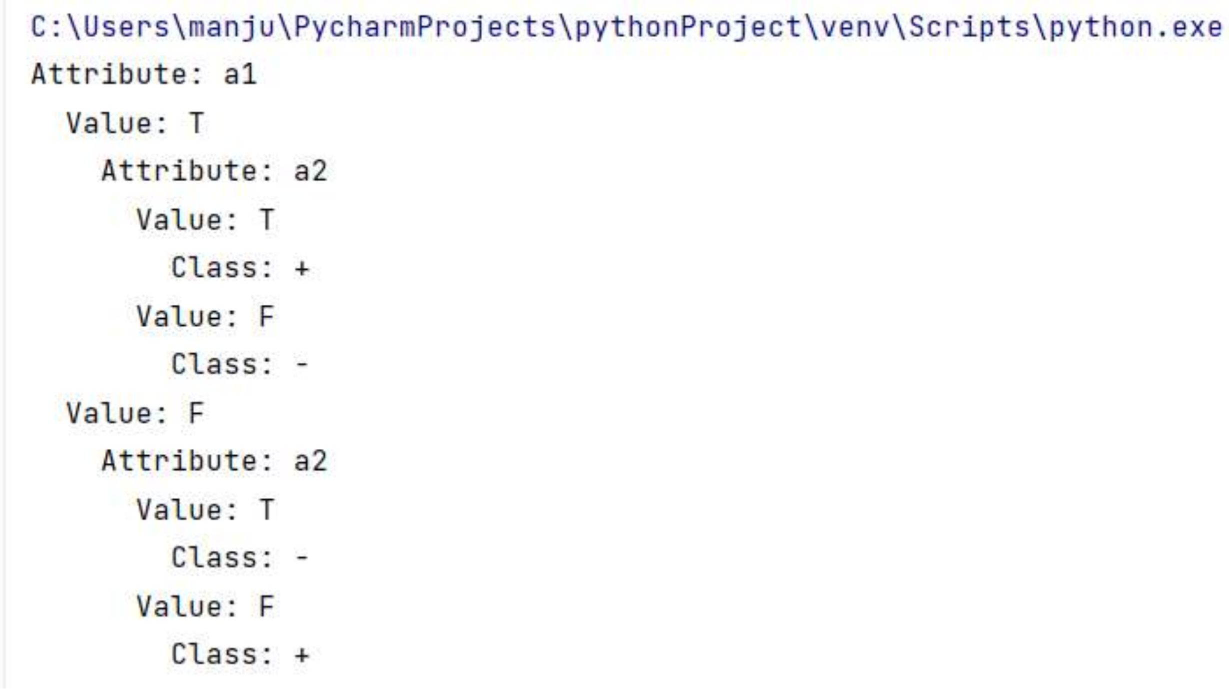
Solution / Answer

Data-Set

Instance	Classification	a1	a2
1	+	T	T
2	+	T	T
3	-	T	F
4	+	F	F
5	-	F	T
6	-	F	T

```
from graphviz import Digraph


class Node:
    def __init__(self, data=None, attribute=None, branches=None, classification=None):
        self.data = data  # Training data at this node
        self.attribute = attribute  # Attribute to split on
        self.branches = branches  # Subtrees (child nodes)
        self.classification = classification  # Class label (if leaf node)


def calculate_entropy(labels):
    from math import log2

    unique_labels = set(labels)
    entropy = 0.0

    for label in unique_labels:
        probability = list(labels).count(label) / len(labels)  # Convert set to list before counting
        entropy -= probability * log2(probability)

    return entropy


def id3(data, attributes, target_attribute):
    if len(set(data[target_attribute])) == 1:
        # If all instances have the same classification, return a leaf node
        return Node(classification=data[target_attribute].iloc[0])

    if len(attributes) == 0:
        # If no attributes left, return a leaf node with the majority class
        majority_class = data[target_attribute].mode()[0]
        return Node(classification=majority_class)

    # Calculate the entropy of the current data
    current_entropy = calculate_entropy(data[target_attribute])
    # Find the attribute with the highest information gain
    max_info_gain = 0
    best_attribute = None

    for attribute in attributes:
        values = set(data[attribute])
        new_entropy = 0.0

        for value in values:
            subset = data[data[attribute] == value]
            subset_entropy = calculate_entropy(subset[target_attribute])
            new_entropy += len(subset) / len(data) * subset_entropy

        info_gain = current_entropy - new_entropy

        if info_gain > max_info_gain:
            max_info_gain = info_gain
            best_attribute = attribute

    if best_attribute is None:
        # If no attribute provides information gain, return a leaf node with the majority class
        majority_class = data[target_attribute].mode()[0]
        return Node(classification=majority_class)

    # Split the data based on the best attribute
    branches = {}
    values = set(data[best_attribute])

    for value in values:
        subset = data[data[best_attribute] == value]
        branches[value] = id3(subset, attributes - {best_attribute}, target_attribute)

    return Node(attribute=best_attribute, branches=branches)


def visualize_tree(node, dot=None, parent_value=None):
    if dot is None:
        dot = Digraph(comment='Decision Tree')

    if node.classification is not None:
        dot.node(str(id(node)), f'Class: {node.classification}\n(for Value: {parent_value})')
    else:
        if parent_value is not None:
            dot.node(str(id(node)), f'Value: {parent_value}\nAttribute: {node.attribute}')
        else:
            dot.node(str(id(node)), f'Attribute: {node.attribute}')

        for value, branch in node.branches.items():
            visualize_tree(branch, dot, value)
            dot.edge(str(id(node)), str(id(branch)), label=str(value))
    return dot


# Example usage
import pandas as pd

# Creating a DataFrame from the provided data
data = {
    'Instance': [1, 2, 3, 4, 5, 6],
    'Classification': ['+', '+', '-', '+', '-', '-'],
    'a1': ['T', 'T', 'T', 'F', 'F', 'F'],
    'a2': ['T', 'T', 'F', 'F', 'T', 'T']
}

df = pd.DataFrame(data)

# Specify the target attribute (Classification) and the attributes to consider for splitting
target_attribute = 'Classification'
attributes = {'a1', 'a2'}

# Build the decision tree
root_node = id3(df, attributes, target_attribute)

# Visualize the decision tree using graphviz
dot = visualize_tree(root_node)
dot.render('decision_tree', format='png', cleanup=True)
```

Out-put:



```
C:\Users\manju\PycharmProjects\pythonProject\venv\Scripts\python.exe
Attribute: a1
  Value: T
    Attribute: a2
      Value: T
        Class: +
      Value: F
        Class: -
  Value: F
    Attribute: a2
      Value: T
        Class: -
      Value: F
        Class: +
```