Machine Learning

Assignment 8: Decision Tree

Code:

```
import pandas as pd

import numpy as np

import math


# Define the dataset

data = {'A1': [True, True, False, False, False, True, True, True, False, False],

    'A2': ['Hot', 'Hot', 'Hot', 'Cool', 'Cool', 'Cool', 'Hot', 'Hot', 'Cool', 'Cool'],

    'A3': ['High', 'High', 'High', 'Normal', 'Normal', 'High', 'High', 'Normal', 'Normal', 'High'],

    'label': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'No', 'Yes', 'Yes', 'Yes']}

df = pd.DataFrame(data)

print(df)


# Define a function to calculate entropy

def entropy(labels):

  _, counts = np.unique(labels, return_counts=True)

  probabilities = counts / len(labels)

  entropy_val = sum([-p * math.log2(p) for p in probabilities])

  print(entropy_val)

  return entropy_val



# Define a function to calculate information gain

def information_gain(data, split_attribute_name, target_attribute_name):

  total_entropy = entropy(data[target_attribute_name])

  _, counts = np.unique(data[split_attribute_name], return_counts=True)

  probabilities = counts / len(data[split_attribute_name])
```

```
    weighted_entropy = sum(probabilities * \
        [entropy(data.where(data[split_attribute_name]==value).dropna()[target_attribute_name]) \
         for value in np.unique(data[split_attribute_name])])
    information_gain = total_entropy - weighted_entropy
    return information_gain
    print(total_entropy)

# Define a function to get the best attribute for splitting
def get_best_attribute(data, target_attribute_name):
    information_gains = [(attribute, information_gain(data, attribute, target_attribute_name)) \
                         for attribute in data.columns if attribute != target_attribute_name]
    best_attribute = max(information_gains, key=lambda x: x[1])[0]
    return best_attribute
# Define a recursive function to build the decision tree
def build_decision_tree(data, target_attribute_name, default_class=None):
    # If all the instances belong to the same class, return that class
    if len(np.unique(data[target_attribute_name])) == 1:
        return np.unique(data[target_attribute_name])[0]
    # If the dataset is empty, return the default class
    elif len(data) == 0:
        return default_class
    # If there are no more attributes to split, return the most common class
    elif len(data.columns) == 1:
        return default_class
    else:
        # Get the best attribute for splitting
        best_attribute = get_best_attribute(data, target_attribute_name)
        # Create a new decision tree node with the best attribute
        tree = {best_attribute: {}}
        # Get the unique values of the best attribute
        unique_values = np.unique(data[best_attribute])
```

```
        # Recursively build the subtree for each unique value
        for value in unique_values:
            subtree = build_decision_tree(data.where(data[best_attribute] == value).dropna(),
                                          target_attribute_name, default_class)
            # Add the subtree to the tree
            tree[best_attribute][value] = subtree
        return tree
# Build the decision tree
tree = build_decision_tree(df, 'label')
print(tree)
# Define a function to make predictions using the decision tree
def predict(instance, tree, default_class=None):
    attribute = next(iter(tree))
    if instance[attribute] in tree[attribute]:
        result = tree[attribute][instance[attribute]]
        print(tree)
```

Output:

```
=========== RESTART: C:\Users\harsh\OneDrive\Desktop\DecisionTree.py ==
      A1    A2      A3 label
0   True   Hot    High    No
1   True   Hot    High    No
2  False   Hot    High   Yes
3  False  Cool  Normal   Yes
4  False  Cool  Normal   Yes
5   True  Cool    High    No
6   True   Hot    High    No
7   True   Hot  Normal   Yes
8  False  Cool  Normal   Yes
9  False  Cool    High   Yes
0.9709505944546686
0.0
0.7219280948873623
0.9709505944546686
0.7219280948873623
0.9709505944546686
0.9709505944546686
0.9182958340544896
0.0
0.7219280948873623
0.7219280948873623
0.7219280948873623
0.0
0.8112781244591328
0.7219280948873623
0.0
0.0
{'A1': {False: 'Yes', True: {'A3': {'High': 'No', 'Normal': 'Yes'}}}}
```

Submitted By:

Harshita Pasupuleti

21BCE8421