9913_exp5

August 30, 2024

[ ]:
```
import pandas as pd
```

[ ]:
```
Data_Frame = pd.read_csv('./PlayTennis.csv')
```

[ ]:
```
import numpy as np
from collections import Counter

# Function to calculate entropy
def entropy(y):
    counts = np.bincount(y)
    probabilities = counts / len(y)
    return -np.sum([p * np.log2(p) for p in probabilities if p > 0])

# Function to calculate information gain
def information_gain(X, y, feature_index):
    # Calculate entropy before the split
    entropy_before = entropy(y)

    # Split data based on the feature
    values = np.unique(X[:, feature_index])
    entropy_after = 0

    for value in values:
        subset_y = y[X[:, feature_index] == value]
        entropy_after += (len(subset_y) / len(y)) * entropy(subset_y)

    # Information gain is the reduction in entropy
    return entropy_before - entropy_after

# Function to find the best feature to split on
def best_feature_to_split(X, y):
    best_gain = 0
    best_feature = None

    for feature_index in range(X.shape[1]):
        gain = information_gain(X, y, feature_index)
        if gain > best_gain:
            best_gain = gain
```

```
            best_feature = feature_index

    return best_feature

# Function to check for homogeneity
def is_homogeneous(y):
    return len(np.unique(y)) == 1

# ID3 Algorithm
def id3(X, y, feature_names):
    # Step 2: Check for homogeneity
    if is_homogeneous(y):
        return y[0]

    # Step 8: Stopping criteria - No more features to split on
    if len(feature_names) == 0:
        return Counter(y).most_common(1)[0][0]

    # Step 5: Select the best feature
    best_feature_index = best_feature_to_split(X, y)
    best_feature_name = feature_names[best_feature_index]

    # Create the tree structure
    tree = {best_feature_name: {}}

    # Step 6: Split the dataset
    feature_values = np.unique(X[:, best_feature_index])
    for value in feature_values:
        subset_X = X[X[:, best_feature_index] == value]
        subset_y = y[X[:, best_feature_index] == value]

        # Remove the best feature from the feature list
        new_feature_names = feature_names[:best_feature_index] +␣
 ↪feature_names[best_feature_index+1:]

        # Step 7: Repeat the process for subsets
        subtree = id3(subset_X, subset_y, new_feature_names)
        tree[best_feature_name][value] = subtree
        return tree
```

[ ]:
```
print(Data_Frame)
```

```
     Outlook Temperature Humidity    Wind Play Tennis
0      Sunny         Hot     High    Weak          No
1      Sunny         Hot     High  Strong          No
2   Overcast         Hot     High    Weak         Yes
3       Rain        Mild     High    Weak         Yes
4       Rain        Cool   Normal    Weak         Yes
```

```
5       Rain        Cool   Normal  Strong          No
6   Overcast        Cool   Normal  Strong         Yes
7      Sunny        Mild     High    Weak          No
8      Sunny        Cool   Normal    Weak         Yes
9       Rain        Mild   Normal    Weak         Yes
10     Sunny        Mild   Normal  Strong         Yes
11  Overcast        Mild     High  Strong         Yes
12  Overcast         Hot   Normal    Weak         Yes
13      Rain        Mild     High  Strong          No
```

[ ]:
```python
# Example usage
if __name__ == "__main__":
    # Example dataset

    df = pd.DataFrame(Data_Frame)

    # Convert categorical data to numerical
    for column in df.columns:
        df[column] = df[column].astype('category').cat.codes

    X = df.drop(columns='Play Tennis').values
    y = df['Play Tennis'].values
    feature_names = df.drop(columns='Play Tennis').columns.tolist()
    print(feature_names)

    # Build the decision tree
    decision_tree = id3(X, y, feature_names)
    print("Decision Tree:", decision_tree)
```

```
['Outlook', 'Temperature', 'Humidity', 'Wind']
Decision Tree: {'Outlook': {0: 1}}
```

[ ]:
```python
    X = df.drop(columns='Temperature').values
    y = df['Temperature'].values
    feature_names = df.drop(columns='Temperature').columns.tolist()
    print(feature_names)
    # Build the decision tree
    decision_tree = id3(X, y, feature_names[1])
    print("Decision Tree:", decision_tree)
```

```
['Outlook', 'Humidity', 'Wind', 'Play Tennis']
Decision Tree: {'u': {0: {'H': {0: {'d': {0: 2}}}}}}
```

[ ]:
```python
    X = df.drop(columns='Outlook').values
    y = df['Outlook'].values
    feature_names = df.drop(columns='Outlook').columns.tolist()
    print(feature_names)
    # Build the decision tree
```

```
    decision_tree = id3(X, y, feature_names[2])
    print("Decision Tree:", decision_tree)
```

```
['Temperature', 'Humidity', 'Wind', 'Play Tennis']
Decision Tree: {'d': {0: {'W': {0: 1}}}}
```