midterm-21127229

March 15, 2024

Họ tên: Dương Trường Bình

MSSV: 21127229

1 Câu 1:

Đưa ra 5 biến thuộc tính và 1 biến quyết định có ý nghĩa

#	x_1	x_2	x_3	x_4	x_5	y
kiểu	nhị phân	tam phân	tứ phân	tứ phân	số thực	nhị phân

Đề xuất bài toán phân loại có phải người châu Á hay không - 5 biến thuộc tính: - Biến kiểu nhị phân: Giới tính (age): (Male, Female) - Biến kiểu tam phân: Chiều cao (height): (Short, Medium, Tall) - Biến kiểu tứ phân: Màu tóc (color_hair): (Black, Brown, Blonde, Red) - Biến kiểu tứ phân: Màu mắt (color_eye): (Brown, Blue, Green, Black) - Biến kiểu số thực: Cân nặng (weight)

- 1 biến quyết định nhị phân: Châu Á (is_Asian): (Yes, No)

2 Câu 2:

Đưa ra 5 luật có ý nghĩa X $\rightarrow$ y

- `Gender` = `Female` && `Height` = `Short` && `Color_hair` = `Black` -> `is_Asian` = `Yes`
- `Gender` = `Female` && `Height` = `Tall` && `Color_hair` = `Blonde` -> `is_Asian` = `No`
- `Gender` = `Male` && `Height` = `Medium` && `Color_hair` = `Black` -> `is_Asian` = `Yes`
- `Gender` = `Male` && `Height` = `Medium` && `Color_hair` = `Blonde` -> `is_Asian` = `No`
- `Gender` = `Male` && `Height` = `Short` && `Color_hair` = `Black` -> `is_Asian` = `Yes`

3 Câu 3:

Tạo tập dữ liệu huấn luyện (train.csv) dạng bảng có 1000 dòng từ 5 luật sao cho 80% là đồng ý với luật và 20% là không đồng ý

[2]:
```
import numpy as np
import pandas as pd
import random
```

```
import matplotlib.pyplot as plt
```

[3]:
```
def generate_data(num_samples):
    """
    Generates a dataset with attributes and labels based on given rules.

    Args:
        num_samples: Number of data points to generate.

    Returns:
        A list of dictionaries, where each dictionary represents a data point.
    """

    data = []
    # 80% data following rules

    for _ in range(int(0.8 * num_samples)):
    # Randomly choose a valid combination based on rules
        choice = random.choice([1, 2, 3, 4, 5])
        if choice == 1:
            data.append({
                "Gender": "Female",
                "Height": "Short",
                "Color_hair": "Black",
                "Color_eye": random.choice(["Black", "Brown", "Blue", "Gray"]),
                "Weight": random.uniform(40, 150),
                "is_Asian": "Yes"
            })
        elif choice == 2:
            data.append({
                "Gender": "Female",
                "Height": "Tall",
                "Color_hair": "Blonde",
                "Color_eye": random.choice(["Black", "Brown", "Blue", "Gray"]),
                "Weight": random.uniform(40,150),
                "is_Asian": "No"
            })
        elif choice == 3:
            data.append({
                "Gender": "Male",
                "Height": "Medium",
                "Color_hair": "Black",
                "Color_eye": random.choice(["Black", "Brown", "Blue", "Gray"]),
                "Weight": random.uniform(40,150),
                "is_Asian": "Yes"
            })
        elif choice == 4:
```

```
            data.append({
                "Gender": "Male",
                "Height": "Medium",
                "Color_hair": "Blonde",
                "Color_eye": random.choice(["Black", "Brown", "Blue", "Gray"]),
                "Weight": random.uniform(40,150),
                "is_Asian": "No"
            })
        elif choice == 5:
            data.append({
                "Gender": "Male",
                "Height": "Short",
                "Color_hair": "Black",
                "Color_eye": random.choice(["Black", "Brown", "Blue", "Gray"]),
                "Weight": random.uniform(40,150),
                "is_Asian": "Yes"
            })


    # 20% wrong data
    for _ in range(int(0.2 * num_samples)):
        choice = random.choice([1, 2, 3, 4,5])
        if choice == 1:
            data.append({
                "Gender": "Female",
                "Height": "Tall",
                "Color_hair": "Blonde",
                "Color_eye": random.choice(["Black", "Brown", "Blue", "Gray"]),
                "Weight": random.uniform(40,150),
                "is_Asian": "Yes"
            })
        elif choice == 2:
            data.append({
                "Gender": "Female",
                "Height": "Short",
                "Color_hair": "Black",
                "Color_eye": random.choice(["Black", "Brown", "Blue", "Gray"]),
                "Weight": random.uniform(40,150),
                "is_Asian": "No"
            })
        elif choice == 3:
            data.append({
                "Gender": "Male",
                "Height": "Medium",
                "Color_hair": "Black",
                "Color_eye": random.choice(["Black", "Brown", "Blue", "Gray"]),
                "Weight": random.uniform(40,150),
```

```
                "is_Asian": "No"
            })
        elif choice == 4:
            data.append({
                "Gender": "Male",
                "Height": "Tall",
                "Color_hair": "Blonde",
                "Color_eye": random.choice(["Black", "Brown", "Blue", "Gray"]),
                "Weight": random.uniform(40,150),
                "is_Asian": "Yes"
            })
        elif choice == 5:
            data.append({
                "Gender": "Male",
                "Height": "Short",
                "Color_hair": "Black",
                "Color_eye": random.choice(["Black", "Brown", "Blue", "Gray"]),
                "Weight": random.uniform(40,150),
                "is_Asian": "No"
            })

    return data

# Example usage:
num_data_points = 1000
train_data = generate_data(num_data_points)
test_data = generate_data(num_data_points)
train_df = pd.DataFrame(train_data)
test_df = pd.DataFrame(test_data)
```

[4]:
```
train_df.to_csv('train.csv', index=False)
test_df.to_csv('test.csv', index=False)
train_df.head()
```

[4]:

	Gender	Height	Color_hair	Color_eye	Weight	is_Asian
0	Male	Short	Black	Brown	122.828078	Yes
1	Female	Tall	Blonde	Blue	51.887998	No
2	Male	Medium	Blonde	Brown	45.331389	No
3	Male	Short	Black	Black	46.140982	Yes
4	Male	Short	Black	Gray	111.029268	Yes

4 Câu 4:

Huấn luyện mô hình cây quyết định dựa trên (train.csv), trực quan và rút ra tối đa 5 luật tốt nhất

4.0.1 Huấn luyện mô hình cây quyết định

[5]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder, LabelEncoder


# Read data
train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')

# Split data
X_train, y_train = train_df.drop(columns=['is_Asian']), train_df['is_Asian']
X_test, y_test = test_df.drop(columns=['is_Asian']), test_df['is_Asian']

X = pd.get_dummies(pd.concat([X_train, X_test]))
X_train = X.iloc[:len(X_train)]
X_test = X.iloc[len(X_train):]

# Train a model
model = DecisionTreeClassifier(random_state=0)
model.fit(X_train, y_train)
```

[5]:
```
DecisionTreeClassifier(random_state=0)
```

4.0.2 Trực quan cây quyết định

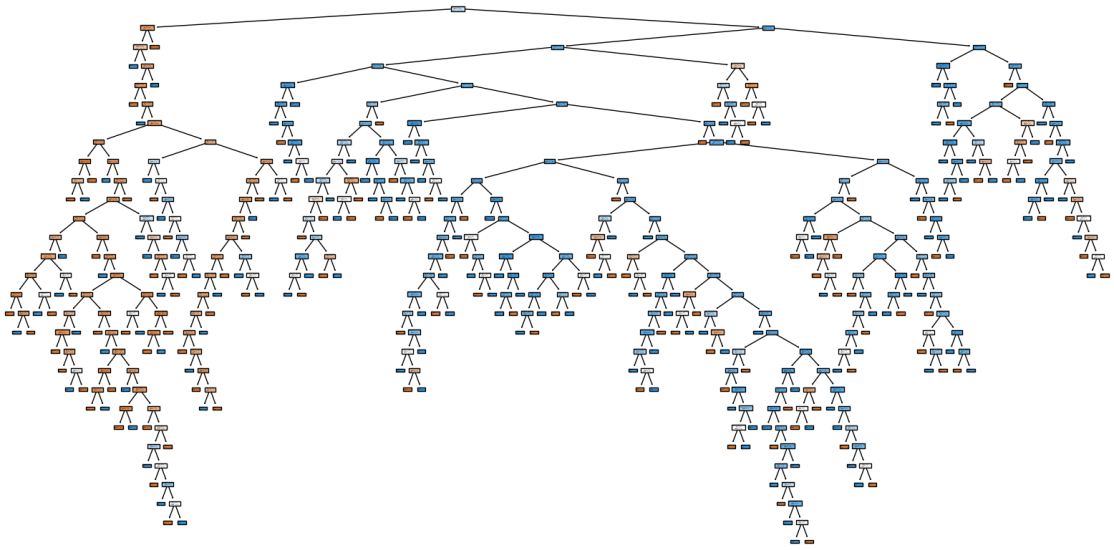
[6]:
```
# trực quan hóa cây quyết định
from sklearn.tree import plot_tree

plt.figure(figsize=(20, 10))

# Get the feature names from the OneHotEncoder
feature_names = list(X.columns)

plot_tree(model, filled=True, feature_names=feature_names)

plt.show()
```



[7]:
```
path = model.decision_path(X_train)

# Get the leaf nodes ids
leaf_ids = model.apply(X_train)

# Initialize a dictionary to hold support for each leaf node
leaf_support = {}
total_samples = len(X_train)
leaf_support = {}
leaf_confidence = {}

for leaf_id in np.unique(leaf_ids):
    # Tính support
    support_count = np.sum(leaf_ids == leaf_id)
    leaf_support[leaf_id] = support_count / total_samples

    # Tính confidence
    class_distribution = model.tree_.value[leaf_id][0]
    majority_class_count = np.max(class_distribution)
    leaf_confidence[leaf_id] = majority_class_count / support_count

# Lọc ra các leaf node có confidence bằng 1
perfect_confidence_leaf_ids = [leaf_id for leaf_id in leaf_confidence if␣
 ↪leaf_confidence[leaf_id] == 1]

# Sắp xếp các leaf node này dựa trên support và chọn ra top 5
top_5_leaf_ids = sorted(perfect_confidence_leaf_ids, key=lambda x:␣
 ↪leaf_support[x], reverse=True)[:5]
```

```
# Xuất ra thông tin của top 5 leaf nodes
top_5_nodes_info = [(leaf_id, leaf_support[leaf_id], leaf_confidence[leaf_id])␣
 ↪for leaf_id in top_5_leaf_ids]
print("Top 5 leaf nodes information:")
for node_info in top_5_nodes_info:
    print(f"Leaf node ID: {node_info[0]}, Support: {node_info[1]}, Confidence:␣
 ↪{node_info[2]}")

# Định nghĩa hàm để tìm path từ root đến một leaf node cụ thể
def find_path_with_output(tree, node_id, feature_names):
    path = []
    output = None
    while node_id != 0:  # 0 là root node
        # Tìm parent node của node hiện tại
        if node_id in tree.children_left:
            parent_node = np.where(tree.children_left == node_id)[0][0]
            split_side = "left"
        else:
            parent_node = np.where(tree.children_right == node_id)[0][0]
            split_side = "right"

        # Lấy feature và threshold tại parent node
        feature_index = tree.feature[parent_node]
        threshold = tree.threshold[parent_node]
        feature_name = feature_names[feature_index]

        # Lấy output tại nút lá
        if tree.children_left[node_id] == tree.children_right[node_id]:
            output = np.argmax(tree.value[node_id])

        # Thêm điều kiện vào path
        if split_side == "left":
            condition = f"{feature_name} <= {threshold}"
        else:
            condition = f"{feature_name} > {threshold}"
        path.append(condition)

        # Di chuyển lên parent node
        node_id = parent_node

    # Đảo ngược path để bắt đầu từ root
    return path[::-1], output

# In ra path và output của top 5 leaf nodes
print("\nPaths to top 5 leaf nodes with output:")
for leaf_id in top_5_leaf_ids:
```

```
    path_to_leaf, output = find_path_with_output(model.tree_, leaf_id, X_train.
↪columns)
    formatted_path = ' -> '.join(path_to_leaf)
    print(f"Leaf node ID: {leaf_id},\nPath: {formatted_path},\nOutput:␣
↪{output}\n")
```

```
Top 5 leaf nodes information:
Leaf node ID: 147, Support: 0.17, Confidence: 1.0
Leaf node ID: 3, Support: 0.047, Confidence: 1.0
Leaf node ID: 444, Support: 0.025, Confidence: 1.0
Leaf node ID: 475, Support: 0.025, Confidence: 1.0
Leaf node ID: 254, Support: 0.024, Confidence: 1.0

Paths to top 5 leaf nodes with output:
Leaf node ID: 147,
Path: Color_hair_Black <= 0.5 -> Height_Medium > 0.5,
Output: 0

Leaf node ID: 3,
Path: Color_hair_Black <= 0.5 -> Height_Medium <= 0.5 -> Gender_Female <= 0.5,
Output: 1

Leaf node ID: 444,
Path: Color_hair_Black > 0.5 -> Weight > 129.16693115234375 -> Weight <=
137.7840347290039 -> Color_eye_Blue <= 0.5,
Output: 1

Leaf node ID: 475,
Path: Color_hair_Black > 0.5 -> Weight > 129.16693115234375 -> Weight >
137.7840347290039 -> Weight > 137.8807144165039 -> Weight > 141.7904052734375 ->
Weight <= 144.53369903564453,
Output: 1

Leaf node ID: 254,
Path: Color_hair_Black > 0.5 -> Weight <= 129.16693115234375 -> Weight <=
126.62903594970703 -> Weight > 46.194114685058594 -> Weight > 56.943504333496094
-> Weight > 63.84756088256836 -> Weight > 63.9512939453125 -> Color_eye_Black <=
0.5 -> Weight <= 92.38833999633789 -> Weight > 74.19501876831055 -> Weight >
77.52743911743164 -> Weight > 79.40025329589844 -> Gender_Female <= 0.5 ->
Color_eye_Blue <= 0.5,
Output: 1
```

5 Câu 5:

Đánh giá mô hình dựa trên (test.csv).

[8]:
```
# Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.74
```

Classification report

[9]:
```
# classification report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

          No       0.72      0.68      0.70       449
         Yes       0.75      0.78      0.77       551

    accuracy                           0.74      1000
   macro avg       0.74      0.73      0.73      1000
weighted avg       0.74      0.74      0.74      1000
```