u6iom48vt

October 17, 2023

1 Ankush Kumar

1.1 AP21110011026

1.1.1 CSE-P

[1]:
```python
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
from sklearn import tree




# Given data
data = {
    'age': ['young', 'young', 'mid', 'old', 'old', 'old', 'mid', 'young',␣
 ↪'young', 'old', 'young', 'mid', 'mid', 'old'],
    'income': ['high', 'high', 'high', 'medium', 'low', 'low', 'low', 'medium',␣
 ↪'low', 'medium', 'medium', 'medium', 'high', 'medium'],
    'student': ['no', 'no', 'no', 'no', 'yes', 'yes', 'yes', 'no', 'yes',␣
 ↪'yes', 'yes', 'no', 'yes', 'no'],
    'credit_rating': ['fair', 'excellent', 'fair', 'fair', 'fair', 'excellent',␣
 ↪'excellent', 'fair', 'fair', 'fair', 'excellent', 'excellent', 'fair',␣
 ↪'excellent'],
    'buys_computer': ['no', 'no', 'yes', 'yes', 'yes', 'no', 'yes', 'no',␣
 ↪'yes', 'yes', 'yes', 'yes', 'yes', 'no']
}

# Convert to DataFrame
df = pd.DataFrame(data)

df
```

[1]:
```
      age  income student credit_rating buys_computer
0   young    high      no          fair            no
1   young    high      no     excellent            no
2     mid    high      no          fair           yes
3     old  medium      no          fair           yes
```

```
4     old     low     yes        fair           yes
5     old     low     yes   excellent            no
6     mid     low     yes   excellent           yes
7   young  medium      no        fair            no
8   young     low     yes        fair           yes
9     old  medium     yes        fair           yes
10  young  medium     yes   excellent           yes
11    mid  medium      no   excellent           yes
12    mid    high     yes        fair           yes
13    old  medium      no   excellent            no
```

[2]:

```
# One-hot encoding for categorical data
df_encoded = pd.get_dummies(df[['age', 'income', 'student', 'credit_rating']])
X = df_encoded
y = df['buys_computer']
```

1.1.2 Step 1: Construct Decision Tree with Information Gain

[3]:

```
# Step 1:
clf_info_gain = DecisionTreeClassifier(criterion="entropy")
clf_info_gain.fit(X, y)
```

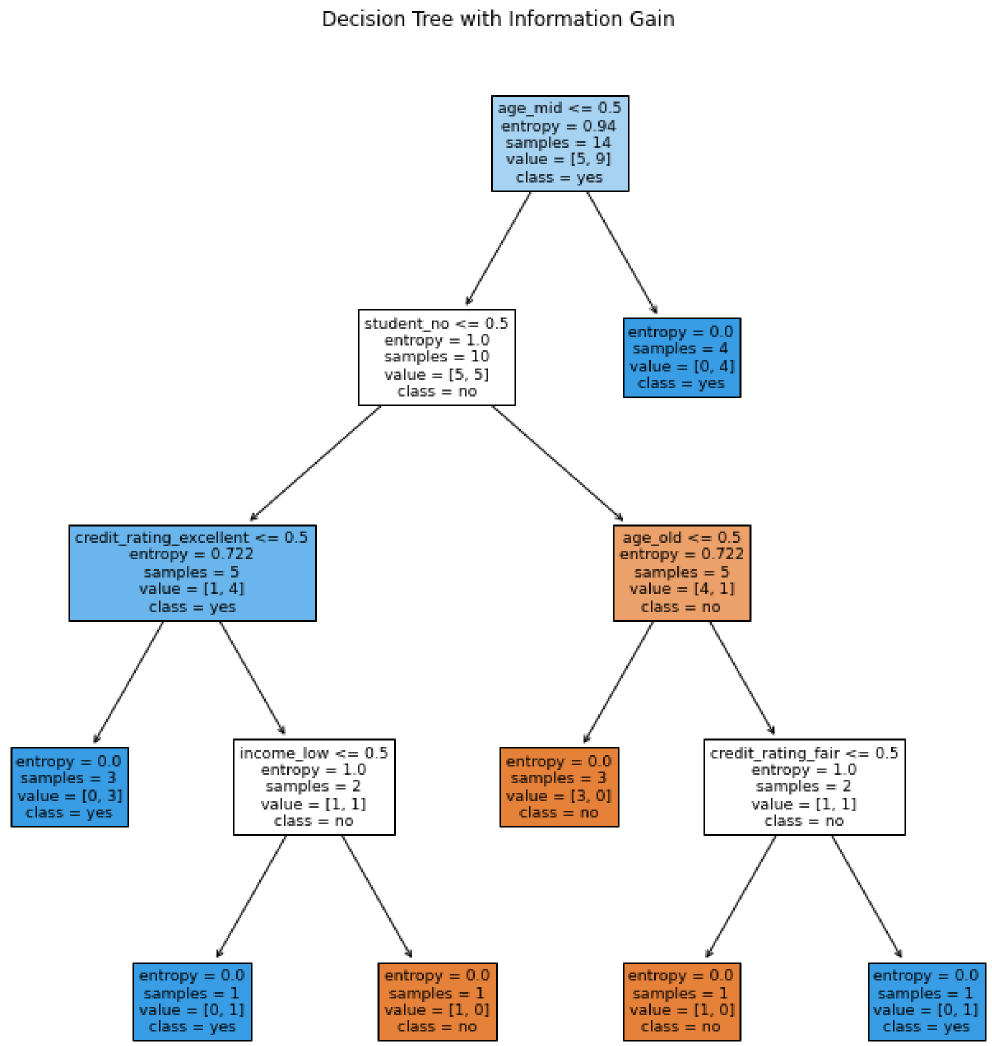
[3]:

```
DecisionTreeClassifier(criterion='entropy')
```

1.1.3 Step 2: Visualizing the Decision Tree with Information Gain

[4]:

```
fig, ax = plt.subplots(figsize=(12, 12))
tree.plot_tree(clf_info_gain, feature_names=X.columns,␣
 ↪class_names=clf_info_gain.classes_, filled=True, ax=ax)
plt.title('Decision Tree with Information Gain')
plt.show()
```



1.1.4 Step 3: Construct Decision Tree with Gain Ratio

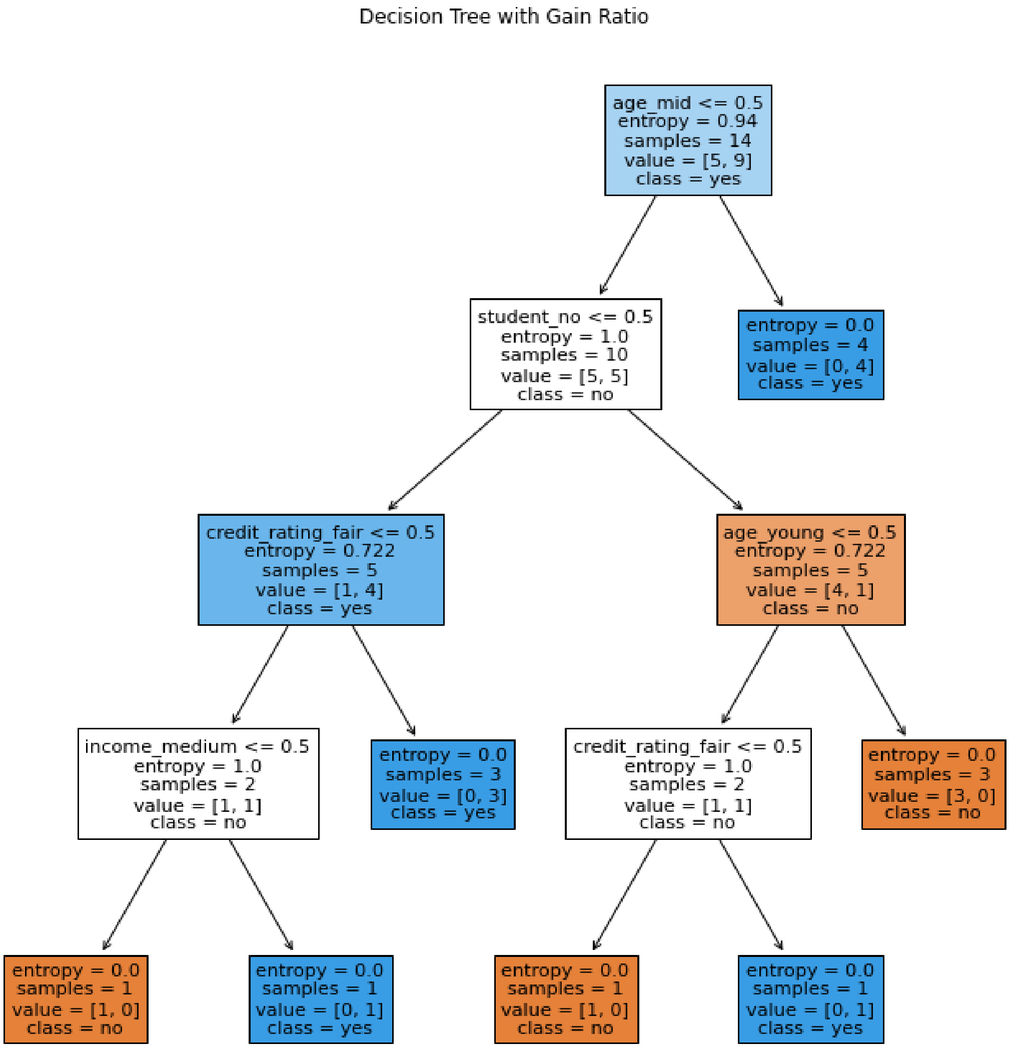
```
[5]: clf_gain_ratio = DecisionTreeClassifier(criterion="entropy", splitter="best")
     clf_gain_ratio.fit(X, y)
```

```
[5]: DecisionTreeClassifier(criterion='entropy')
```

1.1.5 Step 4: Visualizing the Decision Tree with Gain Ratio

```
[6]: fig, ax = plt.subplots(figsize=(12, 12))
     tree.plot_tree(clf_gain_ratio, feature_names=X.columns,␣
      ↪class_names=clf_gain_ratio.classes_, filled=True, ax=ax)
```

```
plt.title('Decision Tree with Gain Ratio')
plt.show()
```



1.1.6 Step 5: Construct Decision Tree with Gini Index

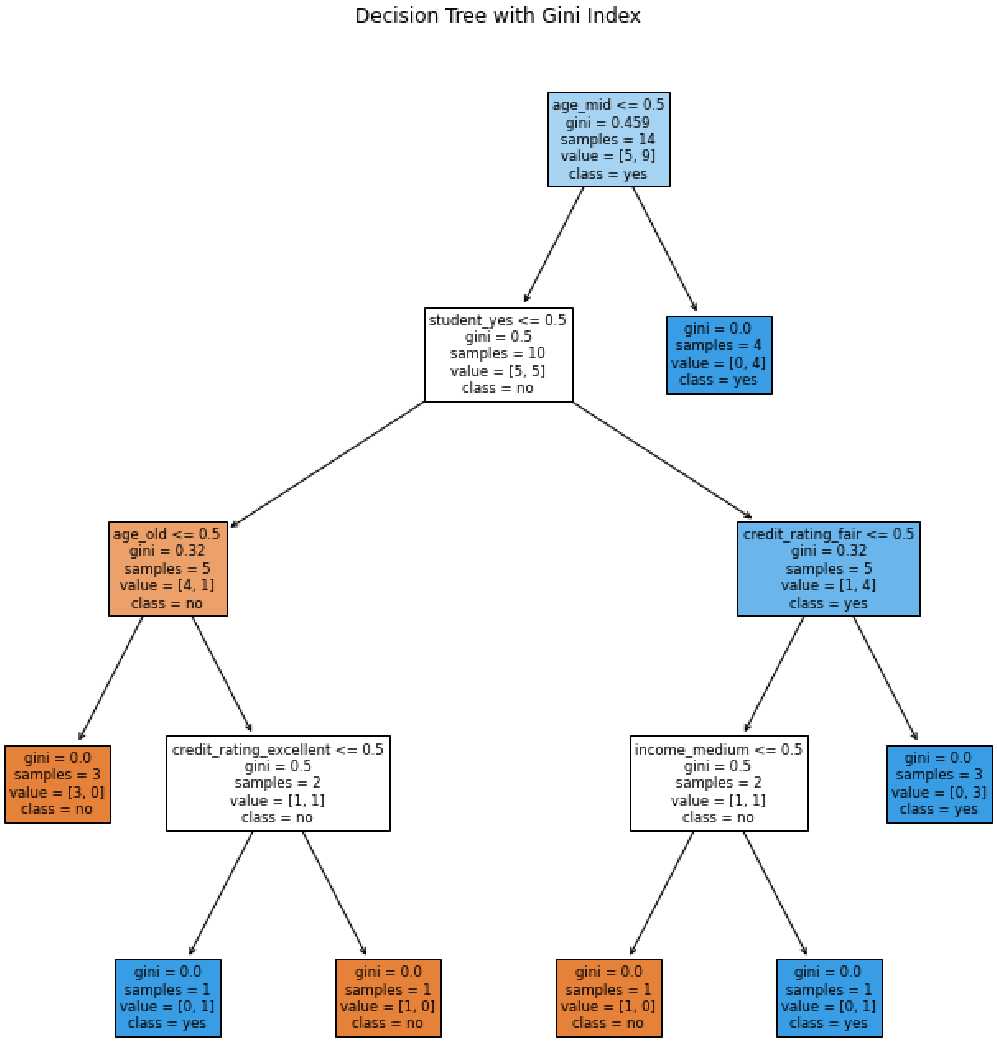
```
[7]: clf_gini = DecisionTreeClassifier(criterion="gini")
     clf_gini.fit(X, y)
```

```
[7]: DecisionTreeClassifier()
```

1.1.7 Step 6: Visualizing the Decision Tree with Gini Index

[8]:

```
fig, ax = plt.subplots(figsize=(12, 12))
tree.plot_tree(clf_gini, feature_names=X.columns, class_names=clf_gini.
 ↪classes_, filled=True, ax=ax)
plt.title('Decision Tree with Gini Index')
plt.show()
```



[ ]: