Aprendizagem 2022

Homework I – Group 019

Diogo Gaspar 99207, Rafael Oliveira 99311

Part I: Pen and paper

1. **Draw the training confusion matrix.**

   After reviewing the decision tree described in the question's statement, we can assert that its confusion matrix is the following:

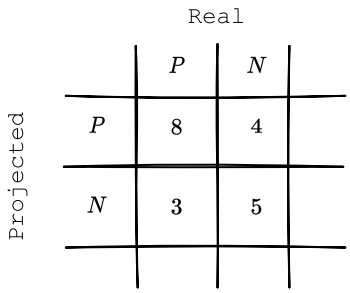

Figure 1: Confusion Matrix

2. **Identify the training F1 after a post-pruning of the given tree under a maximum depth of 1.**

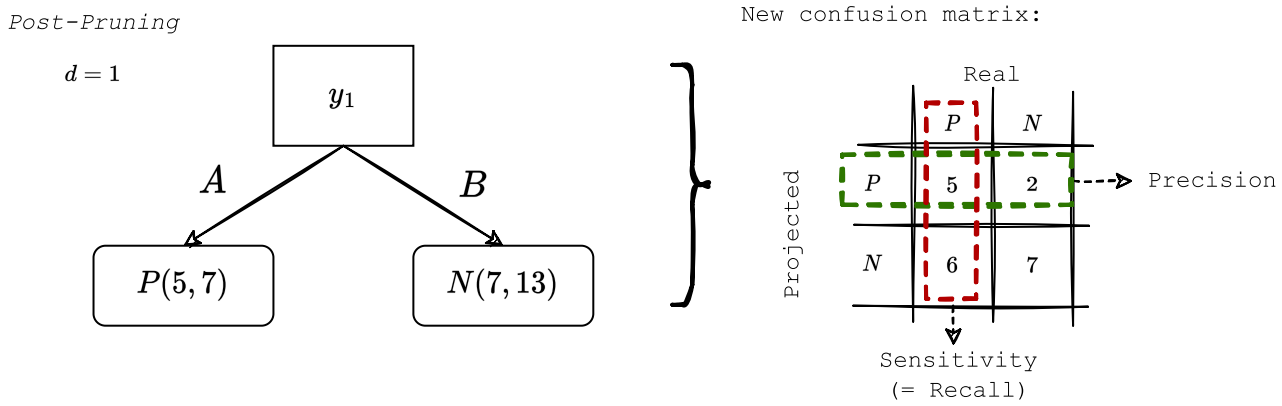

Figure 2: Post-pruning confusion matrix

We know that:

$$F_\beta = \frac{1}{\alpha \frac{1}{\text{Precision}} + (1-\alpha)\frac{1}{\text{Recall}}}, \qquad \text{where} \quad \beta^2 = \frac{1}{\alpha} - 1$$

For $\beta = 1$ (and thus $\alpha = \frac{1}{2}$), we have:

$$d = 1: \quad F_1 = \frac{1}{\frac{1}{2}\frac{1}{\text{Precision}} + \frac{1}{2}\frac{1}{\text{Recall}}}$$

Therefore, we must calculate the associated precision and sensitivity (recall) as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{5}{5+2} = \frac{5}{7}, \quad \text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{5}{5+6} = \frac{5}{11}$$

Finally, we can now calculate the F_1 score:

$$d = 1: \quad F_1 = \frac{1}{\frac{1}{2} \cdot \frac{7}{5} + \frac{1}{2} \cdot \frac{11}{5}} = \frac{5}{9} \approx 0.5556$$

3. **Identify two different reasons as to why the left tree path was not further decomposed.**

   We can find multiple reasons for the left-most tree path not being further decomposed.

   The first, perhaps most obvious reason is that we want to **avoid overfitting** the tree to the training data: given a threshold of $\frac{5}{7}$, it's already possible to make a very fair, general assessment about all instances that fall under the $y_1 = A$ condition (asserting they fall onto the P label). It's obviously not guaranteed that not further decomposing the tree will lead to a better labeling come testing time, but generally speaking it ends up being a good practice to avoid overfitting.

   Related to the concept of overfitting (specifically, its prevention), there also aren't many instances (only 7) in the left path, which makes it so that further decomposing that path doesn't change the classification error by much (if at all), while also adding unnecessary complexity to the tree - therefore, it's **not likely** that there will be a significant **entropy reduction** by further decomposing the left-most tree path.

4. **Compute the information gain of variable y1.**

   We know that:

$$\text{IG}(y_{out} \mid y_1) = \text{H}(y_{out}) - \text{H}(y_{out} \mid y_1) \tag{1}$$

$$\text{H}(Y) = -\sum_{y \in Y} p(y) \log_2 p(y) \tag{2}$$

   By looking at the decision tree's confusion matrix (computed in the first question), we can infer that $P(y_{out} = P) = \frac{11}{20}$ and $P(y_{out} = N) = \frac{9}{20}$. After gathering the probabilities

above, and considering y_{out} as our *output variable*, with possible values P or N, we'll be able to note that:

$$\mathrm{H}(y_{out}) = -\frac{11}{20}\log_2\frac{11}{20} - \frac{9}{20}\log_2\frac{9}{20} \tag{3}$$

To finish being able to calculate the previously mentioned information gain, we still need to mention a few missing probabilities associated with the given decision tree:

- $P(y_1 = A) = \frac{7}{20}, P(y_1 = B) = \frac{13}{20}$, which can be asserted by looking at the decision tree's confusion matrix and adding the values in each column (the **real** values).
- $P(y_{out} = P \mid y_1 = A) = \frac{5}{7}, P(y_{out} = N \mid y_1 = A) = \frac{2}{7}$, which can be gathered by looking at the left-most tree path and checking the true and false P values.
- $P(y_{out} = P \mid y_1 = B) = \frac{6}{13}, P(y_{out} = N \mid y_1 = B) = \frac{7}{13}$, which can be gathered by looking at the right-most tree path's leaves, checking each value's true or false values and adding them up.

$$\mathrm{H}(y_{out} \mid y_1) = \left[\frac{7}{20}\left(-\frac{5}{7}\log_2\frac{5}{7} - \frac{2}{7}\log_2\frac{2}{7}\right) + \frac{13}{20}\left(-\frac{6}{13}\log_2\frac{6}{13} - \frac{7}{13}\log_2\frac{7}{13}\right)\right] \tag{4}$$

We therefore have:

$$\mathrm{IG}(y_{out} \mid y_1) = 0.04345941113$$

Part II: Programming

5. **Using `sklearn`, apply a stratified 70-30 training-testing split with a fixed seed (`random_state=1`), and assess in a single plot the training and testing accuracies of a decision tree with no depth limits (and remaining default behavior) for a varying number of selected features in {5,10,40,100,250,700}. Feature selection should be performed before decision tree learning considering the discriminative power of the input variables according to mutual information criterion (`mutual_info_classif`).**

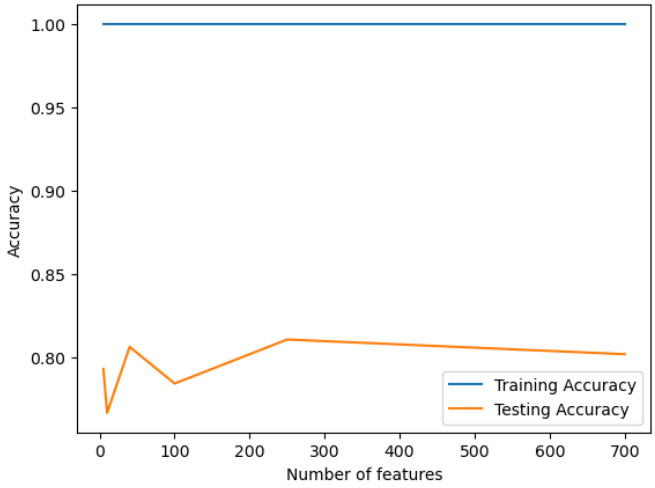

Figure 3: Training and Testing Accuracy vs. Number of Features

The code utilized to generate the plot is present in the report's appendix.

6. **Why is training accuracy persistently 1? Critically analyze the gathered results.**

   Since the decision tree has no depth-limit associated, the worst-case scenario sees the tree holding exactly one leaf per training sample - leading to there always being a "correct path" in the tree for a given observation - and thus the training accuracy is always 1, for any train-test split of the dataset. It's the "limit case" for an overfitting situation, where the tree is completely tailored to the training data.

   The testing accuracy isn't necessarily 1, of course, just like it can be seen in the plot shown above: in the vast majority of cases, the testing set will hold samples which differ from all seen in the training set, effectively evicting the "guarantee" of a correct guess from the decision tree. It can, of course, still take a correct guess (with that being our goal), but it's not guaranteed.

Appendix

The code utilized in the first question of the programming section is shown below:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.io.arff import loadarff
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import mutual_info_classif
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.feature_selection import SelectKBest

FEATURE_AMOUNT = [5, 10, 40, 100, 250, 700]
SEED = 1
SPLIT = 0.3

# Loading data

df = loadarff('data/pd_speech.arff')
df = pd.DataFrame(df[0])
df['class'] = df['class'].str.decode('utf-8')

# The first step should be splitting the dataset into stratified training and
# testing sets (70-30 split) with a fixed seed (random_state=1).

X, y = df.drop('class', axis=1), df['class']
X_train, X_test, y_train, y_test = train_test_split(
  X, y,
  test_size=SPLIT,
  random_state=SEED,
  stratify=y
)

# Now, time to finally assess (in a single plot) both the training and testing
    accuracies
# of a decision tree with no depth limits (and remaining default behavior) for a
    varying number of selected features.

CLF = DecisionTreeClassifier(random_state=SEED)

def train_and_test(X_train, X_test, y_train, y_test):
    y_pred = CLF.predict(X_test)
    return accuracy_score(y_test, y_pred)
def mutual_info_classif_custom(X, y):
    return mutual_info_classif(X, y, random_state=SEED)

train_accuracies = []
test_accuracies = []

for n in FEATURE_AMOUNT:
    selector = SelectKBest(score_func=mutual_info_classif_custom, k=n)
    selector.fit(X_train, y_train)
    X_train_new = selector.transform(X_train)
    X_test_new = selector.transform(X_test)
```

```
    CLF.fit(X_train_new, y_train)
    train_accuracies.append(train_and_test(X_train_new, X_train_new, y_train,
    y_train))
    test_accuracies.append(train_and_test(X_train_new, X_test_new, y_train, y_test)
    )

plt.plot(FEATURE_AMOUNT, train_accuracies, label='Training Accuracy')
plt.plot(FEATURE_AMOUNT, test_accuracies, label='Testing Accuracy')
plt.xlabel('Number of features')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```