Aprendizagem 2022

Homework I – Group 019

Diogo Gaspar 99207, Rafael Oliveira 99311

Part I: Pen and paper

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

   Após consulta da árvore de decisão descrita no enunciado, podemos afirmar que a respetiva matriz de confusão é a seguinte:

		Real	
		P	N
Projected	P	8	4
	N	3	5

Figure 1: Matriz de Confusão

   Note-se que as colunas correspondem aos acontecimentos reais, enquanto que as linhas correspondem aos acontecimentos previstos.

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

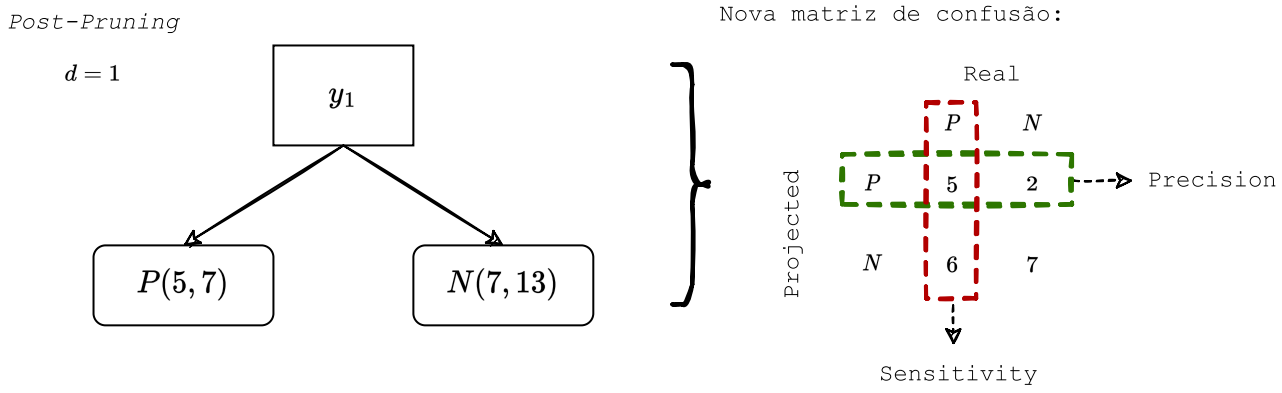

Figure 2: Matriz de Confusão pós-poda

   Temos que:

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

   Precisamos, portanto, de calcular a precisão e a sensibilidade para obter o valor pretendido.

$$\text{Sensitivity} = \frac{TP}{TP + FN} = \frac{5}{5+6} = \frac{5}{11}$$

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{5}{5+2} = \frac{5}{7}$$

Perante os dados obtidos, podemos afirmar que:

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

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

   TODO

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

   Temos que:

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

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

   Ora, considerando aqui y_{out} como a nossa variável de output, com valores P ou N, vamos ter que:

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

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

   Teremos, portanto:

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

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`).**

   Como nota, os gráficos gerados não aparentaram ser determinísticos, mesmo dando *set* à *seed* manualmente - isto é, correr o código várias vezes leva a gráficos diferentes, apesar de semelhantes. A diferença, claro, surge apenas na linha da *training accuracy*.

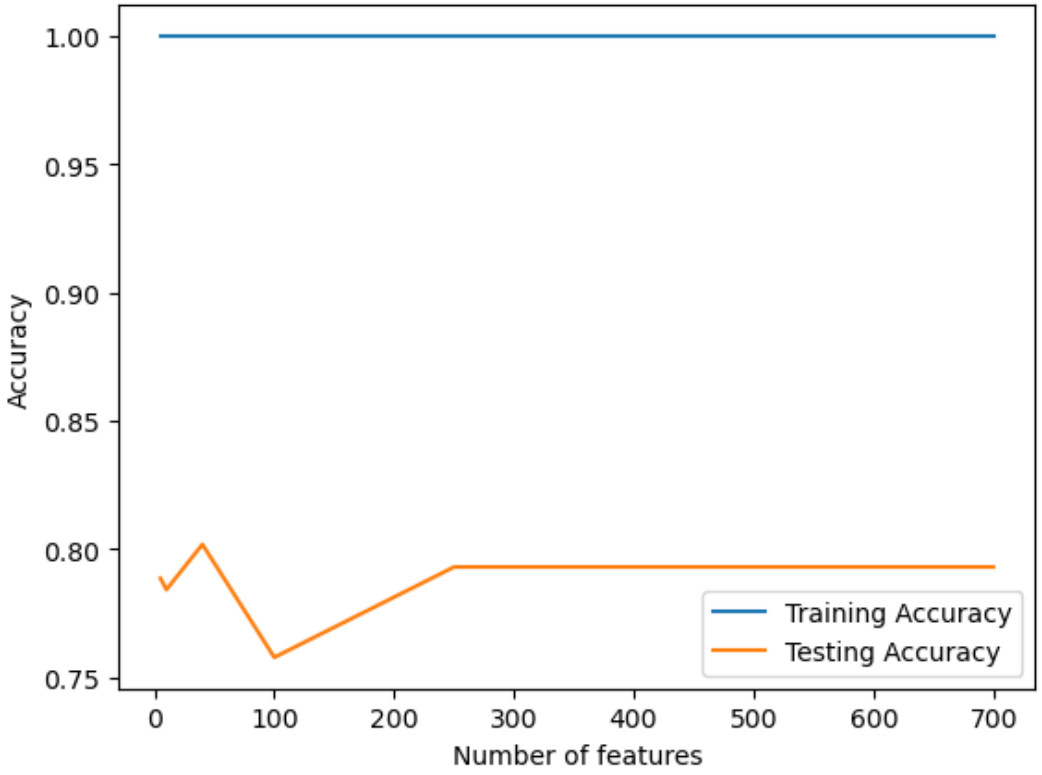

Figure 3: Training and Testing Accuracy vs. Number of Features

   O código utilizado para gerar o gráfico encontra-se no apêndice do presente relatório.

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.

   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

Code utilized in the first question of the programming section shown in the pages 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

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

# 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).

train, test = train_test_split(df, test_size=0.3, random_state=SEED, stratify=df['
    class'])
X = train.drop('class', axis=1)
y = train['class']

# Afterward, we should try to check the discriminative power of each feature
    considering the mutual_info_classif criterion.
# If we can only use a subset of features, we should select the most discriminative
     ones,
# (i.e, if we can use 5 features, we should select the 5 most discriminative ones,
    and so on).

mimportance = mutual_info_classif(X, y)
SORTED_FEATURE_INDICES = np.argsort(mimportance)[::-1]

# 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.

def train_and_test(X_train, X_test, y_train, y_test):
  clf = DecisionTreeClassifier()
```

```
  clf.fit(X_train, y_train)
  y_pred = clf.predict(X_test)
  return accuracy_score(y_test, y_pred)

train_accuracies = []
test_accuracies = []

for n in FEATURE_AMOUNT:
    chosen_features = X.columns[SORTED_FEATURE_INDICES[:n]]
    X_train = train[chosen_features]
    X_test = test[chosen_features]
    train_accuracies.append(train_and_test(X_train, X_train, y, y))
    test_accuracies.append(train_and_test(X_train, X_test, y, test['class']))

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()
```