Aprendizagem 2022/23

Homework I – Group 021

I. Pen-and-paper

1)

1)

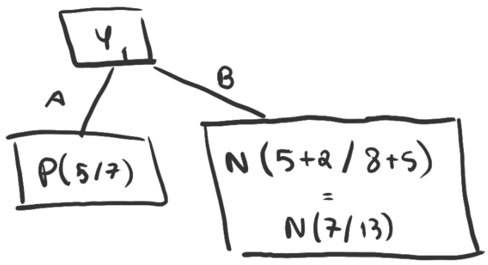
Predicted \ real	P	N
P	5+3	2+2
N	3	5

=

Predicted \ real	P	N
P	8	4
N	3	5

2)

2) Prunning (depth = 1)



CM:

Predicted \ real	P	N
P	5 (TP)	2 (FP)
N	6 (FN)	7 (TN)

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

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

$$F1 = \frac{2\,\text{Prec}\cdot\text{Sens}}{\text{Prec}+\text{Sens}} = \frac{2\cdot\frac{5}{7}\cdot\frac{5}{11}}{\frac{5}{7}+\frac{5}{11}} = \frac{5}{9}$$

3)

3) Two reasons that can justify why the left path was not further decomposed are:

- To avoid overfitting, because exploring the left path would make the model adapted to the training data;
- The left path could be an unecessary branch which would slow down the tree's performance.

4)

$$4)\quad IG(y_{out} \mid y_1) = I(y_{out}) - E(y_{out} \mid y_1)$$

	P	N
y_{out}	$5+3+(8-5) = 11$	$5+(7-5)+(5-3) = 9$

$$\Rightarrow P(y_{out} = P) = 11/20$$
$$P(y_{out} = N) = 9/20$$

$$I(y_{out}) = -\left[\frac{11}{20}\log\left(\frac{11}{20}\right) + \frac{9}{20}\log\left(\frac{9}{20}\right)\right]$$
$$\simeq 0.992774454$$

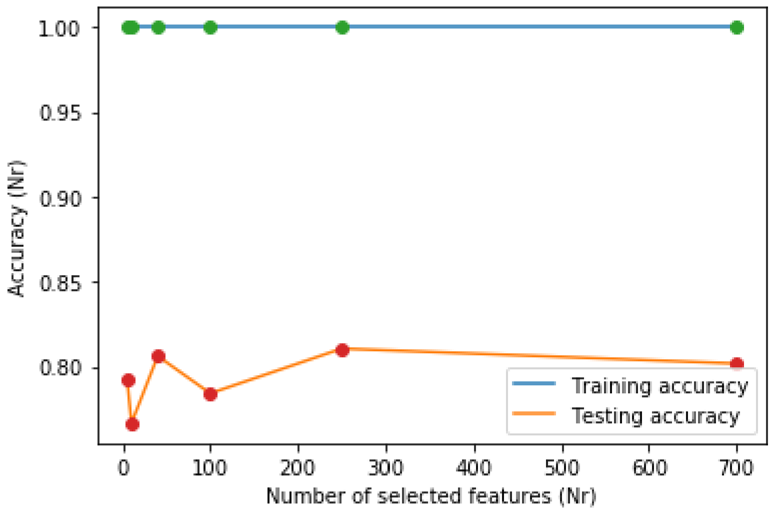
$$E(y_{out} \mid y_1) = \frac{7}{20} I(y_{out} \mid y_1 = A) + \frac{13}{20} I(y_{out} \mid y_1 = B)$$
$$= -\frac{7}{20}\left(\frac{5}{7}\log_2\frac{5}{7} + \frac{2}{7}\log_2\frac{2}{7}\right) - \frac{13}{20}\left(\frac{7}{13}\log\frac{7}{13} + \frac{6}{13}\log\frac{6}{13}\right)$$
$$\simeq 0.949315$$

$$IG(y_{out} \mid y_1) = I(y_{out}) - E(y_{out} \mid y_1)$$
$$\simeq 0.992774454 - 0.949315$$
$$= 0.043459$$

II. Programming and critical analysis

1) The code for this question is in the Appendix of this document.

Resulting Graph:



Aprendizagem 2022/23

Homework I – Group 021

2) The training accuracy is persistently 1 because there are no depth limits on the decision tree. This makes the tree very well trained to the training data (overfitted) and therefore predict the training data flawlessly.

III. APPENDIX

Code used in question 1) of **II. Programming and critical analysis**:

Loads the Parkinson Disease's Data:

```
import pandas as pd
import numpy as np
from scipy.io.arff import loadarff

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

X=df.drop('class', axis=1)
y = df['class']
```

Calculates the Training/Testing Accuracies:

```
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 chi2, SelectKBest

# Splits the data with the fixed seed
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, stratify=y,
random_state=1)

selected_features =  [5, 10, 40, 100, 250, 700]
train_accuracies = []
test_accuracies = []

def classifier_func(X, y):
       return mutual_info_classif(X, y, random_state=1)
```

```python
# Iterates over each feature
for feature in selected_features:
    # Calculates the mutual information between the features and the target,
    # and normalizes it to be a % of the maximum value
    mutual_info = SelectKBest(classifier_func, k=feature)
    mutual_info.fit(X_train, y_train)
    X_train_selected = mutual_info.transform(X_train)
    X_test_selected = mutual_info.transform(X_test)

    # Create the decision tree, train it and predict
    clf = DecisionTreeClassifier(random_state=1)
    clf.fit(X_train_selected, y_train)

    y_train_predict = clf.predict(X_train_selected)
    y_test_predict = clf.predict(X_test_selected)

    # Compute accuracy
    train_accuracy = accuracy_score(y_train, y_train_predict)
    test_accuracy = accuracy_score(y_test, y_test_predict)
    train_accuracies.append(train_accuracy)
    test_accuracies.append(test_accuracy)
```

Plots the accuracies in a graph:

```python
import matplotlib.pyplot as plt

plt.plot(selected_features, train_accuracies, label="Training accuracy")
plt.plot(selected_features, test_accuracies, label="Testing accuracy")

plt.plot(selected_features, train_accuracies, 'o')
plt.plot(selected_features, test_accuracies, 'o')

plt.xlabel("Number of selected features (Nr)")
plt.ylabel("Accuracy (Nr)")
plt.legend()
plt.show()
```