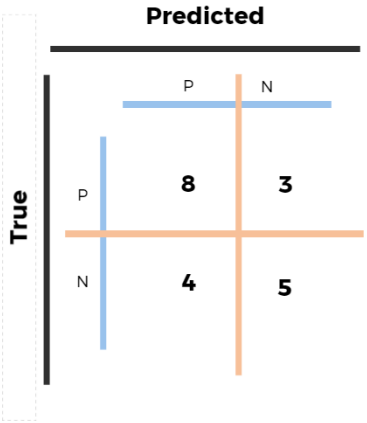
Aprendizagem 2021/22

Homework I – Group 081

I. Pen-and-paper

1) Training confusion matrix result:



Explaining the values:

We sad **(predicted)** that were 12 Positive cases (7+5), in this 12, 8 were **truly** a positive cases and we fail 4.

We sad **(predicted)** that 8 were negative cases (3+5), in this 8, 5 were **truly** negative cases and we fail 3.

2) Post-pruning the given tree and identify the training F1

New maximum depth: 1

Lets try to simplify y2 in order to remove the oldest maximum depth (2)

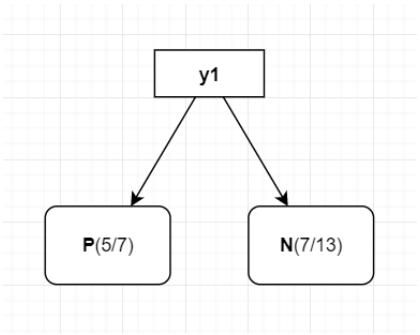
y2 branches : N(5/8) and P(3/5)

True positives: 3
True negatives: 5
False positives: 2
False negatives: 3

We need to "agregate" both branches into one. The new leaf should be a N or a P?
N: True negative + False postive = 5 + 2 = 7
P: True postive + False negative = 3 + 3 = 6

Because 7 > 6 we chose our leaf to be N.

The new tree will end up looking like this:



TÉCNICO LISBOA

Aprendizagem 2021/22

Homework I – Group 081

Calculate F1 using the new tree:

F1 = 2 / [(1/Precision) + (1/Recall)]

True positive = 5
False postive = 2
False negative = 6

Precision = TP/(TP+FP) = 5/7
Recall = TP/(TP+FN) = 5/11

F1 ≈ 0.(5)

3) Why the left tree path was no further decomposed?

There are many reasons why this happened, as a group we decided this two were the most likely to happen in a real world case.

1. In order to avoid overfitting, we prepared our tree to stop once it finds a great estimator to the value, in this way we didn`t search for E (entropy) 0. It might have reached a good value once we go to the left path so the tree didn`t search for other results.

2. We have already reached a limit point, going further will not change our answer, in short, the gain is 0 once we try to branch this path.

4) Calculate information gain on y1 (on the original tree)

(foto with solution down below)

4.

ID I (tabb) : ? = $-f(N)\log_2(f(N)) - f(P)\log_2(f(P))$

↝ $f(N) = 9/20$

$f(P) = 11/20$

$I(tabb) = 0.993$

> $C_A = \{P, P, P, P, P, N, N\}$

> $C_B = \{P, P, P, P, P, P, N, N, N, N, N, N, N\}$

ID $I(C_A) = -f(N) \times \log_2(f(N)) - f(P) \times \log_2(f(P))$

↝ $f(N) = 2/7$

$f(P) = 5/7$

$I(C_A) \tilde{=} 0.8631$

ID $I(C_B) = -f(N) \times \log_2(f(N)) - f(P) \times \log_2(f(P))$

↝ $f(N) = 7/13$

$f(P) = 6/13$

$I(C_B) \tilde{=} 0.996$

$E(y_1) = \frac{7}{20} \times 0.8631 + \frac{13}{20} \times 0.996$

$= 0.9495$

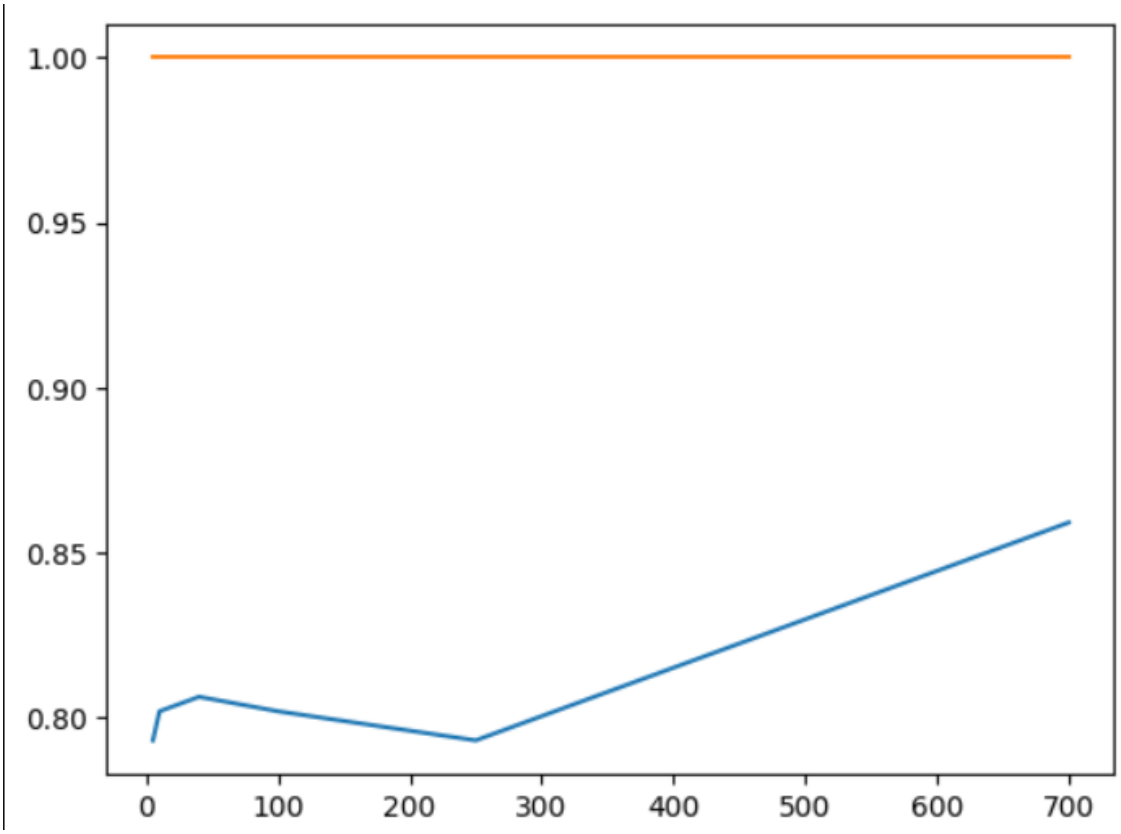
$gain(y_1) = I(tabb) - E(y_1) = 0.0435$

In conclusion:

Gain(y1) ≈ 0.0435

II. Programming and critical analysis

1) Assess in a single plot the training and testing accuracies of a decision tree with no depth limits



2) **Why training accuracy is always 1?**

Of course the training accurary is always 1. We train our classifier with the training data set, so it retains all the results of that data set. Of course when we test the same classifier with the same data set, it will give us the same result on the original data set. This way it will have a accuracy of 1 (it give us the results of the original training set).

III. APPENDIX

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

TÉCNICO LISBOA

Aprendizagem 2021/22

Homework I – Group 081

```
# Reading the ARFF file
data = loadarff('pd_speech.arff')
df = pd.DataFrame(data[0])
df['class'] = df['class'].str.decode('utf-8')

# Split in data and targets
X = df.drop('class', axis=1)
y = df['class']

labels = list(set(y))

# Split data in test and training set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
train_size=0.7, random_state=1, stratify=y)

# Feature selection

result_test = []
result_train = []

for e in [5,10,40,100,250,700]:
    selector = SelectKBest(mutual_info_classif, k=e)
    selector.fit_transform(X_train, y_train)
    cols_indexes = selector.get_support(indices=True)
    selected_columns = X.iloc[:,cols_indexes].columns.tolist()

    # Reduce training and testing sets
    X_train_reduced = X_train[selected_columns]
    X_test_reduced = X_test[selected_columns]

    # Create and train the classifier
    classifier = tree.DecisionTreeClassifier()
    classifier = classifier.fit(X_train_reduced, y_train)

    # Test classifier and compute scores
    y_predicted = classifier.predict(X_test_reduced)

    accuracy = classifier.score(X_test_reduced, y_test)
    train_accuracy = classifier.score(X_train_reduced, y_train)

    result_test.append(accuracy)
    result_train.append(train_accuracy)

print("\n")
print(result_test)
```

```
print(result_train)

print(plt.plot([5,10,40,100,250,700],result_test,[5,10,40,100,250,700],result_train))
```

END