

I. Pen-and-paper

1)

1)

		Predicted	
		P	N
Real	P	8	3
	N	4	5

2)

2) Pruned Tree

```

graph TD
    Y1[Y1] -- A --> P57[P(5/7)]
    Y1 -- B --> N13[N(5+2 / 8+5)]
    N13 --> N713[N(7/13)]
  
```

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

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

$\frac{1}{F} = \frac{1}{2} \left(\frac{11}{5} + \frac{7}{5} \right) (=) F = \frac{5}{9} \approx 0,556$

		Predicted	
		P	N
Real	P	5 True Pos	6 False Neg
	N	2 False Pos	7 True Neg

3)

3).

There should be a loose stopping criterion and use pruning to remove branches that contribute to overfitting.

The left tree path was not further decomposed mainly due to overfitting.

In general, the deeper a tree is allowed to grow, the more complex the model becomes due to there being more splits and captured information about the given data. Therefore one of the reasons why the left path is not further decomposed is to prevent the tree from adjusting perfectly to the training data which also prevents overfitting.

→ Moreover, given the fact that the 5 out of the 7 predicted cases were correct when $y_1 = A$, further decomposing this node of the tree would not be beneficial not only because we already have correctly predicted most of these cases, but also because the tree would be overly adjusted to the training data, resulting in overfitting.

→ Another possible reason why the left tree path was not further decomposed is that there may have been imposed a minimum number of occurrences of a condition in order to expand a node. For instance, there is a possibility that this minimum number was 8, which results in an expansion of the tree in the right side and not the left, as there were only 7 cases which $y_1 = A$. This method also prevents overfitting.

4)

41

$$IG(y_{out} | y_1) = E(y_{out}) - E(y_{out} | y_1)$$

$$E(out) = - \left(\frac{11}{20} \log \frac{11}{20} + \frac{9}{20} \log \frac{9}{20} \right)$$

$$\approx 0,993$$

$$E(y_{out} | y_1) = E(y_{out} | y_1 = A) + E(y_{out} | y_1 = B)$$

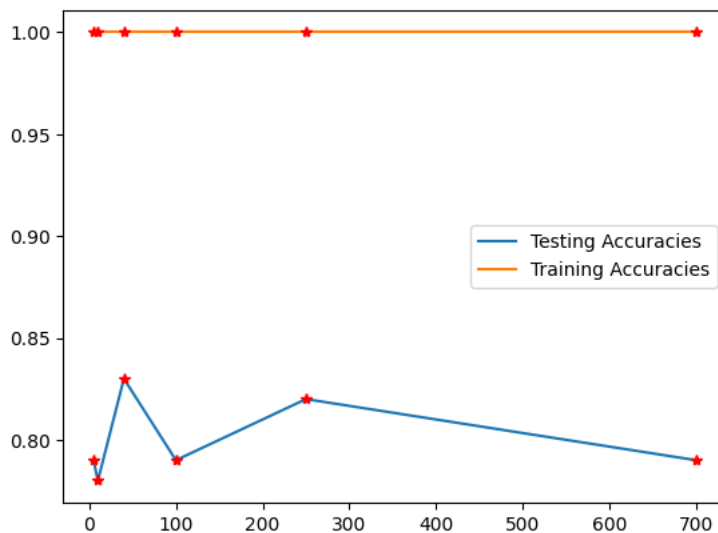
$$= - \frac{7}{20} \left(\frac{5}{7} \log \frac{5}{7} + \frac{2}{7} \log \frac{2}{7} \right) - \frac{13}{20} \left(\frac{6}{13} \log \frac{6}{13} + \frac{7}{13} \log \frac{7}{13} \right)$$

$$\approx 0,949$$

$$IG(y_{out} | y_1) = 0,993 - 0,949 = 0,044$$

II. Programming and critical analysis

1)



2)

2) Training accuracy is persistently 1 due to the fact that there is no depth limit. Consequently, even when training is restricted to as little as 5 features, there is an issue of overfitting, causing the model to adjust too well to the data used in the training process. Therefore, when given the input it used to train, it will predict the exact output also used in training.

III. APPENDIX

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

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

testingAccuracies = []
trainingAccuracies = []

featureNums = (5, 10, 40, 100, 250, 700)

target = df["class"]

variables = []

for i in df.columns:
    variables.append(i)

X = df.copy()
X = X.drop('class', axis = 1)

X_train, X_test, y_train, y_test = train_test_split(X, target, test_size = 0.30, stratify = target, random_state = 1)

for i in featureNums:

    selector_method = SelectKBest(mutual_info_classif, k = i)
    selector_method.fit(X_train, y_train)
```

Aprendizagem 2021/22
Homework I – Group 015

```
X_train_i = selector_method.transform(X_train)
X_test_i = selector_method.transform(X_test)

predictor = tree.DecisionTreeClassifier()
predictor.fit(X_train_i, y_train)

y_predTesting = predictor.predict(X_test_i)
y_predTraining = predictor.predict(X_train_i)

testingAccuracies.append(round(metrics.accuracy_score(y_test, y_predTesting), 2))

trainingAccuracies.append(round(metrics.accuracy_score(y_train, y_predTraining), 2))

print("accuracy on testing set:", round(metrics.accuracy_score(y_test, y_predTesting), 2))

plt.plot(featureNums, testingAccuracies, label = "Testing Accuracies")

plt.plot(featureNums, trainingAccuracies, label = "Training Accuracies")

for i in range(0, len(featureNums)):
    plt.plot(featureNums[i], testingAccuracies[i], 'r*')
    plt.plot(featureNums[i], trainingAccuracies[i], 'r*')

plt.legend()
plt.show()
```

END