I Pen-and-paper

1.

	P	N
P	8	4
N	3	5

TP = 5+3 = 8

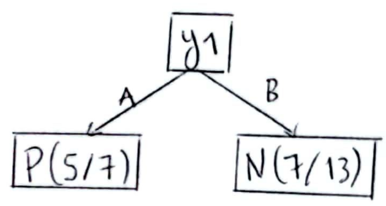
TN = 5

FP = 2+2 = 4

FN = 3

2.

y_1	y_2	O
B	>2	N
B	>2	N
B	>2	N
B	>2	N
B	>2	N
B	>2	P
B	>2	P
B	>2	P
B	≤ 2	P
B	≤ 2	P
B	≤ 2	P
B	≤ 2	N
B	≤ 2	N
A	?	P
A	?	P
A	?	P
A	?	P
A	?	P
A	?	N
A	?	N



TP = 5

TN = 7

FP = 13-7 = 6

FN = 7-5 = 2

$$\text{precision} = \frac{TP}{TP+FP} = \frac{5}{5+6} = \frac{5}{11}$$

$$\text{recall} = \frac{TP}{TP+FN} = \frac{5}{5+2} = \frac{5}{7}$$

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = 2 \times \frac{\frac{5}{11} \times \frac{5}{7}}{\frac{5}{11} + \frac{5}{7}} = \frac{5}{9}$$

3.

O último nó do caminho esquerdo poderia conter apenas elementos da mesma classe, terminando a ~~execução~~ expansão do caminho, ou poderia não haver mais atributos por testar.

4.

$I(\text{table}) =$ $\qquad p(N) = \frac{9}{20}$

$= -\frac{9}{20}\log_2\left(\frac{9}{20}\right) - \frac{11}{20}\log_2\left(\frac{11}{20}\right) =$ $\qquad p(P) = \frac{11}{20}$

$= 0{,}993$ bits

y_1

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

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

$I(C_A) = -\frac{5}{7}\log_2\left(\frac{5}{7}\right) - \frac{2}{7}\log_2\left(\frac{2}{7}\right) = 0{,}8631$ bits

$I(C_B) = -\frac{6}{13}\log_2\left(\frac{6}{13}\right) - \frac{7}{13}\log_2\left(\frac{7}{13}\right) = 0{,}9957$ bits

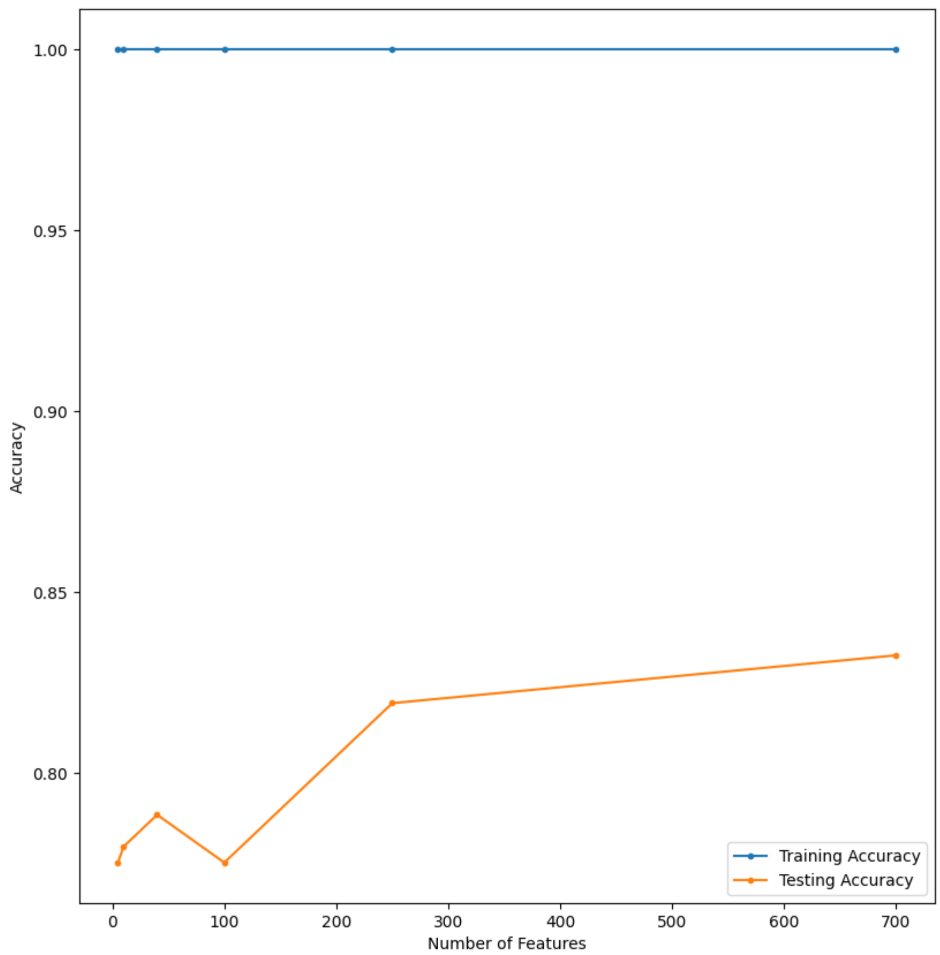
$E(y_1) = \frac{7}{20} \times 0{,}8631 + \frac{13}{20} \times 0{,}9957 =$

$= 0{,}94929$ bits

$\text{gain}(y_1) = I(\text{table}) - E(y_1) = 0{,}04371$ bits

II Programming

1.



2. From the gathered results, we can conclude that there is a large difference between the training accuracy and the testing accuracy. The results show a training accuracy of 1, therefore, the model is overfitting to the training set and doesn't generalize so nicely to the testing set.

III Appendix

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

if __name__ == "__main__":
  data, meta = loadarff("pd_speech.arff")
  df = pd.DataFrame(data)
  df["class"] = df["class"].str.decode("utf-8")
  df.dropna(inplace=True)

  X, y = df.iloc[:, 0:-1], df.iloc[:, -1]
  X_train, X_test, y_train, y_test = train_test_split(X,
                                                      y,
                                                      train_size=0.7,
                                                      stratify=y,
                                                      random_state=1)

  predictor = tree.DecisionTreeClassifier()
  num_features = (5, 10, 40, 100, 250, 700)
  training_accuracy = []
  testing_accuracy = []

  for features in num_features:
    selector = SelectKBest(mutual_info_classif, k=features)
    selector.fit(X_train, y_train)
    X_train_new = selector.transform(X_train)
    X_test_new = selector.transform(X_test)

    predictor.fit(X_train_new, y_train)
    y_pred_train = predictor.predict(X_train_new)
    y_pred_test = predictor.predict(X_test_new)

    training_accuracy += [metrics.accuracy_score(y_train, y_pred_train)]
    testing_accuracy += [metrics.accuracy_score(y_test, y_pred_test)]

  plt.plot(num_features, training_accuracy, marker=".")
  plt.plot(num_features, testing_accuracy, marker=".")
  plt.xlabel("Number of Features")
  plt.ylabel("Accuracy")
  plt.legend(["Training Accuracy", "Testing Accuracy"])
  plt.show()
```