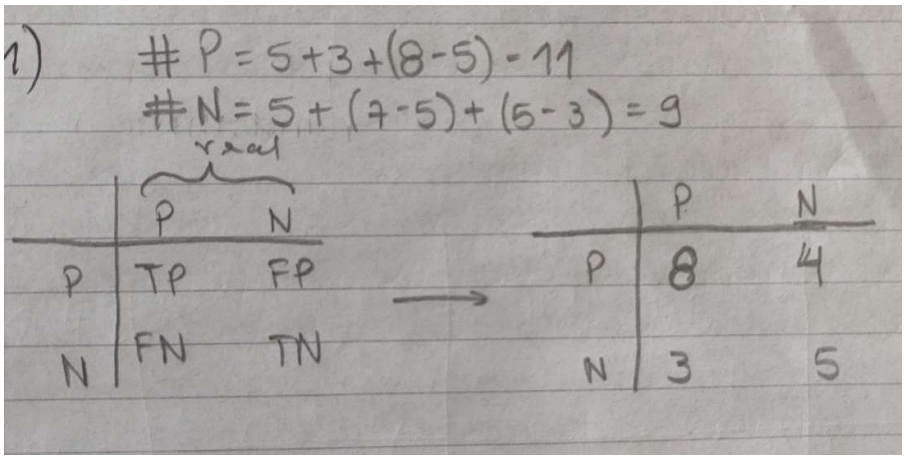
Aprendizagem 2022/23

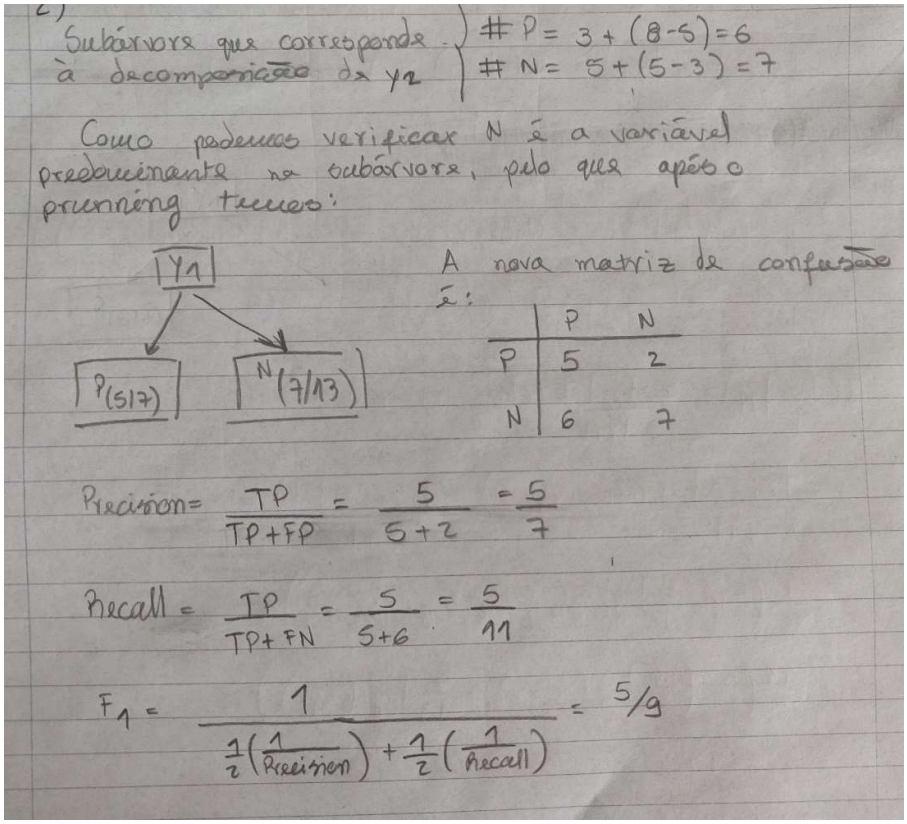
Homework I – Group 084

I. Pen-and-paper

1)



2)

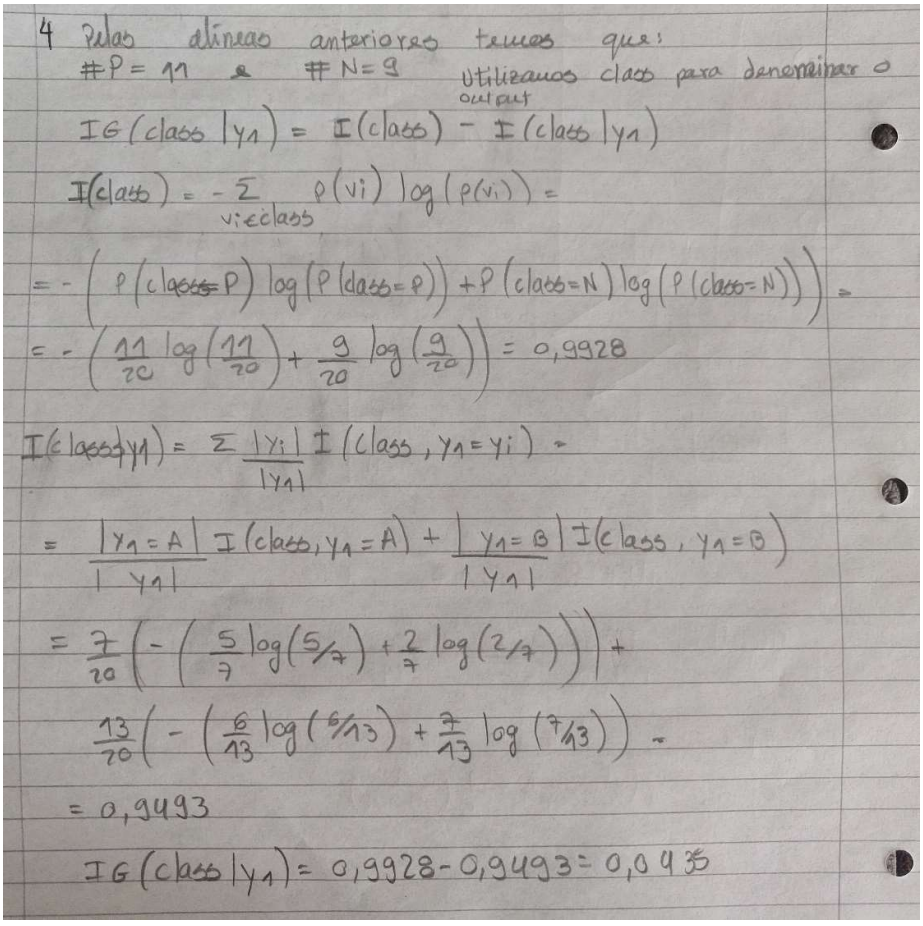


3)

i. Podemos ter que, para qualquer outra variável, o seu information gain quando y1=A seja nulo, pelo que não se justifica realizar a decomposição.

ii. Ao olharmos para a árvore conseguimos verificar que quando y1=A temos uma precisão de 5/7, podendo, desta forma, pressupor que a expansão deste nó levaria a um cenário de overfitting, pelo que o seu crescimento foi interrompido.

4)



4 Pelas alíneas anteriores temos que:
#P = 11 e #N = 9 Utilizamos class para denominar o output

$$IG(class|y_1) = I(class) - I(class|y_1)$$

$$I(class) = -\sum_{v_i \in class} p(v_i)\log(p(v_i)) =$$

$$= -\left(P(class=P)\log(P(class=P)) + P(class=N)\log(P(class=N))\right) =$$

$$= -\left(\frac{11}{20}\log\left(\frac{11}{20}\right) + \frac{9}{20}\log\left(\frac{9}{20}\right)\right) = 0,9928$$

$$I(class|y_1) = \sum \frac{|y_i|}{|y_1|} I(class, y_1 = y_i) =$$

$$= \frac{|y_1 = A|}{|y_1|} I(class, y_1 = A) + \frac{|y_1 = B|}{|y_1|} I(class, y_1 = B)$$

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

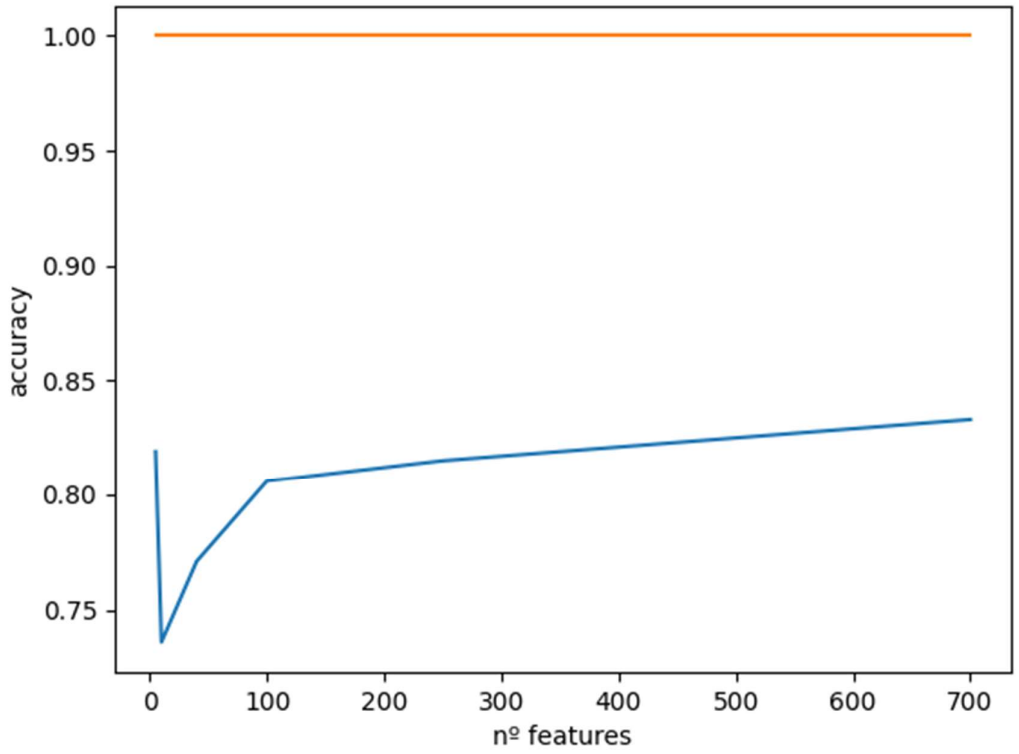
$$\frac{13}{20}\left(-\left(\frac{6}{13}\log(6/13) + \frac{7}{13}\log(7/13)\right)\right) =$$

$$= 0,9493$$

$$IG(class|y_1) = 0,9928 - 0,9493 = 0,0435$$

II. Programming and critical analysis

5)



6)

Como não estabelecemos nenhum limite para a profundidade da árvore de decisão, esta irá ramificar-se indefinidamente, adaptando-se perfeitamente ao training set, fazendo com que a training accuracy seja 1. Ao observarmos o gráfico verificamos que a precisão no test set é significativamente inferior à do training set (<1, como seria de esperar), pelo que podemos concluir que estamos perante um cenário de overfitting.

III. APPENDIX

```
#######         Importing required libraries          #######
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
from sklearn import tree, metrics
import matplotlib.pyplot as plt

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

#######      Creating the training-testing split      #######
X, y = df.drop('class', axis=1), df['class']
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, stratify=y,
random_state=1)

####### Feature ranking based on discriminative power #######
dp = mutual_info_classif(X, y, random_state=1)
dict1, count = {}, 0

for feature in X_train.columns.values:
    dict1[feature] = dp[count]
    count += 1

dict2 = dict(sorted(dict1.items(), key=lambda item: item[1], reverse=True))
features = list(dict2.keys())

#######   Running classifier and attesting results    #######
train, test = [], []
predictor = tree.DecisionTreeClassifier(random_state=1)

for n in [5, 10, 40, 100, 250, 700]:
    predictor.fit(X_train[features[0:n]], y_train)
    y_pred1 = predictor.predict(X_test[features[0:n]])
    y_pred2 = predictor.predict(X_train[features[0:n]])
    test.append(round(metrics.accuracy_score(y_test, y_pred1), 3))
    train.append(round(metrics.accuracy_score(y_train, y_pred2), 3))

#######               Creating plot                 #######
plt.plot([5, 10, 40, 100, 250, 700], test)
plt.plot([5, 10, 40, 100, 250, 700], train)
plt.xlabel('nº features')
plt.ylabel('accuracy')
plt.show()
```

END