Apprendizagem 2021/22

Homework I - Group 103

Henrique Anjos 99081

Vasco Vaz 99133

I. Pen-and-paper

① confusion matrix

		predicted +	predicted −
true	+	8	3
	−	4	5

$\# + = 11$

$\# - = 9$

②

		predicted +	predicted −
true	+	5	6
	−	2	7

sensitivity (depth 1)

$$\underset{depth=1}{S(class+)} = \frac{5}{11}$$

precision (depth 1)

$$\underset{depth=1}{P(class+)} = \frac{5}{7}$$

$$\underset{depth=1}{F_1(class+)} = \frac{2}{\frac{1}{P}+\frac{1}{S}} = \frac{2}{\frac{11}{5}+\frac{7}{5}} = \frac{2}{\frac{18}{5}} = \frac{10}{18} = \frac{5}{9}$$

③ • Não obrigar ir até entropia = 0 para combater Overfitting

• O aumento do Information Gain do ramo espandido não era significativo

④ $P(class+) = \frac{11}{20}$ $\quad P(class-) = \frac{9}{20}$ $\quad P(y_1=A) = \frac{7}{20}$

$E(class) = -\frac{11}{20}\log_2\frac{11}{20} - \frac{9}{20}\log_2\frac{9}{20} = 0.99277$ $\quad P(y_1=B) = \frac{13}{20}$

$P(class+ \mid y_1=A) = \frac{5}{7}$ $\quad P(class+ \mid y_1=B) = \frac{6}{13}$

$P(class- \mid y_1=A) = \frac{2}{7}$ $\quad P(class- \mid y_1=B) = \frac{7}{13}$

$E(class \mid y_1=A) = -\frac{5}{7}\log_2\frac{5}{7} - \frac{2}{7}\log_2\frac{2}{7} = 0.86312$

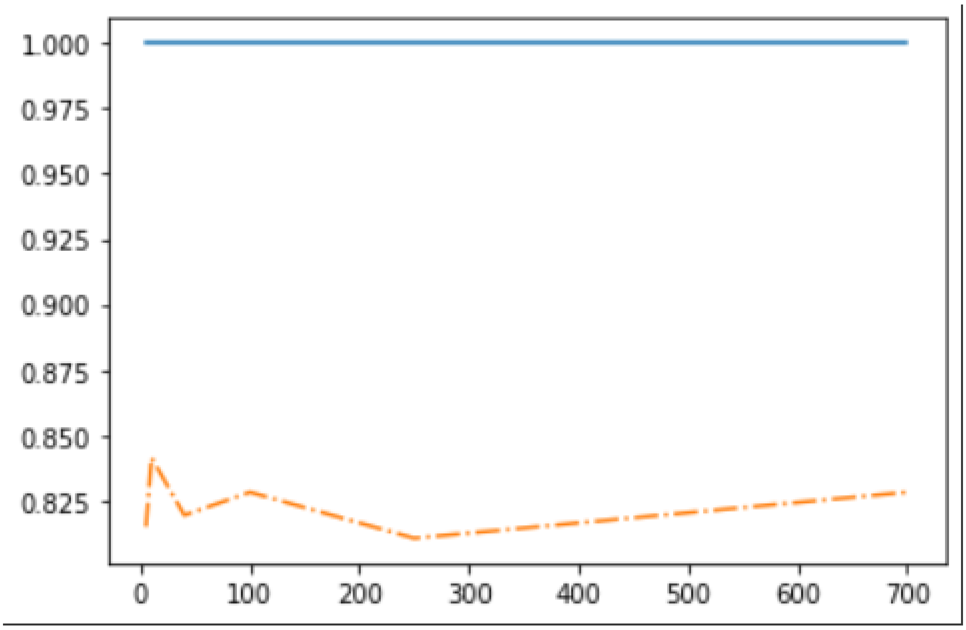
$E(class \mid y_1=B) = -\frac{6}{13}\log_2\frac{6}{13} - \frac{7}{13}\log_2\frac{6}{13} = 0.99573$

$E(class/y_1) = P(y_1=A)\cdot E(class)_{y_1=A}) + P(y_1=B)\cdot E(class \mid y_1=B) =$
$= 0.94932$

$Gain(y_1) = E(class) - E(class \mid y_1) = 0.04346$

II.Programming

1)




```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.io.arff import loadarff
from IPython.display import display
from pandas import DataFrame
from sklearn import metrics
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest, mutual_info_clas
sif
from sklearn.datasets import load_iris

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

selected_features = [5,10,40,100,250,700]
train_accuracy = []
test_accuracy = []

for feature in selected_features:
    X_train, X_test, y_train, y_test = train_test_split(df.drop(col
umns='class'), df["class"],
                            train_size = 0.7, test_size = 0.3, rand
om_state = 1, stratify= y)
```

```
    estimator = tree.DecisionTreeClassifier()
    estimator.fit(X_train, y_train)

    train_predict = estimator.predict(X_train)
    test_predict = estimator.predict(X_test)

    train_accuracy.append(metrics.accuracy_score(y_train, train_pre
dict))
    test_accuracy.append(metrics.accuracy_score(y_test, test_predic
t))

    feature_classifier = SelectKBest(mutual_info_classif, k=feature
)
    feature_classifier.fit_transform(X_train, y_train)
    best_features = feature_classifier.get_support(indices=True)

    X_train = X_train.iloc[:,best_features]
    X_test = X_test.iloc[:,best_features]

plt.plot(np.array(selected_features), np.array(train_accuracy))
plt.plot(np.array(selected_features), np.array(test_accuracy), '-
.')

plt.show()
```

2) Training accuracy sempre de 1 significa que a decision tree tem sempre uma resposta correta. Isto acontece porque os dados que são usados para testar a decision tree já foram previamente usados para a treinar. E, portanto, o output que a decision tree for prever vai ser sempre verdade.