Aprendizagem 2022/23

Homework I – Group 018

I. Pen-and-paper

1) [4v] Draw the training confusion matrix.

		Previstos	
		P	N
Reais	P	8	3
	N	4	5

$\#P_{reais} = 11$, $\#N_{reais} = 9$,

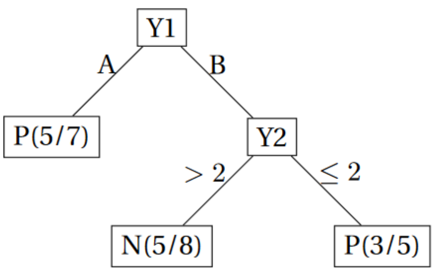
$\#P_{previstos} = 12$, $\#N_{previstos} = 8$, $\#total = 20$.

2) [3v] Identify the training F1 after a post-pruning of the given tree under a maximum depth of 1.

- $F1 = F\,measure_1 = F_{\beta=1} = F_{\alpha=0.5} = \frac{1}{\frac{1}{2}\frac{1}{P}+\frac{1}{2}\frac{1}{R}} = \frac{2}{\frac{1}{P}+\frac{1}{R}}$;

$$accuracy = \frac{\#bem\ classificados}{\#total\ classificados}, \qquad recall/sensitivity = \frac{TP}{TP+FN}, \qquad precision = \frac{TP}{TP+FP}.$$

Decision tree original:



$$Precision_P = \frac{8}{8+4} = \frac{8}{12},$$

$$Recall_P = \frac{8}{8+3} = \frac{8}{11}, \qquad F1_P = \frac{2}{\frac{12}{8}+\frac{11}{8}} = \frac{16}{23}.$$

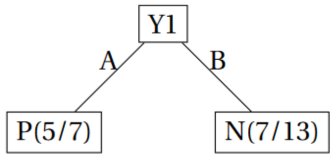
$$Precision_N = \frac{5}{5+3} = \frac{5}{8},$$

$$Recall_N = \frac{5}{4+5} = \frac{5}{9}, \qquad F1_N = \frac{2}{\frac{8}{5}+\frac{9}{5}} = \frac{10}{17}.$$

TÉCNICO LISBOA

Aprendizagem 2022/23
Homework I – Group 018

Decision tree depois do post-pruning:



Previstos

		P	N
Reais	P	5	6
	N	2	7

- confusion matrix da nova decision tree

$Precision_P = \frac{5}{5+2} = \frac{5}{7}$,

$Recall_P = \frac{5}{5+6} = \frac{5}{11}$, $\quad F1_P = \frac{2}{\frac{7}{5}+\frac{11}{5}} = \frac{5}{9}$.

$Precision_N = \frac{7}{7+6} = \frac{7}{13}$,

$Recall_N = \frac{7}{7+2} = \frac{7}{9}$, $\quad F1_N = \frac{2}{\frac{13}{7}+\frac{9}{7}} = \frac{7}{11}$;

3) [2v] Identify two different reasons as to why the left tree path was not further decomposed.

O caminho esquerdo da árvore não foi mais desenvolvido para evitar overfitting de duas formas (pelo menos):

1. Sendo o data split não estatisticamente significante, ao pararmos a expansão do nó, evitamos a produção de filhos que se baseiam em samples muito pequenos;
2. A expansão do nó não respeitaria o valor mínimo de diminuição do seu nível de impureza (definida no parâmetro *min_impurity_decrease* na função *sklearn.tree.DecisionTreeClassifier*, com default = 0).

4) [3v] Compute the information gain of variable y1.

$IG(out \mid y1) = H(out) - H(out \mid y1)$;

$H(out) = -\sum_{v \in out} P(v)\, log_2(P(v))$;

$H(out \mid y1) = -\sum_{v \in y1} P(y1 = v)\, H(out \mid y1 = v)$, $\quad$ *sendo out a variável de output.*

- $H(out) = -\frac{11}{20} log_2(\frac{11}{20}) - \frac{9}{20} log_2(\frac{9}{20}) \simeq 0.99277$
- $H(out \mid y1) = \frac{7}{20}[-\frac{5}{7} log_2(\frac{5}{7}) - \frac{2}{7} log_2(\frac{2}{7})] + \frac{13}{20}[-\frac{6}{13} log_2(\frac{6}{13}) - \frac{7}{13} log_2(\frac{7}{13})] \simeq 0.94932$

Logo, $IG(out \mid y1) = H(out) + H(out \mid y1) = 0.99277 - 0.94932 = 0.04345$ ❐

Aprendizagem 2022/23
Homework I – Group 018

II. Programming and critical analysis

5) Código python:

```python
# Import wall

import pandas as pd

import numpy as np

from sklearn import metrics, datasets, tree

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


# Reads the 'pd_speech.arff' file and creates the desired data frame.

data = loadarff('pd_speech.arff')

df = pd.DataFrame(data[0])


# Discretization of the output variable 'class' (B1 -> 1).

df['class'] = df['class'].str.decode('utf-8')


# Defines the X and Y data sets:

x = df.drop("class", axis=1)

y = np.ravel(df['class'])


num_features = [5, 10, 40, 100, 250, 700]

acc_train = []

acc_test = []


for n in num_features:

    # Feature selection:

    #  - selects the k best variables based on the

    #    mutual info classifier (information gain).

    selector = SelectKBest(mutual_info_classif, k=n)
```

Continuação (II. 5):

```
    # Gets the sub-set to be used on this iteration,
    # based on the selector created above.
    x_reduced = selector.fit_transform(x, y)

    # 70-30 training-testing split.
    x_train, x_test, y_train, y_test = train_test_split(x_reduced, y, test_size = 0.3,
                                                                      random_state=1)

    # Trains the decision tree on the training sets.
    decision_tree = tree.DecisionTreeClassifier()
    predictor = decision_tree.fit(x_train, y_train)

    # Gets the predicted values for both the train
    # and test samples.
    y_train_pred = predictor.predict(x_train)
    y_test_pred = predictor.predict(x_test)

    # Test model performance:
    # - calculates the accuracy of both samples,
    #   train and test.
    acc_train += [round(metrics.accuracy_score(y_train, y_train_pred), 2)]
    acc_test += [round(metrics.accuracy_score(y_test, y_test_pred), 2)]

print(acc_train)
print(acc_test)
```

output:

```
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
[0.75, 0.75, 0.84, 0.81, 0.82, 0.79]
```

Aprendizagem 2022/23

Homework I – Group 018

6) [2v] Why training accuracy is persistently 1? Critically analyze the gathered results.

Pela sua definição, accuracy corresponde ao número de previsões corretas a dividir pelo número total de previsões.
No caso concreto da accuracy de treino, todas as previsões serão corretas (dando sempre accuracy = 1), uma vez que são utilizadas tanto para treinar como para testar o próprio modelo usado no training set (70%).
Da análise dos resultados obtidos, podemos concluir que os modelos criados se adequam ao nosso dataset, com uma accuracy média de 80% (com um training-testing split 70-30).