Aprendizagem 2022

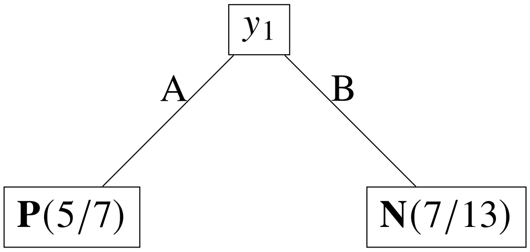
Homework I – Group 58

Part I: Pen and paper

1.

		prediction outcome	
		P	**N**
actual value	**P**	8	3
	N	4	5

2. Post-pruning of the given tree under a maximum depth of 1:



$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad Precision = \frac{\#T_{positives}}{\#T_{positives} + \#F_{positives}} \quad Recall = \frac{\#T_{positives}}{\#T_{positives} + \#F_{negatives}}$$

$$Precision_N = \frac{7}{7+6} = \frac{7}{13} \quad Recall_N = \frac{7}{7+2} = \frac{7}{9} \quad F1_N = \frac{\frac{7}{13} * \frac{7}{9}}{\frac{7}{13} + \frac{7}{9}} = \frac{7}{11}$$

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

3. One reason why the left tree path was not further decomposed is because in the data set, whenever the variable y_1 had the value A, the class was always P, so there was no need to have more nodes.

   Another reason is due to reducing the tree complexity and increasing its efficiency and speed and, therefore, predictive power.

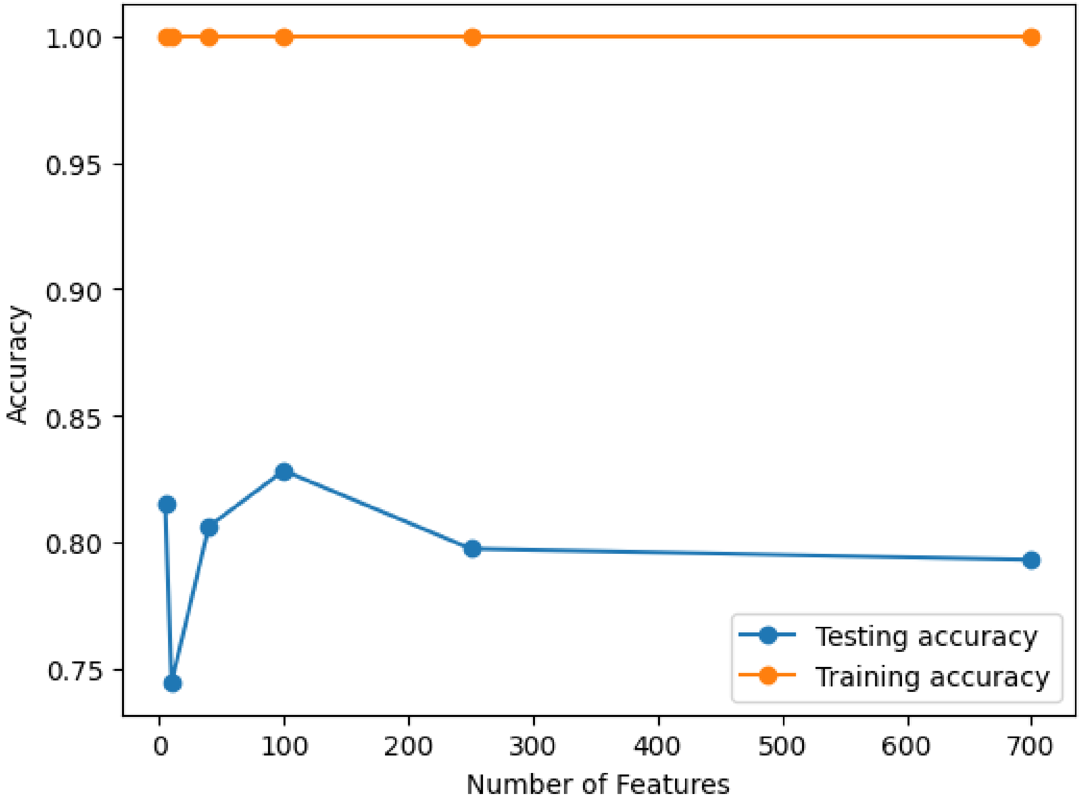
4. $IG(class \mid y_1) = E(class) - E(class \mid y_1)$

   $E(class) = -\sum_{x \in class} P(class = x) * \log_2[P(class = x)]$
   $= -(\frac{11}{20} * \log_2 \frac{11}{20} + \frac{9}{20} * \log_2 \frac{9}{20})$
   $\simeq 0.992774$

$E(class|y_1) = \sum_{x \in y_1} P(y_1 = x) * E(class|y_1 = x)$
$= \frac{7}{20}(-[\frac{5}{7} * \log_2 \frac{5}{7} + \frac{2}{7} * \log_2 \frac{2}{7}]) + \frac{13}{20}(-[\frac{6}{13} * \log_2 \frac{6}{13} + \frac{7}{13} * \log_2 \frac{7}{13}])$
$\simeq 0.949315$

$IG(class|y_1) = 0.992774 - 0.949315 \simeq 0.042459$

Part II: Programming

5. 

6. The training accuracy is persistently 1 because the data model is trained too well on our training data set, therefore, it predicts the correct values 100% of the time on that data set.

Appendix

```
from matplotlib import markers
import pandas as pd
from scipy.io.arff import loadarff
from sklearn.feature_selection import SelectKBest, mutual_info_classif
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
num_feat = [5, 10, 40, 100, 250, 700]
test_acc = []
train_acc = []

data = loadarff(’pd_speech.arff’)
df = pd.DataFrame(data[0])
df[’class’] = df[’class’].str.decode(’utf-8’)
X = df.drop(’class’, axis=1)
y = df[’class’]

for i in num_feat:
    X_new = SelectKBest(mutual_info_classif, k=i).fit_transform(X, y)
    X_train, X_test, y_train, y_test = train_test_split(X_new, y, train_size = 0.7,
     stratify = y,random_state = 1)
    classifier = DecisionTreeClassifier(random_state=1)
    classifier = classifier.fit(X_train, y_train)

    pred = classifier.predict(X_test)
    test_acc.append(metrics.accuracy_score(y_test, pred))
    train_acc.append(classifier.score(X_train, y_train))

fig, ax = plt.subplots()

ax.plot(num_feat, test_acc, label=’Testing accuracy’, marker=’o’)
ax.plot(num_feat, train_acc, label=’Training accuracy’, marker=’o’)
ax.set_xlabel("Number of Features")
ax.set_ylabel("Accuracy")
ax.legend()

plt.show()
```