TÉCNICO LISBOA

Aprendizagem 2022/23

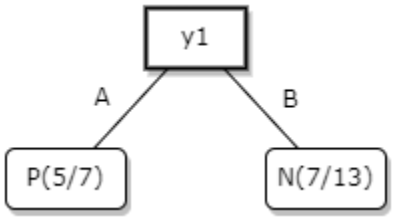
Homework I - Group XXX

I. Pen-and-paper

1) The training confusion matrix of the decision tree is as follows:

		Observed		
		P	**N**	**Total**
Predicted	**P**	**8**	**4**	**12**
	N	**3**	**5**	**8**
	Total	**11**	**9**	**20**

2) After a post-pruning of the given tree, we get the following decision tree:



Having this tree in mind, we can draw its confusion tree to get the relevant information to calculate the recall and precision.

		Observed		
		P	**N**	**Total**
Predicted	**P**	**5**	**2**	**7**
	N	**6**	**7**	**13**
	Total	**11**	**9**	**20**

We can now calculate recall, precision and, with both these metrics, calculate F_1.

$Recall = \frac{TP}{TP + FN} = \frac{5}{5+6} = \frac{5}{11}$

$Precision = \frac{TP}{TP + FP} = \frac{5}{5+2} = \frac{5}{7}$

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2 \times \frac{5}{7} \times \frac{5}{11}}{\frac{5}{7} + \frac{5}{11}} = \frac{5}{9}$$

As such, the F_1 of the given tree after a post-pruning of depth 1 is $\frac{5}{9}$

3) One of the reasons why the left tree path wasn't decomposed might be because the data necessary to create another path isn't statistically significant and as such we cannot decompose any further that path. Another reason for stopping the left path decomposition might be to avoid overfitting the decision tree to the training data.

4) The formula for the information gain goes as follows:

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

As such, we compute both $H(class)$ and $H(class|y_1)$.

$$H(class) = -\sum_{x \in class} P(class = x) log_2(P(class = x) = -[\frac{11}{20} log_2(\frac{11}{20}) + \frac{9}{20} log_2(\frac{9}{20})] = \frac{5}{9}$$

$$H(class|y_1) = \sum_{k \in y_1} P(y_1 = k) H(class|y_1 = k) =$$

$$= -\frac{7}{20}[\frac{5}{7} log_2(\frac{5}{7}) + \frac{2}{7} log_2(\frac{2}{7})] - \frac{13}{20}[\frac{6}{13} log_2(\frac{6}{13}) + \frac{7}{13} log_2(\frac{7}{13})]$$

After replacing the obtained values of $H(class)$ and $H(class|y_1)$ in the information gain formula we get the following result:

$$IG(class|y_1) \simeq 0.04346$$

II. Programming and critical analysis

1) Solution is provided in the Appendix (1) and (2).

2) When we train a decision tree, the model will create leaves until it can classify all of the training data correctly or until it reaches the maximum depth. As we don´t put a maximum depth limit, the decision tree will keep on creating leaves until all of the training data is classified correctly. As a result, the model is trained to perfectly classify the training data (overfitting), and as such, when we test the training accuracy, we will always get a score of 1.

We can also see that the test accuracy goes constantly up and down and converges around the 80% mark. This is another indication that the trained model is overfitted for the training data, and as such, it struggles in predicting the labels of the test dataset.

III. APPENDIX

1)

```
import pandas as pd
from scipy.io.arff import loadarff
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_selection import SelectKBest, mutual_info_classif
from sklearn.metrics import accuracy_score

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

# Splitting the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.drop('class', axis=1),
df['class'], train_size=0.7, test_size=0.3, random_state=1, stratify=df['class'])

# Creating list of the number of features to be selected
n_features = [5, 10, 40, 100, 250, 700]

# Creating a list of the training and testing accuracies
train_acc = []
test_acc = []

# Create criteria for selecting the best features
def criteria(X, y):
        return mutual_info_classif(X, y, random_state=1)

# Loop to calculate the training and testing accuracies
for n in n_features:
        # Selecting the n best features
        selector = SelectKBest(score_func=criteria, k=n)
        selector.fit(X_train, y_train)
        X_train_selected = selector.transform(X_train)
        X_test_selected = selector.transform(X_test)
```

```
        # Training decision tree
        tree = DecisionTreeClassifier(random_state=1)
        tree.fit(X_train_selected, y_train)
        # Computing the accuracies
        train_acc.append(accuracy_score(y_train, tree.predict(X_train_selected)))
        test_acc.append(accuracy_score(y_test, tree.predict(X_test_selected)))

# Plotting accuracies
import matplotlib.pyplot as plt
plt.plot(n_features, train_acc, label='Training accuracy')
plt.plot(n_features, test_acc, label='Testing accuracy')
plt.xlabel('Number of features')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

2)