

Aprendizagem 2023
Homework I – Group 28

Gonçalo Bárias (ist1103124) & Raquel Braunschweig (ist1102624)

Part I: Pen and Paper

Consider the partially learnt decision tree from the dataset D . D is described by four input variables – one numeric with values in $[0, 1]$ and 3 categorical – and a target variable with three classes.

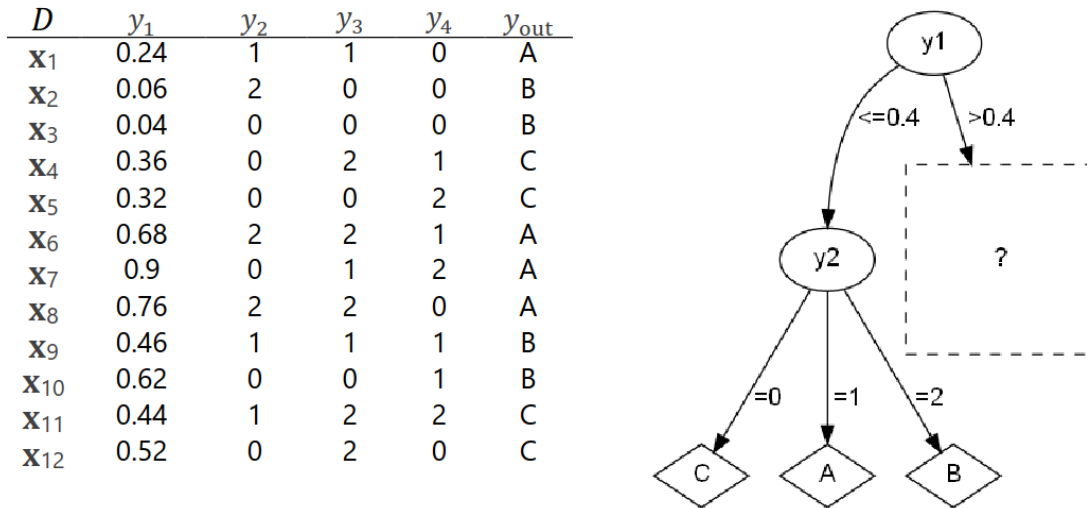


Figure 1: Partially Learnt Decision Tree and Dataset D from Part I

1. Complete the given decision tree using Information gain with Shannon entropy (\log_2). Consider that: i) a minimum of 4 observations is required to split an internal node, and ii) decisions by ascending alphabetic order should be placed in case of ties.

Blah

2. Draw the training confusion matrix for the learnt decision tree.

Blah

3. Identify which class has the lowest training F1 score.

Blah

4. Considering y_2 to be ordinal, assess if y_1 and y_2 are correlated using the Spearman coefficient.

Blah

5. Draw the class-conditional relative histograms of y_1 using 5 equally spaced bins in $[0, 1]$. Find the root split using the discriminant rules from these empirical distributions.

Blah

Part II: Programming

Consider the `column_diagnosis.arff` data available at the homework tab, comprising 6 biomechanical features to classify 310 orthopaedic patients into 3 classes (normal, disk hernia, spondilolysthesis).

1. Apply `f_classif` from `sklearn` to assess the discriminative power of the input variables. Identify the input variable with the highest and lowest discriminative power. Plot the class-conditional probability density functions of these two input variables.

```
1 import numpy as np, matplotlib.pyplot as plt, pandas as pd
2 from scipy.io.arff import loadarff
3 from sklearn.feature_selection import f_classif
4
5 # Read the ARFF file and prepare data
6 data = loadarff("./data/column_diagnosis.arff")
7 df = pd.DataFrame(data[0])
8 df["class"] = df["class"].str.decode("utf-8")
9 X, y = df.drop("class", axis=1), df["class"]
10
11 # Apply f_classif
12 f_scores, p_values = f_classif(X, y)
13
14 # Obtains the variables with the highest and lowest discriminative power.
15 highest_discriminative_power_idx = np.argmax(f_scores)
16 lowest_discriminative_power_idx = np.argmin(f_scores)
17
18 highest_discriminative_power_variable = X.columns[
19     highest_discriminative_power_idx
20 ]
21 lowest_discriminative_power_variable = X.columns[
22     lowest_discriminative_power_idx
23 ]
24
25 # Identifies the input variables requested
26 print(
27     f"Highest discriminative power variable: {
28         highest_discriminative_power_variable}"
29 )
30 print(
31     f"Lowest discriminative power variable: {lowest_discriminative_power_variable}"
32 )
33 plt.figure(figsize=(10, 6))
34
35 # Plot for the highest discriminative power variable
36 for class_label in np.unique(y):
37     class_data = X.loc[y == class_label, highest_discriminative_power_variable]
38     density, bins = np.histogram(class_data, bins=20, density=True)
39     plt.plot(
40         bins[:-1],
41         density,
42         label=f"Class {class_label} - {highest_discriminative_power_variable}",
43         linewidth=2,
44     )
```

```

45
46 # Plot for the lowest discriminative power variable
47 for class_label in np.unique(y):
48     class_data = X.loc[y == class_label, lowest_discriminative_power_variable]
49     density, bins = np.histogram(class_data, bins=20, density=True)
50     plt.plot(
51         bins[:-1],
52         density,
53         linestyle="--",
54         label=f"Class {class_label} - {lowest_discriminative_power_variable}",
55         linewidth=2,
56     )
57
58 plt.xlabel("Value")
59 plt.ylabel("Density")
60
61 plt.legend()
62 plt.grid(True)
63 plt.savefig("./report/class_conditional_probability.svg")
64 plt.show()

```

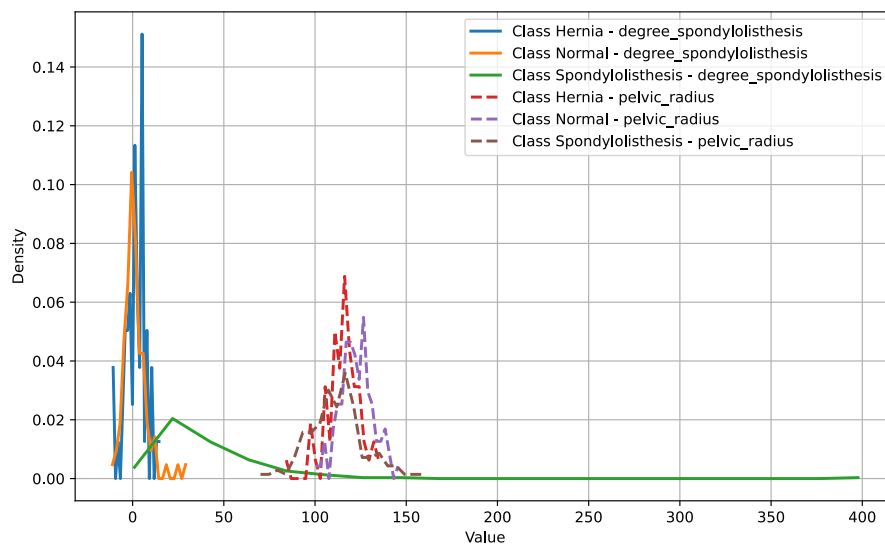


Figure 2: Class-conditional probability density functions of the highest and lowest discriminative power variables.

2. Using a stratified 70-30 training-testing split with a fixed seed (`random_state=0`), assess in a single plot both the training and testing accuracies of a decision tree with depth limits in $\{1, 2, 3, 4, 5, 6, 8, 10\}$ and the remaining parameters as default.

[Optional] Note that split thresholding of numeric variables in decision trees is non-deterministic in sklearn, hence you may opt to average the results using 10 runs per parameterization.

```

1 import pandas as pd, matplotlib.pyplot as plt, numpy as np
2 from scipy.io.arff import loadarff
3 from sklearn import metrics, tree
4 from sklearn.model_selection import train_test_split

```

```

5
6 # Read the ARFF file and prepare data
7 data = loadarff("./data/column_diagnosis.arff")
8 df = pd.DataFrame(data[0])
9 df["class"] = df["class"].str.decode("utf-8")
10 X, y = df.drop("class", axis=1), df["class"]
11
12 DEPTH_LIMIT = [1, 2, 3, 4, 5, 6, 8, 10]
13 training_accuracy, test_accuracy = [], []
14
15 # Split the dataset into a testing set (30%) and a training set (70%)
16 X_train, X_test, y_train, y_test = train_test_split(
17     X, y, test_size=0.3, stratify=y, random_state=0
18 )
19
20 for depth_limit in DEPTH_LIMIT:
21     # Create and fit the decision tree classifier
22     predictor = tree.DecisionTreeClassifier(
23         max_depth=depth_limit, random_state=0
24     )
25     predictor.fit(X_train, y_train)
26
27     # Use the decision tree to predict the outcome of the given observations
28     y_train_pred = predictor.predict(X_train)
29     y_test_pred = predictor.predict(X_test)
30
31     # Get the accuracy of each test
32     train_acc = metrics.accuracy_score(y_train, y_train_pred)
33     test_acc = metrics.accuracy_score(y_test, y_test_pred)
34
35     training_accuracy.append(train_acc)
36     test_accuracy.append(test_acc)
37
38 plt.plot(
39     DEPTH_LIMIT,
40     training_accuracy,
41     label="Training Accuracy",
42     marker="+",
43     color="#f8766d",
44 )
45 plt.plot(
46     DEPTH_LIMIT,
47     test_accuracy,
48     label="Test Accuracy",
49     marker=".",
50     color="#00bfc4",
51 )
52
53 plt.xlabel("Depth Limit")
54 plt.ylabel("Accuracy")
55
56 plt.legend()
57 plt.grid(True)
58 plt.savefig("./report/training_testing accuracies.svg")
59 plt.show()

```

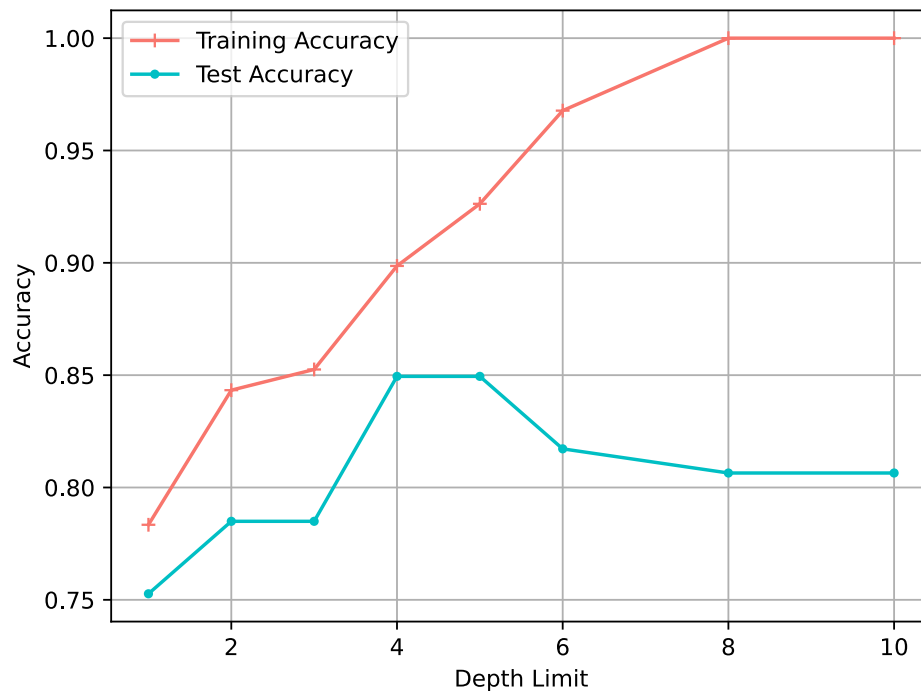


Figure 3: Accuracy of the trained decision tree, applied to both a test and training sets, for varying depth limits.

3. **Comment on the results, including the generalization capacity across settings.**

Blah

4. **To deploy the predictor, a healthcare team opted to learn a single decision tree (`random_state=0`) using *all* available data as training data, and further ensuring that each leaf has a minimum of 20 individuals in order to avoid overfitting risks.**

(a) **Plot the decision tree.**

Blah

(b) **Characterize a hernia condition by identifying the hernia-conditional associations.**

Blah

END