

Aprendizagem 2023
Homework I – Group 28

Gonalo Barias (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.

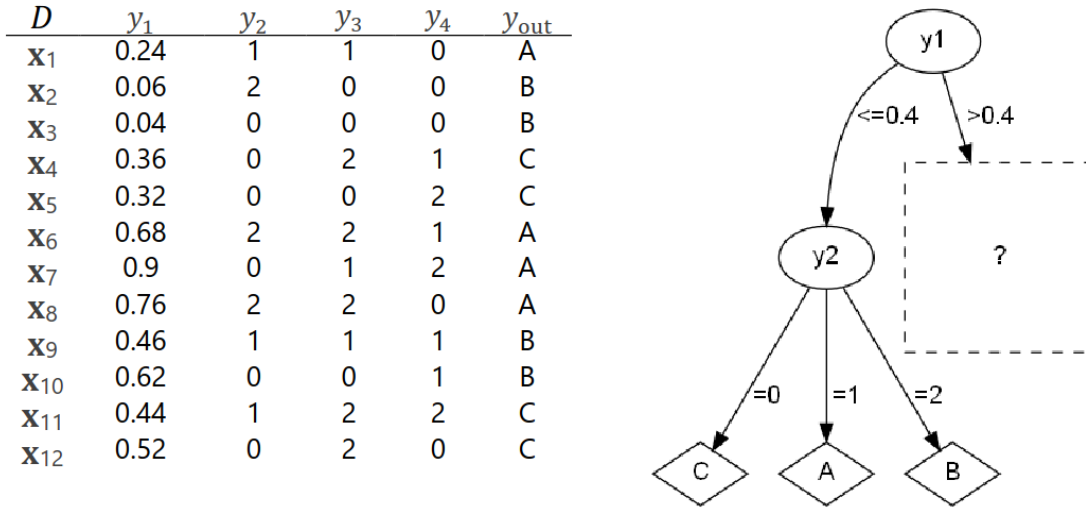


Figure 1: Partially Learnt Decision Tree and Dataset D from Part I

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.

The entropy of y_{out} is given by:

$$\begin{aligned}
 E(y_{out}|y_1 > 0.4) = & p(A, y_1 > 0.4) \log_2 (p(A, y_1 > 0.4)) \\
 & + p(B, y_1 > 0.4) \log_2 (p(B, y_1 > 0.4)) \\
 & + p(C, y_1 > 0.4) \log_2 (p(C, y_1 > 0.4))
 \end{aligned} \tag{1}$$

We can calculate $E(y_{out})$:

$$E(y_{out}) = - \left(\frac{3}{7} \log_2 \left(\frac{3}{7} \right) + \frac{2}{7} \log_2 \left(\frac{2}{7} \right) + \frac{2}{7} \log_2 \left(\frac{2}{7} \right) \right) = 1.5567$$

The next step is calculating $E(y_{out}|y_1 > 0.4, y_x)$, in which x will take the values of 2, 3 or 4:

$$\begin{aligned}
E(y_{out}|y_1 > 0.4, y_x) = & p(y_x = 0)E(y_{out}|y_x > 0.4, y_2 = 0) \\
& + p(y_x = 1)E(y_{out}|y_x > 0.4, y_2 = 1) \\
& + p(y_x = 2)E(y_{out}|y_x > 0.4, y_2 = 2)
\end{aligned} \tag{2}$$

And the information gain of variable y_x is given by

$$IG(y_x) = E(y_{out}) - E(y_{out}|y_1 > 0.4, y_x) \tag{3}$$

Let's start with x = 2:

$$\begin{aligned}
p(y_2 = 0, y_1 > 0.4) &= \frac{3}{7} \\
p(y_2 = 1, y_1 > 0.4) &= \frac{2}{7} \\
p(y_2 = 2, y_1 > 0.4) &= \frac{2}{7} \\
E(y_{out}|y_x > 0.4, y_2 = 0) &= -\left(\frac{1}{3} \log_2 \left(\frac{1}{3}\right) + \frac{1}{3} \log_2 \left(\frac{1}{3}\right) + \frac{1}{3} \log_2 \left(\frac{1}{3}\right)\right) = 1.5849 \\
E(y_{out}|y_x > 0.4, y_2 = 1) &= -\left(\frac{0}{2} \log_2 \left(\frac{0}{2}\right) + \frac{1}{2} \log_2 \left(\frac{1}{2}\right) + \frac{1}{2} \log_2 \left(\frac{1}{2}\right)\right) = 1 \\
E(y_{out}|y_x > 0.4, y_2 = 2) &= -\left(\frac{2}{2} \log_2 \left(\frac{2}{2}\right) + \frac{0}{2} \log_2 \left(\frac{0}{2}\right) + \frac{0}{2} \log_2 \left(\frac{0}{2}\right)\right) = 0
\end{aligned}$$

Therefore, replacing these values on equation (2), gives us:

$$\begin{aligned}
E(y_{out}|y_1 > 0.4, y_2) &= \frac{3}{7} \times 1.5849 + \frac{2}{7} \times 1 + \frac{2}{7} \times 0 \\
&= 0.965.
\end{aligned}$$

Finally, we can calculate the information gain, as per (3),

$$IG(y_2) = 1.5567 - 0.965 = 0.5917$$

Now, let's calculate for x = 3:

$$p(y_3 = 0, y_1 > 0.4) = \frac{1}{7}$$

$$p(y_3 = 1, y_1 > 0.4) = \frac{2}{7}$$

$$p(y_3 = 2, y_1 > 0.4) = \frac{4}{7}$$

$$E(y_{out}|y_x > 0.4, y_3 = 0) = -\left(\frac{0}{1} \log_2 \left(\frac{0}{1}\right) + \frac{1}{1} \log_2 \left(\frac{1}{1}\right) + \frac{0}{1} \log_2 \left(\frac{0}{1}\right)\right) = 0$$

$$E(y_{out}|y_x > 0.4, y_3 = 1) = -\left(\frac{1}{2} \log_2 \left(\frac{1}{2}\right) + \frac{1}{2} \log_2 \left(\frac{1}{2}\right) + \frac{0}{2} \log_2 \left(\frac{0}{2}\right)\right) = 1$$

$$E(y_{out}|y_x > 0.4, y_3 = 2) = -\left(\frac{2}{4} \log_2 \left(\frac{2}{4}\right) + \frac{0}{4} \log_2 \left(\frac{0}{4}\right) + \frac{2}{4} \log_2 \left(\frac{2}{4}\right)\right) = 1$$

Therefore, replacing these values on equation (2), gives us:

$$\begin{aligned} E(y_{out}|y_1 > 0.4, y_3) &= \frac{1}{7} \times 0 + \frac{2}{7} \times 1 + \frac{4}{7} \times 1 \\ &= 0.8571. \end{aligned}$$

Finally, we can calculate the information gain, as per (3),

$$IG(y_3) = 1.5567 - 0.8571 = 0.6996$$

Finally, let's calculate for x = 4:

$$p(y_4 = 0, y_1 > 0.4) = \frac{2}{7}$$

$$p(y_4 = 1, y_1 > 0.4) = \frac{3}{7}$$

$$p(y_4 = 2, y_1 > 0.4) = \frac{2}{7}$$

$$E(y_{out}|y_x > 0.4, y_4 = 0) = -\left(\frac{1}{2} \log_2 \left(\frac{1}{2}\right) + \frac{0}{2} \log_2 \left(\frac{0}{2}\right) + \frac{1}{2} \log_2 \left(\frac{1}{2}\right)\right) = 1$$

$$E(y_{out}|y_x > 0.4, y_4 = 1) = -\left(\frac{1}{3} \log_2 \left(\frac{1}{3}\right) + \frac{2}{3} \log_2 \left(\frac{2}{3}\right) + \frac{0}{3} \log_2 \left(\frac{0}{3}\right)\right) = 0.9183$$

$$E(y_{out}|y_x > 0.4, y_4 = 2) = -\left(\frac{1}{2} \log_2 \left(\frac{1}{2}\right) + \frac{0}{2} \log_2 \left(\frac{0}{2}\right) + \frac{1}{2} \log_2 \left(\frac{1}{2}\right)\right) = 1$$

Therefore, replacing these values on equation (2), gives us:

$$E(y_{out}|y_1 > 0.4, y_4) = \frac{2}{7} \times 1.5849 + \frac{3}{7} \times 1 + \frac{2}{7} \times 0 = 0.965.$$

Finally, we can calculate the information gain, as per (3),

$$IG(y_4) = 1.5849 - 0.965 = 0.5917$$

Upon computing the information gains for each attribute, it is evident that y_3 yields the highest value of 0.6996. Consequently, it is selected as the next node, resulting in the construction of the following decision tree:

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, spondilolsthesis).

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, pandas as pd, seaborn as sns, matplotlib.pyplot as plt
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, _ = f_classif(X, y)
13
14 # Obtains the variables with the highest and lowest discriminative power.
15 h_disc_power_var = X.columns[np.argmax(f_scores)]
16 l_disc_power_var = X.columns[np.argmin(f_scores)]
17
18 plt.figure(figsize=(8, 6))
19
20 # Plot for the highest discriminative power variable
21 for class_label in np.unique(y):
22     class_data = X.loc[y == class_label, h_disc_power_var]
```

```

23     sns.kdeplot(
24         class_data,
25         label=f"Class {class_label} - {h_disc_power_var}",
26         linewidth=2,
27     )
28
29 # Plot for the lowest discriminative power variable
30 for class_label in np.unique(y):
31     class_data = X.loc[y == class_label, l_disc_power_var]
32     sns.kdeplot(
33         class_data,
34         label=f"Class {class_label} - {l_disc_power_var}",
35         linestyle="--",
36         linewidth=2,
37     )
38
39 plt.xlabel("Variables")
40 plt.ylabel("Density")
41
42 plt.legend()
43 plt.grid(True)
44 plt.savefig("./report/class_conditional_probability.svg")
45 plt.show()

```

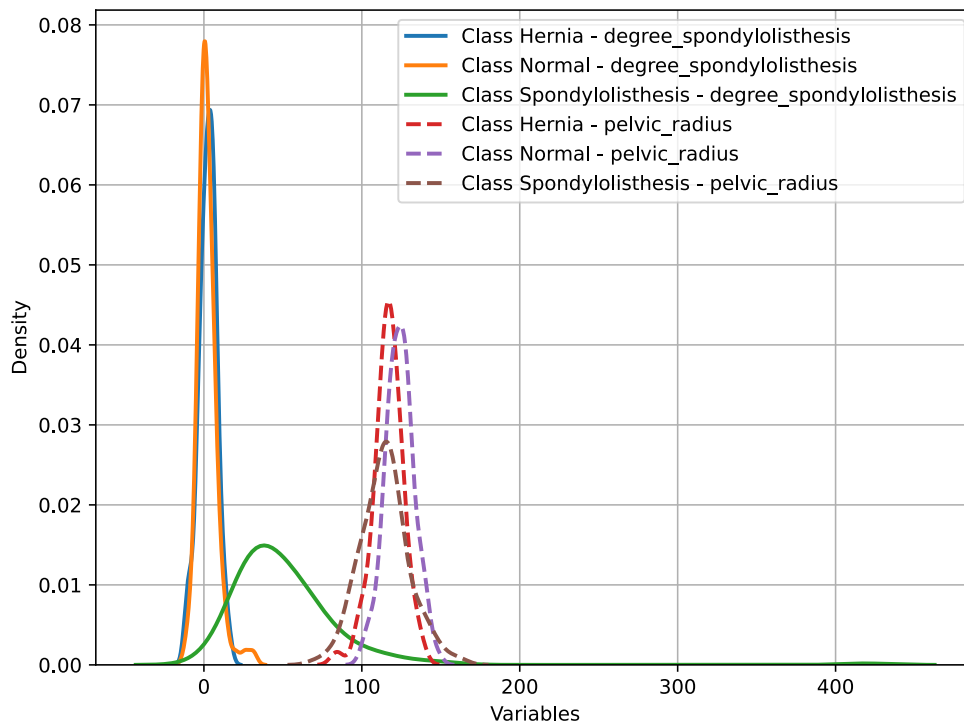


Figure 2: Class-conditional probability density functions of the highest and lowest discriminative power variables.

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
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     training_accuracy.append(train_acc)
34     test_acc = metrics.accuracy_score(y_test, y_test_pred)
35     test_accuracy.append(test_acc)
36
37 plt.plot(
38     DEPTH_LIMIT,
39     training_accuracy,
40     label="Training Accuracy",
41     marker="+",
42     color="#f8766d",
43 )
44 plt.plot(
45     DEPTH_LIMIT,
46     test_accuracy,
47     label="Test Accuracy",
48     marker=".",
49     color="#00bfc4",
50 )
51
```

```

52 plt.xlabel("Depth Limit")
53 plt.ylabel("Accuracy")
54
55 plt.legend()
56 plt.grid(True)
57 plt.savefig("./report/training_testing accuracies.svg")
58 plt.show()

```

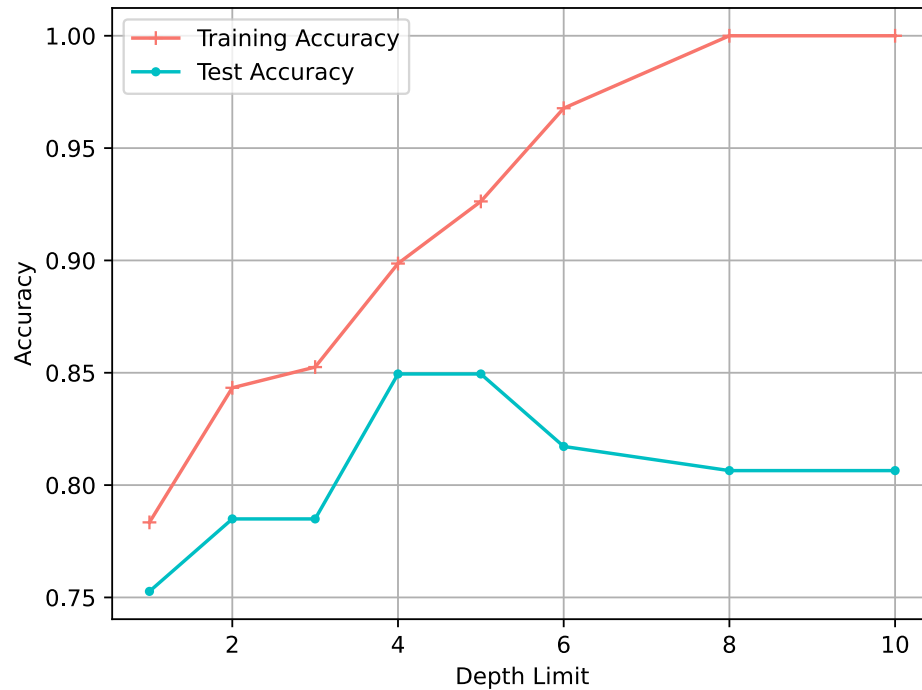


Figure 3: Accuracy of the trained decision tree, applied to both a test and training sets, for varying depth limits.

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.

```

1 import matplotlib.pyplot as plt, pandas as pd, numpy as np
2 from scipy.io.arff import loadarff
3 from sklearn.tree import DecisionTreeClassifier, plot_tree
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 # Create and train the decision tree classifier
12 clf = DecisionTreeClassifier(random_state=0, min_samples_leaf=20)
13 clf.fit(X, y)
14
15 # Set style and plot the decision tree
16 plt.figure(figsize=(15, 10))
17 plot_tree(clf, filled=True, feature_names=list(X.columns),
18           class_names=list(np.unique(y)), rounded=True, fontsize=10)
19 plt.savefig("./report/decision_tree.svg")
20 plt.show()

```

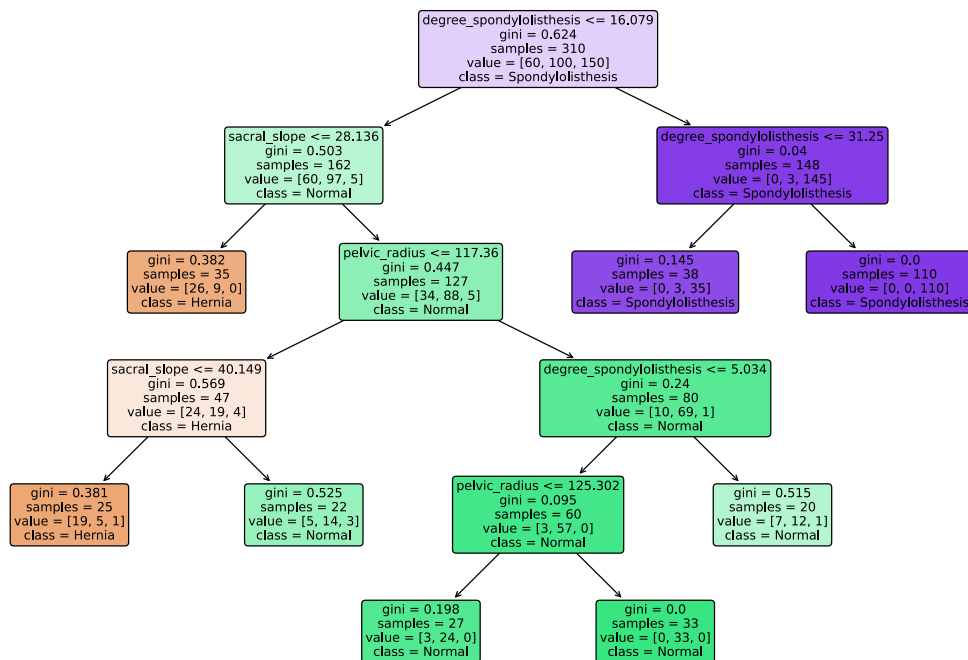


Figure 4: Decision Tree

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

Blah

END