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.

D	y_1	y_2	y_3	y_4	y_{out}
$\mathbf{x}_1$	0.24	1	1	0	A
$\mathbf{x}_2$	0.06	2	0	0	B
$\mathbf{x}_3$	0.04	0	0	0	B
$\mathbf{x}_4$	0.36	0	2	1	C
$\mathbf{x}_5$	0.32	0	0	2	C
$\mathbf{x}_6$	0.68	2	2	1	A
$\mathbf{x}_7$	0.9	0	1	2	A
$\mathbf{x}_8$	0.76	2	2	0	A
$\mathbf{x}_9$	0.46	1	1	1	B
$\mathbf{x}_{10}$	0.62	0	0	1	B
$\mathbf{x}_{11}$	0.44	1	2	2	C
$\mathbf{x}_{12}$	0.52	0	2	0	C

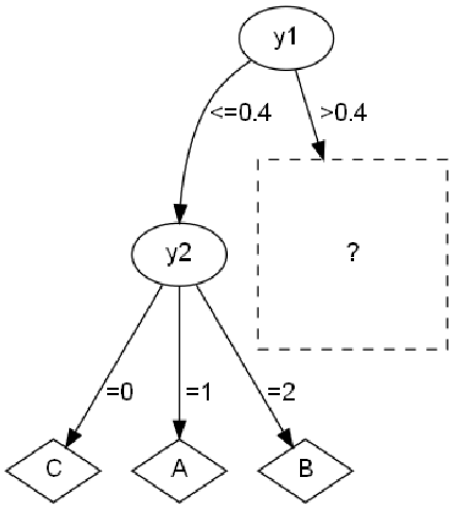

Figure 1: Partially Learnt Decision Tree and Dataset D from Part I

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:

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

   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:

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

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:

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

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

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

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:

$$
\begin{aligned}
p(y_3 = 0, y_1 > 0.4) &= \frac{1}{7} \\
p(y_3 = 1, y_1 > 0.4) &= \frac{2}{7} \\
p(y_3 = 2, y_1 > 0.4) &= \frac{4}{7} \\
E(y_{out}|y_x > 0.4, y_3 = 0) &= -\left(\frac{0}{1}\log_2\left(\frac{0}{1}\right) + \frac{1}{1}\log_2\left(\frac{1}{1}\right) + \frac{0}{1}\log_2\left(\frac{0}{1}\right)\right) = 0 \\
E(y_{out}|y_x > 0.4, y_3 = 1) &= -\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{0}{2}\log_2\left(\frac{0}{2}\right)\right) = 1 \\
E(y_{out}|y_x > 0.4, y_3 = 2) &= -\left(\frac{2}{4}\log_2\left(\frac{2}{4}\right) + \frac{0}{4}\log_2\left(\frac{0}{4}\right) + \frac{2}{4}\log_2\left(\frac{2}{4}\right)\right) = 1
\end{aligned}
$$

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

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

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:

$$p(y_4 = 0, y_1 > 0.4) = \frac{2}{7}$$
$$p(y_4 = 1, y_1 > 0.4) = \frac{3}{7}$$
$$p(y_4 = 1, y_1 > 0.4) = \frac{2}{7}$$
$$E(y_{out}|y_x > 0.4, y_4 = 0) = -\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{0}{2}\log_2\left(\frac{0}{2}\right) + \frac{1}{2}\log_2\left(\frac{1}{3}\right)\right) = 1$$
$$E(y_{out}|y_x > 0.4, y_4 = 1) = -\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{2}{3}\log_2\left(\frac{2}{3}\right) + \frac{0}{3}\log_2\left(\frac{0}{3}\right)\right) = 0.9183$$
$$E(y_{out}|y_x > 0.4, y_4 = 2) = -\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{0}{2}\log_2\left(\frac{0}{2}\right) + \frac{1}{2}\log_2\left(\frac{1}{2}\right)\right) = 1$$

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:

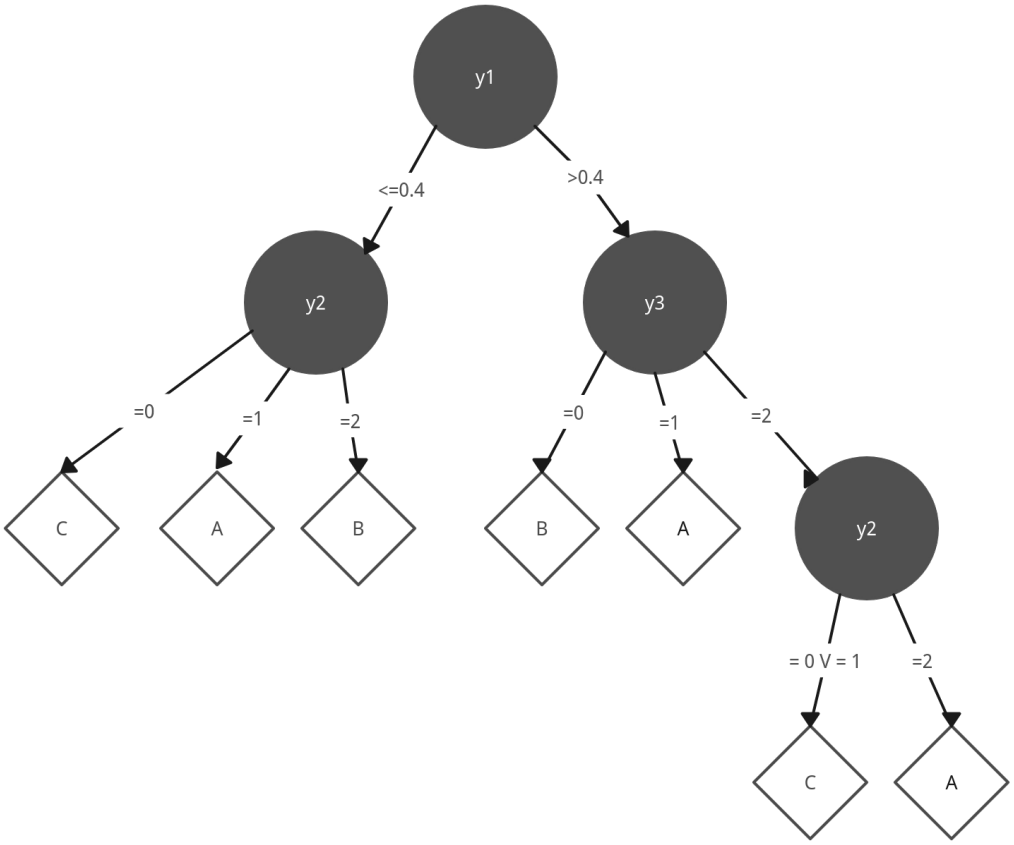

Figure 2: Decision Tree for exercise I.1

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

   Following the learnt decision tree above, we can predict the values for each observation. For each observation, we look at the value for the first variable (y_1) and follow the branch that corresponds with its value. From the node we arrive at, we do the same thing for the next variable, and we keep doing this until we reach a leaf. The class present in this leaf will be the predicted value, while the real value is the value of y_{out} for that observation. Below we present the real values along with the predicted ones:

		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
real	=	[A	B	B	C	C	A	A	A	B	B	C	C]
predicted	=	[A	B	C	C	C	A	A	A	A	B	C	C]

   Finally, we can show the count of each pair of real and predicted values in a confusion matrix (e.g. 4 pairs of AA from observations x_1, x_6, x_7, x_8):

		Real			
		A	**B**	**C**	
Predicted	**A**	4	1	0	5
	B	0	2	0	2
	C	0	1	4	5
		4	4	4	12

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

$F1_{score}$ is given by the following equation:

$$F1_{score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4}$$

And precision and recall are given by:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{5}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{6}$$

Therefore, **let's start by calculating the precision for A, B and C** by replacing the values on (5):

$$Precision_A = \frac{4}{4+1} = \frac{4}{5}$$

$$Precision_B = \frac{2}{2+0} = 1$$

$$Precision_C = \frac{4}{4+1} = \frac{4}{5}$$

Now, it's time to calculate the recalls for A, B and C, using the equation on (6):

$$Recall_A = \frac{4}{4+0} = 1$$

$$Recall_B = \frac{2}{2+2} = \frac{1}{2}$$

$$Recall_C = \frac{4}{4+0} = 1$$

Finally, let's calculate the $F1_{score}$, using the equation (4):

$$F1_{score}A = 2 \cdot \frac{\frac{4}{5} \cdot 1}{\frac{4}{5} + 1} = 0.8889$$

$$F1_{score}B = 2 \cdot \frac{\frac{1}{2} \cdot 1}{\frac{1}{2} + 1} = 0.6667$$

$$F1_{score}C = 2 \cdot \frac{1 \cdot \frac{4}{5}}{1 + \frac{4}{5}} = 0,8889$$

The class with the lowest training score is B, with a score of 0.6667.

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

   To calculate the Spearman coefficient when there's rank, we have to use the following equation:

$$\text{Spearman}(y_x, y_y) = \frac{\text{cov}(y_x, y_y)}{\sigma_{y_x}\sigma_{y_y}} = \frac{\sum_{i=1}^{n}(x_i - \bar{y_x})(y_i - \bar{y_y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{y_x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y_y})^2}} \tag{7}$$

   Firstly, **let's order y_1 and y_2** so we can calulate the ranks and y_1' and y_2':

$$\begin{aligned}
ordered_y_1 &= [0.04, 0.06, 0.24, 0.32, 0.36, 0.44, 0.46, 0.52, 0.62, 0.68, 0.76, 0.9] \\
ranks_y_1 &= [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] \\
y_1' &= [3, 2, 1, 5, 4, 10, 12, 11, 7, 9, 6, 8] \\
ordered_y_2 &= [0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 2] \\
ranks_y_2 &= [3.5, 3.5, 3.5, 3.5, 3.5, 3.5, 8, 8, 8, 11, 11, 11] \\
y_2' &= [8, 11, 3.5, 3.5, 3.5, 11, 3.5, 11, 8, 3.5, 8, 3.5]
\end{aligned}$$

   Now, we have all we need to calculate **the Spearman coefficient** using the expression at (7). Here is the result:

$$Spearman(y_1, y_2) = 0.07966$$

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.**

```python
import numpy as np, pandas as pd, seaborn as sns, matplotlib.pyplot as plt
from scipy.io.arff import loadarff
from sklearn.feature_selection import f_classif

# Read the ARFF file and prepare data
data = loadarff("./data/column_diagnosis.arff")
df = pd.DataFrame(data[0])
df["class"] = df["class"].str.decode("utf-8")
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()
```

As you can see in the graph ahead, the highest discriminative power variable is *degree_spondilolysthesis* and the lowest discriminative power variable is *pelvic_radius*.

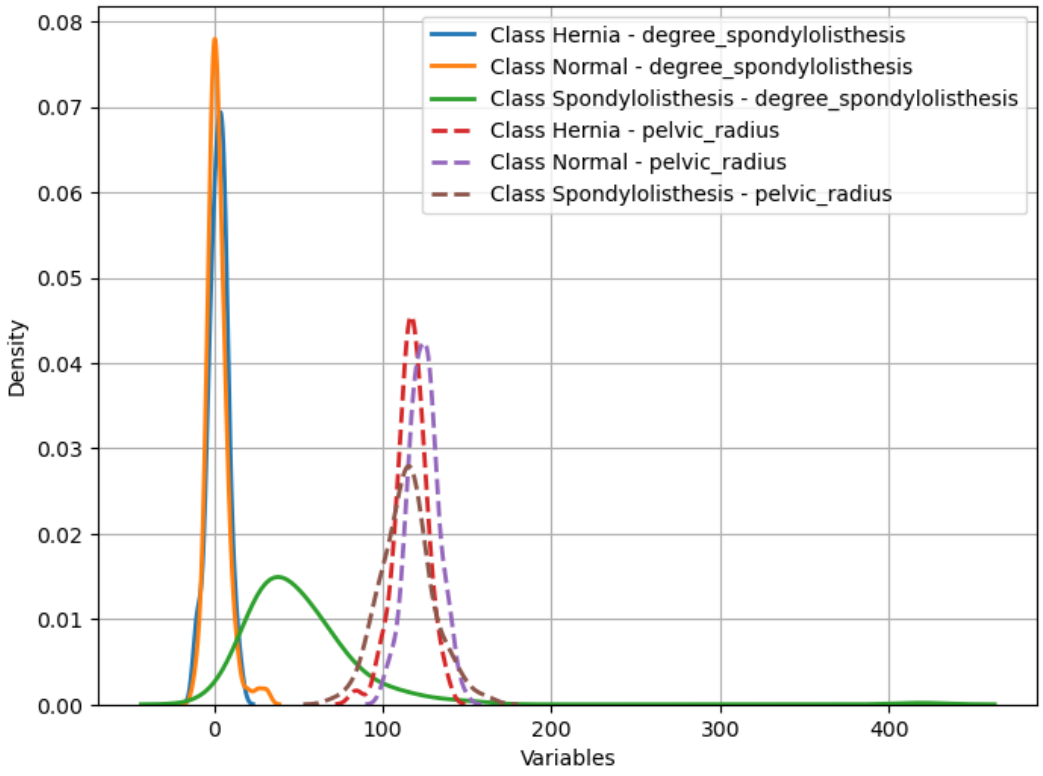

Figure 3: Class-conditional probability density functions of the highest and lowest discriminative power variables.

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.**

```python
import pandas as pd, matplotlib.pyplot as plt
from scipy.io.arff import loadarff
from sklearn import metrics, tree
from sklearn.model_selection import train_test_split

# Read the ARFF file and prepare data
data = loadarff("./data/column_diagnosis.arff")
df = pd.DataFrame(data[0])
df["class"] = df["class"].str.decode("utf-8")
X, y = df.drop("class", axis=1), df["class"]

DEPTH_LIMIT = [1, 2, 3, 4, 5, 6, 8, 10]
training_accuracy, test_accuracy = [], []

# Split the dataset into a testing set (30%) and a training set (70%)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=0
)

```

```
for depth_limit in DEPTH_LIMIT:
    # Create and fit the decision tree classifier
    predictor = tree.DecisionTreeClassifier(
        max_depth=depth_limit, random_state=0
    )
    predictor.fit(X_train, y_train)

    # Use the decision tree to predict the outcome of the given observations
    y_train_pred = predictor.predict(X_train)
    y_test_pred = predictor.predict(X_test)

    # Get the accuracy of each test
    train_acc = metrics.accuracy_score(y_train, y_train_pred)
    training_accuracy.append(train_acc)
    test_acc = metrics.accuracy_score(y_test, y_test_pred)
    test_accuracy.append(test_acc)

plt.plot(
    DEPTH_LIMIT,
    training_accuracy,
    label="Training Accuracy",
    marker="+",
    color="#f8766d",
)
plt.plot(
    DEPTH_LIMIT,
    test_accuracy,
    label="Test Accuracy",
    marker=".",
    color="#00bfc4",
)

plt.xlabel("Depth Limit")
plt.ylabel("Accuracy")

plt.legend()
plt.grid(True)
plt.savefig("./report/training_testing_accuracies.svg")
plt.show()
```

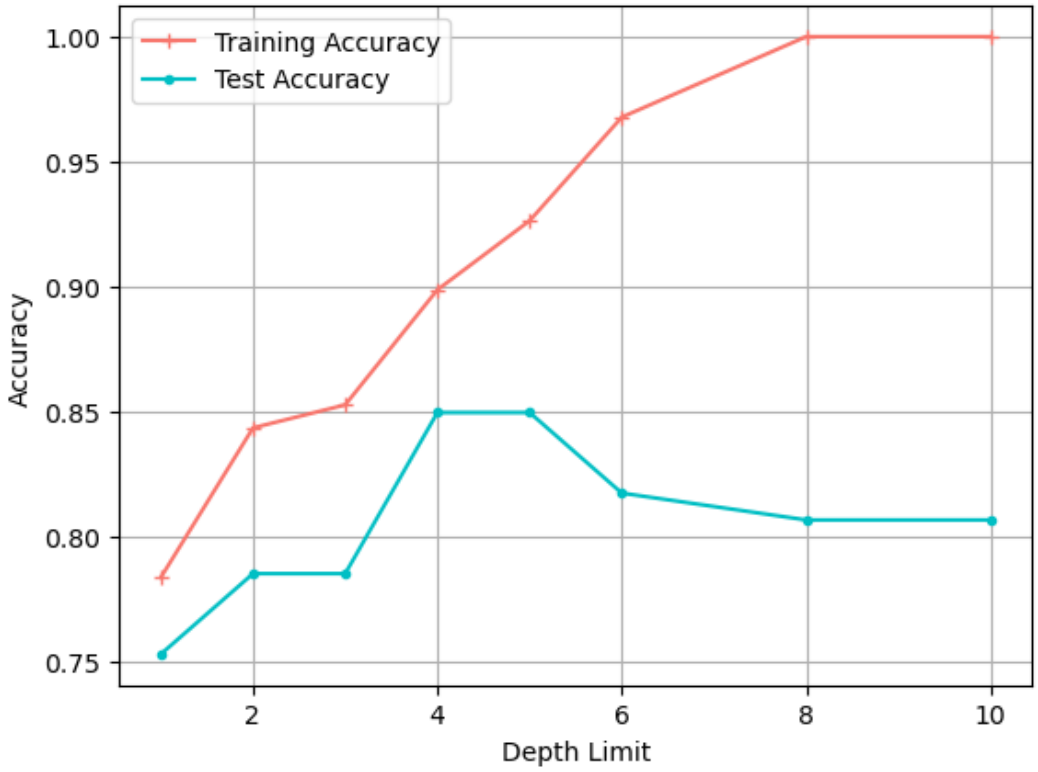

Figure 4: Accuracy of the trained decision tree, applied to both a test and training sets, for varying depth limits.

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

   The graphic illustrates that as the depth limit of the decision tree increases, the training accuracy steadily rises. This observation suggests that deeper trees can better fit the training data, capturing intricate patterns and achieving higher accuracy when evaluated on the same dataset. However, it's crucial to keep in mind that higher training accuracy doesn't necessarily translate to better predictive performance on unseen data.

   The testing accuracy follows a distinct pattern. Initially, it improves as the depth limit increases, indicating improved generalization. However, beyond a certain depth limit (around 4 or 5 in this case), the testing accuracy starts to decline. This signifies a loss in generalization capacity, a phenomenon known as overfitting. It implies that overly complex decision trees can fit noise in the training data and perform poorly on new, unseen data.

   The optimal depth limit appears to be around 4 or 5, striking a balance between model complexity and generalization to new data. It's crucial to avoid both underfitting (too simple) and overfitting (too complex) by selecting a depth limit that maximizes testing accuracy.

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.**

```python
import matplotlib.pyplot as plt, pandas as pd, numpy as np
from scipy.io.arff import loadarff
from sklearn.tree import DecisionTreeClassifier, plot_tree

# Read the ARFF file and prepare data
data = loadarff("./data/column_diagnosis.arff")
df = pd.DataFrame(data[0])
df["class"] = df["class"].str.decode("utf-8")
X, y = df.drop("class", axis=1), df["class"]

# Create and train the decision tree classifier
clf = DecisionTreeClassifier(random_state=0, min_samples_leaf=20)
clf.fit(X, y)

# Set style and plot the decision tree
plt.figure(figsize=(15, 10))
plot_tree(clf, filled=True, feature_names=list(X.columns),
          class_names=list(np.unique(y)), rounded=True, fontsize=12)
plt.savefig("./report/decision_tree.svg")
plt.show()
```

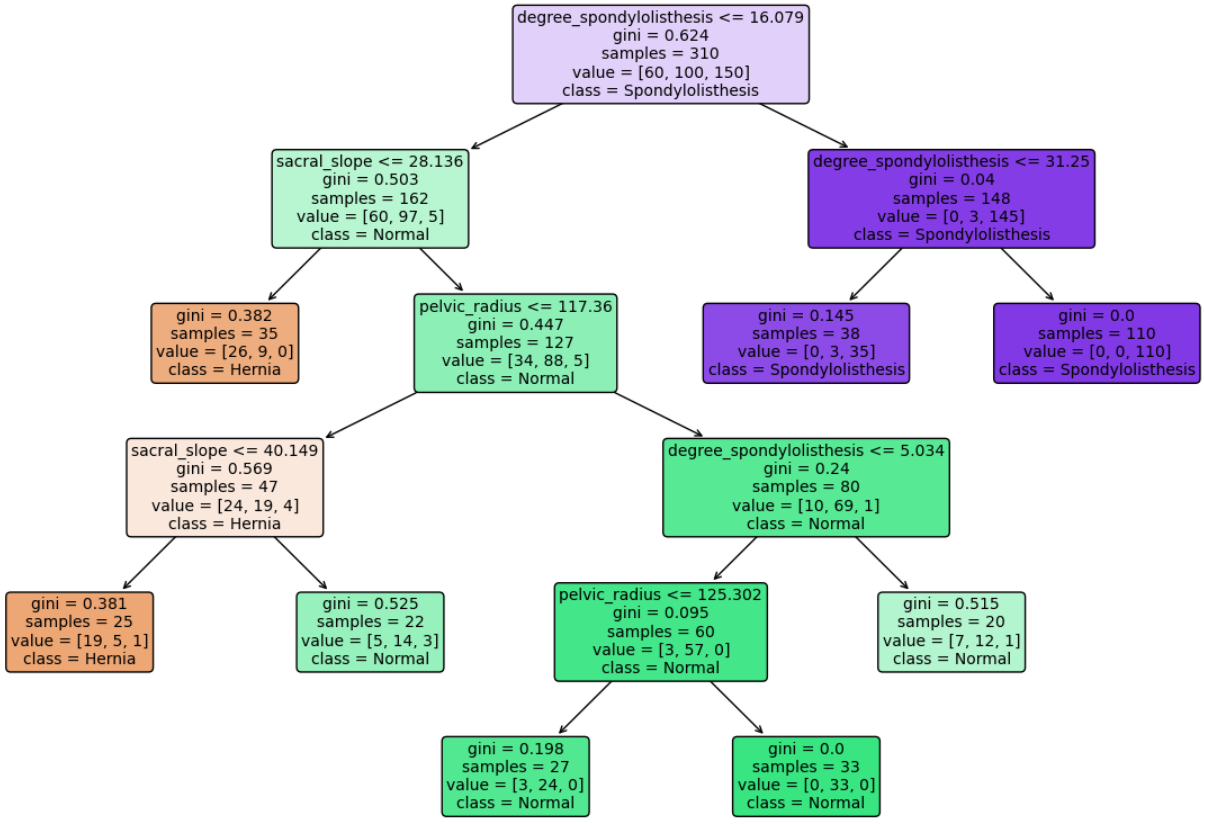

Figure 5: Decision Tree for exercise II.4a

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

The hernia condition can be characterized by:

i. Spondilolysthesis degree $\leq$ 16.079, sacral slope $\leq$ 28.136
ii. Spondilolysthesis degree $\leq$ 16.079, sacral slope $\leq$ 28.136, and pelvic radius $\leq$ 117.36

END