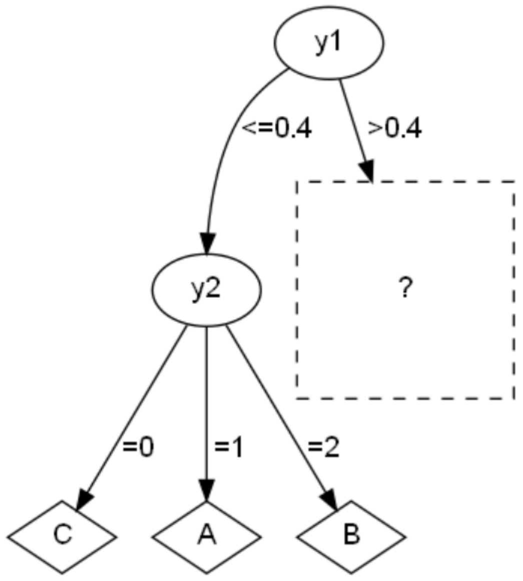
Aprendizagem 2023
Homework I – Group 090
(ist1103335, ist1103573)

## **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.

| $D$ | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_{\text{out}}$ |
|---|---|---|---|---|---|
| $\mathbf{x}_1$ | 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 |



1. [5v] Complete the given decision tree using Information gain with Shannon entropy (log2). 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 information gain of variable $y_2, y_1 > 0.4$ is given by

$$IG(y_2, y_1 > 0.4) = E(y_{\text{out}}|y_1 > 0.4) - E(y_{out}|y_2, y_1 > 0.4) \tag{1}$$

   The entropy of $y_{\text{out}}|y_1 > 0.4$ is given by:

$$E(y_{\text{out}}|y_1 > 0.4) = -\big(p(\text{A}|y_1 > 0.4)\log_2(p(\text{A}|y_1 > 0.4)) + p(\text{B}|y_1 > 0.4)\log_2(p(\text{B}|y_1 > 0.4)) + p(\text{C}|y_1 > 0.4)\log_2(p(\text{C}|y_1 > 0.4))\big)$$

   Since there are **3 As**, **2 Bs** and **2 Cs** we have the following proportions:

$$p(\text{A}|y_1 > 0.4) = \tfrac{3}{7} \quad , \quad p(\text{B}|y_1 > 0.4) = \tfrac{2}{7}, p(\text{C}|y_1 > 0.4) = \tfrac{2}{7}$$

   Therefore, we can calculate $E(y_{out}|y_1 > 0.4)$:

$$\begin{aligned} E(y_{\text{out}}|y_1 > 0.4) &= -\left(\frac{3}{7}\log_2\left(\frac{3}{7}\right) + \frac{2}{7}\log_2\left(\frac{2}{7}\right) + \frac{2}{7}\log_2\left(\frac{2}{7}\right)\right) \\ &= 1.556 \end{aligned}$$

The next step is calculating $E(y_{\text{out}}|y_2, y_1 > 0.4)$:

$$E(y_{\text{out}}|y_2, y_1 > 0.4) = E(y_{\text{out}}|y_2 = 0, y_1 > 0.4) + E(y_{\text{out}}|y_2 = 1, y_1 > 0.4) + E(y_{\text{out}}|y_2 = 2, y_1 > 0.4)$$

Then, we can calculate $E(y_{out}|y_2 = 0, y_1 > 0.4)$, $E(y_{out}|y_2 = 1, y_1 > 0.4)$ and $E(y_{out}|y_2 = 2, y_1 > 0.4)$:

$$E(y_{out}|y_2 = 0, y_1 > 0.4) = -\frac{3}{7}\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right)\right) = 0.67927$$

$$E(y_{out}|y_2 = 1, y_1 > 0.4) = -\frac{2}{7}\left(0 + \frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{1}{2}\log_2\left(\frac{1}{2}\right)\right) = \frac{2}{7}$$

$$E(y_{out}|y_2 = 2, y_1 > 0.4) = -\frac{2}{7}\left(1\log_2(1) + 0 + 0\right) = 0$$

Therefore,

$$E(y_{out}|y_2, y_1 > 0.4) = 0.67927 + \frac{2}{7} + 0 = 0.965$$

Calculation of information gain, as per (1):

$$IG(y_2, y_1 > 0.4) = 1.556 - 0.965 = \mathbf{0.591}$$

Then, we do the same for $y_3$:

$$E(y_{out}|y_3, y_1 > 0.4) = E(y_{out}|y_3 = 0, y_1 > 0.4) + E(y_{out}|y_3 = 1, y_1 > 0.4) + E(y_{out}|y_3 = 2, y_1 > 0.4)$$

$$E(y_{out}|y_3 = 0, y_1 > 0.4) = -\frac{1}{7}\left(0 + 1\log_2(1) + 0\right) = 0$$

$$E(y_{out}|y_3 = 1, y_1 > 0.4) = -\frac{2}{7}\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{1}{2}\log_2\left(\frac{1}{2}\right) + 0\right) = \frac{2}{7}$$

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

Calculation of information gain, as per (1):

$$IG(y_3, y_1 > 0.4) = 1.556 - \frac{6}{7} = \mathbf{0.699}$$

Then, we do the same for $y_4$:

$$E(y_{out}|y_4, y_1 > 0.4) = E(y_{out}|y_4 = 0, y_1 > 0.4) + E(y_{out}|y_4 = 1, y_1 > 0.4) + E(y_{out}|y_4 = 2, y_1 > 0.4)$$

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

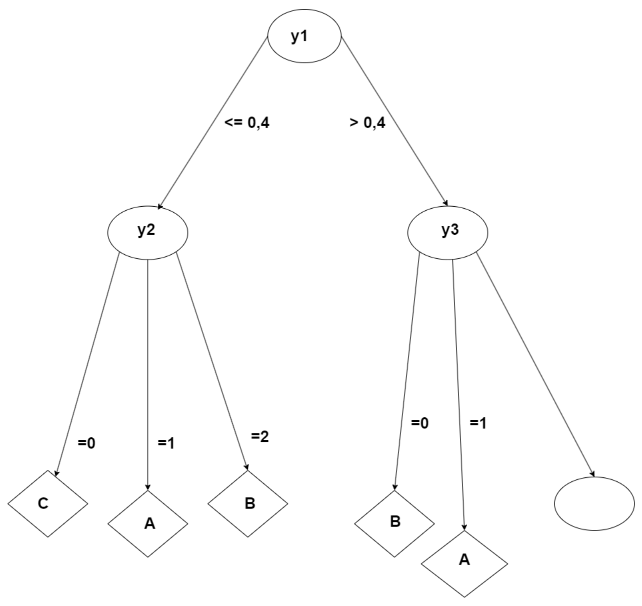
$$E(y_{out}|y_4 = 1, y_1 > 0.4) = -\frac{3}{7}\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{2}{3}\log_2\left(\frac{2}{3}\right) + 0\right) \quad = 0.39356$$

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

Calculation of information gain, as per (1):

$$IG(y_{out}|y_4, y_1 > 0.4) = 1.556 - 0.965 = \mathbf{0.591}$$

After the calculations, we know that the Decision Tree will be like this:



To continue the Tree, we need to calculate $IG(y_2|y_3 = 2, y_1 > 0.4)$ and $IG(y_4|y_3 = 2, y_1 > 0.4)$:

$$\begin{aligned} E(y_{\text{out}}|y_3 = 2, y_1 > 0.4) = &-(p(\text{A}|y_3 = 2, y_1 > 0.4)\log_2(p(\text{A}|y_3 = 2, y_1 > 0.4)). \\ &+ p(\text{B}|y_3 = 2, y_1 > 0.4)\log_2(p(\text{B}|y_3 = 2, y_1 > 0.4)) \\ &+ p(\text{C}|y_3 = 2, y_1 > 0.4)\log_2(p(\text{C}|y_3 = 2, y_1 > 0.4))) \end{aligned}$$

$$E(y_{out}|y_3 = 2, y_1 > 0.4) = -\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right) + 0 + \frac{1}{2}\log_2\left(\frac{1}{2}\right)\right) = 1$$

In the case of y2:

$$E(y_{out}|y_2 = 0, y_3 = 2, y_1 > 0.4) = -\frac{1}{4}\left(0 + 0 + 1\log_2(1)\right) = 0$$

$$E(y_{out}|y_2 = 1, y_3 = 2, y_1 > 0.4) = -\frac{1}{4}\left(0 + 0 + 1\log_2(1)\right) = 0$$

$$E(y_{out}|y_2 = 2, y_3 = 2, y_1 > 0.4) = -\frac{2}{4}\left(1\log_2(1) + 0 + 0\right) = 0$$

$$IG(y_2, y_3 = 2, y_1 > 0.4) = 1 - 0 = \mathbf{1}$$

In the case of y4:

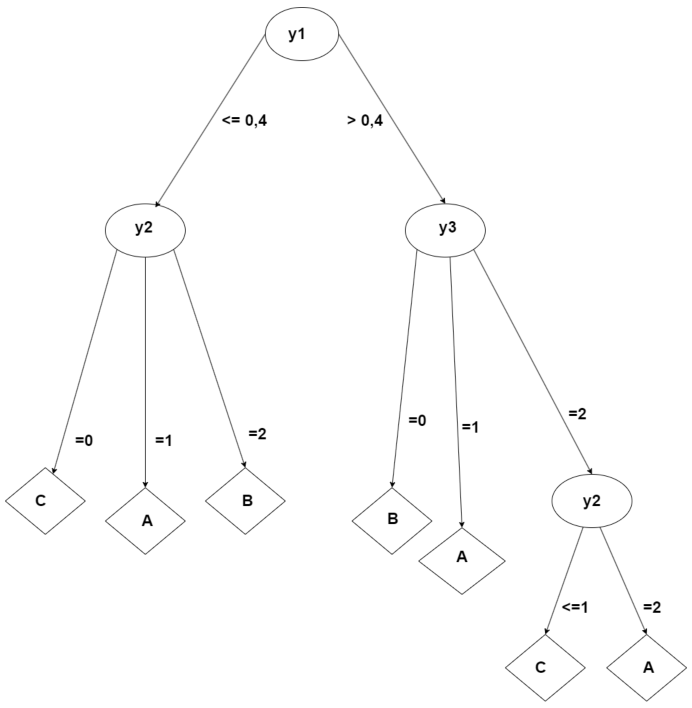
$$E(y_{out}|y_4 = 0, y_3 = 2, y_1 > 0.4) = -\frac{2}{4}\left(\log_2\left(\frac{1}{2}\right) + 0 + \frac{1}{2}\log_2\left(\frac{1}{2}\right)\right) = \frac{1}{2}$$

$$E(y_{out}|y_4 = 1, y_3 = 2, y_1 > 0.4) = -\frac{1}{4}\left(\log_2(1) + 0 + 0\right) = 0$$

$$E(y_{out}|y_4 = 2, y_3 = 2, y_1 > 0.4) = -\frac{1}{4}\left(0 + \log_2(1) + 0\right) = 0$$

$$IG(y_{out}|y_4, y_1 > 0.4) = 1 - 0.5 = \mathbf{0.5}$$

Finally, we conclude that the Decision Tree is:



2. [2.5v] Draw the training confusion matrix for the learnt decision tree.

| | True A | True B | True C | Total |
|---|---|---|---|---|
| Pred A | 4 | 1 | 0 | 5 |
| Pred B | 0 | 2 | 0 | 2 |
| Pred C | 0 | 1 | 4 | 5 |
| Total | 4 | 4 | 4 | 12 |

3. [1.5v] Identify which class has the lowest training F1 score.

The training F1 can be calculated by:

$$F_1 = \frac{1}{\frac{1}{2} \times \left(\frac{1}{P} + \frac{1}{R}\right)} \tag{2}$$

Where $P$ is precision and $R$ is recall, which can be calculated by:

$$P = \frac{\text{true positives}}{\text{true positives+false positives}} \quad R = \frac{\text{true positives}}{\text{true positives+false negatives}} \tag{3}$$

For class **A**, **4 true positives**, **0 false negatives** and **1 false positive**.
As per (2) and (3),

$$\begin{aligned} F_1 &= \frac{1}{\frac{1}{2} \times \left(\frac{4+1}{4} + \frac{4+0}{4}\right)} \\ &= 0.8889 \end{aligned}$$

For class **B**, we have **2 true positives**, **0 false negatives** and **2 false positives**.
As per (2) and (3),

$$\begin{aligned} F_1 &= \frac{1}{\frac{1}{2} \times \left(\frac{2+2}{2} + \frac{2+0}{2}\right)} \\ &= \frac{2}{3} = 0.6667 \end{aligned}$$

For class **C**, we have **4 true positives**, **0 false negatives** and **1 false positives**.
As per (2) and (3),

$$\begin{aligned} F_1 &= \frac{1}{\frac{1}{2} \times \left(\frac{4+1}{4} + \frac{4+0}{4}\right)} \\ &= 0.8889 \end{aligned}$$

In conclusion, **the class with the lowest training F1 score is B**.

4. [1v] Considering y2 to be ordinal, assess if y1 and y2 are correlated using the Spearman coefficient.

We start by collecting data for variables $y_1$ and $y_2$:

$y_1$ = [0.24,0.06,0.04,0.36,0.32,0.68,0.9,0.76,0.46,0.62,0.44,0.52]

$y_2$ = [1,2,0,0,0,2,0,2,1,0,1,0]

Then we do the Ranking of Observations:
$\text{rank}_{y_1}$ = [3,2,1,5,4,10,12,11,7,9,6,8]
$\text{rank}_{y_2}$ = [8,11,3.5,3.5,3.5,11,3.5,11,8,3.5,8,3.5]

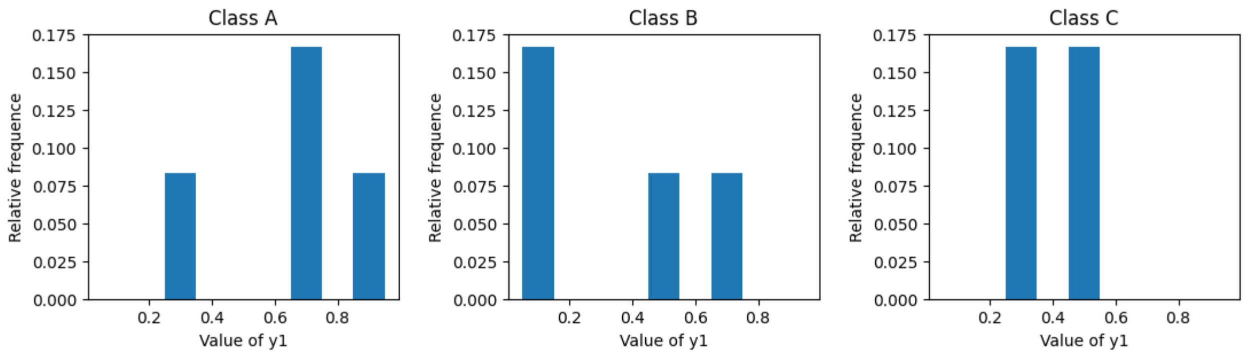
Now, we do the calculation of Spearman's Rank Correlation Coefficient by doing the PCC of the ranks:

$$PCC(rank_{y_1}, rank_{y_2}) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

$$\begin{aligned} Spearman(y_1, y_2) &= PCC([3, 2, 1, 5, 4, 10, 12, 11, 7, 9, 6, 8], [8, 11, 3.5, 3.5, 3.5, 11, 3.5, 11, 8, 3.5, 8, 3.5]) \\ &= 0.080 \end{aligned}$$

The calculated value of $Spearman(y_1, y_2)$ provides a measure of the rank correlation (order) between variables $y_1$ and $y_2$. Since the value is **0.080** we can conclude that the variables have a **very weak correlation** between them.

5. [1v] Draw the class-conditional relative histograms of y1 using 5 equally spaced bins in [0,1]. Challenge: find the root split using the discriminant rules from these empirical distributions.



An appropriate root split, given this histogram, would be to slit $y_1$ into 3 branches:

- $y_1 <= 0.2$ pointing to B;
- $0.2 < y_1 <= 0.6$ pointing to C;
- $y_1 > 0.6$ pointing to A.

## **Part II**: Programming

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.

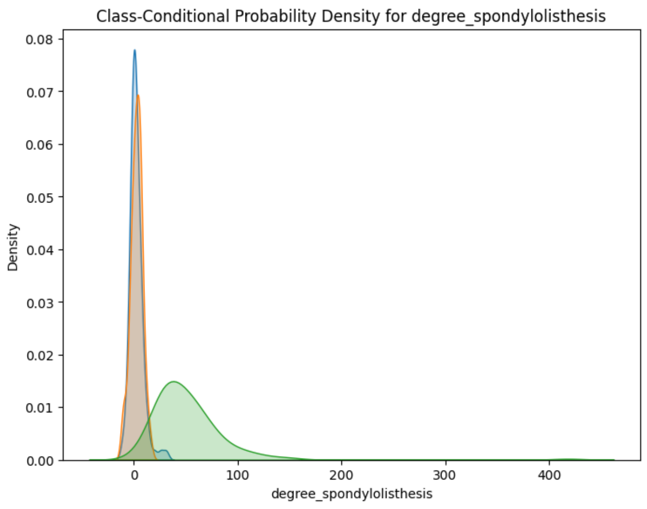
We used the following program:

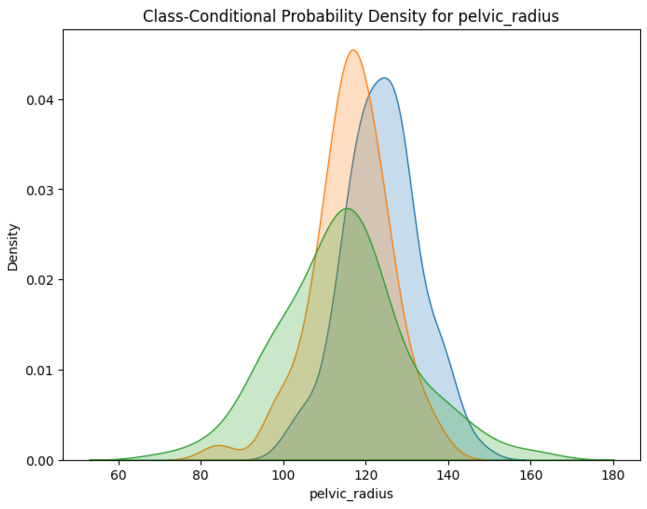
```python

from sklearn.feature_selection import f_classif
import matplotlib.pyplot as plt
import seaborn as sns

X = df.drop('class', axis=1)
y = df['class']

fimportance = f_classif(X, y)
highest_score = fimportance[0].argmax()
highest_feature = X.columns.values[highest_score]
lowest_score = fimportance[0].argmin()
lowest_feature = X.columns.values[lowest_score]

print('The variable with the highest F-score is', highest_feature, \
      'with a score of', fimportance[0][highest_score])
print('The variable with the lowest F-score is', lowest_feature,\
       'with a score of', fimportance[0][lowest_score])

```

```

top_2_features = [highest_feature, lowest_feature]

plot_df = df[top_2_features + ['class']]

print(plot_df)

class_hernia_data = plot_df[plot_df['class'] == 'Hernia']
print(class_hernia_data)
class_normal_data = plot_df[plot_df['class'] == 'Normal']
print(class_normal_data)
class_Spondylolisthesis_data = plot_df[plot_df['class'] == 'Spondylolisthesis']

for feature in top_2_features:
    plt.figure(figsize=(8, 6))
    sns.kdeplot(class_normal_data[feature], label='Normal', fill=True)
    sns.kdeplot(class_hernia_data[feature], label='Hernia', fill=True)
    sns.kdeplot(class_Spondylolisthesis_data[feature], label='Spondylolisthesis',
   fill=True)
    plt.xlabel(feature)
    plt.ylabel('Density')
    plt.title(f'Class-Conditional Probability Density for {feature}')
    plt.show()
```

Listing 1: Ploting the class-conditional probability

After applying the f_classif function we assess that the variable with the **highest discriminative power is degree_spondylolisthesis with a score of 119.12288060759764 an the variable with the lowest is pelvic_radius with a score of 16.86693475538006**.

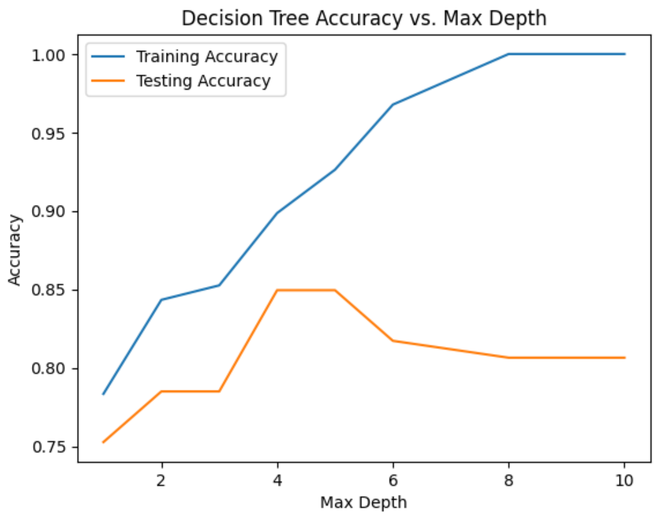




2. Using a stratified 70-30 training-testing split with a fixed seed (random_state=0), assess in a single plot both the training and testing accuracies of a decision tree with depth limits in 1,2,3,4,5,6,8,10 and the remaining parameters as default. [optional] Note that split thresholding of numeric variables in decision trees is non-deterministic in sklearn, hence you may opt to average the results using 10 runs per parameterization.

```python


from scipy.io import arff
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder

data = arff.loadarff('column_diagnosis.arff')
df = pd.DataFrame(data[0])


le = LabelEncoder()
df['class'] = le.fit_transform(df['class'])
X = df.iloc[:, :-1].values
y = df['class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
    random_state=0, stratify=y)

depth_limits = [1, 2, 3, 4, 5, 6, 8, 10]
train_accuracies = []
test_accuracies = []

for depth_limit in depth_limits:
    train_acc = 0
    test_acc = 0
    num_runs = 10

    for _ in range(num_runs):
        clf = DecisionTreeClassifier(max_depth=depth_limit, random_state=0)
        clf.fit(X_train, y_train)

        train_acc += clf.score(X_train, y_train)
        test_acc += clf.score(X_test, y_test)

    train_accuracies.append(train_acc / num_runs)
    test_accuracies.append(test_acc / num_runs)


plt.plot(depth_limits, train_accuracies, label='Training Accuracy')
plt.plot(depth_limits, test_accuracies, label='Testing Accuracy')
plt.xlabel('Max Depth')
plt.ylabel('Accuracy')
plt.title('Decision Tree Accuracy vs. Max Depth')
plt.legend()
plt.show()
```

Listing 2: Ploting the class-conditional probability



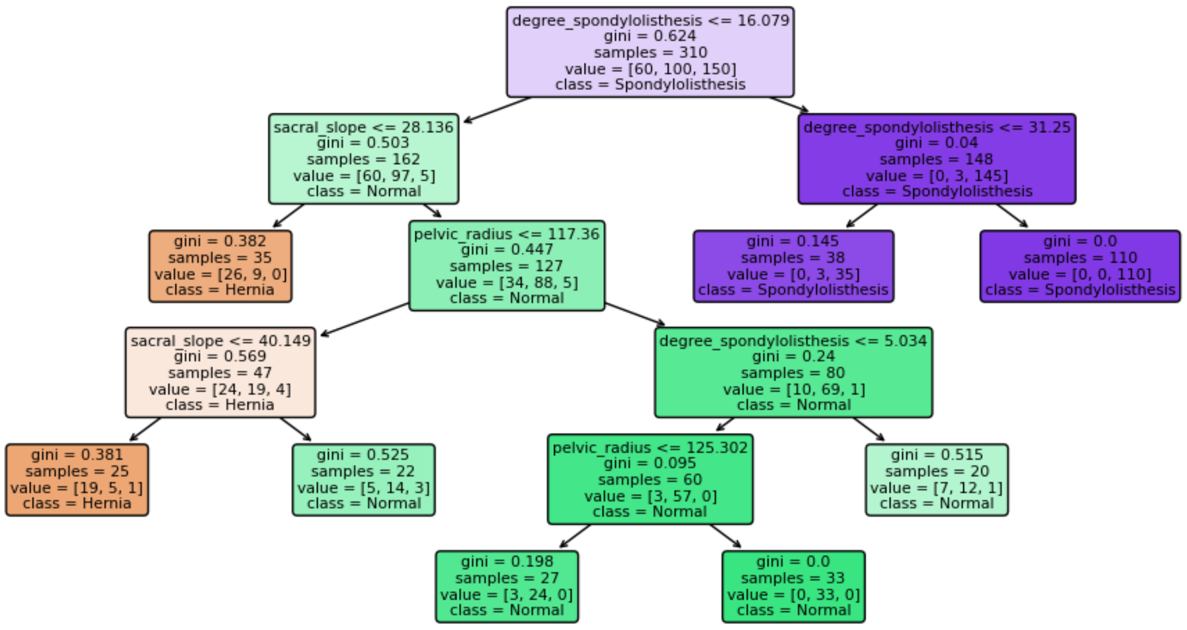
3. Comment on the results, including the generalization capacity across settings.

   Overall, the model seems to perform very well on the training data but poorly on testing data. In the beginning the model seems to perform well since the curves are both similar but when the Depth is near 5 we notice that the Training Accuracy curve increases as the Testing Accuracy decreases showing a significant gap between the training and test performance. This indicates a clear case of overfitting. Therefore we can conclude that the the model has learned to perform well on the training data but struggles to generalize to unseen or new data in different settings so it has a poor generalization and can be considered an unreliable model for a real-world scenario. To improve the model we could review the training data and check if that is any noisy data that does not adequately represent the problem for example. Another possible improvement could be to try to obtain more training data.

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. i. Plot the decision tree.

```
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.model_selection import train_test_split

# Load and partition data from column_diagnosis.arff
data = arff.loadarff('column_diagnosis.arff')
df = pd.DataFrame(data[0])
X = df.iloc[:, :-1].values

# Convert byte strings to regular strings
y_str = df['class'].astype(str)
# Initialize label encoder
le = LabelEncoder()
# Fit label encoder and transform labels
y = le.fit_transform(y_str)

# Learn classifier (Decision Tree) with max depth 3
predictor = tree.DecisionTreeClassifier(min_samples_leaf=20, random_state=0)
predictor.fit(X, y)

# Plot classifier
```

```
plt.figure(figsize=(12, 6))
tree.plot_tree(predictor, feature_names=df.columns[:-1], class_names=le.classes_,
    filled=True, rounded=True)
plt.show()
```



ii. Characterize a hernia condition by identifying the hernia-conditional associations

The decision tree shows that having a $degree\_spondylolisthesis$ <= 16.079 and a $sacral\_slope$ <= 28.136 or having a $degree\_spondylolisthesis$ <= 16.079, 28.136 < $sacral\_slope$ <= 40.149 and a $pelvic\_radius$ <= 117.36 probably means you have a Hernia.

**End note**: do not forget to also submit your Jupyter notebook