HW 1

20 de setembro de 2023 20:34

① $H(y_{out}|y_1>0.4) = -\frac{3}{7}\log_2\frac{3}{7} - 2\left(\frac{2}{7}\log_2\frac{2}{7}\right) \approx 1.557$

Para y_2: $X_1 = \{x_7, x_{10}, x_{12}\}$; $X_2 = \{x_9, x_{11}\}$; $X_3 = \{x_6, x_8\}$

$H(y_{out}|y_2 \wedge y_1>0.4) = \frac{3}{7}I(X_1) + \frac{2}{7}I(X_2) + \frac{2}{7}I(X_3) = \frac{3}{7}\times\left(3\left(-\frac{1}{3}\log_2\frac{1}{3}\right)\right) + \frac{2}{7}\times\left(2\left(-\frac{1}{2}\log\frac{1}{2}\right)\right) + \frac{2}{7}\times 0$

$= \frac{3}{7}\times 1.585 + \frac{2}{7}\times 1 + 0 = 0.965$

$IG(y_2|y_1>0.4) = H(y_{out}|y_1>0.4) - H(y_{out}|y_2 \wedge y_1>0.4) = 1.557 - 0.965 = 0.592$

Para y_3: $X_1 = \{x_{10}\}$; $X_2 = \{x_7, x_9\}$; $X_3 = \{x_6, x_8, x_{11}, x_{12}\}$

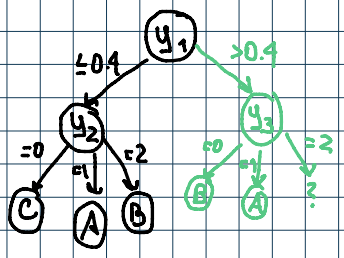
$H(y_{out}|y_3 \wedge y_1>0.4) = \frac{1}{7}I(X_1) + \frac{2}{7}I(X_2) + \frac{4}{7}I(X_3) = \frac{1}{7}\times 0 + \frac{2}{7}\times 1 + \frac{4}{7}\times 1 = 0.857$

$IG(y_3|y_1>0.4) = H(y_{out}|y_1>0.4) - H(y_{out}|y_3 \wedge y_1>0.4) = 1.557 - 0.857 = \boxed{0.7}$ Maior IG

Para y_4: $X_1 = \{x_8, x_{12}\}$; $X_2 = \{x_6, x_9, x_{10}\}$; $X_3 = \{x_7, x_{11}\}$

$H(y_{out}|y_4 \wedge y_1>0.4) = \frac{2}{7}I(X_1) + \frac{3}{7}I(X_2) + \frac{2}{7}I(X_3) = \frac{2}{7}\times 1 + \frac{3}{7}\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right) + \frac{2}{7}\times 1 = 0.965$

$IG(y_4|y_1>0.4) = H(y_{out}|y_1>0.4) - H(y_{out}|y_4 \wedge y_1>0.4) = 1.557 - 0.965 = 0.592$



Escolhe-se y_3 como raíz da sub-árvore, uma vez que tem o maior IG.

Existem menos que 4 observações em $y_3=0$ e $y_3=1$, logo, quando $y_3=0$ escolhe-se B, e quando $y_3=1$ escolhe-se A (empate → ordem alfabética).

Para $y_3=2$:

$H(y_{out}|y_1>0.4 \wedge y_3=2) = 2\left(-\frac{1}{2}\log_2\frac{1}{2}\right) = 1 = H(y_{out})'$ (dois)

Para y_2: $X_1 = \{x_{12}\}$ $X_2 = \{x_{11}\}$ $X_3 = \{x_6, x_8\}$

$H(y_2|y_1>0.4 \wedge y_3=2) = \frac{1}{4}\times 0 + \frac{1}{4}\times 0 + \frac{1}{2}\times 0 = 0 = H(y_2)'$

$IG(y_2|y_1>0.4 \wedge y_3=2) = H(y_{out})' - H(y_2)' = 1-0 = \boxed{1}$ maior IG

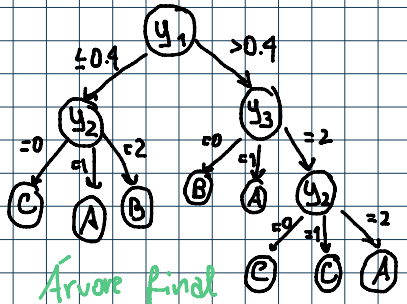
Para y_4: $X_1 = \{x_8, x_{12}\}$ $X_2 = \{x_6\}$ $X_3 = \{x_7\}$

$H(y_4|y_1>0.4 \wedge y_3=2) = \frac{1}{2}\times 1 + \frac{1}{4}\times 0 + \frac{1}{4}\times 0 = 0.5 = H(y_4)'$

$IG(y_4|y_1>0.4 \wedge y_3=2) = H(y_{out})' - H(y_4)' = 1-0.5 = 0.5$

- Escolhe-se y_2 como raíz da subárvore por ter o maior IG.

A árvore final encontra-se à direita →



Árvore final

• Escolhe-se y_2 como raiz da subárvore por ter o maior IG.

A árvore final encontra-se à direita →

Árvore final ⓒ ⓒ Ⓐ

2.

D	y_1	y_2	y_3	y_4	y_{out}	
x_1	0.24	1	1	0	A	✓
x_2	0.06	2	0	0	B	✓
x_3	0.04	0	0	0	~~B~~ C	✓
x_4	0.36	0	2	1	C	✓
x_5	0.32	0	0	2	C	✓
x_6	0.68	2	2	1	A	✓
x_7	0.9	0	1	2	A	✓
x_8	0.76	2	2	0	A	✓
x_9	0.46	1	1	1	~~B~~ A	✓
x_{10}	0.62	0	0	1	B	✓
x_{11}	0.44	1	2	2	C	✓
x_{12}	0.52	0	2	0	C	✓

Certos = 1, 2, 4, 5, 6, 7, 8, 10, 11, 12

errados = 3, 9

A matriz de confusão é a seguinte:

		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. $F_1(A) = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = 2 \times \frac{\frac{4}{5} \times 1}{\frac{4}{5} + 1} = 2 \times \frac{4}{9} = 0.889$

$F_1(B) = 2 \times \frac{1 \times \frac{1}{2}}{1 + \frac{1}{2}} = 2 \times \frac{1}{3} = 0.667$

$F_1(C) = 2 \times \frac{\frac{4}{5} \times 1}{\frac{4}{5} + 1} = \frac{8}{9} \approx 0.889$

Logo, a classe com menor F_1 score é a classe B.

4. y_1 ordenado = [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: 1 2 3 4 5 6 7 8 5 10 11 12 → y_1'

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

ranks: 3.5 3.5 3.5 3.5 3.5 3.5 8 8 8 11 11 11 → y_2'

y_1	y_2
0.24 3	1 8
0.06 2	2 11
0.04 1	0 3.5
0.36 5	0 3.5
0.32 4	0 3.5
0.68 10	2 11
0.9 12	0 3.5
0.76 11	2 11
0.46 7	1 8
0.62 9	0 3.5
0.44 6	1 8
0.52 8	0 3.5

$\Sigma y_1' y_2' = 517.5$

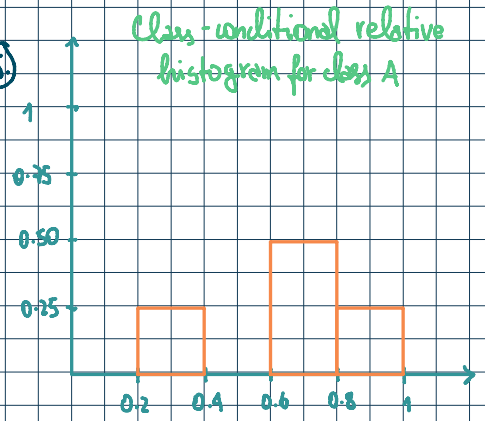
$\overline{y_1'} = 6.5$ $\overline{y_2'} = 6.5$

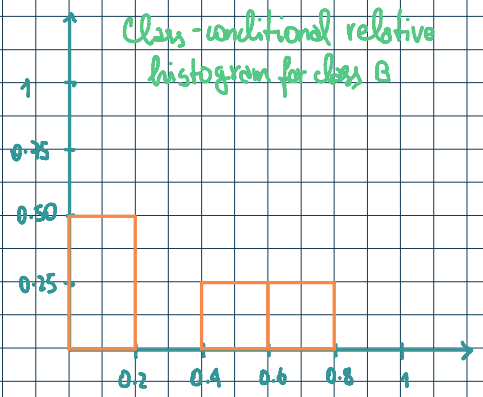
$cov(y_1', y_2') = \frac{\Sigma y_{1i}' y_{2i}' - n \overline{y_1'} \overline{y_2'}}{11} = \frac{517.5 - 12 \times 6.5^2}{11} = 0.955$

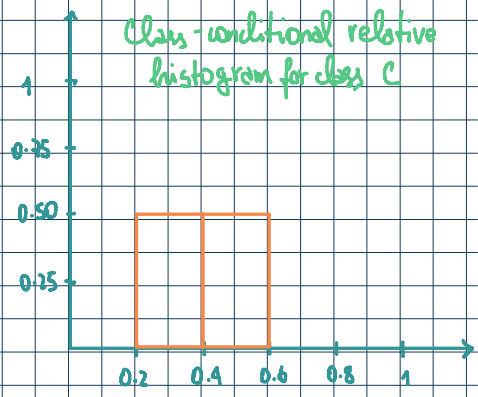
$\sigma_{y_1'} = \sqrt{\frac{\Sigma y_{1i}'^2 - n \overline{y_1'}^2}{n-1}} = \sqrt{\frac{650 - 12 \times 6.5^2}{11}} = 3.61$ $\sigma_{y_2'} = \sqrt{\frac{628.5 - 12 \times 6.5^2}{11}} = 3.32$

$Corr_{Spearman} = \frac{cov(y_1', y_2')}{\sigma_{y_1'} \sigma_{y_2'}} = \frac{0.955}{3.61 \times 3.32} = 0.0797$ → Valor próximo de 0, sugere que y_1 e y_2 estão fracamente relacionados.

(5.)


Class - conditional relative histogram for class A
1
0.75
0.50
0.25
0.2
0.4
0.6
0.8
1

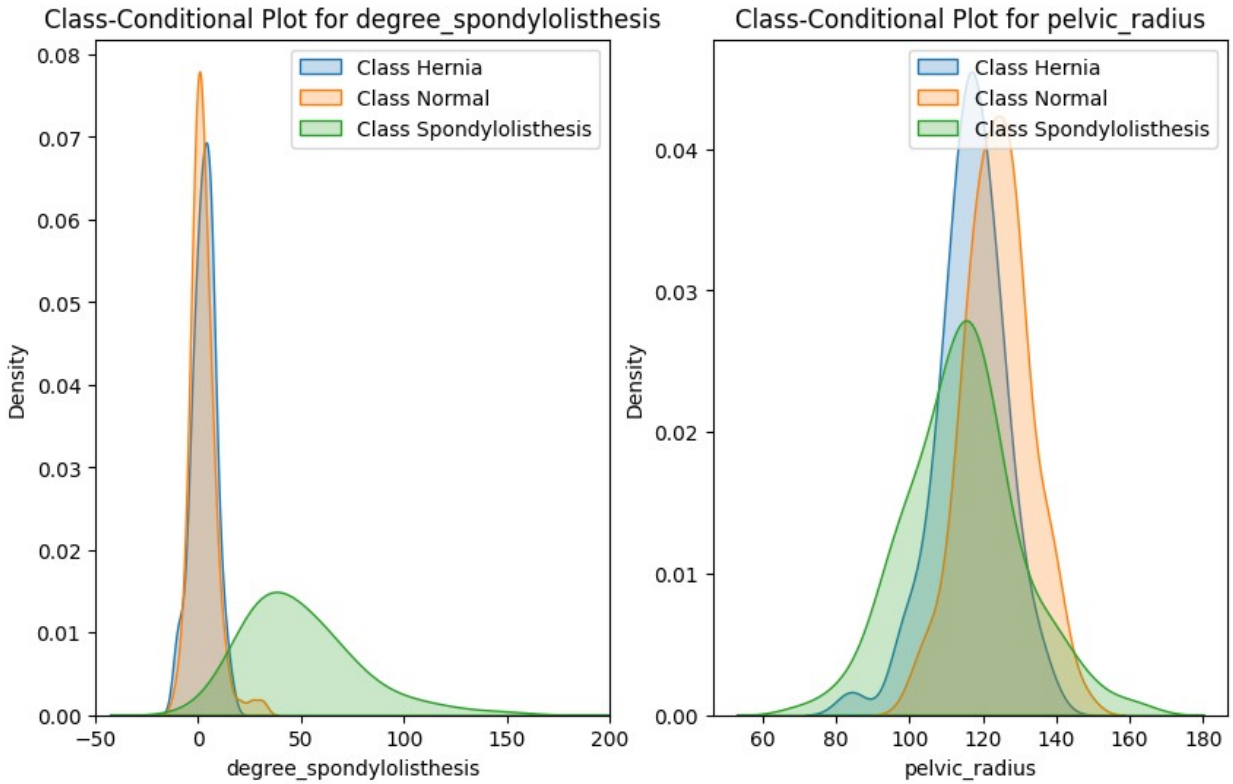

Class - conditional relative histogram for class B
1
0.75
0.50
0.25
0.2
0.4
0.6
0.8
1


Class - conditional relative histogram for class C
1
0.75
0.50
0.25
0.2
0.4
0.6
0.8
1

Exercise 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

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

# Imports data, converts and selects highest and lowest discriminative
power input variables
data = loadarff('column_diagnosis.arff')
df = pd.DataFrame(data[0])
df['class'] = df['class'].str.decode('utf-8')
X = df.drop('class', axis=1)
y = df['class']
fimportance = f_classif(X, y)
max_score = np.argmax(fimportance[0])
min_score = np.argmin(fimportance[0])
max_score_label = X.columns.values[max_score]
min_score_label = X.columns.values[min_score]

# Prob density function plot for highest discriminative power variable
plt.figure(figsize=(10, 6))
plt.subplot(1, 2, 1)
for class_ in np.unique(y):
    subset = X[y == class_][max_score_label]
    sns.kdeplot(subset, label=f'Class {class_}', fill=True)
plt.title(f'Class-Conditional Plot for {max_score_label}')
plt.xlabel(max_score_label)
plt.ylabel('Density')
plt.xlim((-50, 200))
plt.legend()
# Prob density function plot for lowest discriminative power variable
plt.subplot(1, 2, 2)
for class_ in np.unique(y):
    subset = X[y == class_][min_score_label]
    sns.kdeplot(subset, label=f'Class {class_}', fill=True)
plt.title(f'Class-Conditional Plot for {min_score_label}')
plt.xlabel(min_score_label)
plt.ylabel('Density')
plt.legend()
plt.show()
```

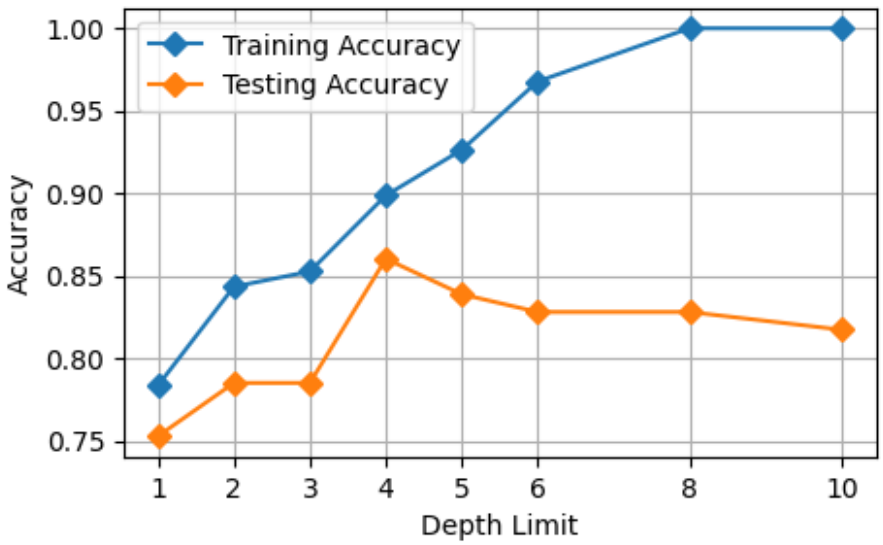


Exercise 2

- 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.

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

# Loads ARFF data
from scipy.io import arff
data, meta = arff.loadarff('column_diagnosis.arff')
df = pd.DataFrame(data)
df['class'] = df['class'].str.decode('utf-8')

# Extract features and labels
X = df.drop('class', axis=1)
y = df['class']

# Splits data
X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.7, random_state=0, stratify=y)
```

```
# Relevant arrays
train_accuracies, test_accuracies = [], []
depth_limits = [1, 2, 3, 4, 5, 6, 8, 10]

# Calculates accuracies for each depth limit
for depth_limit in depth_limits:
    predictor = DecisionTreeClassifier(max_depth=depth_limit)
    predictor.fit(X_train, y_train)
    train_accuracies.append(predictor.score(X_train, y_train))
    test_accuracies.append(predictor.score(X_test, y_test))

# Creates a plot for both accuracies
plt.figure(figsize=(5, 3))
plt.xlabel('Depth Limit')
plt.ylabel('Accuracy')
plt.xticks(depth_limits)
plt.plot(depth_limits, train_accuracies, marker='D', label='Training
Accuracy')
plt.plot(depth_limits, test_accuracies, marker='D', label='Testing
Accuracy')
plt.legend()
plt.grid(True)
plt.show()
```

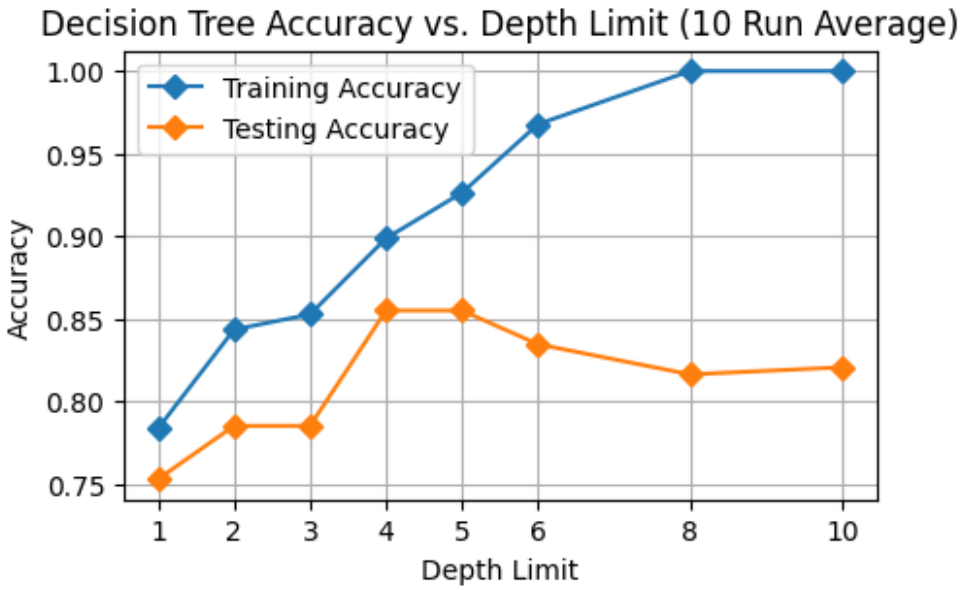


2 (Optional)

- 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.

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

```
from scipy.io import arff

# Same thing as above basically
data, meta = arff.loadarff('column_diagnosis.arff')
df = pd.DataFrame(data)
df['class'] = df['class'].str.decode('utf-8')
X = df.drop('class', axis=1)
y = df['class']

X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.7, random_state=0, stratify=y)

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

# Calculates num_runs accuracies for each depth limit
for depth_limit in depth_limits:
    train_accuracy_sum = test_accuracy_sum = 0
    for _ in range(num_runs):
        predictor = DecisionTreeClassifier(max_depth=depth_limit)
        predictor.fit(X_train, y_train)

        train_accuracy_sum += predictor.score(X_train, y_train)
        test_accuracy_sum += predictor.score(X_test, y_test)

    # Appends results averaged
    train_accuracies.append(train_accuracy_sum / num_runs)
    test_accuracies.append(test_accuracy_sum / num_runs)

# Creates a plot similar to above with avg results
plt.figure(figsize=(5, 3))
plt.title('Decision Tree Accuracy vs. Depth Limit (10 Run Average)')
plt.xlabel('Depth Limit')
plt.ylabel('Accuracy')
plt.xticks(depth_limits)
plt.plot(depth_limits, train_accuracies, marker='D', label='Training
Accuracy')
plt.plot(depth_limits, test_accuracies, marker='D', label='Testing
Accuracy')
plt.legend()
plt.grid(True)
plt.show()
```



Exercise 3

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

Com base no gráfico produzido no exercício anterior, podemos observar que a training accuracy cresce constantamente à medida que o depth limit também cresce. Até ao depth limit 4, observa-se igualmente uma tendência crescente consistente da testing accuracy e proximidade entre os dados de treino e os dados de teste. Entre o depth limit 4 e 5, a testing accuracy mantém-se constante enquanto a training accuracy cresce e a partir do depth limit 6 a diferença entre ambas torna-se cada vez maior. Esta observação aliada ao facto da training accuracy continuar a crescer cada vez mais até atingir 100% de accuracy (possivelmente captando ruído, ou seja, informação irrelevante) indica-nos um claro caso de overfitting do modelo, em que este se ajusta muito bem aos dados de treino mas se torna mais ineficaz a prever dados novos de teste. Uma das estratégias possíveis para evitar overfitting é limitar a profundidade da árvore, e a melhor profundidade para o fazer seria quando o depth limit é igual a 4, devido à proximidade entre a training e testing accuracy e ao facto da testing accuracy ser relativamente elevada (cerca de 0.85).

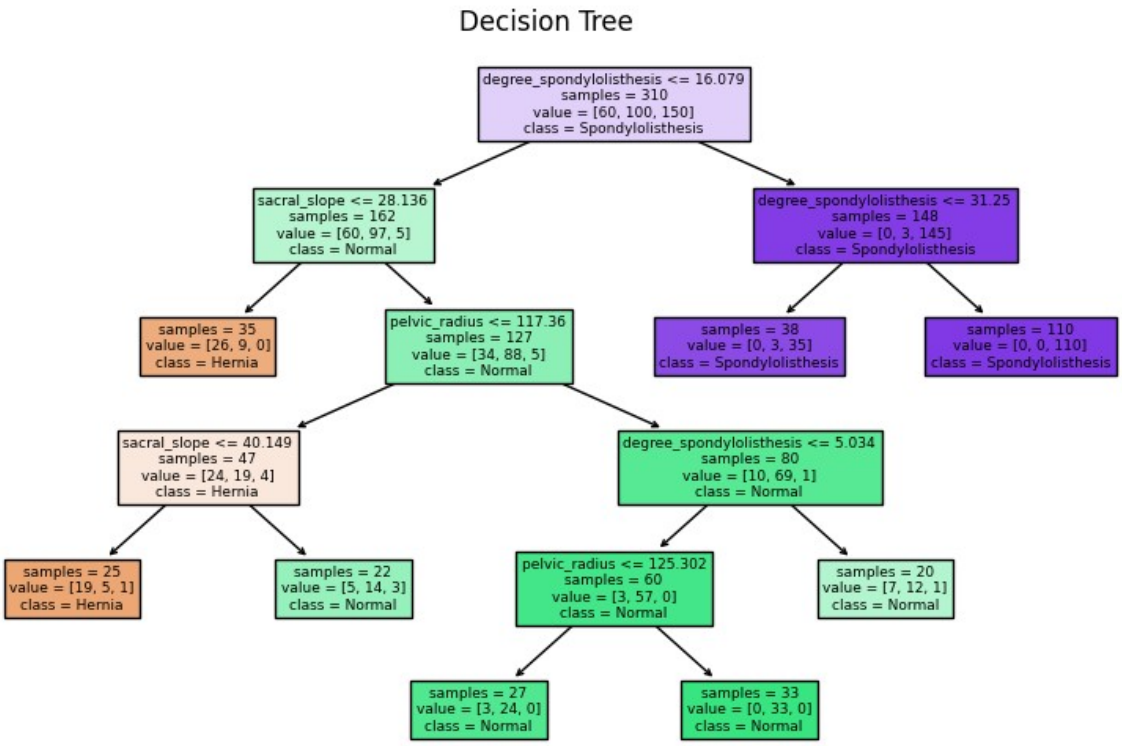
Exercise 4

- 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.

Exercise 4.1

- Plot the decision tree.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier, plot_tree

# 1. load and partition data
from scipy.io import arff
data, meta = arff.loadarff('column_diagnosis.arff')
df = pd.DataFrame(data)
df['class'] = df['class'].str.decode('utf-8')
X = df.drop('class', axis=1)
y = df['class']

# Class names need to be sorted for it to work on the "plot_tree"
df_class_names = df['class'].unique()
df_class_names.sort()

# 2. learn classifier
predictor = DecisionTreeClassifier(random_state=0,
min_samples_leaf=20)
predictor.fit(X, y)

# 3. plot classifier
plt.figure(figsize=(10,6))
plot_tree(predictor, filled=True, feature_names=df.columns[:-1],
class_names=df_class_names, impurity=False)
plt.title("Decision Tree")
plt.show()
```



Exercise 4.2

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

Com base na árvore de decisão, pode-se caraterizar uma hérnia da seguinte forma: o paciente tem um valor de degree_spondylolisthesis igual ou inferior a 16.079 obrigatoriamente. Daqui em diante, se tiver um valor de sacral_slope igual ou menor a 28.136 é classificado como tendo uma hérnia (1º leaf node possível); se tiver um valor de sacral_slope maior que 28.136 e um pelvic_radius menor ou igual a 117.36, encontra-se numa situação na qual já é mais provável que tenha uma hérnia. Ainda, se a este ponto tiver um sacral_slope menor ou igual a 40.149, é classificado como tendo uma hérnia (2º e último leaf node possível).