Aprendizagem

Instituto Superior Técnico

setembro de 2023

Homework 1 - Report

Joana Pimenta (103730), Rodrigo Laia (102674)

Pen and Paper

1. Para decidir qual a variável seguinte a por na árvore de decisão para $y_1 > 0.4$, temos que calcular os information gains das variáveis y_2, y_3 e y_4 condicionadas a $y1 > 0.4$, através da seguinte fórmula:

$$IG(y_n) = H(y_{out}) - H(y_{out}|y_n) \tag{1}$$

$H(y_{out}) = -\frac{3}{7} \cdot \log_2(\frac{3}{7}) - \frac{2}{7} \cdot \log_2(\frac{2}{7}) - \frac{2}{7} \cdot \log_2(\frac{2}{7}) = 1.56$

$H(y_{out}|y_2) = \frac{2}{7} \cdot (-1 \cdot \log_2(1)) + \frac{2}{7} \cdot (-\frac{1}{2} \cdot \log_2(\frac{1}{2}) - \frac{1}{2} \cdot \log_2(\frac{1}{2})) + \frac{3}{7} \cdot (-\frac{1}{3} \cdot \log_2(\frac{1}{3}) - \frac{1}{3} \cdot \log_2(\frac{1}{3}) - \frac{1}{3} \cdot \log_2(\frac{1}{3})) = 0.96$

$H(y_{out}|y_3) = \frac{4}{7} \cdot (-\frac{1}{2} \cdot \log_2(\frac{1}{2}) - \frac{1}{2} \cdot \log_2(\frac{1}{2})) + \frac{2}{7} \cdot (-\frac{1}{2} \cdot \log_2(\frac{1}{2}) - \frac{1}{2} \cdot \log_2(\frac{1}{2})) + \frac{1}{7} \cdot (-1 \cdot \log_2(1)) = 0.86$

$H(y_{out}|y_4) = \frac{2}{7} \cdot (-\frac{1}{2} \cdot \log_2(\frac{1}{2}) - \frac{1}{2} \cdot \log_2(\frac{1}{2})) + \frac{2}{7} \cdot (-\frac{1}{2} \cdot \log_2(\frac{1}{2}) - \frac{1}{2} \cdot \log_2(\frac{1}{2})) + \frac{3}{7} \cdot (-\frac{1}{3} \cdot \log_2(\frac{1}{3}) - \frac{2}{3} \cdot \log_2(\frac{2}{3})) = 0.96$

$IG(y_2) = 0.6$

$IG(y_3) = 0.7$

$IG(y_4) = 0.6$

y_3 é a variável com o maior information gain para $y_1 > 0.4$ logo é a variável seguinte na árvore de decisão

No caso $y_3 = 0$ a variável de output toma o valor B. No caso $y_3 = 1$ a variável de output pode tomar o valor A ou B, no entanto, como apenas temos duas observações, utilizamos o critério de desempate de ordema alfabética crescente escolhendo-se por isso a classe A. No caso $y_3 = 0$ há 4 observações distintas pelo que vamos separar o nó. Para $y_3 = 2$ qual é a variável com maior information gain?

$H(y_{out}) = -\frac{1}{2} \cdot \log_2(\frac{1}{2}) - \frac{1}{2} \cdot \log_2(\frac{1}{2}) = 1$

$H(y_{out}|y_2) = \frac{1}{4} \cdot (-1 \cdot \log_2(1)) + \frac{1}{4} \cdot (-1 \cdot \log_2(1)) + \frac{1}{4} \cdot (-1 \cdot \log_2(1)) = 0$

$H(y_{out}|y_4) = \frac{1}{4} \cdot (-1 \cdot \log_2(1)) + \frac{1}{4} \cdot (-1 \cdot \log_2(1)) + \frac{1}{2} \cdot (-\frac{1}{2} \cdot \log_2(\frac{1}{2}) - \frac{1}{2} \cdot \log_2(\frac{1}{2})) = \frac{1}{2}$

$IG(y_2) = 1 - 0 = 1$

$IG(y_4) = 1 - \frac{1}{2} = \frac{1}{2}$

$IG(y_2) > IG(y_4)$, logo y_2 é a variável seguinte.

A árvore obtida é então:

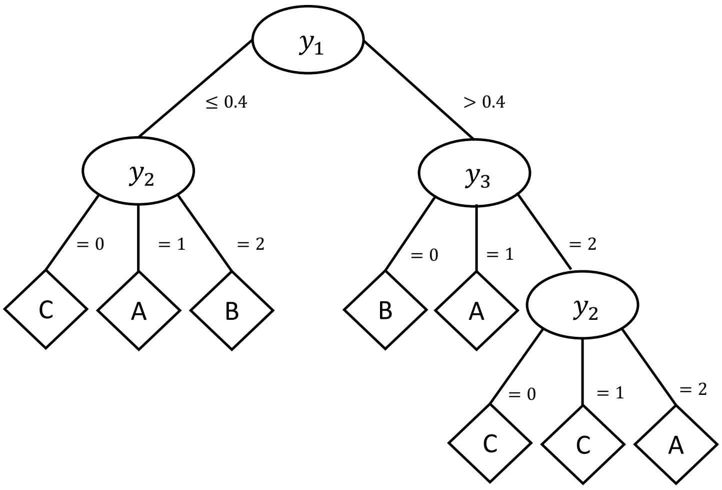


Figura 1: Árvore de decisão

2. Após a consulta da árvore de modo a obter o output previsto, foi possível obter a seguinte matriz de confusão.

D	y_1	y_2	y_3	y_4	y_{out}	$y_{previsto}$
x_1	0.24	1	1	0	A	A
x_2	0.06	2	0	0	B	B
x_3	0.04	0	0	0	B	C
x_4	0.36	0	2	1	C	C
x_5	0.32	0	0	2	C	C
x_6	0.68	2	2	1	A	A
x_7	0.90	0	1	2	A	A
x_8	0.76	2	2	0	A	A
x_9	0.46	1	1	1	B	A
x_{10}	0.62	0	0	1	B	B
x_{11}	0.44	1	2	2	C	C
x_{12}	0.52	0	2	0	C	C

Tabela 1: Output real e output previsto

		real		
		A	B	C
previsto	A	4	1	0
	B	0	2	0
	C	0	1	4

Tabela 2: Matriz de confusão

3. Fórmulas utilizadas:

$$F1_{score} = 2 \cdot \frac{P \cdot R}{P + R} \tag{2}$$

$$P = \frac{TP}{TP + FP} \tag{3}$$

$$R = \frac{TP}{TP + FN} \tag{4}$$

Em que a precisão é representada por P e o recall ou sensibilidade por R. Assim,

Classe	P	R	$F1_{score}$
A	$\frac{4}{4+1} = 0.8$	$R = \frac{4}{4} = 1$	0.89
B	$\frac{2}{2+0} = 1$	$R = \frac{2}{4} = \frac{1}{2}$	0.67
C	$\frac{4}{4+1} = 0.8$	$R = \frac{4}{4} = 1$	0.89

Tabela 3: $F1_{score}$ para as classes A, B e C

A classe com menor $F1_{score}$ é a classe B.

4. Primeiro é preciso ordenar as variáveis y_1 e y_2 obtendo y_1' e y_2', respetivamente.

y_1	y_1'	y_2	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

Tabela 4: Variáveis ordenadas

O coeficiente de Spearman é igual ao coeficiente de Pearson das variáveis ordenadas y_1' e y_2', sendo, por isso, calculado através da seguinte fórmula

$$\rho_{y_1',y_2'} = \frac{\sum_{i=1}^{n}(y_{1_i}' - \bar{y}_1')(y_{2_i}' - \bar{y}_2')}{\sqrt{\sum_{i=1}^{n}(y_{1_i}' - \bar{y}_1')^2} \cdot \sqrt{\sum_{i=1}^{n}(y_{2_i}' - \bar{y}_2')^2}} \tag{5}$$

Cálculos:

$$\bar{y}_1' = \frac{1+2+3+4+5+6+7+8+9+10+11+12}{12} = 6.5$$

$$\bar{y}_2' = \frac{3.5 \times 6 + 8 \times 3 + 11 \times 3}{12} = 6.5$$

Assim,

$$\text{Coeficiente de Spearman} = \frac{41.125 - 6.5 \times 6.5}{\sqrt{54.17 - 6.5^2} \cdot \sqrt{52.375 - 6.5^2}} = 0.080$$

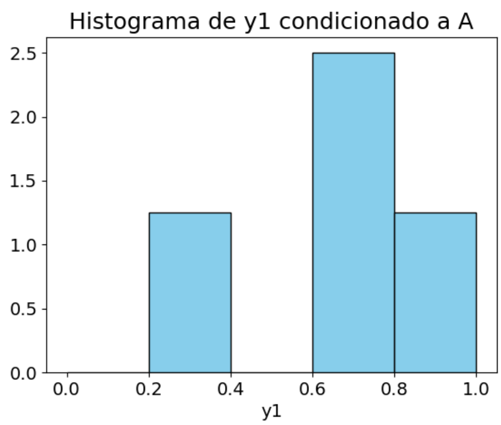
Como este coeficiente é muito próximo de zero, concluímos que as variáveis y_1 e y_2 não estão correladas.

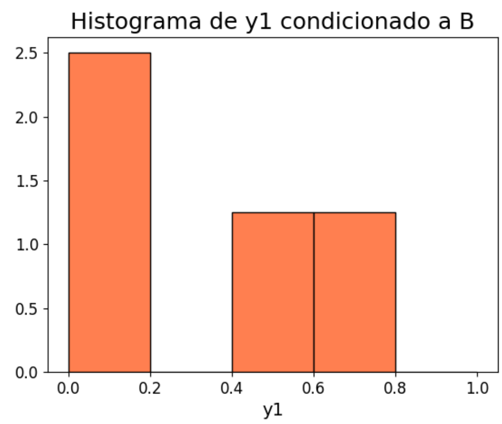
5. Largura dos bins: $l = \frac{1}{5} = 0.2$

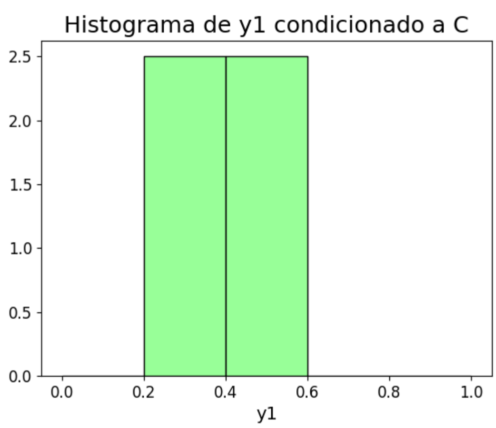
De forma a que a área total do histograma seja 1, para calcular a altura de cada bin usamos a fórmula: $h = \frac{C}{N \times l}$ sendo C o número de contagens para o bin em questão, N o número total de contagens do histograma e l a largura de cada bin.

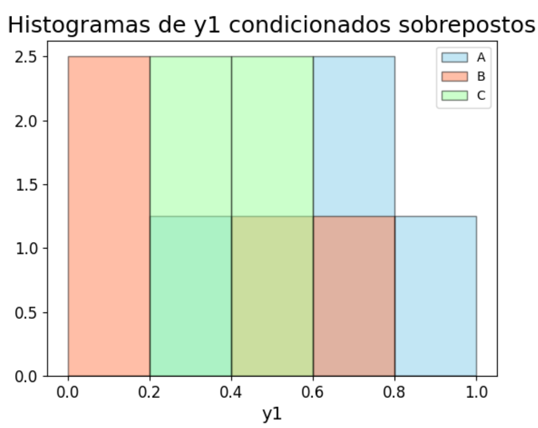
classe	h_{bin1}	h_{bin2}	h_{bin3}	h_{bin4}	h_{bin5}
A	0	1.25	0	2.5	1.25
B	2.5	0	1.25	1.25	0
C	0	2.5	2.5	0	0

Tabela 5: Altura dos bins do histograma para cada classe









Analisando o histograma, concluimos que as regras discriminativas são:

- $0 \leq x_1 \leq 0.2 \implies$ classe B
- $0.2 < x_1 \leq 0.6 \implies$ classe C
- $0.6 < x_1 \leq 1 \implies$ classe A

Assim, o melhor root split a aplicar à variável y_1 seria separar o nó em três sendo o primeiro associado ao intervalo [0;0.2], o segundo ao intervalo (0.2;0.6] e o terceiro ao intervalo (0.6;1.0], uma vez que dentro destes intervalos a variável y_1 está associada a uma determinada classe com muito maior probabilidade do que às outras.

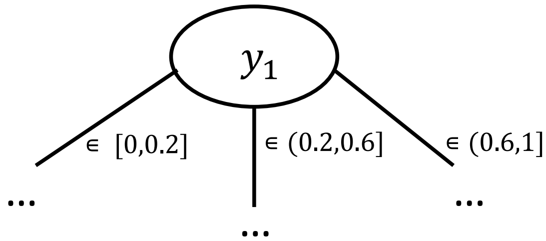


Figura 3: root split sugerido

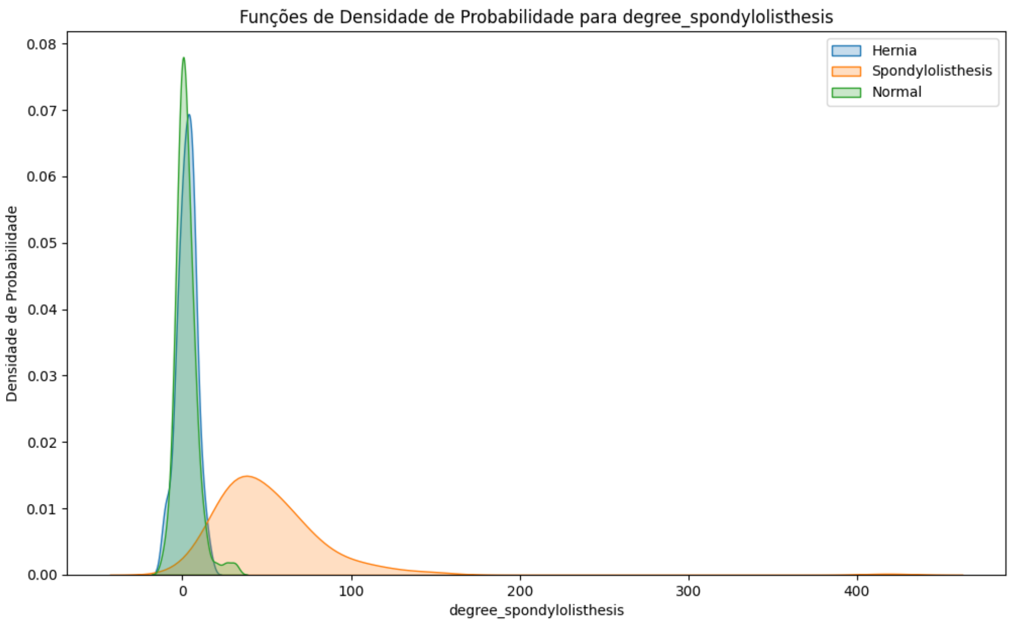
Programming - Código Python e Resultados Obtidos

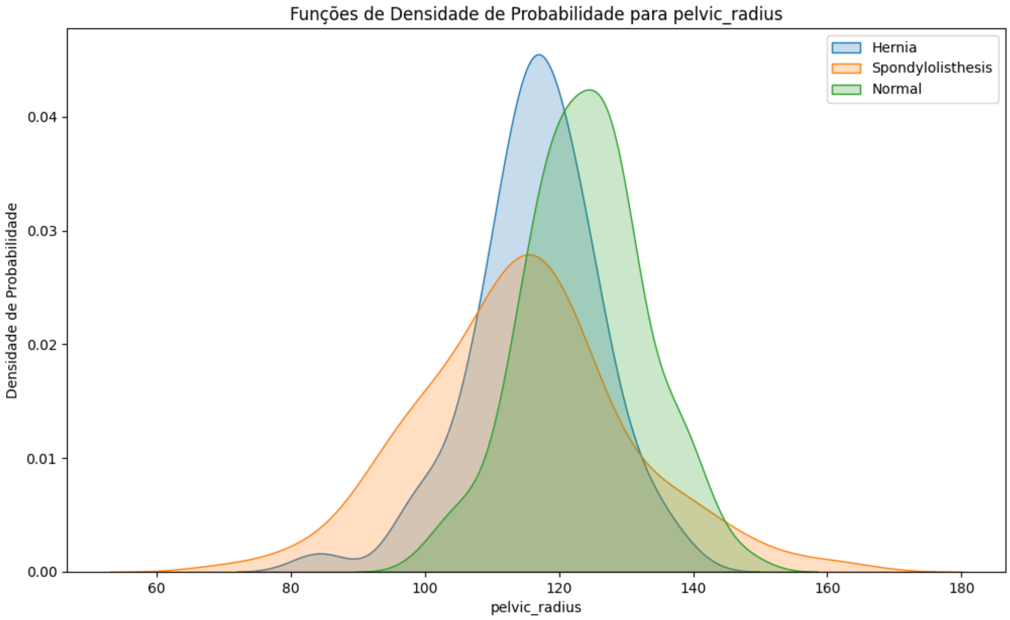
1. Cálculamos o poder discriminativo das variáveis utilizando a função *f_classif* do pacote *sklearn* (quanto maior o f-value maior o poder discriminativo e quanto menor o p-value maior o poder discriminativo). Obtivemos os seguintes valores:

variável	f-value	p-value
pelvic_incidence	98.54	8.75e-34
lumbar_lordosis_angle	21.30	2.18e-09
sacral_slope	89.64	2.18e-31
pelvic_radius	16.87	1.12e-07

Tabela 6: Poder discriminativo das várias variáveis

Assim, a variável com maior poder discriminativo é *degree_spondylolisthesis* e a com menor poder discriminativo é *pelvic_radius*



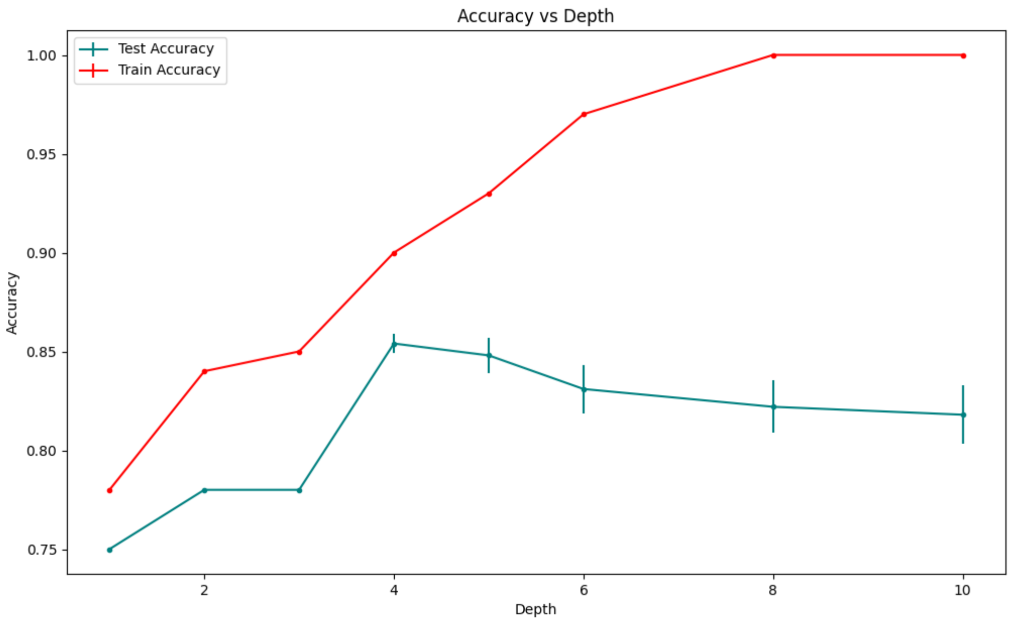


Observando os gráficos percebemos que de facto no gráfico da densidade de probabilidade da variável *degree_ spondylolisthesis* há uma classe que se destaca bastante das restantes, ao contrário do que acontece no gráfico da densidade de probabilidade de *pelvic_ radius*.

```
### Exercise 1 ###

variables = df.drop('class', axis= 1)
target = df['class']

fvalues, pvalues = f_classif(variables, target)

fvalue_df = pd.DataFrame({'variable': variables.columns, 'fvalues': fvalues, 'pvalues': pvalues})
print(fvalue_df.head())

'''the greater the f-value the better
the lower the p-value the better

degree_spondylolisthesis is the variable with higher discriminative power.
pelvic_radius is the variable with lower discriminative power.'''

hernia = df[df['class'] == 'Hernia']
spondylolisthesis = df[df['class'] == 'Spondylolisthesis']
normal = df[df['class'] == 'Normal']

#Graphic1
plt.figure(figsize=(12,7))
sns.kdeplot(hernia['degree_spondylolisthesis'], label= 'Hernia', fill = True)
```

```
    sns.kdeplot(spondylolisthesis['degree_spondylolisthesis'], label= 'Spondylolisthesis', fill = True)
    sns.kdeplot(normal['degree_spondylolisthesis'], label= 'Normal', fill = True)
    plt.xlabel('degree_spondylolisthesis')
    plt.ylabel('Densidade de Probabilidade')
    plt.legend()
    plt.title('Fun  es de Densidade de Probabilidade para degree_spondylolisthesis')
    plt.show()

    #Graphic2
    plt.figure(figsize=(12,7))
    sns.kdeplot(hernia['pelvic_radius'], label= 'Hernia', fill = True)
    sns.kdeplot(spondylolisthesis['pelvic_radius'], label= 'Spondylolisthesis', fill = True)
    sns.kdeplot(normal['pelvic_radius'], label= 'Normal', fill = True)
    plt.xlabel('pelvic_radius')
    plt.ylabel('Densidade de Probabilidade')
    plt.legend()
    plt.title('Fun  es de Densidade de Probabilidade para pelvic_radius')
    plt.show()


```

2. Dividindo as observações do dataset entre 70% para treino e 30% para teste com uma semente fixa, representamos num gráfico a accuracy de treino e de teste para diferentes profundidades da árvore de decisão (1,2,3,4,5,6,8,10). A accuracy representada no gráfico foi calculada através da média de 10 accuracys calculadas para cada profundidade da árvore, de modo a minimizar o erro proveniente da inicialização random da árvore e estimar melhor a performance do modelo. Calculamos também o desvio padrão da accuracy.




```python
### Exercise 2 ###

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

variables_train, variables_test, target_train, target_test= train_test_split(variables, target,
                  train_size=0.7, stratify=target, random_state=0)

final_acc1, final_acc2 = np.array([]), np.array([])
std1, std2 = np.array([]), np.array([])

for depth in depth_limits:
    acc_folder1, acc_folder2 = np.array([]), np.array([])
    for i in range(10):
        tree2 = DecisionTreeClassifier(criterion='gini', max_depth=depth)
        tree2.fit(variables_train, target_train)

        y_pred1 = tree2.predict(variables_test)
        y_pred2 = tree2.predict(variables_train)

        acc_folder2 = np.append(acc_folder2, round(metrics.accuracy_score(target_train, y_pred2),2))
        acc_folder1 = np.append(acc_folder1, round(metrics.accuracy_score(target_test, y_pred1),2))

    final_acc1 = np.append(final_acc1, np.mean(np.array(acc_folder1)))
    final_acc2 = np.append(final_acc2, np.mean(np.array(acc_folder2)))
    std1 =np.append(std1, np.std(acc_folder1))
    std2 =np.append(std2, np.std(acc_folder2))

```

```
    print("accuracy test list:", final_acc1)
    print("accuracy train list:", final_acc2)


    #Graphics
    plt.figure(figsize=(12,7))
    plt.errorbar(depth_limits, final_acc1, yerr=std1, label='Test Accuracy', color='#008080')
    plt.errorbar(depth_limits, final_acc2, yerr=std2, label='Train Accuracy', color='red')
    plt.plot(depth_limits, final_acc1, 'o', color='#008080', markersize=3)
    plt.plot(depth_limits, final_acc2, 'o', color='red', markersize=3)
    plt.title("Accuracy vs Depth")
    plt.ylabel("Accuracy")
    plt.xlabel("Depth")
    plt.legend(loc='upper left', bbox_to_anchor=(0.0, 1.0)) #move the legend to the left
    plt.show()

```

3. Através do gráfico concluímos que a accuracy de treino cresce sempre com o aumento da profundidade enquanto que a accuracy de teste cresce até atingir um máximo a partir do qual decresce, tal como esperado devido ao overfitting - o que indica que o modelo se está a ajustar em excesso aos dados de treino, não generalizando bem. Assim a melhor profundidade a usar seria por volta do máximo da accuracy de teste ou seja aproximadamente quatro. Para além disso, observamos que o desvio padrão aumenta significativamente nos pontos onde há overfitting como seria de esperar devido à maior variação da accuracy para modelos que estão ajustados a um modelo demasiado específico.

4. i) Usando todos os dados como dados de treino, foi feita uma árvore de decisão, sendo que para evitar riscos de overfitting cada folha tem no mínimo 20 indivíduos. A árvore obtida encontra-se na figura seguinte.

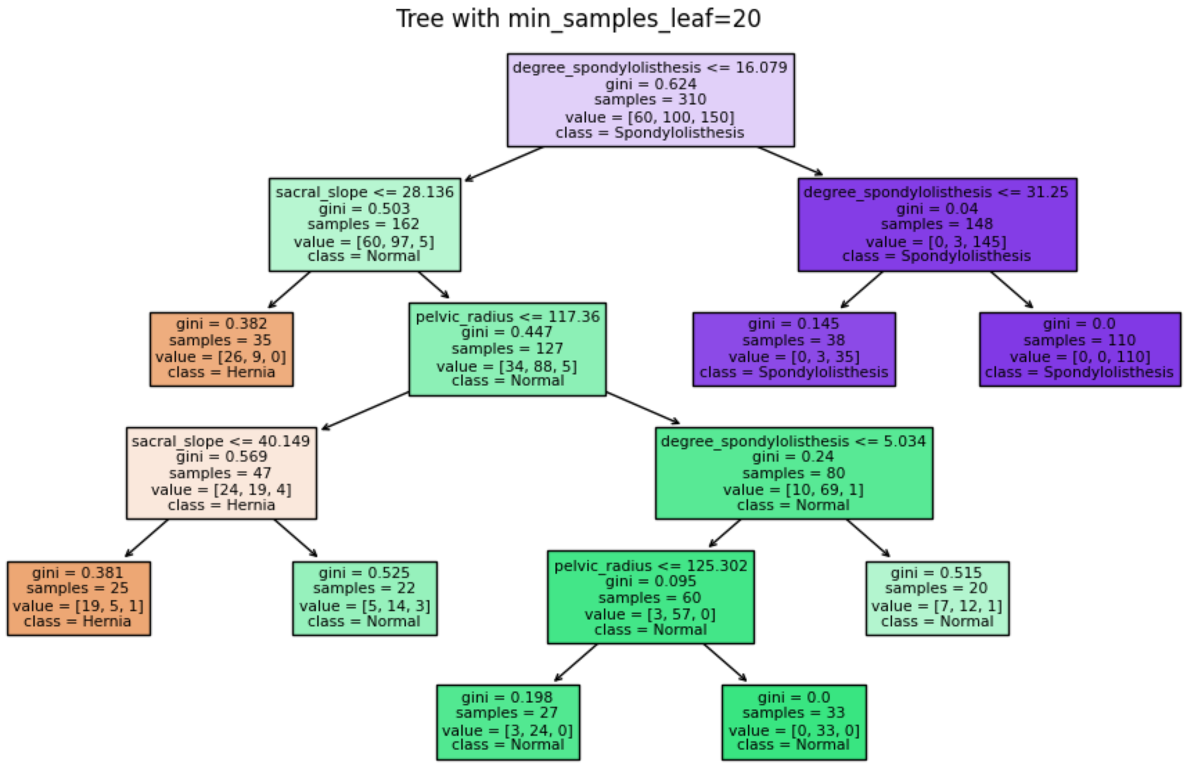


Figura 4: Árvore de decisão - mínimo de 20 indivíduos por folha

```
### Exercise 4 a) ###

tree4 = DecisionTreeClassifier(criterion='gini', min_samples_leaf=20, random_state=0)
tree4.fit(variables, target)
target_pred = tree4.predict(variables)

print('accuracy:', round(metrics.accuracy_score(target, target_pred), 2))

plt.figure(figsize=(12,7))
tree.plot_tree(tree4, filled=True, feature_names=variables.columns, class_names=tree4.classes_)
plt.title("Tree with min_samples_leaf=20")
plt.savefig('tree4.png')
plt.show()
```

ii) Existem dois caminhos que a árvore pode seguir para classificar uma instância como Hernia. Ambos começam para a esquerda na raiz da árvore (quando *degree_spondylolisthesis* é menor ou igual a 16.079). Em seguida, se *scacral_slope* é menor ou igual a 28.136 a classe escolhida é hérnia senão apenas é esta a classe escolhida quando *pelvic_radius* é menor ou igual a 117.36 e *sacral_slope* é menor que 40.149.