Aprendizagem 2022/23

Homework I – Group 107

(ist1103811, ist1103479)

I. Pen-and-paper

1) Answer 1

1) $E(y_{out} | y_1 > 0.4) = -\frac{3}{7}\log(\frac{3}{7}) - \frac{2}{7}\log(\frac{2}{7}) - \frac{2}{7}\log(\frac{2}{7}) \simeq 1.5567$

$E(y_{out} | y_1 > 0.4, y_2) = \frac{3}{7}(-\frac{1}{3}\log(\frac{1}{3}) - \frac{1}{3}\log(\frac{1}{3}) - \frac{1}{3}\log(\frac{1}{3})) + \frac{2}{7}(-0\log 0 - \frac{1}{2}\log(\frac{1}{2}) - \frac{1}{2}\log(\frac{1}{2})) +$
$+ \frac{2}{7}(-1\log 1 - 0\log 0 - 0\log 0) = 0.67927 + \frac{2}{7} \simeq 0.96498$

$E(y_{out} | y_1 > 0.4, y_3) = \frac{1}{7}\times 0 + \frac{2}{7}(-\frac{1}{2}\log(\frac{1}{2}) - \frac{1}{2}\log(\frac{1}{2}) - 0) + \frac{4}{7}(-\frac{1}{2}\log(\frac{1}{2}) - 0 - \frac{1}{2}\log(\frac{1}{2})) =$
$= \frac{2}{7} + \frac{4}{7} = \frac{6}{7} \simeq 0.8571$

$E(y_{out} | y_1 > 0.4, y_4) = \frac{2}{7}(-\frac{1}{2}\log(\frac{1}{2}) - 0 - \frac{1}{2}\log(\frac{1}{2})) + \frac{3}{7}(-\frac{1}{3}\log(\frac{1}{3}) - \frac{2}{3}\log(\frac{2}{3})) + \frac{2}{7}(-\frac{1}{2}\log(\frac{1}{2}) - \frac{1}{2}\log(\frac{1}{2})) =$
$= \frac{2}{7} + 0.3936 + \frac{2}{7} \simeq 0.96498$

$IG(y_{out} | y_1 > 0.4, y_2) = E(y_{out} | y_1 > 0.4) - E(y_{out} | y_1 > 0.4, y_2) = 1.5567 - 0.96498 = 0.59167$

$IG(y_{out} | y_1 > 0.4, y_3) = 1.5567 - 0.8571 = 0.69956$ → escolhemos y_3

$IG(y_{out} | y_1 > 0.4, y_4) = IG(y_{out} | y_1 > 0.4, y_2) = 0.59167$

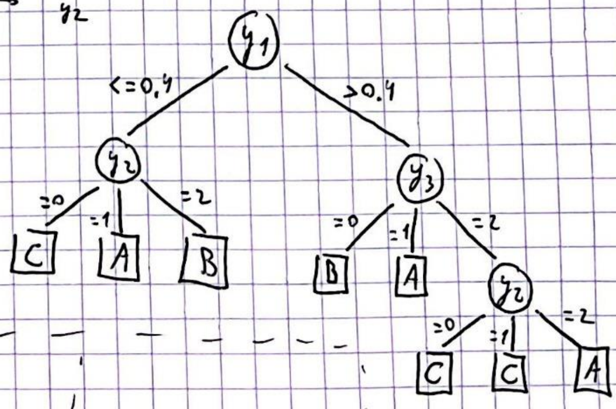
$E(y_{out} | y_1 > 0.4, y_3 = 2) = -\frac{1}{2}\log(\frac{1}{2}) - 0\log 0 - \frac{1}{2}\log(\frac{1}{2}) = 1$

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

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

$IG(y_{out} | y_1 > 0.4, y_3 = 2, y_2) = 1 - 0 = 1$ → escolhemos y_2

$IG(y_{out} | y_1 > 0, y_3 = 2, y_4) = 1 - \frac{1}{2} = \frac{1}{2}$



2) Answer 2

2) Prediction | Real

	Prediction $\hat{z}$	Real z
x_1	A	A
x_2	B	B
x_3	C	B
x_4	C	C
x_5	C	C
x_6	A	A
x_7	A	A
x_8	A	A
x_9	A	B
x_{10}	B	B
x_{11}	C	C
x_{12}	C	C

true

pred	A	B	C	
A	4	1	0	5
B	0	2	0	2
C	0	1	4	5
	4	4	4	

[redacted]

3) Answer 3

[redacted]

3)

$$\text{precision}_A = \frac{\#TP_A}{\#TP_A + \#FP_A} = \frac{4}{5}$$

$$\text{recall}_A = \frac{\#TP_A}{\#TP_A + \#FN_A} = \frac{4}{4} = 1$$

$$F_{1A} = \frac{2 \times \text{precision}_A \times \text{recall}_A}{\text{precision}_A + \text{recall}_A} = \frac{8}{9} \simeq 0.889$$

$$\text{precision}_B = \frac{2}{2} = 1$$

$$\text{recall}_B = \frac{2}{4} = \frac{1}{2}$$

$$F_{1B} = \frac{2}{3} \simeq 0.667$$

$$\left.\begin{array}{l} \text{precision}_C = \frac{4}{5} \\ \text{recall}_C = \frac{4}{4} = 1 \end{array}\right\} F_{1C} = \frac{\frac{8}{5}}{\frac{9}{5}} = \frac{8}{9} = 0.889$$

Lowest F1 score = B

4) Answer 4

4)

$y_1 \to$	r_1	$y_2 \to$	r_2
0.24	10	1	5
0.06	11	2	2
0.04	12	0	9.5
0.36	8	0	9.5
0.32	9	0	9.5
0.68	3	2	2
0.9	1	0	9.5
0.76	2	2	2
0.46	6	1	5
0.62	4	0	9.5
0.44	7	1	5
0.52	5	0	9.5

CA

$\sum r_1 = 78$

$\sum r_2 = 78$

$\sum r_1^2 = 650$

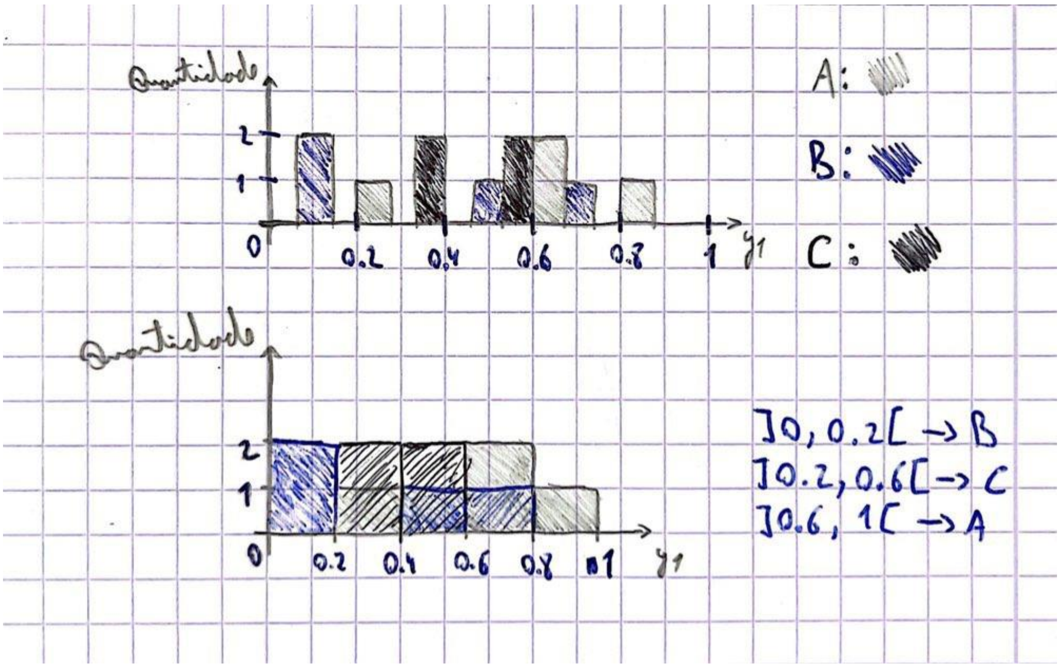
$\sum r_2^2 = 628.5$

$\sum r_1 r_2 = 517.5$

$$r = \frac{\sum r_1 r_2 - \frac{\sum r_1 \sum r_2}{12}}{\sqrt{\left(\sum r_1^2 - \frac{(\sum r_1)^2}{12}\right)\left(\sum r_2^2 - \frac{(\sum r_2)^2}{12}\right)}} = \frac{517.5 - 507}{\sqrt{(650-507)(628.5-507)}} \simeq 0.07966$$

Correlação muito fraca

5) Answer 5



II. Programming and critical analysis

1) Answer 1

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

data, meta = arff.loadarff("column_diagnosis.arff")
df = pd.DataFrame(data)

target_variable = 'class'
X = df.drop(columns=[target_variable])
y = df[target_variable]

f_scores, _ = f_classif(X, y)

feature_scores_df = pd.DataFrame({'Feature': X.columns, 'F-Score': f_scores})
feature_scores_df = feature_scores_df.sort_values(by='F-Score', ascending=False)

highest_discriminative_var = feature_scores_df.iloc[0]['Feature']
print(f"Input Variable with Highest Discriminative Power:\n{fea-
ture_scores_df.head(1)}")

lowest_discriminative_var = feature_scores_df.iloc[-1]['Feature']
print(f"\nInput Variable with Lowest Discriminative Power:\n{fea-
ture_scores_df.tail(1)}")

plt.figure(figsize=(12, 5))
sns.kdeplot(data=df, x=highest_discriminative_var, hue=target_variable, com-
mon_norm=False, shade=True)
plt.title(f'Class-Conditional PDFs for {highest_discriminative_var}')
plt.xlabel(highest_discriminative_var)
plt.ylabel('Density')
plt.legend(title=target_variable)
plt.savefig('high_discriminative_pdf.png')
plt.show()

plt.figure(figsize=(12, 5))
sns.kdeplot(data=df, x=lowest_discriminative_var, hue=target_variable, com-
mon_norm=False, shade=True)
plt.title(f'Class-Conditional PDFs for {lowest_discriminative_var}')
plt.xlabel(lowest_discriminative_var)
plt.ylabel('Density')
plt.legend(title=target_variable)
plt.savefig('low_discriminative_pdf.png')
plt.show()
```

Output:

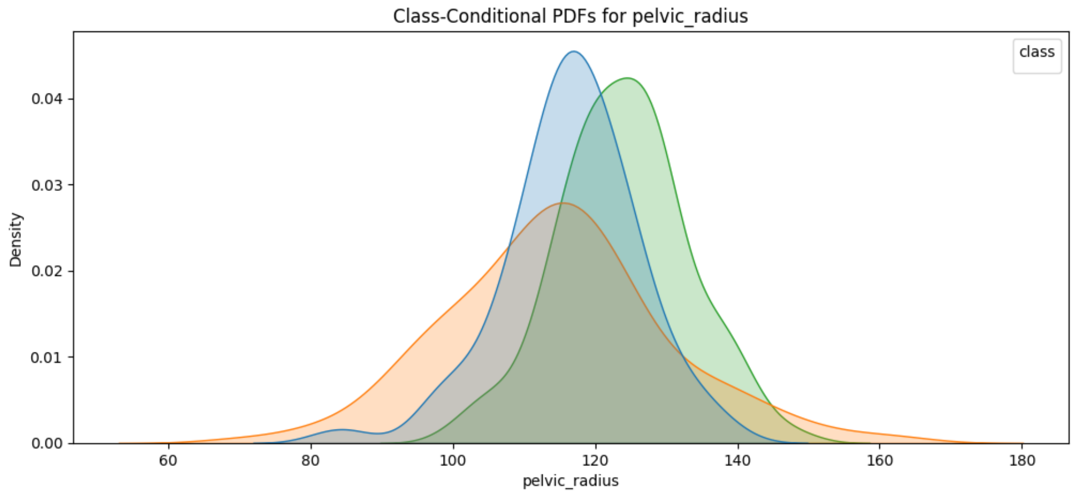
Input Variable with Highest Discriminative Power:

	Feature	F-Score
5	degree_spondylolisthesis	119.122881

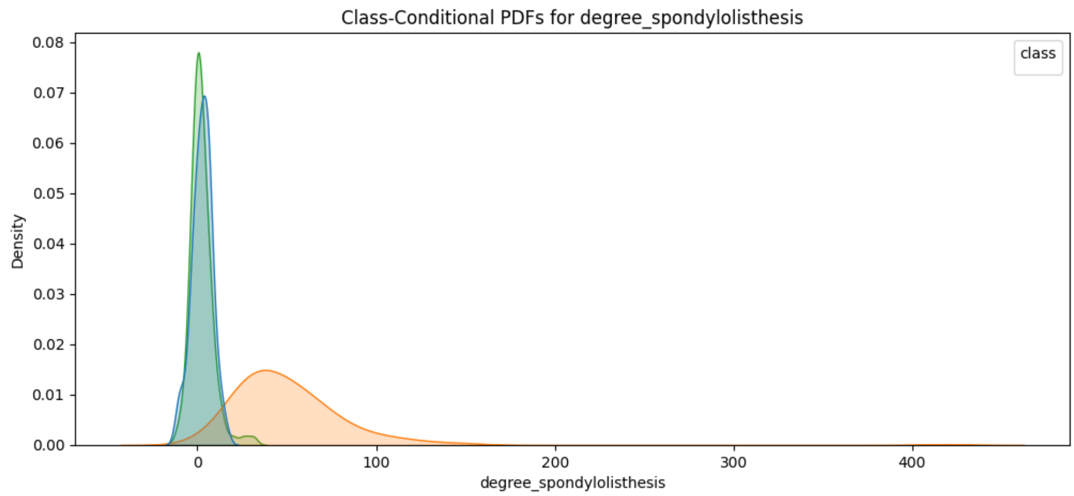
Input Variable with Lowest Discriminative Power:

	Feature	F-Score
4	pelvic_radius	16.866935

Low discriminative plot:



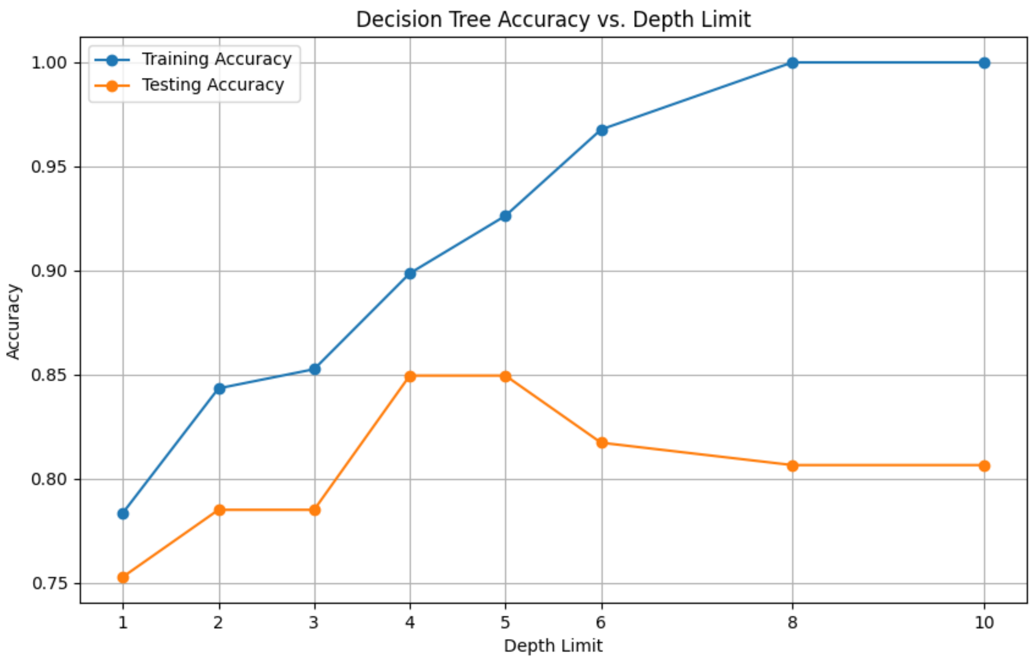
High discriminative plot:



2) Answer 2

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from scipy.io import arff
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
import pandas as pd

data, meta = arff.loadarff("column_diagnosis.arff")
df = pd.DataFrame(data)

X = df.drop(columns=['class']).values
y = df['class']

label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

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

train_accuracies = []
test_accuracies = []

seed = 0
num_runs = 10

for depth_limit in depth_limits:
    train_acc = 0
    test_acc = 0
    for _ in range(num_runs):
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
stratify=y, random_state=seed)

        clf = DecisionTreeClassifier(max_depth=depth_limit, random_state=seed)
        clf.fit(X_train, y_train)

        train_pred = clf.predict(X_train)
        test_pred = clf.predict(X_test)

        train_acc += accuracy_score(y_train, train_pred)
        test_acc += accuracy_score(y_test, test_pred)

    train_avg_acc = train_acc / num_runs
    test_avg_acc = test_acc / num_runs
```

```
    train_accuracies.append(train_avg_acc)
    test_accuracies.append(test_avg_acc)

plt.figure(figsize=(10, 6))
plt.plot(depth_limits, train_accuracies, marker='o', label='Training Accuracy')
plt.plot(depth_limits, test_accuracies, marker='o', label='Testing Accuracy')
plt.title('Decision Tree Accuracy vs. Depth Limit')
plt.xlabel('Depth Limit')
plt.ylabel('Accuracy')
plt.xticks(depth_limits)
plt.legend()
plt.grid(True)
plt.savefig("accuracy_vs_depth.png")
```

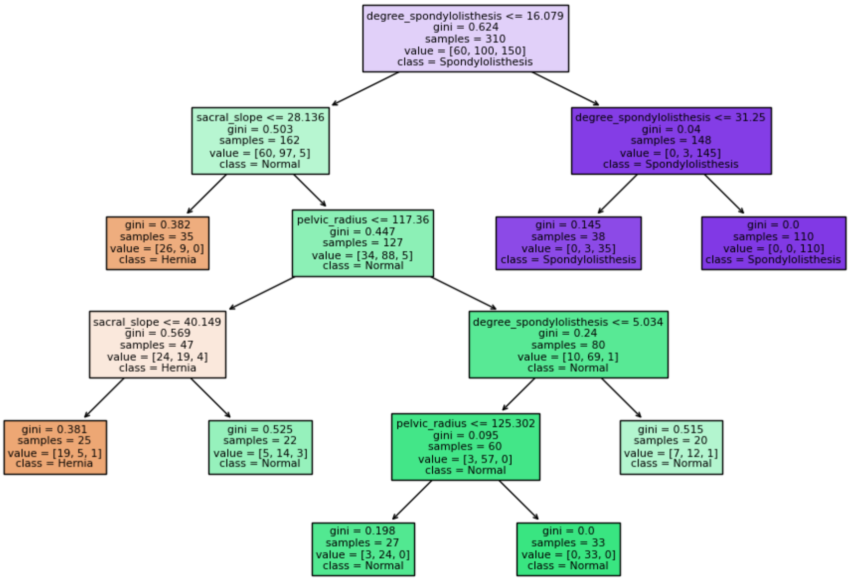


3) Answer 3

Analisando a training accuracy, representada em azul, podemos ver que ela aumenta à medida que o limite de profundidade da árvore aumenta, o que nos mostra que esta árvore de decisão tem uma boa performance nos treinos pelos quais foi submetida. A testing accuracy, por sua vez, também aumenta à medida que o limite de profundidade aumenta, o que nos mostra que o modelo generaliza melhor mesmo com o aumento da complexidade do modelo. Porém, por volta do limite de profundidade 4, a testing accuracy fica constante, até começando a diminuir no limite de profundidade 5. Esta descida na testing accuracy sugere que estamos com um problema de overfitting. O modelo tem um bom desempenho nos treinos, porém não consegue transitar essa boa performance para dados não vistos, acabando por reduzir a testing accuracy. Por último, temos a escolha de profunidade ótima. Tal como o nome indica, este parâmetro designa o pico de precisão de teste (que por observação do gráfico se encontra em torno do limite de profundidade 3 e 4) e a capacidade de generalização óptima do modelo. Se ultrapassarmos este ponto ideal, , o modelo torna-se demasiado complexo e ocorre overfitting, o que resulta numa diminuição da precisão do teste.

Answer 4 (i)

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

from scipy.io import arff

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

df['class'] = df['class'].str.decode('utf-8')

le = LabelEncoder()
df['class'] = le.fit_transform(df['class'])

X = df.drop(columns='class')
y = df['class']

clf = DecisionTreeClassifier(random_state=0, min_samples_leaf=20)
clf.fit(X, y)

plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=le.classes_)

plt.savefig('plot.png')
plt.show()
```


degree_spondylolisthesis <= 16.079
gini = 0.624
samples = 310
value = [60, 100, 150]
class = Spondylolisthesis
sacral_slope <= 28.136
gini = 0.503
samples = 162
value = [60, 97, 5]
class = Normal
degree_spondylolisthesis <= 31.25
gini = 0.04
samples = 148
value = [0, 3, 145]
class = Spondylolisthesis
gini = 0.382
samples = 35
value = [26, 9, 0]
class = Hernia
pelvic_radius <= 117.36
gini = 0.447
samples = 127
value = [34, 88, 5]
class = Normal
gini = 0.145
samples = 38
value = [0, 3, 35]
class = Spondylolisthesis
gini = 0.0
samples = 110
value = [0, 0, 110]
class = Spondylolisthesis
sacral_slope <= 40.149
gini = 0.569
samples = 47
value = [24, 19, 4]
class = Hernia
degree_spondylolisthesis <= 5.034
gini = 0.24
samples = 80
value = [10, 69, 1]
class = Normal
gini = 0.381
samples = 25
value = [19, 5, 1]
class = Hernia
gini = 0.525
samples = 22
value = [5, 14, 3]
class = Normal
pelvic_radius <= 125.302
gini = 0.095
samples = 60
value = [3, 57, 0]
class = Normal
gini = 0.515
samples = 20
value = [7, 12, 1]
class = Normal
gini = 0.198
samples = 27
value = [3, 24, 0]
class = Normal
gini = 0.0
samples = 33
value = [0, 33, 0]
class = Normal

4) Answer 4 (ii)

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from scipy.io import arff

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

unique_classes = df['class'].unique()
print("Unique Classes:", unique_classes)

label_encoder = LabelEncoder()
df['class'] = label_encoder.fit_transform(df['class'])
X = df.drop(columns=['class'])
y = df['class']

clf = DecisionTreeClassifier(random_state=0, min_samples_leaf=20)
clf.fit(X, y)

feature_importances = clf.feature_importances_

associations_df = pd.DataFrame({'Feature': X.columns, 'Association Importance':
feature_importances})
associations_df = associations_df.sort_values(by='Association Importance',
ascending=False)

print("Hernia-Conditional Associations (Feature Importance):")
print(associations_df)
```

Output:

```
Unique Classes: [b'Hernia' b'Spondylolisthesis' b'Normal']
Hernia-Conditional Associations (Feature Importance):
                   Feature  Association Importance
5  degree_spondylolisthesis                0.795233
3               sacral_slope               0.123995
4               pelvic_radius              0.080772
0            pelvic_incidence              0.000000
1                 pelvic_tilt              0.000000
2         lumbar_lordosis_angle            0.000000
```

END