

Aprendizagem

Instituto Superior Técnico

setembro de 2023

Homework 2 - Report

Joana Pimenta (103730), Rodrigo Laia (102674)

Pen and Paper

- (a) y_1, y_2, y_3, y_4 and y_5 independent $\implies p(y_1, y_2, y_3, y_4, y_5) = p(y_1, y_2) \times p(y_3, y_4) \times p(y_5)$

Fórmulas utilizadas:

$$P(y_6 = H|\vec{x}) = \frac{P(\vec{x}|y_6 = H)}{P(\vec{x})} \quad (1)$$

$$P(\vec{x}|\mu, \sigma^2) = \frac{1}{(2\pi)^{m/2} \sqrt{|\Sigma|}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu})^T \cdot \Sigma^{-1} \cdot (\vec{x}-\vec{\mu})} \quad (2)$$

$$\vec{\mu} = \begin{bmatrix} E(y_1) \\ E(y_2) \end{bmatrix} \quad (3)$$

$$cov(x, y) = \sum_{i=1}^n \frac{(x_i - E(x))(y_i - E(y))}{n} \quad (4)$$

$$\Sigma = \begin{bmatrix} cov(y_1, y_2) & cov(y_1, y_1) \\ cov(y_2, y_1) & cov(y_2, y_2) \end{bmatrix} \quad (5)$$

$$|\Sigma| = cov(y_1, y_2) \cdot cov(y_2, y_1) - cov(y_1, y_1) \cdot cov(y_2, y_2) \quad (6)$$

$$\Sigma^{-1} = \frac{1}{|\Sigma|} \cdot \begin{bmatrix} cov(y_2, y_2) & -cov(y_1, y_2) \\ -cov(y_2, y_1) & cov(y_1, y_1) \end{bmatrix} \quad (7)$$

Parâmetros das gaussianas multivariadas:

Classe A:

$$\vec{\mu}_A = \begin{bmatrix} 0.24 \\ 0.52 \end{bmatrix}$$

$$\Sigma_A = \begin{bmatrix} 0.004267 & -0.0064 \\ -0.0064 & 0.02240 \end{bmatrix}$$

$$|\Sigma|_A = 5.4613 \cdot 10^{-5}$$

$$\Sigma_A^{-1} = \begin{bmatrix} 410.1563 & -117.1875 \\ -117.1875 & 78.125 \end{bmatrix}$$

$$P(\vec{x}|A) = N(\vec{x}|\mu_A, \Sigma_A) = \frac{1}{(2\pi)^{m/2} \sqrt{|\Sigma_A|}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu}_A)^T \cdot \Sigma_A^{-1} \cdot (\vec{x}-\vec{\mu}_A)}$$

Classe B:

$$\vec{\mu}_B = \begin{bmatrix} 0.5925 \\ 0.3275 \end{bmatrix}$$

$$\Sigma_B = \begin{bmatrix} 0.01717 & -0.00732 \\ -0.00732 & 0.02362 \end{bmatrix}$$

$$|\Sigma|_B = 3.519 \cdot 10^{-4}$$

$$\Sigma_B^{-1} = \begin{bmatrix} 67.1101 & 20.7954 \\ 20.7954 & 48.7831 \end{bmatrix}$$

$$P(\vec{x}|B) = N(\vec{x}|\mu_B, \Sigma_B) = \frac{1}{(2\pi)^{m/2} \sqrt{|\Sigma_B|}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu}_B)^T \cdot \Sigma_B^{-1} \cdot (\vec{x}-\vec{\mu}_B)}$$

Probabilidades para $\{y_3, y_4\}$ condicionadas a A e B :

Classe A:

	$y_3 = 0$	$y_3 = 1$
$y_4 = 0$	P=0	P=1/3
$y_4 = 1$	P=1/3	P=1/3

Tabela 1: Probabilidades para y_3, y_4 condicionadas a A

Classe B:

	$y_3 = 0$	$y_3 = 1$
$y_4 = 0$	P=1/2	P=1/4
$y_4 = 1$	P=1/4	P=0

Tabela 2: Probabilidades para y_3, y_4 condicionadas a B

Probabilidades para $\{y_5\}$ condicionadas a A e B :

Classe A:

$$P(y_5 = 0|A) = 1/3$$

$$P(y_5 = 1|A) = 1/3$$

$$P(y_5 = 2|A) = 1/3$$

Classe B:

$$P(y_5 = 0|A) = 1/4$$

$$P(y_5 = 1|A) = 1/2$$

$$P(y_5 = 2|A) = 1/4$$

Priors:

$$P(A) = \frac{3}{7}$$

$$P(B) = \frac{4}{7}$$

- (b) Uma vez que o denominador é o mesmo para todas para saber qual a classe mais provável, basta comparar os numeradores das probabilidades.

$$\begin{aligned} P(A|\vec{x}_8) &= \frac{P(\vec{x}_8|A) \cdot P(A)}{P(\vec{x}_8)} \\ &= \frac{P(y_1 = 0.38, y_2 = 0.52|A) \cdot P(y_3 = 0, y_4 = 1|A) \cdot P(y_5 = 0|A) \cdot P(A)}{P(\vec{x}_8)} \\ &= \frac{\frac{3}{7} \cdot 0.3868 \cdot \frac{1}{3} \cdot \frac{1}{3}}{P(\vec{x}_8)} \\ &= \frac{0.018}{P(\vec{x}_8)} \end{aligned}$$

$$\begin{aligned} P(B|\vec{x}_8) &= \frac{P(\vec{x}_8|B) \cdot P(B)}{P(\vec{x}_8)} \\ &= \frac{P(y_1 = 0.38, y_2 = 0.52|B) \cdot P(y_3 = 0, y_4 = 1|B) \cdot P(y_5 = 0|B) \cdot P(B)}{P(\vec{x}_8)} \\ &= \frac{\frac{4}{7} \cdot 1.7678 \cdot \frac{1}{4} \cdot \frac{1}{4}}{P(\vec{x}_8)} \\ &= \frac{0.063}{P(\vec{x}_8)} \end{aligned}$$

Como $P(A|\vec{x}_8) < P(B|\vec{x}_8)$, então \vec{x}_8 é classificado como B.

$$\begin{aligned}
 P(A|\vec{x}_9) &= \frac{P(\vec{x}_9|A) \cdot P(A)}{P(\vec{x}_9)} \\
 &= \frac{P(y_1 = 0.42, y_2 = 0.59|A) \cdot P(y_3 = 0, y_4 = 1|A) \cdot P(y_5 = 0|A) \cdot P(A)}{P(\vec{x}_9)} \\
 &= \frac{\frac{3}{7} \cdot 0.1013 \cdot \frac{1}{3} \cdot \frac{1}{3}}{P(\vec{x}_9)} \\
 &= \frac{0.0048}{P(\vec{x}_9)}
 \end{aligned}$$

$$\begin{aligned}
 P(B|\vec{x}_8) &= \frac{P(\vec{x}_8|B) \cdot P(B)}{P(\vec{x}_8)} \\
 &= \frac{P(y_1 = 0.42, y_2 = 0.59|B) \cdot P(y_3 = 0, y_4 = 1|B) \cdot P(y_5 = 1|B) \cdot P(B)}{P(\vec{x}_8)} \\
 &= \frac{\frac{4}{7} \cdot 1.4927 \cdot \frac{1}{4} \cdot \frac{1}{2}}{P(\vec{x}_8)} \\
 &= \frac{0.1066}{P(\vec{x}_8)}
 \end{aligned}$$

Como $P(A|\vec{x}_9) < P(B|\vec{x}_9)$, então \vec{x}_9 é classificado como B.

- (c) Assumindo o critério de Maximum Likelihood, para classificar uma observação apenas interessam as probabilidades $P(\vec{x}|A)$ e $P(\vec{x}|B)$:

$$h = \operatorname{argmax}(P(\vec{x}_8|h))$$

Considerando diferentes thresholds θ para as probabilidades é possível maximizar a accuracy do nosso classificador:

$$f(\vec{x}_8) = \begin{cases} A & \text{se } P(A|\vec{x}_8) > \theta \\ B & \text{otherwise} \end{cases}$$

$$P(\vec{x}_8|A) = P(y_1 = 0.38, y_2 = 0.52|A) \cdot P(y_3 = 0, y_4 = 1|A) \cdot P(y_5 = 0|A) = 0.043$$

$$P(\vec{x}_9|A) = P(y_1 = 0.42, y_2 = 0.59|A) \cdot P(y_3 = 0, y_4 = 1|A) \cdot P(y_5 = 0|A) = 0.0113$$

Assumindo o critério de maximum Likelihood os priors são todos iguais. Escolhendo qualquer valor no intervalo (0.0113, 0.043) como threshold θ , a accuracy do classificador é de 100% para estas observações de teste.

2. As fórmulas utilizadas neste exercício são:

$$\hat{z} = \frac{\sum_{i=1}^k w_i \cdot z_i}{\sum_{i=1}^k w_i}$$

$$MAE = \frac{\frac{1}{n}}{\sum_{i=1}^n |z_i - \hat{z}|}$$

Para discretizar a variável y_2 , considerando equal-width, é necessário dividir o intervalo $[0, 1]$ em 2 partes iguais. Assim, os intervalos são: $[0, 0.5]$, $[0.5, 1]$.

Para cada observação, y_2 pode assumir os valores 0 ou 1, consoante o intervalo em que se encontra o seu valor.

y_2	0.36	0.48	0.72	0.11	0.39	0.28	0.53	0.52	0.59
y_{2new}	0	0	1	0	0	0	1	1	1

(a) Em 3-Fold cross-validation o dataset é dividido em 3 partes iguais. Duas delas são usadas para teste e uma para treino.

Porque não há shuffling:

Tabela 3: 1^o Fold

x_1	0	1	1	0	A
x_2	0	1	0	1	A
x_3	1	0	1	2	A

Tabela 4: 2^o Fold

x_1	0	0	0	1	B
x_2	0	0	0	0	B
x_3	0	1	0	2	B

Tabela 5: 3^o Fold

x_1	1	0	1	1	B
x_2	1	0	1	0	A
x_3	1	0	1	1	B

(b)

D	y_1	y_2	y_3	y_4	y_5	y_6	
x_1	0.24	0	1	1	0	A	TRAIN
x_2	0.16	0	1	0	1	A	
x_3	0.32	1	0	1	2	A	
x_4	0.54	0	0	0	1	B	
x_5	0.66	0	0	0	0	B	
x_6	0.76	0	1	0	2	B	
x_7	0.41	1	0	1	1	B	TEST
x_8	0.38	1	0	1	0	A	
x_9	0.42	1	0	1	1	B	

Para x_7 :

d	x_1	x_2	x_3	x_4	x_5	x_6
x_7	4	4	2	2	3	4

As observações com menor distância de Hamming são x_3 , x_4 e x_5 . O output previsto pelo kNN é dado por

$$\hat{z}_7 = \frac{1/2 \cdot 0.32 + 1/2 \cdot 0.54 + 1/3 \cdot 0.66}{1/2 + 1/2 + 1/3} = 0.4875$$

Para x_8 :

d	x_1	x_2	x_3	x_4	x_5	x_6
x_8	2	4	1	4	3	5

As observações com menor distância de Hamming são x_1 , x_3 e x_5 . O output previsto pelo kNN é dado por

$$\hat{z}_8 = \frac{1/2 \cdot 0.24 + 1 \cdot 0.32 + 1/3 \cdot 0.66}{1/2 + 1 + 1/3} = 0.36$$

Como as observações x_7 e x_9 são iguais, o output previsto pelo kNN para x_9 é igual ao output previsto para x_7 .

$$\hat{z}_9 = \hat{z}_7 = 0.4875$$

Por fim, o MAE deste classificador é dado por:

$$MAE = \frac{1}{3} \cdot (|0.41 - 0.4875| + |0.38 - 0.36| + |0.42 - 0.4875|) = 0.055$$

Programming - Código Python e Resultados Obtidos

1. O objetivo deste exercício é comparar a performance de dois classificadores (Naive Bayes com distribuição Gaussiana e kNN) para o dataset.
 - (a) Boxplots para as accuracies obtidas para GaussianNB e kNN:

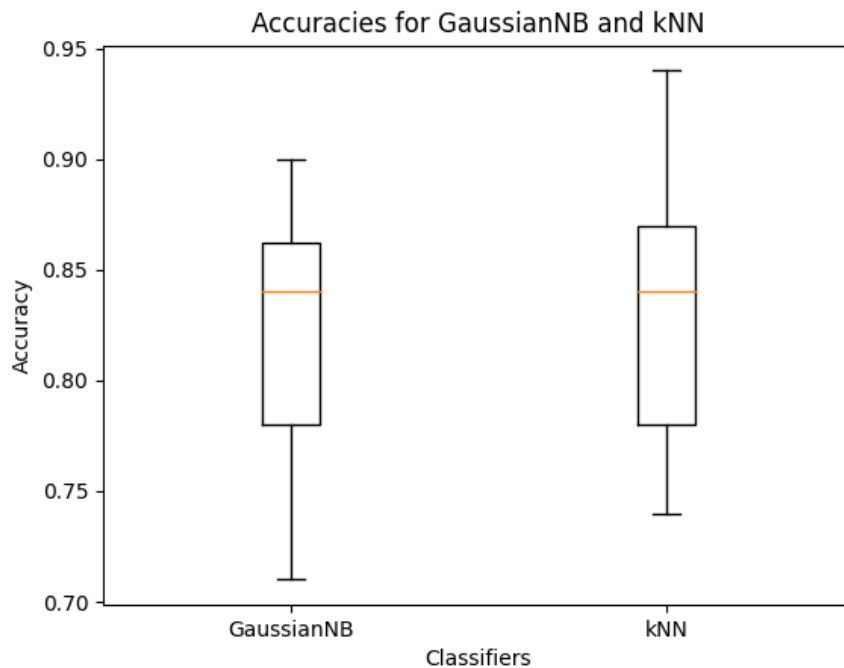


Figura 1: Boxplot para as accuracies obtidas para GaussianNB e kNN

Conclui-se que apesar do o classificador kNN atingir accuracies maiores do que o GaussianNB, a mediana de ambos é quase igual. Além disso, o kNN tem accuracies ligeiramente mais dispersas (a sua distância interquartis é maior). A amplitude do boxplot é parecida para ambos os classificadores.

Código utilizado:

```
1  ### Exercise 1 ###
2
3  #a)
4
5  acc_folds_gauss = []
6  acc_folds_knn = []
7  folds = StratifiedKFold(n_splits=10, shuffle=True, random_state=0)
8
9  # Gaussian Naive Bayes
10 gaussNB = GaussianNB()
11
12 # KNN
13 knn_predictor = KNeighborsClassifier(n_neighbors=5)
14
15 # iterate per fold
16 for train_k, test_k in folds.split(features, target):
17     X_train, X_test = features.iloc[train_k], features.iloc[test_k]
```

```

18     y_train, y_test = target.iloc[train_k], target.iloc[test_k]
19
20     ## train and assess
21     gaussNB.fit(X_train, y_train)
22     y_pred_gauss = gaussNB.predict(X_test)
23     acc_folds_gauss.append(round(metrics.accuracy_score(y_test,
24     y_pred_gauss),2))
25
26     knn_predictor.fit(X_train, y_train)
27     y_pred_knn = knn_predictor.predict(X_test)
28     acc_folds_knn.append(round(metrics.accuracy_score(y_test,
29     y_pred_knn),2))
30
31     print("Fold accuracies GaussianNB:", acc_folds_gauss)
32     print("Fold accuracies kNN:", acc_folds_knn)
33
34     plt.boxplot([acc_folds_gauss, acc_folds_knn], labels=['GaussianNB',
35     'kNN'])
36
37     plt.title('Accuracies for GaussianNB and kNN')
38     plt.xlabel('Classifiers')
39     plt.ylabel('Accuracy')
40     plt.savefig('ex1a_boxplot.png')
41     plt.show()

```

(b) Concluimos que a hipótese nula (H_0 : kNN não é estatisticamente superior a GaussianNB) não é rejeitada, uma vez que o p-value é maior do que 0.05, logo kNN não é estatisticamente superior a GaussianNB.

Código utilizado:

```

1  #b)
2  Null Hypothesis: kNN is not statistically superior to GaussianNB
3  hypothesis = stats.ttest_rel(acc_folds_knn, acc_folds_gauss,
4  alternative='greater')
5
6  if hypothesis[1] < 0.05:
7      print("The null hypothesis is rejected and kNN is statistically
8      superior to GaussianNB")
9
10 else:
11     print("The null hypothesis is not rejected and there is no
12     statistical superiority between kNN and GaussianNB")

```

- O objetivo deste exercício é comparar a performance de dois classificadores : kNN com $k=1$ e kNN com $k=5$. Assim, calculamos as matrizes de confusão para os dados de cada uma das 10 iterações, somamos os valores de cada célula obtendo duas matrizes de confusão cumulativas que subtraímos uma a outra (kNN com $k=1$ - kNN com $k=5$).

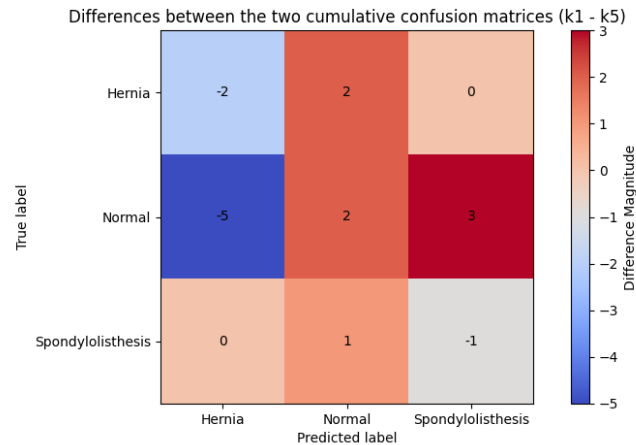


Figura 2: Matriz de confusão cumulativa para kNN com k=1 - kNN com k=5

CONCLUSÕESSSS FAAAAFIIIIII

```

1 ##### Exercise 2 #####
2
3 #Initialize the cumulative confusion matrices
4 cum_conf_matrix1 = np.zeros((3,3))
5 cum_conf_matrix5 = np.zeros((3,3))
6
7 for train_k, test_k in folds.split(features, target):
8     X_train, X_test = features.iloc[train_k], features.iloc[
9         test_k]
10     y_train, y_test = target.iloc[train_k], target.iloc[
11         test_k]
12
13     ## train and assess
14     knn1 = KNeighborsClassifier(n_neighbors=1, weights='
15     uniform', metric='euclidean')
16     knn5 = KNeighborsClassifier(n_neighbors=5, weights='
17     uniform', metric='euclidean')
18
19     knn1.fit(X_train, y_train)
20     knn5.fit(X_train, y_train)
21
22     # Make predictions
23     y_pred1 = knn1.predict(X_test)
24     y_pred5 = knn5.predict(X_test)
25
26     # Calculate confusion matrices
27     conf_matrix1 = confusion_matrix(y_test, y_pred1)
28     conf_matrix5 = confusion_matrix(y_test, y_pred5)
29
30     # Calculate cumulative confusion matrices
31     cum_conf_matrix1 += conf_matrix1
32     cum_conf_matrix5 += conf_matrix5
33
34     #Calculate the difference between the two confusion
35     matrices
36     conf_matrix_diff = cum_conf_matrix1 - cum_conf_matrix5
37
38     confusion1 = pd.DataFrame(conf_matrix_diff, index=knn1.

```

```

34     classes_, columns=['Predicted Hernia', 'Predicted Normal', '
    Predicted Spondylolisthesis'])
35
36     #Plotting
37     plt.figure(figsize=(10, 5))
38     heatmap = plt.imshow(conf_matrix_diff, cmap="coolwarm",
    interpolation='nearest')
39     plt.title('Differences between the two cumulative confusion
    matrices (k1 - k5)')
40     plt.xlabel('Predicted label')
41     plt.xticks([0, 1, 2], ['Hernia', 'Normal', '
    Spondylolisthesis'])
42     plt.yticks([0, 1, 2], ['Hernia', 'Normal', '
    Spondylolisthesis'])
43     plt.ylabel('True label')
44
45     cbar = plt.colorbar(heatmap)
46     cbar.set_label('Difference Magnitude', rotation=90)
47
48     for i in range(conf_matrix_diff.shape[0]):
49         for j in range(conf_matrix_diff.shape[1]):
50             plt.text(j, i, str(int(conf_matrix_diff[i, j])), ha
    ='center', va='center', color='black')
51     plt.savefig('ex2_cummatrix.png')
52     plt.show()

```

3. Apesar de ser uma abordagem fácil e rápida, o Naive Bayes apresenta algumas desvantagens para este dataset. Um problema é, por exemplo, a suposição de que as features têm uma distribuição Gaussiana. Para podermos visualizar a distribuição experimental destes features fizemos um histograma para cada um deles e concluímos que a maioria não tem uma distribuição Gaussiana, especialmente por exemplo a feature *degree_Spondylolisthesis*.

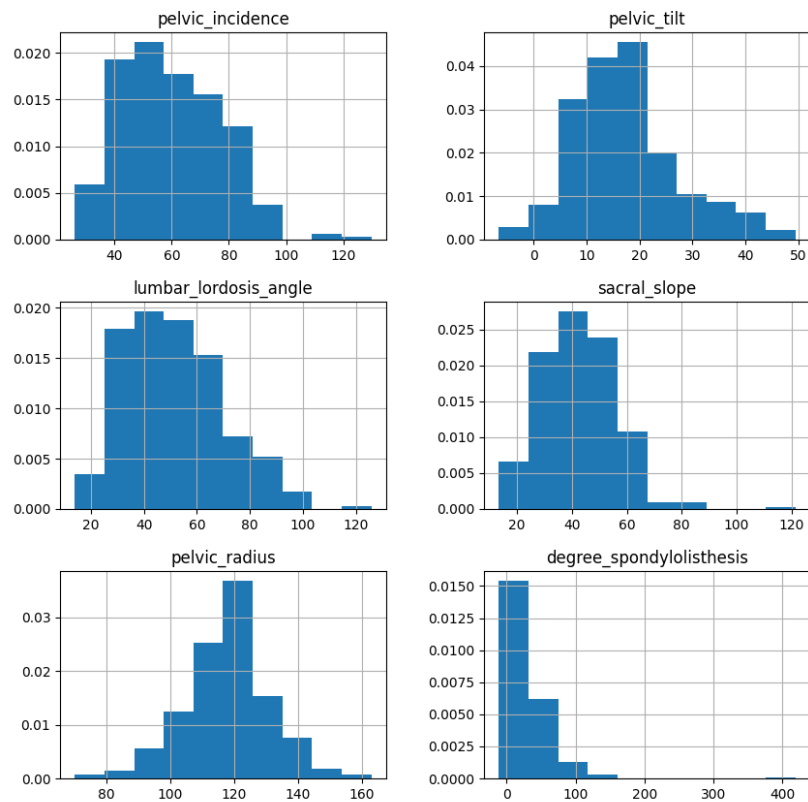


Figura 3: Histogramas para cada feature

Código utilizado:

```

1 ##### Exercise 3 #####
2 #1. The dataset is not normally distributed, which is an
  assumption of the Naive Bayes classifier.
3 #Histograms for each feature:
4 features.hist(figsize=(10,10),density=True)
5 plt.savefig('ex3_1_hist.png')
6 plt.show()

```

Para além disso, outro exemplo de uma dificuldade do Naive Bayes é a suposição de que as features são independentes. Para avaliar se estas features eram independentes ou não, fizemos a seguinte matriz de correlação:

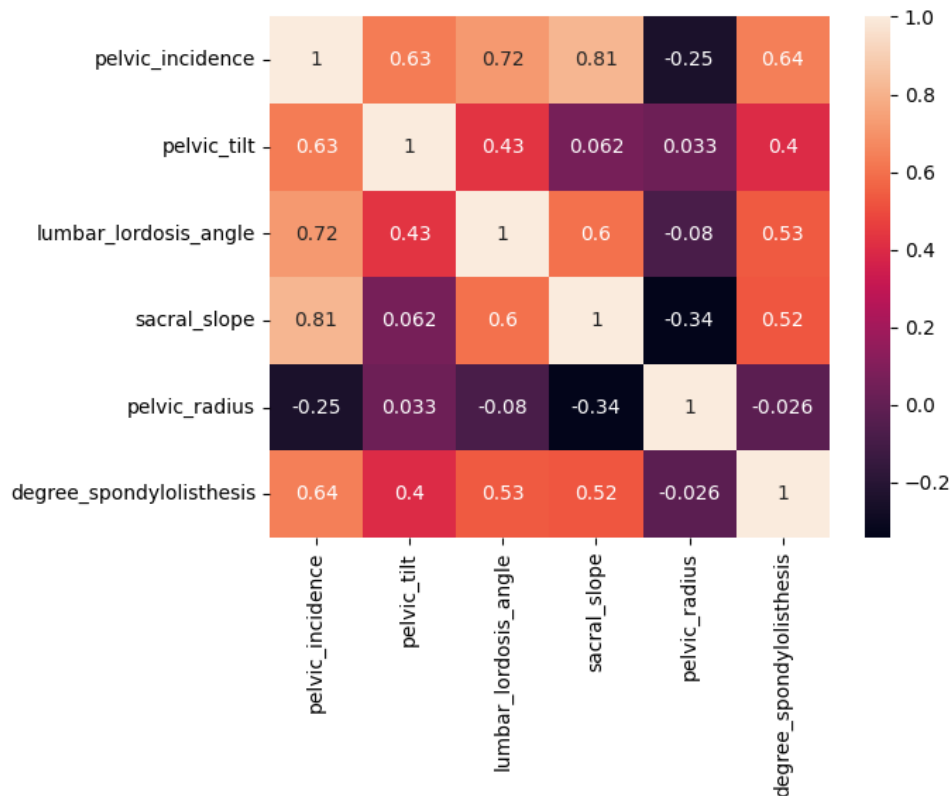


Figura 4: Matriz de correlação

Ao observar a matriz de correlação, concluímos que apesar de algumas features se encontrarem muito pouco correladas, podendo por isso ser aproximadas como independentes (como por exemplo *pelvic_radius* e *pelvic_tilt*), há outras que se encontram bastante correladas e consequentemente não podem ser aproximadas como independentes (como por exemplo *pelvic_incidence* e *sacral_slope*).

Código utilizado: