Aprendizagem

Instituto Superior Técnico

outubro de 2023

Homework 4 - Report

Joana Pimenta (103730), Rodrigo Laia (102674)

Pen and Paper

1. Fórmulas utilizadas:

$$\gamma_{ki} = p(c_k|\mathbf{x}_i) = \frac{p(c_k)p(\mathbf{x}_i|c_k)}{p(\mathbf{x}_i)} \tag{1}$$

$$p(\mathbf{x}_i) = p(c_1)p(\mathbf{x}_i|c_1) + p(c_2)p(\mathbf{x}_i|c_2) \tag{2}$$

$$p(\mathbf{x}_i|c_k) = \begin{cases} p_k \cdot \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) & \text{se } y_1 = 1 \\ (1-p_k) \cdot \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) & \text{se } y_1 = 0 \end{cases} \tag{3}$$

E-step:

Cálculo das probabilidades $p(\mathbf{x}_i)$

$$p(\mathbf{x}_1) = 0.05185$$

$$p(\mathbf{x}_2) = 0.02775$$

$$p(\mathbf{x}_3) = 0.04337$$

$$p(\mathbf{x}_4) = 0.05243$$

Cálculos dos γ_{ki}

$$\gamma_{k=1,i=1} = 0.19259$$

$$\gamma_{k=2,i=1} = 0.80741$$

$$\gamma_{k=1,i=2} = 0.63135$$

$$\gamma_{k=2,i=2} = 0.36865$$

$$\gamma_{k=1,i=3} = 0.55181$$

$$\gamma_{k=2,i=3} = 0.44819$$

$$\gamma_{k=1,i=4} = 0.16892$$

$$\gamma_{k=2,i=4} = 0.83108$$

M-step:

Cada observação $\mathbf{x}_i$ permite atualizar os parâmetros com peso γ_{ki}. Assim calculamos os novos parâmetros atualizados para cada cluster utilizando as seguintes fórmulas.

$$N_k = \sum_{i=1}^{4} \gamma_{ki} \tag{4}$$

$$\pi_k = \frac{N_k}{N} \tag{5}$$

$$P_k(y_1 = 1) = \frac{\sum_{i=1}^{4} \gamma_{ki} \cdot p(y_1 = 1|\mathbf{x}_i)}{\sum_{i=1}^{4} \gamma_{ki}} \tag{6}$$

Nota: A probabilidade $p(y_1 = 1|\mathbf{x}_i)$ é 1 se y_1 de $\mathbf{x}_i$ for 1 e 0 caso contrário.

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{4} \gamma_{ki} \mathbf{x}_i \tag{7}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{4} \gamma_{ki} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \tag{8}$$

Parâmetros atualizados:

$$\pi_1 = 0.38617$$

$$\pi_2 = 0.61383$$

$$P_{k=1}(y_1 = 1) = 0.23404$$

$$P_{k=2}(y_1 = 1) = 0.66732$$

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 0.02651 \\ 0.50713 \end{bmatrix}$$

$$\boldsymbol{\mu}_2 = \begin{bmatrix} 0.30914 \\ 0.21042 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 0.14137 & -0.10541 \\ -0.10541 & 0.09605 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_2 = \begin{bmatrix} 0.10829 & -0.08865 \\ -0.08865 & 0.10412 \end{bmatrix}$$

2. Considerando o critério de *máximo a posteriori* Para decidir a que cluster pertence a observação $\mathbf{x}_{new}$ precisamos de calcular o seu posterior. Para isso utilizamos a seguinte fórmula:

$$p(cluster = k|\mathbf{x}_{new}) = \frac{p(cluster = k)p(\mathbf{x}_{new}|cluster = k)}{p(\mathbf{x}_{new})} \tag{9}$$

Cálculos:

$$p(\mathbf{x}_{new}) = 0.03048$$

$$p(cluster = 1|\mathbf{x}_{new}) = 0.08029$$

$$p(cluster = 2|\mathbf{x}_{new}) = 0.91971$$

Como $p(cluster = 2|\mathbf{x}_{new}) > p(cluster = 1|\mathbf{x}_{new})$, então a observação $\mathbf{x}_{new}$ pertence ao cluster 2.

3. Neste exercício assumimos que o cluster atribuído a cada observação é é escolhido pelo critério de *maximum likelihood*. Assim, o cluster escolhido é dado por:

$$cluster = \arg\max_k p(\mathbf{x}_i|cluster = k) \tag{10}$$

Em que $p(\mathbf{x}_i|cluster = k)$ é dado pela fórmula (3).

Assim,

Observação	$p(\mathbf{x}_i\|cluster = 1)$	$p(\mathbf{x}_i\|cluster = 2)$	Cluster atribuído
$\mathbf{x_1}$	0.59941	1.54691	2
$\mathbf{x_2}$	1.26633	0.08874	1
$\mathbf{x_3}$	1.43811	0.45417	1
$\mathbf{x_4}$	0.02077	0.72331	2

Coeficiente de Silhueta:

$$s_i = 1 - \frac{a(\mathbf{x}_i)}{b(\mathbf{x}_i)} \tag{11}$$

em que $a(\mathbf{x}_i)$ é a distância média entre $\mathbf{x}_i$ e as outras observações no mesmo cluster e $b(\mathbf{x}_i)$ é a distância média entre $\mathbf{x}_i$ e as observações no outro cluster.

A silhueta de um cluster é dada pela média dos coeficientes de silhueta de todas as observações pertencentes a esse cluster.

A silhueta da solução é por sua vez dada pela média das silhuetas de todos os clusters.

Neste caso a distância considerada é a distância de Manhattan, logo:

$$d(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{n} |u_i - v_i| \tag{12}$$

Assim, as silhuetas obtidas foram:

cluster	$\mathbf{x}_i$	a($\mathbf{x}_i$)	b($\mathbf{x}_i$)	s($\mathbf{x}_i$)	s(cluster)	s(sol)
1	$\mathbf{x}_2$	0.3(9)	2.25	0.(6)	0.58(3)	0.702(7)
	$\mathbf{x}_3$	0.9	2.7	0.4(9)		
2	$\mathbf{x}_1$	0.9	1.7(9)	0.8(2)	0.8(2)	
	$\mathbf{x}_4$	0.3(9)	2.25	0.8(2)		

4. A purity é dada por:

$$\text{purity} = \frac{1}{N} \sum_{k=1}^{K} \max_j |c_k \cap t_j| = \frac{1}{N} \left(\max_j |c_1 \cap t_j| + \max_j |c_2 \cap t_j| \right) \tag{13}$$

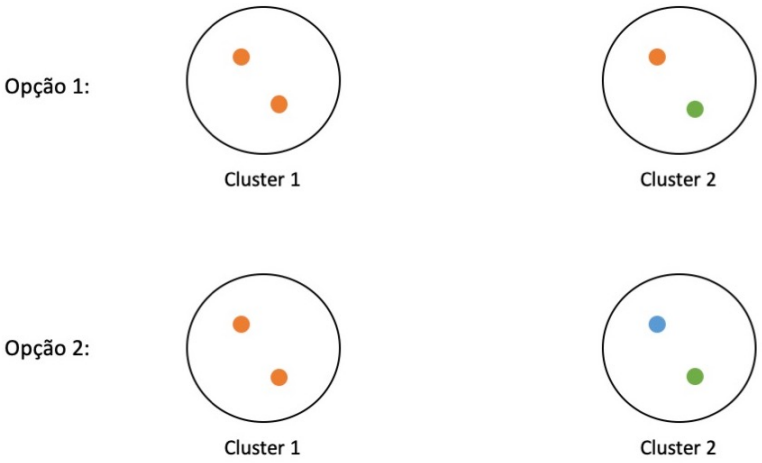
Uma vez que temos uma purity de 0.75 e um número total de observações de 4, então $\frac{1}{N}\left(\max_j |c_1 \cap t_j| + \max_j |c_2 \cap t_j|\right) = 0.75 \times 4 = 3$

Logo podemos ter os seguintes casos:

(a) $\max_j |c_1 \cap t_j| = 3$ e $\max_j |c_2 \cap t_j| = 0$
(b) $\max_j |c_1 \cap t_j| = 2$ e $\max_j |c_2 \cap t_j| = 1$
(c) $\max_j |c_1 \cap t_j| = 1$ e $\max_j |c_2 \cap t_j| = 2$
(d) $\max_j |c_1 \cap t_j| = 0$ e $\max_j |c_2 \cap t_j| = 3$

As opções (a) e (d) não são possíveis porque o cluster 2 e 1 só têm 2 observações cada um.

Opção (b)
Neste caso, as observações do cluster 1 são as duas classificadas corretamente. No cluster 2 uma é corretamente identificada e a outra não. Assim, as observações no cluster 1 têm a mesma classificação; uma das observações do cluster 2 tem classificação diferente das do cluster 1 e a outra pode ter classificação igual às do cluster 1 (opção 1) ou diferente, sendo que neste caso é também diferente da classificação do outra observação do cluster 2 (opção 2). Assim, conclui-se que o número verdadeiro de classes pode ser 2 ou 3.
Para visualizar melhor as opções possíveis fizemos os seguintes esquemas (bolas de cores diferentes representam classes verdadeiras diferentes):



Opção (c)
O raciocínio é semelhante ao da opção (b). Conclui-se que o número verdadeiro de classes pode ser 2 ou 3.

Programming - Código Python e Resultados Obtidos

1. Usando o sklearn, aplicámos clustering K-means totalmente não supervisionada nos dados normalizados com valores de k ∈ 2, 3, 4, 5 (usando random state=0). Calculámos a silhueta e a pureza para cada valor de k.

   Os gráficos obtidos foram os seguintes:

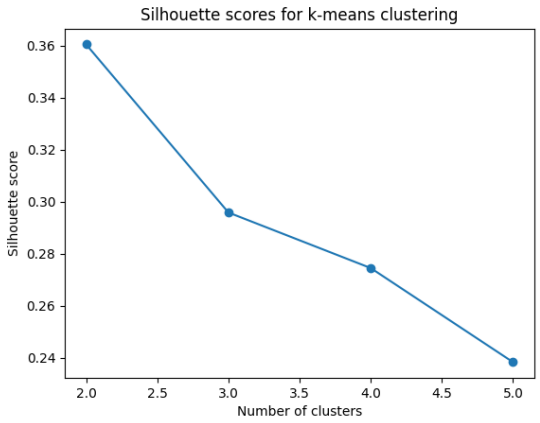


Figura 1: Valores de silhueta para diferentes k

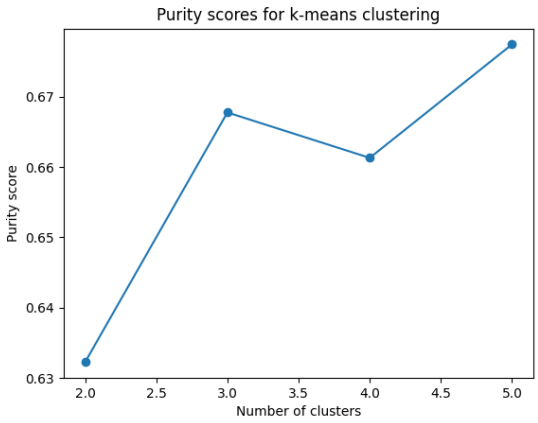


Figura 2: Valores de pureza para diferentes k

   Foi possível observar que, de um modo geral, a pureza aumenta com o número de clusters, enquanto que a silhueta diminui. Ou seja, com o aumento do k, os clusters ficam pior definidos (menos coesos e mais juntos porque a silhueta diminui) mas também mais homogéneos (pureza aumenta).

   Código Utilizado:

```
def purity_score(y_true, y_pred):
    # compute contingency/confusion matrix
    confusion_matrix = metrics.cluster.contingency_matrix(y_true,
   y_pred)
    return np.sum(np.amax(confusion_matrix, axis=0)) / np.sum(
   confusion_matrix)

silhouettes = []
purities = []
for k in range(2, 6):
    kmeans_algo = cluster.KMeans(n_clusters=k, random_state=0)
    kmeans_model = kmeans_algo.fit(features_scaled)
    target_pred = kmeans_model.labels_
    purity = purity_score(target, target_pred)
    silhouette = metrics.silhouette_score(features_scaled,
   target_pred)
    silhouettes.append(silhouette)
    purities.append(purity)

    print("Purity score for k = " , str(k) , " is " , purity)
    print("Silhouette score for k = " , str(k) , " is " ,
   silhouette)
```

```
19
20 #Plot Silhouette
21 plt.plot([2,3,4,5], silhouettes, 'o-')
22 plt.title('Silhouette scores for k-means clustering')
23 plt.xlabel('Number of clusters')
24 plt.ylabel('Silhouette score')
25 plt.savefig('ex1_silhouette.png')
26 plt.show()
27
28 #Plot Purity
29 plt.plot([2,3,4,5], purities, 'o-')
30 plt.title('Purity scores for k-means clustering')
31 plt.xlabel('Number of clusters')
32 plt.ylabel('Purity score')
33 plt.savefig('ex1_purity.png')
34 plt.show()
```

2. Considerando a aplicação do PCA depois da normalização dos dados, identificámos a variabilidade explicada pelos dois primeiros componentes principais.

   Concluímos que as componentes principais (eigenvectors) são:

   - PC1: (0.59162, 0.46704, 0.51508, 0.32569, -0.11582, 0.21693)
   - PC2: (0.10004, -0.67037, 0.08005, 0.44333, -0.58107, 0.00458)

   Obtivemos os seguintes valores para a variabilidade explicada:
   Explained variance (ratio) = (0.56181, 0.20956)

   Somando as variabilidades explicadas das duas componentes principais, obtemos 0.77137, ou seja, 77.137% da variabilidade é explicada pelas mesmas. Neste caso, uma variabilidade considerável é explicada com apenas duas dimensões. O PCA é bastante útil porque permite reduzir as dimensões do dataset utilizando a informação mais relevante para estudar o mesmo. Consoante o nível de conhecimento que pretendemos ter sobre a amostra, podemos utilizar mais componentes.

   Para cada uma das duas componentes principais, ordenámos as variáveis de entrada por relevância.

   Concluímos que as variáveis mais importantes para a primeira componente principal são:

   - pelvic incidence : 0.44394
   - lumbar lordosis angle : 0.38651
   - pelvic tilt : 0.35046
   - sacral slope : 0.24439
   - degree spondylolisthesis : 0.16278
   - pelvic radius : 0.08691

   As variáveis mais importantes para a segunda componente principal são:

- pelvic tilt : 0.37797
- pelvic radius : 0.32762
- sacral slope : 0.24994
- pelvic incidence : 0.05640
- lumbar lordosis angle : 0.04513
- degree spondylolisthesis : 0.00258

Por fim, ordenámos as variáveis por relevância para as duas componentes principais:

- pelvic tilt : 0.51544
- pelvic incidence : 0.44751
- lumbar lordosis angle : 0.38913
- sacral slope : 0.34957
- pelvic radius : 0.33895
- degree spondylolisthesis : 0.16280

Entender a importância de cada variável nas componentes principais é crucial para interpretar e utilizar de forma eficiente a técnica do PCA e ajuda a melhorar a qualidade das análises.

Código Utilizado:

```
pca = PCA(n_components=2)
pca.fit(features_scaled)
X_pca = pca.transform(features_scaled)

print("Components (eigenvectors):\n",pca.components_)
print("Explained variance (ratio) =",pca.explained_variance_ratio_)

xvector = pca.components_[0] * max(X_pca[:,0])
yvector = pca.components_[1] * max(X_pca[:,1])

columns = features.columns
features1_rel = {columns[i] : math.sqrt(xvector[i]**2) for i in
    range(len(columns))}
sorted_features1 = sorted(zip(features1_rel.values(), features1_rel
    .keys()),reverse=True)
print('Features sorted by relevance for the first component: \n')
for i in range(len(sorted_features1)):
    print(f'{sorted_features1[i][1]} : {sorted_features1[i][0]: .5f
    }')

features2_rel = {columns[i] : math.sqrt(yvector[i]**2) for i in
    range(len(columns))}
sorted_features2 = sorted(zip(features2_rel.values(), features2_rel
    .keys()),reverse=True)
print('\nFeatures sorted by relevance for the second component: \n'
    )
for i in range(len(sorted_features2)):
    print(f'{sorted_features2[i][1]} : {sorted_features2[i][0]: .5f
    }')
```

```
23
24 features_rel = {columns[i] : math.sqrt(xvector[i]**2 + yvector[i
       ]**2) for i in range(len(columns))}
25 sorted_features = sorted(zip(features_rel.values(), features_rel.
       keys()),reverse=True)
26 print('\nFeatures sorted by relevance: \n')
27 for i in range(len(sorted_features)):
28     print(f'{sorted_features[i][1]} : {sorted_features[i][0]: .5f}'
       )
```

3. Visualizar os dados com ground diagnosis e os clusters obtidos para k = 3 projetando os dados normalizados num espaço bidimensional PCA.

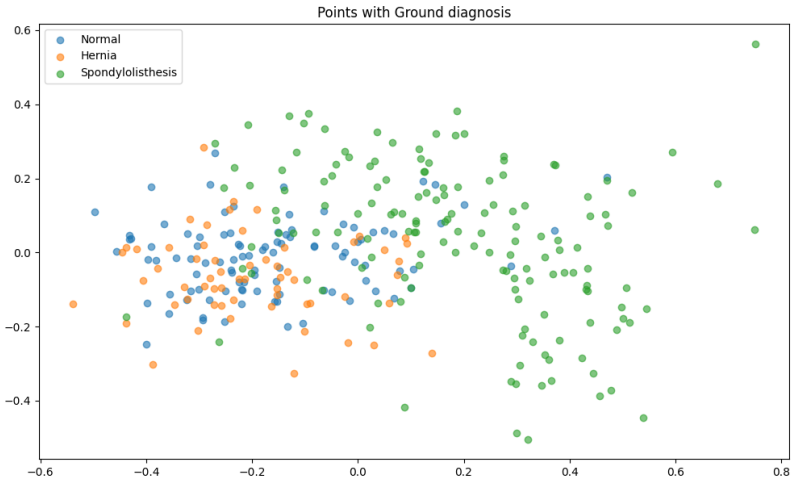


Figura 3: Pontos com Ground Diagnosis

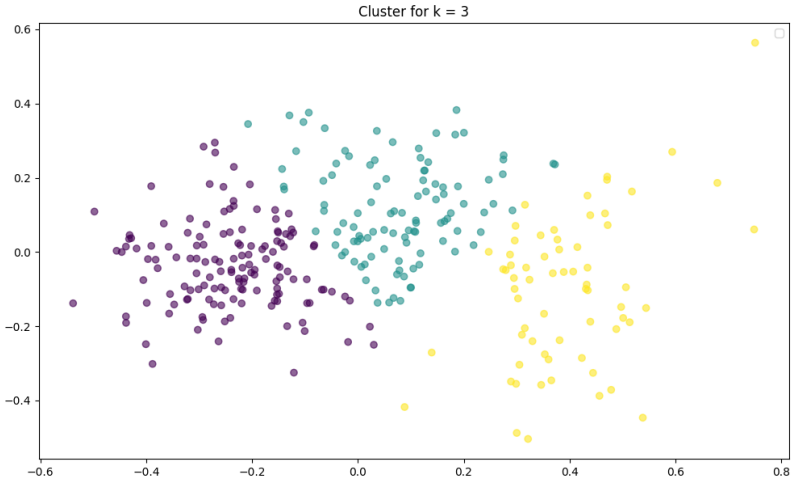


Figura 4: Clusters para k = 3

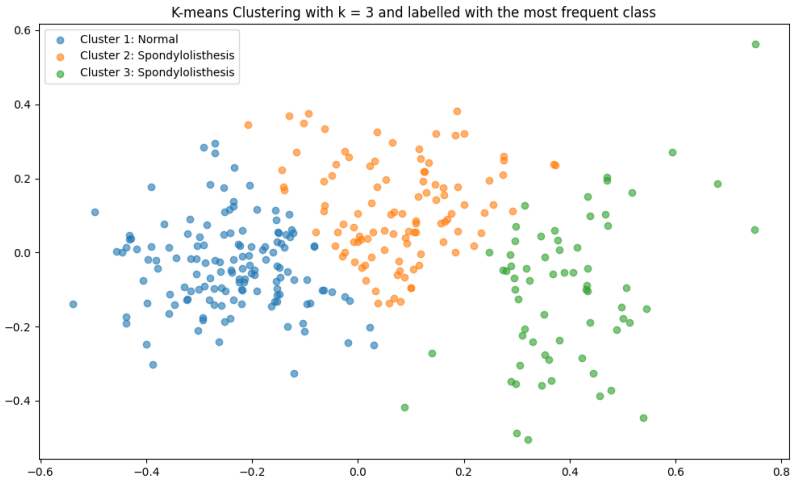


Figura 5: Clusters para k = 3 com a classe mais frequente

No segundo gráfico, visualizamos os pontos separados em 3 clusters. Tendo em conta o primeiro e segundo gráficos, é possível observar que o algoritmo de clustering não conseguiu separar os pontos de forma a que cada cluster correspondesse a uma classe, notando que o cluster mais heterogéneo é o roxo (segundo gráfico).
No ultimo gráfico desta secção, visualizamos os pontos separados em 3 clusters e a classe mais frequente em cada um.

Código Utilizado:

```
# i)
plt.figure(figsize=(12,7))
plt.scatter(X_pca[target=='Normal', 0], X_pca[target=='Normal', 1],
    alpha=0.6, label='Normal')
plt.scatter(X_pca[target=='Hernia', 0], X_pca[target=='Hernia', 1],
    alpha=0.6, label='Hernia')
plt.scatter(X_pca[target=='Spondylolisthesis', 0], X_pca[target=='
   Spondylolisthesis', 1], alpha=0.6, label='Spondylolisthesis')

plt.legend()
plt.title('Points with Ground diagnosis')
plt.savefig('ex3_ground_diagnosis.png')
plt.show()

# ii)

kmeans_algo = cluster.KMeans(n_clusters=3, random_state=0)
kmeans_model = kmeans_algo.fit(features_scaled)
target_pred = kmeans_model.labels_

plt.figure(figsize=(12, 7))
plt.scatter(X_pca[:,0], X_pca[:,1], c=target_pred, alpha=0.6)

plt.legend()
plt.title('Cluster for k = 3')
plt.savefig('ex3_cluster.png')
plt.show()
```

```
25
26 # iii)
27
28 cluster_mapping = pd.DataFrame({'Cluster': target_pred, 'Class':
       target})
29
30 # Calculate the mode class for each cluster
31 cluster_mode = cluster_mapping.groupby('Cluster')['Class'].agg(
       lambda x: x.mode().iat[0])
32
33 plt.figure(figsize=(12, 7))
34 for cluster in set(target_pred):
35     data = X_pca[target_pred == cluster]
36     plt.scatter(data[:, 0], data[:, 1], label=f'Cluster {cluster}',
       alpha=0.6)
37
38 plt.title('K-means Clustering with k = 3 and labelled with the most
       frequent class')
39 plt.savefig('ex3_cluster_labelled.png')
40 # Create a legend using the calculated mode class for each cluster
41 legend_labels = [f'Cluster {cluster+1}: {mode_class}' for cluster,
       mode_class in cluster_mode.items()]
42 plt.legend(legend_labels)
43
44 # Show the plot
45 plt.show()
```

4. Com base nos resultados obtidos nas questões anteriores, concluímos que é possível utilizar o clustering para realizar diagnósticos, identificando grupos de pacientes com características semelhantes, ou decidir qual o melhor tipo de tratamento consoante as diferentes características entre os indivíduos da população identificada com uma doença. Através desses subgrupos de indivíduos com características semelhantes é possível identificar outros tipos da doença.

   Também se usa esta técnica para criar perfis de risco com base nas características da população. Isso pode ser usado para implementar programas para combater os riscos da mesma.

   Analisar mais profundamente os dados de cada cluster poderá ajudar os profissionais a compreender melhor a natureza do problema, diferentes formas de como o diagnosticar e, posteriormente, tratar.

   Contudo, a utilização do clustering requer bastante cuidado visto que é preciso equilibrar os valores de silhueta e de pureza (consoante a informação que pretendemos retirar) e dado que os clusters nem sempre representam indivíduos exclusivamente da mesma classe. Neste caso, os plots anteriores revelaram a dificuldade do algoritmo em separar as populações das pessoas com classe Hérnia e Normal.