Laboratório 2

Análise Exploratória e Aprendizado Não-Supervisionado

CMC-13 Introdução à Ciência de Dados

Aluno: Vinícius José de Menezes Pereira

1. Preparação dos dados

Primeiramente, verificou-se que haviam dados faltantes. Como tais dados eram poucos em comparação com o tamanho do conjunto de dados fornecidos, estes foram descartados. Posteriormente foi feita a normalização dos dados seguindo a seguinte regra sugerida, para cada coluna que não fosse o ID:

$$v_{Novo} = \frac{v_{Atual} - \mu}{\sigma}$$

Onde μ é a média dos dados e σ o desvio padrão. De forma especial, foi criado um conjunto de dados descartando a coluna ID, que não era importante para o agrupamento. Início da tabela de dados normalizados:

	huml	humw	ulnal	ulnaw	feml	femw	tibl	tibw	tarl	tarw
0	0.294591	0.804328	0.047786	0.582155	0.250943	0.236899	-1.558467	0.407223	-0.026234	0.408337
1	0.445173	0.786834	0.192541	0.906727	0.514084	0.534117	0.411378	0.637815	0.094224	0.485888
2	0.279588	0.695861	0.001063	0.765012	0.314338	0.335971	0.282990	0.412027	-0.043013	0.180243
3	0.236618	0.461430	-0.058402	0.531869	0.161887	0.147733	0.120067	0.104571	-0.151855	0.212176
4	-0.038430	0.160520	-0.290655	0.056440	-0.144524	-0.248557	-0.220016	-0.106805	-0.319637	0.084444

Tabela 1: dados normalizados

2. Análise Exploratória

- Scatter-Plot
  Foi feita a matriz de Scatter-Plot, que nos revelou que, em geral, há uma relação de linearidade entre entre as features, excetuando o ID. Ou seja, quando a medida de um osso aumenta, as medidas dos outros ossos

também devem aumentar, o que faz sentido, pois na natureza é comum haver proporcionalidade nos ossos.

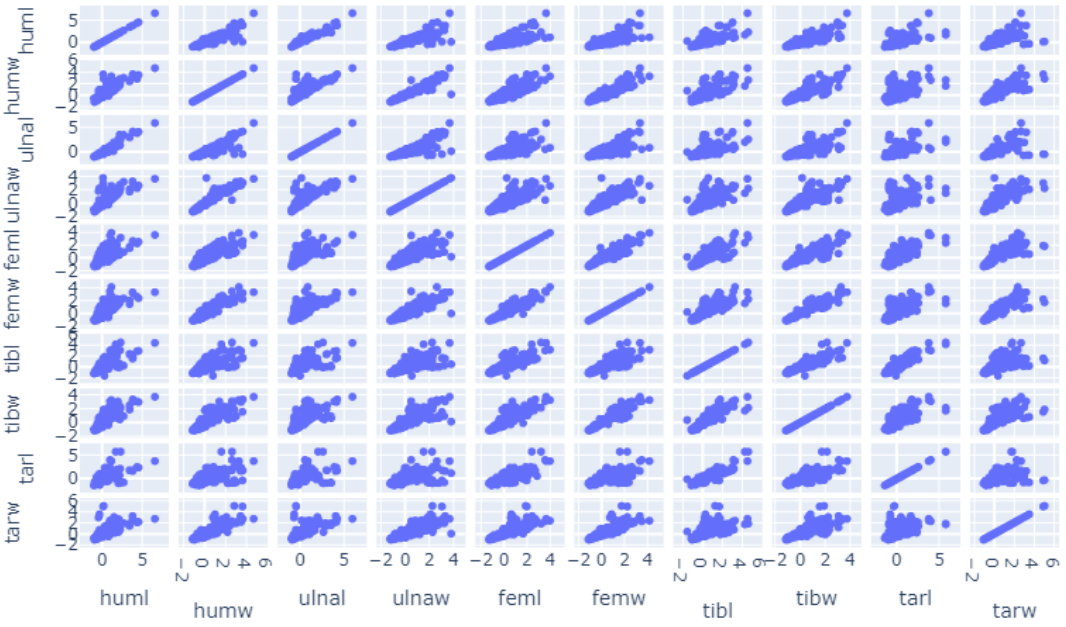

Figura 1: Scatter Plot

- BoxPlot
  O boxplot das features nos revelou que a maior parte das medidas realizadas normalizadas estão entre -1 e 1, o que era de se esperar, já que os dados teoricamente devem rodear a média. Apesar disso, vemos um espalhamento muito grande para os valores das medidas maiores que 0 em comparação com o valores menores que 0. Ou seja, o ‘range’ das medidas acima da média foi bem maior, possuindo até outliers, o que não foi observado para medidas abaixo da média. Além disso, vemos que os valores estão em geral abaixo da média, ao vermos a listra amarela. Tal comportamento é ilustrado no seguinte Boxplot:

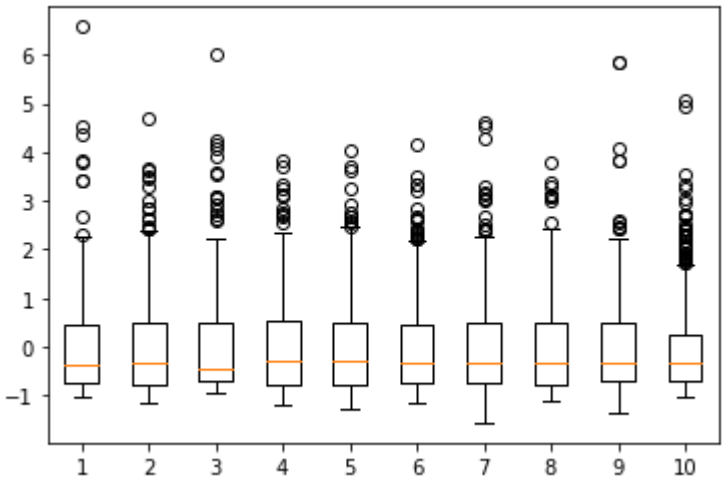

Figura 2: Boxplot da features

- Histogramas
  Ao selecionarmos as features humw e ulnal, vemos o seguinte histograma:

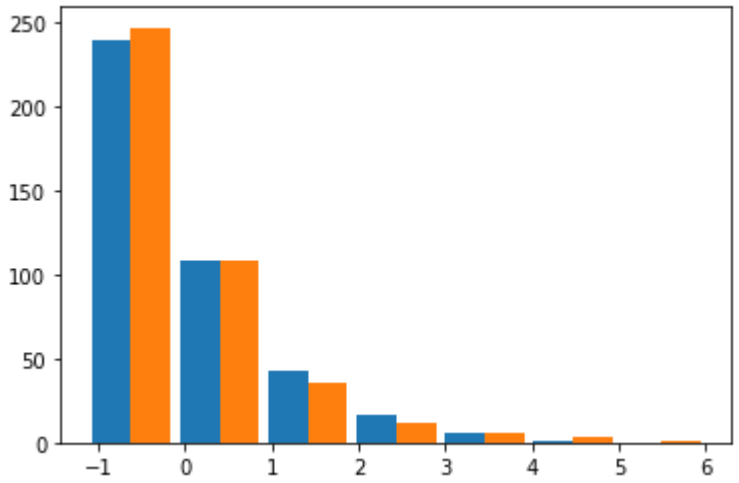

Figura 3: Histograma das features

Vemos que, conforme previsto no boxplot, a maior parte dos dados está abaixo da média, mas os dados acima da média(zero) estão mais espalhados.
Tal histograma não apresenta distribuição uniforme nem de gauss.

O próximo histograma é o do ID

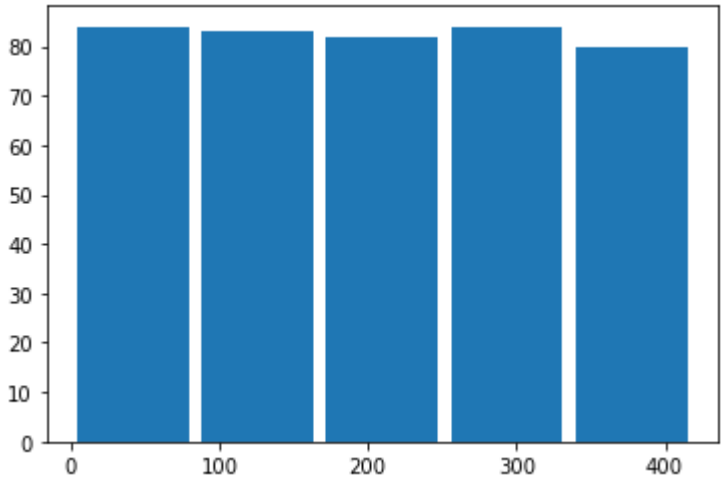

Figura 4: Histograma do ID

Como esperado, o histograma do ID é aproximadamente uniforme, já que para cada amostra só temos um único ID que cresce uniformemente. Pequena discrepância por ter sido escolhido dividir em grupos de 99.

3. Aprendizado não-supervisionado e Clusterização

No laboratório foi implementado o algoritmo Kmeans com dois parâmetros: número máximo de iterações e o próprio k. Tal método busca encontrar ‘centros de massa’ dos dados, que são divididos em k partes, pela escolha de grupos com menores distâncias aos centróides. A implementação em detalhes está no Jupyter Notebook, mas faremos aqui um simples descrição do algoritmo:

Algoritmo k-means

- def **init**(self, k=2, max_iters=100):

Inicializa parâmetros e cria os arrays de centróides e de clusters.

- def evaluate(self):

Utiliza a métrica do desvio padrão das distância dos pontos dos clusters aos centróides dividido pelo número de centróides para avaliar a clusterização.

- def predict(self,data):

Função principal que treina os dados. Utiliza as próximas funções como auxiliares e retorna os clusters e os centróides.

- def cluster_labels(self,clusters):

Rotula os dados conforme seu cluster final.

- def new_clusters(self,centroids):

A partir dos novos centróides, cria os novos clusters selecionando as amostras mais próximas do centróide

- def new_centroids(self,clusters):

Dados os novos clusters, determina as novas posições dos centróides pela média das features que estão em cada cluster.

- def closest_centroid(self,sample,centroids):

Acha o centróide mais próximo da amostra.

- def converged(self,old,current):

Chega a convergência, caso os novos centróides tenham mudado muito pouco de posição.

Depois da implementação do algoritmo, ele foi testado variando o parâmetro k de 1 até 9, o que nos deu o seguinte gráfico de desempenho:

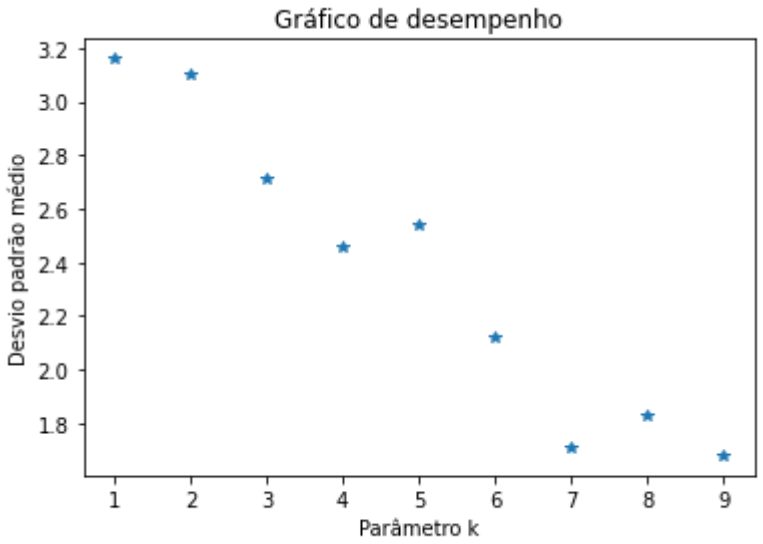

Figura 5: Gráfico de desempenho

Ao observarmos o gráfico, podemos inferir que o melhor número de clusters seria 3 ou 4, pois há neles ainda há uma boa diminuição do desvio padrão sem haver

overfitting. Para termos certeza de qual k iremos escolher, olhemos os seguintes gráficos:

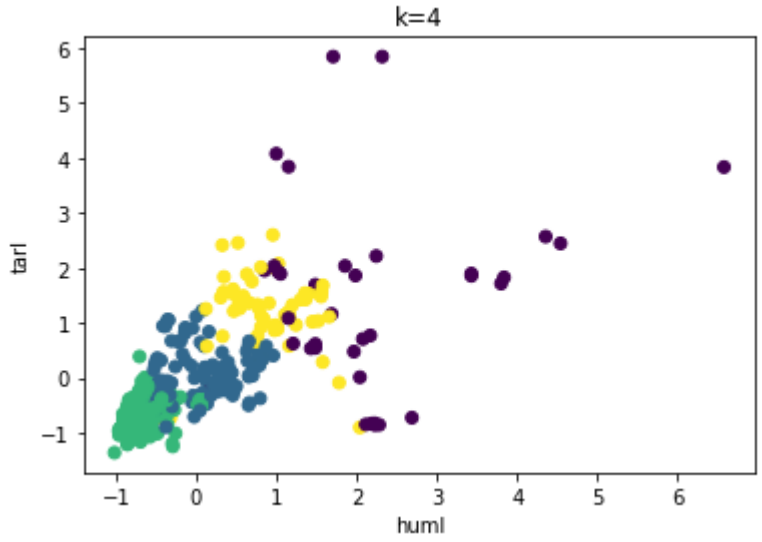

Figura 6: tarl x huml k = 4

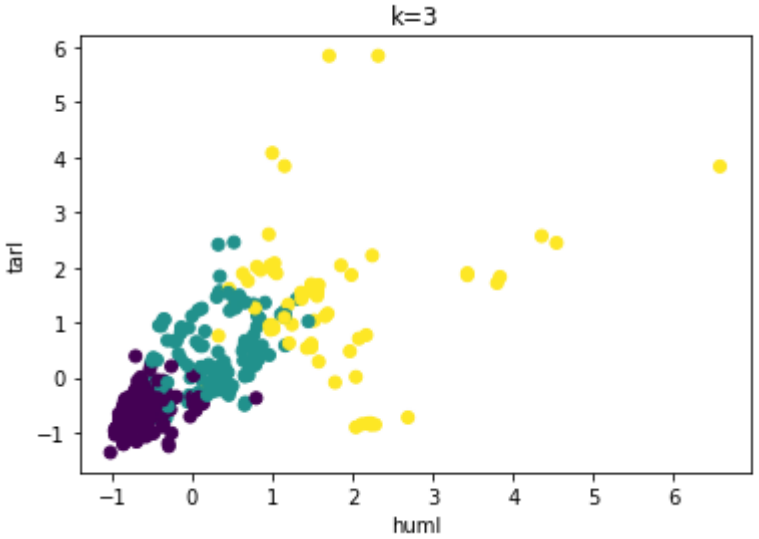

Figura 7: Mesma figura para k = 3.
Percebemos que a melhor escolha de k será 3, pois descreve bem o sistema, agrupando bem os dados e ligando os pontos dispersos a um cluster com relação com continuidade no gráfico. k= 4 acaba criando um cluster desnecessário com pontos dispersos, que sá são englobados nos pontos amarelo de k =3. A matriz de scatter plot obtida também mostra que k = e separa bem os dados, aos olharmos cada caso individualmente:

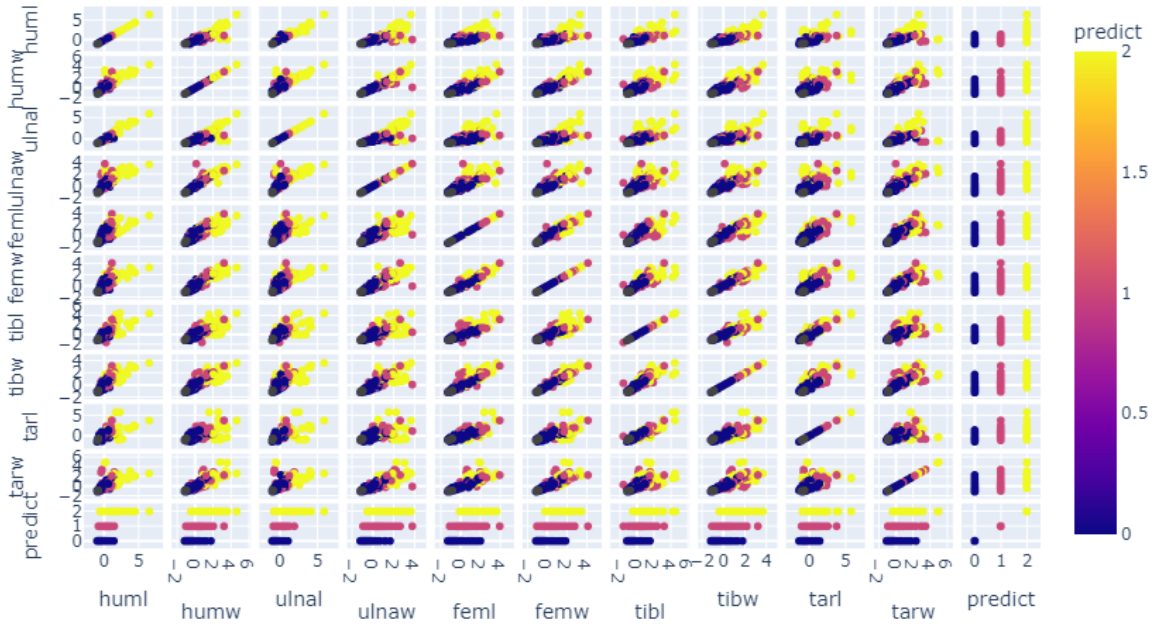

Figura 8: Matriz de Scatterplot para k = 3.

Conclusões

Percebe-se que o laboratório foi de grande proveito para os alunos, não sendo tão trabalho quanto o primeiro e ao mesmo tempo continuando a incentivar os alunos a quererem aprender mais sobre o tema.