

Lista #7

Curso: Ciência da Computação
Disciplina: Inteligência Artificial
Profa. Cristiane Neri Nobre

Pedro Henrique Lima Carvalho
Matrícula: 651230

1-

Kmeans é um algoritmo de agrupamento que busca separar as instâncias de uma base de dados em k grupos chamados clusters.

O algoritmo vincula cada instância a um cluster com base na sua distância ao centro de cada cluster(centroide). A instância pertencerá ao cluster cuja centroide estiver a menor distância, em um espaço n dimensional, sendo n o número de atributos (numéricos) de cada instância. Para o cálculo da distância podem ser usadas diversas fórmulas, como a distância euclidiana, de Manhattan ou Mahanalobis. Algoritmo:

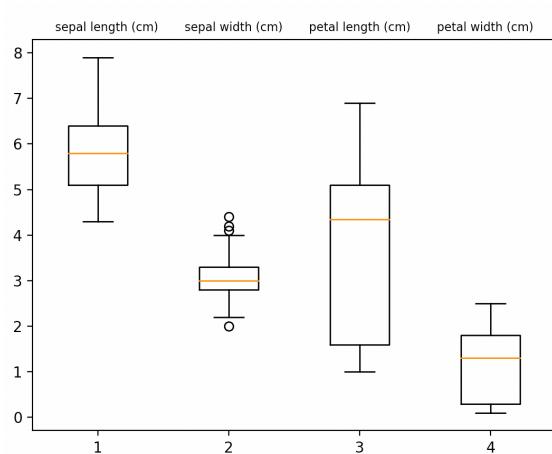
- 1- São escolhidos k pontos aleatórios, centroides iniciais, no espaço.
- 2- Cada instância será atribuída a um cluster com base na menor distância.
- 3- As centroides de cada cluster são recalculadas de forma a minimizar a distância entre a centroide e todas as instâncias do cluster.

Repete-se os passos 2 e 3 até uma cláusula de parada.

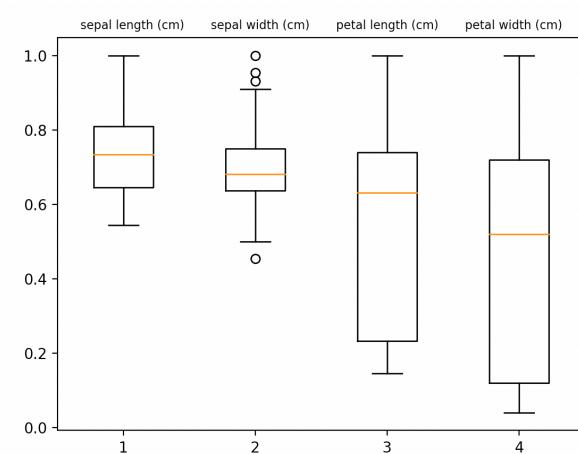
A cláusula de parada pode ser a ausência de troca de clusters por qualquer instância, alcance de um percentual máximo de trocas, número de iterações etc.

2- Para análise dos outliers foi realizado boxplots para cada atributo. Neles se observa que o atributo “Sepal Width” é o que apresenta instâncias como outliers, o que é confirmado quando as instâncias são plotadas.

Original:



Normalizado:



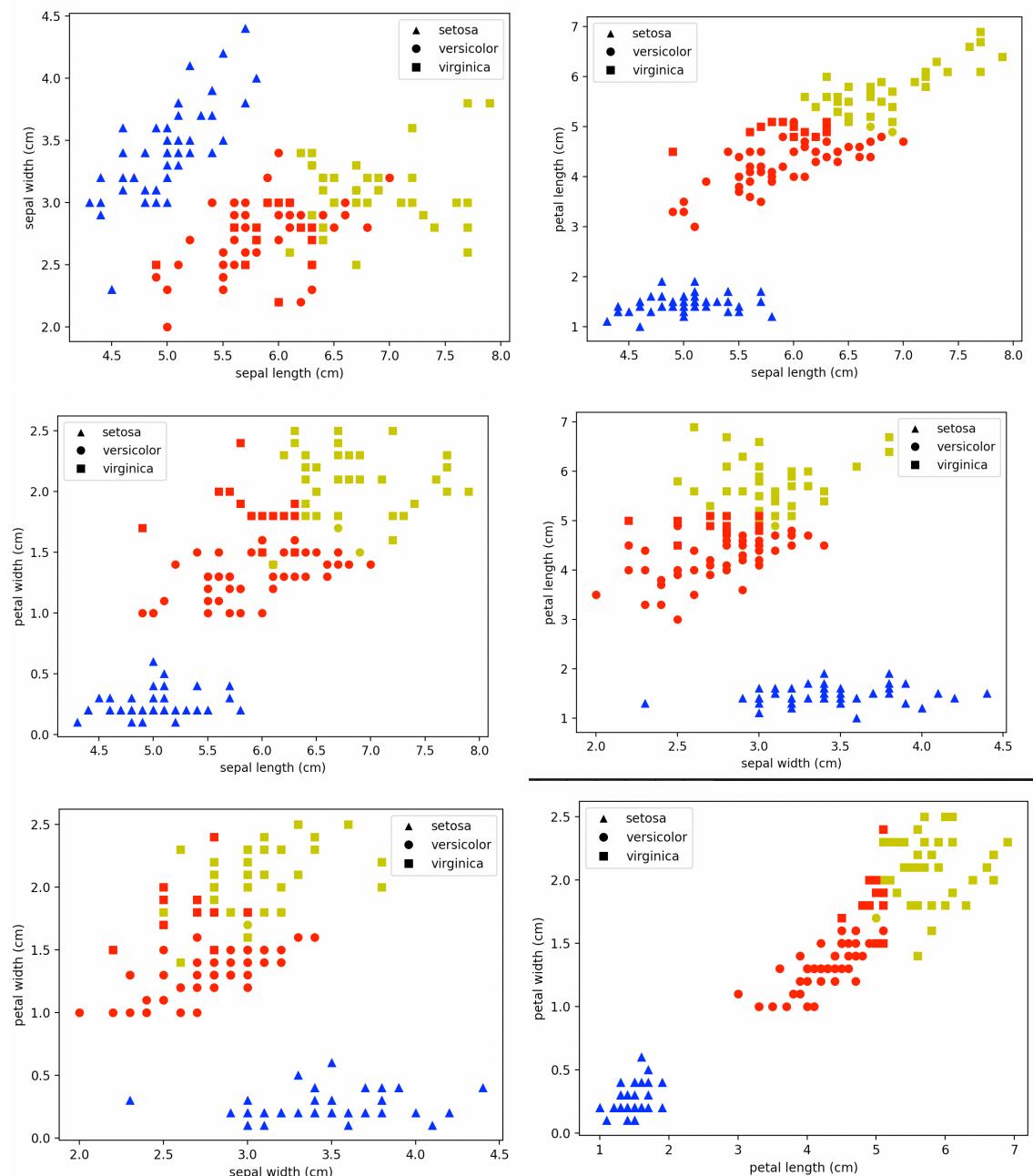
3- Há uma diferença de amplitude entre os atributos, especialmente entre sepal e pedal length com petal width. Isso faz com que os dois primeiros atributos tenham mais peso. O algoritmo se beneficiaria de uma normalização dos dados.

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
Mínimo		4.3	2.0	1.0
Máximo		7.9	4.4	6.9

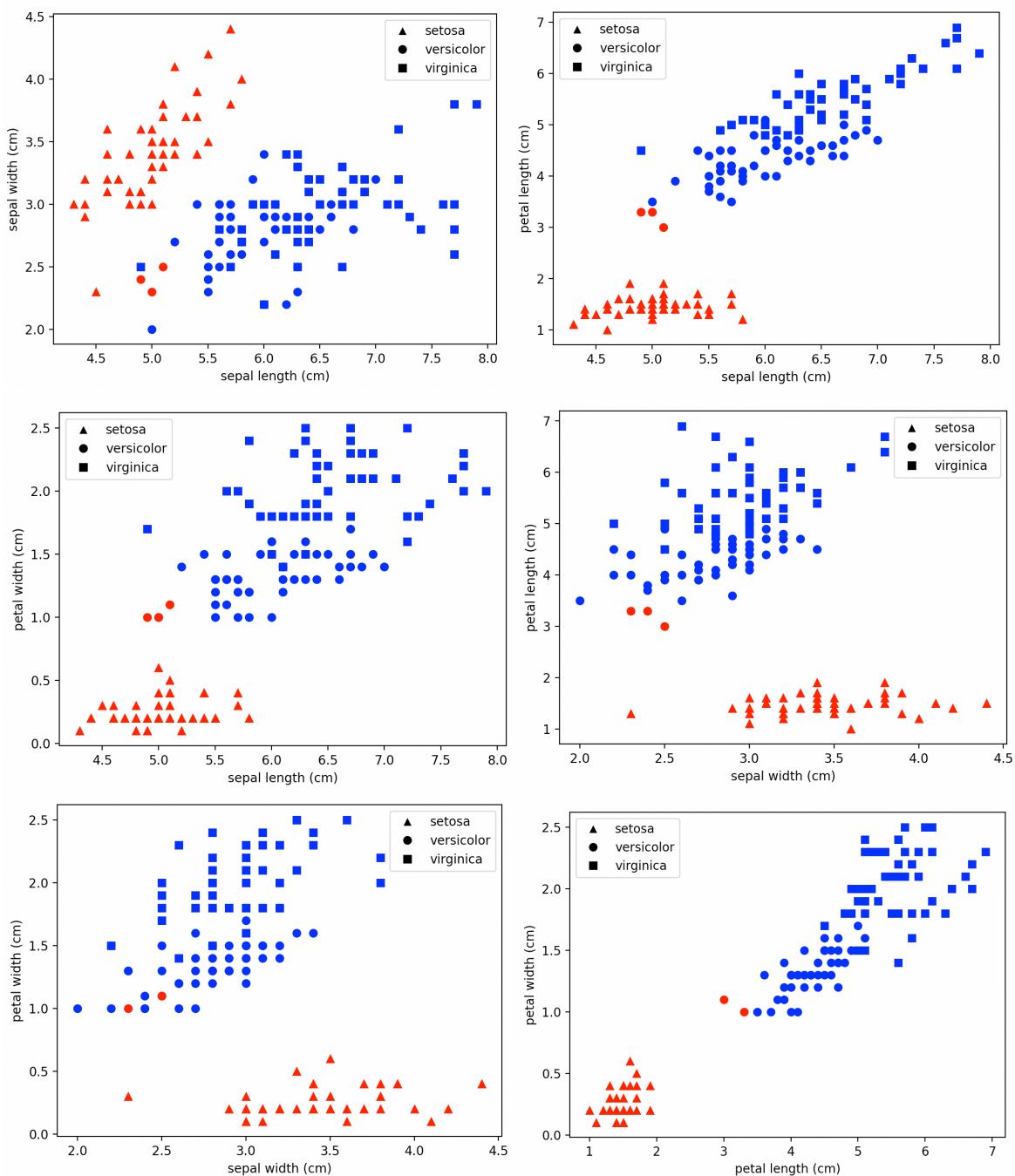
4- Repositório: https://github.com/phlc/ia_lista_7

5 e 6- Agrupamentos (Erradas = Forma diferente da maioria de mesma cor)

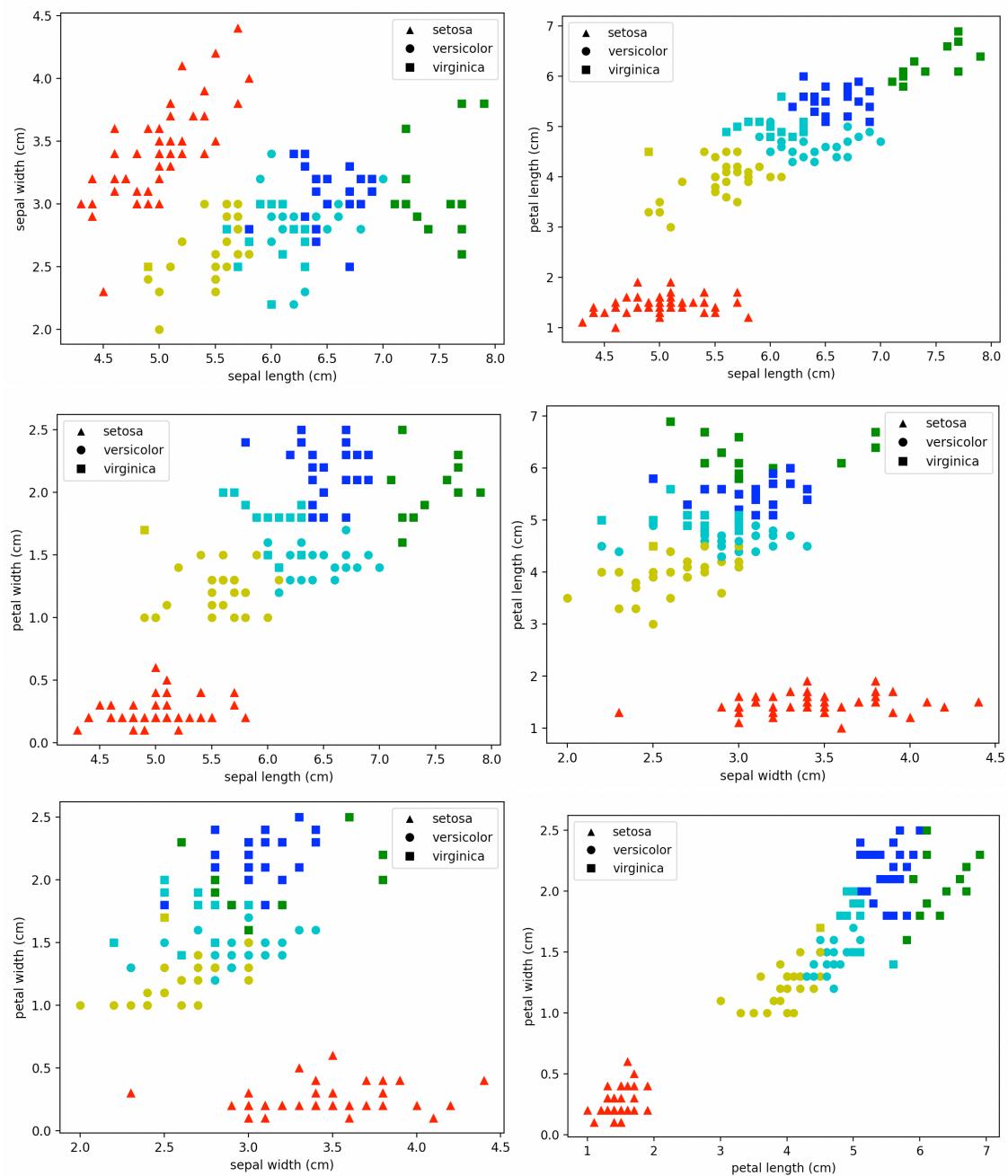
K = 3 (Número Correto de Tipos de Instância)



$K = 2$



$K = 5$



Metricas:

K = 3 (Número Correto de Tipos de Instância)

Davies Bouldin Próprio	Davies Bouldin Metrics	Silhouette Próprio	Silhouette Metrics
0.6619715465007469	0.6619715465007455	0.5528190123564098	0.5528190123564095

K = 3 Normalizado

Davies Bouldin Próprio	Davies Bouldin Metrics	Silhouette Próprio	Silhouette Metrics
0.6277126489524888	0.6277126489524852	0.5772435908622394	0.5772435908622392

K = 2

Davies Bouldin Próprio	Davies Bouldin Metrics	Silhouette Próprio	Silhouette Metrics
0.4042928371730428	0.4042928371730435	0.6810461692117464	0.6810461692117462

K = 2 Normalizado

Davies Bouldin Próprio	Davies Bouldin Metrics	Silhouette Próprio	Silhouette Metrics
0.39094365827437355	0.39094365827437416	0.6836700722159255	0.6836700722159255

K = 4

Davies Bouldin Próprio	Davies Bouldin Metrics	Silhouette Próprio	Silhouette Metrics
0.7803069838811082	0.7803069838811139	0.4980505049972879	0.49805050499728726

K = 4 Normalizado

Davies Bouldin Próprio	Davies Bouldin Metrics	Silhouette Próprio	Silhouette Metrics
0.7657926211502044	0.7657926211502005	0.49851866385678295	0.4985186638567823

K = 5

Davies Bouldin Próprio	Davies Bouldin Metrics	Silhouette Próprio	Silhouette Metrics
0.8193844948350423	0.8193844948350408	0.4930804067193526	0.49308040671935205

K = 5 Normalizado

Davies Bouldin Próprio	Davies Bouldin Metrics	Silhouette Próprio	Silhouette Metrics
0.8695597063905698	0.8695597063905668	0.4625280193129828	0.46252801931298165

Código das Métricas:

Davies Bouldin:

```
# Função para calcular a Distância entre dois pontos no espaço 4D
def distancia(x, y):
    return ((x[0]-y[0])**2 + (x[1]-y[1])**2 + (x[2]-y[2])**2 + (x[3]-y[3])**2)**(0.5)

# Função para calcular a distância média intra cluster
def distancia_intra(X, groups, centroids, cluster):
    number_instances = 0
    average = 0.0
    for i in range(len(X)):
        if(groups[i] == cluster):
            average += distancia(X[i], centroids[cluster])
            number_instances+=1
    return average/number_instances

# Função para calcular a metrica de Davies_bouldin
def davies_bouldin(X, groups, centroids):
    db = 0.0
    for i in range(len(centroids)):
        Di = 0.0
        for j in range(len(centroids)):
            if(i!=j):
                Rij = (distancia_intra(X, groups, centroids, i) + distancia_intra(X, groups, centroids, j))
                Rij = Rij / distancia(centroids[i], centroids[j])
                if(Di<Rij):
                    Di = Rij
        db += Di
    return db/len(centroids)
```

Silhouette:

```
# Função para calcular a metrica Silhouette coefficient
def silhouette_coefficient(X, groups, centroids):
    Sc = 0.0
    for i in range(len(X)):
        ai = 0.0
        bi = 0.0
        intra_cluster = 0
        distance_others = [0.0]*len(centroids)
        number_others = [0]*len(centroids)
        distance_others[groups[i]] = float("inf")
        number_others[groups[i]] = 1

        for j in range(len(X)):
            if(i!=j and groups[i] == groups[j]):
                ai += distancia(X[i], X[j])
                intra_cluster+=1
        ai = ai/intra_cluster

        for j in range(len(X)):
            if(groups[i] != groups[j]):
                distance_others[groups[j]] += distancia(X[i], X[j])
                number_others[groups[j]] += 1

        for j in range(len(distance_others)):
            distance_others[j] /= number_others[j]

        bi = min(distance_others)

        Sc += (bi - ai)/ max(ai, bi)

    return Sc/len(X)
```