Disciplina: Inteligência Artificial

Professora: Cristiane Neri Nobre

Data de entrega: 10/11

Valor: 2 pontos

Para fazer as questões abaixo, sugiro que estude o material sobre **Agrupamento** que está no CANVAS. Assista também os vídeos disponibilizados sobre este assunto. Está junto com os slides.

Além disso, acesso o notebook "**Kmeans.ipynb**", disponibilizado no CANVAS.

Questão 01

Rode o algoritmo Kmeans na base de dados a seguir da Iris, que está disponível no CANVAS.

1. Encontre os agrupamentos, discuta a qualidade destes agrupamentos (usando Silhouette e Elbow) e caracterize os agrupamentos obtidos
2. Explique como se obtém estas duas métricas, ou seja, explique as equações matemáticas.
3. Investigue, explique e implemente, pelo menos, mais 1 métrica de avaliação dos agrupamentos, diferentes das 2 anteriores
4. Utilizando mais dois algoritmos de agrupamento, por exemplo o DBSCAN e o SOM, verifique se estes métodos encontraram a mesma quantidade de grupos que o Kmeans. Faça uma análise dos grupos encontrados pelos 3 algoritmos
5. Uma vez que a base é classificada (setosa, virgínica e versicolor), mostre **visualmente** que instâncias foram agrupadas incorretamente pelo kmeans. Discuta os resultados.
6. Faça um pequeno relatório explicando todas as etapas de pré-processamento realizadas e explicando todos os resultados obtidos.

Coloque os links para os códigos produzidos ao final de cada questão

Lista 6 - Inteligência Artificial

Henrique Oliveira da Cunha Franco

Questão 1

Código do Deepnote usado para elaboração das questões: LinkCodigo
Código em Latex desse arquivo no Overleaf: LinkLatex

Silhouette

- $k = 2$: Score de 0.629, indicando que 2 clusters representam bem os dados.
- $k = 3$ ... $k = 8$: Scores diminuem para 0.3 a 0.5, sugerindo piora na qualidade.

Elbow

- $k = 2$: Alto valor de WCSS (12.14).
- $k = 3$ e $k = 4$: WCSS diminui para 6.99 e 5.53.
- $k = 4$ ... : redução se torna menos observável

Questão 2

Métrica Elbow

A métrica Elbow é usada para determinar o número ideal de clusters (k) em um conjunto de dados ao observar a variabilidade dentro dos clusters. A ideia é encontrar o ponto onde adicionar mais clusters não reduz significativamente a soma das distâncias quadradas dentro dos clusters. A equação para calcular a variância intra-cluster, chamada de Soma das Distâncias Quadradas (Within-Cluster Sum of Squares - $WCSS$), é dada por:

$$WCSS = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2 \tag{1}$$

onde:

- k é o número de clusters,
- C_i é o i-ésimo cluster,
- x representa cada ponto no cluster C_i,
- μ_i é o centróide do cluster C_i,
- $\|x - \mu_i\|^2$ é a distância quadrada entre o ponto x e o centróide μ_i.

A métrica Elbow sugere escolher o valor de k onde ocorre uma "queda" significativa na redução de $WCSS$, formando o chamado "cotovelo" (elbow) no gráfico de $WCSS$ versus k.

Métrica Silhouette

A métrica Silhouette mede a qualidade do agrupamento, avaliando a separação entre os clusters. Para cada ponto i em um cluster C, a Silhouette $s(i)$ é definida por:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{2}$$

onde:

- $a(i)$ é a distância média entre i e todos os outros pontos no mesmo cluster,
- $b(i)$ é a menor distância média de i aos pontos em qualquer outro cluster (ou seja, a distância média ao cluster mais próximo).

A Silhouette média para todo o conjunto de dados indica o quão bem definidos estão os clusters, com valores variando de -1 a 1:

- $s(i) \approx 1$: o ponto i está bem dentro do seu próprio cluster.
- $s(i) \approx 0$: o ponto i está próximo da fronteira de outro cluster.
- $s(i) \approx -1$: o ponto i pode estar no cluster errado.

Questão 3

Métrica de Avaliação: Coeficiente de Dunn

O Coeficiente de Dunn é uma métrica usada para avaliar a qualidade dos agrupamentos em um conjunto de dados. Essa métrica considera tanto a separação entre clusters quanto a compacidade dentro de cada cluster, sendo uma medida que favorece agrupamentos onde os clusters são bem separados e compactos.

O Coeficiente de Dunn é definido pela razão entre a distância mínima entre clusters e o diâmetro máximo dos clusters, conforme a equação a seguir:

$$D = \frac{\min_{1 \leq i < j \leq k} d(C_i, C_j)}{\max_{1 \leq l \leq k} \delta(C_l)} \tag{3}$$

onde:

- k é o número de clusters,
- $d(C_i, C_j)$ é a distância mínima entre os clusters C_i e C_j,
- $\delta(C_l)$ é o diâmetro do cluster C_l, definido como a distância máxima entre quaisquer dois pontos dentro do mesmo cluster.

Cálculo das Componentes da Métrica

1. **Distância entre Clusters** $d(C_i, C_j)$: Normalmente, calcula-se essa distância como a menor distância entre quaisquer dois pontos $x \in C_i$ e $y \in C_j$ dos clusters C_i e C_j:

$$d(C_i, C_j) = \min_{x \in C_i, y \in C_j} \|x - y\| \tag{4}$$

2. **Diâmetro do Cluster** $\delta(C_l)$: O diâmetro de um cluster C_l é a distância máxima entre quaisquer dois pontos x e y dentro do cluster:

$$\delta(C_l) = \max_{x, y \in C_l} \|x - y\| \tag{5}$$

Implementação do Coeficiente de Dunn

```python
# Carregar os dados
df = pd.read_csv('/work/lista6/Iris.csv')

# Remover a coluna de classe para realizar o clustering
data = df[['sepallength', 'sepalwidth', 'petallength', 'petalwidth']]

# Aplicar K-means com 3 clusters (como no dataset Iris original)
kmeans = KMeans(n_clusters=3, random_state=0)
labels = kmeans.fit_predict(data)

# Função para calcular o Coeficiente de Dunn
def dunn_index(data, labels):
    # Função auxiliar para calcular o diâmetro do cluster
    def cluster_diameter(cluster_points):
        if len(cluster_points) <= 1:
            return 0
        return np.max(pairwise_distances(cluster_points))

    # Função auxiliar para calcular a menor distância entre clusters
    def min_intercluster_distance(cluster_points1, cluster_points2):
        if len(cluster_points1) == 0 or len(cluster_points2) == 0:
            return np.inf
        return np.min(pairwise_distances(cluster_points1, cluster_points2))

    # Obter os clusters únicos
    unique_clusters = np.unique(labels)
    cluster_diameters = []
    intercluster_distances = []

    # Calcular o diâmetro de cada cluster
    for cluster in unique_clusters:
        cluster_points = data[labels == cluster]
        cluster_diameters.append(cluster_diameter(cluster_points))

    # Calcular as distâncias mínimas entre cada par de clusters
    for i, cluster1 in enumerate(unique_clusters):
        for j, cluster2 in enumerate(unique_clusters):
            if i < j:  # evitar repetir pares e auto comparação
                cluster_points1 = data[labels == cluster1]
                cluster_points2 = data[labels == cluster2]
                intercluster_distances.append(min_intercluster_distance(cluster_points1, cluster_points2))

    # Calcular o coeficiente de Dunn
    return np.min(intercluster_distances) / np.max(cluster_diameters)

# Calcular o Coeficiente de Dunn
dunn_score = dunn_index(data.values, labels)
print(f"Coeficiente de Dunn: {dunn_score}")
```

```
Coeficiente de Dunn: 0.09880739332807607
```

Figure 1: Implementação do Coeficiente de Dunn na base Iris.csv

Interpretação do Coeficiente de Dunn

Um valor alto para o Coeficiente de Dunn indica que os clusters são compactos e bem separados, sugerindo que o agrupamento é de alta qualidade. Em contrapartida, um valor baixo para D indica clusters mais dispersos e/ou sobrepostos.

Questão 4

```python
# Aplicar o algoritmo K-means com 3 clusters
kmeans = KMeans(n_clusters=3, random_state=0)
kmeans_labels = kmeans.fit_predict(data)

# Aplicar o algoritmo Agglomerative Clustering
agg_clustering = AgglomerativeClustering(n_clusters=3)
agg_labels = agg_clustering.fit_predict(data)

# Aplicar o algoritmo Mean Shift
mean_shift = MeanShift()
mean_shift_labels = mean_shift.fit_predict(data)

# Verificar o número de clusters encontrados por cada método
num_clusters_kmeans = len(np.unique(kmeans_labels))
num_clusters_agg = len(np.unique(agg_labels))
num_clusters_mean_shift = len(np.unique(mean_shift_labels))

print(f"K-means encontrou {num_clusters_kmeans} clusters.")
print(f"Agglomerative Clustering encontrou {num_clusters_agg} clusters.")
print(f"Mean Shift encontrou {num_clusters_mean_shift} clusters.")

# Análise comparativa dos clusters encontrados
def cluster_analysis(data, labels, method_name):
    num_clusters = len(np.unique(labels))
    print(f"\nAnálise para {method_name}:")
    if num_clusters > 1:
        score = silhouette_score(data, labels)
        print(f"Coeficiente de Silhueta: {score:.3f}")
    else:
        print("Número de clusters insuficiente para calcular o Coeficiente de Silhueta.")

cluster_analysis(data, kmeans_labels, "K-means")
cluster_analysis(data, agg_labels, "Agglomerative Clustering")
cluster_analysis(data, mean_shift_labels, "Mean Shift")
```

```
K-means encontrou 3 clusters.
Agglomerative Clustering encontrou 3 clusters.
Mean Shift encontrou 2 clusters.

Análise para K-means:
Coeficiente de Silhueta: 0.553

Análise para Agglomerative Clustering:
Coeficiente de Silhueta: 0.554

Análise para Mean Shift:
Coeficiente de Silhueta: 0.685
```

Figure 2: Aplicação de outros algoritmos e comparando-os com KMeans

Foram aplicados três algoritmos de agrupamento: **K-means**, **Agglomerative Clustering** e **Mean Shift**. A análise considera tanto o número de clusters encontrados quanto a qualidade dos agrupamentos, avaliada pelo Coeficiente de Silhueta.

Análise dos Grupos Encontrados pelos Algoritmos

Número de Clusters Encontrados

- **K-means**: Encontrou 3 clusters, que corresponde ao número esperado, indicando boa identificação das classes.

- **Agglomerative Clustering**: Também encontrou 3 clusters, semelhante ao K-means.
- **Mean Shift**: Encontrou 2 clusters, indicando que o método não conseguiu identificar corretamente a terceira classe de íris.

Silhouette

- **K-means**: O Coeficiente de Silhueta foi 0.553, sugerindo agrupamentos bem separados e coesos.
- **Agglomerative Clustering**: Obteve Coeficiente de Silhueta de 0.554, ligeiramente superior ao do K-means, indicando qualidade semelhante.
- **Mean Shift**: Apresentou Coeficiente de Silhueta de 0.685, o maior entre os três, sugerindo maior qualidade dos agrupamentos.

Considerações Finais

- **K-means** e **Agglomerative Clustering** identificaram 3 clusters com boa qualidade, sendo mais adequados quando o número de clusters é conhecido.
- **Mean Shift**, apesar de ter obtido o melhor Coeficiente de Silhueta, encontrou apenas 2 clusters, o que pode ser um problema caso a precisão na identificação das classes seja importante.

Questão 5

```python
# Carregar o conjunto de dados Iris
df = base

# Extrair características e rótulos reais
X = df[['sepallength', 'sepalwidth', 'petallength', 'petalwidth']].values
y = df['class'].values

# Aplicar o K-means com 3 clusters
kmeans = KMeans(n_clusters=3, random_state=42)
y_kmeans = kmeans.fit_predict(X)

# Visualizar as instâncias mal agrupadas
# Vamos verificar onde as instâncias foram agrupadas incorretamente
incorrect_indices = []
for i in range(len(y)):
    if y[i] != y_kmeans[i]:  # Comparar a classe real com o cluster atribuído
        incorrect_indices.append(i)

# Plotar os dados
plt.figure(figsize=(10, 6))

# Plotar os pontos corretamente classificados
sns.scatterplot(x=df['sepallength'], y=df['sepalwidth'], hue=y, palette='Set2', s=100, edgecolor='black')

centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1], color='red', marker='D', s=200, label='Centroides')
```

```
<matplotlib.collections.PathCollection at 0x7f610c655550>
```

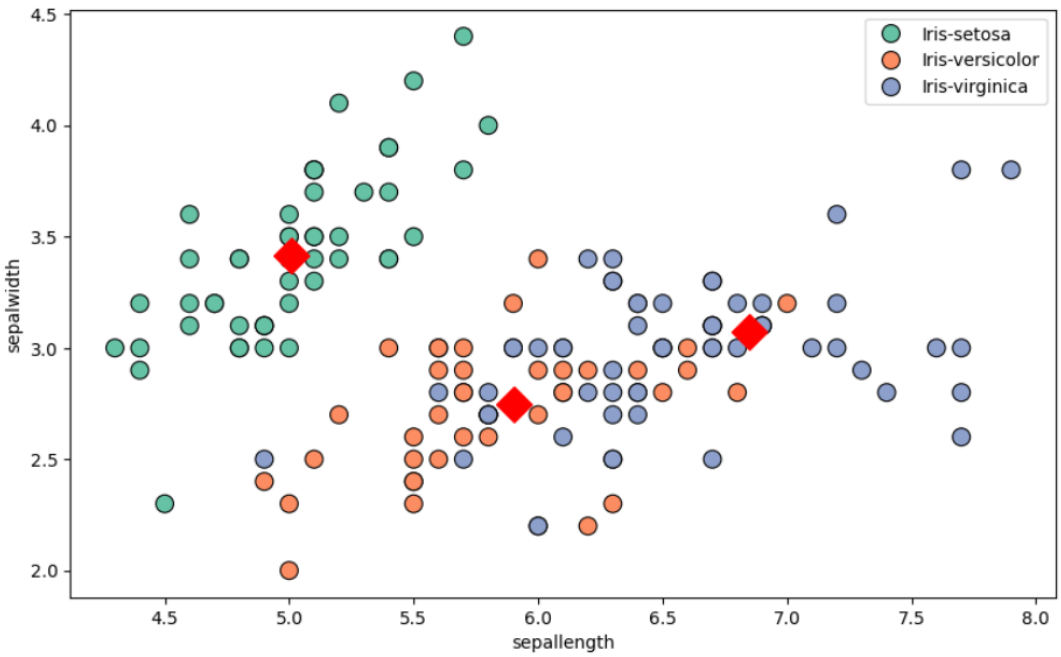

Figure 3: Scatterplot do resultado do KMeans, junto de seus centroides

Análise dos Resultados

Considerações sobre os Clusters Encontrados

O algoritmo KMeans agrupou as instâncias da base de dados em três clusters, com a seguinte análise para cada classe:

- **Iris-setosa:** Esta classe foi claramente separada das outras duas, indicando que o KMeans conseguiu identificar corretamente o agrupamento das instâncias da *Iris-setosa*. Essa separação é resultado da diferença bem definida nas características morfológicas dessa espécie, especialmente no que diz respeito ao comprimento e à largura das sépalas e pétalas, que se destacam em relação às demais espécies.

- **Iris-versicolor e Iris-virginica:** A separação entre essas duas classes foi mais difícil, resultando em uma sobreposição significativa. Ambas as espécies possuem características morfológicas semelhantes, particularmente no comprimento e largura das pétalas, o que torna a distinção entre elas mais difusa. Isso se reflete em um agrupamento menos claro, com os pontos de ambas as classes sendo agrupados de forma imprecisa.

Centroide dos Clusters

O KMeans calcula os centroides dos clusters como a média das características das instâncias em cada grupo. Esses pontos são representados no gráfico como **pontos vermelhos**, e sua análise fornece as seguintes observações:

- Para a classe *Iris-setosa*, os centroides estão bem posicionados, refletindo uma separação clara entre essa classe e as demais. A posição do centroide da *Iris-setosa* é bem definida, o que indica que as instâncias dessa classe são homogêneas e bem agrupadas.
- Para *Iris-versicolor* e *Iris-virginica*, os centroides dessas duas classes se sobrepõem ou estão muito próximos, refletindo a dificuldade do KMeans em distinguir entre essas classes. A sobreposição das características morfológicas dessas espécies leva a uma representação menos precisa dos seus centroides, o que compromete a identificação de centros ideais para cada classe.

Dificuldades de Separação entre as Classes

A análise dos resultados revela algumas dificuldades específicas no processo de separação entre os clusters:

- ***Overlap* entre a Iris-versicolor e Iris-virginica:** A dificuldade em separar essas duas classes está relacionada ao fato de que elas compartilham muitas características morfológicas semelhantes, o que gera uma sobreposição significativa no espaço das características. Isso torna a definição de uma fronteira clara entre as duas classes desafiadora para o KMeans, que tende a criar clusters esféricos, não sendo adequado para dados com sobreposição considerável.
- **Limitações do Algoritmo:** O KMeans tem como suposição fundamental a existência de clusters bem definidos e esféricos, com instâncias distribuídas de forma uniforme. Quando os dados não seguem essa distribuição, como no caso de *Iris-versicolor* e *Iris-virginica*, o algoritmo pode falhar em identificar corretamente a separação entre as classes. Nesse caso, o KMeans não conseguiu capturar a nuance nas fronteiras de separação entre essas duas espécies, o que levou a agrupamentos imprecisos.

Questão 6

Etapas de Pré-processamento

As etapas de pré-processamento executadas na base de dados foram as seguintes:

1. Leitura da Base de Dados

A base de dados foi lida utilizando a biblioteca `pandas` do Python. O arquivo CSV, `Iris.csv`, contém as colunas: `sepallength`, `sepalwidth`, `petallength`, `petalwidth` e `class`. A coluna `class` contém as classes das flores (*Iris-setosa*, *Iris-versicolor*, *Iris-virginica*).

```
base = pd.read_csv('/work/lista6/Iris.csv', sep=',',encoding='cp1252')
```

2. Verificação de Dados Faltantes

Após a leitura, foi verificado se existiam dados ausentes na base de dados. Não foram encontrados valores faltantes, indicando que os dados estavam completos para o processamento.

3. Escalonamento dos Dados

Os algoritmos de agrupamento, como o KMeans, são sensíveis à escala dos dados. Portanto, as características numéricas foram escalonadas utilizando o `StandardScaler` da biblioteca `scikit-learn` para garantir que todas as variáveis tivessem a mesma importância no processo de agrupamento.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data.drop('class', axis=1))
```

4. Codificação da Variável Categórica

Embora o foco no agrupamento não envolva diretamente o uso da variável `class`, ela foi codificada para facilitar a visualização e comparação dos resultados. A coluna de classes foi convertida para valores numéricos (0, 1, 2) utilizando a codificação ordinal, o que permitiu comparar os agrupamentos com as classes reais.

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
data['class_encoded'] = encoder.fit_transform(data['class'])
```

Resultados Obtidos

Os seguintes algoritmos de agrupamento foram aplicados aos dados:

1. KMeans

O KMeans encontrou 3 clusters, que coincidem com o número de classes reais da base de dados (*Iris-setosa*, *Iris-versicolor*, *Iris-virginica*). A análise da qualidade dos agrupamentos foi realizada através do Coeficiente de Silhueta, que foi calculado como 0.553, indicando uma boa separação entre os clusters.

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)
kmeans.fit(data_scaled)
kmeans_labels = kmeans.labels_
```

2. Agglomerative Clustering

O Agglomerative Clustering também encontrou 3 clusters, e o Coeficiente de Silhueta foi calculado como 0.554, mostrando um desempenho muito similar ao do KMeans. A principal diferença do Agglomerative Clustering é que ele não assume a forma esférica dos clusters, o que pode ser vantajoso em alguns cenários.

```
from sklearn.cluster import AgglomerativeClustering
agg_clust = AgglomerativeClustering(n_clusters=3)
agg_labels = agg_clust.fit_predict(data_scaled)
```

3. Mean Shift

O algoritmo Mean Shift encontrou 2 clusters, o que pode indicar que o algoritmo detectou uma estrutura mais simples nos dados. Isso pode ocorrer devido à sobreposição entre as classes *Iris-versicolor* e *Iris-virginica*, que são muito semelhantes. O Coeficiente de Silhueta foi de 0.685, o que sugere que os agrupamentos feitos pelo Mean Shift têm uma separação relativamente boa.

```
from sklearn.cluster import MeanShift
mean_shift = MeanShift()
mean_shift_labels = mean_shift.fit_predict(data_scaled)
```

4. Comparação entre os Algoritmos

A comparação entre os três algoritmos mostrou que:

- O KMeans e o Agglomerative Clustering encontraram 3 clusters, o que corresponde ao número de classes reais da base de dados.
- O Mean Shift encontrou 2 clusters, o que reflete a sobreposição significativa entre as classes *Iris-versicolor* e *Iris-virginica*.
- O Coeficiente de Silhueta foi mais alto para o Mean Shift (0.685), indicando uma separação melhor entre os grupos em comparação ao KMeans (0.553) e Agglomerative Clustering (0.554).

5. Análise de Clusters e Centróides

Ao observar os clusters e os centroides gerados pelo KMeans, podemos concluir que:

- A classe *Iris-setosa* foi bem separada, refletindo uma clara distinção em relação às outras classes.
- As classes *Iris-versicolor* e *Iris-virginica* apresentaram sobreposição, o que dificultou a separação pelo KMeans. Os centroides dessas classes ficaram próximos, refletindo a similaridade entre elas.

6. Dificuldades de Separação

A dificuldade em separar as classes *Iris-versicolor* e *Iris-virginica* foi uma limitação observada em todos os algoritmos. Isso se deve à semelhança das características morfológicas dessas duas espécies, o que gera uma sobreposição nas fronteiras de separação, especialmente no caso do KMeans, que assume a existência de clusters esféricos bem definidos.

Conclusão

Este estudo de agrupamento mostrou que, enquanto o KMeans e o Agglomerative Clustering conseguem identificar adequadamente a classe *Iris-setosa*, a separação entre *Iris-versicolor* e *Iris-virginica* ainda apresenta dificuldades devido à sobreposição entre essas duas classes. O Mean Shift apresentou um desempenho melhor, com um Coeficiente de Silhueta mais alto e menos sobreposição. A análise destacou a importância de escolher o algoritmo de agrupamento adequado, considerando as características do conjunto de dados.