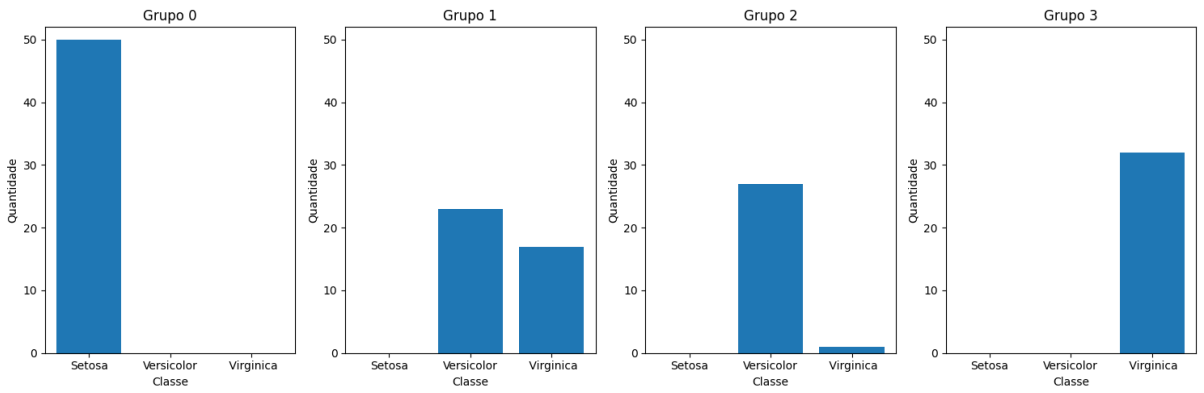
Aprendizagem de Máquina: Atividade 11 − Agrupamento

Carlos Emmanuel Pereira Alves
Curso de Bacharelado em Ciência da Computação
Universidade Federal do Agreste de Pernambuco (UFAPE)
Garanhuns, Brasil
carlos.emmanuel.236@gmail.com

1) Utilizando o algoritmo de agrupamento k-médias com todos os 150 exemplo da base Iris, https://archive.ics.uci.edu/ml/datasets/Iris:
    a) Para o valores de k = 4, calcule um gráfico de barras para cada grupo (quatro gráficos). Cada gráfico deverá ter três barras: quantidade de elementos de cada classe (setosa, versicolor e virginica). Isto é, cada um dos k gráficos deve ter: a quantidade de elementos da classe setosa no grupo, a quantidade de elementos da classe versicolor no grupo, a quantidade de elementos da classe virginica no grupo.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

cols = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']

data = pd.read_csv('iris.data', header=None, names=cols)

data_X = data[cols[:-1]]
data_y = data['class']

kmeans = KMeans(n_clusters=4)
kmeans.fit(data_X)
data['group'] = kmeans.labels_

#Calcula a quantidade de elementos de cada classe em cada grupo
group_class_count = data.groupby(['group', 'class']).size().unstack(fill_value=0)

#Cria os gráficos
fig, axs = plt.subplots(1, 4, figsize=(15,5))
for i in range(4):
    axs[i].bar(['Setosa', 'Versicolor', ' Virginica'], group_class_count.iloc[i])
    axs[i].set_title(f'Grupo {i}')
    axs[i].set_ylim([0, max(group_class_count.sum(axis=1))+2])
    axs[i].set_xlabel('Classe')
    axs[i].set_ylabel('Quantidade')

plt.tight_layout()
plt.show()
```



b) Mostre a média e o desvio padrão das distâncias de cada exemplo ao centróide mais próximo durante 1, 2, 3, . . . , 10 iterações. Monte uma tabela com os duas colunas (média e desvio padrão) e 10 linhas (número de iterações).

2) Utilizado a base Iris, https://archive.ics.uci.edu/ml/datasets/Iris, em um experimento do tipo Holdout 50/50 Estratificado realize seleção de protótipos da seguinte forma, para k = 9:
   a) Execute o k-medóides no conjunto de treino.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn_extra.cluster import KMedoids

cols = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']

data = pd.read_csv('iris.data', header=None, names=cols)

data_X = data[cols[:-1]]
data_y = data['class']

x_train, x_test, y_train, y_test = train_test_split(data_X, data_y, test_size=0.5, stratify=data_y)

kmedoids = KMedoids(n_clusters=9, max_iter=10, random_state=3)

predict = kmedoids.fit_predict(x_train)

print(predict)
```

```
[0 6 1 5 2 3 4 3 8 2 8 3 7 2 4 4 5 1 2 2 2 0 3 3 0 8 3 3 5 3 4 7 2 3 6 7 3
 0 3 8 7 1 0 3 3 3 8 3 7 8 4 3 2 3 2 3 3 8 3 5 4 8 8 8 3 3 5 4 7 7 3 3 8 2
 3]
```

   b) Remova do conjunto de treino todos os elementos que não são centroide de um grupo, isto é, mantém apenas os centróides no conjunto de treino. A redução do conjunto de treino é chamada seleção de protótipos. O conjunto de teste da mesma forma, isto é, os 50% dos dados original. Utilizando o conjunto de treino reduzido (composto apenas pelos centróides), calcule a taxa de acerto POR CLASSE para o conjunto de teste utilizando o classificador 1-NN com distância Euclidiana.

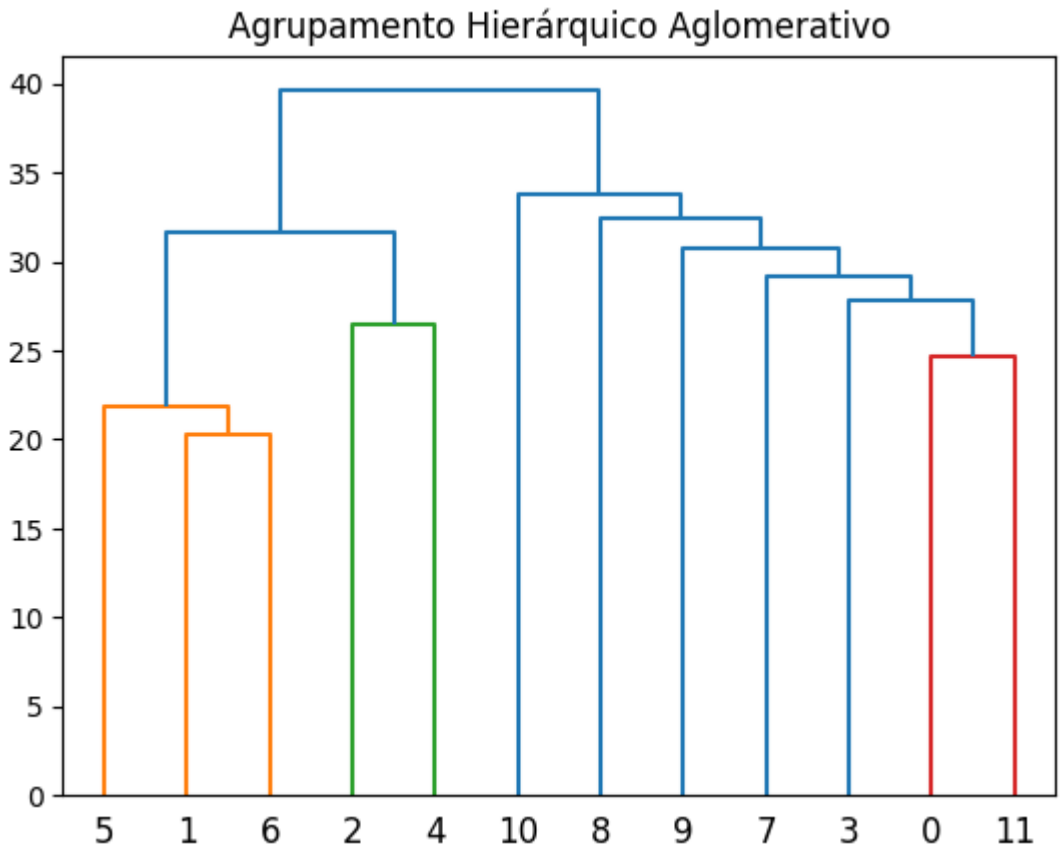
```python
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn_extra.cluster import KMedoids
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

cols = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']

data = pd.read_csv('iris.data', header=None, names=cols)

data_X = data[cols[:-1]]
data_y = data['class']

X_train, X_test, y_train, y_test = train_test_split(data_X, data_y, test_size=0.5, stratify=data_y, random_state=3)

kmedoids = KMedoids(n_clusters=9)
kmedoids.fit(X_train)

centroids_indices = kmedoids.medoid_indices_

centroids = X_train.iloc[centroids_indices]
centroids_labels = y_train.iloc[centroids_indices]

knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(centroids, centroids_labels)

predict = knn.predict(X_test)

accuracy_per_class = {}
for class_name in np.unique(y_train):
    true_mask = (y_test == class_name)
    predicted_mask = np.array(predict) == class_name
    accuracy = accuracy_score(true_mask, predicted_mask)
    accuracy_per_class[class_name] = accuracy

for class_name, accuracy in accuracy_per_class.items():
    print(f"Classe {class_name}: Taxa de Acerto = {accuracy:.2f}")
```

```
Classe Iris-setosa: Taxa de Acerto = 1.00
Classe Iris-versicolor: Taxa de Acerto = 0.93
Classe Iris-virginica: Taxa de Acerto = 0.93
```

3) O arquivo maisAssistidos.csv contém a avaliação dos usuários com notas de 1 a 5 da base Movie Lens 100k https://grouplens.org/datasets/movielens/ para os 12 filmes mais avaliados na base. São 943 usuários e cada filme foi avaliado por mais de 400 destes. Como cada atributo de um lme é a nota de um dos usuários existem vários valores de atributos omissos.
    a) Realize agrupamento hierárquico aglomerativo.

```python
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram
from sklearn.impute import SimpleImputer

def plot_dendrogram(model, **kwargs):
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack(
        [model.children_, model.distances_, counts]
    ).astype(float)

    dendrogram(linkage_matrix, **kwargs)

data = pd.read_csv('maisAssistidos.csv', index_col=0)

data = data.replace('?', np.nan)

imputer = SimpleImputer(strategy='mean', missing_values=np.nan)
imputer = imputer.fit(data)
data = imputer.transform(data)

model = AgglomerativeClustering(distance_threshold=0, n_clusters=None)

model = model.fit(data)
plt.title("Agrupamento Hierárquico Aglomerativo")
plot_dendrogram(model, truncate_mode="level", p=12)
plt.show()
```

b) Gere o dendrograma do agrupamento.



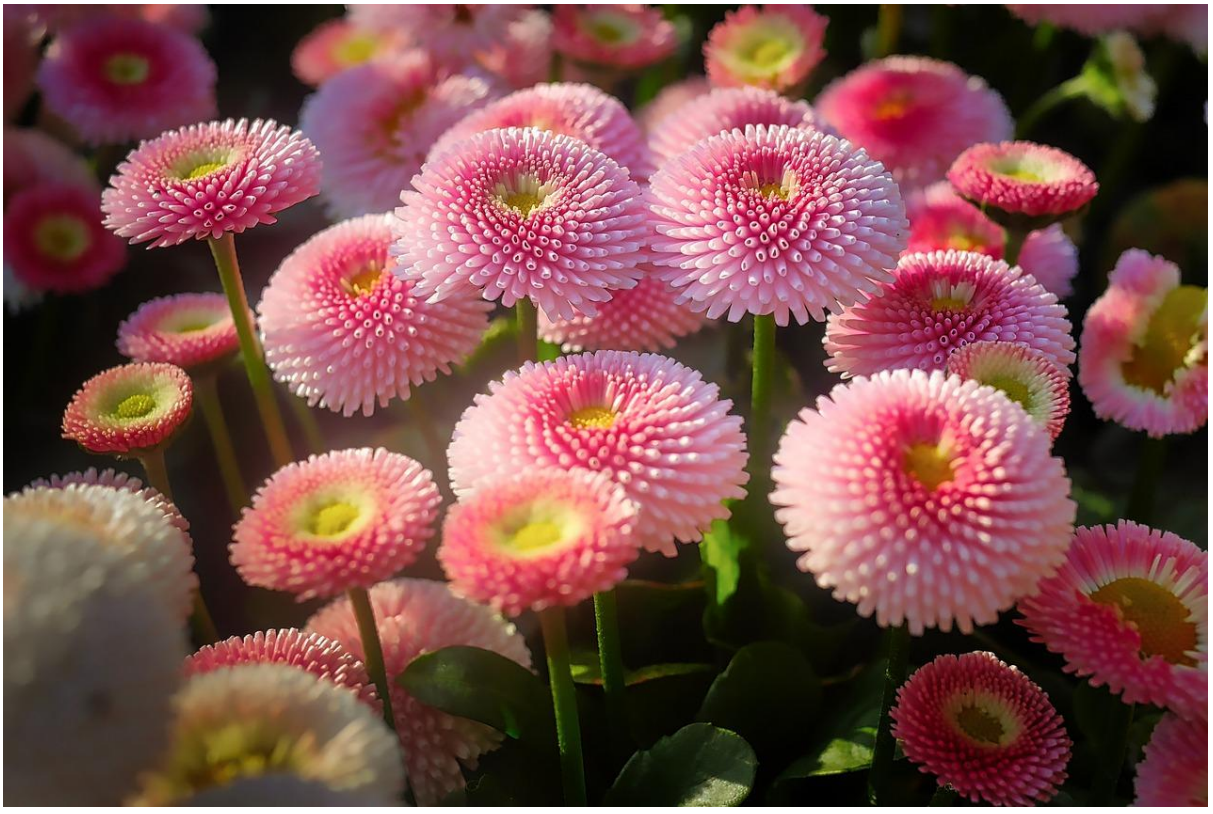
c) Pesquise qual o gênero de cada lme (comédia, policial, ficção científica etc.). Analise se o faz sentido, em relação ao gênero, cada vez que dois grupos são unidos no dendrograma.
Utilizando classificações de gênero, segundo o IMDB, eles ficaram assim:
0- Toy Story - Animação, Aventura, Comédia
1- Star Wars - Ação, Aventura, Fantasia
2- Fargo - Policial, Suspense
3- Independence Day: The ID4 Invasion - Documentário, Ficção Científica
4- The Godfather - Policial, Drama
5- Raiders of the Lost Ark - Ação, Aventura
6- Star Wars: Episode VI - Return of the Jedi - Ação, Aventura, Fantasia
7- Contact - Drama, Mistério, Ficção Científica
8- The English Patient - Drama, Romance, Guerra
9- Scream - Terror, Mistério
10- Liar Liar - Comédia, Fantasia

11- Air Force One - Ação, Drama, Suspense
No começo as uniões fazem até sentido com ele unindo os dois filmes de Star Wars e depois ligando a Raiders of the Lost Ark que tem o mesmo gênero deles. Depois ele liga Toy Story a Air Force One, que não faz sentido segundo o gênero. Após isso ele conecta os dois filmes de policial, Fargo e The Godfather, o que faz muito sentido. Mas depois disso as ligações começam a perder o sentido se formos olhar o gênero.
Ele conecta a ligação de Toy Story e Air Force One, ao documentário Independence Day: The ID4 Invasion, por exemplo, as outras conexões
feitas depois perdem o sentido quando olhamos o gênero.

4) Utilizando uma base construída a partir de uma imagem digital de uma fotografia da natureza (escolha uma imagem, exemplo abaixo), cada pixel da imagem é um elemento do conjunto de dados.
    a) Carregue cada pixel como um vetor de atributos. Cada pixel é representado pelo código RGB, ou seja, o primeiro atributo é o valor de intensidade R, o segundo é o valor de intensidade G e o último o valor de intensidade B.
    b) Execute o algoritmo de agrupamento k-médias para k = 8, 64 e 512. Considere limitar o número máximo de interações se a convergência demorar muito. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
    c) Ao final da convergência do k-médias arredonde os valores da posição de cada centróide para o inteiro mais próximo, exemplo, (15.3; 134.9; 9.4333) para (15; 135; 9). Reconstrua a imagem substituindo cada intensidade original de pixel pela intensidade do seu centróide, você obterá uma imagem na qual o número de cores distintas é k. Veja o exemplo das figuras abaixo.

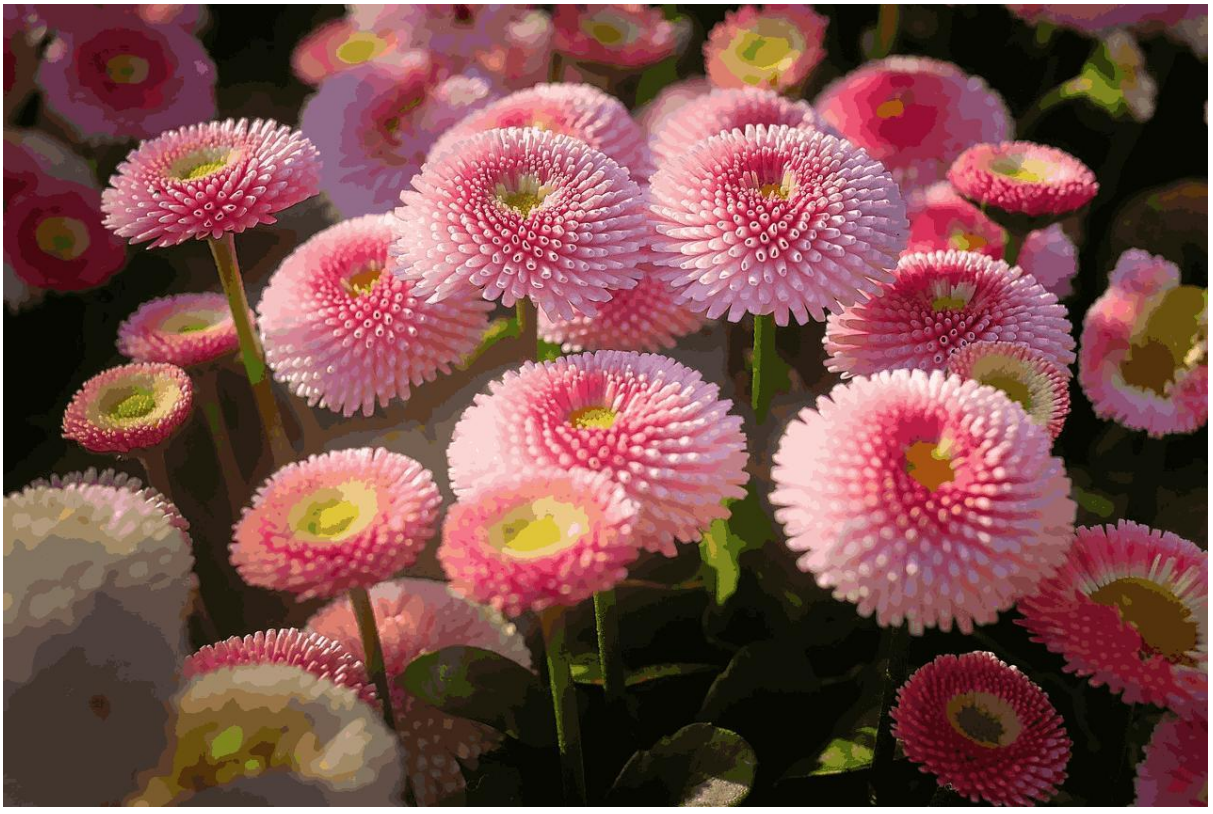
```python
from PIL import Image

image = Image.open('/content/flower.jpg')

image_array = np.array(image)

pixels = image_array.reshape(-1, 3)

num_clusters = [8, 64, 512]

for k in num_clusters:
    kmeans = KMeans(n_clusters=k, max_iter=100)
    kmeans.fit(pixels)

    cluster_centers = kmeans.cluster_centers_

    cluster_centers_rounded = np.round(cluster_centers).astype(int)

    labels = kmeans.labels_

    reconstructed_pixels = cluster_centers_rounded[labels]

    reconstructed_image_array = reconstructed_pixels.reshape(image_array.shape)
    reconstructed_image = Image.fromarray(reconstructed_image_array.astype(np.uint8))

    reconstructed_image.save(f'reconstructed_image_k_{k}.jpg')
```

Original



k = 8

k = 64



k = 512