Aprendizagem de Máquina
Prof. Tiago Buarque A. de Carvalho

Exercícios sobre Agrupamento

Alunos: David Brito e Laisy Cristina

1. Utilizado utilizando o algoritmo de agrupamento *k*-médias com todos os 150 exemplo da base Iris , https://archive.ics.uci.edu/ml/datasets/Iris:

   (a) (15 pontos) Para o valor de $k = 5$, calcule um grá co de barras para cada grupo (cinco grá cos). Cada grá co deverá ter três barras: quantidade de elementos de cada classe (setosa, versicolor e virginica). Isto é, cada um dos cinco grá cos de 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
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

iris = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data", header=Non
iris.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']

X = iris[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]

k = 5

kmeans = KMeans(n_clusters=k, random_state=42).fit(X)

iris['cluster'] = kmeans.predict(X)

groups = iris.groupby(['cluster', 'class']).size().reset_index(name='count')

for i in range(k):
    fig, ax = plt.subplots(figsize=(6, 4))
    group_data = groups[groups['cluster']==i]
    ax.bar(group_data['class'], group_data['count'])
    ax.set_title(f'Agrupamento {i}')
    ax.set_xlabel('Classe')
    ax.set_ylabel('Quantidade')
    plt.show()
```

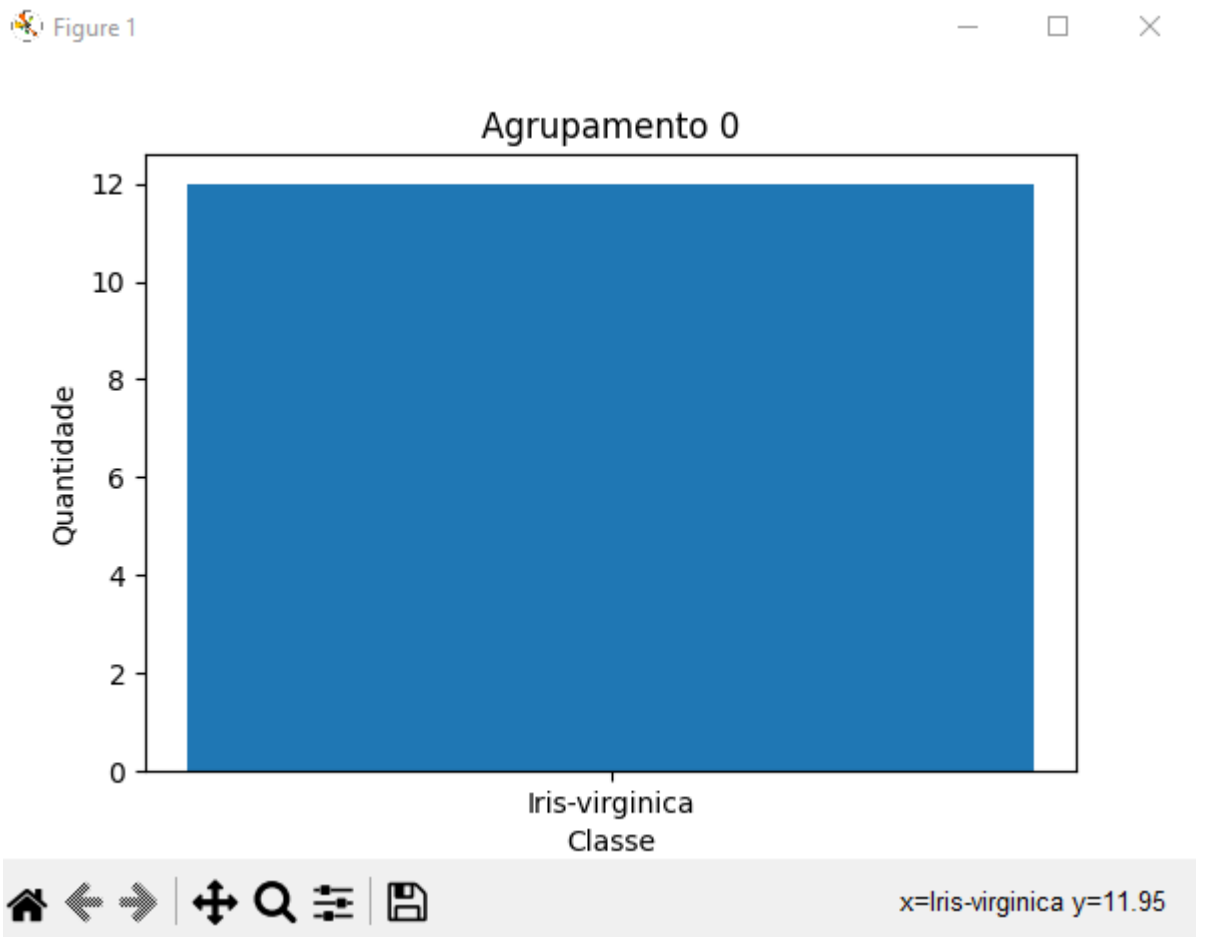


Figure 1
Agrupamento 0
12
10
8
6
4
2
0
Quantidade
Iris-virginica
Classe
x=Iris-virginica y=11.95

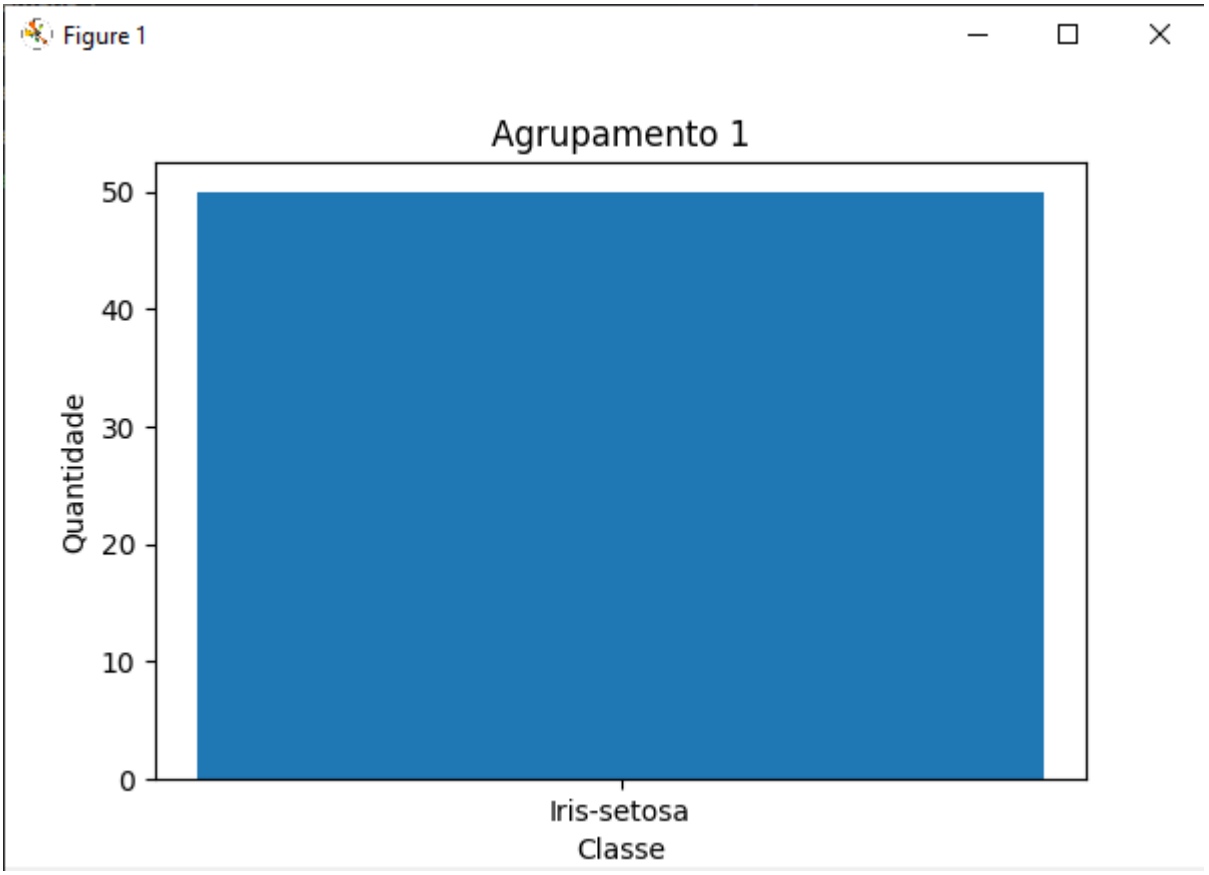


Figure 1
Agrupamento 1
50
40
30
20
10
0
Quantidade
Iris-setosa
Classe

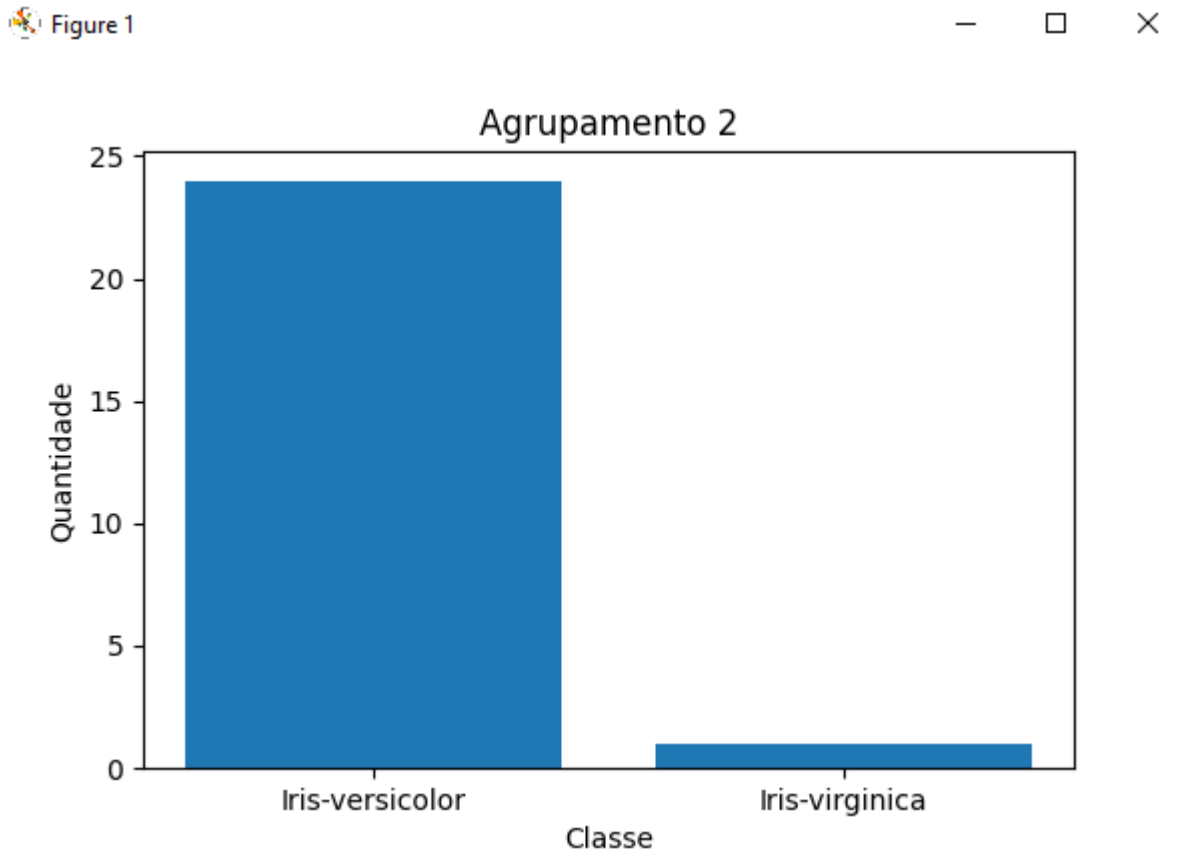


Figure 1
Agrupamento 2
25
20
15
10
5
0
Quantidade
Iris-versicolor
Iris-virginica
Classe

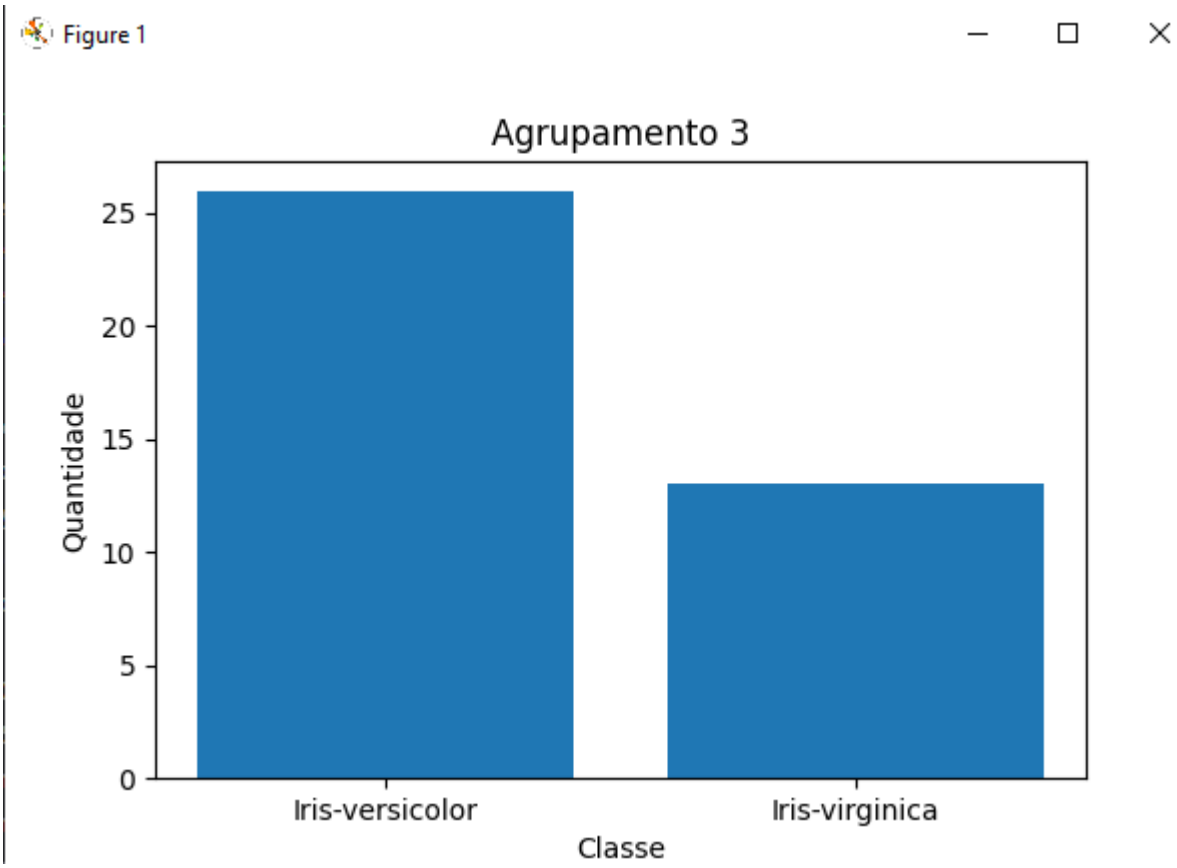
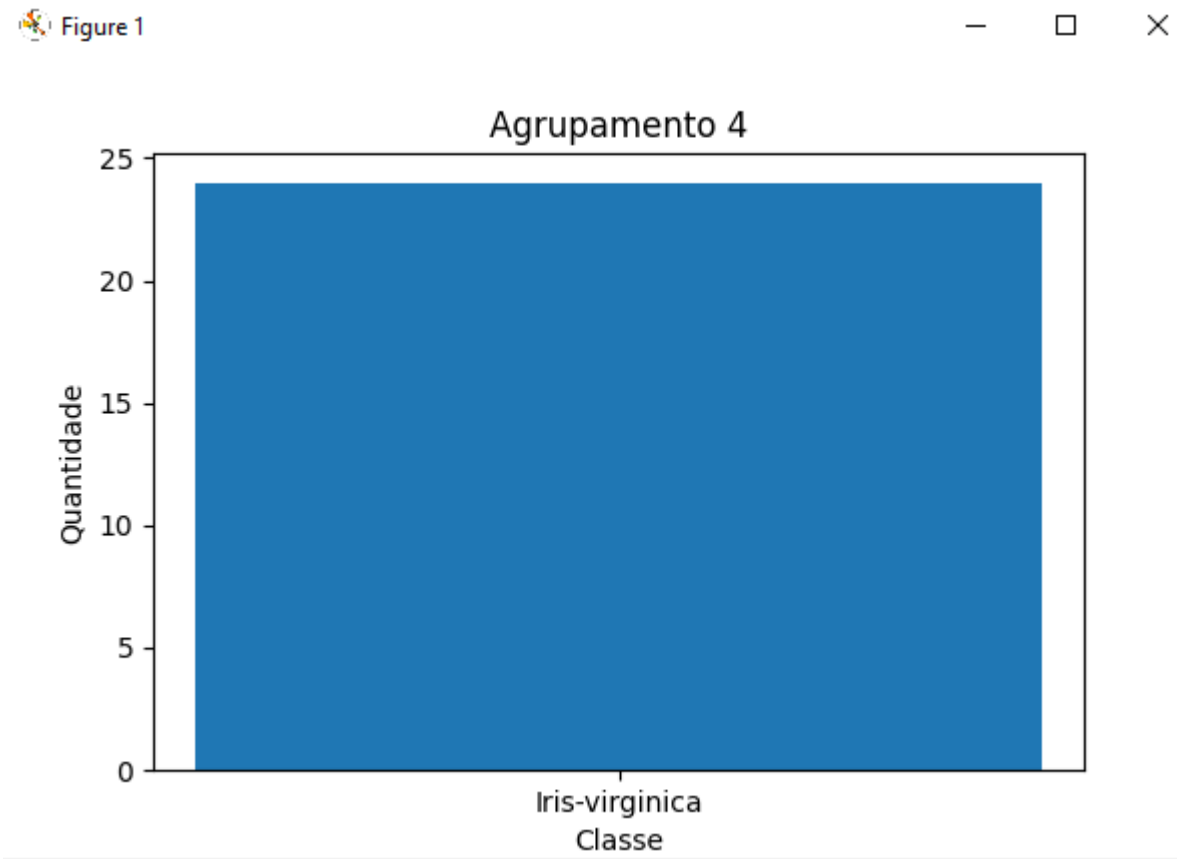


Figure 1
Agrupamento 3
25
20
15
10
5
0
Quantidade
Iris-versicolor
Iris-virginica
Classe



(b) (10 pontos) 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).

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

iris = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data", header=None
iris.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']

X = iris[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]

distance_to_centroids = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42, max_iter=i).fit(X)
    distance = kmeans.transform(X).min(axis=1)
    distance_to_centroids.append(distance)

mean_pattern = [np.mean(distance) for distance in distance_to_centroids]
deviation_pattern = [np.std(distance) for distance in distance_to_centroids]

table = pd.DataFrame({'média': mean_pattern, 'desvio': deviation_pattern}, index=range(1, 11))
print(table)
```

	m?dia	desvio
1	1.943034	0.873755
2	0.860367	0.525982
3	0.648839	0.324468
4	0.558194	0.265592
5	0.511478	0.224266
6	0.466177	0.205473
7	0.438666	0.195529
8	0.411108	0.177647
9	0.400513	0.163334
10	0.383448	0.173828

Caso não tenha construído sua própria implementação utilize alguma biblioteca para as outras partes da questão. Ex.: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html (b) Dica: random_state = constante, max_iter = 1, 2, 3, . . . , 10.

2. Utilizado a base Iris , https://archive.ics.uci.edu/ml/datasets/Iris, em um experi mento do tipo Holdout 50/50 Estrati cado realize seleção de protótipos da seguinte forma, para *k* = 9:

(a) (10 pontos) Execute o *k*-medóides no apenas no conjunto de treino.

```python
from sklearn.datasets import load_iris
from sklearn_extra.cluster import KMedoids
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

iris = load_iris()

X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.5, stratify=iris.target)

n_clusters = 9
kmedoids = KMedoids(n_clusters=n_clusters, random_state=0)

kmedoids.fit(X_train)
```

(b) (15 pontos) 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. O conjunto de treino terá apenas *k* exemplos. A redução do conjunto de treino é chamada seleção de protótipos. O conjunto de teste permanecer inalterado, isto é, os 50% dos dados original (75 exemplos). Utilizando o conjunto de treino reduzido (composto apenas pelos centróides), calcule a taxa de acerto POR CLASSE para o conjunto de teste utilizando o classi cador 1-NN com distância Euclidiana.

```python
from sklearn.datasets import load_iris
from sklearn_extra.cluster import KMedoids
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

iris = load_iris()

X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.5, stratify=iris.targe

n_clusters = 9
kmedoids = KMedoids(n_clusters=n_clusters, random_state=0)

kmedoids.fit(X_train)

X_test_centroids = X_test[kmedoids.medoid_indices_]
y_test_centroids = y_test[kmedoids.medoid_indices_]

knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(X_train, y_train)

y_pred_test = knn.predict(X_test_centroids)

y_pred_train = knn.predict(X_train)
accuracy_train_by_class = []
for i in range(3):
    indices = (y_train == i)
    accuracy_train = sum(y_pred_train[indices] == y_train[indices]) / sum(indices)
    accuracy_train_by_class.append(accuracy_train)
print(f"Taxa de acerto por classe: {accuracy_train_by_class}")
```

```
Taxa de acerto por classe: [1.0, 1.0, 1.0]
```

Caso não tenha feito uma implementação própria utilize alguma biblioteca para as outras partes da questão. Ex.: https://scikit-learn-extra.readthedocs.io/en/latest/generated/sklearn_extra.cluster.KMedoids.html

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 lmes mais ava liados na base. São 943 usuários e cada lme 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) (10 pontos) Realize agrupamento hierárquico aglomerativo.

   (b) (5 pontos) Gere o dendrograma do agrupamento.

   (c) (10 pontos) Pesquise qual o gênero de cada lme (comédia, policial, cção cientí ca etc.). Analise se o faz sentido, em relação ao gênero, cada vez que dois grupos são unidos no dendrograma.

   Dica: utilize bibliotecas para gerar o agrupamento e para construir o dendrograma. Exem plos a seguir. Para realizar o agrupamento: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html Para gerar o dendrograma https://scikit-learn.org/stable/auto_examples/cluster/plot_agglomerative_dendrogram.html

4. Utilizando uma base construída a partir de uma imagem digital de uma fotogra a da natureza (escolha uma imagem, exemplo abaixo), cada pixel da imagem é um elemento do conjunto de dados.

(a) (5 pontos) 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.

```
img_vet = np.reshape(image,(1407*650, 3))
```

```
pixel_df = pd.DataFrame(img_vet,columns=["RED","GREEN","BLUE"])
pixel_df
```

	RED	GREEN	BLUE
0	25	134	191
1	24	133	190
2	25	134	191
3	25	134	191
4	24	133	190
...	...	...	...
914545	36	164	213
914546	37	165	214
914547	37	165	214
914548	33	164	216
914549	33	164	216

914550 rows × 3 columns

(b) (5 pontos) 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) (15 pontos) Ao nal da convergência do *k*-médias arredonde os valor da posição de cada centroide 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 centroide, você obterá uma imagem na qual o número de cores distintas é *k*. Veja o exemplo das guras abaixo.

Figura 1: Exemplo de imagem de entrada (esquerda). Saída da imagem exemplo após o agrupamento utilizando 8 grupos e 1.000 iterações do k-médias (direita).

```
model = cluster.KMeans(8,init="k-means++",max_iter=30)
model.fit_predict(img_vet)

img_clustered = model.cluster_centers_[model.labels_] #substituindo valores do pixel pelo do centroide
img_clustered = np.clip(img_clustered.astype('uint8'), 0, 255) #Corte (limite) os valores em uma matriz.
final = np.reshape(img_clustered,(1407, 650, 3))

plt.axis("off")
plt.imshow(final)
```

✓ 26.2s

```
model = cluster.KMeans(64,init="k-means++",max_iter=30)
model.fit_predict(img_vet)

img_clustered = model.cluster_centers_[model.labels_]
img_clustered = np.clip(img_clustered.astype('uint8'), 0, 255)
final = np.reshape(img_clustered,(1407, 650, 3))

plt.axis("off")
plt.imshow(final)
```

```
model = cluster.KMeans(512,init="k-means++",max_iter=30)
model.fit_predict(img_vet)

img_clustered = model.cluster_centers_[model.labels_]
img_clustered = np.clip(img_clustered.astype('uint8'), 0, 255)
final = np.reshape(img_clustered,(1407, 650, 3))

plt.axis("off")
plt.imshow(final)
```