**Inteligência Artificial k-means clustering**

**Aluno: Guilherme Dias Tiede**
**Matrícula:2018.1.0033.0042-2**

1. Primeiramente importei as bibliotecas sklearn e matplolib do Python, logo em seguida criei dois vetores “X e Y” recebendo a função “make_blobs” gerando assim, pontos aleatórios usando de Distribuição Gaussiana.

```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
import matplotlib as plt

X_random, y_random = make_blobs(n_samples=200, centers=5, random_state=1)

```

2. Logo em seguida imprimir os dois vetores X e Y resultando nos valores aleatório deles baseados nos parâmetros “n_samples, centers e random_satate”.

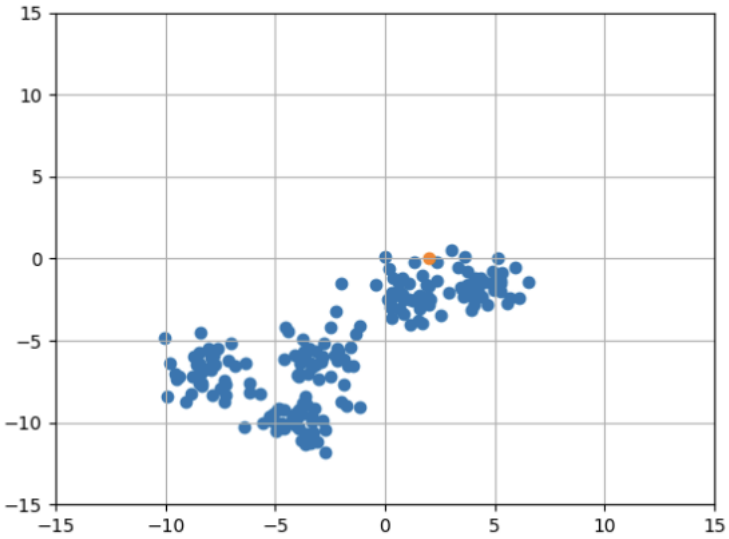
```
print("X_RANDOM",X_random)
print("Y_RANDOM", y_random)
```

```
/usr/bin/python3 /Users/guilhermetiede/PycharmProjects/pythonProject1/main.py
X_RANDOM [[-1.96576392e+00  5.23446451e+00]
 [-5.16022348e+00 -7.04217141e+00]
 [-6.17937069e+00 -2.16733539e+00]
 [-7.39138168e+00 -9.49590389e+00]
 [-6.38481234e+00 -8.47302970e+00]
 [-6.26144310e+00 -3.78347905e+00]
 [-2.04278768e+00  3.07660864e-01]
 [-4.46426086e+00 -4.39451238e+00]
```

```
main
/usr/bin/python3 /Users/guilhermetiede/PycharmProjects/pythonProject1/main.py
Y_RANDOM [0 2 3 2 2 3 4 3 2 0 3 3 4 1 3 0 0 1 2 2 0 3 4 3 4 1 2 4 4 1 2 4 0 3 2 0 1
 2 1 4 2 1 4 4 2 4 1 3 2 0 0 3 3 0 3 1 0 4 2 2 1 3 3 3 2 0 0 1 4 2 1 4 2 4
 4 1 2 3 0 4 1 1 3 2 3 0 2 2 2 1 4 3 0 4 3 1 4 0 2 1 0 2 4 1 4 2 1 2 3 0 3
 3 2 1 3 0 1 4 0 0 2 1 1 3 0 1 2 4 2 4 0 4 3 3 1 1 3 0 0 3 3 3 0 0 4 2 4 0
 2 3 4 3 0 4 3 1 1 1 0 2 4 0 1 0 4 1 1 1 3 4 1 4 4 3 0 2 4 3 2 0 2 0 0 2 1
 4 1 0 0 3 2 4 0 1 1 4 3 4 2 1]
```

3. Após termos visualizados os valores de forma inteira realizei a plotagem do gráfico para melhor visualização. Para isso foi necessário a utilização do “plt.scatter” gerando assim um gráfico de dispersão, onde cada variável tem sua referência.

```
13
14
15    plt.scatter(X_random[:,1], X_random[:,0])
16    plt.scatter(y_random[1],y_random[0])
17    plt.xlim(-15,15)
18    plt.ylim(-15,15)
19    plt.grid()
20    plt.show()
21
```




4. Após plotar o gráfico, criei o método k-means para 5 clusters e o ajustei com algumas Fit Functions

```
k = 5
kmeans = KMeans(n_clusters=k)
kmeans.fit(X_random)
kmeans.fit(X_random)
y_km = kmeans.fit_predict(X_random)
centers = kmeans.cluster_centers_
print(y_km)
print(centers)
```

5. Agora com o k-means definido e aplicado a (X_random) podemos ter os resultados de rotulo e centroides. Primeiramente vou apresentar os valores de rótulos e dos centroides e depois o gráfico com eles reunídos.

```
[1 3 2 3 3 2 0 2 3 1 2 2 2 4 2 1 1 4 3 3 1 2 0 2 0 4 3 0 0 4 3 0 1 2 3 1 4
 3 4 1 4 4 0 0 3 0 4 2 3 0 1 2 2 1 2 4 1 2 3 3 4 2 2 2 3 0 1 4 0 3 4 0 3 0
 0 4 3 4 1 0 4 4 2 3 2 1 3 3 3 4 0 2 1 0 2 4 0 1 3 4 1 3 0 4 0 3 4 3 2 1 2
 2 3 4 2 1 4 0 1 1 3 4 4 2 1 4 3 0 3 0 1 0 2 2 4 4 2 1 1 2 2 2 1 1 0 3 0 1
 3 2 0 2 1 0 2 4 4 4 1 3 0 1 4 1 0 4 4 4 2 0 4 0 0 2 1 3 0 2 3 1 3 1 1 3 4
 0 4 1 1 2 3 0 1 4 4 0 2 0 3 4]
```

```
[[-2.17069756  1.02591979]
 [-1.58338528  4.50520457]
 [-5.90368078 -3.04489641]
 [-6.87958999 -8.11648104]
 [-9.85620522 -3.91021738]]
```

6. Após observar os centroides e os rótulos conseguimos plotar o gráfico usando

```
plt.scatter(X_random[:,0], X_random[:,1], c=y_km, cmap='viridis', s=50)

plt.scatter(centers[:, 0], centers[:, 1], c='red', s=100)

plt.show()
```

7. E assim obtendo o gráfico esperado.