Nome: Nathana Facion

Exercício 4

Aprendizado de Máquina

O relatório está dividido em partes, a função Main que começa o programa se encontra no tópico Main. O código se encontra documentado, ou seja, a descrição do que foi feito em cada parte se encontra no próprio código.

1. Bibliotecas:

```
import numpy as np
from numpy import genfromtxt
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from sklearn import metrics
```

2. Funções usadas:

```
#  Use alguma metrica interna (algum Dunn, Silhouette, Calinski-Harabaz index)
def metric_intern(estimator, data):
  estimator.fit(data)
  return metrics.silhouette_score(data, estimator.labels_,
                    metric='euclidean')

# Use alguma medida externa (Normalized/adjusted Rand, Mutual information, variation of information) para
decidir no k.
def metric_extern(estimator, data, class_origin):
  estimator.fit(data)
  return metrics.adjusted_mutual_info_score(class_origin,  estimator.labels_)
```

3. Main

```
def main():
  fileName = "/home/nathana/AM/exercicio4//cluster-data.csv"
  fileNameClass = "/home/nathana/AM/exercicio4//cluster-class.csv"


  data = genfromtxt(fileName, delimiter=',')[1:]
```

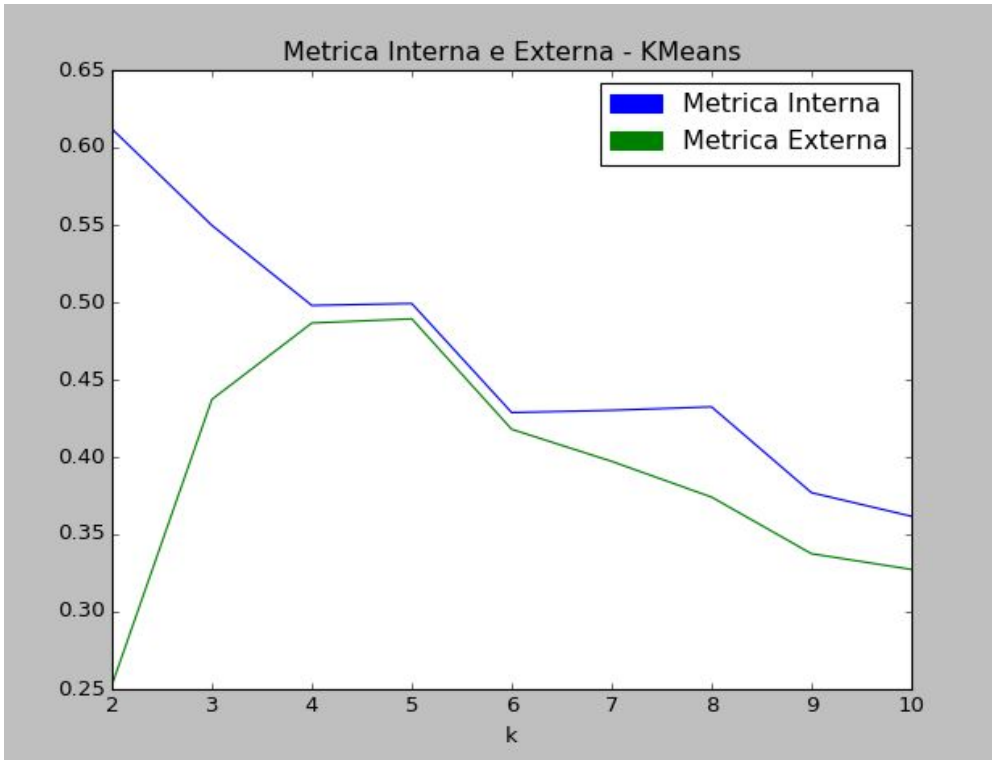
```
dataClass = genfromtxt(fileNameClass, delimiter=',')[1:]

fileData = np.array([d for d in data])
fileClass = np.array([float(d) for d in dataClass])

# k a se escolher
k_init =  2
k_end = 10

# Rode o kmeans nos dados, com numero de restarts = 5
n_init = 5

x = []
y_intern = []
y_extern = []
for k in xrange(k_init, k_end + 1):
   kmeans = KMeans(n_clusters = k, n_init = n_init)
   intern = metric_intern(kmeans,data = fileData)
   extern = metric_extern(kmeans,data = fileData,  class_origin = fileClass)
   x.append(k)
   y_intern.append(intern)
   y_extern.append(extern)
   print 'k: ', k, 'intern: ', intern, 'extern: ', extern

# Plote os graficos correspondentes das 2 metricas (interna e externa)
plt.plot(x, y_intern)
plt.plot(x, y_extern)
plt.xlabel('k')
plt.title('Metrica Interna e Externa - KMeans')
blue_patch = mpatches.Patch(color = 'blue', label = 'Metrica Interna')
green_patch = mpatches.Patch(color = 'green', label = 'Metrica Externa')
plt.legend(handles = [blue_patch,green_patch])
plt.show()

plt.plot(x, y_extern)
plt.xlabel('k')
plt.title('Metrica Externa - KMeans')
blue_patch = mpatches.Patch(color='blue', label='Metrica Externa')
plt.legend(handles=[blue_patch])
plt.show()

plt.plot(x, y_intern)
plt.xlabel('k')
plt.title('Metrica Interna - KMeans')
```

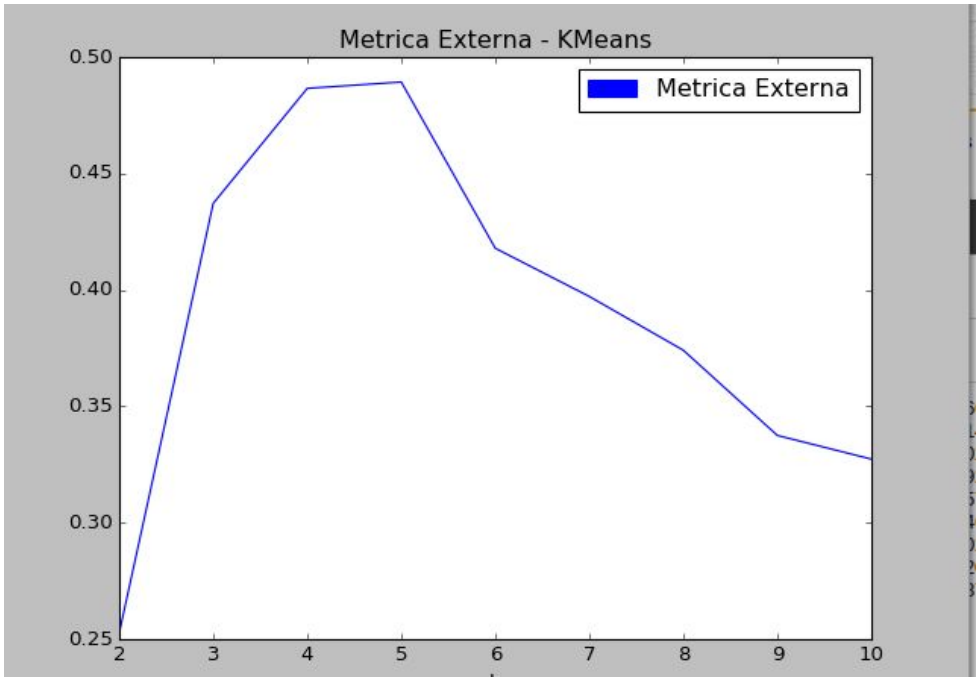
```
  blue_patch = mpatches.Patch(color='blue', label='Metrica Interna')
  plt.legend(handles=[blue_patch])
  plt.show()
if __name__ == '__main__':
  main()
```

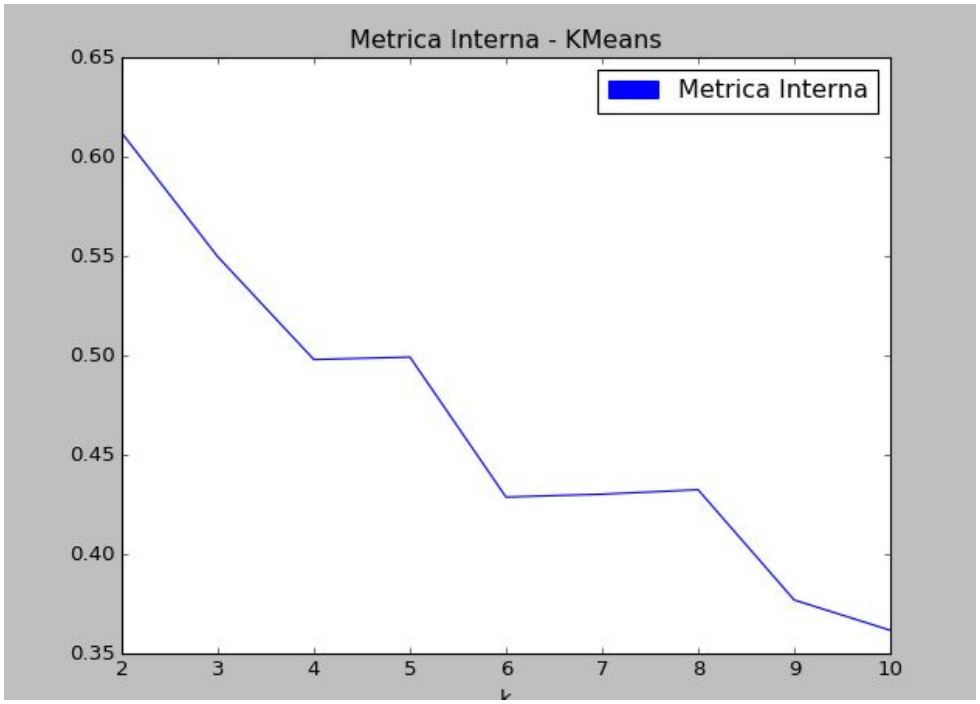
4. Saída

```
k:  2 intern:  0.611941670666 extern:  0.252702262904
k:  3 intern:  0.549640507142 extern:  0.437172019705
k:  4 intern:  0.497901554024 extern:  0.486547750147
k:  5 intern:  0.499160885925 extern:  0.489268020069
k:  6 intern:  0.428696724576 extern:  0.417869431002
k:  7 intern:  0.430128776463 extern:  0.397169698181
k:  8 intern:  0.432388146029 extern:  0.37406973399
k:  9 intern:  0.376913983209 extern:  0.337476253715
k:  10 intern:  0.361658748791 extern:  0.327306661756
```

5. Gráficos




Metrica Externa - KMeans
Metrica Externa
0.50
0.45
0.40
0.35
0.30
0.25
2
3
4
5
6
7
8
9
10


Metrica Interna - KMeans
Metrica Interna
0.65
0.60
0.55
0.50
0.45
0.40
0.35
2
3
4
5
6
7
8
9
10