**I. Pen-and-paper**

1. Phase E:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Likelihood | | Joint Probability | | Normalized Posteriors | |
| p(x|c=1) | p(x|c=2) | p(x,c=1) | p(x,c=2) | k1 = p(c=1|x) | k2= p(c=2|x) |
| x1 |  |  |  |  |  |  |
| x2 |  |  |  |  |  |  |
| x3 |  |  |  |  |  |  |
| x4 |  |  |  |  |  |  |

p(x,c=1)= p(x|c=1)p(c=1)=0.7 p(x|c=1); p(x,c=2)= p(x|c=2)p(c=2)=0.3 p(x|c=2)

Example for x1:

* c=1:
* c=2:
* Normalized Posteriors:

Phase M:

* Priors:
* Centroids:

Conclusion:

|  |  |  |
| --- | --- | --- |
|  | k=1 | k=2 |
| Priors, |  |  |
| Centroids, |  |  |
| Covariances, |  |  |
| p(x|c=k) |  |  |

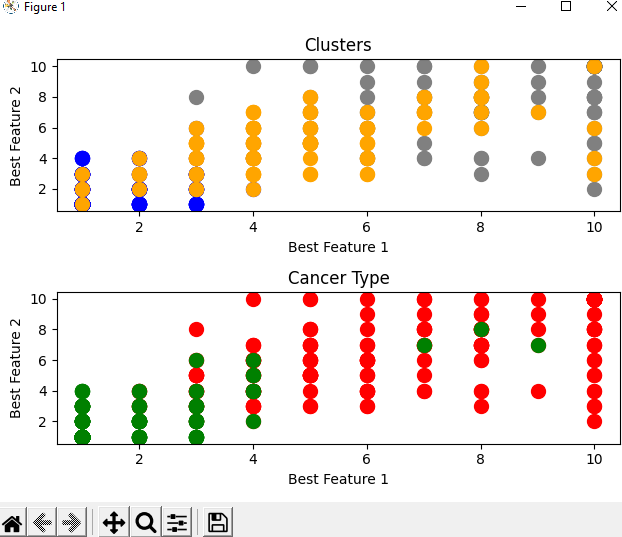


Depois é mais do mesmo para os outros xi

We need more training observations to avoid overfitting in A when compared to B, because it

**II. Programming and critical analysis**

1. ECR (k=2) = 13.5; ECR (k=3) = 6.(6)

1. s(k=2) = 0.5968 s(k=3) = 0.5245
2. 

Best Feature 1: y2 = Cell\_Size\_Uniformity

Best Feature 2: y3 = Cell\_Shape\_Uniformity

(According to selector2.get\_support())

Cluster 1: gray

Cluster 2: blue

Cluster 3: orange

Benign Cancer: green

Malignant Cancer: red

For easier visualization purposes (since some observations in different clusters and cancer types overlap):



**III. APPENDIX**

import pandas as pd; from sklearn.metrics import silhouette\_score

from sklearn.cluster import KMeans;from scipy.io import arff; import numpy as np

from sklearn.feature\_selection import SelectKBest, mutual\_info\_classif

import matplotlib.pyplot as plt

tableVar = pd.DataFrame(arff.loadarff("breast.w.arff")[0]).dropna()

tableVar["Class"] = tableVar["Class"].apply(lambda y : y.decode("utf-8"))

x=[]

for i in range(len(tableVar)):x += [list(tableVar.iloc[i][0:9])]

x= np.array(x)

y = list(tableVar["Class"]); y= np.array(y)

for el in range(len(y)):

    if y[el] == "benign":y[el] = 0

    else:y[el] = 1

cluster\_dict\_k\_2= {0:[0,0], 1:[0,0]}; cluster\_dict\_k\_3= {0:[0,0], 1:[0,0], 2:[0,0]}

phi\_dict\_k\_2= {0:[0,0], 1:[0,0]}; phi\_dict\_k\_3= {0:[0,0], 1:[0,0], 2:[0,0]}

erc\_d = {2:0, 3:0}; sil\_d = {2:0, 3:0}

def compute\_ben\_mal\_all(\_x\_predict, cluster\_dict\_k):

  for i in range(len(\_x\_predict)):

    cluster\_x = \_x\_predict[i]; label\_x = int(y[i])

    cluster\_dict\_k[cluster\_x][label\_x]+=1

  return cluster\_dict\_k

def computePhiDict(phiD, clusterD):

  for i in range(len(clusterD)):

    phi = max(clusterD[i]);cMod= np.sum(clusterD[i]);phiD[i] = [phi, cMod]

  return phiD

def computeErc(phiD):

  erc = 0

  for i in range(len(phiD)):

    phi = phiD[i][0];cMod = phiD[i][1];erc += cMod-phi

  erc = erc/len(phiD)

  return erc

for n in [2,3]:

    km = KMeans(n\_clusters=n)

    x\_predict=km.fit\_predict(x)

    if (n==2):

      compute\_ben\_mal\_all(x\_predict, cluster\_dict\_k\_2)

      computePhiDict(phi\_dict\_k\_2, cluster\_dict\_k\_2)

      erc\_d[n] = computeErc(phi\_dict\_k\_2)

    elif (n==3):

      compute\_ben\_mal\_all(x\_predict, cluster\_dict\_k\_3)

      computePhiDict(phi\_dict\_k\_3, cluster\_dict\_k\_3)

      erc\_d[n] = computeErc(phi\_dict\_k\_3)

    #====================================== b ==================================

    score = silhouette\_score(x, km.labels\_)

    sil\_d[n] = score

print("silhouette\_dict", sil\_d)

print("erc\_d", erc\_d)

km2 = KMeans(n\_clusters=3)

selector2 = SelectKBest(score\_func= mutual\_info\_classif, k=2)

selector2.fit(x,y); c\_pred = km2.fit\_predict(x); t1 = selector2.transform(x)

filtered\_label0 = t1[c\_pred == 0]; filtered\_label1 = t1[c\_pred == 1];

filtered\_label2 = t1[c\_pred == 2]; fig, axs = plt.subplots(2)

axs[0].scatter(t1[c\_pred==0, 0], t1[c\_pred==0, 1], s=100, c='gray', label ='Cluster 1')

axs[0].scatter(t1[c\_pred==1, 0], t1[c\_pred==1, 1], s=100, c='blue', label ='Cluster 2')

axs[0].scatter(t1[c\_pred==2, 0], t1[c\_pred==2, 1], s=100, c='orange', label ='Cluster 3')

axs[0].set\_title('Clusters'); axs[0].set\_ylabel("Best Feature 2")

axs[0].set\_xlabel("Best Feature 1")

axs[1].scatter(t1[y=="1", 0], t1[y=="1", 1], s=100, c='red', label ='y 2')

axs[1].scatter(t1[y=="0", 0], t1[y=="0", 1], s=100, c='green', label ='y 1')

axs[1].set\_title('Cancer Type'); axs[1].set\_ylabel("Best Feature 2")

axs[1].set\_xlabel("Best Feature 1")

fig.tight\_layout()

plt.show()

**END**