UNIVERSIDADE CATÓLICA DE PELOTAS
Mestrado em Engenharia Eletrônica e Computação - PGEEC

LISTA DE EXERCÍCIOS 3

PEDRO A. TAVARES

DISCIPLINA: Reconhecimento de Padrões
PROFESSOR: Jose C. M. Bermudez

Pelotas, dezembro de 2019

EXERCÍCIO 1

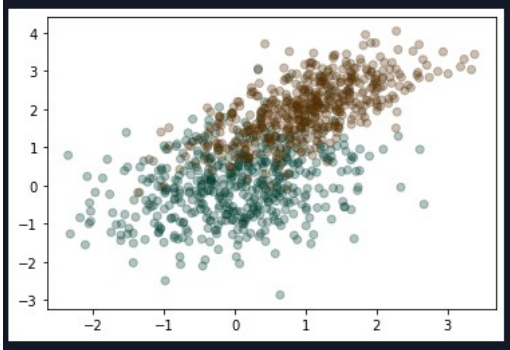
CÓDIGO EM PYTHON:

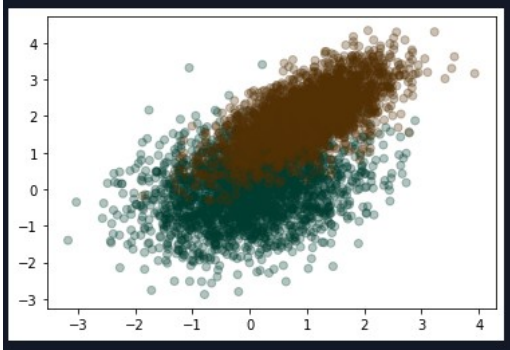
```
"""
LISTA 3 - EXERCÍCIO 1

Implementação de um classificador K-NN

@author: pedro
"""
import numpy as np
from sklearn.neighbors import KNeighborsClassifier as KNN
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal

################
##  Item A)  ##
################

m1 = [0, 0];
m2 = [1, 2];
S1 = [[0.8, 0.2], [0.2, 0.8]];
S2 = [[0.6, 0.4], [0.4, 0.6]];

x11 = (np.random.multivariate_normal(m1, S1, 500).T)
x12 = (np.random.multivariate_normal(m2, S2, 500).T)

x21 = (np.random.multivariate_normal(m1, S1, 2500).T)
x22 = (np.random.multivariate_normal(m2, S2, 2500).T)

X1= np.zeros((2,1000), dtype=float)
X1[0,range(0,500)]   =x11[0, :]
X1[1,range(0,500)]   =x11[1, :]
X1[0,range(500,1000)]=x12[0, :]
X1[1,range(500,1000)]=x12[1, :]

X2= np.zeros((2,5000), dtype=float)
X2[0,range(0,2500)]   =x21[0, :]
X2[1,range(0,2500)]   =x21[1, :]
X2[0,range(2500,5000)]=x22[0, :]
X2[1,range(2500,5000)]=x22[1, :]

y1 = np.ones ((1, 1000), dtype=int)
labelS1= np.ones ((1, 500), dtype=int)
labelS2= np.zeros((1, 500), dtype=int)
y1[0, range(0,500) ]=labelS1
y1[0, range(500,1000)]=labelS2
y1 = np.transpose(y1)

y2 = np.ones ((1, 5000), dtype=int)
labelS21= np.ones ((1, 2500), dtype=int)
labelS22= np.zeros((1, 2500), dtype=int)
y2[0, range(0,2500) ]=labelS21
y2[0, range(2500,5000)]=labelS22
y2 = np.transpose(y2)



################
##  Item B)  ##
################
print("ITEM B):")

plt.scatter(X1[0, :], X1[1, :], c=y1[:, 0], cmap='BrBG', alpha=0.3)
plt.show()
plt.scatter(X2[0, :], X2[1, :], c=y2[:, 0], cmap='BrBG', alpha=0.3)
plt.show()

X1 = np.transpose(X1)

classificador = KNN (n_neighbors=3, weights='uniform', algorithm='brute',
                     metric='euclidean')
classificador.fit(X1, y1[:, 0])
```

```
###############
##  Item C)  ##
###############
print("")
print("ITEM C):")

X2 = np.transpose(X2)

classificacoes = classificador.predict(X2)
plt.scatter(X2[:, 0], X2[:, 1], c=classificacoes, cmap='BrBG', alpha=0.3)
plt.show()

###############
##  Item D)  ##
###############
print("")
print("ITEM D):")

count = 0;
for i in range(0, 5000):
    count = count + (classificacoes[i]-y2[i,0])**2
probal= (count/5000);

def Aux_KNN(X2, K, X1, y1, y2):
    classificador = KNN (n_neighbors=K, weights='uniform', algorithm='brute', metric='euclidean')
    classificador.fit(X1, y1[:, 0]);
    classificacoes = classificador.predict(X2);
    plt.scatter(X2[:, 0], X2[:, 1], c=classificacoes, cmap='BrBG', alpha=0.3)
    plt.show()
    count = 0;
    for i in range(0, 5000):
        count = count + (classificacoes[i]-y2[i,0])**2
    return (count/5000)

probK5 = Aux_KNN(X2, 5, X1, y1, y2)
probK7 = Aux_KNN(X2, 7, X1, y1, y2)
probK15 = Aux_KNN(X2, 15, X1, y1, y2)
probK17 = Aux_KNN(X2, 17, X1, y1, y2)

print("")
print("Prob. de Erro para (k=3) =", (probal*100))
print("Prob. de Erro para (k=5) =", (probK5*100))
print("Prob. de Erro para (k=7) =", (probK7*100))
print("Prob. de Erro para (k=15) =", (probK15*100))
print("Prob. de Erro para (k=17) =", (probK17*100))

###############
##  Item E)  ##
###############
print("")
print("ITEM E):")

x11 = (multivariate_normal(m1, S1))
x21 = (multivariate_normal(m2, S2))
pdf1 = x11.pdf(X2)
pdf2 = x21.pdf(X2)
bay1 = (pdf1/pdf2)>1
bay1 = bay1 * 1
plt.scatter(X2[:, 0], X2[:, 1], c=bay1, cmap='BrBG', alpha=0.3)

count = 0
for i in range(0, 5000):
    count = count + (bay1[i]-y2[i,0])**2
probay= (count/5000)
print("")
print("Prob. de Erro para Bayes =", (probay*100))
```

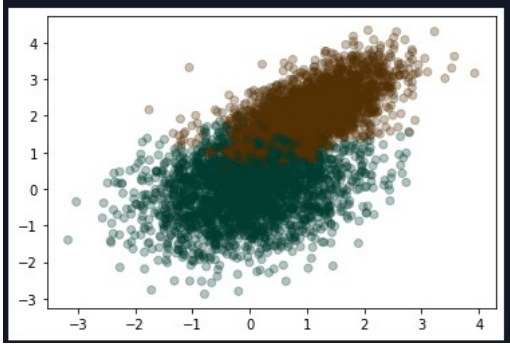
SAÍDA NO CONSOLE DO SPYDER:

```
In [9]: runfile('C:/Users/Pedro/Desktop/Reconhecimento de Padrões/Lista3/Exercicio1.py', wdir='C:/Users/Pedro/Desktop/
Reconhecimento de Padrões/Lista3')
ITEM B):
```

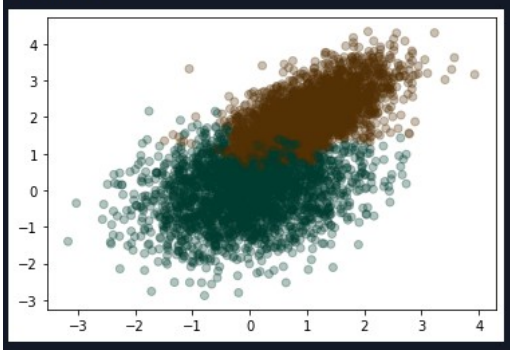


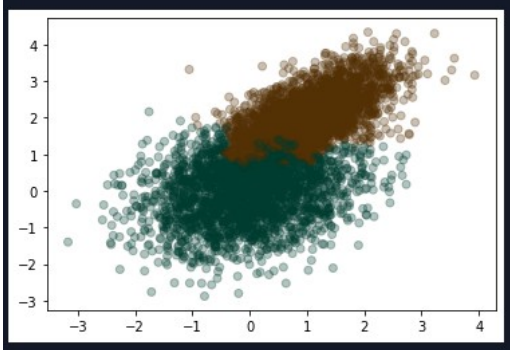


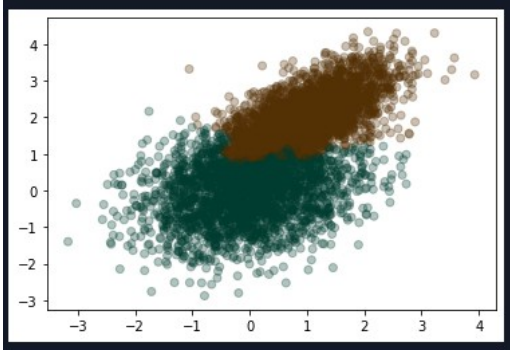
```
ITEM C):
```

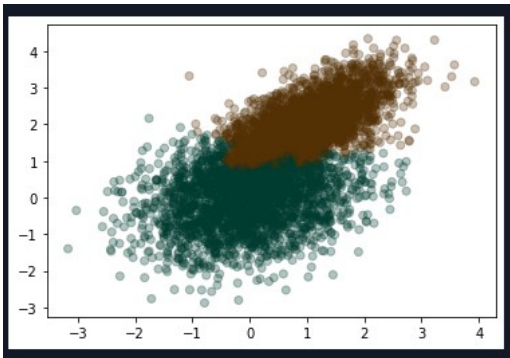


ITEM D):

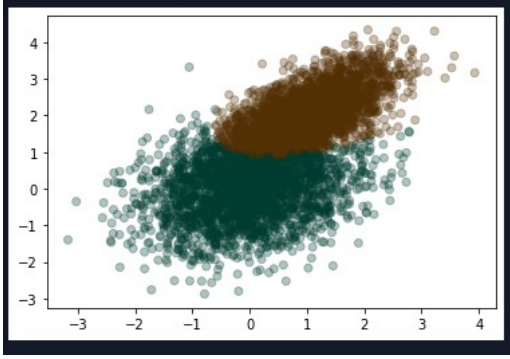









```
Prob. de Erro para (k=3) = 12.559999999999999
Prob. de Erro para (k=5) = 11.62
Prob. de Erro para (k=7) = 11.4
Prob. de Erro para (k=15) = 10.84
Prob. de Erro para (k=17) = 10.86

ITEM E):

Prob. de Erro para Bayes = 10.54
```



EXERCÍCIO 2

CÓDIGO EM PYTHON:

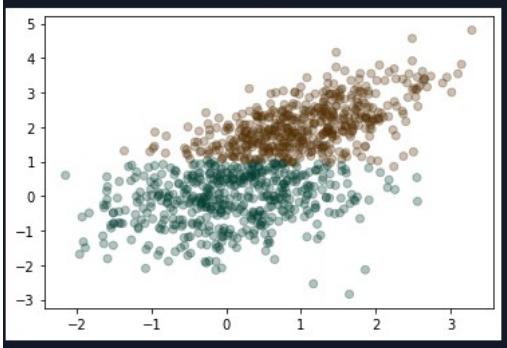
```
# -*- coding: utf-8 -*-
"""
LISTA 3 - EXERCÍCIO 2

Implementação de um classificador SVM linear

@author: pedro
"""
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
from sklearn.svm import SVC

###############
##  Item A)  ##
###############
m1 = [0, 0]
m2 = [1, 2]
s1 = [[0.8, 0.2], [0.2, 0.8]]
s2 = [[0.6, 0.4], [0.4, 0.6]]

x11 = (np.random.multivariate_normal(m1, s1, 500).T)
x12 = (np.random.multivariate_normal(m2, s2, 500).T)

x21 = (np.random.multivariate_normal(m1, s1, 2500).T)
x22 = (np.random.multivariate_normal(m2, s2, 2500).T)

X1= np.zeros((2,1000), dtype=float)
X1[0,range(0,500)]   =x11[0, :]
X1[1,range(0,500)]   =x11[1, :]
X1[0,range(500,1000)]=x12[0, :]
X1[1,range(500,1000)]=x12[1, :]

X2= np.zeros((2,5000), dtype=float)
X2[0,range(0,2500)]   =x21[0, :]
X2[1,range(0,2500)]   =x21[1, :]
X2[0,range(2500,5000)]=x22[0, :]
X2[1,range(2500,5000)]=x22[1, :]

y1 = np.ones ((1, 1000), dtype=int)
labels1= np.ones ((1, 500), dtype=int)
labels2= np.zeros((1, 500), dtype=int)
y1[0, range(0,500) ]=labels1
y1[0, range(500,1000)]=labels2
y1 = np.transpose(y1)

y2 = np.ones ((1, 5000), dtype=int)
labels21= np.ones ((1, 2500), dtype=int)
labels22= np.zeros((1, 2500), dtype=int)
y2[0, range(0,2500) ]=labels21
y2[0, range(2500,5000)]=labels22
y2 = np.transpose(y2)

X1 = np.transpose(X1)
X2 = np.transpose(X2)

###############
##  Item B)  ##
###############

def Classificador(C):
    classif = SVC (C=C, kernel='linear', shrinking=True, probability=False,
                   tol=0.001, class_weight=None, verbose=False, max_iter=-1,
                   decision_function_shape='ovr', random_state=None)
    return classif
```

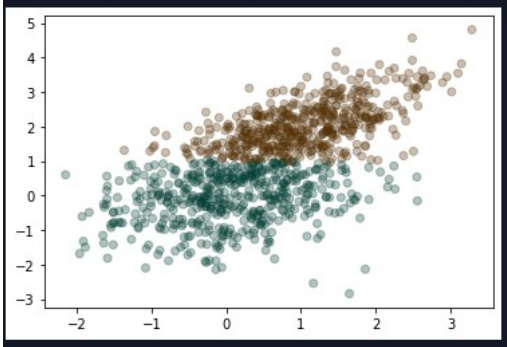
```
###############
##  Item C)  ##
###############

c01= Classificador(0.1)
c02= Classificador(0.2)
c05= Classificador(0.5)
c1 = Classificador(1)
c2 = Classificador(2)
c20= Classificador(20)

###############
##  Item D)  ##
###############

def Aux_SVCX1(X2, classificador, X1, y1, y2):
    classificador.fit(X1, y1[:, 0])
    classificacoes = classificador.predict(X1)
    plt.scatter(X1[:, 0], X1[:, 1], c=classificacoes, cmap='BrBG', alpha=0.3)
    plt.show()
    count = 0
    for i in range(0, 1000):
        count = count + (classificacoes[i]-y1[i,0])**2
    return (count/5000)

def Aux_SVCX2(X2, classificador, X1, y1, y2):
    classificador.fit(X1, y1[:, 0])
    classificacoes = classificador.predict(X2)
    plt.scatter(X2[:, 0], X2[:, 1], c=classificacoes, cmap='BrBG', alpha=0.3)
    plt.show()
    count = 0
    for i in range(0, 5000):
        count = count + (classificacoes[i]-y2[i,0])**2
    return (count/5000)

PE_X1_01 = Aux_SVCX1 (X1, c01, X1, y1, y2)
PE_X1_02 = Aux_SVCX1 (X1, c02, X1, y1, y2)
PE_X1_05 = Aux_SVCX1 (X1, c05, X1, y1, y2)
PE_X1_1  = Aux_SVCX1 (X1, c1,  X1, y1, y2)
PE_X1_2  = Aux_SVCX1 (X1, c2,  X1, y1, y2)
PE_X1_20 = Aux_SVCX1 (X1, c20, X1, y1, y2)

print("")
print("Prob. de Erro de Treinamento para (c=0.1) =", (PE_X1_01*100))
print("Prob. de Erro de Treinamento para (c=0.2) =", (PE_X1_02*100))
print("Prob. de Erro de Treinamento para (c=0.5) =", (PE_X1_05*100))
print("Prob. de Erro de Treinamento para (c=1) =", (PE_X1_1*100))
print("Prob. de Erro de Treinamento para (c=2) =", (PE_X1_2*100))
print("Prob. de Erro de Treinamento para (c=20) =", (PE_X1_20*100))

PE_X2_01 = Aux_SVCX2 (X2, c01, X1, y1, y2)
PE_X2_02 = Aux_SVCX2 (X2, c02, X1, y1, y2)
PE_X2_05 = Aux_SVCX2 (X2, c05, X1, y1, y2)
PE_X2_1  = Aux_SVCX2 (X2, c1,  X1, y1, y2)
PE_X2_2  = Aux_SVCX2 (X2, c2,  X1, y1, y2)
PE_X2_20 = Aux_SVCX2 (X2, c20, X1, y1, y2)

print("")
print("Prob. de Erro de Teste para (c=0.1) =", (PE_X2_01*100))
print("Prob. de Erro de Teste para (c=0.2) =", (PE_X2_02*100))
print("Prob. de Erro de Teste para (c=0.5) =", (PE_X2_05*100))
print("Prob. de Erro de Teste para (c=1) =", (PE_X2_1*100))
print("Prob. de Erro de Teste para (c=2) =", (PE_X2_2*100))
print("Prob. de Erro de Teste para (c=20) =", (PE_X2_20*100))

## Vetores de suporte
v01 = c01.n_support_
v02 = c02.n_support_
v05 = c05.n_support_
v1 = c1.n_support_
v2 = c2.n_support_
v20 = c20.n_support_
```

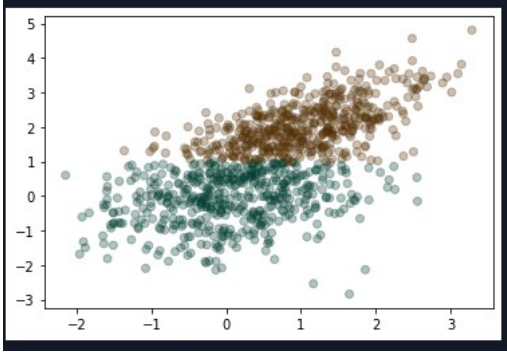
```
print("")
print ("Vetor de suporte para (c=0.1) = ", v01)
print ("Vetor de suporte para (c=0.2) = ", v02)
print ("Vetor de suporte para (c=0.5) = ", v05)
print ("Vetor de suporte para (c=1) = ", v1)
print ("Vetor de suporte para (c=2) = ", v2)
print ("Vetor de suporte para (c=20) = ", v20)

## Margem
def margem (X2, C, X1, y1, y2):
    PE_X2_01 = Aux_SVCX2 (X2, c01, X1, y1, y2)
    w = c01.coef_[0]
    a = -w[0] / w[1]
    xx = np.linspace(-5, 5)
    yy = a * xx - (c01.intercept_[0]) / w[1]
    margem = 1 / np.sqrt(np.sum(c01.coef_ ** 2))
    yy_down = yy - np.sqrt(1 + a ** 2) * margem
    yy_up = yy + np.sqrt(1 + a ** 2) * margem
    fignum = 1
    plt.figure(fignum, figsize=(4, 3))
    plt.plot(xx, yy, 'k-')
    plt.plot(xx, yy_down, 'k--')
    plt.plot(xx, yy_up, 'k--')

M_01 = margem (X2, c01, X1, y1, y2)
M_02 = margem (X2, c02, X1, y1, y2)
M_05 = margem (X2, c05, X1, y1, y2)
M_1  = margem (X2, c1,  X1, y1, y2)
M_2  = margem (X2, c2,  X1, y1, y2)
M_20 = margem (X2, c20, X1, y1, y2)


## Bayes
x11 = (multivariate_normal(m1, s1))
x21 = (multivariate_normal(m2, s2))
pdf1 = x11.pdf(X2)
pdf2 = x21.pdf(X2)
bay1 = (pdf1/pdf2)>1
bay1 = bay1 * 1
plt.scatter(X2[:, 0], X2[:, 1], c=bay1, cmap='BrBG', alpha=0.3)

count = 0
for i in range(0, 5000):
    count = count + (bay1[i]-y2[i,0])**2
probay= (count/5000)

print("")
print("Prob. de Erro para Baye s=", (probay*100))
```

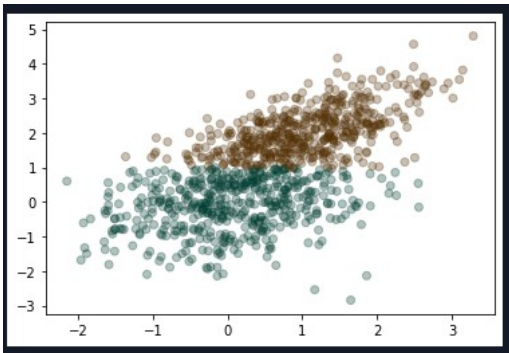
SAÍDA NO CONSOLE DO SPYDER:

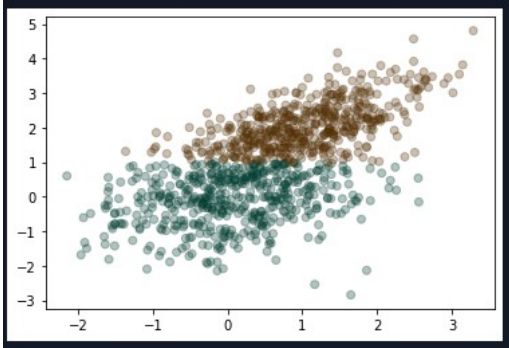
```
In [12]: runfile('C:/Users/Pedro/Desktop/Reconhecimento de Padrões/Lista3/Exercicio2.py', wdir='C:/Users/Pedro/Desktop/
Reconhecimento de Padrões/Lista3')
```

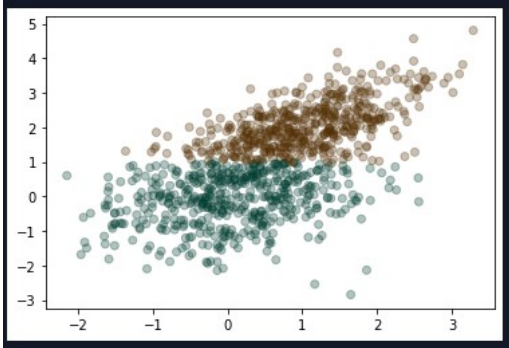




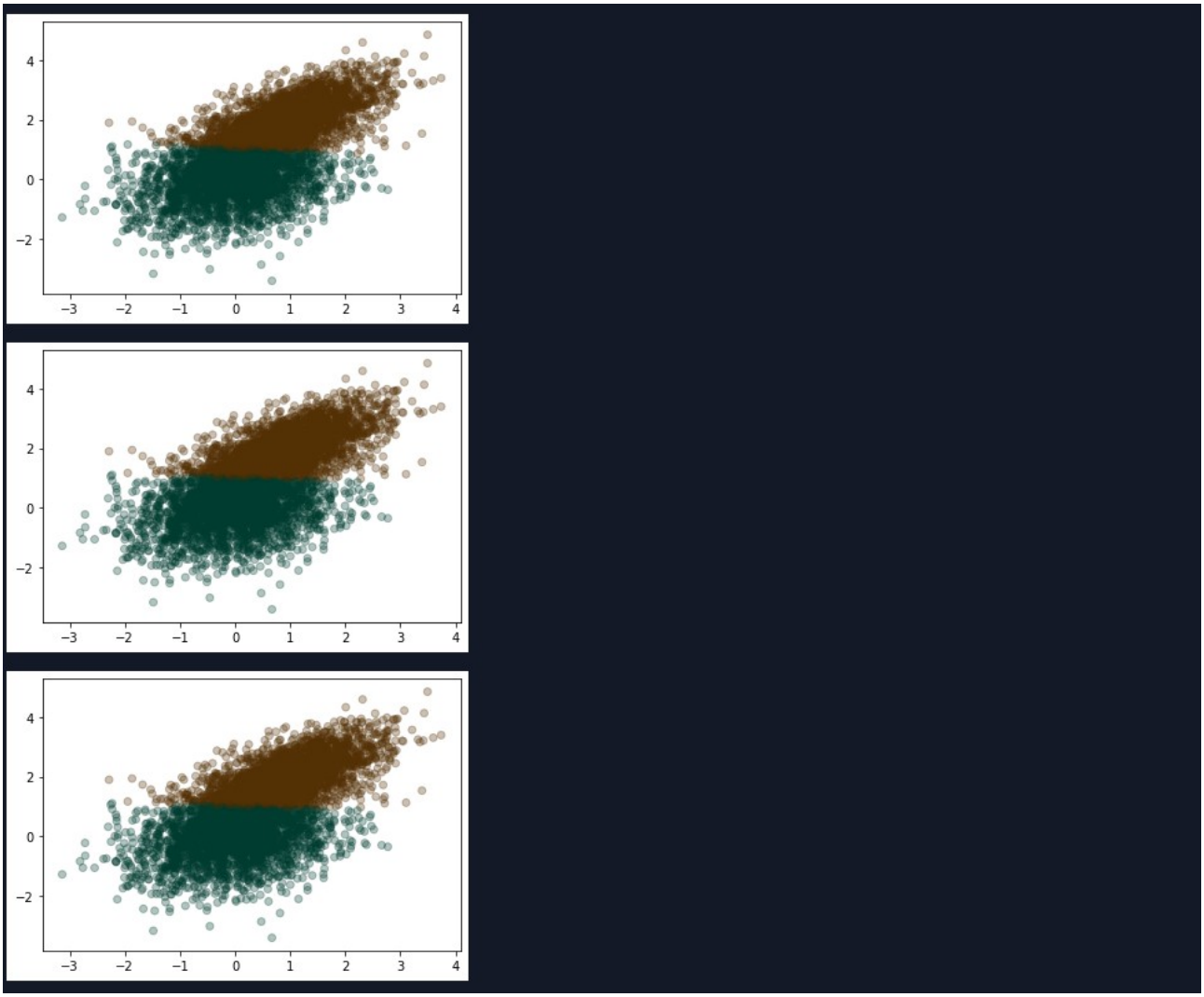


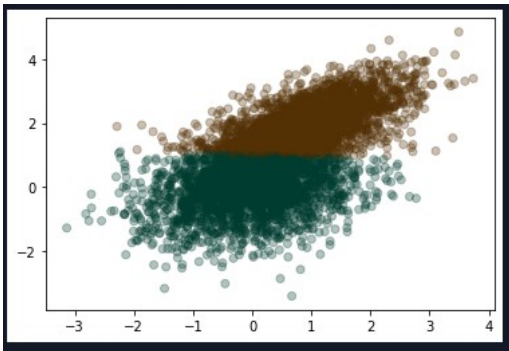


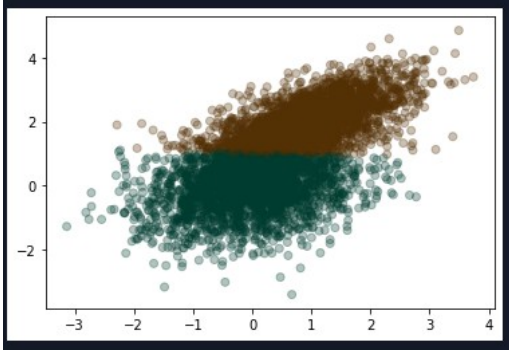


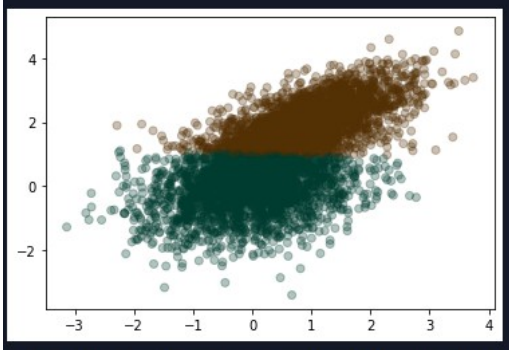



```
Prob. de Erro de Treinamento para (c=0.1) = 2.12
Prob. de Erro de Treinamento para (c=0.2) = 2.1
Prob. de Erro de Treinamento para (c=0.5) = 2.12
Prob. de Erro de Treinamento para (c=1) = 2.12
Prob. de Erro de Treinamento para (c=2) = 2.12
Prob. de Erro de Treinamento para (c=20) = 2.12
```


4
2
0
−2
−3
−2
−1
0
1
2
3
4

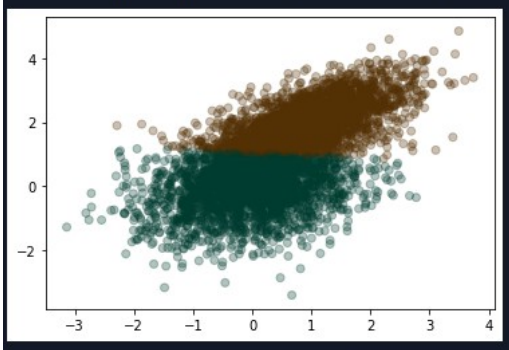


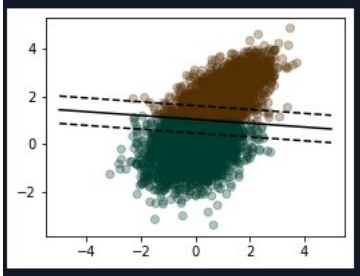


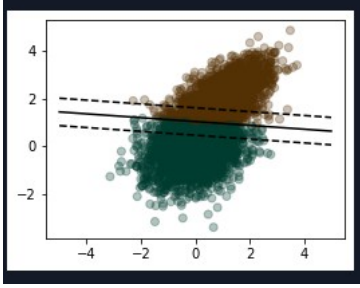


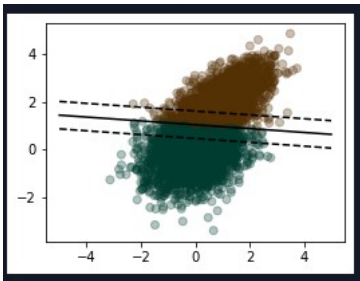

```
Prob. de Erro de Teste para (c=0.1) = 12.540000000000001
Prob. de Erro de Teste para (c=0.2) = 12.520000000000001
Prob. de Erro de Teste para (c=0.5) = 12.479999999999999
Prob. de Erro de Teste para (c=1) = 12.5
Prob. de Erro de Teste para (c=2) = 12.5
Prob. de Erro de Teste para (c=20) = 12.5
```

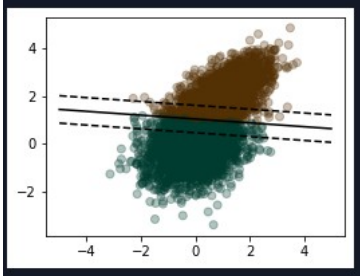
```
Vetor de suporte para (c=0.1) =  [154 154]
Vetor de suporte para (c=0.2) =  [147 147]
Vetor de suporte para (c=0.5) =  [143 142]
Vetor de suporte para (c=1) =  [140 141]
Vetor de suporte para (c=2) =  [139 140]
Vetor de suporte para (c=20) =  [139 139]
```

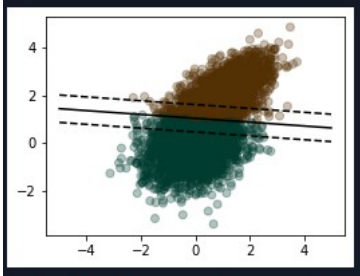




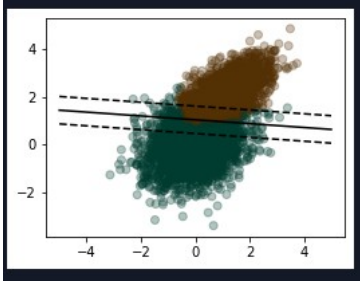









```
Prob. de Erro para Baye s= 11.62
```



EXERCÍCIO 3

CÓDIGO EM PYTHON:

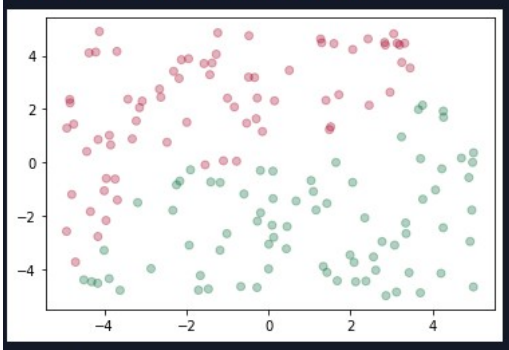
```
# -*- coding: utf-8 -*-
"""
LISTA 3 - EXERCÍCIO 3

@author: pedro
"""

import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC

x10 = 10 * np.random.random_sample((150)) -5
x11 = 10 * np.random.random_sample((150)) -5


aux= np.zeros((150, 1), dtype=float)
teste= np.zeros((150, 1), dtype=float)
for i in range(0, 150):
    aux[i, 0] = 0.05*(x10[i]**3 + x10[i]**2 + x10[i]+ 1)
    if (aux[i, 0] > x11[i]):
        teste[i, 0] = 1
    else:
        teste[i, 0] = 0

plt.scatter(x10, x11, c=teste[:, 0], cmap='RdYlGn', alpha=0.3)
plt.show()

################
##  Item A)  ##
################
print("")
print("ITEM A):")

x20 = 10 * np.random.random_sample((150)) -5
x21 = 10 * np.random.random_sample((150)) -5

aux2= np.zeros((150, 1), dtype=float)
teste2= np.zeros((150, 1), dtype=float)
for i in range(0, 150):
    aux2[i, 0] = 0.05*(x10[i]**3 + x10[i]**2 + x10[i]+ 1)
    if (aux2[i, 0] > x11[i]):
        teste2[i, 0] = 1
    else:
        teste2[i, 0] = 0

plt.scatter(x10, x11, c=teste2[:, 0], cmap='RdYlGn', alpha=0.3)
plt.show()

################
##  Item B)  ##
################
print("")
print("ITEM B):")

X1= np.zeros((2,150), dtype=float)
X1[0,range(0,150)]= x10
X1[1,range(0,150)]= x11
X1 = np.transpose(X1)

X2= np.zeros((2,150), dtype=float)
X2[0,range(0,150)]= x20
X2[1,range(0,150)]= x21
X2 = np.transpose(X2)

def Classificador(C):
    classif = SVC (C=C, kernel='linear', shrinking=True, probability=False,
                   tol=0.001, class_weight=None, verbose=False, max_iter=-1,
                   decision_function_shape='ovr', random_state=None)
    return classif

c01= Classificador(2)
```

```
def Aux_SVCX1(classificador, X1, y1):
    classificador.fit(X1, y1[:, 0])
    classificacoes = classificador.predict(X1)
    plt.scatter(X1[:, 0], X1[:, 1], c=classificacoes, cmap='RdYlGn', alpha=0.3)
    plt.show()
    count = 0
    for i in range(0, 150):
        count = count + (classificacoes[i]-y1[i,0])**2
    return (count/150)

def Aux_SVCX2(X2, classificador, X1, y1, y2):
    classificador.fit(X1, y1[:, 0])
    classificacoes = classificador.predict(X2)
    plt.scatter(X2[:, 0], X2[:, 1], c=classificacoes, cmap='RdYlGn', alpha=0.3)
    plt.show()
    count = 0
    for i in range(0, 150):
        count = count + (classificacoes[i]-y2[i,0])**2
    return (count/150)

PE_X1 = Aux_SVCX1 (c01, X1, teste)
PE_X2 = Aux_SVCX2 (X2, c01, X1, teste, teste2)
plt.show()

print("")
print("Prob. de Erro de Treinamento =", (PE_X1*100))
print("Prob. de Erro de Teste =", (PE_X2*100))

v01 = c01.n_support_
print ("Vetor de suporte = ", v01)

################
##  Item C)  ##
################
print("")
print("ITEM C):")

classif01= SVC(C=2, kernel='rbf', degree=3, gamma=100,
        coef0=0.0, shrinking=True, probability=False, tol=0.001,
        cache_size=200, class_weight=None, verbose=False, max_iter=-1,
        decision_function_shape='ovr', random_state=None)


PE_X1 = Aux_SVCX1 (classif01, X1, teste)
PE_X2 = Aux_SVCX2 (X2, classif01, X1, teste, teste2)
plt.show()

print("")
print("Prob. de Erro de Treinamento 0.1=", (PE_X1*100))
print("Prob. de Erro de Teste 0.1=", (PE_X2*100))

v02 = classif01.n_support_
print ("Vetor de suporte 0.1 = ", v02)


classif2= SVC(C=2, kernel='rbf', degree=3, gamma=0.25,
        coef0=0.0, shrinking=True, probability=False, tol=0.001,
        cache_size=200, class_weight=None, verbose=False, max_iter=-1,
        decision_function_shape='ovr', random_state=None)

PE_X1 = Aux_SVCX1 (classif2, X1, teste)
PE_X2 = Aux_SVCX2 (X2, classif2, X1, teste, teste2)
plt.show()

print("")
print("Prob. de Erro de Treinamento 2 =", (PE_X1*100))
print("Prob. de Erro de Teste 2 =", (PE_X2*100))

v02 = classif2.n_support_
print ("Vetor de suporte 2 = ", v02)
```

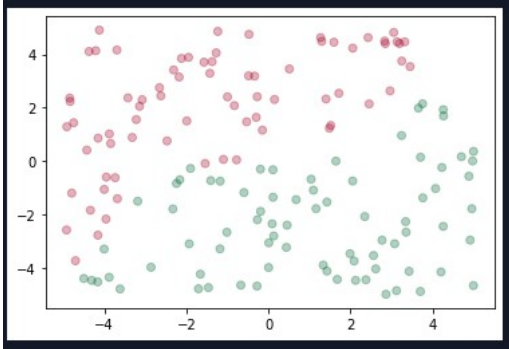
```
## Função para plotar o SVM
def plot_svc(modelo, ax=None, plot_suporte=True):
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = modelo.decision_function(xy).reshape(X.shape)
    ax.contour(X, Y, P, colors='k',
               levels=[-1, 0, 1], alpha=0.5,
               linestyles=['--', '-', '--'])
    if plot_suporte:
        ax.scatter(modelo.support_vectors_[:, 0],
                   modelo.support_vectors_[:, 1],
                   s=300, linewidth=1, facecolors='none')
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)

plt.scatter(X1[:, 0], X1[:, 1], c=teste[:,0], s=50, cmap='Wistia')
plot_svc(classif01)
plt.scatter(classif01.support_vectors_[:, 0], classif01.support_vectors_[:, 1],
            s=300, lw=1, facecolors='none')
plt.show()

plt.scatter(X2[:, 0], X2[:, 1], c=teste2[:,0], s=50, cmap='Wistia')
plot_svc(classif01)
plt.scatter(classif01.support_vectors_[:, 0], classif01.support_vectors_[:, 1],
            s=300, lw=1, facecolors='none');
plt.show()

plt.scatter(X1[:, 0], X1[:, 1], c=teste[:,0], s=50, cmap='Wistia')
plot_svc(classif2)
plt.scatter(classif2.support_vectors_[:, 0], classif2.support_vectors_[:, 1],
            s=300, lw=1, facecolors='none');
plt.show()

plt.scatter(X2[:, 0], X2[:, 1], c=teste2[:,0], s=50, cmap='Wistia')
plot_svc(classif2)
plt.scatter(classif2.support_vectors_[:, 0], classif2.support_vectors_[:, 1],
            s=300, lw=1, facecolors='none');
plt.show()
```

SAÍDA NO CONSOLE DO SPYDER:

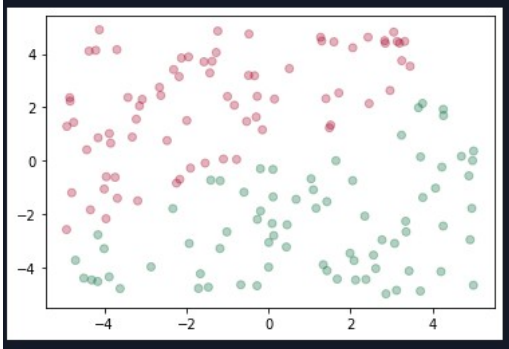
```
In [22]: runfile('C:/Users/Pedro/Desktop/Reconhecimento de Padrões/Lista3/Exercicio3.py', wdir='C:/Users/Pedro/Desktop/
Reconhecimento de Padrões/Lista3')
```

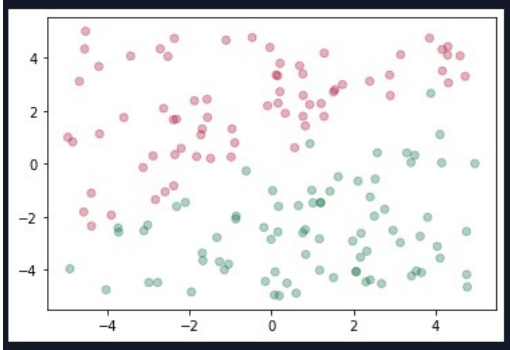


```
ITEM A):
```



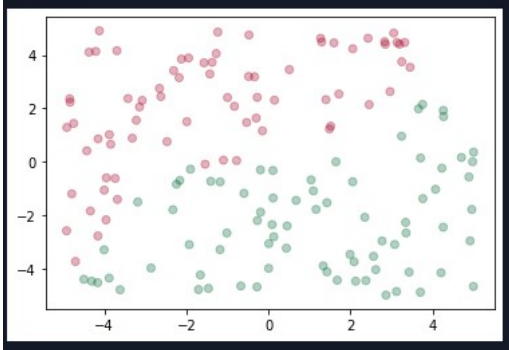
```
ITEM B):
```

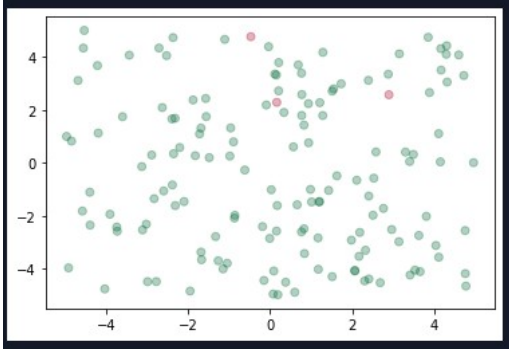




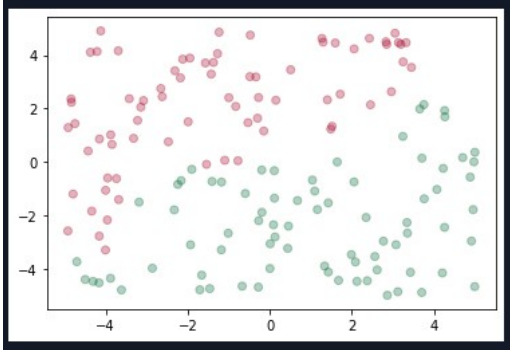
```
Prob. de Erro de Treinamento = 4.0
Prob. de Erro de Teste = 44.0
Vetor de suporte =  [10 11]
```

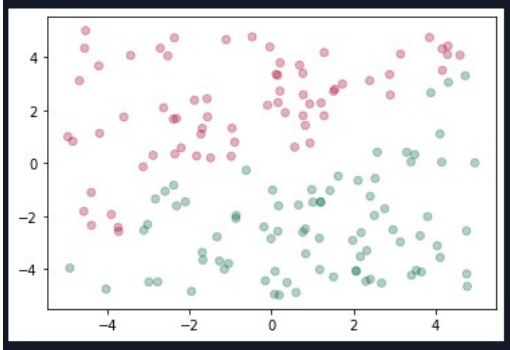
ITEM C):



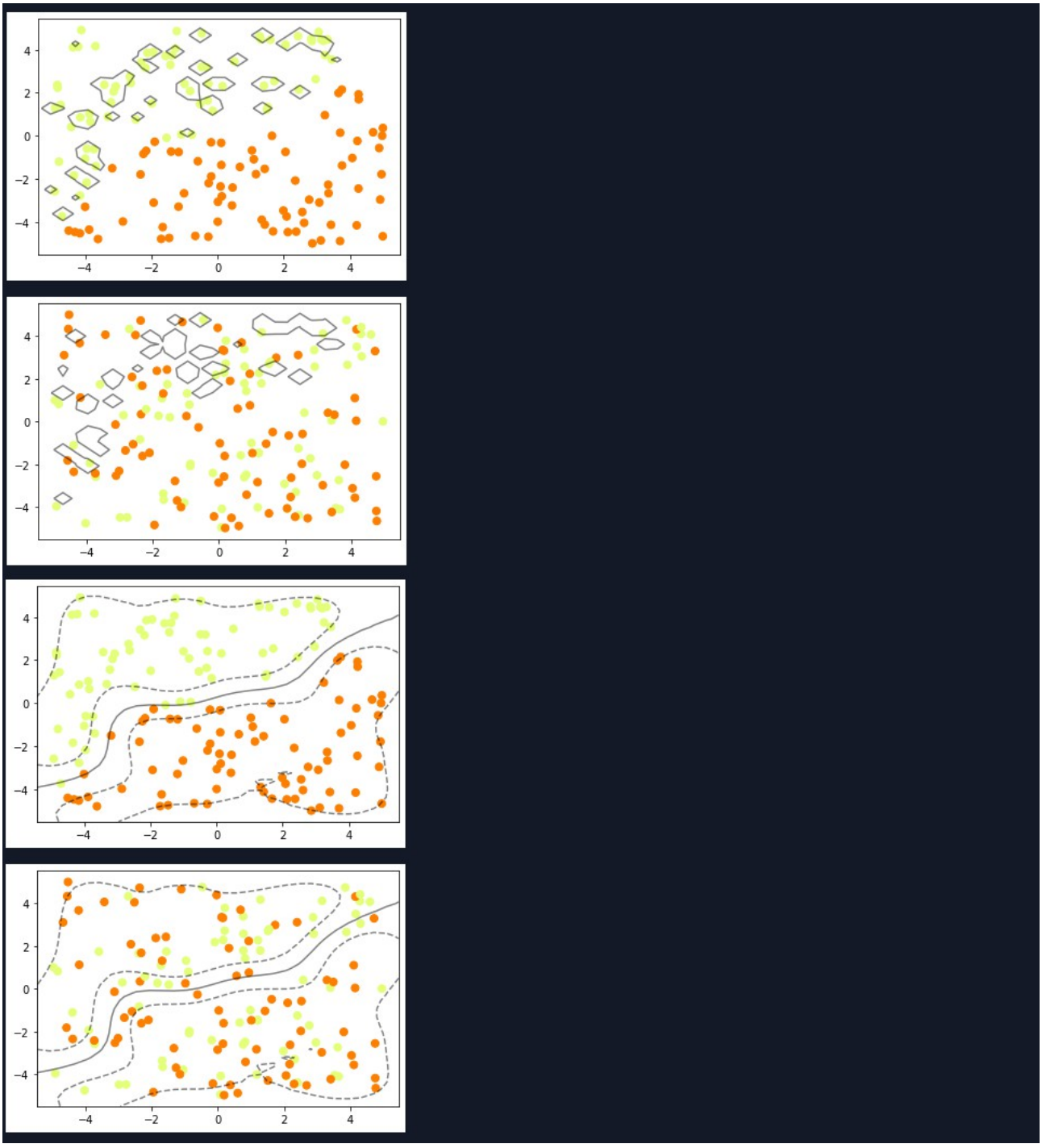



```
Prob. de Erro de Treinamento 0.1= 0.0
Prob. de Erro de Teste 0.1= 46.666666666666664
Vetor de suporte 0.1 =  [73 77]
```






```
Prob. de Erro de Treinamento 2 = 1.3333333333333335
Prob. de Erro de Teste 2 = 44.0
Vetor de suporte 2 =  [20 20]
```


4
2
0
−2
−4
−4
−2
0
2
4