```
import scipy.io as sio
import sklearn.model_selection as sms
import sklearn.naive_bayes as snb
import  sklearn.neighbors as sn
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mc
import math

#1
banana = sio.loadmat("banana.mat")
train_data = banana["train_data"]
train_labels = banana["train_labels"]
train_labels = np.array(train_labels)
test_data = banana["test_data"]
test_labels = banana["test_labels"]
test_labels = np.array(test_labels)
train, dummy, train_targets, dummy = sms.train_test_split  (train_data,
train_labels.ravel(), test_size=0.70)
dummy, test, dummy, test_targets = sms.train_test_split  (test_data,
test_labels.ravel(), test_size=0.70)
#1


#2
gaussiannb = snb.GaussianNB()
tmp = gaussiannb.fit(train, train_targets)
Z = tmp.predict(test)
#2


#3
c1 = (Z == 1).nonzero()
c2 = (Z == 2).nonzero()
plt.scatter(test[c1, 0], test[c1, 1], c="g", label="Grupa 1")
plt.scatter(test[c2, 0], test[c2, 1], c="r", label="Grupa 2")
plt.legend()
# obszary decyzyjne
C = 1.0
h = .02
x_min, x_max = test[:, 0].min() - 1, test[:, 0].max() + 1
y_min, y_max = test[:, 1].min() - 1, test[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),  np.arange(y_min, y_max,
h))
Z = tmp.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contour(xx, yy, Z, cmap=plt.cm.Paired)
plt.show()
#3


#4
print(round(tmp.score(test, test_targets) * 100, 2))
#4


#5
tmp = sn.NearestCentroid()
#5
```

```
#6
tmp.fit(train, train_targets)
Z = tmp.predict(test)
#6


#7
plt.close()
c1 = (Z == 1).nonzero()
c2 = (Z == 2).nonzero()
plt.scatter(test[c1, 0], test[c1, 1], c="g", label="Klasa 1")
plt.scatter(test[c2, 0], test[c2, 1], c="r", label="Klasa 2")
plt.legend()
plt.scatter(tmp.centroids_[:, 0], tmp.centroids_[:, 1], c="b")
plt.show()
#7


#8
print("Sprawnosc klasyfikatora: ", tmp.score(test, test_targets))
#8


#9
bestScore = 0
bestK = 0
for k in range(1, 10):
    clf = sn.KNeighborsClassifier(k, weights='uniform',
metric='euclidean')
    clf.fit(train, train_targets)
    tempScore = clf.score(test, test_targets)
    if tempScore > bestScore:
        bestScore = tempScore
        bestK = k
print("Best score: ", bestScore, ", for k: ", bestK)
#9


#10
cmap_light = mc.ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = mc.ListedColormap(['#FF0000', '#00FF00', '#0000FF'])
Z = sn.KNeighborsClassifier(bestK, weights='uniform',
metric='euclidean').fit(train, train_targets).predict(  np.c_[xx.ravel(),
yy.ravel()])
Z = Z.reshape(xx.shape)
plt.figure()
plt.pcolormesh(xx, yy, Z)
plt.scatter(test[:, 0], test[:, 1], c=test_targets, cmap=cmap_bold)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.show()
#10


#11
clf = sn.KNeighborsClassifier(bestK, weights='uniform',
metric='euclidean')
clf.fit(train, train_targets)
clfScore = clf.score(test, test_targets)
print("Sprawnosc: ",clfScore)
print("zle zakwalifikowanych: ", math.floor(len(test_data) * (1 -
```

```
clfScore)))
#11
```