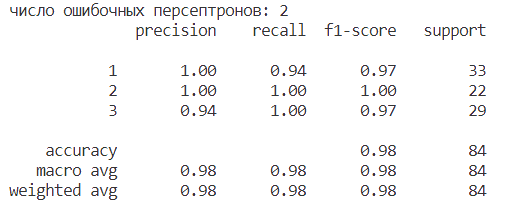
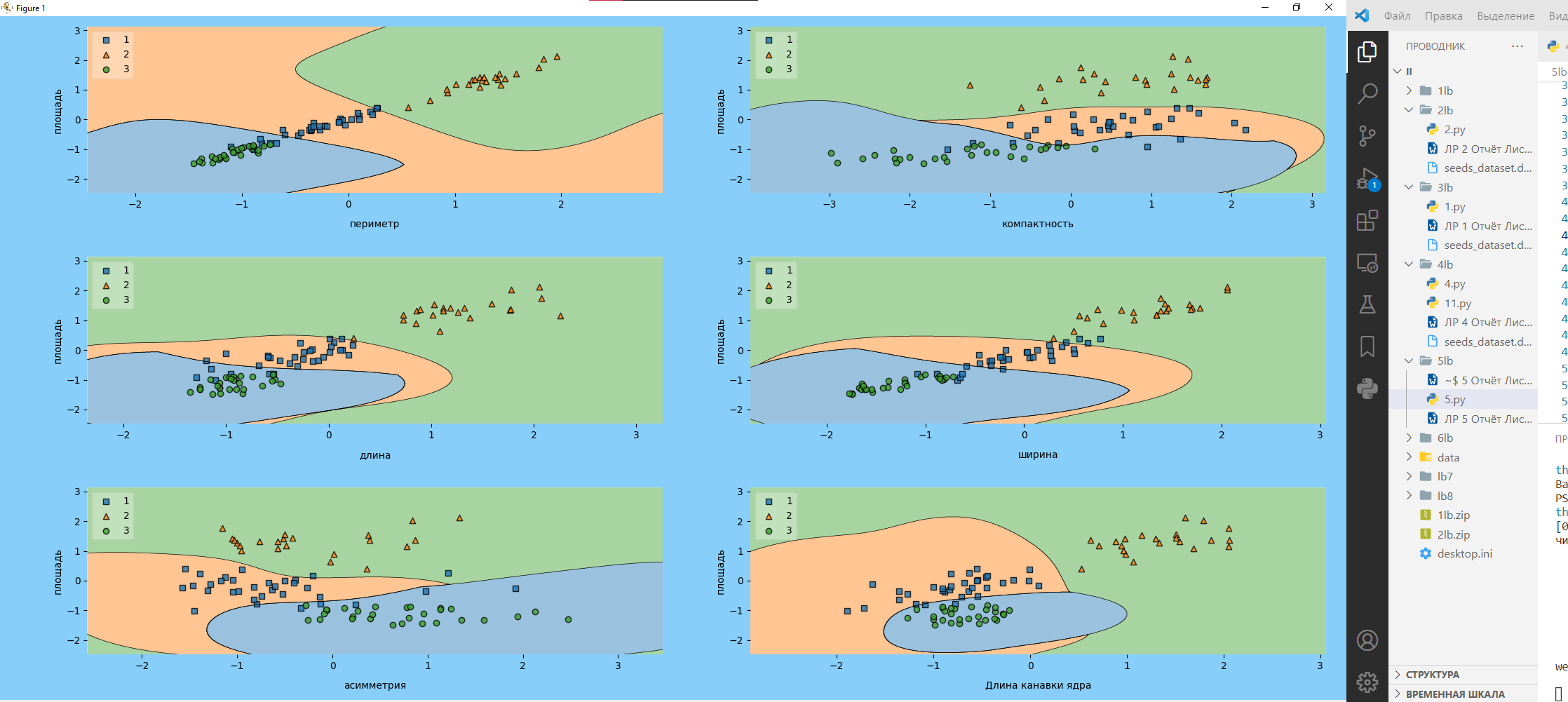
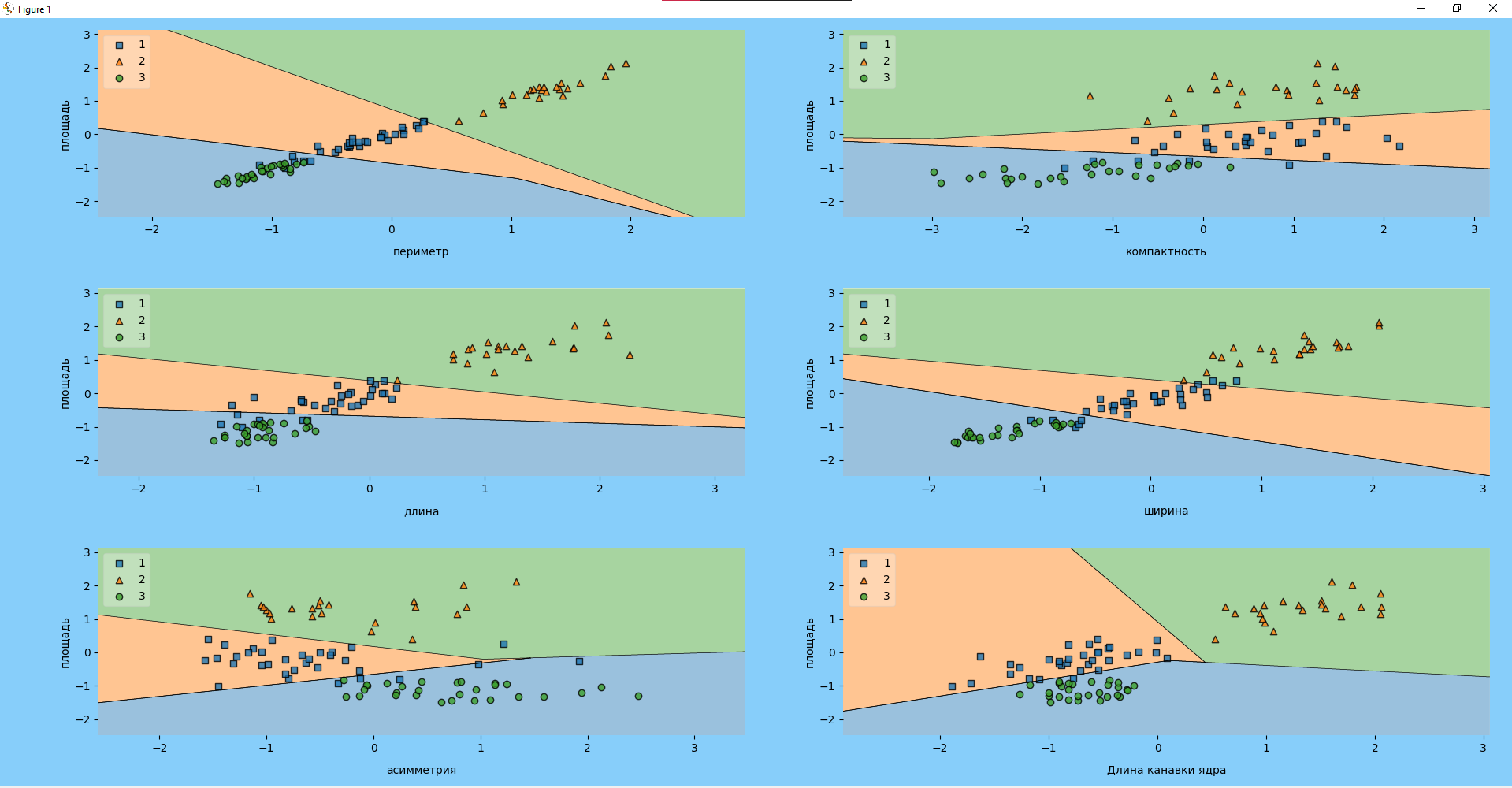
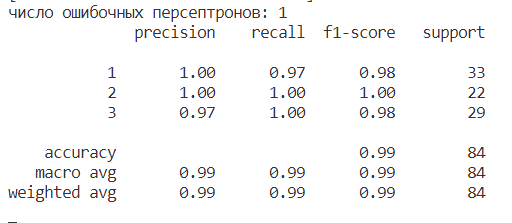
C = 1.0 kernel = 'rbf'

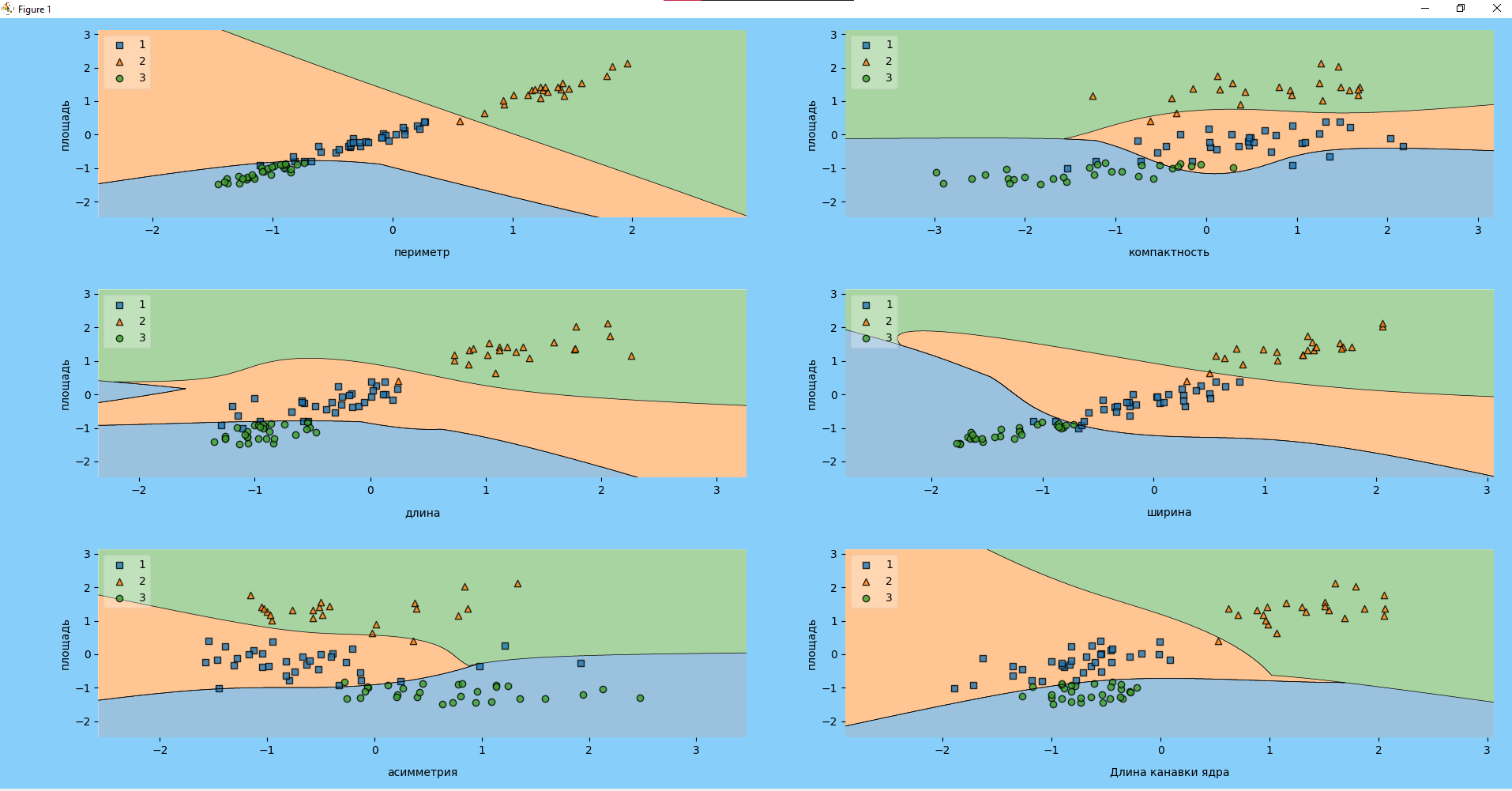


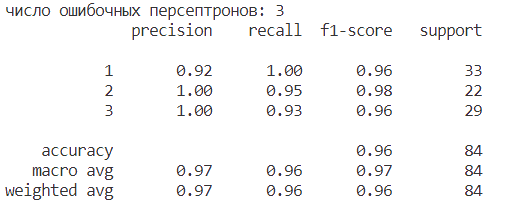
C = 1.0 kernel = 'linear'



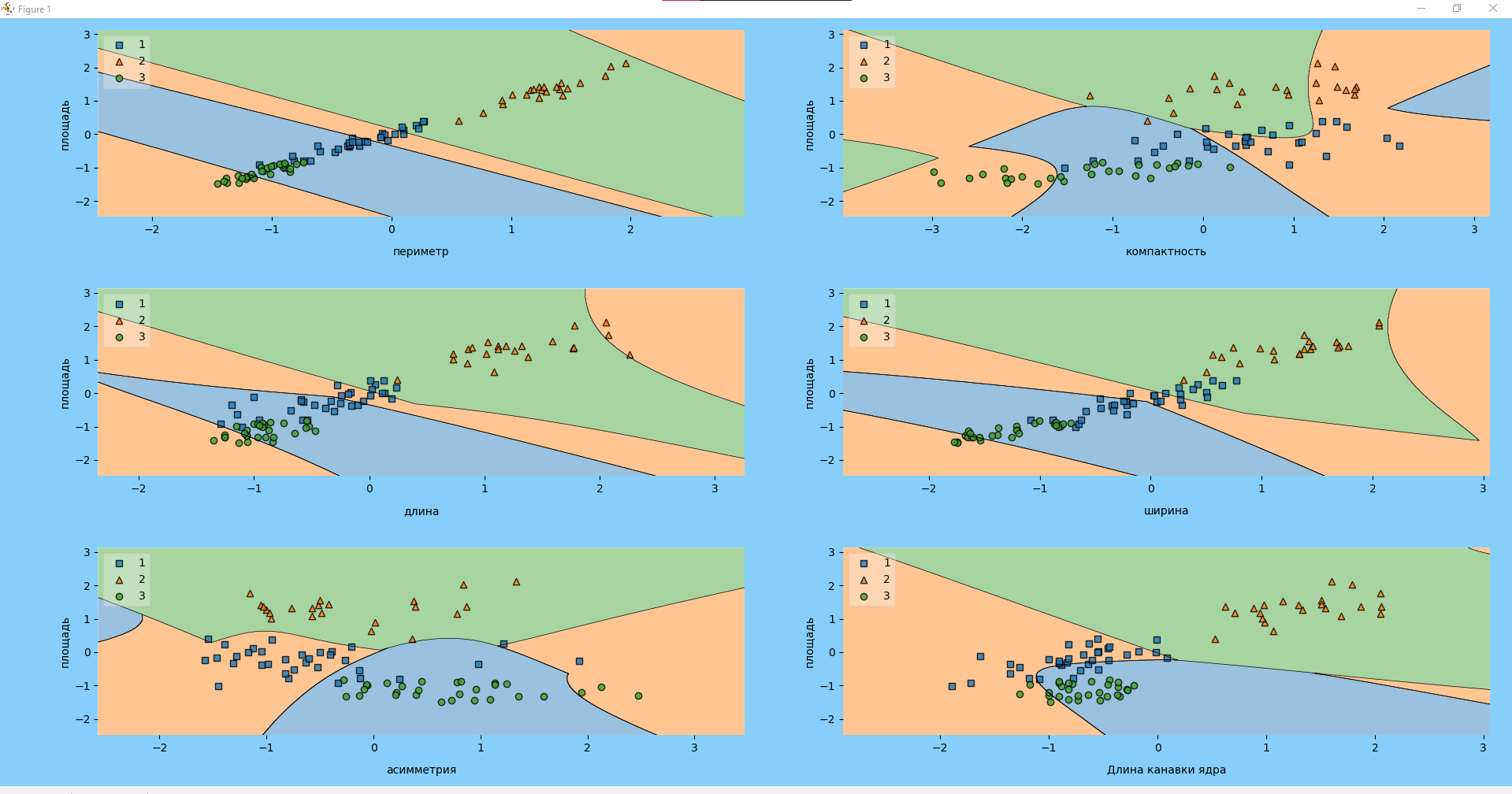


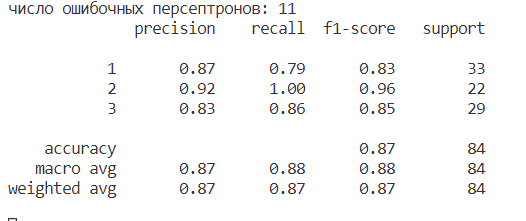
C=1.0, kernel = 'poly'





C=1.0, kernel = 'sigmoid'





Из проделанного исследования можно сделать вывод, что алгоритм linear больше подходит для этих данных

**Код программы**

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix, classification\_report

from sklearn.metrics import recall\_score

from sklearn.svm import SVC

import pandas as pd

from matplotlib.colors import ListedColormap

from mlxtend.plotting import plot\_decision\_regions

url = "C:/Users/fox/Desktop/ii/data/seeds\_dataset.data"

#опорных векторов

names = [

    "площадь",

    "периметр",

    "компактность",

    "длина",

    "ширина",

    "асимметрия",

    "Длина канавки ядра",

    "Сорт"

] #название атрибутов

dataset = pd.read\_csv(url, names=names) #чтение файла с данными

dataset.head()

X = dataset.iloc[:, :-1] #выбор

Y = dataset.iloc[:,7]

#print(dataset.shape)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split( X, Y, test\_size=0.40,  random\_state = 1)

scaler  = StandardScaler()

scaler.fit(X\_train)

X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

Y\_test=Y\_test.to\_numpy()

fig = plt.figure(figsize=(10,10), facecolor='#87CEFA')

for i in range(0,6):

    d = fig.add\_subplot(3,2,i+1)#сетка

    classifier = SVC(C=1.2, kernel = 'rbf', random\_state = 8000, gamma=1 ) #'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'

    classifier.fit(X\_train[:,[i+1,0]], Y\_train)

    plot\_decision\_regions(X\_test[:,[i+1,0]], Y\_test, clf=classifier, legend=2)

    plt.xlabel(dataset.columns[i+1],labelpad=10)            #название оси

    plt.ylabel(dataset.columns[0])                          #название оси

    plt.tight\_layout(h\_pad=3)                               #регулировка растояние между графиками

y\_predict = classifier.predict(X\_test[:,[i+1,0]])

print(recall\_score(Y\_test, y\_predict, average=None))

print('число ошибочных персептронов: %d' % (Y\_test != y\_predict).sum())

pred\_svc2 = classifier.predict(X\_test[:,[i+1,0]])

print(classification\_report(Y\_test, pred\_svc2))

plt.show()