МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра систем автоматизации управления

Дата сдачи на проверку:

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г.

Проверено:

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г.

**Классификация методом опорных векторов (SVM)**

Отчет по лабораторной работе №1  
по дисциплине

«Методы искусственного интеллекта»

Разработал студент гр. ИТб-3303-02-00\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Шешегов М.Г./

(Подпись)

Руководитель ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Родионов К.В./

(Подпись)

Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г.

Киров 2023

Цель работы: изучение алгоритма классификации данных «Метод опорных векторов».

В ходе работы необходимо выполнить следующее задание:

* исследовать параметры Cи и sigma(σ);
* исследовать линейное и гауссово ядро;
* получить графики исследований.

Ход работы:

Задание 1: загрузить данные из файла dataset1.mat, и отобразить на графике.

Код, необходимый для 1 задания, представлен в листинге 1.

График, необходимый для 1 задания, представлен на рисунке 1.

Листинг 1 – Программа для 1 задания

**from labs.labs1 import svm  
from scipy.io import loadmat  
from svm import \*  
import numpy as np  
  
*#task 1*data = loadmat('dataset1.mat')  
  
y = data['y'].astype(np.float64)  
X = data['X']  
  
title = 'Исходные данные'  
visualize\_boundary\_linear(X, y, None, title)**

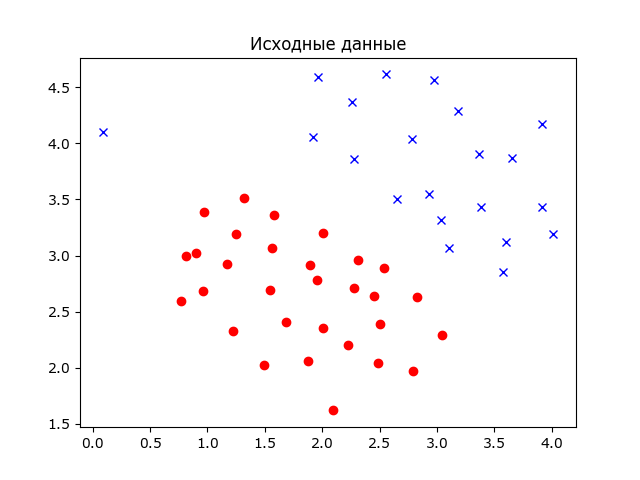


Рисунок 1 – Исходные данные из файла dataset1.mat

Задание 2: обучить классификатор на обучающей выборке и отобразить границу.

Код, необходимый для 2 задания, представлен в листинге 2.

График, необходимый для 2 задания, представлен на рисунке 2.

Листинг 2 – Программа для 2 задания

***# task 2*C = 1  
model = svm.svm\_train(X, y, C, svm.linear\_kernel, 0.001, 20)  
svm.visualize\_boundary\_linear(X, y, model, title='Визуализация гиперплоскости')**

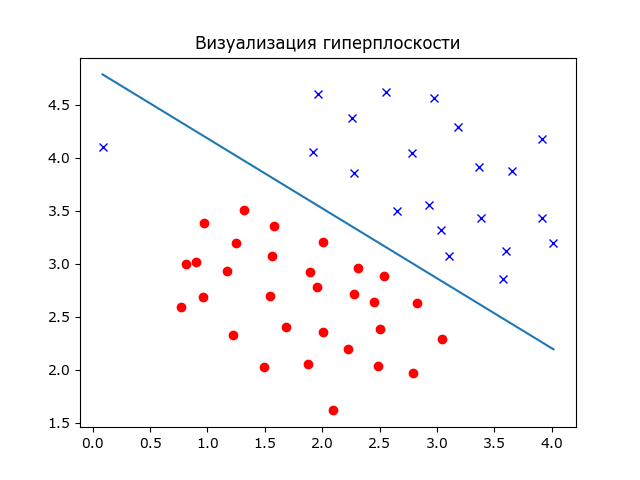


Рисунок 2 – Разделяющая граница при С = 1

Задание 3: выполнить обучение модели с С = 100, привести график.

Код, необходимый для 3 задания, представлен в листинге 3.

График, необходимый для 3 задания, представлен на рисунке 3

Листинг 3 – Программа для 3 задания

***#task 3*C = 100  
  
model = svm.svm\_train(X, y, C, svm.linear\_kernel, 0.001, 20)  
  
svm.visualize\_boundary\_linear(X, y, model, 'Разделяюшая граница при С = 100')**

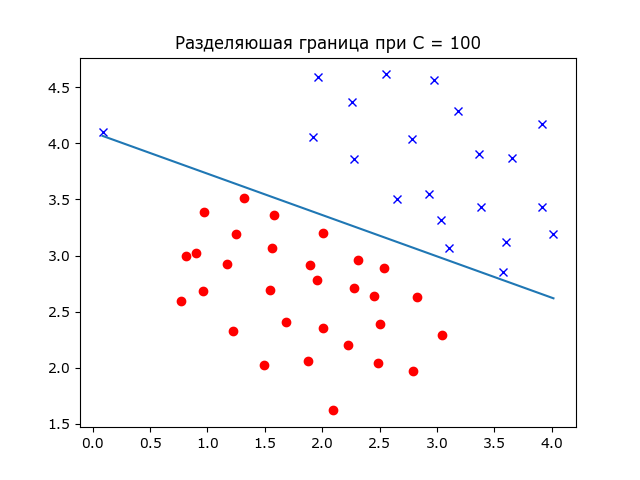


Рисунок 3 – Разделяющая граница при С = 100

Задание 4: реализовать функцию гауссова ядра, построить контурные графики для sigma = 1 и sigma = 3.

Код, необходимый для 4 задания, представлен в листинге 4.

Графики, необходимые для 4 задания, представлены на рисунках 4 и 5.

Листинг 4 – Программа для 4 задания

***# task 4*def gaussian\_kernel(x1, x2, sigma=1.0):  
 return np.exp(-np.sum(np.power(x1 - x2, 2)) / (2 \* sigma\*\*2))  
  
svm.contour(1)  
svm.contour(3)**



Рисунок 4 – Контурный график гауссова ядра при sigma = 1



Рисунок 5 – Контурный график гауссова ядра sigma = 3

Задание 5: загрузить и отобразить данные из файла dataset2.mat.

Код, необходимый для 5 задания, представлен в листинге 5.

График, необходимый для задания 5, представлен на рисунке 6.

Листинг 5 – Программа для 5 задания

***#task 5*data = loadmat('dataset2.mat')  
  
y = data['y'].astype(np.float64)  
X = data['X']  
  
title = ' Данные из файла dataset2.mat'  
visualize\_boundary\_linear(X, y, None, title)**

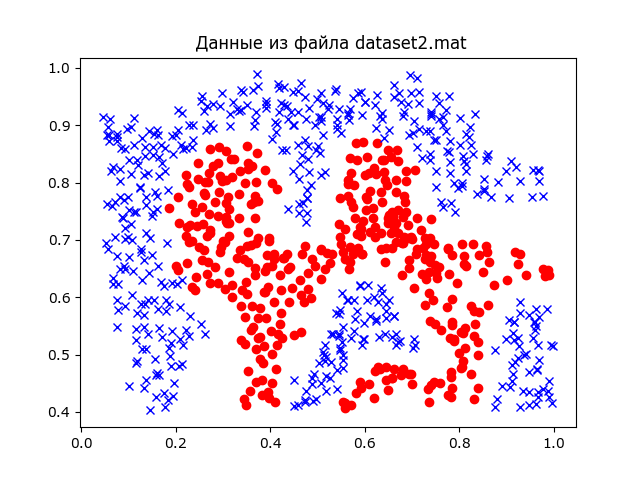


Рисунок 6 – Исходные данные из файла dataset2.mat

Задание 6: запустить обучение модели, отобразить получившуюся границу.

Код, необходимый для 6 задания, представлен в листинге 6.

График, необходимый для задания 7, представлен на рисунке 7.

Листинг 6 – Программа для 6 задания

***#task 6*title = 'Результирующая граница на основе гауссова ядра'  
C =1.0  
sigma =0.1  
gaussian = svm.partial(svm.gaussian\_kernel, sigma=sigma)  
gaussian.\_\_name\_\_ = svm.gaussian\_kernel.\_\_name\_\_  
model = svm.svm\_train(X, y, C, gaussian)  
svm.visualize\_boundary(X, y, model, title)**

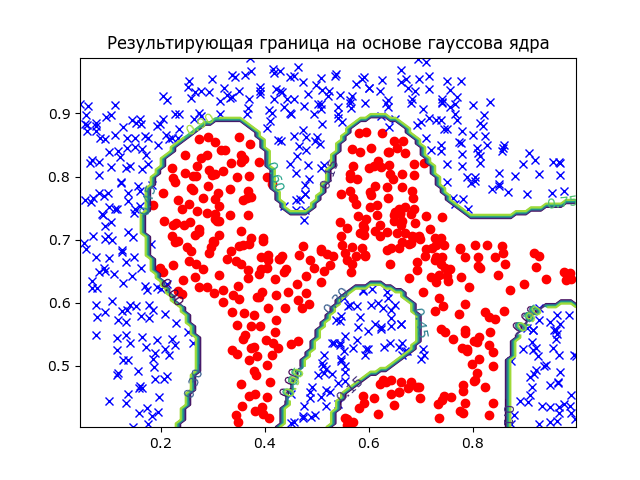


Рисунок 7 – Результирующая граница на основе гауссова ядра

Задание 7: загрузить и отобразить данные из файла dataset3.mat.

Код, необходимый для 7 задания, представлен в листинге 7.

Графики, необходимые для 7 задания, представлены на рисунке 8 и 9.

Листинг 7 – Программа для 7 задания

***#task 7*data = loadmat('dataset3.mat')  
X = data['X']  
y = data['y'].astype(np.float64)  
  
Xval = data['Xval']  
yval = data['yval'].astype(np.float64)  
  
title = 'Обучающая выборка'  
visualize\_boundary\_linear(X, y, None, title)  
  
title = ' Тестовая выборка'  
visualize\_boundary\_linear(Xval, yval, None, title)**

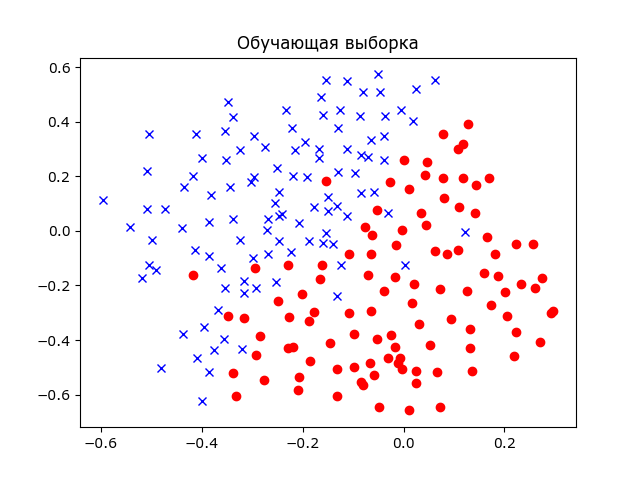


Рисунок 8 – Данные для обучающей выборки из файла dataset3.mat

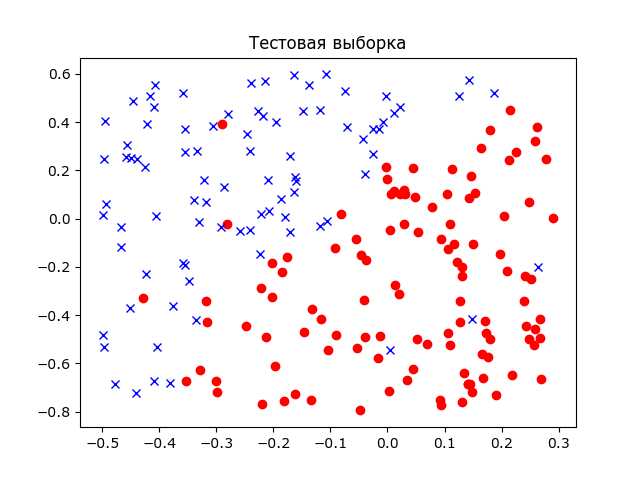


Рисунок 9 – Данные для тестовой выборки из файла dataset3.mat

Задание 8: выполнить обучение модели с неоптимальными параметрами, вывести график.

Код, необходимый для 8 задания, представлен в листинге 8.

График, необходимый для 8 задания, представлен на рисунке 10.

Листинг 8 – Программа для 8 задания

***# task 8*C =1.0  
sigma =0.5  
gaussian = svm.partial(svm.gaussian\_kernel, sigma=sigma)  
gaussian.\_\_name\_\_ = svm.gaussian\_kernel.\_\_name\_\_  
model = svm.svm\_train(X, y, C, gaussian)  
svm.visualize\_boundary(X, y, model,'Модель при неоптимальных параметрах')**

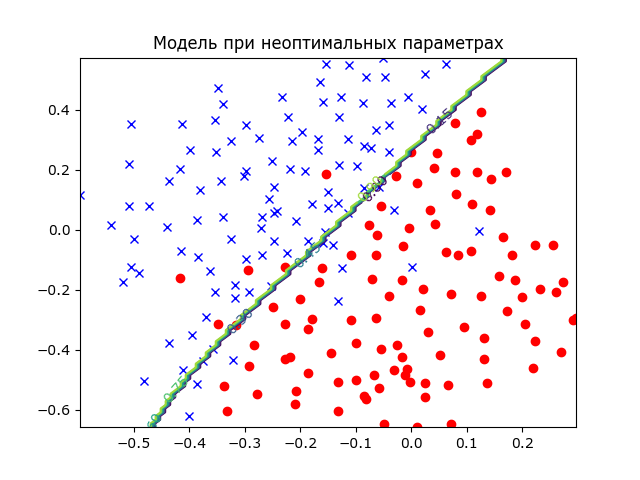


Рисунок 10 – График модели при неоптимальных параметрах

Задание 9: вычисление оптимальных значений Си и sigma.

Код, необходимый для 9 задания, представлен в листинге 9.

График, необходимый для 9 задания, представлен на рисунке 11.

Листинг 9 – Программа для 9 задания

# task 9

best\_model = None

best\_C = None

best\_sigma = None

best\_error = float('inf')

for C in [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]:

for sigma in [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]:

gaussian = svm.partial(svm.gaussian\_kernel, sigma=sigma)

gaussian.\_\_name\_\_ = svm.gaussian\_kernel.\_\_name\_\_

model = svm.svm\_train(X3, y3, C, gaussian)

ypred = svm.svm\_predict(model, Xval)

error = np.mean(ypred != yval.ravel())

if error < best\_error:

best\_error = error

best\_model = model

best\_C = C

best\_sigma = sigma

print("Наилучшие параметры:")

print(f"C = {best\_C}")

print(f"sigma = {best\_sigma}")

print(f"Ошибка = {best\_error}")

svm.visualize\_boundary(X3, y3, best\_model, title='Наилучшая модель (обучающая выборка)')

svm.visualize\_boundary(Xval, yval, best\_model, title='Наилучшая модель (тестовая выборка)')

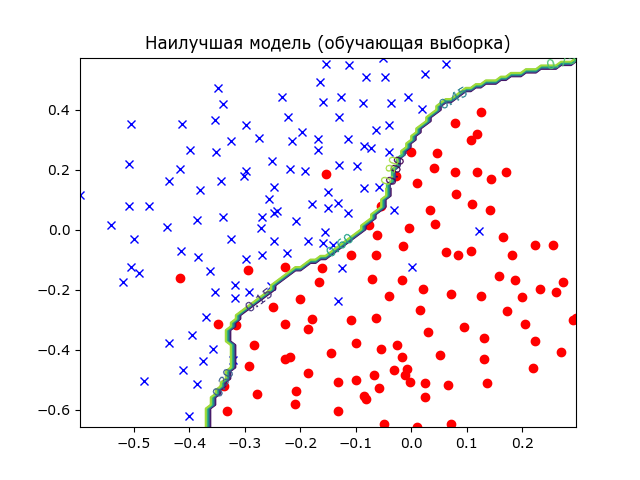


Рисунок 11 – График модели с оптимальными параметрами для обучающей выборки

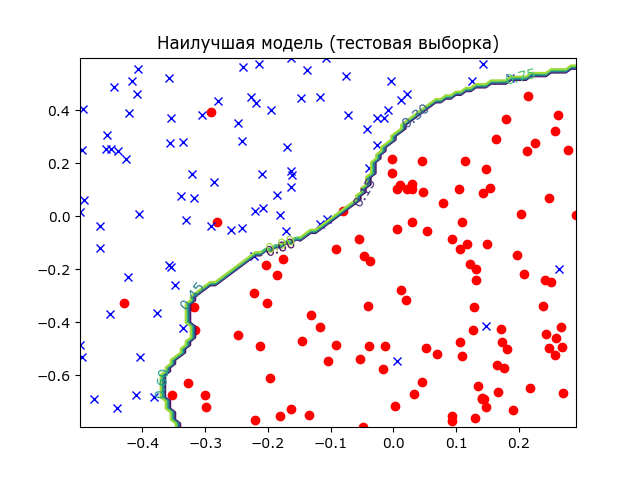


Рисунок 12 – График модели с оптимальными параметрами для тестовой выборки

Вывод: в ходе работы была изучена классификация метом опорных векторов, исследованы параметры Си и sigma, а также линейное и гауссово ядро.