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 электронных писем с помощью SVM**

Отчет по лабораторной работе №2  
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: научится использовать алгоритма классификации данных «Метод опорных векторов» на практике.

В ходе работы необходимо выполнить следующее задание:

* предобработка писем;
* формирование матрицы признаков X.

**Задание 1: загрузка и вывод текста электронного письма.**

**Задание 2: дополнение функции предобработки письма process\_email.**

**Задание 3: преобразование текста письма в вектор признаков.**

**Задание 4: загрузка обучающей выборки и обучение классификатора.**

**Задание 5: оценка точности классификатора на тестовой выборке.**

**Задание 6: определение, какие слова чаще встречаются в письмах со спамом.**

**Задание 7: проверка писем из своего почтового ящика.**

**Результат работы программы представлен на рисунке 1.**

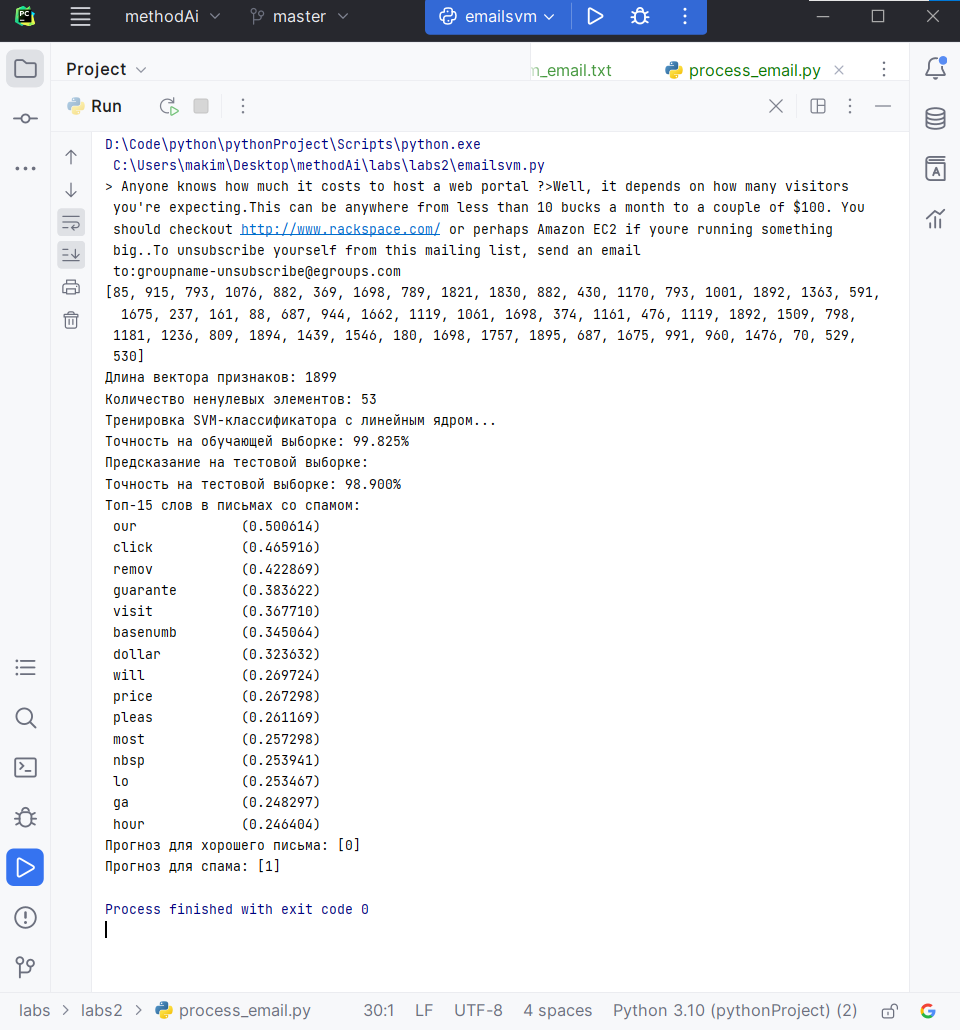


Рисунок 1 － Результат работы программы emailssvm.py

Вывод

В ходе выполнения задания были получены навыки работы с методом опорных векторов (SVM) для классификации электронных писем на спам и не спам. Процесс включал в себя: п**одготовка данных, предобработка текста, обучение модели, оценка точности, анализ топ-слов.**

**Приложение А**

(обязательное)

**Листинг программного кода реализации алгоритма классификации электронных писем с помощью SVM**

import numpy as np  
import scipy.io  
from sklearn import svm  
from collections import OrderedDict  
from process\_email import process\_email  
from process\_email import email\_features  
from process\_email import get\_dictionary  
  
with open('../../Metodichka/lab/lab2/email.txt', 'r') as file:  
 email = file.read().replace('\n', '')  
 print(email)  
*#task2*word\_indices = process\_email(email)  
print(word\_indices)  
*#task3*features = email\_features(word\_indices)  
print(f"Длина вектора признаков: {len(np.array(features))}")  
print("Количество ненулевых элементов: {}".format(sum(np.array(word\_indices) > 0)))  
*#task4*data = scipy.io.loadmat('train.mat')  
X = data['X']  
y = data['y'].flatten()  
print('Тренировка SVM-классификатора с линейным ядром...')  
clf = svm.SVC(C=0.1, kernel='linear', tol=1e-3)  
model = clf.fit(X, y)  
p = model.predict(X)  
*# Оценка точности*accuracy = np.mean(p == y)  
*# Вывод точности классификации на обучающей выборке в процентах*print(f'Точность на обучающей выборке: {accuracy \* 100:.3f}%')  
*#task5*data = scipy.io.loadmat('test.mat')  
X = data['Xtest']  
y = data['ytest'].flatten()  
print('Предсказание на тестовой выборке: ')  
p = model.predict(X)  
*# Оценка точности*accuracy = np.mean(p == y)  
*# Вывод точности классификации на обучающей выборке в процентах*print(f'Точность на тестовой выборке: {accuracy \* 100:.3f}%')  
*#task6*t = sorted(list(enumerate(model.coef\_[0])),key=lambda e: e[1], reverse=True)  
d = OrderedDict(t)  
idx = list(d.keys())  
weight = list(d.values())  
dictionary = get\_dictionary()  
print('Топ-15 слов в письмах со спамом: ')  
for i in range(15):  
 print(' %-15s (%f)' %(dictionary[idx[i]], weight[i]))  
*# вывод: в спам зачастую попадают письма, с рекламными предложениями которые содержат в себе слова которые представлены в консоли  
#task7  
  
# Загрузка и предобработка письма*with open('C:/Users/makim/Desktop/methodAi/labs/labs2/mye-mail/good\_email.txt', 'r') as file:  
 good\_email = file.read().replace('\n', '')  
with open('C:/Users/makim/Desktop/methodAi/labs/labs2/mye-mail/spam\_email.txt', 'r', encoding='utf-8') as file:  
 spam\_email = file.read().replace('\n', '')  
  
  
good\_indices = process\_email(good\_email)  
spam\_indices = process\_email(spam\_email)  
  
*# Создание векторов признаков*good\_features = email\_features(good\_indices)  
spam\_features = email\_features(spam\_indices)  
  
*# Проход через модель*good\_prediction = model.predict(good\_features.reshape(1, -1))  
spam\_prediction = model.predict(spam\_features.reshape(1, -1))  
  
*# Печать результатов*print("Прогноз для хорошего письма:", good\_prediction)  
print("Прогноз для спама:", spam\_prediction)

**Приложение Б**

(обязательное)

**Листинг программного кода реализации предобработки писем**

import numpy as np  
from porterStemmer import porterStemmer  
import re  
  
  
def get\_dictionary():  
 with open('dictionary.txt') as f:  
 dictionary = []  
 for line in f:  
 idx, w = line.split()  
 dictionary.append(w)  
 return np.array(dictionary)  
  
  
def process\_email(email):  
 vocabList = get\_dictionary()  
 word\_indices = []  
  
 *# приведение текста к нижнему регистру* email = email.lower()  
  
 *# удаление html-тегов из текста письма* rx = re.compile('<[^<>]+>|\n')  
 email = rx.sub(' ', email)  
  
 *# числа заменяются на строку 'number'* rx = re.compile('[0-9]+')  
 email = rx.sub('number ', email)  
  
 *# ссылки заменяются на строку 'httpaddr'* rx = re.compile('(http|https)://[^\s]\*')  
 email = rx.sub('httpaddr ', email)  
  
 *# электронные адреса заменяются на строку 'emailaddr'* rx = re.compile('[^\s]+@[^\s]+')  
 email = rx.sub('emailaddr ', email)  
  
 *# значок $ заменяется на строку 'dollar'* rx = re.compile('[$]+')  
 email = rx.sub('dollar ', email)  
  
 *# удаление не буквенно-цифровых символов* rx = re.compile('[^a-zA-Z0-9]')  
 email = rx.sub(' ', email).split()  
  
 for str in email:  
 *# приведение каждого слова к существительному в единственном числе* try:  
 str = porterStemmer(str.strip())  
 except:  
 str = ''  
 continue  
  
 if len(str) < 1:  
 continue  
  
 *# TODO: необходимо в word\_indices добавить индекс слова str из словаря vocabList.* ans = np.where(np.array(vocabList) == str)  
 if len(ans[0]) > 0:  
 word\_indices.append(ans[0][0])  
  
 return word\_indices  
  
  
def email\_features(word\_indices):  
 n = 1899 *# общее число слов в словаре* x = np.zeros(n)  
  
 *# TODO: необходимо установить в 1 элементы массива x,* for i in word\_indices:  
 x[i] = 1  
  
 return x