МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Кафедра «Измерительно-вычислительные комплексы»

«Методы искусственного интеллекта»

Отчёт по лабораторной работе №4

Вариант №6

Выполнил:
студент группы ИСТбд-42
Евтушенко Александр
Проверил:
доцент кафедры ИВК, к.т.н.
Шишкин В.В.

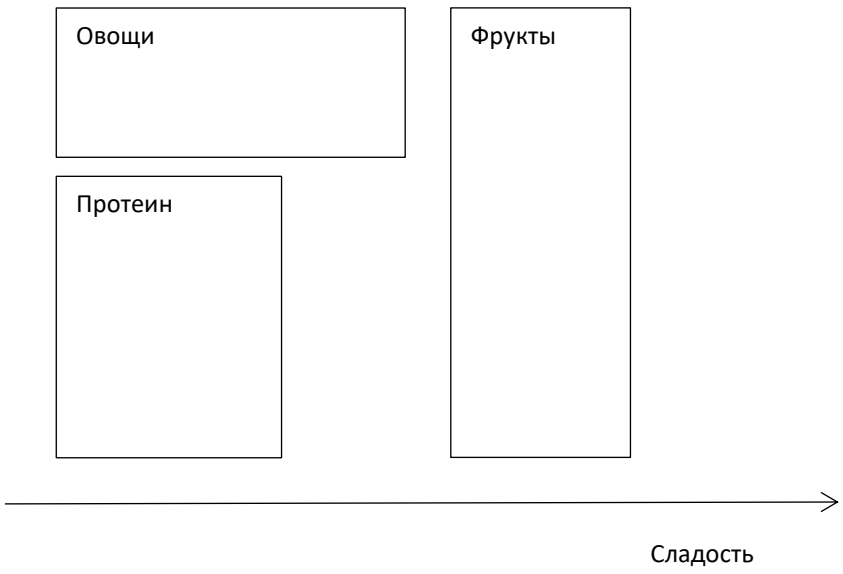
Ульяновск
2022

Задание 1.
“Генерация данных”.

Создать симулированный набор данных и записать его на диск в виде csv файла со следующими параметрами:
- продукт;
- сладость;
- хруст;
-класс.

продукт	сладость	хруст	класс
Яблоко	7	7	Фрукт
салат	2	5	Овощ
бекон	1	2	Протеин
банан	9	1	Фрукт
орехи	1	5	Протеин
рыба	1	1	Протеин
сыр	1	1	Протеин
виноград	8	1	Фрукт
морковь	2	8	Овощ
апельсин	6	1	Фрукт

Подготовить для классификации несколько примеров в соответствии с рисунком.



Результат.

(Часть набора данных)

```
DATA
Входящие данные
[['Продукт', 'Сладость', 'Хруст', 'Класс'], ['Яблоко', '7', '7', '0'], ['Салат', '2', '5', '1'],
```

Задание 2.
“Получение классификаторов”.

Запрограммировать метрический классификатор по методу k-NN. Для проверки решить ту же задачу методом k-NN библиотеки sklearn.

Результат.

(Пример работы алгоритма knn)

```
Классификация для k= 1
0. Классифицируем продукт  Слива
индекс соседа = 8, сосед - Виноград
расстояние [0, 0, 0]
Мы присваиваем класс = 0
0
0
Правильно
1. Классифицируем продукт  Сельдерей
индекс соседа = 3, сосед - Орехи
расстояние [0, 0, 0]
Мы присваиваем класс = 1
0
1
Не правильно
2. Классифицируем продукт  Шницель
```

(Пример работы алгоритма sklearn knn)

```
SKLEARN KNN
Параметры обучающей выборки
[[-0.57935845 -1.25411943]
 [-0.92015754  0.83607962]
 [ 1.12463699 -1.25411943]
 [-0.23855936  1.18444612]
 [ 1.80623517 -0.90575292]
 [-0.92015754  0.48771311]
 [-0.57935845  0.1393466 ]
 [ 1.46543608  0.83607962]
 [-0.23855936  1.18444612]
 [-0.92015754 -1.25411943]]
Параметры тестовой выборки
[[ 0.10223973  1.53281263]
 [-0.23855936  0.1393466 ]
 [ 1.80623517 -0.55738641]
 [ 1.80623517 -1.25411943]
 [ 1.46543608 -1.60248593]
 [-0.57935845  1.53281263]
 [ 2.14703426 -1.25411943]
 [ 1.46543608 -1.25411943]
 [-0.57935845 -1.25411943]
 [-0.57935845 -0.90575292]]
```

Задание 3.
“Классификация”.

Прочитать сгенерированный набор данных. Настроить классификатор. Провести эксперимент по классификации с контролем для подготовленных примеров.

Результат.

(Пример работы алгоритма knn)

```
Классификация для k= 1
0. Классифицируем продукт  Слива
индекс соседа = 8, сосед - Виноград
расстояние [0, 0, 0]
Мы присваиваем класс = 0
0
0
Правильно
1. Классифицируем продукт  Сельдерей
индекс соседа = 3, сосед - Орехи
расстояние [0, 0, 0]
Мы присваиваем класс = 1
0
1
Не правильно
2. Классифицируем продукт  Шницель
```

(Пример работы алгоритма sklearn knn)

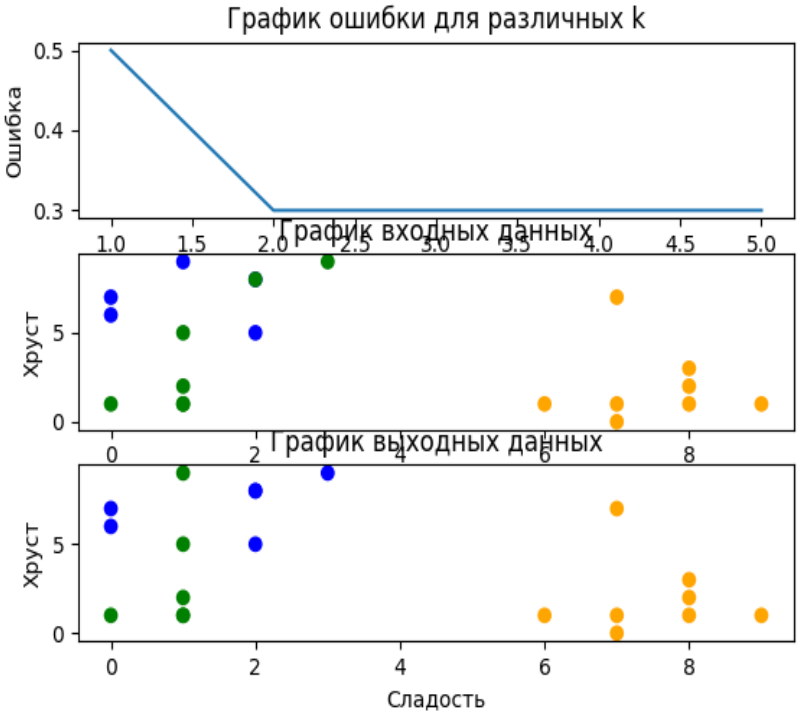
```
Подсчёт ошибки
0 0
1 1
2 2
2 2
2 2
2 2
0 0
1 1
0 0
0 0
1 0
1 1
0 2
0 0
0 1
1 2
0 0
0 1
2 2
2 0
0.3
```

Задание 4.
“Визуализация”.

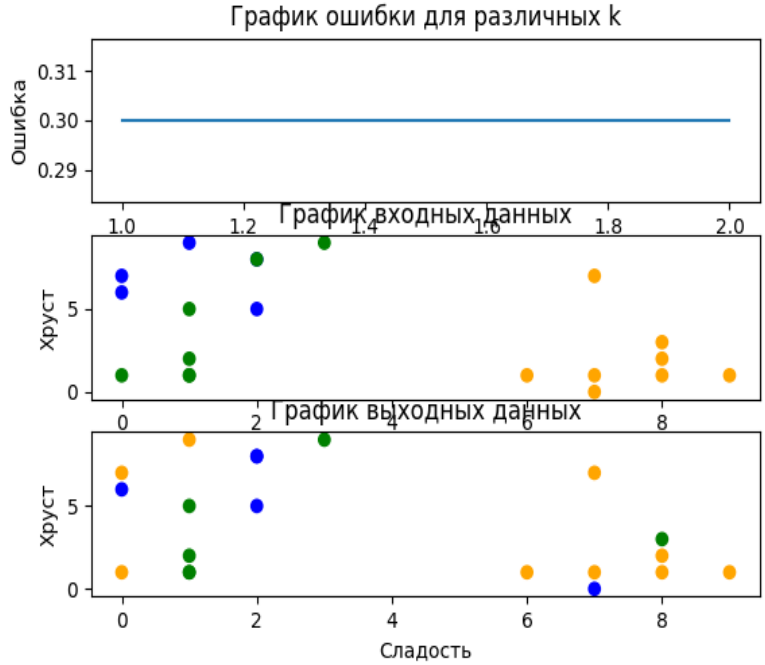
По возможности результаты визуализировать.

Результат.

(Результат работы алгоритма knn)



(Результат работы алгоритма sklearn knn)

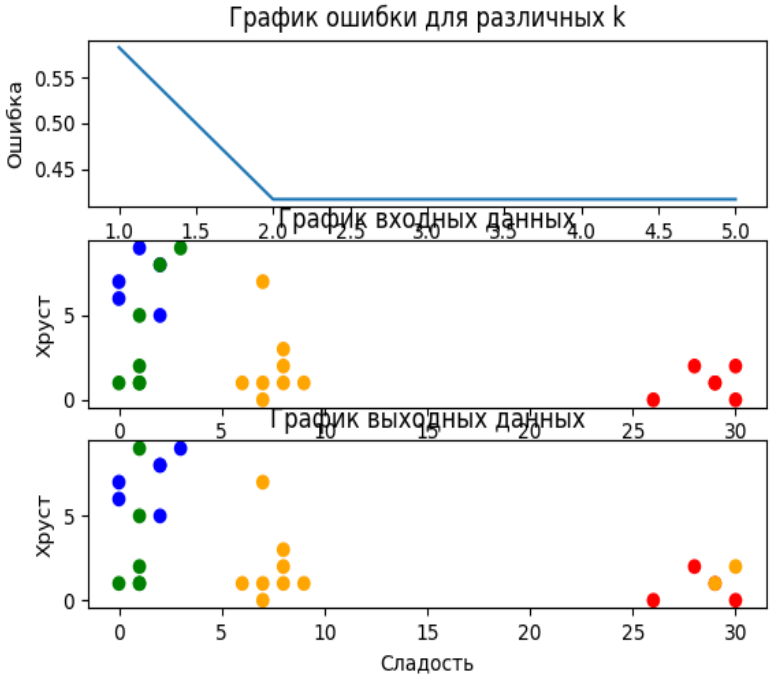


Задание 5.
“Добавление нового класса”.

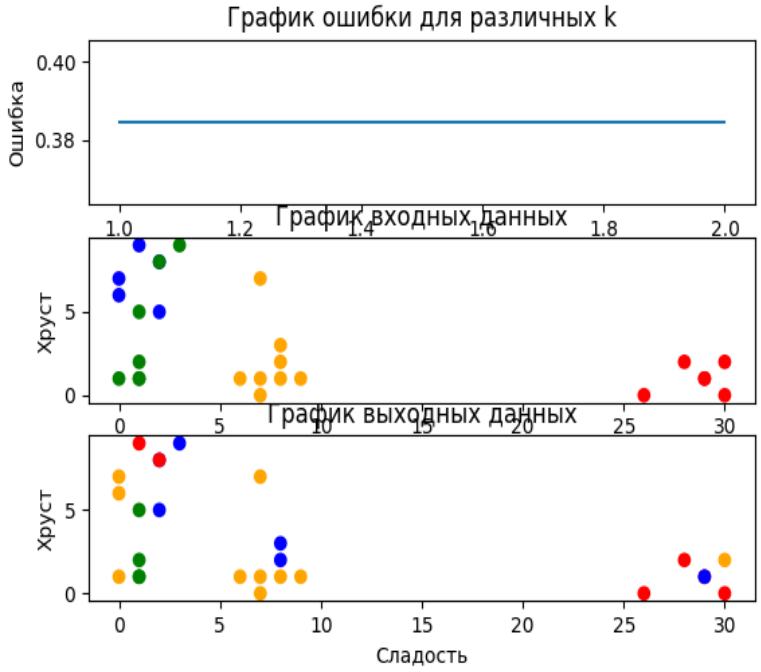
Ввести в набор данных и примеры продукты еще одного класса (возможно изменив набор параметров) и повторить эксперимент.

Результат.

(Результат работы алгоритма knn)



(Результат работы алгоритма sklearn knn)



Код.

```
import csv

import sklearn
import pandas as pds
import numpy as nmp
import pylab
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler

def knn_alg(ldata, tdata, k, w_size, class_num):
    all_data = []
    for i in range(len(ldata)):
        all_data.append(ldata[i])
    for j in range(len(tdata)):
        all_data.append(tdata[j])

    lsize = len(ldata) - 1
    tsize = len(all_data) - 1 - lsize

    k_max = k
    distance = nmp.zeros((tsize, lsize))
```

```
    for i in range(tsize):
        for j in range(lsize):
            distance[i][j] = ((int(all_data[lsize + 1 + i][1]) - int(all_data[j + 1][1])) ** 2
+ (int(all_data[lsize + 1 + i][2]) - int(all_data[j + 1][2])) ** 2) ** (1/2)

    er_k = [0] * k_max
    for k in range(k_max):
        print('\n^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^')
        print('\nКлассификация для k=', k + 1)
        sucsess = 0
        er = [0] * tsize
        classes = [0] * tsize

        for i in range(tsize):
            qwant_dist = [0]*class_num
            print(str(i) + '. ' + 'Классифицируем продукт ', all_data[lsize + i + 1][0])
            tmp = nmp.array(distance[i, :])
            dist_max = max(tmp)

            for j in range(k + 1):
                ind_min = list(tmp).index(min(tmp))
                if (tmp[j] < w_size):
                    qwant_dist[int(all_data[ind_min + 1][3])] += dist_max - tmp[j]
                else:
                    qwant_dist[int(all_data[ind_min + 1][3])] += 0

                tmp[ind_min] = 1000
                max1 = max(qwant_dist)

                print('индекс соседа = ' + str(ind_min) + ', сосед - ' + all_data[ind_min
+ 1][0])
                print('расстояние ' + str(qwant_dist))

            class_ind = list(qwant_dist).index(max1)
            classes[i] = class_ind

            print('Мы присваиваем класс = ' + all_data[lsize + i + 1][3])
            print(classes[i])
            print(all_data[lsize + i + 1][3])
            if (int(classes[i]) == int(all_data[lsize + i + 1][3])):
                print('Правильно')
                sucsess += 1
                er[i] = 0
            else:
                print('Не правильно')
```

```
                er[i] = 1

        er_k[k] = nmp.mean(er)

        print('Ошибка для ' + str(k) + ' соседа')
        print(er_k)

    return er_k,classes

def sklearn_alg(dat,clas,k,tsz):

    X_train, X_test, y_train, y_test = train_test_split(
        dat, clas, test_size=tsz, random_state=0
    )

    scaler = StandardScaler()
    scaler.fit(X_train)

    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)

    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train, y_train)

    predictions = model.predict(X_test)

    print('Параметры обучающей выборки')
    print(X_train)
    print('Параметры тестовой выборки')
    print(X_test)
    print('Классы обучающей выборки')
    print(y_train)
    print('Классы тестовой выборки')
    print(y_test)
    print('Результат')
    print(predictions)

    return X_train, X_test, y_train, y_test, predictions

def graphics_gen(k,er,swt,crc,dta,clrs,cls):
    pylab.subplot(3, 1, 1)
    plt.plot([i for i in range(1, k + 1)], er)
    plt.title('График ошибки для различных k')
    plt.xlabel('k')
    plt.ylabel('Ошибка')
```

```
    colour_list = [clrs[str(i)] for i in cls]

    pylab.subplot(3, 1, 2)
    plt.scatter(swt, crc, c=colour_list)
    plt.title('График входных данных')
    plt.xlabel('Сладость')
    plt.ylabel('Хруст')

    colour_list = [clrs[str(i)] for i in dta]

    pylab.subplot(3, 1, 3)
    plt.scatter(swt, crc, c=colour_list)
    plt.title('График выходных данных')
    plt.xlabel('Сладость')
    plt.ylabel('Хруст')
    plt.show()

if __name__ == '__main__':
    print('\nYYYYYYYYYYYYYYYYYYYYYYYYY')
    print('\nDATA')
    data = [['Продукт', 'Сладость', 'Хруст', 'Класс'],
        ['Яблоко', '7', '7', '0'],
        ['Салат', '2', '5', '1'],
        ['Бекон', '1', '2', '2'],
        ['Орехи', '1', '5', '2'],
        ['Рыба', '1', '1', '2'],
        ['Сыр', '1', '1', '2'],
        ['Бананы', '9', '1', '0'],
        ['Морковь', '2', '8', '1'],
        ['Виноград', '8', '1', '0'],
        ['Апельсин', '6', '1', '0'],
        #test set of 10 (row 11-16)
        ['Слива', '7', '0', '0'],
        ['Сельдерей', '0', '6', '1'],
        ['Шницель', '0', '1', '2'],
        ['Мандарин', '7', '1', '0'],
        ['Капуста', '0', '7', '1'],
        ['Сухарики', '2', '8', '2'],
        ['Ежевика', '8', '2', '0'],
        ['Огурец', '1', '9', '1'],
        ['Гренки', '3', '9', '2'],
        ['Манго', '8', '3', '0'],
        ]
```

```
with open('dat.csv', 'w', encoding='utf8') as f:
    writer = csv.writer(f, lineterminator="\r")
    for row in data:
        writer.writerow(row)

print('Входящие данные')
print(data)

#knn

print('\nYYYYYYYYYYYYYYYYYYYYYYYYY')
print('\nKNN')

k_max=5

window=7

er_k , classes = knn_alg(data[0:11],data[11:],k_max,window,3)

dataset = pds.read_csv("dat.csv")

start_data = dataset[:10]['Класс']

s1 = pds.Series(classes)
start_data = pds.concat([start_data, s1])

sweet = dataset['Сладость']
crunch = dataset['Хруст']

colours = {'0': 'orange', '1': 'blue', '2': 'green'}

classes_info = dataset['Класс']

print(start_data)
print(sweet)
graphics_gen(k_max,er_k,sweet,crunch,start_data,colours,classes_info)

#sklearn

print('\nYYYYYYYYYYYYYYYYYYYYYYYYY')
print('\nSKLEARN KNN')

k_max = 2

my_dataset = pds.read_csv('dat.csv')
```

```
sweetness=my_dataset['Сладость']
crunch=my_dataset['Хруст']

values=nmp.array(list(zip(sweetness, crunch)), dtype=nmp.float64)

classes=my_dataset['Класс']

test_size=0.5

X_train, X_test, y_train, y_test, predictions = sklearn_alg(values,classes,k_max,test_size)

colours = {'0': 'orange', '1': 'blue', '2': 'green'}

classes_info = my_dataset['Класс']

start_data = my_dataset[:10]['Класс']

s1 = nmp.concatenate((y_train,y_test), axis=0)

s1 = pds.Series(s1)
predictions = pds.Series(predictions)
start_data = pds.Series(start_data)
start_data=pds.concat([start_data, predictions])

er=0;
ct=0;

truthClasses=pds.Series(my_dataset['Класс'])
testClasses=pds.concat([pds.Series(my_dataset[:10]['Класс']) ,predictions])

print('Подсчёт ошибки')
for i in testClasses:
    print(str(i)+' '+str(truthClasses[ct]))

    if(i==truthClasses[ct]):
        er+=0
    else:
        er+=1
    ct+=1

er=er/ct
print(er)

er_k = []
```

```
for i in range(1, k_max + 1):
    er_k.append(er)

graphics_gen(k_max, er_k, sweet, crunch, start_data, colours, classes_info)

#add new data
print('\nYYYYYYYYYYYYYYYYYYYYYYY')
print('\nNEW DATA')

new_data = data[0:11]
new_data.append(['Мёд', '30', '0', '3'])
new_data.append(['Хворост', '28', '2', '3'])
new_data.append(['Зефир', '29', '1', '3'])
new_data.append(['Мармелад', '26', '0', '3'])


new_data = new_data + data[11:]
new_data.append(['Фруктовая палочка', '30', '2', '3'])
new_data.append(['Маршмеллоу', '29', '1', '3'])

print('Новые данные')
print(new_data)

with open('dat.csv', 'w', encoding='utf8') as f:
    writer = csv.writer(f, lineterminator="\r")
    for row in new_data:
        writer.writerow(row)

#knn with new data

print('\nYYYYYYYYYYYYYYYYYYYYYYY')
print('\nKNN WITH NEW DATA')

k_max = 5

window = 7

er_k, classes = knn_alg(new_data[0:15], new_data[15:], k_max, window, 4)

dataset = pds.read_csv("dat.csv")

start_data = dataset[:14]['Класс']

s1 = pds.Series(classes)
```

```
    start_data = pds.concat([start_data, s1])

    sweet = dataset['Сладость']
    crunch = dataset['Хруст']

    colours = {'0': 'orange', '1': 'blue', '2': 'green', '3':'red'}

    classes_info = dataset['Класс']

    graphics_gen(k_max, er_k, sweet, crunch, start_data, colours, classes_info)

    #sklearn with new data

    print('\nYYYYYYYYYYYYYYYYYYYYYYYYY')
    print('\nSKLEARN KNN WITH NEW DATA')

    k_max = 2

    my_dataset = pds.read_csv('dat.csv')
    sweetness = my_dataset['Сладость']
    crunch = my_dataset['Хруст']

    values = nmp.array(list(zip(sweetness, crunch)), dtype=nmp.float64)

    classes = my_dataset['Класс']

    test_size = 0.461

    X_train, X_test, y_train, y_test, predictions = sklearn_alg(values, classes,
k_max, test_size)

    colours = {'0': 'orange', '1': 'blue', '2': 'green', '3':'red'}

    classes_info = my_dataset['Класс']

    start_data = my_dataset[:14]['Класс']

    s1 = nmp.concatenate((y_train, y_test), axis=0)

    s1 = pds.Series(s1)
    predictions = pds.Series(predictions)
    start_data = pds.Series(start_data)
    start_data = pds.concat([start_data, predictions])

    er = 0;
```

```
ct = 0;

truthClasses = pds.Series(my_dataset['Класс'])
testClasses = pds.concat([pds.Series(my_dataset[:14]['Класс']), predictions])

print('Подсчёт ошибки')
for i in testClasses:
    print(str(i) + ' ' + str(truthClasses[ct]))

    if (i == truthClasses[ct]):
        er += 0
    else:
        er += 1
    ct += 1

er = er / ct
print(er)

er_k = []

for i in range(1, k_max + 1):
    er_k.append(er)

graphics_gen(k_max, er_k, sweet, crunch, start_data, colours, classes_info)
```