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 №15

Выполнила:
студентка группы ИСТбд-42
Саушкина Наталья
Проверил:
доцент кафедры ИВК, к.т.н.
Шишкин В.В.

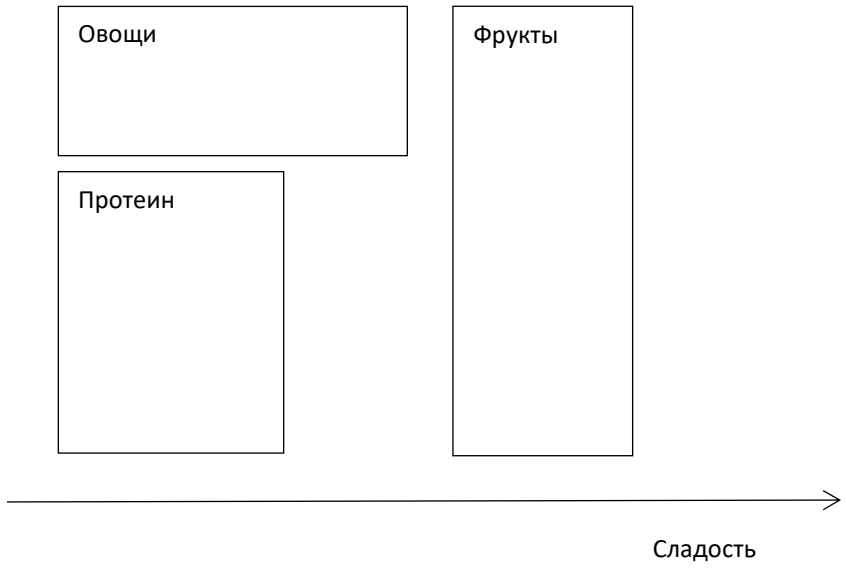
Ульяновск
2022

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:

```
Data
[['продукт', 'сладость', 'хруст', 'класс'], ['Apple', '7', '7', '0'],
['Salad', '2', '25', '1'],
['Bacon', '1', '2', '2'],
['Nuts', '1', '5', '2'],
['Fish', '1', '1', '2'],
['Cheese', '1', '1', '2'],
['Banana', '9', '1', '0'],
['Carrot', '2', '29', '1'],
['Grape', '8', '1', '0'],
['Orange', '6', '1', '0'],
['Kiwifruit', '8', '1', '0'],
['Potato', '1', '27', '1'],
['Quark', '1', '4', '2'],
['Watermelon', '10', '5', '0'],
['Bean', '1', '5', '2'],
['Cabbage', '2', '30', '1'],
['Cucumber', '1', '19', '1'],
['Garnet', '7', '2', '0'],
['Pumpkin', '2', '28', '1'],
['Meat', '1', '1', '2']]
```

Задание 2.

“Получение классификаторов”.

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

Результат.

```
классификация для k = 1
0. классификация Kiwifruit
индекс соседа = 8, сосед - Grape
qwant_dist[0, 0, 0]
класс классифицируемого элемента = 0
0
0
совпал
1. классификация Potato
индекс соседа = 1, сосед - Salad
qwant_dist[0, 0, 0]
класс классифицируемого элемента = 1
0
1
не совпал
2. классификация Quark
индекс соседа = 3, сосед - Nuts
qwant_dist[0, 0, 0]
класс классифицируемого элемента = 2
0
2
не совпал
3. классификация Watermelon
индекс соседа = 0, сосед - Apple
qwant_dist[21.69267000588305, 0, 0]
класс классифицируемого элемента = 0
0
0
совпал
```

(Пример работы алгоритма sklearn knn)

```
параметры обучающей выборки
[[-0.60547036 -1.03112822]
[-0.60547036 -0.68159323]
[ 1.7232618 -1.03112822]
[-0.13972393 1.41561671]
[-0.60547036 0.54177924]
[-0.60547036 1.24084922]
[-0.60547036 -0.68159323]
[ 2.18900823 -0.50682574]
[-0.13972393 1.50300046]
[-0.60547036 -0.76897698]]
параметры тестовой выборки
[[-0.13972393 1.32823297]
[-0.13972393 1.06608172]
[-0.60547036 -1.03112822]
[ 2.65475467 -1.03112822]
[ 2.65475467 -1.03112822]
[ 2.18900823 -0.94374448]
[ 3.1205011 -1.03112822]
[ 3.58624753 -0.68159323]
[-0.60547036 -1.03112822]
[-0.60547036 -0.94374448]]
классы обучающей выборки
5 2
14 2
9 0
```

Задание 3.
“Классификация”.

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

Результат.

(Пример работы алгоритма knn)

```
7. классификация Garnet
индекс соседа = 8, сосед - Grape
qwant_dist[0, 0, 0, 0]
индекс соседа = 9, сосед - Orange
qwant_dist[0, 0, 0, 0]
индекс соседа = 6, сосед - Banana
qwant_dist[0, 0, 0, 0]
класс классифицируемого элемента = 0
0
0
совпал
8. классификация Pumpkin
индекс соседа = 7, сосед - Carrot
qwant_dist[0, 0, 0, 0]
индекс соседа = 1, сосед - Salad
qwant_dist[0, 0, 0, 0]
индекс соседа = 0, сосед - Apple
qwant_dist[0, 0, 0, 0]
класс классифицируемого элемента = 1
0
1
не совпал
9. классификация Meat
индекс соседа = 4, сосед - Fish
qwant_dist[0, 0, 0, 0]
индекс соседа = 5, сосед - Cheese
qwant_dist[0, 0, 0, 0]
индекс соседа = 2, сосед - Bacon
qwant_dist[0, 0, 28.017236257093817, 0]
класс классифицируемого элемента = 2
2
2
совпал
```

(Пример работы алгоритма sklearn knn)

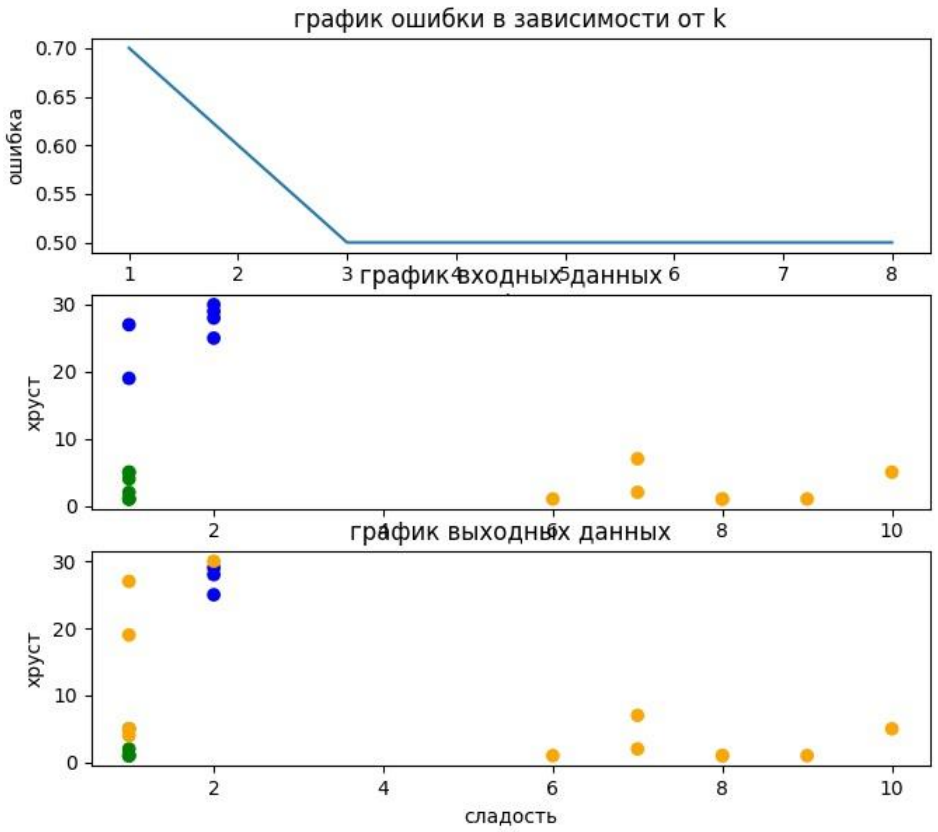
```
предсказания
[1 1 2 2 2 2 2 2 2]
определение ошибки
0 0
1 1
2 2
2 2
2 2
2 2
0 0
1 1
0 0
0 0
1 0
1 1
2 2
2 0
2 2
2 1
2 1
2 0
2 1
2 2
0.3
```

Задание 4.
“Визуализация”.

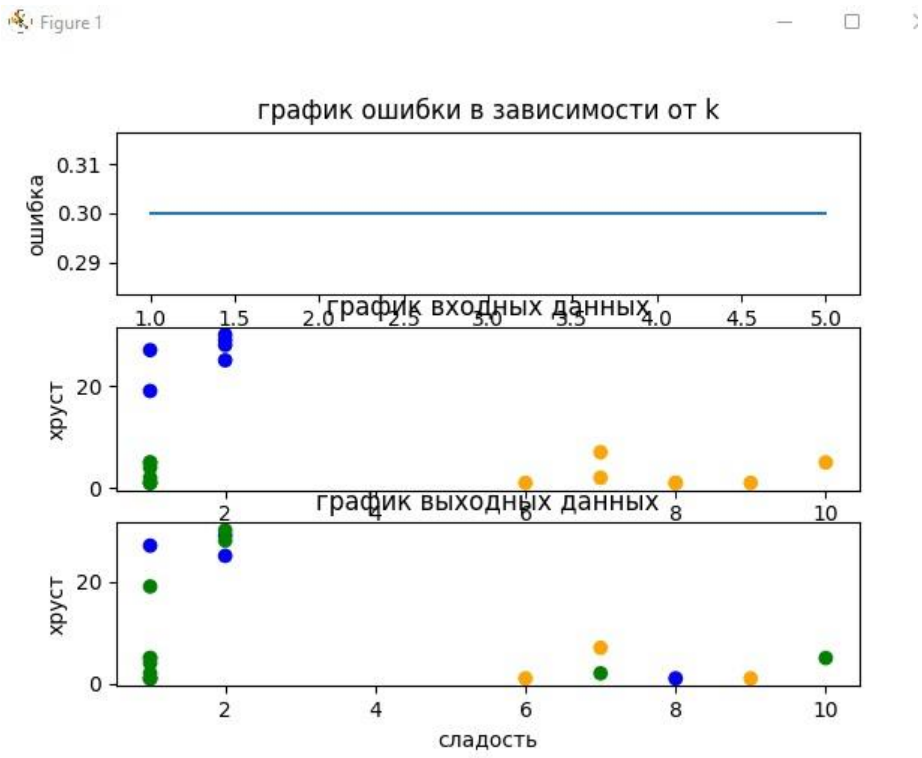
По возможности результаты визуализировать.

Результат.

(Результат работы алгоритма knn)



(Результат работы алгоритма sklearn knn)



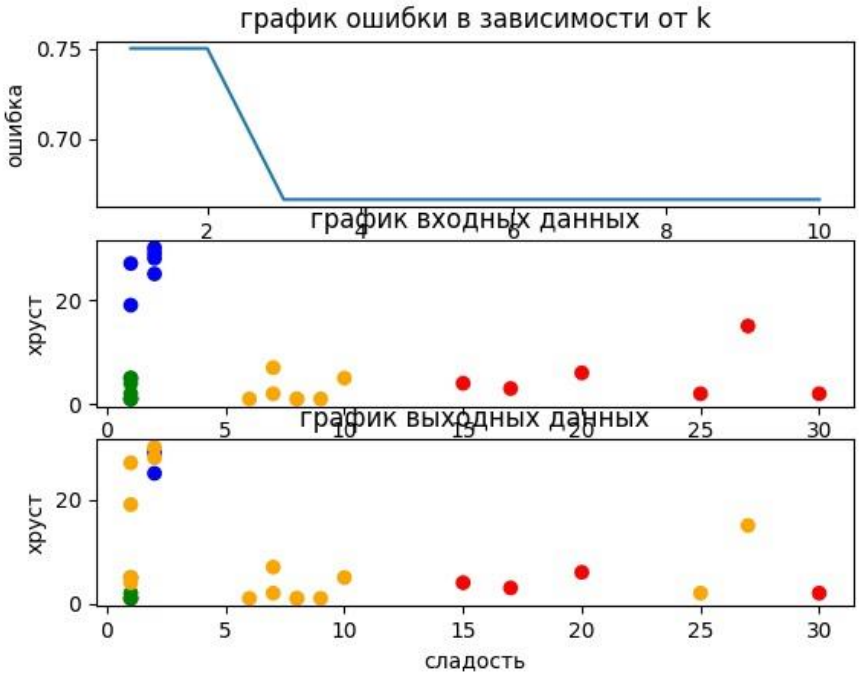
Задание 5.

“Добавление нового класса”.

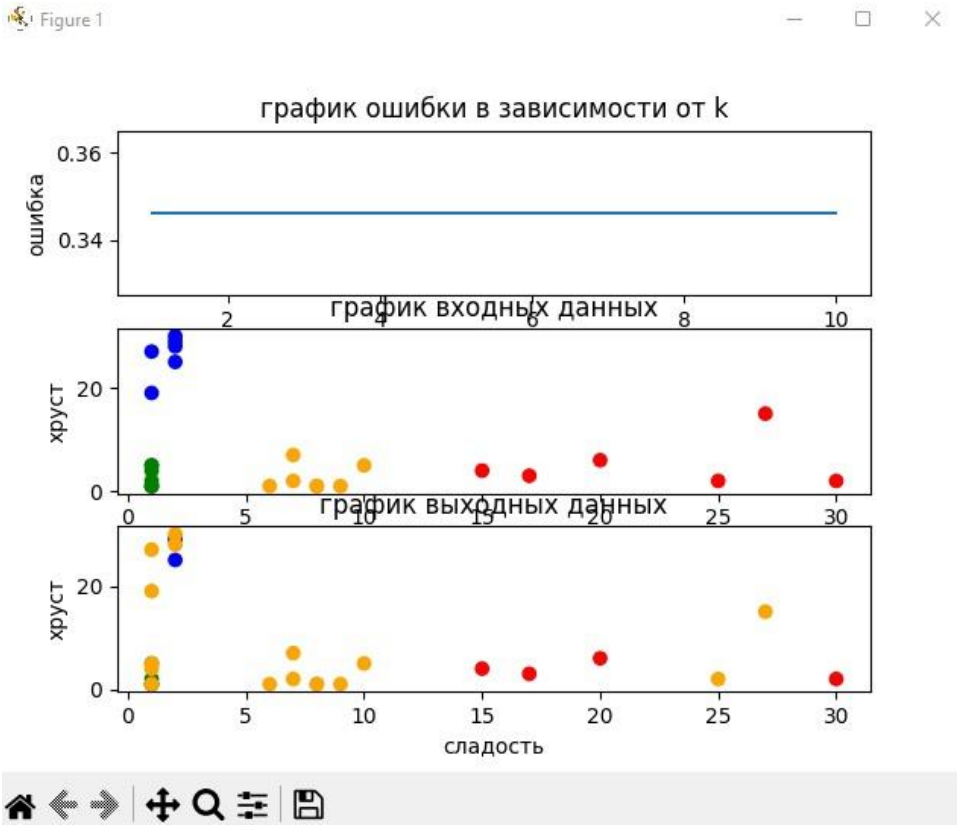
Ввести в набор данных и примеры продукты еще одного класса (возможно изменив набор параметров) и повторить эксперимент.

Результат.

(Результат работы алгоритма knn)



(Результат работы алгоритма sklearn knn)



Листинг программы.

```
import csv

import pandas as pd
import numpy as np
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 print_result(k_max,er_k,sweet,crunch,start_data,colours,classes_info):
    pylab.subplot(3, 1, 1)
    plt.plot([i for i in range(1, k_max + 1)], er_k)
    plt.title('график ошибки в зависимости от k')
    plt.xlabel('k')
    plt.ylabel('ошибка')

    colour_list = [colours[str(i)] for i in classes_info]

    pylab.subplot(3, 1, 2)
    plt.scatter(sweet, crunch, c=colour_list)
    plt.title('график входных данных')
    plt.xlabel('сладость')
    plt.ylabel('хруст')
```

```
    colour_list = [colours[str(i)] for i in start_data]

    pylab.subplot(3, 1, 3)
    plt.scatter(sweet, crunch, c=colour_list)
    plt.title('график выходных данных')
    plt.xlabel('сладость')
    plt.ylabel('хруст')
    plt.show()

def ground(xt1, xt2, xi1, xi2):
    return ((xt1 - xi1) ** 2 + (xt2 - xi2) ** 2) ** (1/2)

def knn(train, test, k_in, w_size, kind_number):
    data = []
    for i in range(len(train)):
        data.append(train[i])
    for j in range(len(test)):
        data.append(test[j])

    train_size = len(train) - 1
    test_size = len(data) - 1 - train_size

    k_max = k_in
    new_dist = np.zeros((test_size, train_size))

    for i in range(test_size):
        for j in range(train_size):
            new_dist[i][j] = ground(int(data[train_size + 1 + i][1]),
                                    int(data[train_size + 1 + i][2]), int(data[j + 1][1]),
                                    int(data[j + 1][2]))

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

        for i in range(test_size):
            qwant_dist = [0]*kind_number
            print(str(i) + '. ' + 'классификация ', data[train_size + i + 1][0])
            tmp = np.array(new_dist[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(data[ind_min + 1][3])] += dist_max - tmp[j]
                else:
                    qwant_dist[int(data[ind_min + 1][3])] += 0

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

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

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

        print('класс классифицируемого элемента = ' + data[train_size + i + 1][3])
        print(classes[i])
        print(data[train_size + i + 1][3])
        if (int(classes[i]) == int(data[train_size + i + 1][3])):
            print('совпал')
            sucsess += 1
            er[i] = 0
        else:
            print('не совпал')
            er[i] = 1

    er_k[k] = np.mean(er)

    print('значение ошибки для ' + str(k) + ' соседа')
    print(er_k)

  return er_k,classes

def knn_sklearn(values,classes,k,test_sz):

  X_train, X_test, y_train, y_test = train_test_split(
    values, classes, test_size=test_sz, 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('предсказания')
  print(predictions)
```

```
    return X_train, X_test, y_train, y_test, predictions

if __name__ == '__main__':
    data = [['продукт', 'сладость', 'хруст', 'класс'],
        ['Apple', '7', '7', '0'],
        ['Salad', '2', '25', '1'],
        ['Bacon', '1', '2', '2'],
        ['Nuts', '1', '5', '2'],
        ['Fish', '1', '1', '2'],
        ['Cheese', '1', '1', '2'],
        ['Banana', '9', '1', '0'],
        ['Carrot', '2', '29', '1'],
        ['Grape', '8', '1', '0'],
        ['Orange', '6', '1', '0'],
        ['Kiwifruit', '8', '1', '0'],
        ['Potato', '1', '27', '1'],
        ['Quark', '1', '4', '2'],
        ['Watermelon', '10', '5', '0'],
        ['Bean', '1', '5', '2'],
        ['Cabbage', '2', '30', '1'],
        ['Cucumber', '1', '19', '1'],
        ['Garnet', '7', '2', '0'],
        ['Pumpkin', '2', '28', '1'],
        ['Meat', '1', '1', '2'],
        ]

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

    print('Data')
    print(data)

    #knn
    print('knn with old data')

    k_max=8

    window=4

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

    dataset = pd.read_csv("data.csv")

    start_data = dataset[:10]['класс']

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

    sweet = dataset['сладость']
    crunch = dataset['хруст']

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

```
classes_info = dataset['класс']

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

print('sklearn knn with old data')

k_max = 5

my_dataset = pd.read_csv('data.csv')
sweetness=my_dataset['сладость']
crunch=my_dataset['хруст']

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

classes=my_dataset['класс']

test_size=0.5

X_train, X_test, y_train, y_test, predictions = knn_sklearn(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 = np.concatenate((y_train,y_test), axis=0)

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

er=0;
ct=0;

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

print('определение 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)

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

print('knn with new data')

new_data = data[0:11]
new_data.append(['Cookies', '15', '4', '3'])
new_data.append(['Halva', '20', '6', '3'])
new_data.append(['Gingerbreads', '17', '3', '3'])
new_data.append(['Cake', '30', '2', '3'])


new_data = new_data + data[11:]
new_data.append(['Marshmallow', '25', '2', '3'])
new_data.append(['Candy', '27', '15', '3'])

print('New data')
print(new_data)

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


k_max = 10

window = 2

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

dataset = pd.read_csv("data.csv")

start_data = dataset[:14]['класс']

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

sweet = dataset['сладость']
crunch = dataset['хруст']

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

classes_info = dataset['класс']

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

print('sklearn knn with new data')

k_max = 10
```

```
my_dataset = pd.read_csv('data.csv')
sweetness = my_dataset['сладость']
crunch = my_dataset['хруст']

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

classes = my_dataset['класс']

test_size = 0.461

X_train, X_test, y_train, y_test, predictions = knn_sklearn(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 = np.concatenate((y_train, y_test), axis=0)

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

er = 0;
ct = 0;

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

print('определение 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)

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