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 №7

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

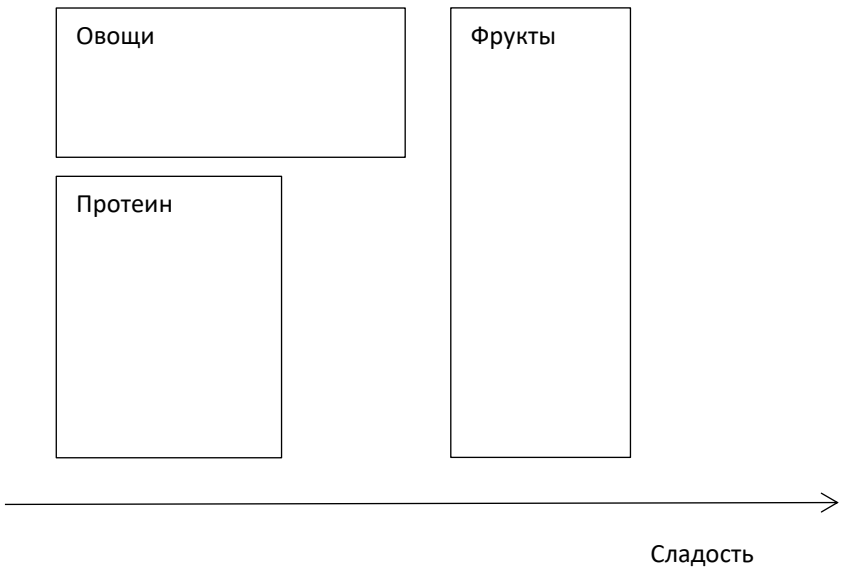
Ульяновск
2022

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
[['Продукт', 'Сладость', 'Хруст', 'Класс'], ['Apple', '7', '7', '0'], ['Salad', '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. Классификация  Strawberry
индекс соседа = 6, сосед - Banana
qwant_dist[0, 0, 0]
Класс классифицируемого элемента = 0
0
0
Совпал
1. Классификация  Lettuce
индекс соседа = 7, сосед - Carrot
qwant_dist[0, 0, 0]
Класс классифицируемого элемента = 1
0
1
не совпал
2. Классификация  Shashlik
индекс соседа = 4, сосед - Fish
qwant_dist[0, 0, 0]
Класс классифицируемого элемента = 2
0
2
не совпал
3. Классификация  Pear
индекс соседа = 9, сосед - Orange
qwant_dist[0, 0, 0]
Класс классифицируемого элемента = 0
0
0
Совпал
```

(Пример работы алгоритма sklearn knn)

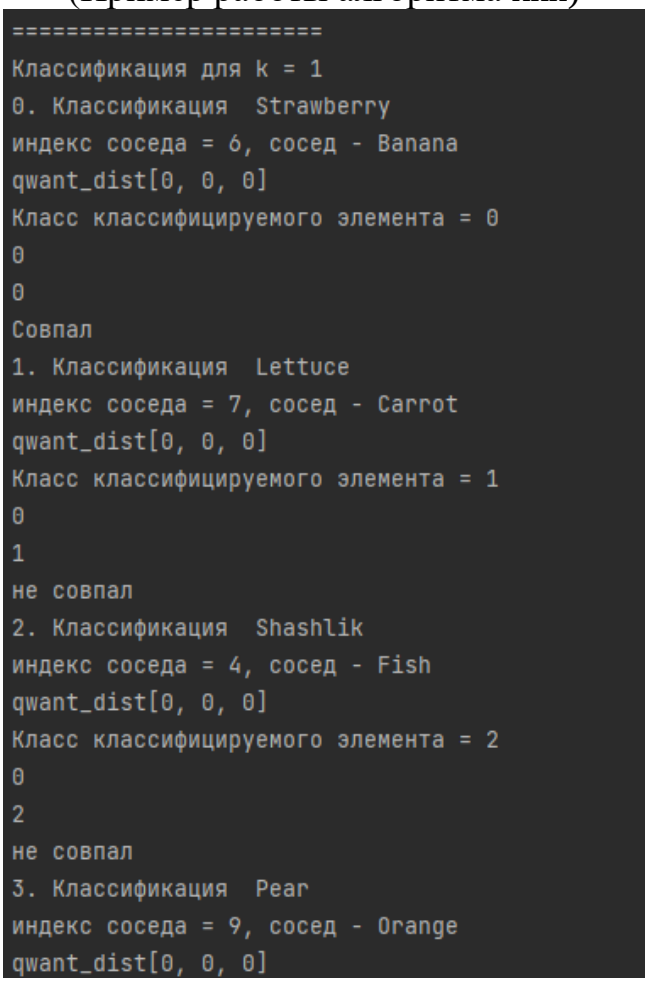
```
Параметры обучающей выборки
[[-0.91520863 -1.36176581]
 [-0.91520863 -0.12379689]
 [ 0.7190925  -1.36176581]
 [-0.58834841  0.8046798 ]
 [ 2.0265334   1.11417203]
 [-0.26148818  0.49518757]
 [-0.91520863 -0.12379689]
 [ 1.04595272  0.49518757]
 [ 0.7190925   1.42366426]
 [-0.91520863 -1.36176581]]
Параметры тестовой выборки
[[-0.26148818 -0.43328912]
 [-0.58834841 -0.12379689]
 [ 2.0265334   0.49518757]
 [ 1.37281295 -1.36176581]
 [ 1.69967317 -1.36176581]
 [ 1.37281295  0.18569534]
 [ 1.69967317 -1.36176581]
 [ 0.39223227 -0.74278135]
 [-0.91520863 -1.36176581]
 [-0.91520863 -1.05227358]]
Классы обучающей выборки
5     2
14    1
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)



```
======================
Классификация для k = 1
0. Классификация  Strawberry
индекс соседа = 6, сосед - Banana
qwant_dist[0, 0, 0]
Класс классифицируемого элемента = 0
0
0
Совпал
1. Классификация  Lettuce
индекс соседа = 7, сосед - Carrot
qwant_dist[0, 0, 0]
Класс классифицируемого элемента = 1
0
1
не совпал
2. Классификация  Shashlik
индекс соседа = 4, сосед - Fish
qwant_dist[0, 0, 0]
Класс классифицируемого элемента = 2
0
2
не совпал
3. Классификация  Pear
индекс соседа = 9, сосед - Orange
qwant_dist[0, 0, 0]
```

(Пример работы алгоритма sklearn knn)

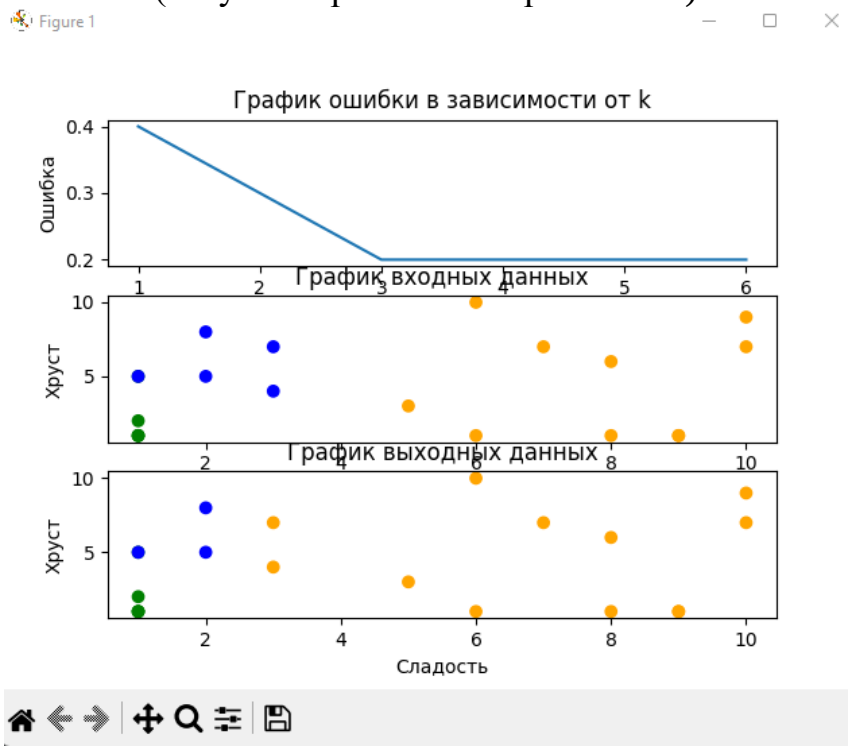
```
Предсказания
[2 1 0 0 0 0 0 0 2 2]
Подсчёт ошибки
0 0
1 1
2 2
2 2
2 2
2 2
0 0
1 1
0 0
0 0
2 0
1 1
0 2
0 0
0 1
0 0
0 0
0 0
2 1
2 0
0.25
```

Задание 4.
“Визуализация”.

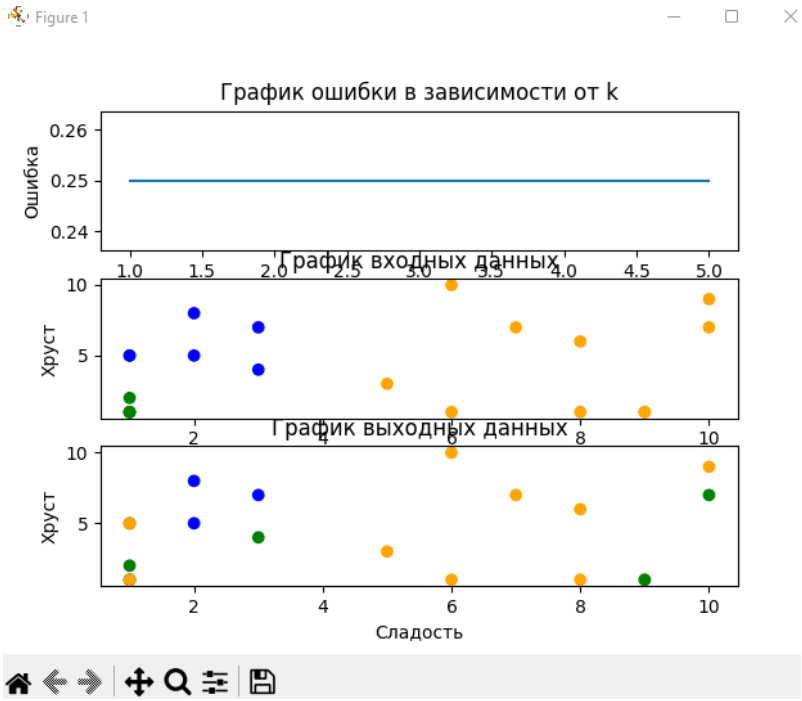
По возможности результаты визуализировать.

Результат.

(Результат работы алгоритма knn)



(Результат работы алгоритма sklearn knn)

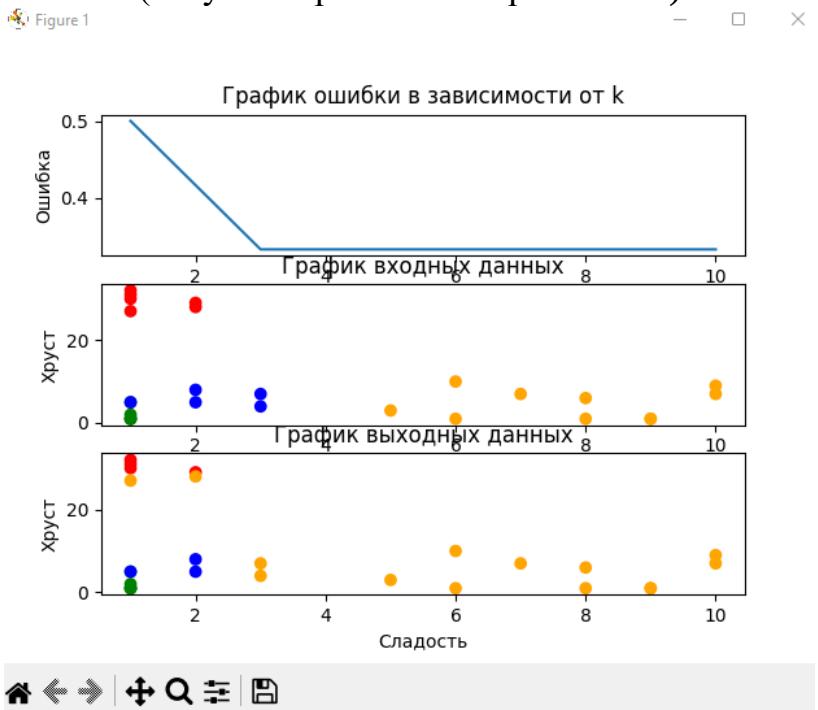


Задание 5.
“Добавление нового класса”.

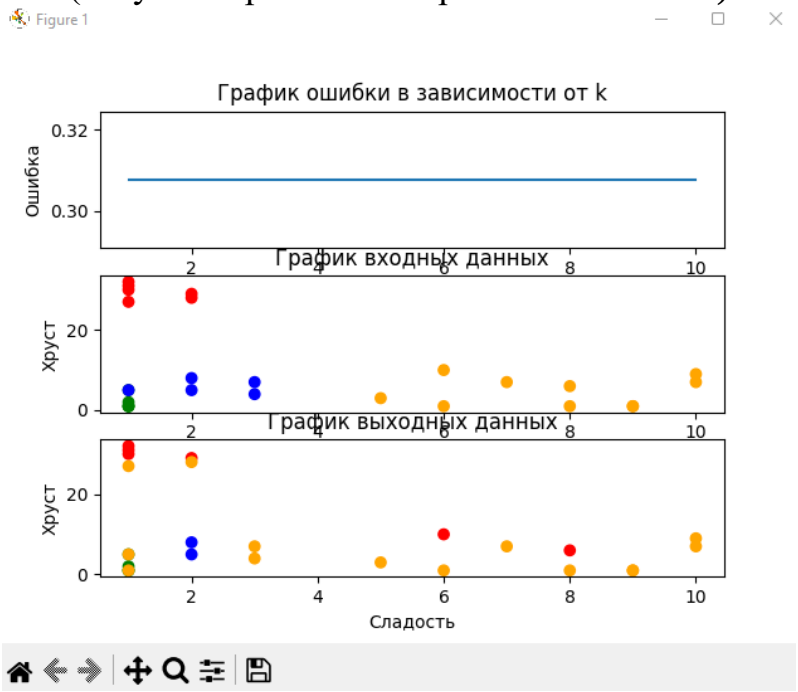
Ввести в набор данных и примеры продукты еще одного класса (возможно изменив набор параметров) и повторить эксперимент.

Результат.

(Результат работы алгоритма knn)



(Результат работы алгоритма sklearn knn)



Код.

```
import csv

import sklearn
import pandas as 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 distance(xt1, xt2, xi1, xi2):
    return ((xt1 - xi1) ** 2 + (xt2 - xi2) ** 2) ** (1/2)

def knn(teach, test, k_in, window_size, class_num):
    data = []
    for i in range(len(teach)):
        data.append(teach[i])
    for j in range(len(test)):
        data.append(test[j])

    teach_size = len(teach) - 1
    test_size = len(data) - 1 - teach_size

    k_max = k_in # число соседей
    new_dist = np.zeros((test_size, teach_size))

    for i in range(test_size):
```

```
    for j in range(teach_size):
      new_dist[i][j] = distance(int(data[teach_size + 1 + i][1]), int(data[teach_size + 1 + i][2]), int(data[j + 1][1]), int(data[j + 1][2]))


  er_k = [0] * k_max  # ошибка
  for k in range(k_max):  # факториальный перебор числа соседей для поиска наилучшего k
    print('\n======================\nКлассификация для k =', k + 1)
    sucsess = 0
    er = [0] * test_size
    classes = [0] * test_size


    for i in range(test_size):  # тестовая выборка (иссследуемая)
      qwant_dist = [0]*class_num  # веса для проверяемой точки
      print(str(i) + '. ' + 'Классификация ', data[teach_size + i + 1][0])  # имя элемента тестовой выборки
      tmp = np.array(new_dist[i, :])  # tmp - текущая строка new_dist
      dist_max = max(tmp)


      for j in range(k + 1):  # количесво соседей , каждая итерация - проверка нового соседа
        ind_min = list(tmp).index(min(tmp))  # ind_min - индекс минимального значения из tmp (индекс ближайшего соседа)
        # qwant_dist[int(data[ind_min + 1][3])] += 1  # увеличение веса отношения исследуемой точки к проверяемому классу
        # qwant_dist[int(data[ind_min + 1][3])] += dist_max-new_dist[i][j]  # увеличение веса отношения исследуемой точки к проверяемому классу
        if (tmp[j] < window_size):  # с парзеновским окном
          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  # сброс нынешней минамальной длины ближайшей точки
        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)  # полученный класс
      classes[i] = class_ind
      # проверка на совпадение класса
      print('Класс классифицируемого элемента = ' + data[teach_size + i + 1][3])
      print(classes[i])
      print(data[teach_size + i + 1][3])
      if (int(classes[i]) == int(data[teach_size + i + 1][3])):
        print('Совпал')
        sucsess += 1
        er[i] = 0  # если класс совпал ошибка 0
      else:
        print('не совпал')
        er[i] = 1  # если класс не совпал ошибка 1


    er_k[k] = np.mean(er)  # среднее значение


    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)

    # Предсказывание
    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

def graphs(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()
```

```
if __name__ == '__main__':
    data = [['Продукт', 'Сладость', 'Хруст', 'Класс'],
        ['Apple', '7', '7', '0'],
        ['Salad', '2', '5', '1'],
        ['Bacon', '1', '2', '2'],
        ['Nuts', '1', '5', '2'],
        ['Fish', '1', '1', '2'],
        ['Cheese', '1', '1', '2'],
        ['Banana', '9', '1', '0'],
        ['Carrot', '2', '8', '1'],
        ['Grape', '8', '1', '0'],
        ['Orange', '6', '1', '0'],
        #test set of 10 (row 11-16)
        ['Strawberry', '9', '1', '0'],
        ['Lettuce', '3', '7', '1'],
        ['Shashlik', '1', '1', '2'],
        ['Pear', '5', '3', '0'],
        ['Celery', '1', '5', '1'],
        ['Apple pie', '6', '10', '0'],
        ['Brownie', '10', '9', '0'],
        ['Puff with cottage cheese', '8', '6', '0'],
        ['Cabbage', '3', '4', '1'],
        ['Cinnabon', '10', '7', '0'],
        ]

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

    print('Data')
    print(data)

    #knn

    k_max=6

    window=2

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

    dataset = pd.read_csv("food_csv.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']
```

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

#sklearn

k_max = 5

my_dataset = pd.read_csv('food_csv.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('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)

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

#add new data

new_data = data[0:11]
new_data.append(['Crackers', '1', '32', '3'])
new_data.append(['Chips', '2', '29', '3'])
new_data.append(['Salty cookies', '1', '31', '3'])
new_data.append(['Crispy chicken', '1', '30', '3'])



new_data = new_data + data[11:]
new_data.append(['Salty bagel', '2', '28', '3'])
new_data.append(['Baguette', '1', '27', '3'])

print('New data')
print(new_data)

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

#knn with new data

k_max = 10

window = 2

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

dataset = pd.read_csv("food_csv.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']

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

#sklearn with new data

k_max = 10
```

```
my_dataset = pd.read_csv('food_csv.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('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)

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