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 №2

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

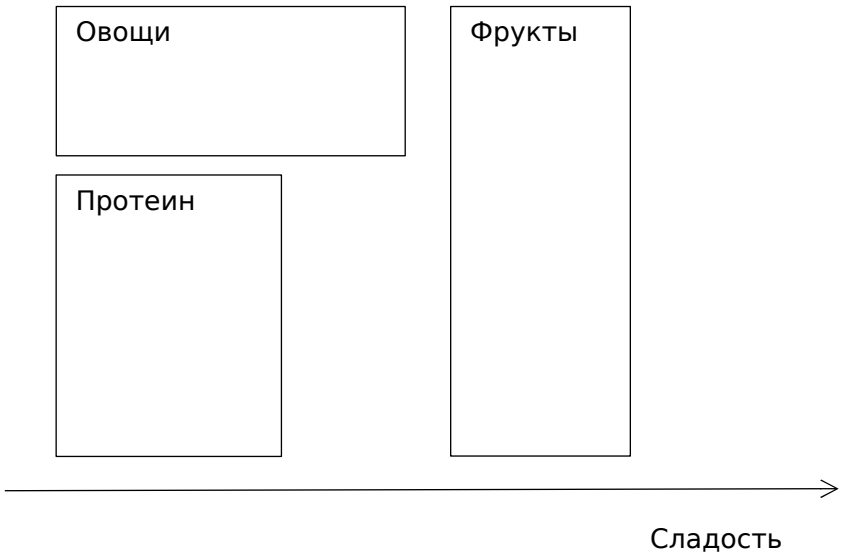
Ульяновск
2022

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)

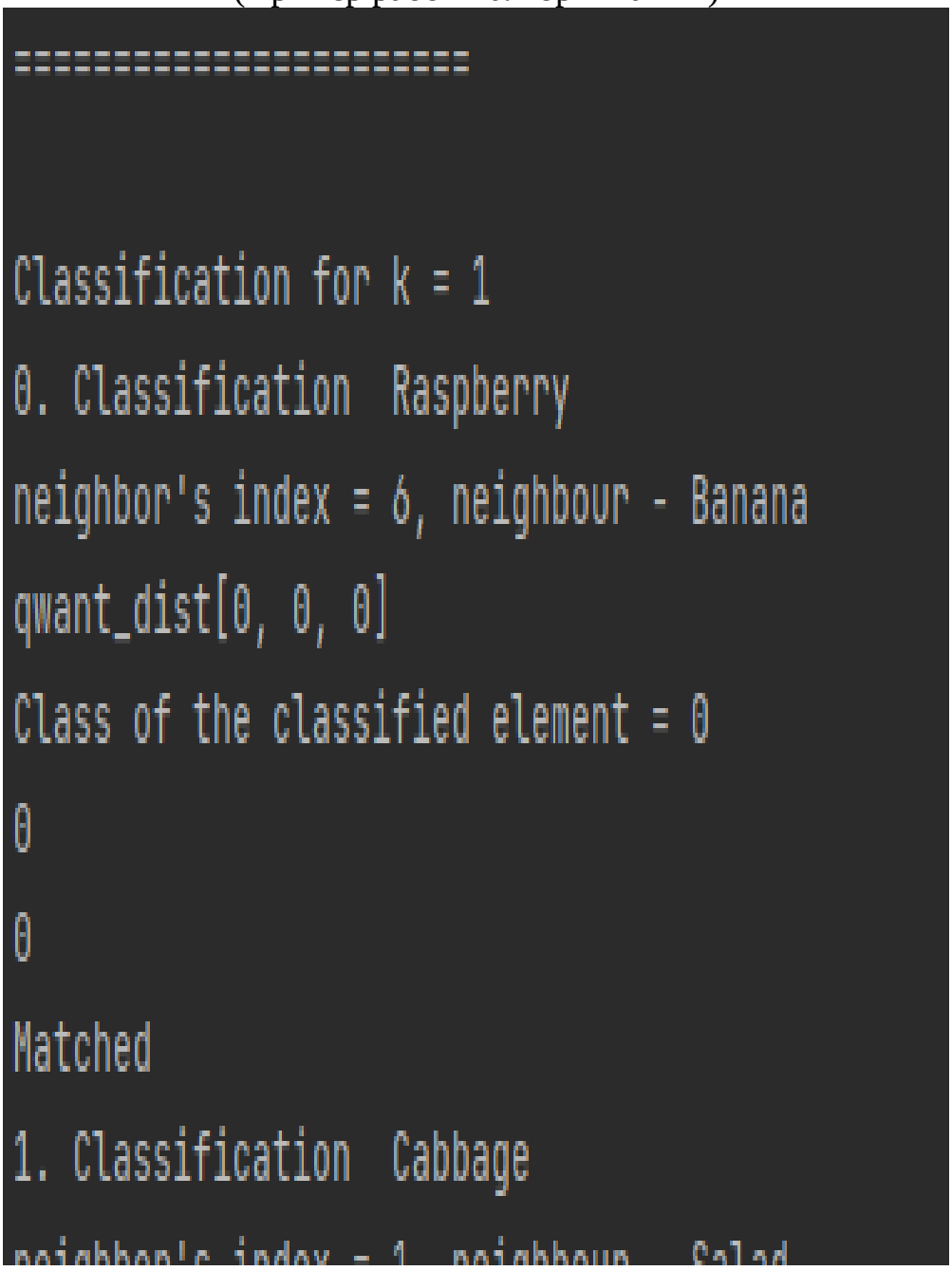
```
Data
[['Product', 'Sweetness', 'Crunch', 'Class'], ['Apple', '7', '7', '0'], ['Salad', '2', '5', '1'], ['Bacon', '1', '2', '2'], ['Nuts', '1', '5', '2'], ['Fish', '1', '1', '2'],
```

Задание 2.
“Получение классификаторов”.

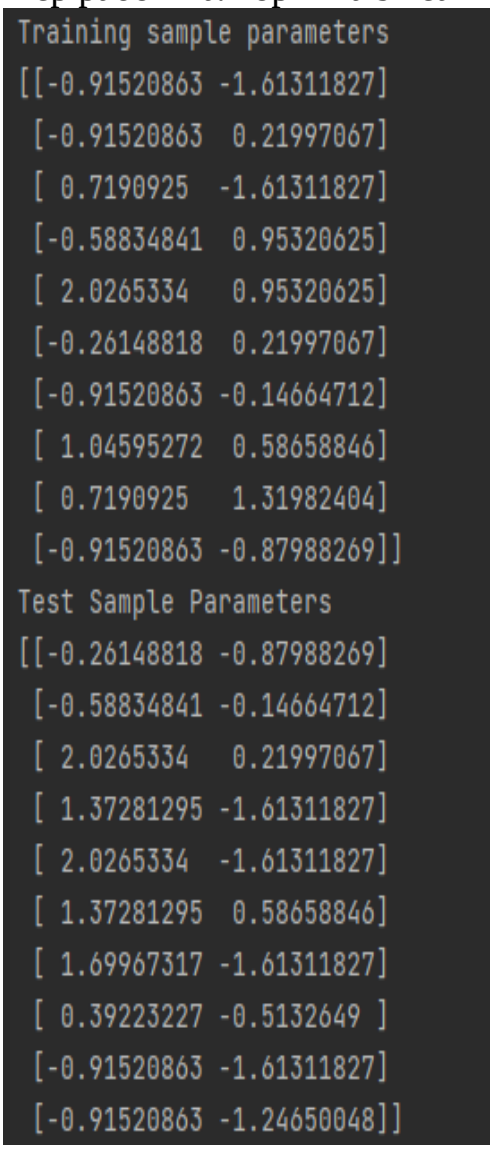
Запрограммировать метрический классификатор по методу k-NN. Для проверки решить ту же задачу методом k-NN библиотеки sklearn.

Результат.

(Пример работы алгоритма knn)



(Пример работы алгоритма sklearn knn)

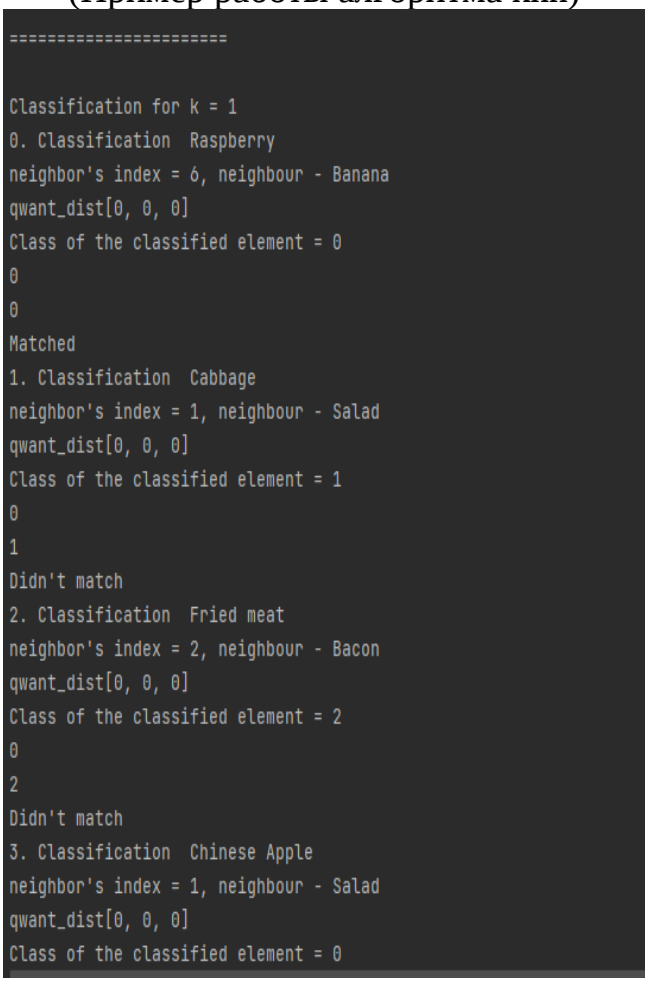


Задание 3.
“Классификация”.

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

Результат.

(Пример работы алгоритма knn)



```
======================

Classification for k = 1
0. Classification  Raspberry
neighbor's index = 6, neighbour - Banana
qwant_dist[0, 0, 0]
Class of the classified element = 0
0
0
Matched
1. Classification  Cabbage
neighbor's index = 1, neighbour - Salad
qwant_dist[0, 0, 0]
Class of the classified element = 1
0
1
Didn't match
2. Classification  Fried meat
neighbor's index = 2, neighbour - Bacon
qwant_dist[0, 0, 0]
Class of the classified element = 2
0
2
Didn't match
3. Classification  Chinese Apple
neighbor's index = 1, neighbour - Salad
qwant_dist[0, 0, 0]
Class of the classified element = 0
```

(Пример работы алгоритма sklearn knn)

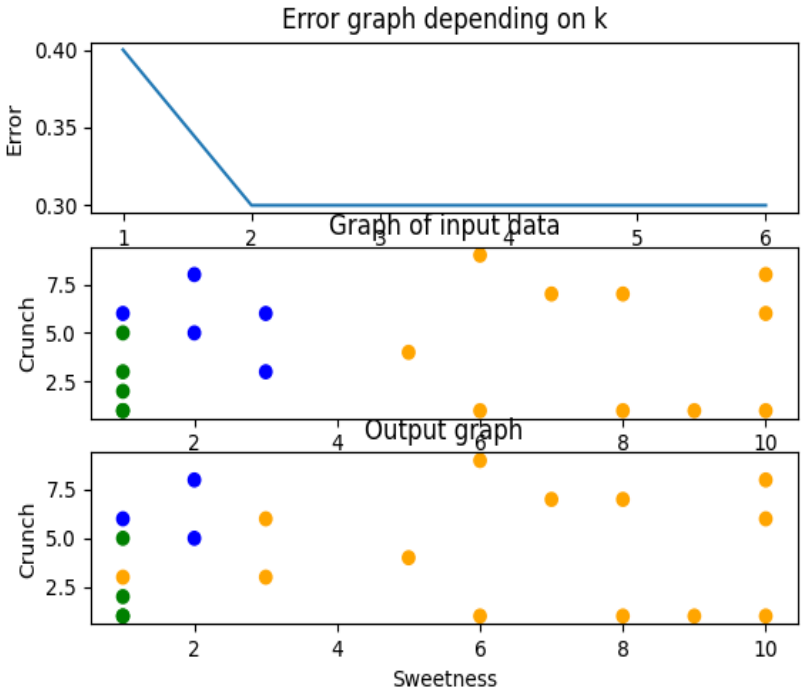
```
Predictions
[2 1 0 0 0 0 0 0 2 2]
Error counting
0 0
1 1
2 2
2 2
2 2
2 2
0 0
1 1
0 0
0 0
2 0
1 1
0 2
0 0
0 1
0 0
0 0
0 0
2 1
2 0
0.25
```

Задание 4.
“Визуализация”.

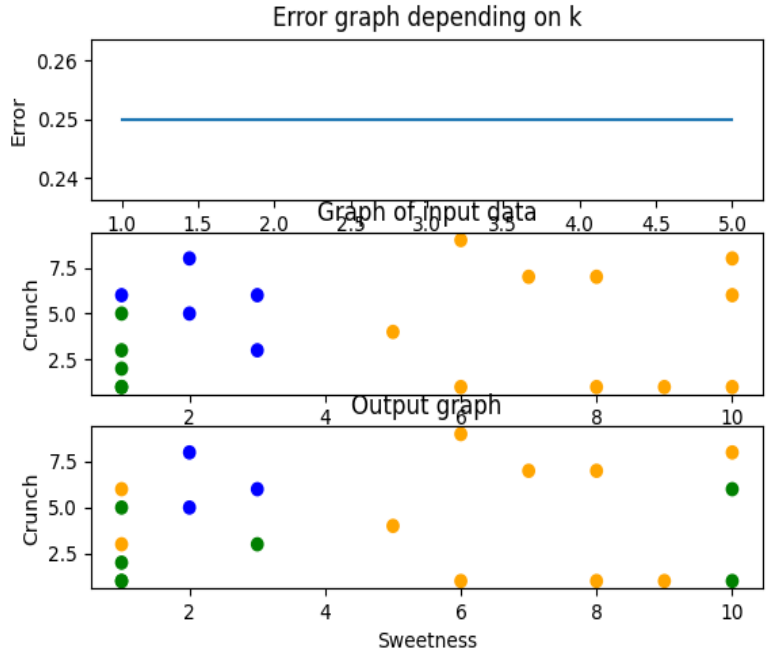
По возможности результаты визуализировать.

Результат.

(Результат работы алгоритма knn)



(Результат работы алгоритма sklearn knn)

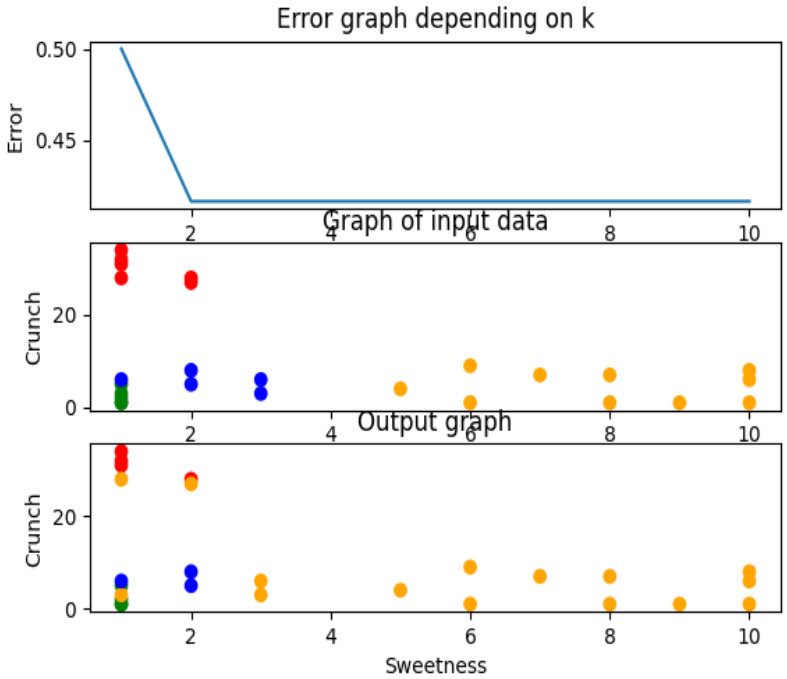


Задание 5.
“Добавление нового класса”.

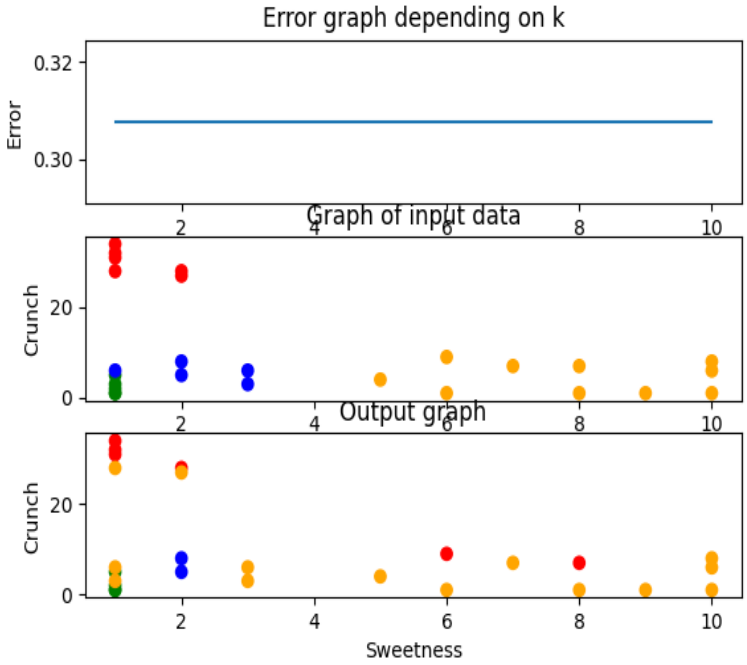
Ввести в набор данных и примеры продукты еще одного класса (возможно изменив набор параметров) и повторить эксперимент.

Результат.

(Результат работы алгоритма knn)



(Результат работы алгоритма sklearn knn)



Код.

```
import csv

import sklearn
import pandas as pads
import numpy as nump
import pylab
import matplotlib.pyplot as plot
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler


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

def my_knn(teach_data, test_data, k_val, win_size, num_class):
    all_data = []
    for i in range(len(teach_data)):
        all_data.append(teach_data[i])
    for j in range(len(test_data)):
        all_data.append(test_data[j])

    teach_size = len(teach_data) - 1
    test_size = len(all_data) - 1 - teach_size

    k_max = k_val
    distance = nump.zeros((test_size, teach_size))

    for i in range(test_size):
        for j in range(teach_size):
            distance[i][j] = dist(int(all_data[teach_size + 1 + i][1]), int(all_data[teach_size + 1 +
i][2]), int(all_data[j + 1][1]), int(all_data[j + 1][2]))
```

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

        for i in range(test_size):
            qwant_dist = [0]*num_class
            print(str(i) + '. ' + 'Classification ', all_data[teach_size + i + 1][0])
            tmp = nump.array(distance[i, :])
            dist_max = max(tmp)

            for j in range(k + 1):
                ind_min = list(tmp).index(min(tmp))
                if (tmp[j] < win_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('neighbor\'s index = ' + str(ind_min) + ', neighbour - ' + all_data[ind_min +
1][0])
                print('qwant_dist' + str(qwant_dist))

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

            print('Class of the classified element = ' + all_data[teach_size + i + 1][3])
            print(classes[i])
            print(all_data[teach_size + i + 1][3])
            if (int(classes[i]) == int(all_data[teach_size + i + 1][3])):
                print('Matched')
                sucsess += 1
                er[i] = 0
            else:
                print('Didn\'t match')
                er[i] = 1

        er_k[k] = nump.mean(er)

        print('Error value for  ' + str(k) + ' neighbor')
        print(er_k)

    return er_k,classes

def sklearn_knn(values_data,classes_data,k,test_sz):

    X_train, X_test, y_train, y_test = train_test_split(
        values_data, classes_data, 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('Training sample parameters')
    print(X_train)
    print('Test Sample Parameters')
    print(X_test)
    print('Training sample classes')
    print(y_train)
    print('Test sample classes')
    print(y_test)
    print('Predictions')
    print(predictions)

    return X_train, X_test, y_train, y_test, predictions

def graphics(k_max,er_k,sweetness,crunch,all_data,colours,classes):
    pylab.subplot(3, 1, 1)
    plot.plot([i for i in range(1, k_max + 1)], er_k)
    plot.title('Error graph depending on k')
    plot.xlabel('k')
    plot.ylabel('Error')

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

    pylab.subplot(3, 1, 2)
    plot.scatter(sweetness, crunch, c=colour_list)
    plot.title('Graph of input data')
    plot.xlabel('Sweetness')
    plot.ylabel('Crunch')

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

    pylab.subplot(3, 1, 3)
    plot.scatter(sweetness, crunch, c=colour_list)
    plot.title('Output graph')
    plot.xlabel('Sweetness')
    plot.ylabel('Crunch')
    plot.show()

if __name__ == '__main__':
    data = [['Product', 'Sweetness', 'Crunch', 'Class'],
        ['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)
        ['Raspberry', '10', '1', '0'],
        ['Cabbage', '3', '6', '1'],
        ['Fried meat', '1', '3', '2'],
        ['Chinese Apple', '5', '4', '0'],
        ['Leek', '1', '6', '1'],
        ['Peer pie', '6', '9', '0'],
        ['Cupcake', '10', '8', '0'],
```

```
    ['Jam pie', '8', '7', '0'],
    ['Cauliflower', '3', '3', '1'],
    ['Bread with poppy seeds', '10', '6', '0'],
    ]

with open('data_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 = my_knn(data[0:11],data[11:],k_max,window,3)

dataset = pads.read_csv("data_food_csv.csv")

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

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

sweet = dataset['Sweetness']
crunch = dataset['Crunch']

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

classes_info = dataset['Class']

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

#sklearn

k_max = 5

my_dataset = pads.read_csv('data_food_csv.csv')
sweetness=my_dataset['Sweetness']
crunch=my_dataset['Crunch']

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

classes=my_dataset['Class']

test_size=0.5

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

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

classes_info = my_dataset['Class']

start_data = my_dataset[:10]['Class']

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

s1 = pads.Series(s1)
```

```
predictions = pads.Series(predictions)
start_data = pads.Series(start_data)
start_data=pads.concat([start_data, predictions])

er=0;
ct=0;

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

print('Error counting')
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(k_max, er_k, sweet, crunch, start_data, colours, classes_info)

#add new data

new_data = data[0:11]
new_data.append(['Bread rolls', '1', '34', '3'])
new_data.append(['Fried potatoes', '2', '28', '3'])
new_data.append(['French fries', '1', '31', '3'])
new_data.append(['Spicy chicken', '1', '32', '3'])


new_data = new_data + data[11:]
new_data.append(['Salted puff', '2', '27', '3'])
new_data.append(['Freshly baked bread', '1', '28', '3'])

print('New data')
print(new_data)

with open('data_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 = my_knn(new_data[0:15], new_data[15:], k_max, window, 4)

dataset = pads.read_csv("data_food_csv.csv")

start_data = dataset[:14]['Class']
```

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

    sweet = dataset['Sweetness']
    crunch = dataset['Crunch']

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

    classes_info = dataset['Class']

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

    #sklearn with new data

    k_max = 10

    my_dataset = pads.read_csv('data_food_csv.csv')
    sweetness = my_dataset['Sweetness']
    crunch = my_dataset['Crunch']

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

    classes = my_dataset['Class']

    test_size = 0.461

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

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

    classes_info = my_dataset['Class']

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

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

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

    er = 0;
    ct = 0;

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

    print('Error counting')
    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(k_max, er_k, sweet, crunch, start_data, colours, classes_info)
```