**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Факультет информационных систем и технологий**

**Кафедра:** «Измерительно-вычислительные комплексы» **Дисциплина**: «Методы искусственного интеллекта»

**Отчет**

по лабораторной работе № 4  
по теме: **«Классификация. Метод k-ближайших соседей (k-NN)»**

Выполнил:

студент гр. ИСТбд-42

Курбаниязов. Ш.Р

Проверил:

к.т.н., доцент

*Шишкин В.В.*

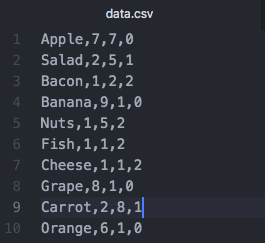
Ульяновск 2022 г.

Выполнение лабораторной работы по теме:

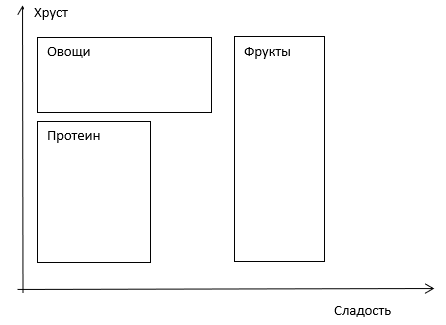
«Классификация. Метод k-ближайших соседей (k-NN)»

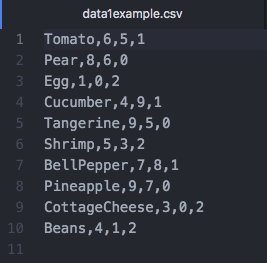
1. Создать симулированный набор данных и записать его на диск в виде csv-файла со следующими параметрами: продукт, сладость, хруст, класс.

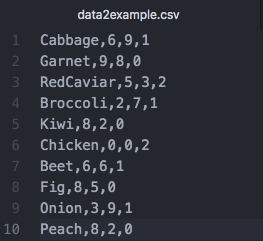
0 – «Фрукты», 1 – «Овощи», 2 – «Протеин»



Подготовить для классификации несколько примеров в соответствии с рисунком:







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

Метрический классификатор по методу k-NN:

import csv

import math

#загрузка данных из сsv-файла

def load\_csv(filename):

dataset = list()

with open("filename, 'r') as csvfile:

lines = csv.reader(csvfile)

for row in lines:

if not row:

continue

dataset.append(row)

return dataset

#вычисление расстояние по метрике Манхэттена

def dist(point1, point2):

return math.fabs(int(point1[1]) - int(point2[1])) + math.fabs(int(point1[2]) - int(point2[2]))

#получение отсортированной по возрастанию коллекции расстояний и индексов

def getNeighbors(trainData, testData, k):

neighborsData = []

distData = []

for i in range(len(trainData)):

distance = dist(testData, trainData[i])

distData.append((trainData[i], distance))

distData.sort(key = lambda x: x[1])

#выбор первых k записей из отсортированной коллекции

for x in range(k):

neighborsData.append(distData[x][0])

return neighborsData

#выбор наиболее часто встречающегося значения (одного) выбранныч ранее меток (расстояний) k

def getLabel(neighborsData):

labels = {}

for x in range(len(neighborsData)):

labels[neighborsData[x][-1]] = labels.get(neighborsData[x][-1], 0) + 1

labels = sorted(labels.items(), key = lambda x: x[1], reverse = True)

return labels[0][0]

#функция оценки точности (сравнения исходных значений и предсказанных)

def getAccuracy(testData, predictions):

correct = 0

for x in range(len(testData)):

if int(testData[x][-1]) == int(predictions[x]):

correct += 1

return (correct/float(len(testData)))\*100

#загрузка обучающей выборки

trainData = load\_csv("data.csv")

#загрузка тестовой выборки

testData = load\_csv("data1example.csv")

k = 3

print("Значение k = ", k)

#предсказывание класса

predictions = []

for x in range(len(testData)):

print("Значение: ", testData[x])

neighbors = getNeighbors(trainData, testData[x], k)

print("Наилучшие соседи: ", neighbors)

result = getLabel(neighbors)

predictions.append(result)

print("предсказываемый класс " + str(result) + " актуальный класс " + repr(testData[x][-1]))

accuracy = getAccuracy(testData, predictions)

print("Точность " + str(accuracy) + "%")

Метрический классификатор с помощью sklearn:

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, Normalizer

from sklearn.metrics import accuracy\_score

#knn-классификатор с помощью бибилиотеки sklearn

def knn\_sklearn(trainData, testData, trainClasses, testClasses, k):

x\_train = trainData

x\_test = testData

y\_train = trainClasses

y\_test = testClasses

#стандартизация и нормализация данных

scaler = StandardScaler().fit(x\_train)

scaler.transform(x\_train)

scaler.transform(x\_test)

scaler = Normalizer().fit(x\_train)

scaler.transform(x\_train)

scaler.transform(x\_test)

model = KNeighborsClassifier(n\_neighbors=k)

model.fit(x\_train, y\_train)

# Предсказывание

predictions = model.predict(x\_test)

print('Классы тестовой выборки')

print(y\_test)

print('Предсказанные классы для тестовой выборки')

print(predictions)

return x\_train, x\_test, y\_train, y\_test, predictions

#загрузка обучающей выборки

trainData = load\_csv("data.csv")

#загрузка тестовой выборки

testData = load\_csv("data1example.csv")

k = 3

#подготовка обучающих данных и целевого столбца

classes\_train = []

for i in range(len(trainData)):

classes\_train.append(trainData[i][-1])

del trainData[i][-1]

del trainData[i][0]

#подготовка тестовых данных и целевого столбца

classes\_test = []

for i in range(len(testData)):

classes\_test.append(testData[i][-1])

del testData[i][-1]

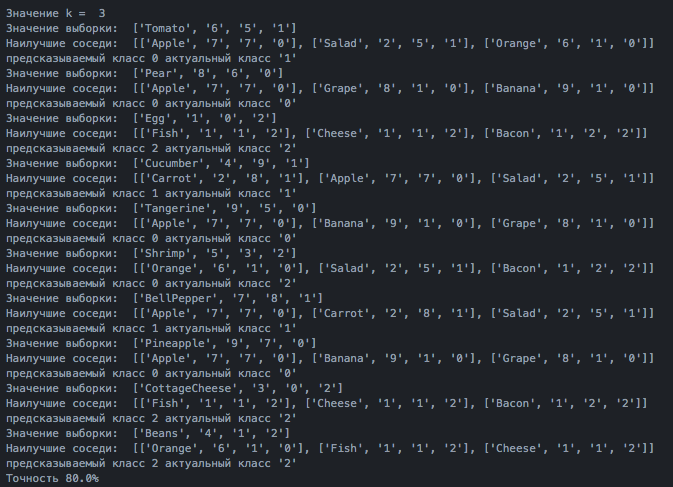
del testData[i][0]

X\_train, X\_test, y\_train, y\_test, predictions = knn\_sklearn(trainData, testData, classes\_train, classes\_test, k)

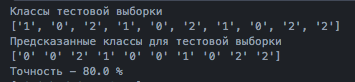
print("Точность - " + str((accuracy\_score(y\_test, predictions))\*100.0) + " %")

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

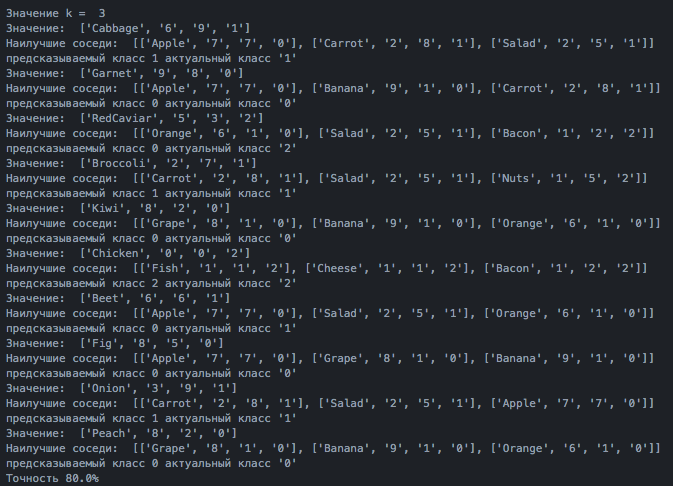
Вывод - пример 1:



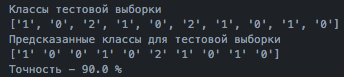
Вывод - пример 1 (классификатор sklearn):



Вывод - пример 2:

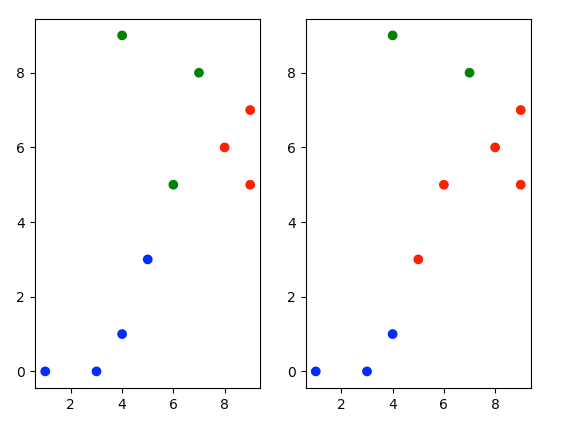


Вывод - примера 2 (классификатор sklearn):

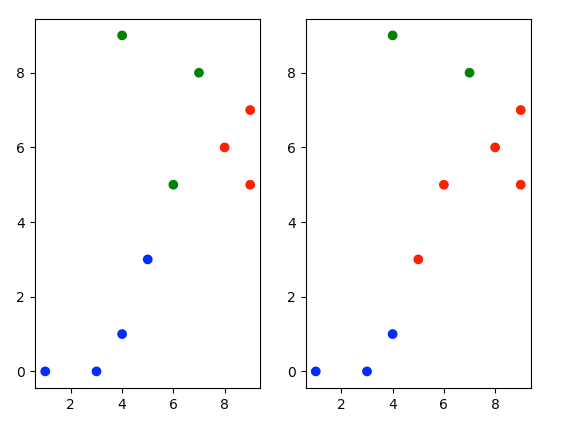


1. Визуализировать данные.

Классификация тестовой выборки и результат классификации по методу kNN:



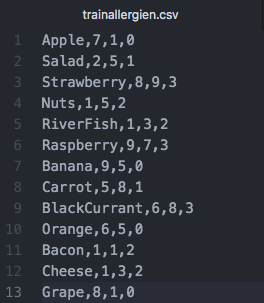
Классификация тестовой выборки и результат классификации sklearn:

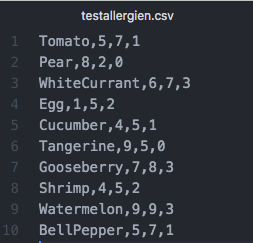


1. Ввести в набор данных примеры продуктов еще одного класса (возможно изменив набор параметров) и повторить эксперимент.

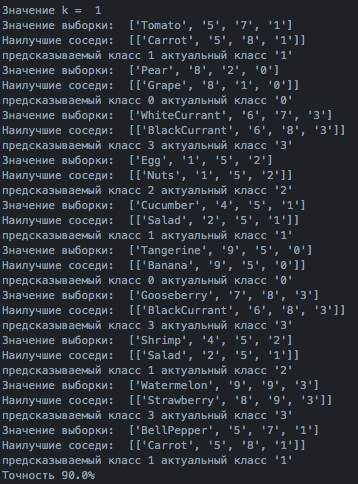
Набор параметров был изменен с сладость-хруст на сладость-аллергенность, а также был добавлен новый класс – ягоды.

Новые наборы данных:

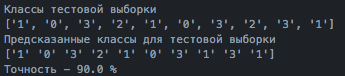




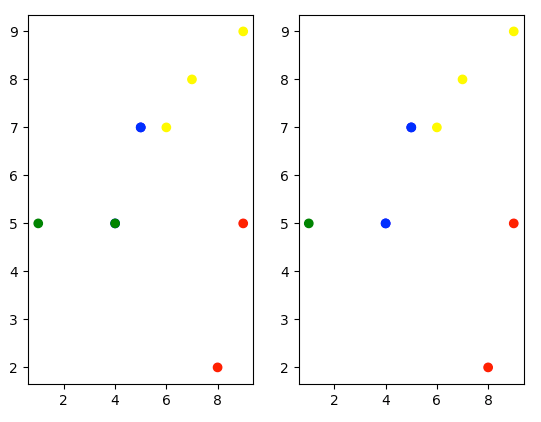
Вывод:



Вывод (классификатор sklearn):



Классификация тестовой выборки и результат классификации по методу kNN:



Классификация тестовой выборки и результат классификации sklearn:

