**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-4

Проверил:

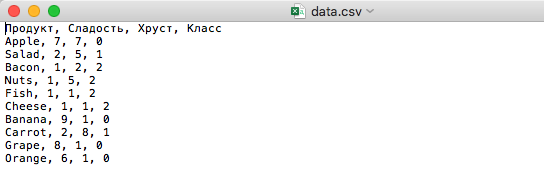
к.т.н., доцент

*Шишкин В.В.*

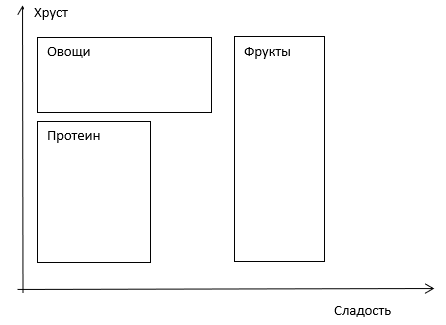
Ульяновск 2022 г.

Выполнение лабораторной работы по теме: «Классификация. Метод ближайших соседей (k-NN)»

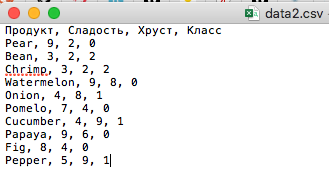
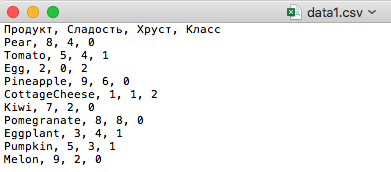
1. Для начала создадим симулированный набор данных в виде csv-файла с параметрами: продукт, сладость, хруст, класс.



1. Затем подготовим для классификации несколько примеров в соответствии с рисунком:



Пример 1 Пример 2



1. Запрограммируем метрический классификатор по методу k-NN. Для проверки решим ту же задачу методом k-NN библиотеки skilearn.

Скрипт метрического классификатора по методу k-NN:

import csv

import math

#загрузка 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

#основная задача метрического классификатора - определить класс объекта тестовой выборки

#определить значение ошибки (точности)

#определение меры сходства

#подсчет евклидова расстояния между двумя двумерными точками

def dist(instance1, instance2):

return math.sqrt((int(instance1[1]) - int(instance2[1]))\*\*2 + (int(instance1[2]) - int(instance2[2]))\*\*2)

#определение k наиболее похожих соседей из обучающего набора для тестовой выборки

def getNeighbors(trainingData, testingData, k):

neighborsData = []

#для каждого значения тестируемой выборки определим меру сходства и занесем в список

distData = []

for i in range(len(trainingData)):

distance = dist(testingData, trainingData[i])

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

#сортировка полученного списка расстояний по возрастанию

def keyFuncSort(item):

return item[1]

distData.sort(key = keyFuncSort)

#получение k наиболее подходящих соседей по расстоянию

for x in range(k):

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

return neighborsData

#создание словаря классов с результатами "голосования" соседей - возвращает наиболее часто встречающееся значение выбранной метки

def getLabel(neighborsData):

labels = {}

for x in range(len(neighborsData)):

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

def keyFuncSort(item):

return item[1]

labels = sorted(labels.items(), key = keyFuncSort, reverse = True)

return labels[0][0]

def getAccurancy(testingData, predictions):

correct = 0

for x in range(len(testingData)):

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

correct += 1

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

#загрузка обучающей выборки

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

#загрузка тестовой выборки

testData = load\_csv("data1.csv")

k = 1

print("Значение k = ", k)

#предсказывание класса

predictions = []

for x in range(len(testData)):

print("Значение выборки: ", 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]))

accurancy = getAccurancy(testData, predictions)

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

Скрипт метрического классификатора с помощью skilearn:

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

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

from sklearn.preprocessing import StandardScaler

from sklearn import preprocessing

from sklearn.metrics import accuracy\_score

#загрузка 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

def knn\_sklearn(trainData, testData, classes\_train, classes\_testing, k):

X\_train = trainData

X\_testing = testData

y\_train = classes\_train

y\_testing = classes\_testing

scalerror = StandardScaler()

scalerror.fit(X\_train)

X\_train = scalerror.transform(X\_train)

X\_testing = scalerror.transform(X\_testing)

model = KNeighborsClassifier(n\_neighbors=k)

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

#Предсказывание

predictions = model.predict(X\_testing)

print('Параметры обучающей выборки')

print(X\_train)

print('Параметры тестовой выборки')

print(X\_testing)

print('Классы обучающей выборки')

print(y\_train)

print('Классы тестовой выборки')

print(y\_testing)

print('Предсказания')

print(predictions)

return X\_train, X\_testing, y\_train, y\_testing, predictions

#загрузка обучающей выборки

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

#загрузка тестовой выборки

testData = load\_csv("data1.csv")

k = 1

classes\_train = []

for i in range(len(trainData)):

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

classes\_testing = []

for i in range(len(testData)):

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

sweetness\_train = []

for i in range(len(trainData)):

sweetness\_train.append(int(trainData[i][1]))

crunch\_train= []

for i in range(len(trainData)):

crunch\_train.append(int(trainData[i][2]))

trainData=np.array(list(zip(sweetness\_train, crunch\_train)), dtype=np.float64)

sweetness\_test = []

for i in range(len(testData)):

sweetness\_test.append(int(testData[i][1]))

crunch\_test= []

for i in range(len(testData)):

crunch\_test.append(int(testData[i][2]))

testData=np.array(list(zip(sweetness\_test, crunch\_test)), dtype=np.float64)

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

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

1. Прочитаем сгенерированный набор данных, настроим классификатор и проведем эксперимент по классификации с контролем для подготовленных примеров.

Вывод для примера 1 (результат написанного классификатора):

Значение k = 1

Значение выборки: ['Pear', '8', '4', '0']

Наилучшие соседи: [['Grape', '8', '1', '0']]

предсказываемый класс 0 актуальный класс '0'

Значение выборки: ['Tomato', '5', '4', '1']

Наилучшие соседи: [['Salad', '2', '5', '1']]

предсказываемый класс 1 актуальный класс '1'

Значение выборки: ['Egg', '2', '0', '2']

Наилучшие соседи: [['Fish', '1', '1', '2']]

предсказываемый класс 2 актуальный класс '2'

Значение выборки: ['Pineapple', '9', '6', '0']

Наилучшие соседи: [['Apple', '7', '7', '0']]

предсказываемый класс 0 актуальный класс '0'

Значение выборки: ['CottageCheese', '1', '1', '2']

Наилучшие соседи: [['Fish', '1', '1', '2']]

предсказываемый класс 2 актуальный класс '2'

Значение выборки: ['Kiwi', '7', '2', '0']

Наилучшие соседи: [['Grape', '8', '1', '0']]

предсказываемый класс 0 актуальный класс '0'

Значение выборки: ['Pomegranate', '8', '8', '0']

Наилучшие соседи: [['Apple', '7', '7', '0']]

предсказываемый класс 0 актуальный класс '0'

Значение выборки: ['Eggplant', '3', '4', '1']

Наилучшие соседи: [['Salad', '2', '5', '1']]

предсказываемый класс 1 актуальный класс '1'

Значение выборки: ['Pumpkin', '5', '3', '1']

Наилучшие соседи: [['Orange', '6', '1', '0']]

предсказываемый класс 0 актуальный класс '1'

Значение выборки: ['Melon', '9', '2', '0']

Наилучшие соседи: [['Banana', '9', '1', '0']]

предсказываемый класс 0 актуальный класс '0'

Точность 90.0%

Вывод для примера 1 (результат классификатора sklearn):

Параметры обучающей выборки

[[ 1.02429504 1.44038628]

[-0.57616596 0.68228824]

[-0.89625816 -0.45485883]

[-0.89625816 0.68228824]

[-0.89625816 -0.83390785]

[-0.89625816 -0.83390785]

[ 1.66447944 -0.83390785]

[-0.57616596 1.8194353 ]

[ 1.34438724 -0.83390785]

[ 0.70420284 -0.83390785]]

Параметры тестовой выборки

[[ 1.34438724 0.30323922]

[ 0.38411064 0.30323922]

[-0.57616596 -1.21295687]

[ 1.66447944 1.06133726]

[-0.89625816 -0.83390785]

[ 1.02429504 -0.45485883]

[ 1.34438724 1.8194353 ]

[-0.25607376 0.30323922]

[ 0.38411064 -0.0758098 ]

[ 1.66447944 -0.45485883]]

Классы обучающей выборки

['0', '1', '2', '2', '2', '2', '0', '1', '0', '0']

Классы тестовой выборки

['0', '1', '2', '0', '2', '0', '0', '1', '1', '0']

Предсказания

['0' '1' '2' '0' '2' '0' '0' '1' '0' '0']

Точность - 90.0 %

Вывод для примера 2 (результат написанного классификатора):

Значение k = 1

Значение выборки: ['Pear', '9', '2', '0']

Наилучшие соседи: [['Banana', '9', '1', '0']]

предсказываемый класс 0 актуальный класс '0'

Значение выборки: ['Bean', '3', '2', '2']

Наилучшие соседи: [['Bacon', '1', '2', '2']]

предсказываемый класс 2 актуальный класс '2'

Значение выборки: ['Chrimp', '3', '2', '2']

Наилучшие соседи: [['Bacon', '1', '2', '2']]

предсказываемый класс 2 актуальный класс '2'

Значение выборки: ['Watermelon', '9', '8', '0']

Наилучшие соседи: [['Apple', '7', '7', '0']]

предсказываемый класс 0 актуальный класс '0'

Значение выборки: ['Onion', '4', '8', '1']

Наилучшие соседи: [['Carrot', '2', '8', '1']]

предсказываемый класс 1 актуальный класс '1'

Значение выборки: ['Pomelo', '7', '4', '0']

Наилучшие соседи: [['Apple', '7', '7', '0']]

предсказываемый класс 0 актуальный класс '0'

Значение выборки: ['Cucumber', '4', '9', '1']

Наилучшие соседи: [['Carrot', '2', '8', '1']]

предсказываемый класс 1 актуальный класс '1'

Значение выборки: ['Papaya', '9', '6', '0']

Наилучшие соседи: [['Apple', '7', '7', '0']]

предсказываемый класс 0 актуальный класс '0'

Значение выборки: ['Fig', '8', '4', '0']

Наилучшие соседи: [['Grape', '8', '1', '0']]

предсказываемый класс 0 актуальный класс '0'

Значение выборки: ['Pepper', '5', '9', '1']

Наилучшие соседи: [['Apple', '7', '7', '0']]

предсказываемый класс 0 актуальный класс '1'

Точность 90.0%

Вывод для примера 2 (результат классификатора sklearn):

Параметры обучающей выборки

[[ 1.02429504 1.44038628]

[-0.57616596 0.68228824]

[-0.89625816 -0.45485883]

[-0.89625816 0.68228824]

[-0.89625816 -0.83390785]

[-0.89625816 -0.83390785]

[ 1.66447944 -0.83390785]

[-0.57616596 1.8194353 ]

[ 1.34438724 -0.83390785]

[ 0.70420284 -0.83390785]]

Параметры тестовой выборки

[[ 1.66447944 -0.45485883]

[-0.25607376 -0.45485883]

[-0.25607376 -0.45485883]

[ 1.66447944 1.8194353 ]

[ 0.06401844 1.8194353 ]

[ 1.02429504 0.30323922]

[ 0.06401844 2.19848433]

[ 1.66447944 1.06133726]

[ 1.34438724 0.30323922]

[ 0.38411064 2.19848433]]

Классы обучающей выборки

['0', '1', '2', '2', '2', '2', '0', '1', '0', '0']

Классы тестовой выборки

['0', '2', '2', '0', '1', '0', '1', '0', '0', '1']

Предсказания

['0' '2' '2' '0' '1' '0' '1' '0' '0' '0']

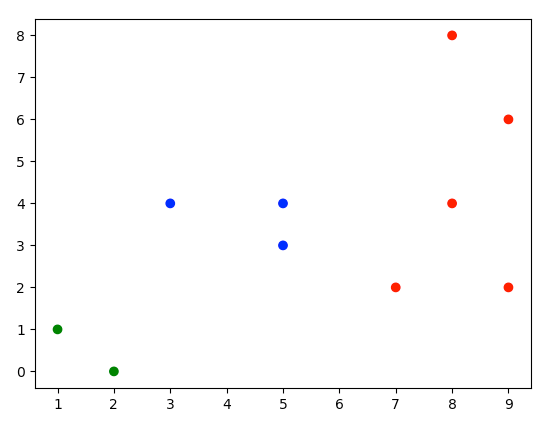
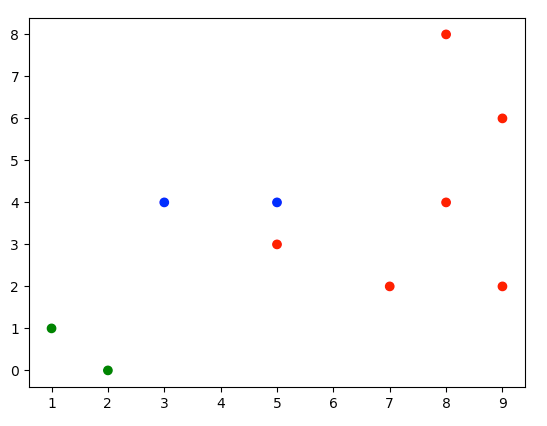
Точность - 90.0 %

'0' '0' '2' '2']

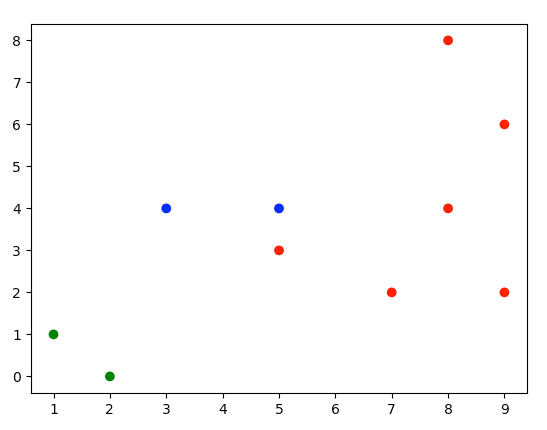
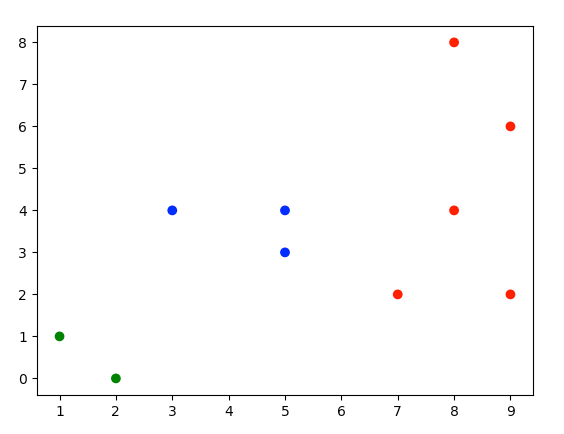
Точность - 90.0 %

1. Визуализируем данные для первого примера и получим следующие графические представления:

Визуализация предсказанных данных и актуальных данных (Python):



Визуализация предсказанных данных и актуальных данных (SkLearn):

1. Введем в набор данных примеры продуктов еще одного класса и повторим эксперимент.

Значение k = 1

Значение выборки: ['Pear', '8', '4', '0']

Наилучшие соседи: [['Grape', '8', '1', '0']]

предсказываемый класс 0 актуальный класс '0'

Значение выборки: ['Tomato', '5', '4', '1']

Наилучшие соседи: [['Salad', '2', '5', '1']]

предсказываемый класс 1 актуальный класс '1'

Значение выборки: ['DriedFruit', '8', '6', '3']

Наилучшие соседи: [['Apple', '7', '7', '0']]

предсказываемый класс 0 актуальный класс '3'

Значение выборки: ['Pineapple', '9', '6', '0']

Наилучшие соседи: [['Popcorn', '9', '7', '3']]

предсказываемый класс 3 актуальный класс '0'

Значение выборки: ['CottageCheese', '1', '1', '2']

Наилучшие соседи: [['Fish', '1', '1', '2']]

предсказываемый класс 2 актуальный класс '2'

Значение выборки: ['Kiwi', '7', '2', '0']

Наилучшие соседи: [['Orange', '6', '1', '0']]

предсказываемый класс 0 актуальный класс '0'

Значение выборки: ['Seeds', '1', '7', '3']

Наилучшие соседи: [['Carrot', '2', '8', '1']]

предсказываемый класс 1 актуальный класс '3'

Значение выборки: ['Eggplant', '3', '4', '1']

Наилучшие соседи: [['Salad', '2', '5', '1']]

предсказываемый класс 1 актуальный класс '1'

Значение выборки: ['Pocky', '9', '8', '3']

Наилучшие соседи: [['Cookies', '9', '9', '3']]

предсказываемый класс 3 актуальный класс '3'

Значение выборки: ['Melon', '9', '2', '0']

Наилучшие соседи: [['Banana', '9', '1', '0']]

предсказываемый класс 0 актуальный класс '0'

Значение выборки: ['Egg', '2', '0', '2']

Наилучшие соседи: [['Fish', '1', '1', '2']]

предсказываемый класс 2 актуальный класс '2'

Значение выборки: ['Pomegranate', '8', '8', '0']

Наилучшие соседи: [['Apple', '7', '7', '0']]

предсказываемый класс 0 актуальный класс '0'

Значение выборки: ['Pumpkin', '5', '3', '1']

Наилучшие соседи: [['Orange', '6', '1', '0']]

предсказываемый класс 0 актуальный класс '1'

Точность 69.23076923076923%

Параметры обучающей выборки

[[ 0.76086957 0.81743996]

[-0.65217391 0.19233881]

[-1.2173913 1.44254111]

[-0.93478261 0.19233881]

[-0.93478261 -1.05786348]

[ 1.32608696 1.44254111]

[ 1.32608696 -1.05786348]

[-0.65217391 1.12999054]

[ 1.32608696 0.81743996]

[ 0.47826087 -1.05786348]

[-0.93478261 -0.74531291]

[-0.93478261 -1.05786348]

[ 1.04347826 -1.05786348]]

Параметры тестовой выборки

[[ 1.04347826 -0.12021176]

[ 0.19565217 -0.12021176]

[ 1.04347826 0.50488939]

[ 1.32608696 0.50488939]

[-0.93478261 -1.05786348]

[ 0.76086957 -0.74531291]

[-0.93478261 0.81743996]

[-0.36956522 -0.12021176]

[ 1.32608696 1.12999054]

[ 1.32608696 -0.74531291]

[-0.65217391 -1.37041405]

[ 1.04347826 1.12999054]

[ 0.19565217 -0.43276233]]

Классы обучающей выборки

['0', '1', '3', '2', '2', '3', '0', '1', '3', '0', '2', '2', '0']

Классы тестовой выборки

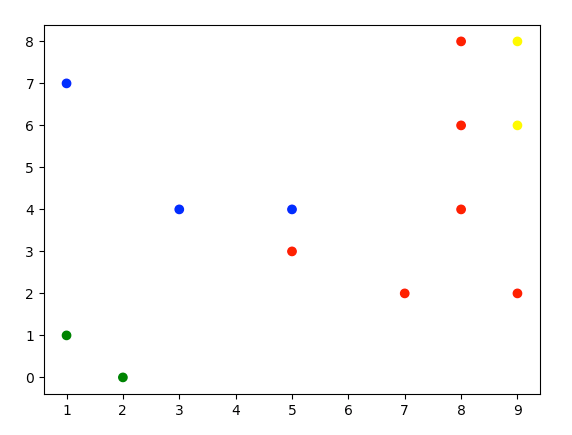
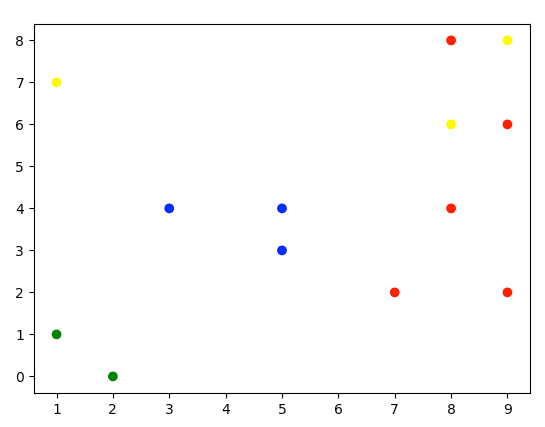
['0', '1', '3', '0', '2', '0', '3', '1', '3', '0', '2', '0', '1']

Предсказания

['0' '1' '3' '3' '2' '0' '1' '1' '3' '0' '2' '3' '0']

Точность - 69.23076923076923 %

Визуализация предсказанных данных и актуальных данных (Python):



Визуализация предсказанных данных и актуальных данных (SkLearn):

