Домашно 2 по „Изкуствен Интелект“
k Nearest Neighbours

1. Описание на проблема.

Дадено е множество от обучаващи примери, включващи данни за профили на клиенти на фирма. Примерите включват данни за:

- сумата от разходите, осъществени от клиента (на тримесечна основа, в хил. лв.);
- честотата на извършване на покупки;
- вида на извършените покупки (0 – покупки от стандартната ценова листа, 1 – покупки от специалните оферти в каталога на фирмата)

Задачата е да се разработи програмна система, която да получава данни за очакваните тримесечни разходи и честота на покупките на нов клиент на фирмата и да предвижда вида на неговите покупки (ще търси специални оферти или ще се насочи към стандартната ценова листа), като за целта прилага метода на k най-близки съседи (k-NN).

2. Описание на използвания метод за решаване на задачата.

При решаването на задачата се налага да мерим близост на база два параметъра. За целта първо ги нормираме, за да имат подобни влияния върху крайния резултат. Впоследствие прилагаме метода „k най-близки съседи“ върху данните, разглеждайки близостта на новопостъпилия запис до останалите, взимайки под внимание и двата параметъра (подобно на Евклидовото разстояние).

Методът „k най-близки съседи“ (k Nearest Neighbours, k-NN) е метод за машинно самообучение, който, на база предварително зададени обучаващи примери, взима решение как да класифицира новопостъпил запис/пример/обект. В тази задача възможните класификации са две и затова можем теоретично да си позволим да ограничим числото k да бъде нечетно и на база мнозинството измежду взетите k най-близки съседи можем винаги да класифицираме новия, без да се налага да правим избори на сляпо, ако се случи да имаме еднакъв брой представители на двата класа измежду взетите k най-близки съседи.

3. Описание на реализацията с псевдокод.

```
enum ExpenseFrequency {
        Never = 0,
        Rarely = 1,
        Sometimes = 2,
        Often = 3,
        VeryOften = 4,
} - изброен тип за честотата на покупките, в който се съдържат 5 поредни тъпа със съответна номерация.

struct Client{ - структура, която съдържа 3 полета:
        integer  expenses - разноски на клиента за тримесечие (цяло число в хиляди лева)
        ExpenseFrequency  frequency - честота на покупките
        boolean  discount - дали клиентът се интересува от специални оферти (true или false)
}
```

struct Pair {
	Client client - клиент от множеството обучаващи примери
	integer distance - изчисленото разстояние между гореспоменатия и некласифицирания клиент на база нормите на първите им две полета (подробности по-надолу)
}

Function initializeData(**Client array**, name of **input file**, **minimum value** from the training data, **maximum value** from the training data)
{
	skip the first line from the input file
	for each row in the input file:
		push the new data for a client to the back of the array
}

normalizedDistanceBetween(**client** from the training set, **client** to be classified, **minimum value** from the training data, **maximum value** from the training data)
{
	normalize both clients' expenses
	normalize both clients' frequency of purchases

	return the square root of the sum of the squared distances between both clients' expenses and purchase frequencies
}

normalizeData(**Client array** (initialized by the upper function), **client** to be classified, **minimum value** from the training data, **maximum value** from the training data, **priority queue** of **Pair**s)
{
	for each client in the **Client array**:
		push the **Pair** (client, *distance*) into the **priority queue**, where *distance* is the distance between the client from the array and the client to be classified (calculated by the upper function)
}

majority(k - integer, **priority queue** of **Pair**s) - returns a boolean preference
{
	take the first **k** elements from the queue and count their discount preferences
	print the **k** nearest neighbours' data and their distance from the unclassified one

	if there is an equal number of preferences, generate a random preference and return it

	otherwise return the preference with more occurences
}

main()
{
	set **minimum** and **maximum** to opposite extremes

declare the **Client array**
declare the **queue of Pairs**

set **k** to be a number of your choice for an amount of nearest neighbours to look at
set the unclassified **client**'s expenses and frequency of purchases

initialize and normalize the training data (using the corresponding functions)
set the unclassified **client**'s discount preference to the value passed by the function *majority*

print the newly classified **client**'s data
}

4. Инструкции за компилиране на програмата.

Програмата е реализирана в един единствен файл и не би трябвало да представлява затруднение за Visual Studio 2019 с компилатор за C++ 14 или дори за C++ 11.

5. Примерни резултати.

Пробягвайки през интервала от най-малката до най-голямата похарчена сума (закръглени) за всяка нечетна стойност на k (ефективно избягвайки дори шанс за взимане на най-близкия съсед при евентуално равенство) и всяка честота на покупки за нов клиент, виждаме, че до стойност около 950 на похарчената сума и от стойност около 1200 задача почти няма. Интересен е интервалът [950; 1200], където се проявяват същинските възможности на този алгоритъм.

В състоянието, в което е качено това решение, можете да тествате единични примери за клиент с произволно поведение и при колкото пожелаете най-близки съседи (от 0 до 40, защото толкова са наличните обучителни примери).

Приложеният файл с резултати към това решение се генерира чрез закоментираната част от функцията main. За коректно генериране на нови данни, закоментирайте редове от 187 до 198, както и от 144 до 164, и разкоментирайте редовете от 200 до 236. След това свободно променяйте размера на стъпките за променливи *k* и *exp* в съответните им for цикли.