Московский государственный технический

университет им. Н.Э. Баумана

Факультет «Радиотехника»
Кафедра «Информатика и вычислительная техника»

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашнему заданию

Тема «Простой классификатор на f#»

Выполнил:
студент группы РТ5-31Б:
Михеева В.А.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2023 г.

ОПИСАНИЕ ЗАДАНИЯ

1. Выберите язык программирования (который Вы ранее не изучали) и (1) напишите по нему реферат с примерами кода или (2) реализуйте на нем небольшой проект (с детальным текстовым описанием).
2. Реферат (проект) может быть посвящен отдельному аспекту (аспектам) языка или содержать решение какой-либо задачи на этом языке.
3. Необходимо установить на свой компьютер компилятор (интерпретатор, транспилятор) этого языка и произвольную среду разработки.
4. В случае написания реферата необходимо разработать и откомпилировать примеры кода (или модифицировать стандартные примеры).
5. В случае создания проекта необходимо детально комментировать код.
6. При написании реферата (создании проекта) необходимо изучить и корректно использовать особенности парадигмы языка и основных конструкций данного языка.
7. Приветствуется написание черновика статьи по результатам выполнения ДЗ. Черновик статьи может быть подготовлен группой студентов, которые исследовали один и тот же аспект в нескольких языках или решили одинаковую задачу на нескольких языках.

Я создавала простой классификатор методом вычисления крачайшего расстояния до соседей. На языке f#.

Задача выполнена на Polyglot notebook.

ТЕКСТ ПРОГРАМЫ:

```
open System.IO

//Классификация производится по petal_length и petal_width

let train_set = "iris_doc.txt"

let test_set = "iris_test.txt"

let read_file fn = File.ReadAllLines(fn)


let data_test = read_file test_set|> Array.map(fun s -> s.Split(' '))|>
Array.map(fun s -> s[2]|> double, s[3]|> double, s[4])

let data = read_file train_set

            |> Array.skip 1

            |> Array.map(fun s -> s.Split(' '))

            |> Array.map(fun s -> s[2]|> double, s[3]|> double, s[4])
```

```
let data_for_see = Array.truncate 5 data

let mutable k=0

for n in data_for_see do

    n

    printf $"{n} "

    printf "\n"

data_test

let getDistance ((x1,y1, s): double*double*string) ((x2,y2, s2):
double*double*string) =

    (x1-x2)**2 + (y1-y2)**2

    |> sqrt

(getDistance(data[0])(data_test[1]) )

let second ((_, c): double*string) = c

let third ((_, _, c): double*double*string) = c

third(data[0])

let get_neighbors (train: (double*double*string) array, test) =

    let mutable distances = [for i=0 to (Array.length train)-1 do
getDistance(train[i])(test)]

    let mutable distances_vid = [for i=0 to (Array.length train)-1 do
third(train[i])]

    let mutable distances_complete = List.map2 (fun x y -> x, y) distances
distances_vid

    let distance_sort = List.sort distances_complete

    let min = second(distance_sort[0])

    min

let accuracy =

    let correct = 0

    printfn "expected_answer     answer"

    for i=0 to (Array.length data_test)-1 do

        let answer = get_neighbors (data, data_test[i])

        let expected_answer = third(data_test[i])


        printfn $"   {expected_answer}           {answer} "
```

Результат Работы:

```
expected_answer answer setosa setosa setosa setosa setosa setosa setosa setosa
setosa setosa setosa setosa versicolor versicolor versicolor versicolor
versicolor versicolor versicolor versicolor versicolor versicolor versicolor
versicolor virginica virginica virginica virginica virginica virginica virginica
versicolor virginica virginica virginica virginica None virginica
```