Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Отчёт по лабораторной работе №1

# Метод k взвешенных ближайших соседей

Выполнил:

студент гр. ИП-111

Кузьменок Д.В.

Проверил:

Старший преподаватель кафедры ПМиК

Дементьева К.И.

Новосибирск, 2024 г.

# Задание

Входные данные:

К заданию на лабораторную работу прилагаются файлы, в которых представлены наборы данных из объектов. Каждый объект описывается двумя признаками ($f_i(x) \in R$) и соответствующим ему классом ($y \in \{0,1\}$).

Задание:

Суть лабораторной работы заключается в написании классификатора на основе метода k ближайших соседей. Данные из файла необходимо разбить на две выборки, обучающую и тестовую, согласно общепринятым правилам разбиения. На основе этих данных необходимо обучить разработанный классификатор и протестировать его на обеих выборках. В качестве отчёта требуется представить работающую программу и таблицу с результатами тестирования для каждого из 10 разбиений. Разбиение выборки необходимо выполнять программно, случайным образом, при этом, не нарушая информативности обучающей выборки. Разбивать рекомендуется по следующему правилу: делим выборку на 3 равных части, 2 части используем в качестве обучающей, одну в качестве тестовой. Кроме того, обучающая выборка должна быть сгенерирована таким образом, чтобы минимизировать разницу между количеством представленных в ней объектов разных классов, т.е. abs(|$\{(x_i, y_i) \in X^l | y_i$ = -1$\}$| - |$\{(x_i, y_i) \in X^l | y_i$ = 1$\}$|) →min.

Параметр q по формуле: $w_i = q^i$, $i \in (0,1)$.

Номер файла: data5.csv.

## Результаты

Ход вычисления оптимальных значений k и q:

```
96.42857142857143 1 0.1
1
0.1
96.42857142857143
Оптимальное значение в обучающем наборе k = 1 при q = 0.1
```

Самый оптимальный вариант на обучающей модели получается при k = 1, q = 0.1. При таком выборе параметров вероятность правильной классификации равна 96.428571%.

# Код программы

```
import pandas as pd
import numpy as np
import math
import random
import ast

def split_csv(input_file, testing_file, learning_file):
    data = pd.read_csv(input_file)

    num_testing_rows = round(len(data) / 3)
    testing_data = data.iloc[:num_testing_rows]
    learning_data = data.iloc[num_testing_rows:]

    testing_data.to_csv(testing_file, index=False, header=False)
    learning_data.to_csv(learning_file, index=False, header=False)

def sort_learning_data(learning_file):
    data = pd.read_csv(learning_file, header=None)

    sorted_data = data.sort_values(by=[data.columns[0], data.columns[1]],
ignore_index=True)

    sorted_data.to_csv(learning_file, index=False, header=False)

def calculate_distances_index_weight(learning_file):
    data = pd.read_csv(learning_file, header=None)

    distances = []

    for i in range(len(data)):
        point_distances = []

        for j in range(len(data)):

            x1 = data.iloc[i, 0]
            y1 = data.iloc[i, 1]
            x2 = int(data.iloc[j, 0])
            y2 = int(data.iloc[j, 1])
            classes = int(data.iloc[j, 2])


            distance = math.sqrt(math.pow((x2 - x1), 2) + math.pow((y2 - y1),
2))
            point_distances.append((x2, y2, distance, classes))
        sorted_point_distances = list(sorted(point_distances, key=lambda x:
x[2]))
        for k, point_distance in enumerate(sorted_point_distances):
            x2, y2, distance, classes = point_distance
            index = k
            if distance == 0:
                index = 0
            sorted_point_distances[k] = [x2, y2, index, distance, classes]
        distances.append(sorted_point_distances)
    return distances


def knn(distances):
```

```
    best_k = 0
    best_q = 0
    best_degree = 0
    print(type(distances))

    for q in range(1, 5):
        for k in range(1, len(distances)):
            correct_answers = 0
            predicted = -1

            for i in range(len(distances)):
                accuracy_0 = 0
                accuracy_1 = 0

                if (k + 2) > len(distances):
                    break

                for j in range(1, k + 2):
                    distances[i][j].append(math.pow(q / 10,
distances[i][j][2]))

                    if distances[i][j][4] == 0:
                        accuracy_0 += distances[i][j][5]
                    else:
                        accuracy_1 += distances[i][j][5]

                if accuracy_0 > accuracy_1:
                    predicted = 0
                elif accuracy_0 < accuracy_1:
                    predicted = 1

                if predicted == distances[i][0][4]:
                    correct_answers += 1

            accurany = (correct_answers / len(distances)) * 100

            if best_degree < accurany:
                best_degree = accurany
                best_k = k
                best_q = q / 10
                print(f"{best_degree} {best_k} {best_q}")

    print(best_k)
    print(best_q)
    print(best_degree)
    return best_k, best_q


def save_distances_to_csv(distances, output_file):
    distances_df = pd.DataFrame(distances)

    distances_df.to_csv(output_file, index=False, header=False, sep=",")

def main():
    input_file = "data5.csv"
    testing_file = "testing.csv"
    learning_file = "learning.csv"
    output_file = "output.csv"
    #split_csv(input_file, testing_file, learning_file)
```

```
    distances = calculate_distances_index_weight(testing_file)
    target_index = 0
    k, q = knn(distances)
    print(f"Оптимальное значение в обучающем наборе k = {k} при q = {q}")

if __name__ == '__main__':
    main()
```