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 () и соответствующим ему классом ().

Задание:

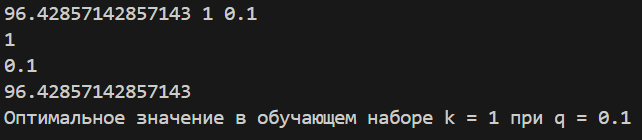
Суть лабораторной работы заключается в написании классификатора на основе метода k ближайших соседей. Данные из файла необходимо разбить на две выборки, обучающую и тестовую, согласно общепринятым правилам разбиения. На основе этих данных необходимо обучить разработанный классификатор и протестировать 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(|{() | = -1}| - |{() | = 1}|) →min.

Параметр q по формуле: = , .

Номер файла: data5.csv.

**Результаты**

Ход вычисления оптимальных значений k и q:



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()