1. **Analisis (Identifikasi Masalah)**

Pada tugas program 3 diberikan himpunan data berisi 800 data yang memiliki 5 atribut input (X1, X2, X3, X4, dan X5) dan 1 output yang memiliki 4 kelas/label (0, 1, 2, dan 3) dan terdapat 200 *data* *testing* yang belum diketahui kelasnya. Untuk menentukan kelas pada data Test tersebut tugas program 3 menggunakan K-Nearest Neighbor (KNN). Himpunan data tersebut merupakan data training yang mengharuskan untuk dicari nilai K terbaik untuk menentukan nilai K yang akan digunakan untuk menentukan kelas pada data Testing.

1. **Desain Program**

*K*-Nearest Neighbor adalah salah satu metode *learning* yang digunakan dalam klasifikasi. KNN bekerja dengan mengelompokkan data baru berdasarkan jarak dengan beberapa data *K* terdekat. *K-Nearest Neighbour* berdasarkan konsep pembelajaran berdasarkan data training yang telah diberikan.

Data *training* dideskripsikan dengan atribut numerik *n* dimensi. Jarak antara data dapat dihitungan dengan data *training* dihitung dengan cara mengukur jarak antara titik yang merepresentasikan data dengan semua titik yang merepresentasikan data *training* dengan rumus *Euclidean Distance*

Dimana P dan Q adalah titik pada ruang vektor *n* dimensi sedangkan adalah besaran skalar untuk dimensi ke *i* dalam ruang vektor *n*  dimensi.

* **Data yang diberikan dan Inputan**
* Data Training
* Data Testing

### Proses Mencari Nilai K

Data Training yang diberikan yakni sejumlah 800 data yang akan dihitung dengan menggunakan *split* data perjumlah data *training*. Pada algoritma program yang dikerjakan yakni menggunakan 100 sebagai data *split* untuk dipisah per 800 data *Training*. Jadi akan dihitung 100/700 per data *Training* dan seterusnya untuk mengetahui nilai k dengan menggunakan akurasi mana yang terbaik yang akan dijadikan nilai K untuk menentukan kelas pada data Testing yakni sejumlah 200 data.

Nilai *K* yang paling tepat tidak bisa langsung didapatkan, karena (jika *K* dipilih random) tidak ada yang dapat memastikan apakah klasifikasi yang dihasilkan oleh *K* tertentu sudah benar (tervalidasi). Oleh karena itu, dibutuhkan suatu himpunan data yang sudah diketahui kelas dari data *Training*. Setelah itu bisa dipastikan nilai yang akurat dari model (dalam kasus *K-NN, K* adalah model), dilakukan perulangan dari 1 sampai seperempat data train (800 data). Perulangan ini bertujuan untuk *“mencoba”* berapa akurasi yang akan dihasilkan jika *K* = 1, *K* = 2, *K* = 3, hingga *K*=199. *K* dengan akurasi terbesar akan menjadi ***K terbaik.***

Pseudo code beserta komentar yang akan menjelaskan cara kerja dari KNN untuk tugas program 3 telah dilampirkan.

1. **Output Program**

Berikut ini adalah merupakan hasil output program dengan menggunakan Bahasa Python:

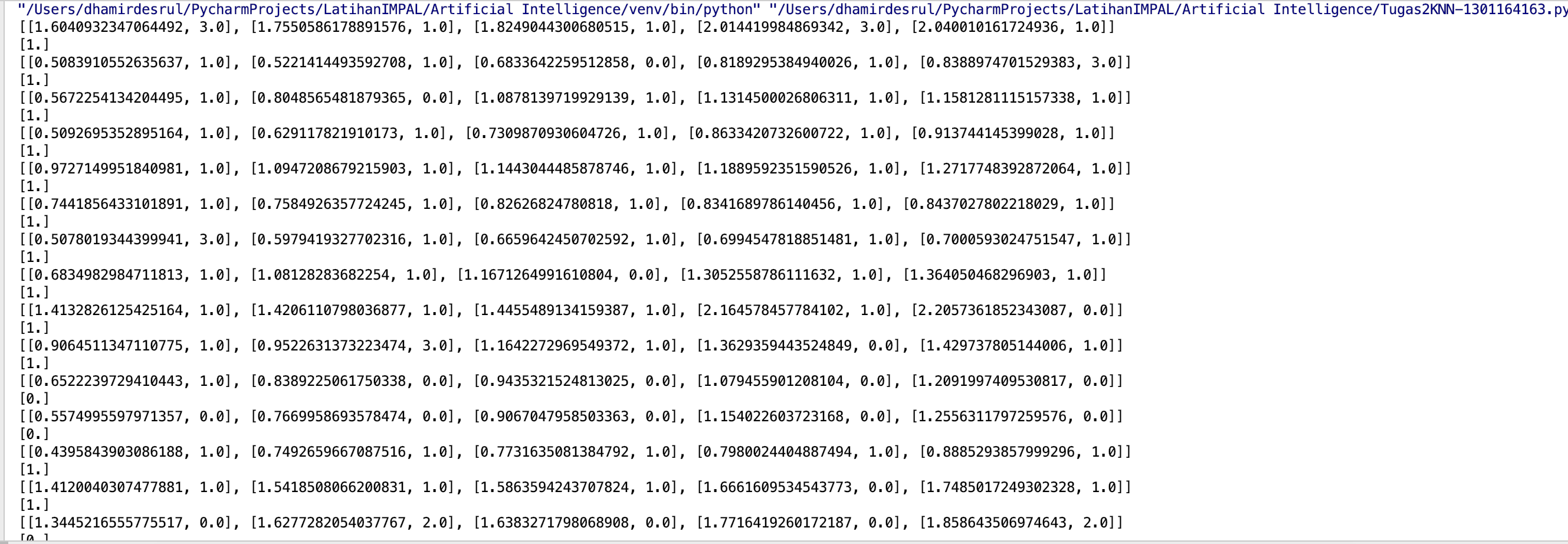


Figure 1. Hasil keluaran kelas yang pada data Testing



Figure 2. Hasil akurasi yang telah dicari berdasarkan split data untuk menentukan nilai K terbaik

Lampiran Pesudo Code

**Mencari nilai K terbaik**

**import** csv

**import** math

**from** scipy **import** stats

**import** numpy

**function** hitungpanjangakurasi(sumakurasi)->real

**algoritma**

**->** sum(sumakurasi)/len(sumakurasi) *#menghitung hasil akurasi berdasarkan split data*

**function** hitungakurasi(z, validasiDataTrain)->real

**algoritma**

**->** (z/len(validasiDataTrain)) \* 100 *#menghitung akurasi yakni dengan jumlah banyaknya data dengan data train yang sudah di validasi dengan kengalikan 100 persen*

**function** pengurangan(x,y) ->real *#pengurangan yang ditujukan untuk mencari selisih jarak*

**kamus**

tot: real

**algoritma**

tot <- (x-y)\*\*2

**->** tot

**function** jumlahtotdistance(dataTest, dataTrain)->real *#menggunakan prosedur pengurangan yang nantinya akan menghitung berdasarkan nilai x1 hingga x5 pada data test dan data train yang nantinya akan dimasukkan ke dalam rumus euclidan*

**kamus**

**function** pengurangan(x,y) -> array of list

hasil : array of list

**algoritma**

xsatu <- pengurangan(dataTest[1], dataTrain[1])

xdua <- pengurangan(dataTest[2], dataTrain[2])

xtiga <- pengurangan(dataTest[3], dataTrain[3])

xempat <- pengurangan(dataTest[4], dataTrain[4])

xlima <- pengurangan(dataTest[5], dataTrain[5])

hasil <- math.sqrt(xsatu + xdua + xtiga + xempat + xlima)

**->** hasil

**function** dataTrain(path)->array of list:

**kamus**

data: array of list

**algoritma**

**with** open(path) **as** csvfile:

spamreader <- csv.reader(csvfile)

next(spamreader, **None**)

**for** row **to** spamreader:

data <- (

[int(row[0]), float(row[1]), float(row[2]), float(row[3]), float(row[4]), float(row[5]), float(row[6])])

*# print(data)*

**->** data

**function** cariKelas(KNN, datax)->real *#menentukan kelas mana yang lebih dominan dari hasil perhitungan euclidan*

**kamus**

a: array of real

**algoritma**

**for** i **to**(KNN) **do**

a <- (datax[i][1])

**->** stats.mode(a)[0] *#memakai fungsi mod untuk mencari nilai terbanyak dari kelas yang telah dilakukan splitting*

**function** p(prediksi, test, z)->real *#menghitung nilai prediksi yang ada di kolom 6 pada dataTRrain yakni penfunctioninisian kelasnya untuk mengecek apakah nilai tersebut sama jika ya akan diberikan spesifikasi kelas yang sesuai dengan data training yang sudah dihitung*

**if** (prediksi = test[6]) **then**

z <- z + 1

**->** z

**function** splittingDataTrain()->array of list

**kamus**

jk : array of list

num\_split, fork, i, j,z: array of integer

panjang: array of real

sumakurasi : array of list

validasiDataTrain : array of list

dataHasil : array of list

y : real

**function** jumlahtotdistance(dataTest, dataTrain)->real

**function** p(prediksi, test, z)->real

**function** cariKelas(KNN, datax)->real

**function** hitungpanjangakurasi(sumakurasi)->real

**function** hitungakurasi(z, validasiDataTrain)->real

**algoritma**

dataT <- dataTrain(**'DataTrain\_Tugas3\_AI.csv'**) *#menfunctioninisikan dataTRain ke dalam variable dataT*

num\_split <- 100 *#data yang akan dibagi yang ditujukan untuk menghitung data per data total hingga mendapatkan akurasi terbaik*

panjang <- len(dataT)-num\_split *#pengurangan hasil pembanding yang akan dibandingkan dengan num\_split yang telah ditentukan*

**for** fork **to**(1,200) **do** *#melakukan perulangan yang khendaknya sesuai dengan jumlah data test untuk menentukan k terbaik dari k <- 1 hingga 199*

**for** i **to**(0, panjang, num\_split) **do** *#perbandingan dilakukan dengan menggunakan perulangan yang nantinya perulangan sendiri itu akan melakukan kelipatan berdasarkan jumlah data awal yang telah di train lalu akan terus menambahkan kelipatannya berdasarkan data split yang awal sudah ditentukan*

z <- 0

dTrain <- list(dataT[1:]) *#train data berdasarkan list yang telah tersisia*

validasiDataTrain <- dTrain[i: i +10] *#validasi dari dTrain*

**for** j **to** (i+10) **do** *#dikarenakan data yang telah di train di awal tidak boleh di lakukan train kembali maka data yang telah di train akan dihapus di dalam array*

dTrain.pop(j)

j = j-1

i = i-1

**for** test **to** validasiDataTrain **do** *#perhitungan jarak yang menggunakan iterasi test yang menentukan akurasinya*

**for** train **to** dTrain **do**

y <- jumlahtotdistance(test,train)

dataHasil<-([y, train[6]])

dataHasil.sort(key=**lambda** x: x[0]) *#melakukan sorting berdasarkan indeks pertama pada file data hasil yang sudah ditentukan dengan jumlah jarak pada variable y*

prediksi <- cariKelas(fork, dataHasil)

z <- p(prediksi, test, z)

iniakurasi <- hitungakurasi(z, validasiDataTrain)

sumakurasi<-iniakurasi

x <- (hitungpanjangakurasi(sumakurasi))

jk <- x

print(**'k =‘**, fork, **'akurasi= '**, x)

**->** jk

**program knnTraing**

**kamus**

jk : array of list

**algoritma**

jk <- splittingDataTrain()

numpy.savetxt(**'akurasi.csv'**, jk, delimiter<-**','**, fmt<-**'%s'**) *#nilai akurasi akan dimasukkan ke dalam cs untuk mengetahui nilai akurasi terbaik dari k <-1 hingga 199*

Program Utama

**import** csv

**import** math

**from** scipy **import** stats

**import** numpy

**import** KNN\_TRAINING

**function** pengurangan(x,y)->real *#pengurangan yang digunakan untuk menghitung selisih jarak yang nantinya akan dimasukkan ke dalam rumus euclidian*

tot <- (x-y)\*\*2

**->** tot

**function** jumlahtotdistance(dataTest, dataTrain)-> array of list *#perhitungan euclidian*

**kamus**

hasil : array of list

**algoritma**

xsatu <- pengurangan(dataTest[1], dataTrain[1])

xdua <- pengurangan(dataTest[2], dataTrain[2])

xtiga <- pengurangan(dataTest[3], dataTrain[3])

xempat <- pengurangan(dataTest[4], dataTrain[4])

xlima <- pengurangan(dataTest[5], dataTrain[5])

hasil <- math.sqrt(xsatu + xdua + xtiga + xempat + xlima)

**->** hasil

**function** dataTrain(path) -> array of list *#untuk memasukkan nilai data train ke dalam variable*

**kamus**

data : array of list

**algoritma**

**with** open(path) **as** csvfile:

spamreader <- csv.reader(csvfile)

next(spamreader, **None**)

**for** row **to** spamreader:

data<-(

[int(row[0]), float(row[1]), float(row[2]), float(row[3]), float(row[4]), float(row[5]), float(row[6])])

*# output(data)*

**->** data

**function** dataTest(path) -> array of list *#untuk memasukkan nilai data test ke dalam variable*

**kamus**

data : array of list

**algoritma**

**with** open(path) **as** csvfile:

spamreader <- csv.reader(csvfile)

next(spamreader, **None**)

**for** row **to** spamreader:

data<-(

[int(row[0]), float(row[1]), float(row[2]), float(row[3]), float(row[4]), float(row[5]), row[6]])

*# output(data)*

**->** (data)

*# function vote(distance, dataTrain, KNN):*

*# terpilih <- 0*

*# tidak <- 0*

*# for i to(0, KNN):*

*# if dataTrain[distance[i][1]][6] <-<- "1":*

*# terpilih +<- 1*

*# else:*

*# tidak +<- 1*

*# if (terpilih>tidak):*

*# -> 1*

*# else:*

*# -> 0*

**function** cariKelas(KNN, datax) -> array of list *#menentukan kelas untuk data yang telah ditentukan perhitunga euclidian*

**kamus**

a : arrauy of list

**algoritma**

**for** i **to** range(KNN) do

a<-(datax[i][1])

**->** stats.mode(a)[0] *#menggunakan libary statistik untuk menentukan nilai terbesar yang akan dijadikan kelas yang ditentukan untuk data test*

**program utama**

**kamus**

x : array of list

dTrain: array of list

dTest: array of list

dataHasil: array of list

y : real

i, j, k: integer

**algoritma**

k <- 13 *#nilai k yang sudah ditentukan dalam knn\_training.py*

dTrain <- dataTrain(**'DataTrain\_Tugas3\_AI.csv'**)

dTest <- dataTest(**'DataTest\_Tugas3\_AI.csv'**)

**for** i **to**(200) do *#200 berdasarkan data yang berada pada dTest*

dataHasil<- []

**for** j **to**(800) do *#800 berdasarkan data training*

y <- jumlahtotdistance(dTest[i],dTrain[j]) *#menghitung dengan menggunakan rumus euclidan*

dataHasil<-([y, dTrain[j][6]]) *#memasukkan nilai pada indeks ke 6 yakni kelas ke dalam data hasil*

dataHasil.sort(key<-**lambda** x: x[0])

output(dataHasil[:5]) *#mengeluarkan nilai dataHasil dari awal hingga indeks ke 5*

x<-(cariKelas(k, dataHasil))

output(cariKelas(k, dataHasil))

numpy.savetxt(**'TebakanTugas3.csv'**, x, delimiter<-**','**, fmt<-**'%s'**)