

A. 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.

B. 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 dihitung dengan data *training* dihitung dengan cara mengukur jarak antara titik yang merepresentasikan data dengan semua titik yang merepresentasikan data *training* dengan rumus *Euclidean Distance*

$$Distance(P, Q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

Dimana P dan Q adalah titik pada ruang vektor *n* dimensi sedangkan p_i dan q_i 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.

C. Output Program

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

```
"/Users/dhamirdesru/PycharmProjects/LatihanIMPAL/Artificial Intelligence/venv/bin/python" "/Users/dhamirdesru/PycharmProjects/LatihanIMPAL/Artificial Intelligence/Tugas2KNN-1301164163.py"
[[1.6040932347064492, 3.0], [1.750508617891576, 1.0], [1.8249044300600515, 1.0], [2.014419984069342, 3.0], [2.040010161724936, 1.0]]
[1.]
[[0.5083910552635637, 1.0], [0.5221414493592708, 1.0], [0.6833642259512858, 0.0], [0.8189295384940026, 1.0], [0.8388974701529383, 3.0]]
[1.]
[[0.5672254134204495, 1.0], [0.8048565481879365, 0.0], [1.0878139719929139, 1.0], [1.1314500026006311, 1.0], [1.1581281115157338, 1.0]]
[1.]
[[0.5092695352895164, 1.0], [0.629117821910173, 1.0], [0.7309870930604726, 1.0], [0.8633420732600722, 1.0], [0.913744145399028, 1.0]]
[1.]
[[0.9727149951840981, 1.0], [1.0947208679215903, 1.0], [1.1443044485878746, 1.0], [1.1889592351590526, 1.0], [1.2717748392872064, 1.0]]
[1.]
[[0.7441856433101891, 1.0], [0.7584926357724245, 1.0], [0.82626824780818, 1.0], [0.8341689786140456, 1.0], [0.8437027802218029, 1.0]]
[1.]
[[0.5078019344399941, 3.0], [0.5979419327702316, 1.0], [0.6659642450702592, 1.0], [0.6994547818851481, 1.0], [0.7000593024751547, 1.0]]
[1.]
[[0.6834982984711813, 1.0], [1.08128283682254, 1.0], [1.1671264991610804, 0.0], [1.3052558786111632, 1.0], [1.364050468296903, 1.0]]
[1.]
[[1.4132826125425164, 1.0], [1.4206110798036877, 1.0], [1.4455489134159387, 1.0], [2.164578457784102, 1.0], [2.2057361852343087, 0.0]]
[1.]
[[0.9064511347110775, 1.0], [0.9522631373223474, 3.0], [1.1642272969549372, 1.0], [1.3629359443524049, 0.0], [1.429737805144006, 1.0]]
[1.]
[[0.6522239729410443, 1.0], [0.8389225961750338, 0.0], [0.9435321524813025, 0.0], [1.079455901208104, 0.0], [1.2091997409530817, 0.0]]
[0.]
[[0.5574995597971357, 0.0], [0.7669958693578474, 0.0], [0.9067047958503363, 0.0], [1.154022603723168, 0.0], [1.2556311797259576, 0.0]]
[0.]
[[0.4395843903086188, 1.0], [0.7492659667007516, 1.0], [0.7731635081384792, 1.0], [0.7900024404807494, 1.0], [0.8885293857999296, 1.0]]
[1.]
[[1.4120040307477881, 1.0], [1.5418508066200831, 1.0], [1.5863594243707824, 1.0], [1.6661609534543773, 0.0], [1.7485017249302328, 1.0]]
[1.]
[[1.3445216555775517, 0.0], [1.6277282054037767, 2.0], [1.6383271798068908, 0.0], [1.7716419260172187, 0.0], [1.858643506974643, 2.0]]
[0.]
```

Figure 1. Hasil keluaran kelas yang pada data Testing

```
Run: KNN_TRAINING
"/Users/dhamirdesru/PycharmProjects/LatihanIMPAL/Artificial Intelligence/venv/bin/python" "/Users/dhamirdesru/PycharmProjects/LatihanIMPAL/Artificial Intelligence/KNN_TRAINING.py"
k = 1 akurasi = 84.28571428571429
k = 2 akurasi = 87.14285714285714
k = 3 akurasi = 85.71428571428571
k = 4 akurasi = 82.85714285714286
k = 5 akurasi = 84.28571428571429
k = 6 akurasi = 85.71428571428571
k = 7 akurasi = 90.0
k = 8 akurasi = 90.0
k = 9 akurasi = 90.0
k = 10 akurasi = 88.57142857142857
k = 11 akurasi = 90.0
k = 12 akurasi = 90.0
k = 13 akurasi = 91.42857142857143
k = 14 akurasi = 91.42857142857143
k = 15 akurasi = 91.42857142857143
k = 16 akurasi = 91.42857142857143
k = 17 akurasi = 91.42857142857143
k = 18 akurasi = 91.42857142857143
k = 19 akurasi = 91.42857142857143
k = 20 akurasi = 91.42857142857143
k = 21 akurasi = 91.42857142857143
k = 22 akurasi = 90.0
k = 23 akurasi = 90.0
k = 24 akurasi = 90.0
```

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 pengalihan 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 data TRrain 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 jumahtotdistance(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 pembandingan 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 in 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 : array of list

algoritma

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

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

-> stats.mode(a)[0] *#menggunakan library 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')
```