LAPORAN TUGAS PROGRAM 3

ARTIFICIAL INTELLIGENCE

k-Nearest Neighbor

Oleh

Nama : I Putu Indra Aristya

NIM : 1301154219

Kelas : IF39-09

Mata Kuliah : Kecerdasan Buatan

Universitas Telkom

BANDUNG

2017

I. Deskripsi Masalah

Dalam tugas program-3 ini, kita diminta untuk menentukan sebuah kebenaran (hoax atau bukan) dari suatu berita dengan 4 atribut linguistik, yaitu berdasar Like, Provokasi, Komentar dan Emosi. Metode yang digunakan untuk menentukan kebenaran berita tersebut adalah *learning* menggunkan k-NN (*k-nearest neighbor*). Untuk menyelesaikan tugas program ini diberikan 2 buat tipe data, yaitu 4000 data *training* dan 1000 data *testing*. Data *training* digunakan untuk melatih/sebagai pembanding dengan data *testing* atau data yang ingin dicari kebenarannya.

II. Landasan Teori dan Rancangan Program

Metode k-NN (*k-nearest neighbor*) adalah salah satu metode mengajarkan komputer (*learning*) yang biasa juga disebut dengan *lazy learning*. Disebut seperti itu karena metode ini membandingkan data yang ingin dicari dengan semua data latih yang sudah memiliki hasil dan diklasifikasikan dalam kelas. Dalam kasus ini, kelas yang ada hanya 2, yaitu kelas hoax (atribut bernilai 1) dan kelas tidak hoax (atribut bernilai 0).

Konsep dari k-NN ini adalah sesuai dengan arti dari k-NN jika diterjemahkan ke dalam bahasa Indonesia, yaitu tetangga yang terdekat. Tetangga terdekat tersebut di dapat dari menghitung jarak dari data yang ingin dicari dengan semua data latih. Pasangan data dengan jarak terkecil memiliki kemungkinan kemiripan lebih besar. Penghitungan jarak dapat digunakan dengan perhitungan *Manhattan* atau *Ecluidean.* Pada tugas ini, saya menggunakan perhitungan *Manhattan* dengan rumus sebagai berikut.

$$d_i(x_i, y_i) = \sum_i |x_i - y_i|$$

dengan x dan y adalah nilai dari masing-masing atribut dari data latih dan data *test*. Karena atribut pada tugas program ini lebih dari 1, maka x dan y mewakili setiap data latih dan data *test*-nya kemudian dijumlahkan dengan atribut lain.

Variabel *k* pada k-NN mewakili suatu nilai integer yang berguna untuk mengambil *k* banyak data yang terkecil untuk dilakukan klasifikasi.

Dari landasan teori tersebut, dapat diringkas algoritma k-NN ini menjadi:

1. Hitung jarak dari data *test* dengan semua data latih yang ada

2. Pilih *k* nilai terkecil dari jarak yang sudah dihitung tadi
3. Lihat label kebenaran (1/0) dari data latih dengan indeks dari *k* nilai terkecil tersebut
4. Jika label 1 lebih banyak maka data tersebut hoax dan jika label 0 lebih banyak maka data tersebut tidak hoax
5. Lakukan langkah 1 sampai 4 tadi untuk semua data *test*

Pada tugas program ini, saya menggunakan perhitungan jarak Manhattan karena telihat lebih sederhana dan penentuan nilai *k* adalah dengan random. Jadi saya mencoba setiap kemungkinan nilai *k* yang bisa digunakan kemudian mencoba menjalankan program. Dari hasil dengan membagi data latih menjadi data latih (3500 data pertama dari data latih) dan data validasi (500 data setelahnya dari data latih) ditentukan label dari data validasi dengan k = 59 didapat akurasi 70.8%.

III. *Code* Program

Dalam tugas program-1 ini, saya menggunakan bahasa pemrograman *Python* dengan IDE *PyCharm CE*. Berikut adalah *code* program yang sudah saya buat.

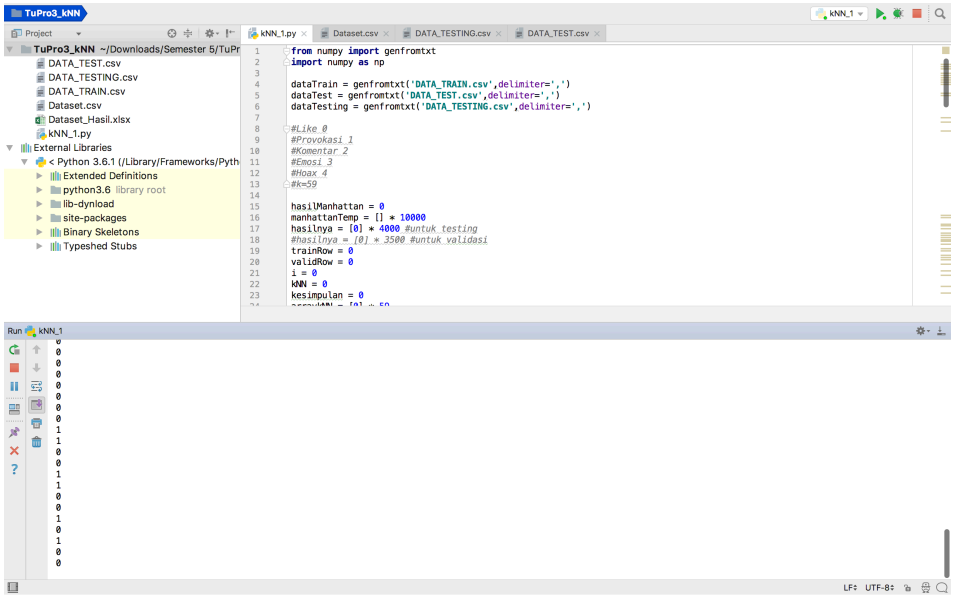
```
from numpy import genfromtxt
import numpy as np

dataTrain = genfromtxt('DATA_TRAIN.csv',delimiter=',')
dataTest = genfromtxt('DATA_TEST.csv',delimiter=',')
dataTesting = genfromtxt('DATA_TESTING.csv',delimiter=',')

#Like 0
#Provokasi 1
#Komentar 2
#Emosi 3
#Hoax 4
#k=59

hasilManhattan = 0
manhattanTemp = [] * 10000
hasilnya = [0] * 4000 #untuk testing
#hasilnya = [0] * 3500 #untuk validasi
trainRow = 0
validRow = 0
i = 0
kNN = 0
kesimpulan = 0
arraykNN = [0] * 59
countHoax = 0
countTdk = 0
```

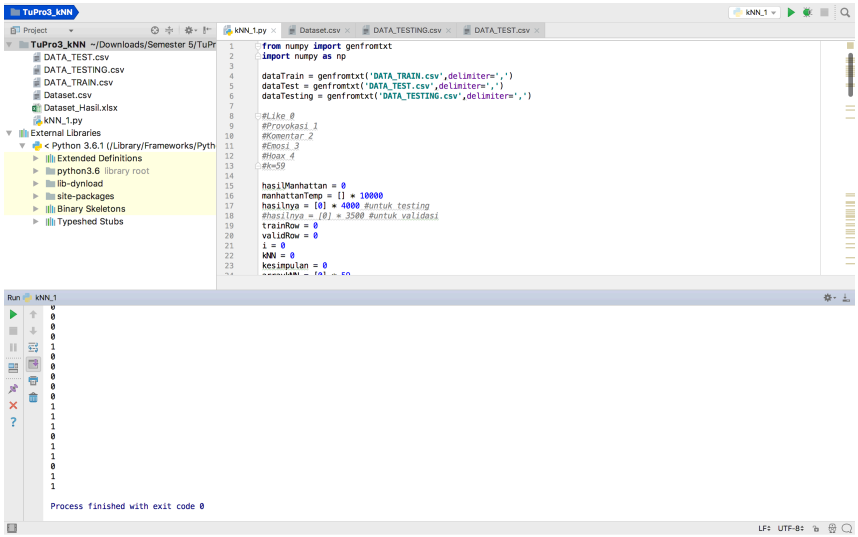
```
def manhattan(x,y):
    hasil = x - y
    if (hasil < 0):
        hasil = -1*hasil
    return hasil

#---------CODE UNTUK VALIDASI---------#
# for validRow in range(0,500):
#     for trainRow in range(0,3499):
#         like = manhattan((dataTrain[trainRow, 0]), (dataTest[validRow, 0]))
#         provo = manhattan((dataTrain[trainRow,1]),(dataTest[validRow,1]))
#         komen = manhattan((dataTrain[trainRow,2]),(dataTest[validRow,2]))
#         emosi = manhattan((dataTrain[trainRow,3]),(dataTest[validRow,3]))
#         #print(i)
#         hasilnya[i] = like + provo + komen + emosi
#         i=i+1
#         if (i == 3499):
#             arrayManhattan = np.array(hasilnya)
#             indexSorted = np.argsort(arrayManhattan)
#             list.sort(hasilnya)
#             # print(arrayManhattan)
#             # print(indexSorted)
#
#             while (kNN < 58):
#                 arraykNN[kNN] = dataTrain[indexSorted[kNN],4]
#                 #print(arraykNN[kNN])
#                 if (arraykNN[kNN] == 0.0):
#                     countTdk = countTdk + 1
#                 elif (arraykNN[kNN] == 1.0):
#                     countHoax = countHoax + 1
#                 kNN = kNN + 1
#             if (countTdk > countHoax):
#                 ht = 0
#             elif (countTdk < countHoax):
#                 ht = 1
#             #print("KNN: ",kNN)
#             #print(countTdk, " ",countHoax," ",ht)
#             print(ht)
#
#             kNN = 0
#             i = 0
#             countHoax = 0
#             countTdk = 0

#print(indexSorted)
#print(validRow)


#---------CODE UNTUK TESTING---------#
for testRow in range(0,1000):
    for trainRow in range(0,3999):
        like = manhattan((dataTrain[trainRow, 0]), (dataTesting[testRow, 0]))
        provo = manhattan((dataTrain[trainRow,1]),(dataTesting[testRow,1]))
        komen = manhattan((dataTrain[trainRow,2]),(dataTesting[testRow,2]))
        emosi = manhattan((dataTrain[trainRow,3]),(dataTesting[testRow,3]))
        hasilnya[i] = like + provo + komen + emosi #perhitungan distance
ditampung pada array hasilnya.
        i=i+1
        if (i == 3999):
            arrayManhattan = np.array(hasilnya)
            indexSorted = np.argsort(arrayManhattan) #melakukan sorting
```

```
ascending pada indeks dari data distance
            list.sort(hasilnya) #melakukan sorting ascending pada data distance

            while (kNN < 58):
                arraykNN[kNN] = dataTrain[indexSorted[kNN],4] #menyimpan label
kelas dari 59 data terkecil
                if (arraykNN[kNN] == 0.0):
                    countTdk = countTdk + 1
                elif (arraykNN[kNN] == 1.0):
                    countHoax = countHoax + 1
                kNN = kNN + 1
            if (countTdk > countHoax):
                kesimpulan = 0
            elif (countTdk < countHoax):
                kesimpulan = 1

            print(kesimpulan)

            kNN = 0 #reset variabel yang digunakan dalam program untuk
perulangan
            i = 0
            countHoax = 0
            countTdk = 0
```

IV. Hasil Keluaran Program

Berikut adalah hasil keluaran dari program yang dijalankan




TuPro3_kNN
kNN_1
Project
kNN_1.py
Dataset.csv
DATA_TESTING.csv
DATA_TEST.csv
TuPro3_kNN ~/Downloads/Semester 5/TuPr
DATA_TEST.csv
DATA_TESTING.csv
DATA_TRAIN.csv
Dataset.csv
Dataset_Hasil.xlsx
kNN_1.py
External Libraries
< Python 3.6.1 (/Library/Frameworks/Pyth
Extended Definitions
python3.6 library root
lib-dynload
site-packages
Binary Skeletons
Typeshed Stubs
from numpy import genfromtxt
import numpy as np
dataTrain = genfromtxt('DATA_TRAIN.csv',delimiter=',')
dataTest = genfromtxt('DATA_TEST.csv',delimiter=',')
dataTesting = genfromtxt('DATA_TESTING.csv',delimiter=',')
#Like 0
#Provokasi 1
#Komentar 2
#Emosi 3
#Hoax 4
#k=59
hasilManhattan = 0
manhattanTemp = [] * 10000
hasilnya = [0] * 4000 #untuk testing
#hasilnya = [0] * 3500 #untuk validasi
trainRow = 0
validRow = 0
i = 0
kNN = 0
kesimpulan = 0
Run kNN_1
0
0
0
0
1
0
0
0
0
0
1
1
1
0
1
1
0
1
1
Process finished with exit code 0
LF: UTF-8: