2017

k-Nearest Neighbor HoaxOrNot

Laporan Tugas Program 3 Kecerdasan Buatan

Dosen Pengampu : UNTARI NOVIA WISESTY, S.T., M.T.

Oleh :

Fakhri Fauzan

1301154374

IF – 39 – 10

12/2/2017

1. Deskripsi Kasus

Algoritma k-nearest neighbor (k-NN atau KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Algoritma kNN sangatlah sederhana, bekerja berdasarkan jarak terpendek dari query instance ke training sample untuk menentukan KNN-nya. Training sample diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi training sample. Untuk menentukan jarak antar node digunakan rumus *Euclidian Distance* sebagai berikut :

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Dalam kasus yang diberikan data Training dan Validasi dalam file excel. Jumlah data yang diberikan adalah 4000 data Training dan 1000 data Testing. Data tersebut merupakan data set untuk klasifikasi berita Hoax atau Tidak. Terdapat 4 atribut yang menentukan klasifikasi ini diantaranya adalah Like, Provokasi, Komentar, Emosi. Sehingga rumus Euclidian Distance untuk kasus tersebut adalah :

$$D(p,q) = \sqrt{(q_{like} - p_{like})^2 + (q_{provokasi} - p_{provokasi})^2 + (q_{komentar} - p_{komentar})^2 + (q_{emosi} - p_{emosi})^2}$$

2. Rancangan Metode

Untuk menyelesaikan permasalahan diatas, diperlukan tahap-tahap sebagai berikut :

1. Dataset Training

   Dalam Kasus ini dataset Training disediakan 4000 data dalam file data.xlsx pada Sheet 'Data Training'.
2. Dataset Testing

   Dalam Kasus ini dataset Training disediakan 1000 data dalam file data.xlsx pada Sheet 'Data Testing'.
3. Menghitung Eulidian Distance

   Setiap data testing akan dihitung nilai D-nya terhadap seluruh data training yang ada, nantinya nilai D akan disimpan dalam sebuah Array.

4. Menentukan nilai K

   Dalam menentukan nilai K, saya menggunakan metode bruteforce algoritma di running sebanyak jumlah data training / 2 karena hanya mengginakan nilai ganjil, yaitu dengan mendapatkan K = 3. Nantinya nilai K ini akan digunakan untuk memilih sebanyak K data dari jarak terpendek dari data testing terhadap seluruh node.

5. Menklasifikasikan berdasarkan nilai K

   Setelah sebanyak data K telah di dapatkan, lalu data tersebut kita hitung jumlah klasifikasinya, dan nantinya akan mendapatkan dari klasifikasi sebuah data testing.

3. Simulasi Metode

Berikut adalah script program kNNHoaxOrNot menggunakan python.

```
import math
import random
import xlrd as x
import xlsxwriter as w

class kNN :
    def __init__(self, training, testing, k):
        self.training = training
        self.testing = testing
        self.hasilHoax = []
        self.hasilSort = []
        self.distance = []
        self.k = k

    def getClasification(self):
        for i in range(len(self.testing)):
            self.hasilSort = []
            self.distance = []
            cYa = 0
            cTidak = 0

            for j in range(len(self.training)):
                d = math.sqrt(((self.training[j][0] - self.testing[i][0]) ** 2) +
((self.training[j][1] - self.testing[i][1]) ** 2) + ((self.training[j][2] -
self.testing[i][2]) ** 2) + ((self.training[j][3] - self.testing[i][3]) ** 2))
                self.distance.append([i + 1, j + 1, self.training[j][4], d])

            self.hasilSort = sorted(self.distance, key=lambda distance:
distance[3], reverse=True)

            for y in range(self.k):
                if (self.hasilSort[y][2] == 1.0):
                    cYa += 1
                else:
                    cTidak += 1
```

```
            if (cYa > cTidak):
                klasifikasi = 1.0
            else:
                klasifikasi = 0.0

            self.hasilHoax.append([self.testing[i][0], self.testing[i][1],
self.testing[i][2], self.testing[i][3], klasifikasi])

    def getAccuracy(self):
        cSama = 0.0
        for x in range(len(self.testing)):
            if self.testing[x][4] == self.hasilHoax[x][4]:
                    cSama += 1
        akurasi = (cSama / len(self.testing)) * 100
        return akurasi

    def main(self):
        self.getClasification()
        return self.getAccuracy()

    def printResult(self):
        workbook = w.Workbook('result.xlsx')
        worksheet = workbook.add_worksheet('Result Testing')

        worksheet.write(0, 0, 'Berita')
        worksheet.write(0, 1, 'Like')
        worksheet.write(0, 2, 'Provokasi')
        worksheet.write(0, 3, 'Komentar')
        worksheet.write(0, 4, 'Emosi')
        worksheet.write(0, 5, 'Hoax')

        baris = 1
        kolom = 0
        berita = 4001
        for i in range(len(self.hasilHoax)) :
            worksheet.write(baris, kolom, 'B'+str(berita))
            worksheet.write(baris, kolom+1, self.hasilHoax[i][0])
            worksheet.write(baris, kolom+2, self.hasilHoax[i][1])
            worksheet.write(baris, kolom+3, self.hasilHoax[i][2])
            worksheet.write(baris, kolom+4, self.hasilHoax[i][3])
            worksheet.write(baris, kolom+5, self.hasilHoax[i][4])
            baris += 1
            berita += 1
        print ('data telah disimpan')
        workbook.close()


#Main Program
file = x.open_workbook('data.xlsx')
sheet = file.sheet_by_index(0)
count = sheet.nrows
sheetTest = file.sheet_by_index(1)
countTest = sheetTest.nrows

dataTraining = []
dataTesting = []
akurasi = []

k = 3

for i in range(1,count) :
    dataTraining.append(sheet.row_values(i,1))

for z in range(1,countTest) :
    dataTesting.append(sheetTest.row_values(z,1))
```

```
for j in range(k):
    validasi = []
    training = dataTraining
    random.shuffle(training)

    for n in range(1000) :
        validasi.append(training.pop())

    klasifikasi = kNN(training, validasi, k)
    result = klasifikasi.main()
    akurasi.append(result)
    print (result)

print ('Rata-rata : ' + str(sum(akurasi) / float(len(akurasi))))

#cekDataTesting
for x in range(1):
    klasTest = kNN(dataTraining, dataTesting, k)
    klasTest.main()
    klasTest.printResult()
```

Hasil Running ***Data Testing*** dan ***Data Training*** dari Program diatas sebanyak 5 kali untuk mendapatkan nilai akurasi sbb :

Running	Rata – Rata Akurasi	Screenshot
1	44.3	40.300000000000004 39.900000000000006 52.800000000000004 Rata-rata : 44.333
2	43.76	37.8 50.8 42.69999999999999 Rata-rata : 43.76
3	51.13	51.300000000000004 48.8 53.300000000000004 Rata-rata : 51.133
4	50.19	49.2 53.7 47.699999999999996 Rata-rata : 50.199
5	41.13	39.0 41.0 43.4 Rata-rata : 41.13
Total	230.51	
Rata – Rata Akurasi	46.102 %	

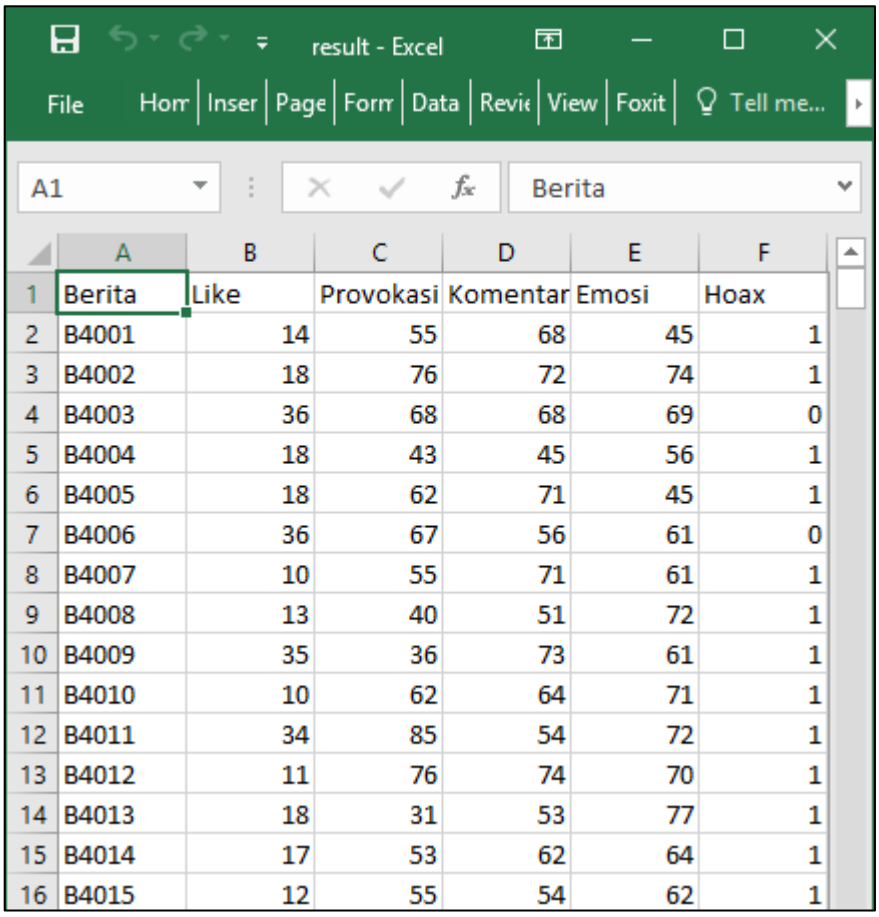
Akurasi Model = 46.102%.

4. Screenshot Output Program

Output dari program tersebut. Data hasil klasifikasi akan ditulis dalam file excel 'result.xlsx'

```
"C:\Users\Fakhri Fauzan\AppData\Local\Programs\Python\Python36-32\python.exe"
32.5
43.0
50.4
Rata-rata : 41.96666666666667
data telah disimpan

Process finished with exit code 0
```



	A	B	C	D	E	F
1	Berita	Like	Provokasi	Komentar	Emosi	Hoax
2	B4001	14	55	68	45	1
3	B4002	18	76	72	74	1
4	B4003	36	68	68	69	0
5	B4004	18	43	45	56	1
6	B4005	18	62	71	45	1
7	B4006	36	67	56	61	0
8	B4007	10	55	71	61	1
9	B4008	13	40	51	72	1
10	B4009	35	36	73	61	1
11	B4010	10	62	64	71	1
12	B4011	34	85	54	72	1
13	B4012	11	76	74	70	1
14	B4013	18	31	53	77	1
15	B4014	17	53	62	64	1
16	B4015	12	55	54	62	1