LAPORAN TUGAS PROGRAM

ARTIFICIAL INTELLEGENCE

K-Nearest Neighbor (KNN)

Dipersiapkan oleh :

Raden Muhammad Imam (1301154106)

Universitas Telkom

Bandung

2017

1. *K-Nearest Neighbor*

A. Pengertian *K-Nearest Neighbor*

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.

K-Nearest Neighbor berdasarkan konsep '*learning by analog*y'. Data *learning* dideskripsikan dengan atribut numerik *n*-dimensi. Tiap data *learning* merepresentasikan sebuah titik, yang ditandai dengan *c*, dalam ruang *n*-dimensi. Jika sebuah data *query* yang labelnya tidak diketahui diinputkan, maka *K-Nearest Neighbor* akan mencari *k* buah data *learning* yang jaraknya paling dekat dengan data *query* dalam ruang *n*-dimensi. Jarak antara data *query* dengan data *learning* dihitung dengan cara mengukur jarak antara titik yang merepresentasikan data *query* dengan semua titik yang merepresentasikan data *learning* dengan rumus *Euclidean Distance*.

Pada fase *training*, algoritma ini hanya melakukan penyimpanan vektor-vektor fitur dan klasifikasi data *training sample*. Pada fase klasifikasi, fitur – fitur yang sama dihitung untuk *testing data* (klasifikasinya belum diketahui). Jarak dari vektor yang baru ini terhadap seluruh vektor *training sample* dihitung, dan sejumlah *k* buah yang paling dekat diambil. Titik yang baru klasifikasinya diprediksikan termasuk pada klasifikasi terbanyak dari titik – titik tersebut.

Nilai *k* yang terbaik untuk algoritma ini tergantung pada data; secara umumnya, nilai *k* yang tinggi akan mengurangi efek *noise* pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi lebih kabur. Nilai *k* yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan cross-validation. Kasus khusus di mana klasifikasi diprediksikan berdasarkan data pembelajaran yang paling dekat (dengan kata lain, $k = 1$) disebut algoritma *nearest neighbor*.

Ketepatan algoritma *K-Nearest Neighbor* ini sangat dipengaruhi oleh ada atau tidaknya fitur-fitur yang tidak relevan, atau jika bobot fitur tersebut tidak setara dengan relevansinya terhadap klasifikasi. Riset terhadap algoritma ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur, agar performa klasifikasi menjadi lebih baik.

K buah data *learning* terdekat akan melakukan *voting* untuk menentukan label mayoritas. Label data *query* akan ditentukan berdasarkan label mayoritas dan jika ada lebih dari satu label mayoritas maka label data *query* dapat dipilih secara acak di antara label-label mayoritas yang ada.

B. Rancangan *K-Nearest Neighbor*

Anologinya seperti ini, bertanya pada tetangga untuk mendatangi ke sebuah pertemuan. Namun, kita tidak tahu tema dari pertemuan tersebut maupun kegiatan apa saja yang dilakukan dipertemuan tersebut. Kita tidak benar-benar tidak mengetahui pertemuan itu akan bermanfaat atau tidak untuk kita. Yang kita tahu, beberapa dari teman kita juga diundang ke acara yang sama. Dalam kondisi tersebut, apa yang kita lakukan?

1. Menentukan nilai K yang akan dipakai untuk mendapatkan tetangga terdekat.
2. Menghitung jarak eucledian dengan rumus:

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

3. Melakukan sorting terhadap hasil eucledian dan mengambil nilai terbaik dari K. Berarti nilai terbaik yang kita harapkan itu 3.
4. Mengambil value dari 3 terbaik.
5. Membandingkan ketika Hoax > Tidak Hoax maka Hoax hasilnya begitupun selanjutnya.

C. Prosedur Algoritma *K-Nearest Neighbor*

Berikut merupakan prosedur algoritma *K-Nearest Neightbor* :

1. Pertama memasukkan data excel ke code dengan cara import.

1301154106.py

```
import xlrd
import math

file   = xlrd.open_workbook("Dataset Tugas 3 AI 1718.xlsx")
sheet  = file.sheet_by_index(0)
sheet2 = file.sheet_by_index(1)
```

2. Ketika sudah memasukkan data excel, selanjutnya kita mendeklarasikan variabel.

```
x1          = []
x2          = []
y1          = []
y2          = []
k1          = []
k2          = []
c1          = []
c2          = []
n1          = []
n2          = []
Tempat_Hoax = []
```

3. Membuat fungsi menghitung jarak.

```
def Mencari_Jarak(x1 , x2 , y1 , y2 , k1 , k2 , c1 , c2) :
    a = (x1 - x2) ** 2 + (y1 - y2) ** 2 + (k1 - k2) ** 2 + (c1 - c2) ** 2
    Akar = math.sqrt(a)
    return Akar
```

4. Memasukkan data excel ke variabel.

```
#Untuk Data_Train
for i in range(1,sheet.nrows):
    x1.append(sheet.cell_value(i , 1))
    y1.append(sheet.cell_value(i , 2))
    k1.append(sheet.cell_value(i , 3))
    c1.append(sheet.cell_value(i , 4))
    n1.append(sheet.cell_value(i , 5))

#Untuk Data_Test
for i in range(1 , sheet2.nrows):
    x2.append(sheet2.cell_value(i , 1))
    y2.append(sheet2.cell_value(i , 2))
    k2.append(sheet2.cell_value(i , 3))
    c2.append(sheet2.cell_value(i , 4))
    n2.append(sheet2.cell_value(i , 5))
    Tempat_Hoax.append(sheet2.cell_value(i , 5))
```

5. Melakukan perulangan pada data test.

```
Tempat_Keseluruhan = []
for i in range(len(x2)):
    z          = []
    a          = []
    e          = []
    g          = []
    Hoax       = 0
    Tidak_Hoax = 0
```

6. Memanggil fungsi cari jarak dan melakukan sorting dan mengambil 3 terbaik.

```
    for j in range(len(x1)):
        a = Mencari_Jarak(x1[j] , x2[i] , y1[j] , y2[i] , k1[j] , k2[i] , c1[j] , c2[i])
        z.append(a)
    best = sorted(z)[0:3]
```

7. Melakukan perulangan sesuai dengan banyaknya best.

```
    for k in best:
        if n1[z.index(k)] == 1.0:
            Hoax          = Hoax + 1
        else:
            Tidak_Hoax    = Tidak_Hoax + 1
    if Hoax > Tidak_Hoax:
        n2 = 1.0
    else:
        n2 = 0.0
    Tempat_Keseluruhan.append(n2)
    print(n2)
count = 0
```

8. Melakukan metode searching untuk mencari banyaknya data yang benar.

```
for i in range(len(x2)):
    if(Tempat_Keseluruhan[i] == Tempat_Hoax[i]):
        count                    = count + 1
```

9. Menghitung berapa persen dari akurasi.

```
akurasi  = (count / len(x2)) * 100
print("Akurasi : " , akurasi , "%")
```

D. Hasil Test Akurasi