LAPORAN TUGAS PEMROGRAMAN 3
MATA KULIAH PENGANTAR KECERDASAN BUATAN

Disusun oleh :

Imam Rafiif Arrazaan (1301194152) IF4307

M Mirza Rizkiawan (1301194330) IF4307

Arvinda Dwi Safira (1301190083) IF4307

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM BANDUNG

BANDUNG

2021

Daftar Isi

A. Strategi Penyelesaian Masalah

1. Penggunaan Ukuran Jarak Yang Digunakan

Terdapat banyak formula yang dapat digunakan untuk menghitung jarak atribut numerik antara dua objek, beberapa diantaranya adalah *Euclidean Distance, Manhattan Distance, Minkowski Distance,* dan *Supremum Distance*. Empat formula tersebut merupakan formula yang paling populer atau sering digunakan dalam mencari jarak atribut numerik.

Euclidean Distance menggunakan sebuah formula yang menjumlahkan seluruh selisih dari atribut yang telah dikuadratkan, kemudian diakarkan. Berikut ini adalah formula dari *Euclidean Distance*:

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{ip} - x_{jp})^2}$$

Keterangan:

d = jarak	i : data 1
x = nilai atribut data	j : data 2
p = atribut	

Manhattan Distance menggunakan sebuah formula yang menjumlahkan seluruh selisih dari atribut yang telah dimutlakkan. Berikut ini adalah formula dari *Manhattan Distance*:

$$d(i,j) = \left|x_{i1} - x_{j1}\right| + \left|x_{i2} - x_{j2}\right| + \ldots + \left|x_{ip} - x_{jp}\right|$$

Minkowski Distance menggunakan sebuah formula yang menjumlahkan seluruh selisih dari atribut yang telah dimutlakkan dan dipangkatkan dengan nilai h (yang biasa digunakan adalah 1.5), kemudian diakarkan dengan nilai h tersebut. Berikut ini adalah formula dari *Minkowski Distance*:

$$d(i,j) = \sqrt[h]{\left|x_{i1} - x_{j1}\right|^h + \left|x_{i2} - x_{j2}\right|^h + \ldots + \left|x_{ip} - x_{jp}\right|^h}$$

Supremum Distance menggunakan sebuah formula yang mencari nilai maksimum pada selisih dari atribut yang telah dimutlakkan, kemudian . Berikut ini adalah formula dari *Supremum Distance*:

$$d(i,j) = \lim_{h\to\infty}\left(\sum_{f=1}^{p}\left|x_{if} - x_{jf}\right|^h\right)^{\frac{1}{h}} = \max_{f}^{p}\left|x_{if} - x_{jf}\right|$$

2. Teknik Pra Pemrosesan Data

Dalam pengerjaan tugas ini, kami tidak melakukan teknik pra pemrosesan data apapun karena kami rasa tidak diperlukan pra pemrosesan data. Hal itu dikarenakan data yang ada sudah bersih dan tidak memerlukan pra pemrosesan.

3. Strategi Penggunaan Algoritma kNN

Pada Algoritma kNN, terdapat dua langkah yang perlu diterapkan, yaitu menyimpan pola latih dan klasifikasi. Algoritma kNN dapat dituliskan seperti berikut:

1. Untuk setiap pola latih <x, f(x)>, tambahkan pola tersebut ke dalam **Daftar Pola Latih**.
2. Untuk setiap sebuah pola masukan xq:
   - Misalkan x1, x2, …, xk adalah k pola yang memiliki jarak terdekat (tetangga) dengan xq.
   - Kembalikan kelas yang memiliki jumlah pola paling banyak di antara k pola tersebut sebagai kelas keputusan.

B. Proses yang dibangun

1. Membaca File

Metode yang kami gunakan untuk membaca file mobil.xls adalah menggunakan pandas sebagai library agar struktur data dan analisis data mudah digunakan dan untuk membuat tabel, mengubah dimensi data, mengecek data, dan lain sebagainya. Sebelumnya file mobil.xls telah dimasukkan ke github terlebih dahulu untuk memudahkan proses pembacaan file. Implementasi pada program sebagai berikut:

```
import pandas as pd

df = pd.read_excel('https://rarrazaan.github.io/mobil.xls')

print("Input(ukuran, kenyamanan, irit, kecepatan):")
si, co, ec, sp = map(int, input().split())
pr = float(input("Harga: "))
data = {'Ukuran': [si], 'Kenyamanan': [co], 'Irit' : [ec], 'Kecepatan' : [sp], 'Harga (Ratus Juta)' : [pr]}
inp = pd.DataFrame(data)
inp.head()
```

2. Pra Pemprosesan Data

Dalam pengerjaan tugas ini, kami tidak melakukan teknik pra pemrosesan data apapun karena kami rasa tidak diperlukan pra pemrosesan data. Hal itu dikarenakan data yang ada sudah bersih dan tidak memerlukan pra pemrosesan.

3. Perhitungan Ukuran Jarak

Pada tugas ini, kami menggunakan empat formula untuk menghitung ukuran jarak atribut, yaitu *Euclidean Distance, Manhattan Distance, Minkowski Distance,* dan *Supremum Distance*. Implementasi pada program sebagai berikut:

a. *Euclidean Distance*

```
# Euclidean distance
def distance(data1, data2):
    size1 = data1.get('Ukuran')
    comfort1 = data1.get('Kenyamanan')
    eco1 = data1.get('Irit')
    speed1 = data1.get('Kecepatan')
    price1 = data1.get('Harga (Ratus Juta)')

    size2 = data2.get('Ukuran')
    comfort2 = data2.get('Kenyamanan')
    eco2 = data2.get('Irit')
    speed2 = data2.get('Kecepatan')
    price2 = data2.get('Harga (Ratus Juta)')

    jarak = ((size1 - size2)**2 + (comfort1 - comfort2)**2 + (eco1 - eco2)**2 + (speed1 - speed2)**2 + (price1 - price2)**2)**0.5

    return jarak
```

```
def kNN(k):
    ans = []
    har = []
    for i in range(len(inp)):
        list_jarak = []
        for j in range(len(df)):
            pair = (distance(inp.iloc[i], df.iloc[j]), j)
            list_jarak.append(pair)

        list_jarak.sort()
        #print(list_jarak)

    for j in range(k):
        neighbour_index = list_jarak[j][1]
        ans.append(df.iloc[neighbour_index]['Nama Mobil'])
        har.append(df.iloc[neighbour_index]['Harga (Ratus Juta)'])
    return ans, har


euc, euchar = kNN(3)
```

```
receu = {'Nama Mobil' : [i for i in euc], 'Harga' : [str(int(round(j*100))) +' Juta' for j in euchar]}
euclidean  = pd.DataFrame(receu)
```

b. *Manhattan Distance*

```
#Manhattan distance
def distance(data1, data2):
    size1 = data1.get('Ukuran')
    comfort1 = data1.get('Kenyamanan')
    eco1 = data1.get('Irit')
    speed1 = data1.get('Kecepatan')
    price1 = data1.get('Harga (Ratus Juta)')

    size2 = data2.get('Ukuran')
    comfort2 = data2.get('Kenyamanan')
    eco2 = data2.get('Irit')
    speed2 = data2.get('Kecepatan')
    price2 = data2.get('Harga (Ratus Juta)')
    jarak = (abs(size1 - size2) + abs(comfort1 - comfort2) + abs(eco1 - eco2) + abs(speed1 - speed2) + abs(price1 - price2))

    return jarak
```

```
def kNN(k):
    ans = []
    har = []
    for i in range(len(inp)):
        list_jarak = []
        for j in range(len(df)):
            pair = (distance(inp.iloc[i], df.iloc[j]), j)
            list_jarak.append(pair)
        list_jarak.sort()

    for j in range(k):
        neighbour_index = list_jarak[j][1]
        ans.append(df.iloc[neighbour_index]['Nama Mobil'])
        har.append(df.iloc[neighbour_index]['Harga (Ratus Juta)'])
    return ans, har

man, manhar = kNN(3)
```

```
recman = {'Nama Mobil' : [i for i in man], 'Harga' : [str(int(round(j*100))) +' Juta' for j in manhar]}
manhattan  = pd.DataFrame(recman)
```

c. *Minkowski Distance*

```
#Minkowski distance
def distance(data1, data2):
    size1 = data1.get('Ukuran')
    comfort1 = data1.get('Kenyamanan')
    eco1 = data1.get('Irit')
    speed1 = data1.get('Kecepatan')
    price1 = data1.get('Harga (Ratus Juta)')

    size2 = data2.get('Ukuran')
    comfort2 = data2.get('Kenyamanan')
    eco2 = data2.get('Irit')
    speed2 = data2.get('Kecepatan')
    price2 = data2.get('Harga (Ratus Juta)')
    p = 1.5
    jarakeu = (abs(size1 - size2)**p + abs(comfort1 - comfort2)**p + abs(eco1 - eco2)**p + abs(speed1 - speed2)**p + abs(price1 - price2)**p)**1/p

    return jarakeu
```

```
def kNN(k):
    ans = []
    har = []
    for i in range(len(inp)):
        list_jarak = []
        for j in range(len(df)):
            pair = (distance(inp.iloc[i], df.iloc[j]), j)
            list_jarak.append(pair)
        list_jarak.sort()

    for j in range(k):
        neighbour_index = list_jarak[j][1]
        ans.append(df.iloc[neighbour_index]['Nama Mobil'])
        har.append(df.iloc[neighbour_index]['Harga (Ratus Juta)'])
    return ans, har

mink, minkhar = kNN(3)
```

```
recmin = {'Nama Mobil' : [i for i in mink], 'Harga' : [str(int(round(j*100))) +' Juta' for j in minkhar]}
minkowski  = pd.DataFrame(recmin)
```

d. *Supremum Distance*

```
#Supremum distance
def distance(data1, data2):
    size1 = data1.get('Ukuran')
    comfort1 = data1.get('Kenyamanan')
    eco1 = data1.get('Irit')
    speed1 = data1.get('Kecepatan')
    price1 = data1.get('Harga (Ratus Juta)')

    size2 = data2.get('Ukuran')
    comfort2 = data2.get('Kenyamanan')
    eco2 = data2.get('Irit')
    speed2 = data2.get('Kecepatan')
    price2 = data2.get('Harga (Ratus Juta)')
    jarakeu = max(abs(size1 - size2), abs(comfort1 - comfort2), abs(eco1 - eco2), abs(speed1 - speed2), abs(price1 - price2))

    return jarakeu
```

```
def kNN(k):
    ans = []
    har = []
    for i in range(len(inp)):
        list_jarak = []
        for j in range(len(df)):
            pair = (distance(inp.iloc[i], df.iloc[j]), j)
            list_jarak.append(pair)
        list_jarak.sort()

    for j in range(k):
        neighbour_index = list_jarak[j][1]
        ans.append(df.iloc[neighbour_index]['Nama Mobil'])
        har.append(df.iloc[neighbour_index]['Harga (Ratus Juta)'])
    return ans, har

sup, suphar = kNN(3)
```

```
recsup = {'Nama Mobil' : [i for i in sup], 'Harga' : [str(int(round(j*100))) +' Juta' for j in suphar]}
supremum  = pd.DataFrame(recsup)
```

4. Klasifikasi kNN

Pola latih berasal dari inputan user. Output dari program kNN adalah 3 mobil yang jarak titiknya terdekat dengan *input user.*

5. Hasil

Proses output hasil pada program dilakukan dengan membuat data rekomendasi mobil dengan keterangan nama mobil beserta harga mobil. Sesuai dengan jumlah formula perhitungan jarak yang digunakan, terdapat empat jenis output rekomendasi mobil berdasarkan formula yang digunakan. Data rekomendasi mobil tersebut berupa file excel rekomendasi.xls. Implementasi code pada program menggunakan bahasa python 3 sebagai berikut:

```
writer = pd.ExcelWriter('rekomendasi.xls', engine='xlsxwriter')
euclidean.to_excel(writer, index=False, sheet_name='Euclidean')
manhattan.to_excel(writer, index=False, sheet_name='Manhattan')
minkowski.to_excel(writer, index=False, sheet_name='Minkowski')
supremum.to_excel(writer, index=False, sheet_name='Supremum')
writer.save()
```

C. Parameter Paling Optimum

Setelah dilakukan analisis terhadap setiap atribut, disimpulkan bahwa atribut Kenyamanan merupakan atribut yang paling berpengaruh terhadap hasil.

D. Tautan Video Presentasi

http://bit.ly/PresentasiTupro3Kel6_IF4307

E. Hasil Percobaan

Percobaan dilakukan dengan menggunakan input sebagai berikut:

- Ukuran : 4
- Kenyamanan : 5
- Irit : 6
- Kecepatan : 7
- Harga (Ratus Juta) : 8.0

a. *Euclidean Distance*

Nama Mobil	Harga
Toyota Corolla Altis	600 Juta
Honda City	270 Juta
Toyota Avanza	200 Juta

Euclidean | Manhattan | Minkowski | Supremum

b. *Manhattan Distance*

Nama Mobil	Harga
Toyota Corolla Altis	600 Juta
Honda City	270 Juta
Toyota Avanza	200 Juta

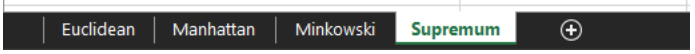
Euclidean | Manhattan | Minkowski | Supremum

c. *Minkowski Distance*

Nama Mobil	Harga
Toyota Corolla Altis	600 Juta
Toyota Avanza	200 Juta
Honda City	270 Juta

d. *Supremum Distance*

Nama Mobil	Harga
Toyota Innova	400 Juta
Toyota Corolla Altis	600 Juta
Alphard	1000 Juta



F. Kesimpulan

Berdasarkan hasil percobaan yang telah dilakukan, keempat formula yang diimplementasikan pada program dapat memberikan hasil yang diinginkan, yaitu memberikan rekomendasi mobil sesuai dengan *input* dari *user*. Penerapan formula *Euclidean Distance* dan *Minkowski Distance* memberikan hasil rekomendasi mobil yang sama karena nilai p atau h pada *Minkowski Distance* yang digunakan adalah 1.5. Sedangkan, pada *Manhattan Distance* dan *Supremum Distance* memberikan hasil rekomendasi mobil yang berbeda. Kesamaan dan perbedaan output dipengaruhi juga oleh besar nilai *input* dari *user*.

Referensi

Salindia Pengantar Kecerdasan Buatan Pokok Bahasan 11