

# IMPLEMENTASI SISTEM REKOMENDASI BERBASIS KNN

Azriel Naufal Aulia<sup>1</sup>, Muhammad Faiz Abdurrahman Djauhar<sup>2</sup>, Ryan Abdurohman<sup>3</sup>

<sup>1</sup>1301190374, IF-43-10, S-1 Informatika, Telkom University

<sup>2</sup>1301190361, IF-43-10, S-1 Informatika, Telkom University

<sup>3</sup>1301191171, IF-43-10, S-1 Informatika, Telkom University

## 1. Pendahuluan

Penggunaan sistem kecerdasan buatan sering kita jumpai di kehidupan kita sehari-hari. Perkembangan yang cepat dari sisi *hardware* dan *software*, telah membuahkan hasil berupa produk-produk AI yang inovatif di lingkungan sekitar kita. Di sini produk-produk tersebut dapat kita kelompokkan ke dalam 4 teknik dalam konsep AI, yaitu : *searching*, *reasoning*, *planning*, dan *learning*. Sesuai dengan judul dari laporan yang telah kami tulis, kita akan lebih menjelaskan salah satu implementasi dari penggunaan konsep teknik *learning* di AI, yaitu menggunakan salah satu metode *collaborative filtering*, dikenal *K-nearest neighbor* (KNN).

Dengan berkembangnya industri otomotif di Indonesia, mendorong para pelaku usaha untuk terus melakukan inovasi dari produk kendaraan yang dibuatnya, Menciptakan lingkungan bisnis yang sangat kompetitif, siapapun dia yang ketinggalan dari perusahaan kompetitornya akan kalah telak dalam persaingan. Masing-masing perusahaan otomotif harus melakukan pembaruan dari sisi mesin, tampilan dan fitur supaya tampak lebih nyaman dan menarik di mata konsumen. Akibatnya konsumen harus selektif dalam memilih kendaraan yang akan dibeli berdasarkan parameter-parameter yang dapat dibandingkan (contoh : ukuran, kenyamanan, kecepatan, dan harga). Namun masalahnya, konsumen sering kesulitan dalam menilai parameter-parameter tersebut, karena tidak adanya subjek pembanding yang dapat dijadikan standar penilaian. Salah satu solusi yang dapat dipakai dari studi kasus tersebut, adalah dengan membuat sistem rekomendasi dengan metode KNN.

Dalam sistem rekomendasi pemilihan kendaraan menggunakan metode KNN, pemilihan akan dilakukan berdasarkan jarak kedekatan data testing (database) dengan data training (data sampel). Kedekatan data (kemiripan data) tersebut digunakan untuk merekomendasikan pilihan kendaraan ke pengguna. Dengan konsep KNN, sistem ini dapat mengantisipasi jika konsumen kurang paham dengan parameter-parameter tersebut, karena metode ini menerapkan prinsip pencarian menggunakan jarak kedekatan (kemiripan data) antara sampel dan database. Kemiripan data dapat dihitung dengan formula jarak. Berikut formula jarak yang dapat digunakan, yaitu :

- *Euclidean Distance*
- *Manhattan Distance*
- *Minkowski Distance*

- *Supremum Distance*

Untuk perhitungan dari formula diatas, akan dijelaskan lebih ringkas di metodologi.

## 2. Metodologi

Sesuai dengan ketentuan di soal, masalah inti yang diselesaikan dari sistem rekomendasi kendaraan adalah untuk mencari tetangga terdekat dari parameter masukan yang diberikan oleh user terhadap data training yang dimiliki sistem. Pada penelitian ini, parameter yang digunakan pada metode KNN ada:

- Ukuran
- Kenyamanan
- Irit
- Kecepatan
- Harga (Dalam Ratus Juta)

Berikut adalah data training yang diberikan dari soal, untuk menetapkan parameter masukan yang nanti akan diberikan dari user ke program: (dalam file excel)

	A	B	C	D	E	F	G
1	Nama Mobil	Ukuran	Kenyamanan	Irit	Kecepatan	Harga (Ratus Juta)	
2	Toyota Agya	4	4	9	6	1	
3	Daihatsu Alya	4	3	9	6	1.1	
4	Toyota Avanza	6	5	6	6	2	
5	Daihatsu Xenia	6	4	6	6	1.75	
6	Xpander	7	7	6	7	2.25	
7	Livina	7	7	6	7	2.1	
8	Karimun	3	4	10	5	1.2	
9	Toyota Innova	8	8	5	7	4	
10	Alphard	9	10	4	8	10	
11	Toyota Vios	5	7	9	8	2.5	
12	Honda City	5	8	7	8	2.7	
13	Toyota Hiace	10	5	8	6	5	
14	Toyota Fortuner	9	8	5	8	5	
15	Toyota Foxy	9	9	5	7	5.5	
16	Toyota Corolla Altis	5	9	7	9	6	
17	Suzuki Ertiga	7	7	7	7	2.3	
18	Suzuki Carry	7	3	9	5	0.8	

Berikut adalah codingan algoritma dari fungsi-fungsi yang dipakai di program sistem rekomendasi kendaraan dengan metode KNN.

Berikut beberapa *library module* yang dipakai untuk mempermudah pembuatan program. Terdapat *library math* yang dipakai untuk mempermudah perhitungan formula jarak, *library xlwt* dan *xlrd* untuk membaca dan menulis data *training* dari file excel (format xls), dan *library copy* untuk menyalin nilai objek secara rekursif.

```
tugas AI p3 > tuproAI3.py
1  from copy import deepcopy
2  from xlwt import Workbook
3  import xlrd
4  import math
5  import xlwt
```



Langkah-langkah dalam pembuatan sistem rekomendasi dimulai dengan membaca data *training* dalam file excel yang didapat dari ketentuan tugas yang diberikan. Proses pembacaan data dalam file excel dan menyalin dalam bentuk array, dilakukan dalam fungsi `baca_data`.

```
def baca_data(path):
    data = xlrd.open_workbook(path)
    wsData = data.sheet_by_index(0)
    df = []
    for i in range(1, 18):
        tmp = []
        d = wsData.cell(i, 0)
        tmp.append(d.value)
        d = wsData.cell(i, 1)
        tmp.append(d.value)
        d = wsData.cell(i, 2)
        tmp.append(d.value)
        d = wsData.cell(i, 3)
        tmp.append(d.value)
        d = wsData.cell(i, 4)
        tmp.append(d.value)
        d = wsData.cell(i, 5)
        tmp.append(d.value)
        df.append(tmp)
    return df
```

Setelah menerima data *training* diatas, untuk menghindari terjadinya berbagai anomali data dan tidak konsistensinya data pada data *training*, kami memutuskan untuk melakukan proses normalisasi pada saat proses perancangan. Hasil proses normalisasi, diharapkan memberikan data *training* yang memiliki tabel-tabel normal. Formula normalisasi yang kami pakai, antara lain:

### Normalization Formula

$$X_{normalized} = \frac{(X - X_{minimum})}{(X_{maximum} - X_{minimum})}$$

Proses normalisasi, kami terapkan pada data *training* dan juga data *testing* yang dimasukkan oleh user ke program. Proses normalisasi untuk setiap data dilakukan di dalam fungsi normalisasi dan prapemrosesan.

```
def normalisasi(x, up, down):
    return float(float(x-down)/float(up-down))

def prapemrosesan(training, testing):
    norm = training[:]
    norm.append(testing)

    low = 0
    high = 0
    for row in range(len(norm)):
        if norm[row][5] >= high:
            high = norm[row][5]
        if norm[row][5] <= low:
            low = norm[row][5]
    for row in range(len(norm)):
        for col in range(1, len(norm[0])):
            if col != len(norm[0])-1:
                norm[row][col] = normalisasi(float(norm[row][col]), float(10), float(0))
            else:
                norm[row][col] = normalisasi(float(norm[row][col]), float(high), float(low))
    trains = norm[:-1]
    tests = norm[-1:]
    return trains, tests
```

Sesuai dengan parameter-parameter pada data *training*, User akan memasukkan nilai yang diinginkan pada setiap parameter dari ukuran, kenyamanan, irit, kecepatan, dan harga sesuai dengan interval yang ditetapkan. Fungsi `input_test` di bawah akan melakukan proses masukan dari user ke program.

```
def input_test():
    test = []
    test.append("sample_test")
    t = float(input("Ukuran (1-10) : "))
    test.append(t)
    t = float(input("Kenyamanan (1-10) : "))
    test.append(t)
    t = float(input("Irit (1-10) : "))
    test.append(t)
    t = float(input("Kecepatan (1-10) : "))
    test.append(t)
    t = float(input("Harga (dalam ratus juta) : "))
    test.append(t)
    return test
```

Dalam sistem rekomendasi yang kami buat, kemiripan data dihitung menggunakan beberapa formula jarak, terdapat :

1. *Euclidean Distance*

$$d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{ip} - x_{jp})^2} \quad (1)$$

## 2. Manhattan Distance

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{ip} - x_{jp}| \quad (2)$$

## 3. Minkowski Distance

$$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \dots + |x_{ip} - x_{jp}|^h} \quad (3)$$

## 4. Supremum Distance

$$d(i, j) = \lim_{h \rightarrow \infty} \left( \sum_{f=1}^p |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_f |x_{if} - x_{jf}| \quad (4)$$

Sesuai dengan rumus pada setiap formula jarak diatas, Masing-masing formula akan dibuatkan fungsinya masing-masing. Yang menerima (parameter fungsi) nilai dari data *training* dan data *testing* dan menghasilkan nilai kedekatan (kemiripan data). Pada nilai kedekatan, semakin kecil nilai tersebut, antara data *testing* dan data *training* dianggap memiliki kemiripan yang tinggi.

```
### Euclidean distance
"""
```

```
def euclidean(x1, x2):
    a = 0
    i = 1
    l = len(x2)
    while i < l:
        a += ((x1[i]-x2[i])**2)
        i += 1
    return math.sqrt(a)
```

```
"""### Manhattan distance
"""
```

```
def manhattan(x1, x2):
    a = 0
    i = 1
    l = len(x2)
    while i < l:
        a += abs(x1[i]-x2[i])
        i += 1
    return a
```

```
"""### Minkowski distance
```

```
"""
```

```
def minkowski(x1, x2, h):
```

```
    a = 0
```

```
    i = 1
```

```
    l = len(x2)
```

```
    while i < l:
```

```
        a += (abs(x1[i]-x2[i])**h)
```

```
        i += 1
```

```
    return math.pow(a, 1/h)
```

```
"""### Supremum distance"""
```

```
def supremum(x1, x2):
```

```
    a = []
```

```
    i = 1
```

```
    l = len(x2)
```

```
    while i < l:
```

```
        a.append(abs(x1[i]-x2[i]))
```

```
        i += 1
```

```
    return max(a)
```

Setelah melakukan normalisasi pada data *testing* dan data *training*, akan dihitung dengan masing-masing formula jarak di atas antara data *testing* dengan setiap data *training* yang disediakan, yang kemudian disimpan dalam *dictionary*.

```
def kalkulasi(train, test, ntrain, ntest):
```

```
    df = []
```

```
    for i in range(len(train)-1):
```

```
        try:
```

```
            t = {}
```

```
            t['train'] = train[i]
```

```
            t['norm'] = ntrain[i]
```

```
            t['dist'] = {}
```

```
            t['dist']['euclidean'] = euclidean(ntrain[i], ntest)
```

```
            t['dist']['manhattan'] = manhattan(ntrain[i], ntest)
```

```
            t['dist']['minkowski'] = minkowski(ntrain[i], ntest, 1.5)
```

```
            t['dist']['supremum'] = supremum(ntrain[i], ntest)
```

```
            df.append(t)
```

```
        except IndexError:
```

```
            print(i)
```

```
            return
```

```
    return df
```

Pada fungsi knn, setelah kita menghitung seluruh nilai kedekatan antara data *testing* dan setiap data *training*, kita akan melakukan pengurutan membesar dan mengambil 3 data (karena  $k = 3$ , diminta soal) awal atau dengan nilai kedekatan terendah (semakin kecil nilai tersebut, dianggap semakin mirip), yang nantinya akan menentukan hasil rekomendasi kendaraan dan ditulis didalam file berformat excel.

```
def knn(df):
```

```
    knn = {}
```

```
    knn['eu'] = sorted(df, key = lambda i: (i['dist']['euclidean'], i['train'][0]))
```

```
    knn['ma'] = sorted(df, key = lambda i: (i['dist']['manhattan'], i['train'][0]))
```

```
    knn['mi'] = sorted(df, key = lambda i: (i['dist']['minkowski'], i['train'][0]))
```

```
    knn['su'] = sorted(df, key = lambda i: (i['dist']['supremum'], i['train'][0]))
```

```
    return knn['eu'][:3], knn['ma'][:3], knn['mi'][:3], knn['su'][:3]
```

Fungsi main, merupakan fungsi utama kita (*driver code*).

```
def main():
    train = baca_data('mobil.xls')
    test = input_test()
    ntrain, ntest = prapemrosesan(deepcopy(train), deepcopy(test))
    df = kalkulasi(deepcopy(train), deepcopy(test), deepcopy(ntrain), deepcopy(ntest))
    eu, ma, mi, su = knn(deepcopy(df))
    wb = Workbook()
    sheet1 = wb.add_sheet('rekomendasi')
    sheet1.write(0, 0, eu[0]['norm'][0])
    sheet1.write(1, 0, eu[1]['norm'][0])
    sheet1.write(2, 0, eu[2]['norm'][0])
    wb.save('rekomendasi.xls')

if __name__ == "__main__":
    main()
```

Sesuai pseudocode diatas hasil rekomendasi yang keluar berdasarkan nilai kedekatan hasil formula *Euclidean Distance*. User dapat bereksperimen, menggunakan formula jarak yang berbeda untuk menghasilkan rekomendasi yang beragam (kita akan melakukan eksperimen yang hasilnya merupakan bab hasil dan pembahasan).

### 3. Hasil dan pembahasan

Dalam hasil dan pembahasan disini, kita akan melakukan observasi dimana kita akan memasukkan nilai 10 untuk masing-masing parameter (ukuran, kenyamanan, irit, kecepatan, dan harga) dan memperhatikan hasil rekomendasi yang dihasilkan berdasarkan formula jarak yang berbeda-beda.

Hasil

#### 1. Data *training* sebelum dilakukan normalisasi

	A	B	C	D	E	F	G
1	Nama Mobil	Ukuran	Kenyamanan	Irit	Kecepatan	Harga (Ratus Juta)	
2	Toyota Agya	4	4	9	6	1	
3	Daihatsu Alya	4	3	9	6	1.1	
4	Toyota Avanza	6	5	6	6	2	
5	Daihatsu Xenia	6	4	6	6	1.75	
6	Xpander	7	7	6	7	2.25	
7	Livina	7	7	6	7	2.1	
8	Karimun	3	4	10	5	1.2	
9	Toyota Innova	8	8	5	7	4	
10	Alphard	9	10	4	8	10	
11	Toyota Vios	5	7	9	8	2.5	
12	Honda City	5	8	7	8	2.7	
13	Toyota Hiace	10	5	8	6	5	
14	Toyota Fortuner	9	8	5	8	5	
15	Toyota Foxy	9	9	5	7	5.5	
16	Toyota Corolla Altis	5	9	7	9	6	
17	Suzuki Ertiga	7	7	7	7	2.3	
18	Suzuki Carry	7	3	9	5	0.8	

## 2. Data *training* setelah dilakukan normalisasi

data training					
Nama Mobil	Ukuran	Kenyamanan	Irit	Kecepatan	Harga (Ratus Juta)
Toyota Agya	0.4	0.4	0.9	0.6	0.1
Daihatsu Alza	0.4	0.3	0.9	0.6	0.11
Toyota Avanza	0.6	0.5	0.6	0.6	0.2
Daihatsu Xenia	0.6	0.4	0.6	0.6	0.175
Xpander	0.7	0.7	0.6	0.7	0.225
Livina	0.7	0.7	0.6	0.7	0.21
Karimun	0.3	0.4	1	0.5	0.12
Toyota Innova	0.8	0.8	0.5	0.7	0.4
Alphard	0.9	1	0.4	0.8	1
Toyota Vios	0.5	0.7	0.9	0.8	0.25
Honda City	0.5	0.8	0.7	0.8	0.27
Toyota Hiace	1	0.5	0.8	0.6	0.5
Toyota Fortuner	0.9	0.8	0.5	0.8	0.5
Toyota Foxy	0.9	0.9	0.5	0.7	0.55
Toyota Corolla Altis	0.5	0.9	0.7	0.9	0.6
Suzuki Ertiga	0.7	0.7	0.7	0.7	0.23

## 3. Data *testing* / masukan nilai parameter dari user.

21		Ukuran	Kenyamanan	Irit	Kecepatan	Harga (Ratus Juta)
22	data testing	10	10	10	10	10

## 4. Hasil jarak masing-masing formula antara data *training* (yang telah dinormalisasikan) dan data *testing*

Nama Mobil	euclidean	manhattan	minkowski	supremum
Toyota Agya	1.30384048	2.6	1.62314776	0.9
Daihatsu Alza	1.34614264	2.69	1.6785186	0.89
Toyota Avanza	1.17046999	2.5	1.49505583	0.8
Daihatsu Xenia	1.23313625	2.625	1.57310767	0.825
Xpander	1.01519703	2.075	1.26821244	0.775
Livina	1.02669372	2.09	1.2799677	0.79
Karimun	1.36908729	2.68	1.706619	0.88
Toyota Innova	0.88317609	1.8	1.10496757	0.6
Alphard	0.64031242	0.9	0.70012866	0.6
Toyota Vios	0.97596106	1.85	1.18407562	0.75
Honda City	0.97616597	1.93	1.20360844	0.73
Toyota Hiace	0.83666003	1.6	1.03275438	0.5
Toyota Fortuner	0.76811457	1.5	0.9442935	0.5
Toyota Foxy	0.75	1.45	0.92038355	0.5
Toyota Corolla Altis	0.72111026	1.4	0.88609039	0.5
Suzuki Ertiga	0.97616597	1.97	1.21117506	0.77

## 5. Melakukan pengurutan membesar berdasarkan masing-masing formula dan mengambil 3 data awal/teratas sebagai hasil rekomendasi



Menggunakan *Euclidean Distance*

Nama Mobil	euclidean
Alphard	0.640312424
Toyota Corolla Altis	0.721110255
Toyota Foxy	0.75
Toyota Fortuner	0.768114575
Toyota Hiace	0.836660027
Toyota Innova	0.883176087
Toyota Vios	0.975961065
Honda City	0.976165969
Suzuki Ertiga	0.976165969
Xpander	1.015197025
Livina	1.026693723
Toyota Avanza	1.170469991
Daihatsu Xenia	1.233136246
Toyota Agya	1.303840481
Daihatsu Alya	1.346142637
Karimun	1.369087287

Menggunakan *Manhattan Distance*

Nama Mobil	manhattan
Alphard	0.9
Toyota Corolla Altis	1.4
Toyota Foxy	1.45
Toyota Fortuner	1.5
Toyota Hiace	1.6
Toyota Innova	1.8
Toyota Vios	1.85
Honda City	1.93
Suzuki Ertiga	1.97
Xpander	2.075
Livina	2.09
Toyota Avanza	2.5
Toyota Agya	2.6
Daihatsu Xenia	2.625
Karimun	2.68
Daihatsu Alya	2.69

Menggunakan <i>Minkowski Distance</i>	<table> <tr> <th>Nama Mobil</th><th>minkowski</th></tr> <tr><td>Alphard</td><td>0.70012866</td></tr> <tr><td>Toyota Corolla Altis</td><td>0.88609039</td></tr> <tr><td>Toyota Foxy</td><td>0.92038355</td></tr> <tr><td>Toyota Fortuner</td><td>0.9442935</td></tr> <tr><td>Toyota Hiace</td><td>1.03275438</td></tr> <tr><td>Toyota Innova</td><td>1.10496757</td></tr> <tr><td>Toyota Vios</td><td>1.18407562</td></tr> <tr><td>Honda City</td><td>1.20360844</td></tr> <tr><td>Suzuki Ertiga</td><td>1.21117506</td></tr> <tr><td>Xpander</td><td>1.26821244</td></tr> <tr><td>Livina</td><td>1.2799677</td></tr> <tr><td>Toyota Avanza</td><td>1.49505583</td></tr> <tr><td>Daihatsu Xenia</td><td>1.57310767</td></tr> <tr><td>Toyota Agya</td><td>1.62314776</td></tr> <tr><td>Daihatsu Alya</td><td>1.6785186</td></tr> <tr><td>Karimun</td><td>1.706619</td></tr> </table>	Nama Mobil	minkowski	Alphard	0.70012866	Toyota Corolla Altis	0.88609039	Toyota Foxy	0.92038355	Toyota Fortuner	0.9442935	Toyota Hiace	1.03275438	Toyota Innova	1.10496757	Toyota Vios	1.18407562	Honda City	1.20360844	Suzuki Ertiga	1.21117506	Xpander	1.26821244	Livina	1.2799677	Toyota Avanza	1.49505583	Daihatsu Xenia	1.57310767	Toyota Agya	1.62314776	Daihatsu Alya	1.6785186	Karimun	1.706619
Nama Mobil	minkowski																																		
Alphard	0.70012866																																		
Toyota Corolla Altis	0.88609039																																		
Toyota Foxy	0.92038355																																		
Toyota Fortuner	0.9442935																																		
Toyota Hiace	1.03275438																																		
Toyota Innova	1.10496757																																		
Toyota Vios	1.18407562																																		
Honda City	1.20360844																																		
Suzuki Ertiga	1.21117506																																		
Xpander	1.26821244																																		
Livina	1.2799677																																		
Toyota Avanza	1.49505583																																		
Daihatsu Xenia	1.57310767																																		
Toyota Agya	1.62314776																																		
Daihatsu Alya	1.6785186																																		
Karimun	1.706619																																		
Menggunakan <i>Supremum Distance</i>	<table> <tr> <th>Nama Mobil</th><th>supremum</th></tr> <tr><td>Toyota Hiace</td><td>0.5</td></tr> <tr><td>Toyota Fortuner</td><td>0.5</td></tr> <tr><td>Toyota Foxy</td><td>0.5</td></tr> <tr><td>Toyota Corolla Altis</td><td>0.5</td></tr> <tr><td>Toyota Innova</td><td>0.6</td></tr> <tr><td>Alphard</td><td>0.6</td></tr> <tr><td>Honda City</td><td>0.73</td></tr> <tr><td>Toyota Vios</td><td>0.75</td></tr> <tr><td>Suzuki Ertiga</td><td>0.77</td></tr> <tr><td>Xpander</td><td>0.775</td></tr> <tr><td>Livina</td><td>0.79</td></tr> <tr><td>Toyota Avanza</td><td>0.8</td></tr> <tr><td>Daihatsu Xenia</td><td>0.825</td></tr> <tr><td>Karimun</td><td>0.88</td></tr> <tr><td>Daihatsu Alya</td><td>0.89</td></tr> <tr><td>Toyota Agya</td><td>0.9</td></tr> </table>	Nama Mobil	supremum	Toyota Hiace	0.5	Toyota Fortuner	0.5	Toyota Foxy	0.5	Toyota Corolla Altis	0.5	Toyota Innova	0.6	Alphard	0.6	Honda City	0.73	Toyota Vios	0.75	Suzuki Ertiga	0.77	Xpander	0.775	Livina	0.79	Toyota Avanza	0.8	Daihatsu Xenia	0.825	Karimun	0.88	Daihatsu Alya	0.89	Toyota Agya	0.9
Nama Mobil	supremum																																		
Toyota Hiace	0.5																																		
Toyota Fortuner	0.5																																		
Toyota Foxy	0.5																																		
Toyota Corolla Altis	0.5																																		
Toyota Innova	0.6																																		
Alphard	0.6																																		
Honda City	0.73																																		
Toyota Vios	0.75																																		
Suzuki Ertiga	0.77																																		
Xpander	0.775																																		
Livina	0.79																																		
Toyota Avanza	0.8																																		
Daihatsu Xenia	0.825																																		
Karimun	0.88																																		
Daihatsu Alya	0.89																																		
Toyota Agya	0.9																																		

Dari hasil perhitungan masing-masing formula, kita mendapatkan untuk hasil rekomendasi dari masing-masing formula, yaitu :

- Untuk *Euclidean Distance*

Nama Mobil	euclidean
Alphard	0.640312424
Toyota Corolla Altis	0.721110255
Toyota Foxy	0.75

Hasil rekomendasi merupakan mobil Alphard, Toyota Corolla Altis dan Toyota Foxy

- Untuk *Manhattan Distance*

Nama Mobil	manhattan
Alphard	0.9
Toyota Corolla Altis	1.4
Toyota Foxy	1.45

Hasil rekomendasi merupakan mobil Alphard, Toyota Corolla Altis dan Toyota Foxy

- Untuk *Minkowski Distance*

Nama Mobil	minkowski
Alphard	0.70012866
Toyota Corolla Altis	0.88609039
Toyota Foxy	0.92038355

Hasil rekomendasi merupakan mobil Alphard, Toyota Corolla Altis dan Toyota Foxy

- Untuk *Supremum Distance*

Nama Mobil	supremum
Toyota Hiace	0.5
Toyota Fortuner	0.5
Toyota Foxy	0.5

Hasil rekomendasi merupakan mobil Toyota Hiace, Toyota Fortuner dan Toyota Foxy

Jika diperhatikan dari hasil observasi rekomendasi diatas, hasil dari formula *Euclidean Distance*, *Manhattan Distance* dan *Minkowski Distance*, cenderung menghasilkan rekomendasi kendaraan yang sama. Namun untuk *Supremum Distance*, hasil rekomendasi kendaraan cukup berbeda dengan formula lainnya, hanya memiliki kesamaan 1 dari 3 rekomendasi kendaraan dengan yang lainnya, yaitu Toyota Foxy. berdasarkan penilaian kami, hasil rekomendasi berbeda yang didapatkan dari *Supremum Distance*, dikarenakan beberapa faktor, antara lain :

- Rendahnya tingkat presisi yang dihasilkan. Berbeda halnya dengan *Euclidean Distance* dan *Minkowski Distance* yang memiliki tingkat presisi yang tinggi.
- Dari hasil *Supremum Distance*, 3 data hasil rekomendasi bahkan memiliki nilai yang sama (bahkan apabila diambil 4 data/  $k = 4$ , didapat 4 data hasil rekomendasi yang sama). Jika terdapat nilai yang sama seperti kasus diatas, maka proses pengurutan dikatakan tidak efektif dan bisa menyesatkan. Karena dengan menggunakan algoritma pengurutan yang berbeda dapat merubah hasil rekomendasinya. Contoh jika hasil *Supremum Distance* diatas, karena nilainya sama, akan berbeda apabila kita urutkan berdasarkan nama mobilnya.
- Secara umum, formula *Minkowski Distance* merupakan generalisasi formula *Euclidean Distance* dan *Manhattan Distance*. Jika diperhatikan rumus *Minkowski Distance* (pada rumus 3 diatas), apabila  $h = 1$  maka formula *Minkowski Distance* sama dengan *Manhattan Distance*. Sedangkan  $h = 2$  maka sama dengan *Euclidean Distance*. Oleh karena itu, hasil rekomendasi berdasarkan formula *Euclidean Distance*, *Manhattan Distance* dan *Minkowski Distance* memungkinkan untuk sama (walaupun tidak selalu).

Berbeda dengan formula *Supremum Distance* yang jika diperhatikan, memiliki formula yang cukup berbeda atau tidak memiliki kesamaan pola dengan formula lainnya.

#### 4. Simpulan

- Hasil rekomendasi yang dihitung berdasarkan formula *Euclidean Distance*, dan *Minkowski Distance* memiliki tingkat presisi yang cukup tinggi
- Hasil rekomendasi yang dihitung berdasarkan formula *Manhattan Distance*, dan *Supremum Distance* memiliki tingkat presisi yang cukup rendah, dengan *Supremum Distance* yang terendah. Hasil dengan presisi yang rendah rentan dengan kesalahan ambiguitas seperti masalah yang didapat dari hasil observasi kita, mendapat data rekomendasi dengan nilai yang sama.

#### 5. Video

Ryan Abdurohman : <https://youtu.be/r2t9yTDYIUU>

Muhammad Faiz Abdurrahman Djauhar : <http://gg.gg/v6ny4>

Azriel Naufal Aulia : [https://youtu.be/TAh2n3Y\\_9rQ](https://youtu.be/TAh2n3Y_9rQ)

#### 6. Daftar Pustaka

- [https://www.researchgate.net/publication/336160521\\_REKOMENDASI\\_SISTEM\\_PEMILIHAN\\_MOBIL\\_MENGGUNAKAN\\_K-NEAREST\\_NEIGHBOR\\_KNN\\_COLLABORATIVE\\_FILTERING](https://www.researchgate.net/publication/336160521_REKOMENDASI_SISTEM_PEMILIHAN_MOBIL_MENGGUNAKAN_K-NEAREST_NEIGHBOR_KNN_COLLABORATIVE_FILTERING)
- [https://www.academia.edu/19603162/Analisa\\_Nilai\\_Lamda\\_Model\\_Jarak\\_Minkowsky\\_Untuk\\_Penentuan\\_Jurusan\\_SMA](https://www.academia.edu/19603162/Analisa_Nilai_Lamda_Model_Jarak_Minkowsky_Untuk_Penentuan_Jurusan_SMA)