LAPORAN : TUGAS PEMROGRAMAN 03
IMPLEMENTASI KNN DALAM PEMBUATAN SISTEM REKOMENDASI MOBIL

Kelompok A - IF4302
Kaenova Mahendra Auditma (1301190324)[1]
Adhe Akram Azhari (1301194063)[2]
Elita Aurora Az Zahra (1301194127)[3]
Pengantar Kecerdasan Buatan (CII2M3-IF-43-02)
Informatika, FI, Universitas Telkom. 2021.
email:
[1]kaenova@student.telkomuniversity.ac.id
[2]adheakramazhari@student.telkomuniversity.ac.id
[3]elitaaurora@student.telkomuniversity.ac.id

1. PENDAHULUAN

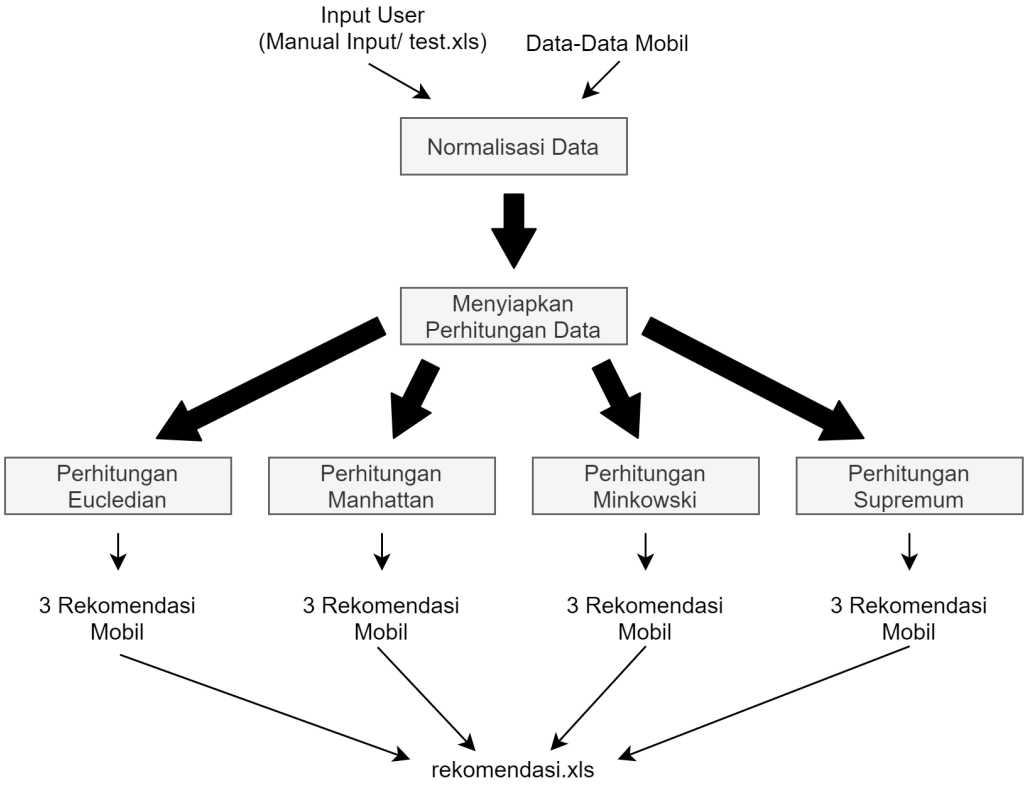
Tugas Pemrograman 03 merupakan tugas terakhir dari ketiga projek tugas yang ada pada Mata Kuliah Pengantar Kecerdasan Buatan. Pada tugas ini, kami diminta untuk membuat suatu sistem rekomendasi mobil berbasis kNN untuk memilih 3 (tiga) mobil terbaik sesuai dengan inputan user. Sistem membaca masukan file mobil.xls dan mengeluarkan output berupa sebuah file rekomendasi.xls yang berisi satu vektor kolom berisi 3 (tiga) baris string yang menyatakan nama mobil yang direkomendasikan pada file mobil.xls.

[1]kNN merupakan salah satu algoritma yang ada pada kecerdasan buatan. kNN yang memiliki singkatan k-Nearest Neighbor merupakan sebuah metode yang melakukan klasifikasi secara langsung berdasarkan tetangga terdekat. Terdapat beberapa teknik pencarian algoritma kNN yang digunakan pada Tugas Pemrograman 03, yaitu:

1) Euclidean Distance
2) Manhattan Distance
3) Minkowski Distance
4) Supremum Distance

2. RANCANGAN ALGORITMA

kNN atau k-Nearest Neighbor merupakan algoritma yang dapat digunakan sebagai sistem rekomendasi dengan mencari "jarak" terdekat dari input yang kita inginkan. "Jarak" yang ditentukan dapat ditentukan dengan beberapa perhitungan. Pada laporan ini kami menggunakan beberapa perhitungan diantaranya Euclidean Distance, Manhattan Distance, Minkowski Distance, dan Supremum Distance. Secara garis besar kami membuat model dengan langkah-langkah seperti pada gambar di bawah:



Gambar 2.1
Overview Model

Dari model yang sudah direncanakan kami akan mendapatkan 3 rekomendasi dari setiap perhitungan jarak dan selanjutnya pada laporan ini akan kami berikan contoh hasil akhir rekomendasi dari data dan input yang sudah kami tentukan.

2.1. Normalisasi Data dan Menyiapkan Perhitungan (Pra Pemprosesan Data)

Seperti yang sudah dijelaskan pada dokumen tugas, kami diberikan suatu data-data mobil yang memiliki atribut-atribut ukuran, kenyamanan, irit, dan kecepatan dengan skala 0-10. Terakhir ada atribut harga dengan satuan ratusan juta, sehingga ketika kita memiliki angka 1 maka, akan menjadi 100 juta. Hal ini dilakukan untuk penyerasian data pada saat perhitungan sehingga hasil-hasil tersebut dapat terlihat mirip dan mudah untuk dianalisa.

Untuk menormalisasikan data-data tersebut kami menggunakan *min-max scaler* pada setiap atribut. [2]*Min-max scaler* memiliki rumus seperti di bawah:

$$X_{normal} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

(2.2)

Mengaplikasikan rumus tersebut pada setiap atribut memiliki nilai *min* dan *max* yang berbeda. Untuk atribut ukuran, kenyamanan, irit, dan kecepatan karena sudah diketahui skala yang digunakan ialah 0-10 sehingga nilai X_{min} dan X_{max} ialah 0 dan 10. Pada atribut harga, sudah dipastikan nilai terkecil adalah 0 sedangkan untuk nilai terbesarnya kita harus mengetahui dari *input user* dan nilai terbesar dari data yang kita miliki lalu membandingkan dari kedua nilai tersebut. Agar membantu saat perhitungan pada model yang kami buat, suatu objek mobil akan didefinisikan yang akan berisi dari nama dan atribut-atribut dari data mobil yang sudah dinormalisasikan.

2.2. Euclidean Distance

[3][4]Euclidean Distance adalah salah satu metode perhitungan jarak pada kNN yang digunakan untuk menghitung jarak dari dua buah titik dalam *Euclidean Space* yang meliputi bidang Euclidean dua dimensi, tiga dimensi, atau lebih. Adapun rumus dari Euclidean Distance sebagai berikut:

$$d_{Euclidean}(x,y) = \sqrt{\sum_{i} (x_i - y_i)^2}$$

Gambar (2.3)
Rumus Euclidean Distance

Keterangan rumus:
d = jarak euclidean
xi = input ke i
yi = data ke i

2.3. Manhattan Distance

[3]Manhattan Distance digunakan untuk menentukan/menghitung perbedaan absolut (mutlak) suatu nilai atau koordinat sepasang objek. Rumus yang digunakan sebagai berikut:

$$d(x,y) = \sum_{i=1}^{n} |xi - yi|$$

Gambar (2.4)
Rumus Manhattan Distance

dimana,
d = Jarak antara I1 dan I2
x = Inputan
y = Data pada atribut
i = Setiap data
xi = Inputan ke i
yi = Data pada atribut ke i
n = Jumlah data

2.4. Minkowski Distance

[3]Minkowski distance merupakan sebuah metrik dalam ruang vektor di mana suatu norma didefinisikan (normed vector space) sekaligus dianggap sebagai generalisasi dari Euclidean distance dan Manhattan distance. Dalam pengukuran jarak objek menggunakan minkowski distance biasanya digunakan nilai p adalah 1 atau 2. Adapun rumus yang digunakan sebagai berikut :

$$d(x,y) = \left(\sum_{i=1}^{n} |xi - yi|^{p}\right)^{1/p}$$

Gambar (2.5)
Rumus Minkowski Distance

2.5. Supremum Distance

[4]Supremum distance atau juga disebut sebagai $L\ max$, $L\ \infty$ norma dan sebagai Chebyshev distance merupakan generalisasi dari jarak Minkowski untuk $h \rightarrow \infty$. Untuk menghitung Supremum Distance ditemukan atribut f yang memberikan perbedaan nilai maksimum antara dua objek, perbedaan adalah jarak tertinggi. Adapun rumus Supremum Distance sebagai berikut:

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

Gambar (2.6)
Rumus Supremum Distance

2.6. Klasifikasi kNN

Pada tahap ini, kami membandingkan sebuah hasil perhitungan jarak antara semua data mobil dan data yang diinputkan oleh pengguna. Persoalan ini meminta kami untuk mengeluarkan 3 rekomendasi mobil, sehingga nilai k yang digunakan dalam kNN ini ialah 3. Pada setiap perhitungan akan dilakukan pengurutan secara membesar lalu akan diambil 3 nilai terkecil dan akan menjadi rekomendasi kami dari perhitungan jarak tertentu.

3. IMPLEMENTASI ALGORITMA DAN HASIL PROGRAM

Setelah merencanakan model, kami mengimplementasikan model ini menggunakan Bahasa Pemrograman Python serta bantuan library standar dan *pandas* yang membantu kami untuk membaca dan membuat *file excel* sebagai *input* dan *output*.

3.1. Input Data

```
pilihan = int(input("Input user data:\n 1. Manual Input\n 2. Using ./data/raw/test.xls\n
Choose: "))
if pilihan == 1:
    print("========= Input Ukuran Mobil =========")
    cek = True
    while cek:
        input_ukuran = int(input("Masukkan Ukuran Mobil (0-10): "))
        if input_ukuran >= 0 and input_ukuran <= 10:
            cek = False
        if cek:
```

```
            print("Input Tidak berada dalam Skala")
    print("========= Input Kenyamanan Mobil =========")
    cek = True
    while cek:
        input_kenyamanan = int(input("Masukkan Kenyamanan Mobil (0-10): "))
        if input_kenyamanan >= 0 and input_kenyamanan <= 10:
            cek = False
        if cek:
            print("Input Tidak berada dalam Skala")
    print("========= Input Keiritan Mobil =========")
    cek = True
    while cek:
        input_irit = int(input("Masukkan Iritnya Mobil (0-10): "))
        if input_irit >= 0 and input_irit <= 10:
            cek = False
        if cek:
            print("Input Tidak berada dalam Skala")
    print("========= Input Kecepatan Mobil =========")
    cek = True
    while cek:
        input_kecepatan = int(input("Masukkan Kecepatan Mobil (0-10): "))
        if input_kecepatan >= 0 and input_kecepatan <= 10:
            cek = False
        if cek:
            print("Input Tidak berada dalam Skala")
    print("========= Input Harga Mobil =========")
    cek = True
    while cek:
        input_harga = float(input("Masukkan Harga Mobil (Ratusan Juta): "))
        if input_kecepatan >= 0:
            cek = False
        if cek:
            print("Input Tidak dapat kurang dari 0")
elif pilihan == 2:
    df_input = pd.read_excel("./data/raw/test.xls")
    df_input[["Ukuran", "Kenyamanan", "Irit", "Kecepatan"]] = df_input[["Ukuran",
"Kenyamanan", "Irit", "Kecepatan"]].astype(float)
    input_ukuran = df_input.iloc[0][0]
    input_kenyamanan = df_input.iloc[0][1]
    input_irit = df_input.iloc[0][2]
    input_kecepatan = df_input.iloc[0][3]
    input_harga = df_input.iloc[0][4]
    if input_ukuran > 10 or input_ukuran < 0:
        raise ValueError("Nilai Ukuran tidak berada pada skala")
    if input_kenyamanan > 10 or input_kenyamanan < 0:
        raise ValueError("Nilai Kenyamanan tidak berada pada skala")
    if input_irit > 10 or input_irit < 0:
        raise ValueError("Nilai Irit tidak berada pada skala")
    if input_kecepatan > 10 or input_kecepatan < 0:
        raise ValueError("Nilai Kecepatan tidak berada pada skala")
    if input_harga < 0:
        raise ValueError("Nilai Harga tidak dapat kurang dari 0")
```

```
else:
    raise ValueError("Pilihan tidak valid")
```

3.2. Membaca File Training

```
df = pd.read_excel("./data/raw/Dataset Tugas 3.xls")
```

3.3. Normalisasi Data dan Menyiapkan Perhitungan (Pra Pemprosesan Data)

```
df_normalize = df.copy()

normal_attr = {
    "uku_min" : 0,
    "uku_max" : 10,
    "ken_min" : 0,
    "ken_max" : 10,
    "iri_min" : 0,
    "iri_max" : 10,
    "kec_min" : 0,
    "kec_max" : 10,
    "har_min" : 0,
    "har_max" : max(max(df["Harga (Ratus Juta)"]), input_harga),
}

df_normalize[["Ukuran", "Kenyamanan", "Irit", "Kecepatan"]] = df_normalize[["Ukuran",
"Kenyamanan", "Irit", "Kecepatan"]].astype(float)

for i in range(len(df_normalize)):
    df_normalize.at[i, "Ukuran"]= (df_normalize.iloc[i][1] - normal_attr["uku_min"]) /
(normal_attr["uku_max"] - normal_attr["uku_min"])
    df_normalize.at[i, "Kenyamanan"] = (df_normalize.iloc[i][2] - normal_attr["ken_min"]) /
(normal_attr["ken_max"] - normal_attr["ken_min"])
    df_normalize.at[i, "Irit"] = (df_normalize.iloc[i][3] - normal_attr["iri_min"]) /
(normal_attr["iri_max"] - normal_attr["iri_min"])
    df_normalize.at[i, "Kecepatan"] = (df_normalize.iloc[i][4] - normal_attr["kec_min"]) /
(normal_attr["kec_max"] - normal_attr["kec_min"])
    df_normalize.at[i, "Harga (Ratus Juta)"] = (df_normalize.iloc[i][5] -
normal_attr["har_min"]) / (normal_attr["har_max"] - normal_attr["har_min"])

class Mobil:
    def __init__(self, nama,ukuran,kenyamanan,irit,kecepatan,harga):
        self.nama = nama
        self.ukuran = ukuran
        self.kenyamanan = kenyamanan
        self.irit =  irit
        self.kecepatan = kecepatan
        self.harga = harga

    def PrintMobil(self):
        print("Mobil {}\n Ukuran: {}\n Kenyamanan: {}\n Irit: {}\n Kecepatan: {}\n Harga:
{}".format(self.nama, self.ukuran, self.kenyamanan, self.irit, self.kecepatan, self.harga))
```

```
mobil_data = []
#jumlah_distance = {}
for i in range(len(df)):
    mobil_temp = Mobil(df_normalize.iloc[i][0], df_normalize.iloc[i][1],
df_normalize.iloc[i][2], df_normalize.iloc[i][3], df_normalize.iloc[i][4],
df_normalize.iloc[i][5])
    mobil_data.append(mobil_temp)

inputusr = {
    "ukuran": (input_ukuran - normal_attr["uku_min"]) / (normal_attr["uku_max"] -
normal_attr["uku_min"]),
    "kenyamanan": (input_kenyamanan - normal_attr["ken_min"]) / (normal_attr["ken_max"] -
normal_attr["ken_min"]),
    "irit": (input_irit - normal_attr["iri_min"]) / (normal_attr["iri_max"] -
normal_attr["iri_min"]),
    "kecepatan": (input_kecepatan - normal_attr["kec_min"]) / (normal_attr["kec_max"] -
normal_attr["kec_min"]),
    "harga": (input_harga - normal_attr["har_min"]) / (normal_attr["har_max"] -
normal_attr["har_min"])
}

euclidean_perhitungan = []
manhattan_perhitungan = []
minkowski_perhitungan = []
supremum_perhitungan = []
```

3.4. Euclidean Distance

```
def hitungEuclidean(list_mobil, input_user):
    out_perhitungan = []
    for i in range(len(list_mobil)):
        out_temp = []
        perhitungan = ((input_user["ukuran"] - list_mobil[i].ukuran)**2 +
(input_user["kenyamanan"] - list_mobil[i].kenyamanan)**2 + (input_user["irit"] -
list_mobil[i].irit)**2 + (input_user["kecepatan"] - list_mobil[i].kecepatan)**2 +
(input_user["harga"] - list_mobil[i].harga)**2)**0.5
        out_temp.append(perhitungan)
        out_temp.append(list_mobil[i].nama)
        out_perhitungan.append(out_temp)
    return out_perhitungan

euclidean_perhitungan = hitungEuclidean(mobil_data, inputusr)
```

3.5. Manhattan Distance

Pada program ini atau perhitungan ini kami munggunakan rumus manhattan distance pada **Gambar (2.4).** Pertama kita akan menerima input dari user lalu meng-output kan 3 mobil yang direkomendasikan.

```
def hitungManhattan(list_mobil, input_user):
    out_perhitungan = []
```

```
    for i in range(len(list_mobil)):
        out_temp = []
        sum = 0
        sum = abs(input_user["ukuran"] - list_mobil[i].ukuran) +
abs(input_user["kenyamanan"] - list_mobil[i].kenyamanan) + abs(input_user["irit"] -
list_mobil[i].irit) + abs(input_user["kecepatan"] - list_mobil[i].kecepatan)+
abs(input_user["harga"] - list_mobil[i].harga)
        out_temp.append(sum)
        out_temp.append(list_mobil[i].nama)
        out_perhitungan.append(out_temp)
    return out_perhitungan

manhattan_perhitungan = hitungManhattan(mobil_data, inputusr)
```

3.6. Minkowski Distance

Pada perhitungan jarak Minkowski pangkat yang digunakan bernilai 1.1.

```
def hitungMinkowski(list_mobil, input_user, pangkat):
    out_perhitungan = []
    for i in range(len(list_mobil)):
        out_temp = []
        minkowski = (abs(float(input_user["ukuran"]-list_mobil[i].ukuran)) ** pangkat +
abs(float(input_user["kenyamanan"]-list_mobil[i].kenyamanan))**pangkat +
abs(float(input_user["irit"]-list_mobil[i].irit))**pangkat +
abs(float(input_user["kecepatan"]-list_mobil[i].kecepatan))**pangkat +
abs(float(input_user["harga"]-list_mobil[i].harga))**pangkat)**(1/pangkat)
        out_temp.append(minkowski)
        out_temp.append(list_mobil[i].nama)
        out_perhitungan.append(out_temp)
    return out_perhitungan

minkowski_perhitungan = hitungMinkowski(mobil_data, inputusr, 1.1)
```

3.7. Supremum Distance

```
def hitungSupremum(list_mobil, input_user):
    out_perhitungan = []
    for i in range(len(list_mobil)):
        out_temp = []
        supremum = max([[abs(input_user["ukuran"]-list_mobil[i].ukuran)],
[abs(input_user["kenyamanan"]-list_mobil[i].kenyamanan)],
[abs(input_user["irit"]-list_mobil[i].irit)],
[abs(input_user["kecepatan"]-list_mobil[i].kecepatan)],
[abs(input_user["harga"]-list_mobil[i].harga)]])
        out_temp.append(supremum[0])
        out_temp.append(list_mobil[i].nama)
        out_perhitungan.append(out_temp)
    return out_perhitungan

supremum_perhitungan = hitungSupremum(mobil_data, inputusr)
```

3.8. Klasifikasi kNN

```
euclidean_perhitungan = sorted(euclidean_perhitungan, key=lambda x:x[0])
manhattan_perhitungan = sorted(manhattan_perhitungan, key=lambda x:x[0])
minkowski_perhitungan = sorted(minkowski_perhitungan, key=lambda x:x[0])
supremum_perhitungan = sorted(supremum_perhitungan, key=lambda x:x[0])

df_euclidean = pd.DataFrame(euclidean_perhitungan[:3], columns=['Distance', 'Nama Mobil'])
df_manhattan = pd.DataFrame(manhattan_perhitungan[:3], columns=['Distance', 'Nama Mobil'])
df_minkowski = pd.DataFrame(minkowski_perhitungan[:3], columns=['Distance', 'Nama Mobil'])
df_supremum = pd.DataFrame(supremum_perhitungan[:3], columns=['Distance', 'Nama Mobil'])
```

3.9. Output Data

```
df_euclidean.to_excel("./data/processed/rekomendasi_eucledian.xls")
df_manhattan.to_excel("./data/processed/rekomendasi_manhattan.xls")
df_minkowski.to_excel("./data/processed/rekomendasi_minkowski.xls")
df_supremum.to_excel("./data/processed/rekomendasi_supremum.xls")
```

4. ANALISIS REKOMENDASI

Dari data yang kami punya seperti di bawah ini:

Nama Mobil	Ukuran	Kenyamanan	Irit	Kecepatan	Harga (Ratus Juta)
Toyota Agya	4	4	9	6	1
Daihatsu Alya	4	3	9	6	1.1
Toyota Avanza	6	5	6	6	2
Daihatsu Xenia	6	4	6	6	1.75
Xpander	7	7	6	7	2.25
Livina	7	7	6	7	2.1
Karimun	3	4	10	5	1.2
Toyota Innova	8	8	5	7	4
Alphard	9	10	4	8	10
Toyota Vios	5	7	9	8	2.5
Honda City	5	8	7	8	2.7
Toyota Hiace	10	5	8	6	5
Toyota Fortuner	9	8	5	8	5
Toyota Foxy	9	9	5	7	5.5
Toyota Corolla Altis	5	9	7	9	6
Suzuki Ertiga	7	7	7	7	2.3
Suzuki Carry	7	3	9	5	0.8

Dan input yang menjadi salah satu faktor perhitungan kami seperti di bawah ini:

Ukuran	Kenyamanan	Irit	Kecepatan	Harga (Ratus Juta)
3	5	2	7	9

Kami memiliki output dari setiap perhitungan seperti di bawah ini:
Euclidean

Distance	Nama Mobil
0.7615773106	Toyota Corolla Altis
0.8185352772	Alphard
0.8246211251	Toyota Innova

Manhattan

Distance	Nama Mobil
1.5	Toyota Avanza
1.5	Alphard
1.6	Toyota Innova

Minkowski

Distance	Nama Mobil
0.9662047132	Toyota Corolla Altis
0.9805717711	Alphard
1.022924959	Toyota Avanza

Supremum

Distance	Nama Mobil
0.5	Toyota Corolla Altis
0.5	Toyota Innova
0.6	Alphard

Dengan cara voting pada setiap perhitungan, kami bisa mendapatkan hasil bahwa mobil: Alphard dengan nilai voting 4, Toyota Innova dengan nilai 3, dan Toyota Corolla Altis dengan nilai 3 yang menjadi rekomendasi akhir kami berdasarkan input dan perhitungan tersebut.

5. KESIMPULAN

Berdasarkan tugas yang kami buat, kami berhasil mengimplementasikan algoritma kNN pada suatu sistem rekomendasi mobil dengan 4 teknik pencarian yang berbeda yaitu dengan Euclidean Distance, Manhattan Distance, Minkowski Distance, dan Supremum Distance yang kemudian akan diambil voting dari hasil perhitungan masing-masing teknik pencarian yang merupakan 3 terbaik.

Lampiran
Video Penjelasan oleh Kaenova Mahendra Auditama: https://youtu.be/Nvzp35KhXUQ
Video Penjelasan oleh Adhe Akram Azhari: https://youtu.be/kL1_5bZLJMM
Video Penjelasan oleh Elita Aurora Az Zahra: https://drive.google.com/file/d/1XMzxl3WNTlukpn82dAVSGUCmXPVIcVUP/view?usp=sharing
Notebooks Program: https://kaenova.github.io/AI_Tupro3/
Source Code Program: https://github.com/kaenova/AI_Tupro3

REFERENSI

[1]Altman, Naomi S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression" (PDF). The American Statistician. 46 (3): 175–185.

[2]"How to Scale data into the 0-1 range using Min-Max Normalization ...." 19 Oct. 2020, https://androidkt.com/how-to-scale-data-to-range-using-minmax-normalization/. Accessed 22 Jun. 2021.

[3](PDF) Perbandingan Akurasi Euclidean Distance, Minkowski Distance, dan Manhattan Distance pada Algoritma K-Means Clustering berbasis Chi-Square

[4] Li, Dapeng. (2018). "Euclidean Distance" https://www.sciencedirect.com/topics/computer-science/euclidean-distance