Tugas Besar Tahap Pertama Machine Learning

Clustering

oleh:

Muhammad Hafidh Raditya (NIM 1301184079)

IF-42-03

Program Studi S1 Informatika
Fakultas Informatika
Universitas Telkom
Bandung

2021

A. Formulasi Masalah

Pada tugas besar kali ini, saya mendapatkan dataset nomor 2 yang berisi tentang berbagai macam atribut yang memengaruhi apakah pada suatu hari akan turun salju atau tidak.

Tugas saya adalah untuk mendesain model clustering dari dataset tersebut, kemudian membuat prediksi jika nanti diberikan satu record baru dengan jenis atribut yang sama, apakah hari tersebut akan turun salju atau tidak.

B. Data Pre-Processing

Pertama-tama yang saya lakukan adalah mengecek pada dataset, apakah terdapat *record* yang memiliki attribute yang bernilai *null* atau tidak. Jika ada, maka *drop record* tersebut.

```
dataset.isnull().sum()
```

```
id                            0
Tanggal                       0
KodeLokasi                    0
SuhuMin                    1122
SuhuMax                     929
Hujan                      2431
Penguapan                 47024
SinarMatahari             52379
ArahAnginTerkencang        7744
KecepatanAnginTerkencang   7696
ArahAngin9am               7923
ArahAngin3pm               3197
KecepatanAngin9am          1353
KecepatanAngin3pm          2303
Kelembaban9am              2002
Kelembaban3pm              3374
Tekanan9am                11327
Tekanan3pm                11308
Awan9am                   41844
Awan3pm                   44471
Suhu9am                    1340
Suhu3pm                    2698
BersaljuHariIni            2431
BersaljuBesok              2431
dtype: int64
```

```
dataset.dropna(inplace=True)
dataset.shape
```

```
(42411, 24)
```

```
dataset.isnull().sum()
```

```
id                             0
Tanggal                        0
KodeLokasi                     0
SuhuMin                        0
SuhuMax                        0
Hujan                          0
Penguapan                      0
SinarMatahari                  0
ArahAnginTerkencang            0
KecepatanAnginTerkencang       0
ArahAngin9am                   0
ArahAngin3pm                   0
KecepatanAngin9am              0
KecepatanAngin3pm              0
Kelembaban9am                  0
Kelembaban3pm                  0
Tekanan9am                     0
Tekanan3pm                     0
Awan9am                        0
Awan3pm                        0
Suhu9am                        0
Suhu3pm                        0
BersaljuHariIni                0
BersaljuBesok                  0
dtype: int64
```

Setelah dataset bersih dari *record* yang bernilai *null*, saya akan mengecek korelasi atau keterkaitan antar attribut dengan menggunakan *heatmap*. Namun dikarenakan dataset tersebut memiliki beberapa atribut kategorikal, maka saya harus mengkonversikan atribut-atribut kategorikal tersebut menjadi atribut numerikal agar dapat dibanding nilainya dan dicek keterkaitannya menggunakan fungsi *corr()* pada bahasa Python.

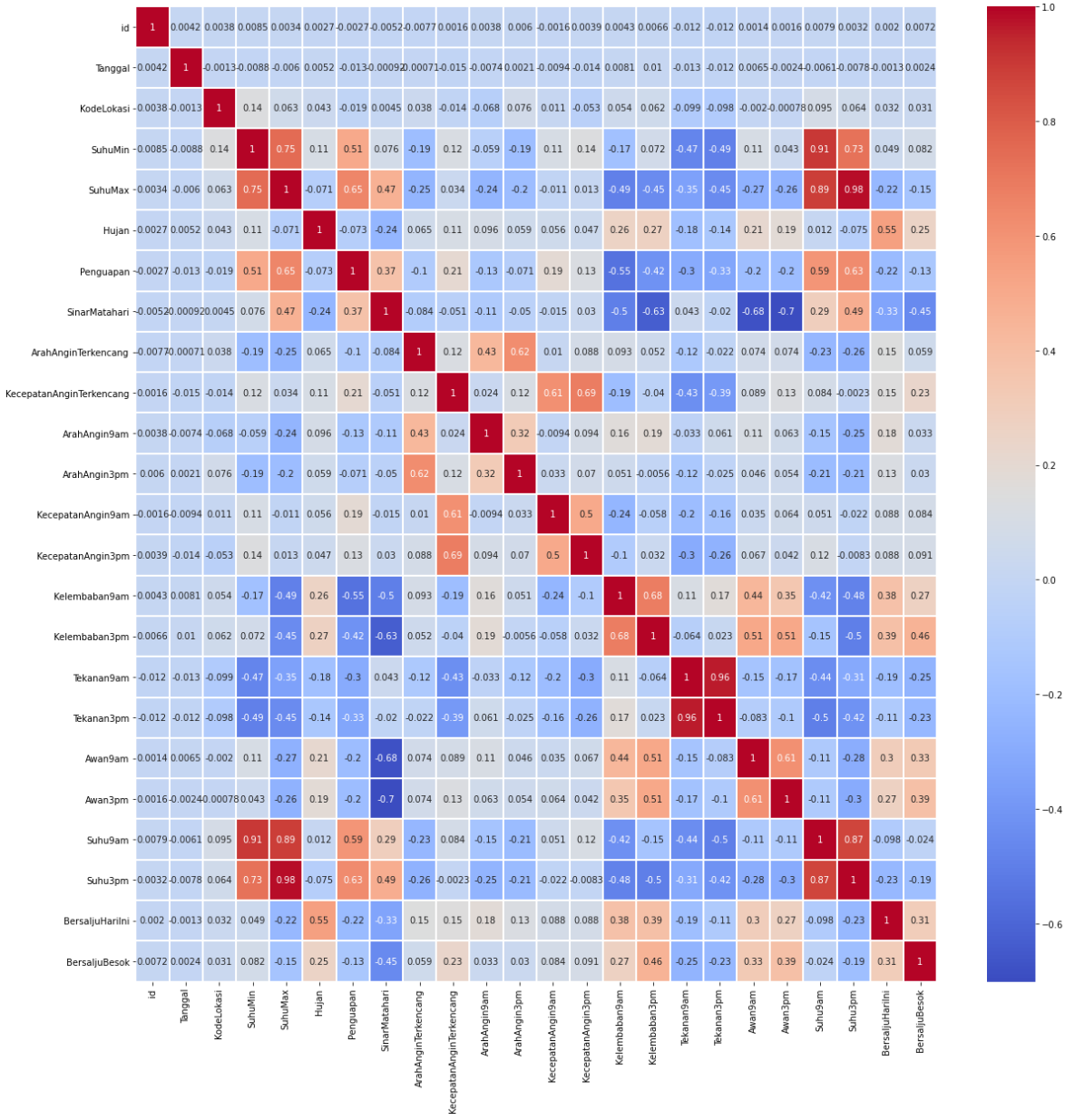
```
categorical = dataset.dtypes==object
categorical_cols = dataset.columns[categorical].tolist()
dataset[categorical_cols] = dataset[categorical_cols].apply(lambda col: LabelEncoder().fit_transform(col))
dataset[categorical_cols].head()
dataset
```

id	Tanggal	KodeLokasi	SuhuMin	SuhuMax	Hujan	Penguapan	SinarMatahari	ArahAnginTerkencang	KecepatanAnginTerkencang	...	Kelembaban9am	Kele
4	849	15	7.3	24.5	0.0	8.4	10.4	11	54.0	...	25.0	
5	3188	23	5.9	20.3	0.0	3.6	12.6	3	37.0	...	55.0	
6	1271	2	14.4	21.8	0.0	3.2	4.4	12	39.0	...	63.0	
7	1380	15	7.7	18.7	0.2	5.6	9.7	14	46.0	...	69.0	
9	1903	24	18.4	35.3	0.0	10.0	12.5	1	33.0	...	44.0	
...	...	...	...	...	...	...	...	...	...	...	...	
09081	232	18	16.8	34.1	0.0	12.8	10.3	1	85.0	...	48.0	
09083	2396	4	8.7	19.0	0.0	1.4	9.6	13	24.0	...	81.0	
09089	1877	6	14.3	26.2	0.0	8.0	12.6	5	50.0	...	51.0	
09091	3309	17	20.1	23.7	0.0	7.2	8.9	2	43.0	...	74.0	
09094	1696	1	10.8	29.8	0.0	7.8	11.2	0	48.0	...	35.0	

s × 24 columns

Setelah selesai proses konversi, barulah saya bisa membandingkan keterkaitan antar atribut menggunakan *heatmap*.

```
plt.figure(figsize=(20,20))
sns.heatmap(dataset.corr(),
            cmap='coolwarm',
            annot=True,linewidths=1);
```



Setelah saya menganalisis hasil dari *heatmap*, saya menyeleksi beberapa atribut yang memiliki nilai korelasi yang tergolong besar yang akan saya pakai selanjutnya. Atribut-atribut tersebut adalah BersaljuBesok, SuhuMin, SuhuMax, Suhu9am, Suhu3pm, Penguapan, SinarMatahari, Awan9am, Awan3pm, Kelembaban9am, dan Kelembaban3pm.

```
newdata = ["BersaljuBesok", "SuhuMin", "SuhuMax", "Suhu9am", "Suhu3pm", "Penguapan", "SinarMatahari", "Awa
databaru = dataset[newdata]
databaru.head()
databaru
```

	BersaljuBesok	SuhuMin	SuhuMax	Suhu9am	Suhu3pm	Penguapan	SinarMatahari	Awan9am	Awan3pm	Kelembaban9am	Kelembaban3pm
3	0	7.3	24.5	15.3	23.2	8.4	10.4	1.0	7.0	25.0	17.0
4	0	5.9	20.3	12.4	18.1	3.6	12.6	2.0	6.0	55.0	48.0
5	0	14.4	21.8	16.7	21.1	3.2	4.4	7.0	7.0	63.0	52.0
6	0	7.7	18.7	11.3	18.3	5.6	9.7	1.0	1.0	69.0	31.0
8	0	18.4	35.3	23.7	34.9	10.0	12.5	0.0	0.0	44.0	18.0
...	...	...	...	...	...	...	...	...	...	...	...
109080	0	16.8	34.1	25.6	33.0	12.8	10.3	1.0	4.0	48.0	28.0
109082	0	8.7	19.0	10.8	16.5	1.4	9.6	2.0	2.0	81.0	59.0
109088	0	14.3	26.2	21.1	25.5	8.0	12.6	0.0	2.0	51.0	37.0
109090	1	20.1	23.7	22.0	22.1	7.2	8.9	4.0	6.0	74.0	70.0
109093	0	10.8	29.8	21.7	29.2	7.8	11.2	0.0	1.0	35.0	18.0

42411 rows × 11 columns

Setelah itu, saya akan mengecek apakah pada setiap atribut memiliki *record* data yang tergolong sebagai *outlier* atau tidak menggunakan boxplot. Setelah saya cek, ternyata ada beberapa atribut yang memiliki outlier. Atribut-atribut tersebut adalah SuhuMin, SuhuMax, Suhu9am, Suhu3pm, Penguapan, dan Kelembaban9am. Maka saya akan *drop record* data yang tergolong sebagai *outlier* tersebut.

```
databaru.drop(databaru[databaru.SuhuMin < -5].index ,inplace =True)
databaru.drop(databaru[databaru.SuhuMax > 40].index ,inplace =True)
databaru.drop(databaru[databaru.Suhu9am > 37].index ,inplace =True)
databaru.drop(databaru[databaru.Suhu3pm > 41].index ,inplace =True)
databaru.drop(databaru[databaru.Penguapan > 13].index ,inplace =True)
databaru.drop(databaru[databaru.Kelembaban9am < 20].index ,inplace =True)
databaru
```

	BersaljuBesok	SuhuMin	SuhuMax	Suhu9am	Suhu3pm	Penguapan	SinarMatahari	Awan9am	Awan3pm	Kelembaban9am	Kelembaban3pm
3	0	7.3	24.5	15.3	23.2	8.4	10.4	1.0	7.0	25.0	17.0
4	0	5.9	20.3	12.4	18.1	3.6	12.6	2.0	6.0	55.0	48.0
5	0	14.4	21.8	16.7	21.1	3.2	4.4	7.0	7.0	63.0	52.0
6	0	7.7	18.7	11.3	18.3	5.6	9.7	1.0	1.0	69.0	31.0
8	0	18.4	35.3	23.7	34.9	10.0	12.5	0.0	0.0	44.0	18.0
...	...	...	...	...	...	...	...	...	...	...	...
109080	0	16.8	34.1	25.6	33.0	12.8	10.3	1.0	4.0	48.0	28.0
109082	0	8.7	19.0	10.8	16.5	1.4	9.6	2.0	2.0	81.0	59.0
109088	0	14.3	26.2	21.1	25.5	8.0	12.6	0.0	2.0	51.0	37.0
109090	1	20.1	23.7	22.0	22.1	7.2	8.9	4.0	6.0	74.0	70.0
109093	0	10.8	29.8	21.7	29.2	7.8	11.2	0.0	1.0	35.0	18.0

40421 rows × 11 columns

Setelah dataset bersih dari *null value* dan *outlier*, yang selanjutnya saya lakukan adalah menyeragamkan *scaling* pada setiap atribut. Hal ini dilakukan agar setiap atribut memiliki skala minimum dan maksimum yang sama yang nilainya berkisar antara 0 sampai 1. Tentunya proses ini juga memudahkan nanti dalam proses visualisasi clustering menggunakan *plot*.

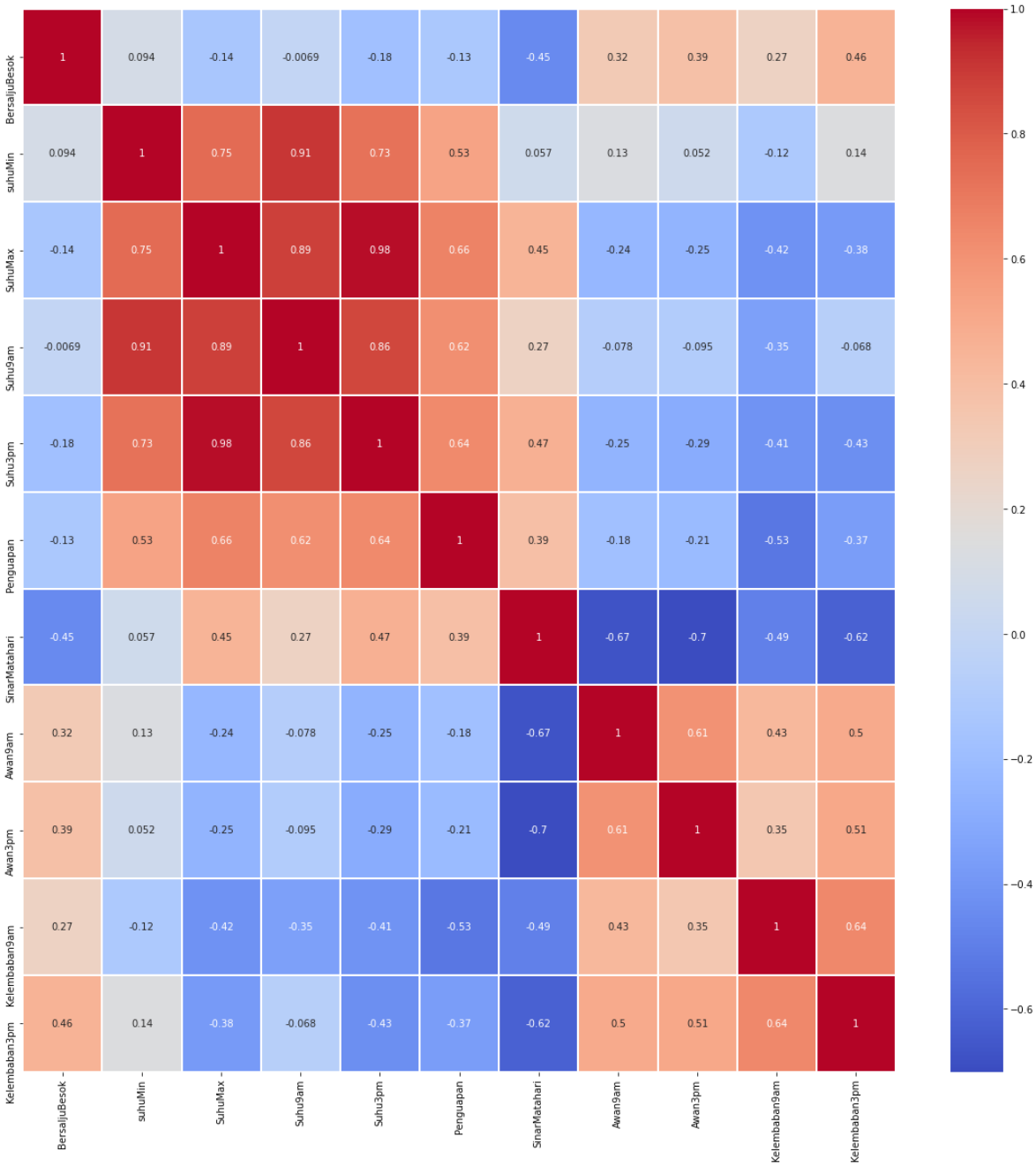
```
mms = MinMaxScaler()
scaler = mms.fit_transform(databaru)
col_new = ["BersaljuBesok", "suhuMin", "SuhuMax", "Suhu9am", "Suhu3pm", "Penguapan","SinarMa
scaledData = pd.DataFrame(scaler,columns=col_new)
scaledData
```

	BersaljuBesok	suhuMin	SuhuMax	Suhu9am	Suhu3pm	Penguapan	SinarMatahari	Awan9am	Awan3pm	Kelembaban9am	Kelembaban3pm
0	0.0	0.352601	0.540059	0.431267	0.535411	0.646154	0.727273	0.125	0.777778	0.0625	0.161616
1	0.0	0.312139	0.415430	0.353100	0.390935	0.276923	0.881119	0.250	0.666667	0.4375	0.474747
2	0.0	0.557803	0.459941	0.469003	0.475921	0.246154	0.307692	0.875	0.777778	0.5375	0.515152
3	0.0	0.364162	0.367953	0.323450	0.396601	0.430769	0.678322	0.125	0.111111	0.6125	0.303030
4	0.0	0.673410	0.860534	0.657682	0.866856	0.769231	0.874126	0.000	0.000000	0.3000	0.171717
...	...	...	...	...	...	...	...	...	...	...	...
40416	0.0	0.627168	0.824926	0.708895	0.813031	0.984615	0.720280	0.125	0.444444	0.3500	0.272727
40417	0.0	0.393064	0.376855	0.309973	0.345609	0.107692	0.671329	0.250	0.222222	0.7625	0.585859
40418	0.0	0.554913	0.590504	0.587601	0.600567	0.615385	0.881119	0.000	0.222222	0.3875	0.363636
40419	1.0	0.722543	0.516320	0.611860	0.504249	0.553846	0.622378	0.500	0.666667	0.6750	0.696970
40420	0.0	0.453757	0.697329	0.603774	0.705382	0.600000	0.783217	0.000	0.111111	0.1875	0.171717

40421 rows × 11 columns

Pada tahap ini, saya membuat dua macam dataset untuk disimpan, untuk kemudian dibandingkan pada tahap analisis dan kesimpulan. Dataset yang pertama adalah dataset yang sudah dilakukan proses *scaling*. Dataset yang kedua adalah dataset yang belum dilakukan proses *scaling.*

Lalu yang saya lakukan adalah mengecek *heatmap* korelasi dataset yang sudah bersih ini. Hal ini saya lakukan untuk mencari atribut mana yang paling berpengaruh terhadap *value* dari atribut “BersaljuBesok”. Dikarenakan tujuan dari tugas ini adalah untuk memprediksi apakah besok akan turun salju atau tidak, maka saya memilih atribut “BersaljuBesok” sebagai patokan.

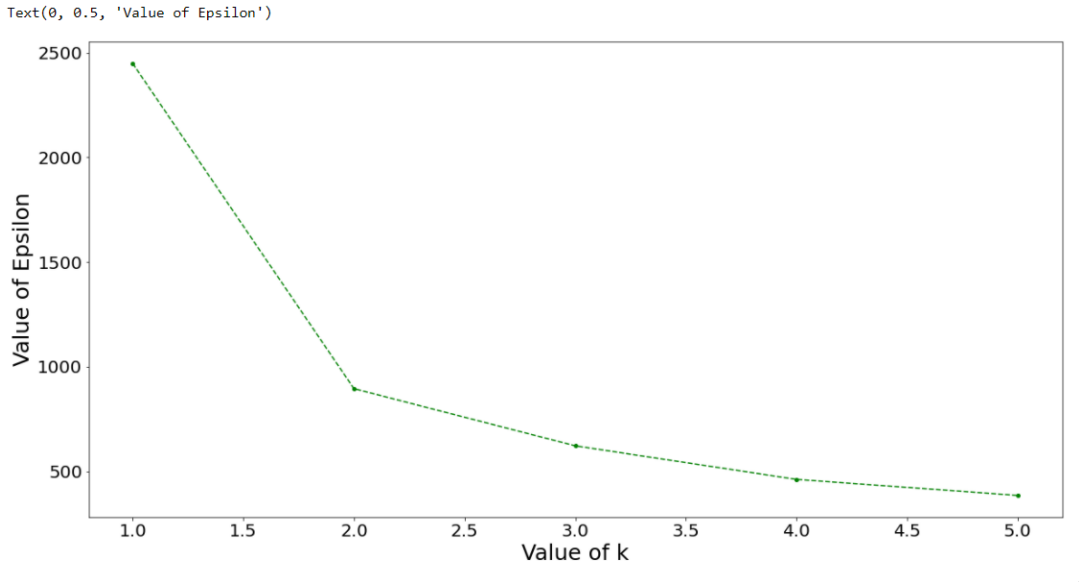


Setelah saya analisis, saya mendapatkan kalau atribut Awan3pm dan Kelembaban3pm merupakan atribut yang paling berpengaruh terhadap atribut BersaljuBesok. Maka saya akan menyeleksi kedua atribut tersebut untuk dilakukan proses *clustering*

C. Proses Clustering

Sebelum saya melakukan proses *clustering* menggunakan metode *KMeans*, saya melakukan proses evaluasi terlebih dahulu menggunakan metode *Sum Square Error* (SSE) dan metode *Elbow* untuk mencari berapa nilai k atau jumlah *cluster* yang optimal untuk digunakan pada dataset ini.

```
epsilon = list(range(5))

for k in range(1,6):

    cluster = pd.read_csv("scaled_salju_train.csv", usecols=["Awan3pm", "Kelembaban3pm"], nrows=20000)

    rows = cluster.shape[0]
    cols = cluster.shape[1]

    centroids = cluster.loc[np.random.randint(1,rows+1,k)]
    centroids['new'] = list(range(1,k+1))
    centroids.set_index('new',inplace = True)
    d = np.random.rand(rows)

    number_of_iterations = 15
    temp_epsilon = list(range(number_of_iterations))

    for i in range(0,number_of_iterations):

        for j in range(0,rows):
            d[j] = ((centroids - cluster.loc[j])**2).sum(axis = 1).idxmin()
        cluster['centroid number'] = d

        mean_x = list(range(k))
        mean_y = list(range(k))
        for m in range(0,k):
            mean_x[m] = cluster[cluster['centroid number'] == (m+1)]['Awan3pm'].mean()
            mean_y[m] = cluster[cluster['centroid number'] == (m+1)]['Kelembaban3pm'].mean()
        centroids.replace(list(centroids['Awan3pm']),mean_x,inplace = True)
        centroids.replace(list(centroids['Kelembaban3pm']),mean_y,inplace = True)

        z = list(range(k))
        for p in range(0,k):
            z[p] = ((cluster[cluster['centroid number'] == p+1][['Awan3pm','Kelembaban3pm']] - centroids.iloc[p])**2).values.sum(
        temp_epsilon[i] = sum(z)

        epsilon[k-1] = temp_epsilon[i]

    %reset_selective -f centroids
```



Setelah saya analisis grafik *elbow* yang telah saya buat, pada titik k=2 sudah terbentuk pola siku atau *elbow*. Sehingga dapat disimpulkan kalau jumlah *cluster* yang optimal adalah 2 *cluster*.

Selanjutnya adalah langkah untuk proses *clustering*. Ada 3 buah fungsi yang saya siapkan untuk mempermudah proses ini.

```
def manhattanDistance(centroid, data):
    hasil = abs(float(centroid[0]-data[0])) + abs(float(centroid[1]-data[1]))
    return hasil
```

Yang pertama adalah fungsi manhattanDistance untuk mencari nilai *distance* atau jarak antara dua buah titik.

```
def newCentroid(cluster):
    x = 0
    y = 0
    for i in range(len(cluster)):
        x = x+cluster[i][0]
        y = y+cluster[i][1]
    averageX = x/len(cluster)
    averageY = y/len(cluster)
    centroid = [averageX, averageY]
    return centroid
```

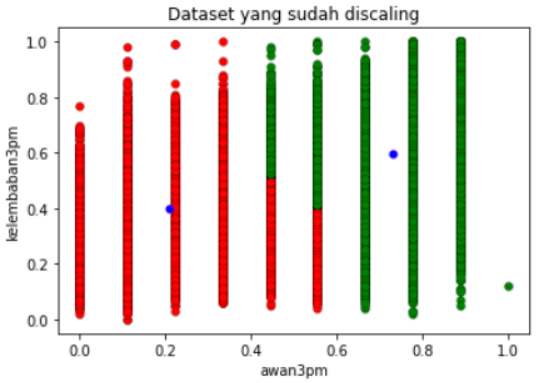
Yang kedua adalah fungsi newCentroid. Fungsi ini digunakan untuk meng-*generate centroid* baru pada saat tiap kali iterasi pada proses *clustering*.

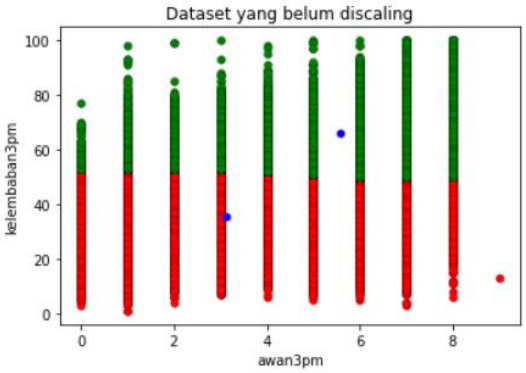
```
def KMeans(dataset, max_iteration):
    centroid1 = dataset[rd.randint(0,39744)]
    centroid2 = dataset[rd.randint(0,39744)]
    selisih = 1
    iteration = 0
    while (selisih!=0) and (iteration<max_iteration):
        cluster1 = []
        cluster2 = []
        oldCentroid1 = centroid1
        oldCentroid2 = centroid2
        for j in range(len(dataset)):
            distance1 = manhattanDistance(oldCentroid1, dataset[j])
            distance2 = manhattanDistance(oldCentroid2, dataset[j])
            if distance1<distance2:
                cluster1.append(dataset[j])
            else:
                cluster2.append(dataset[j])
        centroid1 = newCentroid(cluster1)
        centroid2 = newCentroid(cluster2)
        selisih = (centroid1[0]-oldCentroid1[0])+(centroid1[1]-oldCentroid1[1])+(centroid2[0]-oldCentroid2[0])+(centroid2[1]-old
        iteration += 1
    centroids = [centroid1, centroid2]
    return centroids, cluster1, cluster2
```

Dan yang terakhir adalah fungsi Kmeans yaitu fungsi utama dari proses *KMeans clustering*. Jika dijelaskan menggunakan bahasa natural, maka algoritma KMeans memiliki langkah-langkah sebagai berikut:

1. Buat 2 centroid baru secara random.
2. Hitung jarak antara setiap data terhadap kedua centroid yang sudah dibuat.
3. Jika jarak terhadap centroid 1 lebih kecil daripada jarak terhadap centroid 2, maka masukkan data tersebut ke dalam cluster 1. Begitu juga sebaliknya.
4. Buat 2 buah centroid baru dari nilai rata-rata pada masing-masing cluster.
5. Ulangi langkah 2-4 sampai menemui kondisi dimana centroid pada iterasi ke-n sama dengan centroid pada iterasi ke n+1, atau pada saat jumlah iterasi menyentuh nilai max_iteration yang sudah ditentukan.

D. Analisis





	Awan3pm	Kelembaban3pm	Cluster
0	0.111111	0.303030	Tidak Bersalju
1	0.000000	0.171717	Tidak Bersalju
2	0.111111	0.353535	Tidak Bersalju
3	0.444444	0.282828	Tidak Bersalju
4	0.000000	0.111111	Tidak Bersalju
...	...	...	...
21537	0.777778	0.484848	Bersalju
21538	0.555556	0.676768	Bersalju
21539	0.777778	0.656566	Bersalju
21540	0.777778	0.717172	Bersalju
21541	0.666667	0.696970	Bersalju

40421 rows × 3 columns

```
scaledFrameClusters.groupby("Cluster").size()
```

```
Cluster
Bersalju          21542
Tidak Bersalju    18879
dtype: int64
```

Sudah discaling

	Awan3pm	Kelembaban3pm	Cluster
0	7.0	17.0	Tidak Bersalju
1	6.0	48.0	Tidak Bersalju
2	1.0	31.0	Tidak Bersalju
3	0.0	18.0	Tidak Bersalju
4	1.0	36.0	Tidak Bersalju
...	...	...	...
20478	5.0	68.0	Bersalju
20479	7.0	66.0	Bersalju
20480	7.0	72.0	Bersalju
20481	2.0	59.0	Bersalju
20482	6.0	70.0	Bersalju

40421 rows × 3 columns

```
unscaledFrameClusters.groupby("Cluster").size()
```

```
Cluster
Bersalju          20483
Tidak Bersalju    19938
dtype: int64
```

Belum discaling

Setelah saya lakukan proses labeling pada cluster 1 dan cluster 2 pada kedua jenis dataset tersebut, secara garis besar, data pada cluster 1 dan cluster 2 antara kedua dataset yang sudah dilakukan scaling dan belum dilakukan scaling memiliki jumlah yang tidak begitu

berbeda jauh, namun memiliki persebaran data pada cluster yang sangat berbeda. Namun jika dilihat visualisasi atau peta persebaran clusternya, asumsi saya adalah cluster yang dibuat menggunakan dataset yang sudah dilakukan scaling lebih baik daripada cluster yang dibuat menggunakan dataset yang belum dilakukan scaling. Alasannya adalah jika dilihat pada visualisasi datanya, pada dataset yang sudah dilakukan scaling, terdapat peralihan antara cluster 1 dan cluster 2 yang cukup jelas. Dapat dilihat pada saat interval atribut awan3pm berkisar antara 0.4-0.6, terdapat dua jenis cluster pada interval tersebut yang menunjukkan bahwa terdapat peralihan yang jelas antara cluster 1 dan cluster 2. Namun analisis ini masih berupa hipotesis saya saja, dikarenakan untuk melihat bagus atau tidaknya proses clustering yang sudah dibuat, harus dilakukan proses klasifikasi untuk mengecek akurasi pada proses clustering yang sudah dibuat, yang artinya saya harus melakukan proses klasifikasi pada tubes tahap 2 terlebih dahulu untuk mengetahui jenis dataset mana, yang sudah discaling atau belum discaling, yang lebih baik akurasinya.

E. Kesimpulan

Kesimpulan yang saya dapat dari tugas kali ini adalah, ternyata jika hasil clustering pada dua jenis dataset yang sudah discaling dan belum discaling memiliki persebaran data yang sangat berbeda, terbukti pada saat dilihat dari visualisasi datanya. Namun untuk mengetahui jenis dataset mana yang lebih baik, harus dilakukan proses klasifikasi terlebih dahulu untuk mengetahui akurasi clusteringnya.