LAPORAN
PEMBELAJARAN MESIN

Laporan ini dibuat untuk memenuhi Tugas Pemrograman Tahap 1
Pembelajaran Mesin IF-42-03

Disusun Oleh :

Adabi Raihan Muhammad 1301180379

Telkom University

Bandung

2021

Formulasi Masalah

Pada tugas machine learning tahap 1 ini, saya diharuskan membuat model clustering dari dataset salju yang nantinya akan memprediksi apakah di hari tersebut akan turun salju / tidak.

Data Exploration

Tahap pertama dalam tugas ini yaitu Data Exploration, berguna untuk melihat kriteria dari dataset yang ingin diolah (clustering) berikut metode Data Exploration yang saya gunakan dalam dataset salju :

- data.head()

```
data.head()
```

	id	Tanggal	KodeLokasi	SuhuMin	SuhuMax	Hujan	Penguapan	SinarMatahari	ArahAnginTerkencang	KecepatanAnginTerkencang	...	Kelembaban9am
0	1	01/06/2014	C4	10.4	15.5	4.8	NaN	NaN	WSW	24.0	...	78.0
1	2	15/07/2014	C10	9.0	17.0	8.0	2.6	7.4	NaN	NaN	...	80.0
2	3	16/02/2011	C46	18.2	32.0	0.0	NaN	NaN	ESE	44.0	...	62.0
3	4	08/08/2012	C36	7.3	24.5	0.0	8.4	10.4	SSW	54.0	...	25.0
4	5	29/10/2016	C7	5.9	20.3	0.0	3.6	12.6	N	37.0	...	55.0

5 rows × 24 columns

Berfungsi untuk melihat 5 data pertama, saya menggunakan metode data.head() karena ingin melihat apa saja isi dari dataset salju dengan hanya menampilkan 5 buat data.

- data.describe()

```
data.describe()
```

	id	SuhuMin	SuhuMax	Hujan	Penguapan	SinarMatahari	KecepatanAnginTerkencang	KecepatanAngin9am	KecepatanA
count	109095.000000	107973.000000	108166.000000	106664.000000	62071.000000	56716.000000	101399.000000	107742.000000	10679
mean	54548.000000	12.196183	23.214819	2.385005	5.462440	7.599527	40.032002	14.052115	1
std	31493.158146	6.389419	7.106596	8.588155	4.201638	3.789042	13.617554	8.926092	
min	1.000000	-8.500000	-4.800000	0.000000	0.000000	0.000000	7.000000	0.000000	
25%	27274.500000	7.600000	17.900000	0.000000	2.600000	4.800000	31.000000	7.000000	1
50%	54548.000000	12.000000	22.600000	0.000000	4.800000	8.400000	39.000000	13.000000	1
75%	81821.500000	16.800000	28.200000	0.800000	7.400000	10.600000	48.000000	19.000000	2
max	109095.000000	33.900000	47.300000	371.000000	145.000000	14.300000	135.000000	130.000000	8

Berfungsi untuk melihat count, mean, dll dari dataset salju.

- data.info()

```
In [4]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 109095 entries, 0 to 109094
Data columns (total 24 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   id                         109095 non-null  int64
 1   Tanggal                    109095 non-null  object
 2   KodeLokasi                 109095 non-null  object
 3   SuhuMin                    107973 non-null  float64
 4   SuhuMax                    108166 non-null  float64
 5   Hujan                      106664 non-null  float64
 6   Penguapan                  62071 non-null   float64
 7   SinarMatahari              56716 non-null   float64
 8   ArahAnginTerkencang        101351 non-null  object
 9   KecepatanAnginTerkencang   101399 non-null  float64
 10  ArahAngin9am               101172 non-null  object
 11  ArahAngin3pm               105898 non-null  object
 12  KecepatanAngin9am          107742 non-null  float64
 13  KecepatanAngin3pm          106792 non-null  float64
 14  Kelembaban9am              107093 non-null  float64
 15  Kelembaban3pm              105721 non-null  float64
 16  Tekanan9am                 97768 non-null   float64
 17  Tekanan3pm                 97787 non-null   float64
 18  Awan9am                    67251 non-null   float64
 19  Awan3pm                    64624 non-null   float64
 20  Suhu9am                    107755 non-null  float64
 21  Suhu3pm                    106397 non-null  float64
 22  BersaljuHariIni            106664 non-null  object
 23  BersaljuBesok              106664 non-null  object
dtypes: float64(16), int64(1), object(7)
memory usage: 20.0+ MB
```

Berfungsi untuk melihat tipe data dari tiap column dan juga melihat data mana yang mempunyai nilai null paling tinggi.

- data.corr()

```
data.corr()
```

	id	SuhuMin	SuhuMax	Hujan	Penguapan	SinarMatahari	KecepatanAnginTerkencang	KecepatanAngin9am	KecepatanA
id	1.000000	0.007520	0.004129	0.001892	-0.002019	-0.003793	-0.003330	0.001467	-
SuhuMin	0.007520	1.000000	0.735500	0.104391	0.465508	0.072501	0.177627	0.174358	
SuhuMax	0.004129	0.735500	1.000000	-0.074669	0.585475	0.470991	0.068244	0.013632	
Hujan	0.001892	0.104391	-0.074669	1.000000	-0.058940	-0.226845	0.135081	0.087157	
Penguapan	-0.002019	0.465508	0.585475	-0.058940	1.000000	0.364745	0.198965	0.188950	
SinarMatahari	-0.003793	0.072501	0.470991	-0.226845	0.364745	1.000000	-0.035040	0.002004	
KecepatanAnginTerkencang	-0.003330	0.177627	0.068244	0.135081	0.198965	-0.035040	1.000000	0.604677	
KecepatanAngin9am	0.001467	0.174358	0.013632	0.087157	0.188950	0.002004	0.604677	1.000000	
KecepatanAngin3pm	-0.000819	0.174122	0.050793	0.059562	0.128217	0.051812	0.686629	0.518093	
Kelembaban9am	0.002330	-0.232101	-0.504014	0.223661	-0.499698	-0.489242	-0.214062	-0.272566	-
Kelembaban3pm	0.002229	0.007974	-0.507921	0.254898	-0.387161	-0.627920	-0.026049	-0.032366	
Tekanan9am	-0.004429	-0.449684	-0.331431	-0.168638	-0.268648	0.041321	-0.460665	-0.229197	-
Tekanan3pm	-0.004715	-0.459596	-0.426012	-0.127506	-0.291740	-0.020052	-0.415953	-0.176461	-
Awan9am	0.001144	0.077349	-0.291573	0.195540	-0.181615	-0.676965	0.071323	0.024425	
Awan3pm	0.000388	0.021105	-0.279039	0.170791	-0.181563	-0.703482	0.105835	0.051593	
Suhu9am	0.006948	0.901208	0.886984	0.011781	0.542808	0.291601	0.150043	0.127675	
Suhu3pm	0.004265	0.707578	0.984434	-0.079160	0.570758	0.490993	0.033001	0.004132	

Berfungsi untuk melihat korelasi antar data.

Data Pre-Processing

Tahap kedua ada Data Pre-Processing yang bertujuan untuk memilah data mana saja yang penting / tidak, guna saat melakukan olah data (clustering) akan menghasilkan data cluster yang optimal, berikut metode yang saya gunakan untuk Data Pre-Processing:

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

```
(42411, 24)
```

```
data.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
```

Drop atribut yang memilki nilai null secara automatis, lalu cek apakah masih ada/tidak yang memiliki nilai null pada setiap atribut.

Categorical attribute -> Numerical Attribute

```
Category = data.dtypes==object
CategoryColumn = data.columns[Category].tolist()

data[CategoryColumn] = data[CategoryColumn].apply(lambda col: LabelEncoder().fit_transform(col))
data[CategoryColumn].head()

data
```

	id	Tanggal	KodeLokasi	SuhuMin	SuhuMax	Hujan	Penguapan	SinarMatahari	ArahAnginTerkencang	KecepatanAnginTerkencang	...	Kelembaba
3	4	849	15	7.3	24.5	0.0	8.4	10.4	11	54.0	...	
4	5	3188	23	5.9	20.3	0.0	3.6	12.6	3	37.0	...	
5	6	1271	2	14.4	21.8	0.0	3.2	4.4	12	39.0	...	
6	7	1380	15	7.7	18.7	0.2	5.6	9.7	14	46.0	...	
8	9	1903	24	18.4	35.3	0.0	10.0	12.5	1	33.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
109080	109081	232	18	16.8	34.1	0.0	12.8	10.3	1	85.0	...	
109082	109083	2396	4	8.7	19.0	0.0	1.4	9.6	13	24.0	...	
109088	109089	1877	6	14.3	26.2	0.0	8.0	12.6	5	50.0	...	
109090	109091	3309	17	20.1	23.7	0.0	7.2	8.9	2	43.0	...	
109093	109094	1696	1	10.8	29.8	0.0	7.8	11.2	0	48.0	...	

42411 rows × 24 columns

Men-konversi atribut category → numerical agar data dapat diolah.

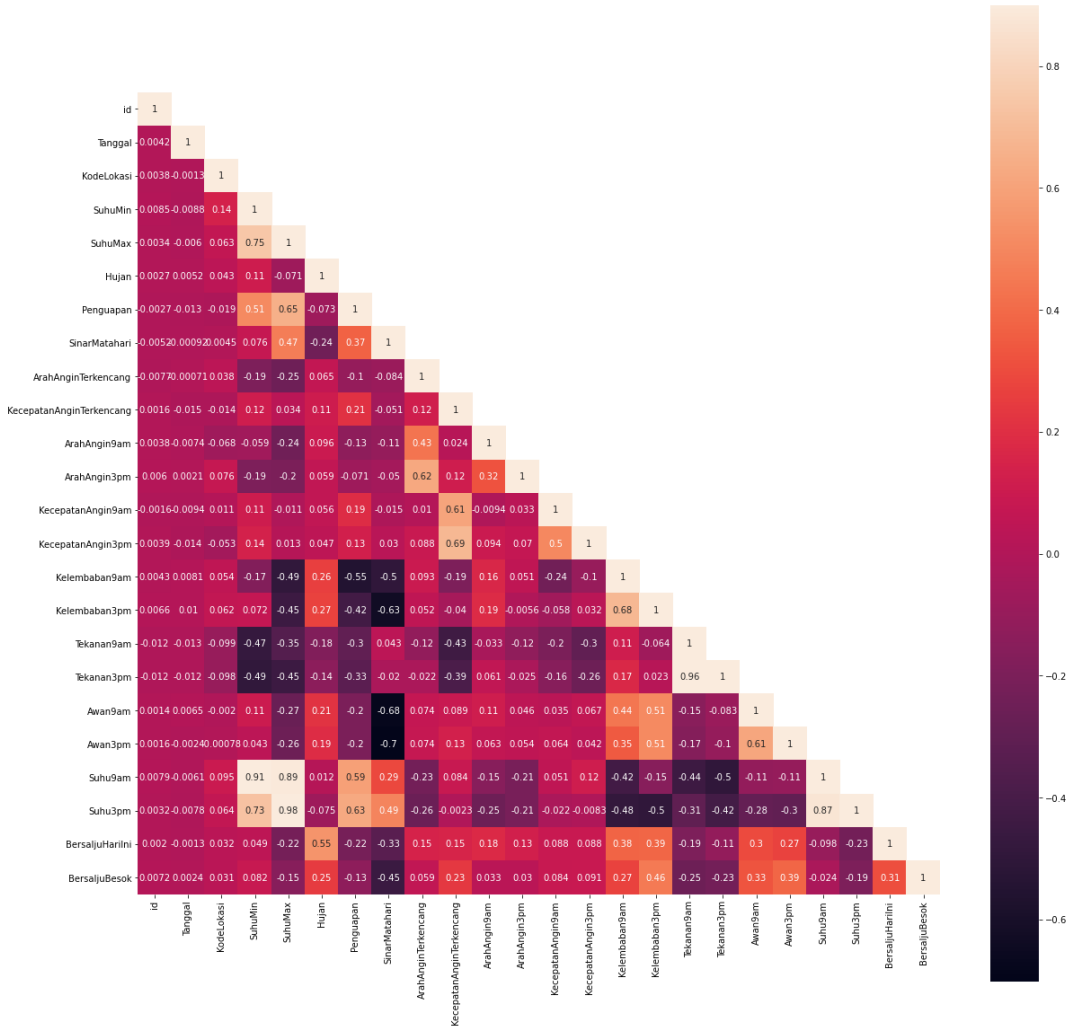
Visualisasi Data

Tahapan ketiga ada visualisasi data, bertujuan untuk mempermudah melihat data mana saja yang memiliki tingkat korelasi tinggi, lalu saya juga membuat sebuah array yang isinya merupakan data yang memiliki korelasi tinggi, lalu saya juga menggunakan boxplot untuk melihat apakah data tersebut (korelasi tinggi) memiliki outlier / tidak, jika memiliki outlier akan di drop outlier tersebut.

Visualisasi

```
#Buat korelasi dengan heatmap

corr = data.corr(method = 'pearson')

temp = np.array(corr)

temp[np.tril_indices_from(temp)] = False
fig, ax = plt.subplots(figsize = (10,10))
fig.set_size_inches(20,20)
sns.heatmap(corr, temp = temp, vmax = 0.9, square = True, annot = True)
```

```
<AxesSubplot:>
```



Saya menggunakan visualisasi data berupa Heat Map agar dapat dengan mudah melihat korelasi data yang nantinya akan saya olah.

SELEKSI DATA YANG PUNYA KORELASI TINGGI

```
KorelasiTinggi = ["BersaljuBesok", "SuhuMin", "SuhuMax", "Suhu9am", "Suhu3pm", "Penguapan", "SinarMatahari", "Awan9am", "Awan3pm"
KorelasiTinggi = data[KorelasiTinggi]
KorelasiTinggi.head()
KorelasiTinggi
```

	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

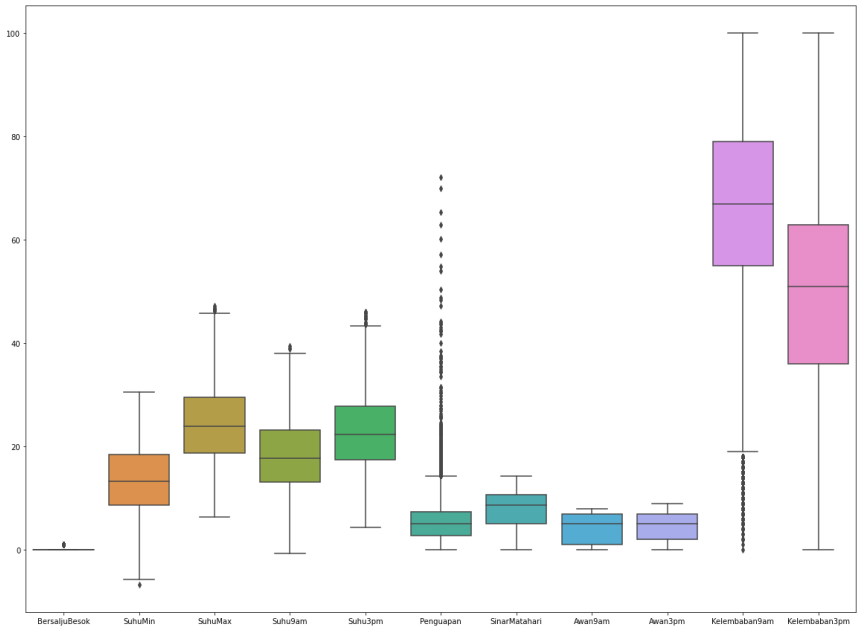
42411 rows × 11 columns

Setelah melihat dengan menggunakan heat map, saya mendapatkan bahwa data-data berikut memiliki korelasi paling tinggi diantara data-data lainnya.

KorelasiTinggi = ["BersaljuBesok", "SuhuMin", "SuhuMax", "Suhu9am", "Suhu3pm", "Penguapan", "SinarMatahari", "Awan9am", "Awan3pm", "Kelembaban9am", "Kelembaban3pm"]

Menampilkan Boxplot

```
fig = plt.figure(figsize = (20,2))
fig.set_size_inches(20,15)
sns.boxplot(data = KorelasiTinggi)
plt.show()
```



Saya menggunakan Boxplot sebagai visualisasi data selanjutnya untuk melihat outlier yang terdapat pada setiap data yang nantinya akan di drop (jika memiliki outlier).

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

KorelasiTinggi
```

	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 mendapatkan outlier dari boxplot sebelumnya, saya melakukan drop sesuai nilai outlier dari setiap data.

Scaling

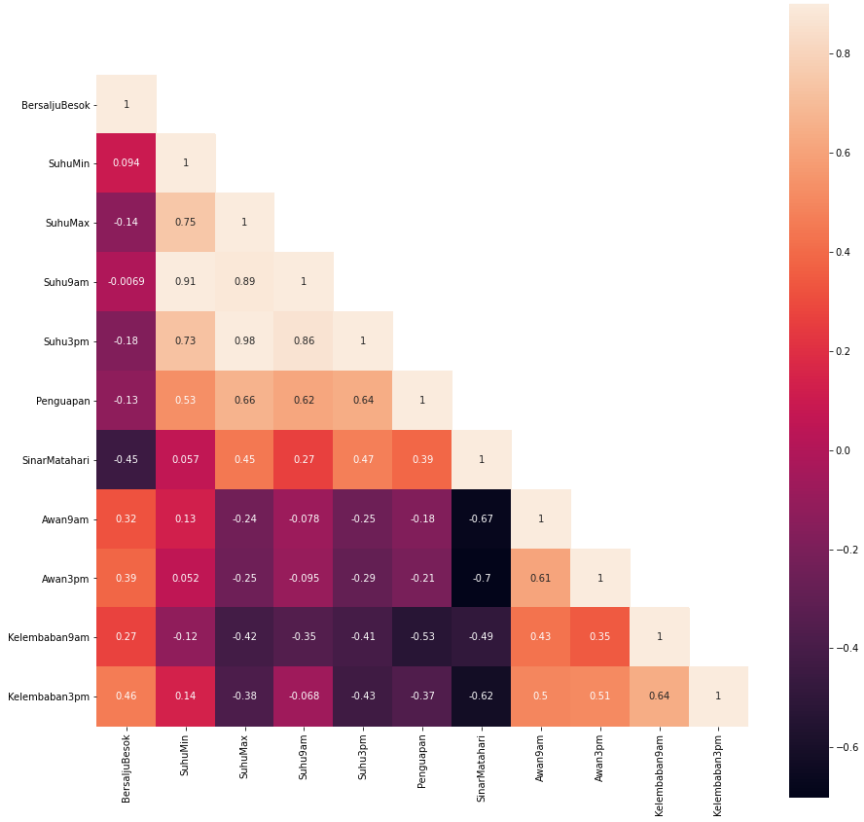
Tahapan kelima yaitu melakukan scaling terhadap data yang sebelumnya, scaling bertujuan agar setiap atribut dari data tesebut memiliki skala minimum dan maksimum yang sama nilai nya (0 sampai 1), dan juga agar mempermudah pemrosesan data di tahap selanjutnya.

```
Scaler = MinMaxScaler()
Scaling = Scaler.fit_transform(KorelasiTinggi)
NewCol = ["BersaljuBesok", "SuhuMin", "SuhuMax", "Suhu9am", "Suhu3pm", "Penguapan", "SinarMatahari", "Awan9am", "Awan3pm", "Kelem
AfterScaling = pd.DataFrame(Scaling,columns=NewCol)

AfterScaling
```

	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

Di tahap scaling saya menggunakan fungsi MinMaxScaler() untuk melakukan scaling terhadap data data yang sudah ada, setelah di scaling dapat dilihat bahwa semua atribut sudah mempunyai nilai 0 sampai 1.



Setelah data di scaling saya menggunakan heatmap lagi untuk memastikan apa data benar sudah di scaling / belum.

```
AfterScaling.to_csv('Data_After_Scaling.csv')
```

Lalu saya memasukan data yang sudah di scaling menjadi file baru agar dapat mudah di proses di tahap selanjutnya tanpa ada nya data ambigu.

```
Scaled = pd.read_csv("Data_After_Scaling.csv", usecols=["Awan3pm", "Kelembaban3pm", "BersaljuBesok"])
Scaled.head()
```

Saya juga melakukan perubahan dari dataframe menjadi sebuah array agar memudahkan pemroresan data.

```
dfAwan3pm = Scaled.Awan3pm
dfKelembaban3pm = Scaled.Kelembaban3pm

ScaledArray = []
i = 0

while i<len(Scaled):
    AllScaledArray = [dfAwan3pm[i], dfKelembaban3pm[i]]
    ScaledArray.append(AllScaledArray)
    i+=1
```

Mengevaluasi dataset dengan menentukan nilai K dengan Elbow Method

Di tahapan ini, saya menggunakan elbow method untuk mengetahui berapa nilai K optimal untuk nanti saat clustering.

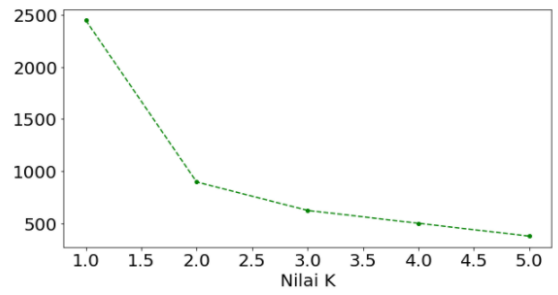
NOTE : Dalam melakukan Elbow Method ini saya mengambil referensi dari

https://medium.com/analytics-vidhya/elbow-method-of-k-means-clustering-algorithm-a0c916adc540

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

for k in range(1,6):

    cluster = pd.read_csv("Data_After_Scaling.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
```

```
k = list(range(1,6))
plt.figure(figsize=(10,5))
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.plot(k,epsilon,'go--', linewidth=1.5, markersize=4)
plt.xlabel


plt.show()
```

```
k = list(range(1,6))
plt.figure(figsize=(10,5))
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.plot(k,epsilon,'go--', linewidth=1.5, markersize=4)
plt.xlabel('Nilai K', fontsize = 20)


plt.show()
```



Setelah menggunakan Elbow Method didapati bahwa tekukan pertama ada di angka 2 yang menjadikan angka 2 tersebut adalah nilai K yang paling optimal diantara nilai lainnya.

Clustering

Di proses Clustering saya menggunakan 3 buah fungsi yaitu :

- Euclidean Distance

```
#Euclidean Distance
def Euclidean(centroid, data):
    Hasil = 0
    for i in range(len(data)-1):
         Hasil += (data[i] - centroid[i])**2
    return math.sqrt(Hasil)
```

Fungsi Euclidean Distance bertujuan untuk mencari jarak antara dua titik.

- Create Centroid

```
#Centroid
def CreateCentroid(cluster):
    x = 0
    y = 0

    for i in range(len(cluster)):
        x = x + cluster[i][0]
        y = y + cluster[i][1]

    avgX = x/len(cluster)
    avgY = y/len(cluster)
    centroid = [avgX, avgY]
    return centroid
```

Fungsi Create Centroid bertujuan untuk membuat centroid baru di proses clustering.

- KMeans

```
#KMeans
def KMeans(data, max_iteration):
    centro1 = data[rd.randint(0,39744)]
    centro2 = data[rd.randint(0,39744)]
    selisih = 1
    i = 0

    while (selisih!=0) and (i < max_iteration):
        cluster1 = []
        cluster2 = []
        TempCentro1 = centro1
        TempCentro2 = centro2
        for j in range(len(data)):
            distance1 = Euclidean(TempCentro1, data[j])
            distance2 = Euclidean(TempCentro2, data[j])
            if distance1<distance2:
                cluster1.append(data[j])
            else:
                cluster2.append(data[j])
        centro1 = CreateCentroid(cluster1)
        centro2 = CreateCentroid(cluster2)
        selisih = (centro1[0]-TempCentro1[0])+(centro1[1]-TempCentro1[1])+(centro2[0]-TempCentro2[0])+(centro2[1]-TempCentro2[1])
        i += 1

    NewCentroids = [centro1, centro2]

    return NewCentroids, cluster1, cluster2
```

```
: NewCentroids, cluster1, cluster2 = KMeans(ScaledArray, 100)
```

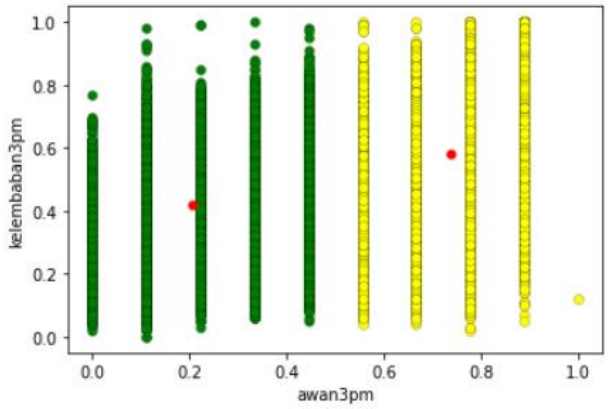
Fungsi KMeans bertujuan untuk melakukan tujuan utama yaitu Clustering.

Hasil

Di tahapan terakhir ini saya akan menampilkan penyebaran cluster nya pada data yang sebelumnya telah melalui beberapa tahapan.

```
awan3pmCentroids1 = []
awan3pmCentroids2 = []
kelembaban3pmCentroids1 = []
kelembaban3pmCentroids2 = []

for k in range(len(cluster1)):
    awan3pmCentroids1.append(cluster1[k][0])
    kelembaban3pmCentroids1.append(cluster1[k][1])

for l in range(len(cluster2)):
    awan3pmCentroids2.append(cluster2[l][0])
    kelembaban3pmCentroids2.append(cluster2[l][1])
```

```
plt.scatter(awan3pmCentroids1, kelembaban3pmCentroids1, color ='green', edgecolors='black', linewidth=0.19)
plt.scatter(awan3pmCentroids2, kelembaban3pmCentroids2, color ='yellow', edgecolors='black', linewidth=0.19)
plt.scatter(NewCentroids[0][0], NewCentroids[0][1], color ='red', linewidth = 0.19)
plt.scatter(NewCentroids[1][0], NewCentroids[1][1], color ='red', linewidth = 0.19)

plt.xlabel('awan3pm')
plt.ylabel('kelembaban3pm')

plt.show()
```



Dari hasil visualisasi data clustering data berikut, dapat dilihat bahwa ada peralihan cluster yang cukup jelas antara cluster1 dan cluster2, tetapi ini belum 100% pasti dikarenakan dibutuhkan tahapan lebih lanjut yaitu klasifikasi agar dapat menambah keakuratan apakah benar penyebaran cluster tersebut akurat / tidak.

```
awan3pmCentroids1, kelembaban3pmCentroids1 = zip(*cluster1)
frameCluster1 = pd.DataFrame({'Awan3pm': awan3pmCentroids1, 'Kelembaban3pm': kelembaban3pmCentroids1}, columns=['Awan3pm', 'Kelem
frameCluster1

awan3pmCentroids2, kelembaban3pmCentroids2 = zip(*cluster2)
frameCluster2 = pd.DataFrame({'Awan3pm': awan3pmCentroids2, 'Kelembaban3pm': awan3pmCentroids2}, columns=['Awan3pm', 'Kelembaban3
```

```
frameCluster1['Cluster'] = 'Tidak Bersalju'
frameCluster2['Cluster'] = 'Bersalju'

frameClusters = pd.concat([frameCluster1, frameCluster2], axis=0)
frameClusters


frameClusters
frameClusters.to_csv('Data_After_Clustering.csv')
```

Di akhir saya konversi kan kembali array yang sebelumnya adalah dataframe menjadi balik ke asalnya menjadi dataframe lagi agar dapat dibuat ke dalam file .csv

Kesimpulan

Saya dapat menarik kesimpulan bahwa dengan menggunakan metode mencari antar atribut dengan korelasi tinggi, scaling,elbow dan clustering dapat membuat suatu prediksi akan suatu hal (sesuai dataset). Tetapi prediksi tersebut belum tentu PASTI dikarenakan dibutuhkan tahapan selanjutnya yaitu klasifikasi untuk lebih menambah keakuratan-nya.

Link Youtube :

https://www.youtube.com/watch?v=lWj0u1TMruc&t=166s