Laporan Tugas Besar Pembelajaran Mesin
IF-42-03

Disusun Oleh :

1301184219	Sya Raihan Heggi

S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2021

DAFTAR ISI

PENDAHULUAN 3
1.1 Latar Belakang Masalah 3
1. 2 Tujuan 3
METODE 3
2.1 Data Preprocessing 3
2.2 Clustering 12
DAFTAR PUSTAKA 21

PENDAHULUAN

1.1 Latar Belakang Masalah

Saat ini ketersediaan data sangat berlimpah baik yang dihasilkan dari penggunaan teknologi informasi atau pengumpulan data yang berkaitan dengan semua bidang kehidupan, hal ini menimbulkan kebutuhan untuk mendapatkan informasi apa yang dapat didapatkan dari data yang kita proses ini, sehingga nantinya data ini akan dilakukan eksplorasi, analisis, dan ekstraksi informasi itu sendiri dari data, teknik-teknik yang dapat digunakan untuk pengekstrakan pengetahuan dapat menggunakan bantuan *machine learning* yaitu dengan metode Clustering dan Classification, yang dapat digunakan untuk menemukan kategori atau klasifikasi kemungkinan data tersebut.

Klasterisasi adalah metode untuk melakukan pengkategorisasian objek yang ada didalam data ini menjadi beberapa kluster yang memiliki kemiripan atau karakter yang sama, dan kemudian ada Klasifikasi merupakan metode pemrosesan untuk menemukan sebuah model atau fungsi yang menjelaskan dan mencirikan konsep atau kelas data, untuk kepentingan tertentu. Pada tugas besar ini algoritma yang akan digunakan untuk melakukan Clustering adalah K-Means Clustering dan untuk Classification akan membandingkan antara algoritma Naive Bayes dan Decision Tree Classification, dataset yang digunakan pada tugas besar ini adalah dataset data yang berkaitan dengan prediksi apakah akan turun salju besok hari atau tidak dataset sendiri terdiri dari *(127277,23)* data, yang kemudian akan dibagi menjadi (109095, 24) untuk train set (kelebihan 1 atribut karena ada id yang akan di drop nantinya) dan (18182, 23) untuk tes set. Pemilihan algoritma k means dikarenakan merupakan sebuah metode clustering yang paling sederhana dan umum dan juga karena K-means mempunyai kemampuan mengelompokkan data dalam jumlah yang cukup besar dengan waktu komputasi yang cepat dan efisien. Dan metode Naïve Bayes adalah klasifikasi statistic yang dapat digunakan untuk memprediksi suatu kelas

1. 2 Tujuan

Tujuan dari penelitian ini adalah untuk melakukan clustering dan classification terhadap data Salju, dan kemudian melakukan proses pengolahan data dan analisa mengenai performansi dari proses yang sudah dilakukan.

METODE

2.1 Data Preprocessing

Pada tahapan ini akan dilakukan eksplorasi data, pengenalan terhadap data, dan kemudian melakukan pemrosesan terhadap data yaitu **Data Cleaning, Data Transformation, Data Reduction**, untuk tahapan pertama adalah mengenal dataset yang

sudah diberikan dataset salju ini terdiri dari 2 bagian yaitu training dan test sehingga kita dapat menggunakannya untuk klasifikasi dan clustering, untuk klasifikasi kita bisa membagi training data lagi menjadi train dan validation kemudian set test, sementara untuk clustering hal ini tidak terlalu berpengaruh karena seperti prinsip dan cara kerjanya clustering tidak terkait dengan prediktor namun bisa melabelkan namun untuk kualitasnya nanti bisa kita cek berdasarkan kualitas dari kluster yang dibuat hal ini bisa dihitung dengan **SSE** atau **Silhouette Coefficient,**

2.1.1 Analisis Informasi Dataset

```
In [3]: df_train.head()
```

Out[3]:

	id	Tanggal	KodeLokasi	SuhuMin	SuhuMax	Hujan	Penguapan	SinarMatahari	ArahAnginTerkencang	KecepatanAnginTerkencang	...	Kelembaban9
0	1	01/06/2014	C4	10.4	15.5	4.8	NaN	NaN	WSW	24.0	...	7
1	2	15/07/2014	C10	9.0	17.0	8.0	2.6	7.4	NaN	NaN	...	8
2	3	16/02/2011	C46	18.2	32.0	0.0	NaN	NaN	ESE	44.0	...	6
3	4	08/08/2012	C36	7.3	24.5	0.0	8.4	10.4	SSW	54.0	...	2
4	5	29/10/2016	C7	5.9	20.3	0.0	3.6	12.6	N	37.0	...	5

5 rows × 24 columns

```
In [4]: df_test.head()
```

Out[4]:

	Tanggal	KodeLokasi	SuhuMin	SuhuMax	Hujan	Penguapan	SinarMatahari	ArahAnginTerkencang	KecepatanAnginTerkencang	ArahAngin9am	...
0	04/11/2010	C39	11.0	27.5	0.0	NaN	6.4	WSW	46.0	W	...
1	26/03/2015	C35	10.0	19.9	0.2	NaN	NaN	WNW	56.0	W	...
2	22/03/2016	C18	9.2	27.2	0.0	5.2	10.4	SSW	33.0	NE	...
3	09/12/2011	C31	17.7	27.0	0.0	4.6	6.7	SW	35.0	E	...
4	20/05/2017	C14	2.3	7.9	88.0	NaN	NaN	NW	46.0	W	...

5 rows × 23 columns

Pada dataset ini terdapat 23 atribut dan kurang lebih 127277 data yang sudah dibagi menjadi dua dataset yaitu *Training* dan *Test*, dan untuk lebih jelasnya mengenai jumlah data dapat dilihat menggunakan .shape seperti pada gambar berikut ini.

```
In [5]: df_train.shape
Out[5]: (109095, 24)

In [6]: df_test.shape
Out[6]: (18182, 23)
```

Bila diperhatikan lagi ada perbedaan jumlah atribut dari train dan test namun menurut saya ini tidak berpengaruh karena atribut yang berbeda ini hanya id saja dan bisa di drop dan digunakan ke 23 lainnya yang sama dengan atribut pada dataset, membahas mengenai 23 atribut dari analisa didapati 23 atribut yang ada adalah berikut ini.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18182 entries, 0 to 18181
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Tanggal                   18182 non-null  object
 1   KodeLokasi                18182 non-null  object
 2   SuhuMin                   18017 non-null  float64
 3   SuhuMax                   18017 non-null  float64
 4   Hujan                     17795 non-null  float64
 5   Penguapan                 10326 non-null  float64
 6   SinarMatahari             9464 non-null   float64
 7   ArahAnginTerkencang       16901 non-null  object
 8   KecepatanAnginTerkencang  16908 non-null  float64
 9   ArahAngin9am              16874 non-null  object
 10  ArahAngin3pm              17686 non-null  object
 11  KecepatanAngin9am         17984 non-null  float64
 12  KecepatanAngin3pm         17828 non-null  float64
 13  Kelembaban9am             17852 non-null  float64
 14  Kelembaban3pm             17634 non-null  float64
 15  Tekanan9am                16317 non-null  float64
 16  Tekanan3pm                16329 non-null  float64
 17  Awan9am                   11140 non-null  float64
 18  Awan3pm                   10726 non-null  float64
 19  Suhu9am                   17963 non-null  float64
 20  Suhu3pm                   17740 non-null  float64
 21  BersaljuHariIni           17795 non-null  object
 22  BersaljuBesok             17763 non-null  object
dtypes: float64(16), object(7)
memory usage: 3.2+ MB
```

Dari atribut yang dimiliki bisa dilihat bahwa ada beberapa atribut yang mempunyai tipe object atau data Kategorik/Nominal hingga nanti perlu ada dilakukan transformasi baik itu menggunakan **Label Encode** maupun **One-Hot-Encode,** kemudian akan di cek apakah data ini memiliki *missing value*, *duplicate value*, dan *wrong values* pertama-tama akan dicek apakah dari dataset ii memiliki missing value dan dimanakah missing value itu berada untuk lebih jelasnya ada pada gambar berikut ini.

```
In [9]: df_train.isnull().sum()
```

```
Out[9]: 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
```

```
In [10]: df_test.isnull().sum()
```

```
Out[10]: Tanggal                     0
         KodeLokasi                  0
         SuhuMin                   165
         SuhuMax                   165
         Hujan                     387
         Penguapan                7856
         SinarMatahari            8718
         ArahAnginTerkencang      1281
         KecepatanAnginTerkencang 1274
         ArahAngin9am             1308
         ArahAngin3pm              496
         KecepatanAngin9am         198
         KecepatanAngin3pm         354
         Kelembaban9am             330
         Kelembaban3pm             548
         Tekanan9am               1865
         Tekanan3pm               1853
         Awan9am                  7042
         Awan3pm                  7456
         Suhu9am                   219
         Suhu3pm                   442
         BersaljuHariIni           387
         BersaljuBesok             419
         dtype: int64
```

dilihat dari sebaran missing value ini akan sangat merugikan bagi kita untuk melakukan drop data karena mengurangi jumlah dari dataset yang bisa kita gunakan yang akan berakibat kepada overfitting atau underfitting sehingga diputuskan penanganan dari missing value ini akan dilakukan dengan melakukan pengisian nilai dengan Mean/Modus dari kolom tersebut, kemudian akan dicek apakah terdapat duplicate value pada dataset yang dapat dilihat dengan gambar berikut ini.

Check Duplicate Value

```
In [11]: df_train.duplicated().sum()
Out[11]: 0

In [12]: df_test.duplicated().sum()
Out[12]: 0
```

kemudian akan dicek apakah ada masukan yang wrong values maksud wrong values disini adalah nilai yang berada pada atribut tidak sesuai seperti contoh berikut ini.

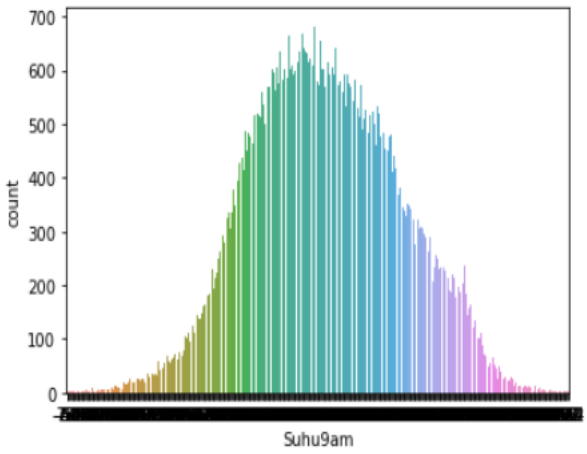
```
df['tumor-size'].value_counts()

30-34    60
25-29    54
20-24    50
15-19    30
Oct-14   28
40-44    22
35-39    19
50-54     8
0-4       8
05-Sep    4
45-49     3
Name: tumor-size, dtype: int64
```

Pada gambar sebelumnya dapat dilihat dari atribut tumor-size seharusnya berisi tentang ukuran dari sebuah tumor namun didalamnya ada nilai yang berisi tanggal hal seperti ini harus ditangani dengan mengubah dengan nilai yang sesuai atau menggantinya dengan nilai mean/modus, namun pada dataset salju yang sudah diberikan bisa dibilang untuk wrong values ini tidak ada sebagai contoh berikut ini.

```
In [20]: df_train['ArahAnginTerkencang'].value_counts()
Out[20]: W      7491
         SE     7078
         N      6955
         S      6931
         E      6902
         SSE    6882
         WSW    6824
         SW     6656
         SSW    6495
         WNW    6202
         ENE    6125
         NW     6087
         ESE    5429
         NE     5342
         NNW    5025
         NNE    4927
         Name: ArahAnginTerkencang, dtype: int64
```

bisa dilihat pada data ini nilainya sesuai karena arah angin ini ditangai dengan mata angin dan parameter yang dimasukan merupakan arah mata angin dari segala penjuru sisi, kemudian yang terakhir dilakukan pada eksplorasi data dilihat persebaran dari data yang ada dan juga tingkat skewnessnya apakah data terdistribusi dengan baik atau tidak.

```
In [38]: sn.countplot(x=df_train['Suhu9am'])
Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x24e62d26f48>
```




```
In [58]: df_train.skew()
Out[58]: id                          0.000000
         SuhuMin                     0.018027
         SuhuMax                     0.220794
         Hujan                      10.059372
         Penguapan                   3.916398
         SinarMatahari              -0.493987
         KecepatanAnginTerkencang    0.875967
         KecepatanAngin9am           0.783795
         KecepatanAngin3pm           0.622092
         Kelembaban9am              -0.481203
         Kelembaban3pm               0.033181
         Tekanan9am                 -0.098483
         Tekanan3pm                 -0.049309
         Awan9am                    -0.230423
         Awan3pm                    -0.227323
         Suhu9am                     0.085140
         Suhu3pm                     0.238056
         dtype: float64
```

Skewness disini menunjukan kecenderungan dari persebaran data bisa jadi lebih condong ke satu sisi atau satu sisi sehingga dapat diketahui bagaimana sebaran datanya.

2.1.2 Data Cleansing

Pada tahapan ini kita akan membersihkan data dari eksplorasi yang sudah kita lakukan sebelumnya dalam hal ini menangani **Outliers, Missing Value, Duplicate Value** pertama-tama yang akan ditangani adalah Missing Value seperti yang sudah dijelaskan karena tingkat missing value yang tinggi dapat merugikan bila kita melakukan drop maka akan dilakukan input nilai dengan mean/modus dari kolom tersebut kurang lebihnya untuk data numerik akan diisi dengan nilai Mean dan untuk Object akan diisi dengan modus metode ini akan dilaksanakan kurang lebih seperti berikut ini.

```
In [60]:
df_train['Tanggal'].fillna(df_train['Tanggal'].mode().iloc[0], inplace=True)
df_train['KodeLokasi'].fillna(df_train['KodeLokasi'].mode().iloc[0], inplace=True)
df_train['ArahAnginTerkencang'].fillna(df_train['ArahAnginTerkencang'].mode().iloc[0], inplace=True)
df_train['ArahAngin9am'].fillna(df_train['ArahAngin9am'].mode().iloc[0], inplace=True)
df_train['ArahAngin3pm'].fillna(df_train['ArahAngin3pm'].mode().iloc[0], inplace=True)
df_train['BersaljuHariIni'].fillna(df_train['BersaljuHariIni'].mode().iloc[0], inplace=True)
df_train['BersaljuBesok'].fillna(df_train['BersaljuBesok'].mode().iloc[0], inplace=True)
```

```
In [61]:
df_train['SuhuMin'].fillna(df_train['SuhuMin'].mean(), inplace=True)
df_train['SuhuMax'].fillna(df_train['SuhuMax'].mean(), inplace=True)
df_train['Hujan'].fillna(df_train['Hujan'].mean(), inplace=True)
df_train['Penguapan'].fillna(df_train['Penguapan'].mean(), inplace=True)
df_train['SinarMatahari'].fillna(df_train['SinarMatahari'].mean(), inplace=True)
df_train['KecepatanAnginTerkencang'].fillna(df_train['KecepatanAnginTerkencang'].mean(), inplace=True)
df_train['KecepatanAngin9am'].fillna(df_train['KecepatanAngin9am'].mean(), inplace=True)
df_train['KecepatanAngin3pm'].fillna(df_train['KecepatanAngin3pm'].mean(), inplace=True)
df_train['Tekanan9am'].fillna(df_train['Tekanan9am'].mean(), inplace=True)
df_train['Tekanan3pm'].fillna(df_train['Tekanan3pm'].mean(), inplace=True)
df_train['Kelembaban9am'].fillna(df_train['Kelembaban9am'].mean(), inplace=True)
df_train['Kelembaban3pm'].fillna(df_train['Kelembaban3pm'].mean(), inplace=True)
df_train['Awan9am'].fillna(df_train['Awan9am'].mean(), inplace=True)
df_train['Awan3pm'].fillna(df_train['Awan3pm'].mean(), inplace=True)
df_train['Suhu9am'].fillna(df_train['Suhu9am'].mean(), inplace=True)
df_train['Suhu3pm'].fillna(df_train['Suhu3pm'].mean(), inplace=True)
```

Kemudian dapat dipastikan kembali apakah missing value ini sudah teratasi atau belum dengan mengecek kembali jumlah dari missing valuenya yang dapat dijelaskan kurang lebih seperti berikut ini.

```
In [64]: df_train.isnull().sum()

Out[64]: 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
```

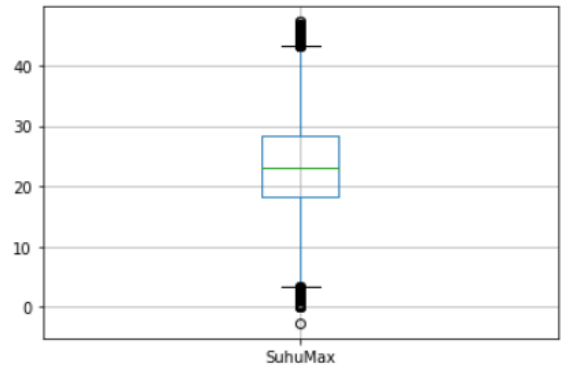
Kemudian lakukan hal ini juga untuk data train, selanjutnya akan dilakukan drop data yang bernilai duplicate dapat dilakukan dengan berikut ini.

Drop Duplicate Values

```
In [66]: df_train.drop_duplicates(keep=False,inplace=True)
         df_test.drop_duplicates(keep=False,inplace=True)

In [67]: df_train.duplicated().sum()
Out[67]: 0

In [68]: df_test.duplicated().sum()
Out[68]: 0
```

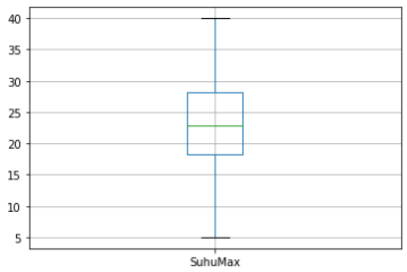
Kemudian pastikan kembali bila nilai duplicate sudah tidak ada pada dataset, selanjutnya akan ditangani masalah outlier untuk hal ini bisa menggunakan boxplot untuk melihat apakah ada outlier pada kolom yang dimiliki seperti berikut ini.

```
In [81]: df_train_final.boxplot(column = ['SuhuMax'])
Out[81]: <matplotlib.axes._subplots.AxesSubplot at 0x24e75f0ae88>
```



Tangani dengan membuat nilai yang termasuk outlier dan hasilnya kurang lebih akan seperti berikut ini.

```
In [82]: df_train_final.drop(df_train_final[((df_train_final['SuhuMax'] > 40) | (df_train_final['SuhuMax'] < 5))].index ,inplace =True
         df_train_final.boxplot(column = ['SuhuMax'])
Out[82]: <matplotlib.axes._subplots.AxesSubplot at 0x24e78281308>
```



Dan tahapan cleansing yang dilakukan kurang lebih hal-hal yang sudah dijelaskan sebelumnya.

2.1.3 Data Transformation

Pada tahapan ini kita akan mengubah data yang bukan numerik menjadi numerik berdasarkan pengamatan yang dilakukan data dapat diubah menjadi numerik dengan menggunakan LabelEncoder karena tidak ada yang memiliki hierarki dan beberapa object sudah terpisah sehingga dapat dilakukan kurang lebih berikut ini.

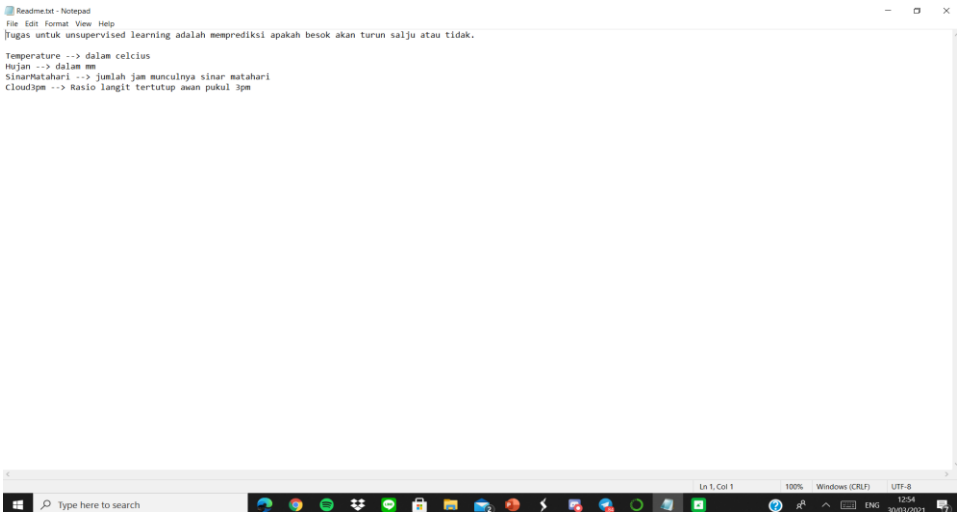
```
In [*]:  from sklearn.preprocessing import LabelEncoder
         from sklearn.preprocessing import MinMaxScaler
```

```
In [*]:  le = LabelEncoder()
         df_train['Tanggal'] = le.fit_transform(df_train['Tanggal'])
         df_train['KodeLokasi'] = le.fit_transform(df_train['KodeLokasi'])
         df_train['ArahAnginTerkencang'] = le.fit_transform(df_train['ArahAnginTerkencang'])
         df_train['ArahAngin9am'] = le.fit_transform(df_train['ArahAngin9am'])
         df_train['ArahAngin3pm'] = le.fit_transform(df_train['ArahAngin3pm'])
         df_train['BersaljuHariIni'] = le.fit_transform(df_train['BersaljuHariIni'])
         df_train['BersaljuBesok'] = le.fit_transform(df_train['BersaljuBesok'])
```

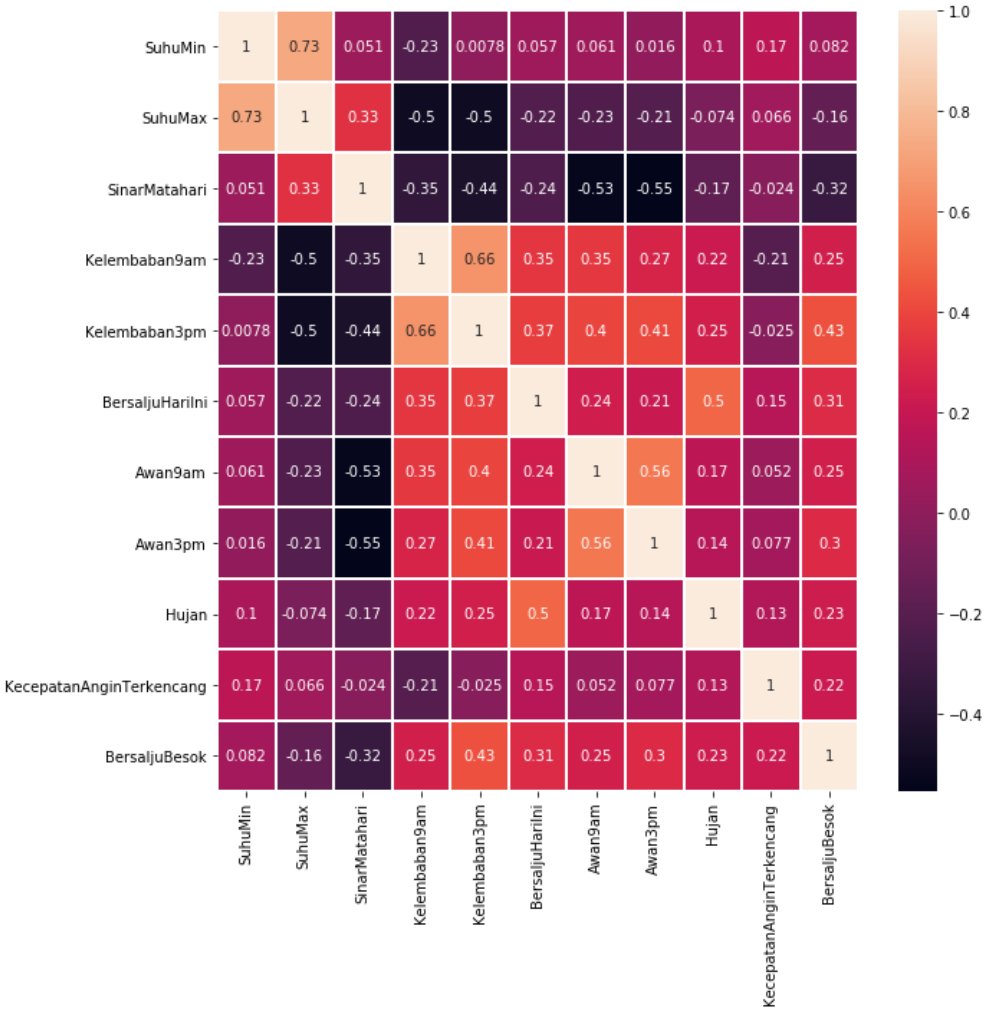
```
In [*]:  df_test['Tanggal'] = le.fit_transform(df_test['Tanggal'])
         df_test['KodeLokasi'] = le.fit_transform(df_test['KodeLokasi'])
         df_test['ArahAnginTerkencang'] = le.fit_transform(df_test['ArahAnginTerkencang'])
         df_test['ArahAngin9am'] = le.fit_transform(df_test['ArahAngin9am'])
         df_test['ArahAngin3pm'] = le.fit_transform(df_test['ArahAngin3pm'])
         df_test['BersaljuHariIni'] = le.fit_transform(df_test['BersaljuHariIni'])
         df_test['BersaljuBesok'] = le.fit_transform(df_test['BersaljuBesok'])
```

2.1.4 Data Reduction

Pada tahapan ini akan dilakukan pemilihan atribut mana saja yang akan digunakan pertimbangan yang dilakukan adalah dengan catatan yang ada pada dataset readme seperti berikut ini.



dan kemudian hasil ini dibandingkan dengan heatmap yang didapati pada dataset yang kurang lebih sudah saya seleksi menjadi seperti berikut ini.



Dan akhirnya diputuskan untuk memilih atau melibatkan Kelembaban3pm dan Awan3pm untuk melakukan clustering klasifikasi nantinya alasan mengapa keduanya terpilih adalah pertama memiliki nilai heatmap yang lumayan tinggi terhadap BersaljuBesok dan juga keduanya memiliki keterkaitan karena waktunya sama dan bila kita merujuk kepada peristiwa yang ada di alam Awan dan Kelembaban merupakan salah satu penyebab terjadinya salju sehingga kedua ini nantinya akan digunakan.

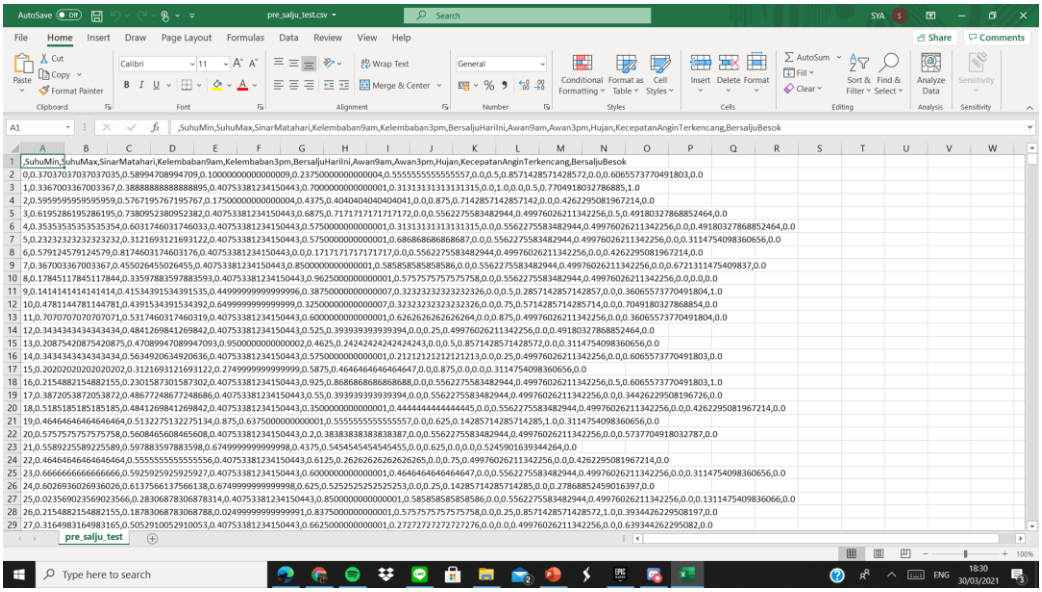
Setelah melewati tahapan tadi data kurang lebih dataset yang akan siap untuk diproses adalah berikut ini.

```
In [92]: databaru.head()
```

Out[92]:

	Kelembaban3pm	Awan3pm
0	0.355556	0.303228
1	0.777778	0.800000
2	0.622222	0.303228
3	0.400000	0.400000
4	0.355556	0.303228

Tetapi data dengan heatmap yang terpilih pada tahapan pertama akan disimpan karena nanti akan diperlukan untuk penggunaan klasifikasi.

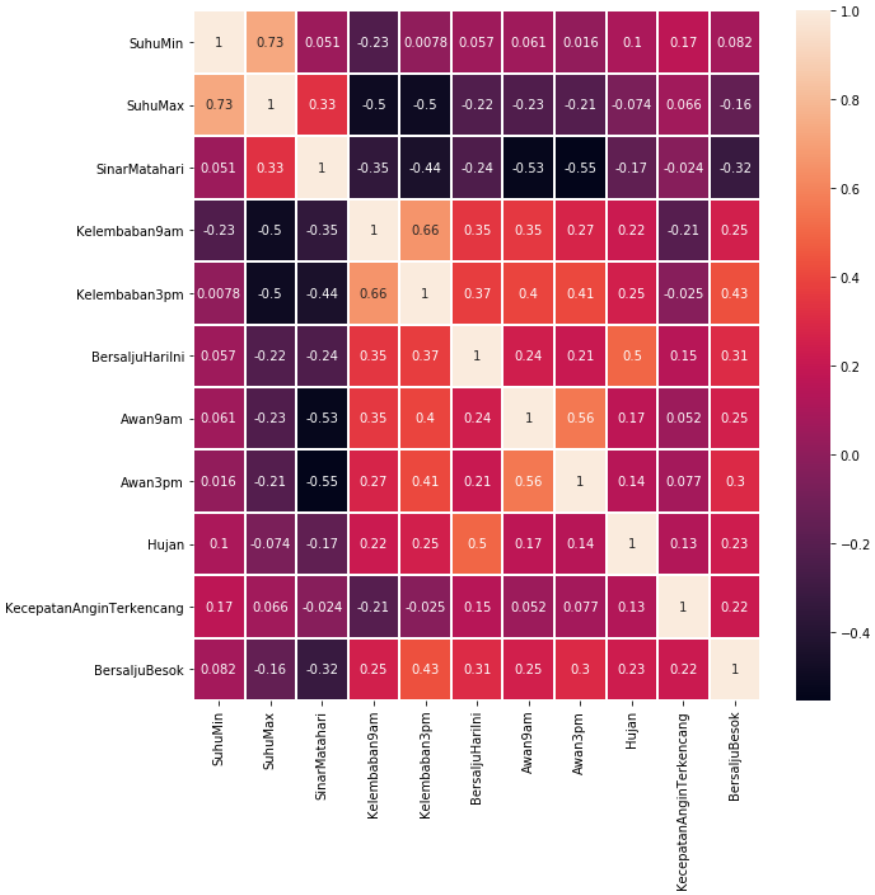


2.2 Clustering

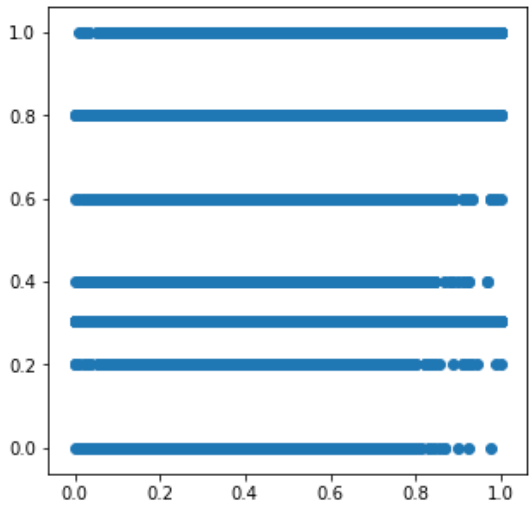
K-Means merupakan sebuah metode clustering yang paling sederhana dan umum dan juga karena k-means mempunyai kemampuan mengelompokkan data dalam jumlah yang cukup besar dengan waktu komputasi yang cepat dan efisien. K-Means merupakan salah satu algoritma clustering dengan metode partisi (partitioning method) yang berbasis titik pusat (centroid-based clustering). Algoritma k-Means dalam penerapannya memerlukan tiga parameter yang seluruhnya ditentukan pengguna yaitu jumlah cluster k, inisialisasi klaster, dan jarak system. Algoritma K-means mendefinisikan centroid atau pusat cluster dari cluster menjadi rata-rata point dari cluster tersebut.Dalam penerapan algoritma K-means, jika diberikan sekumpulan data x= {x1, x2, …,xn} dimana xi = (xi1, xi2, …, xin) adalah sistem dalam ruang real Rn, maka algoritma K-means akan menyusun partisi X dalam sejumlah k cluster (a priori). Setiap cluster memiliki titik tengah (centroid) yang merupakan nilai rata rata (mean) dari data-data dalam cluster tersebut. Tahapan awal, algoritma k-Means adalah memilih secara acak k buah objek sebagai centroid dalam data. Kemudian, jarak antara objek dan centroid dihitung menggunakan rumus perhitungan distance pada penelitian ini akan menggunakan Manhattan dan Euclidean Distance (**Untuk dibandingkan pola yang dihasilkan**). Algoritma K-means secara iteratif meningkatkan variasi nilai dalam dalam tiap tiap cluster dimana obyek selanjutnya ditempatkan dalam kelompok yang terdekat, dihitung dari titik tengah klaster. Titik tengah baru ditentukan bila semua data telah ditempatkan dalam cluster terdekat

2.2.1 Pemilihan Atribut dan Alasannya

Pada percobaan ini saya melakukan pengambilan atribut untuk clustering ini menggunakan panduan dari file readme dan korelasi heatmapnya yang dapat dilihat berikut ini.



Berdasarkan dari heatmap ini sendiri data **Kelembaban3pm** dan **Awan3pm** memiliki korelasi yang cukup tinggi kepada kemungkinan akan Bersalju besok oleh karena itu saya memilih atribut ini untuk dilakukan clustering, dan kedua parameter ini juga berpengaruh untuk terjadinya salju di dunia [1], sehingga layak untuk dicoba dalam parameter ini, untuk gambaran sebaran kedua data ini dapat dilihat seperti berikut ini.



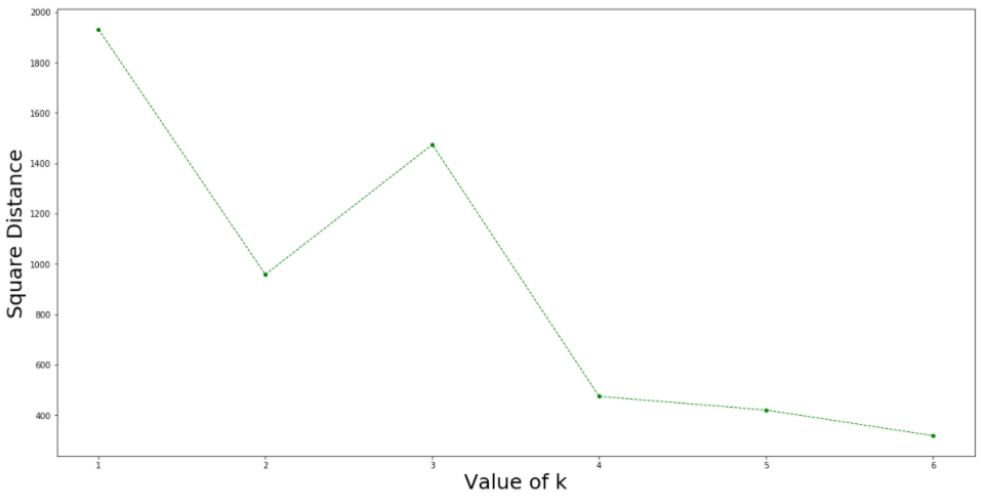
2.2.2 Menentukan Jumlah Cluster

In [112]:
```
costStore_1 = list(range(6))

for k in range(1,7):
        cluster = pd.read_csv('pre_salju_train.csv',usecols=['Kelembaban3pm','Awan3pm'], nrows=20000)

        row_1 = cluster.shape[0] #Mendapatkan Banyak Row
        column_1 = cluster.shape[1] #Mendapatkan Banyak Kolom

        centroids_1 = cluster.loc[np.random.randint(1, row_1+1, k)] #Mengambil Random Centroids
        centroids_1['index'] = list(range(1,k+1)) #Membuat Row Baru yang akan digunakan Index
        centroids_1.set_index('index',inplace = True) #Dijadikan Index dari data
        distance = np.random.rand(row_1)

        iteration = 6
        temp_sse = list(range(iteration))

        for i in range(0, iteration):
            for j in range(0,row_1):
                distance[j] = ((centroids_1 - cluster.loc[j])**2).sum(axis=1).idxmin() #Mengecek jarak data terdekat dengan c
            cluster['centroids'] = distance #simpan nilai distance yang terdekat dengan centroids

            mean_x = list(range(k)) #Mempersiapkan List Mean Centroids Sumbu X
            mean_y = list(range(k)) #Mempersiapkan List Mean Centroids Sumbu Y
            for calculate in range(0,k):
                mean_x[calculate] = cluster[cluster['centroids'] == (calculate+1)]['Kelembaban3pm'].mean() #cek Mean Centroid
                mean_y[calculate] = cluster[cluster['centroids'] == (calculate+1)]['Awan3pm'].mean() #cek Mean Centroids
            centroids_1.replace(list(centroids_1['Kelembaban3pm']),mean_x,inplace = True) #ReplAce Centroids Lama dengan yang
            centroids_1.replace(list(centroids_1['Awan3pm']),mean_y,inplace = True)#Replace Centroids lama dengan yang baru

            sse = list(range(k))
            for p in range(0,k): #Menghitung SSE
                sse[p] = ((cluster[cluster['centroids'] == p+1][['Kelembaban3pm','Awan3pm']] - centroids_1.iloc[p])**2).value
            temp_sse[i] = sum(sse)
            costStore_1[k-1] = temp_sse[i]

        %reset_selective -f centroids_1 #Centroids berikutnya akan dikosongkan untuk looping k berbeda
```

Untuk menentukan jumlah cluster akan digunakan metode elbow, dengan menggunakan SSE terlebih dahulu namun untuk coding yang saya buat dari scratch memakan waktu yang sangat lama untuk menghasilkan kesimpulan dan gambaran grafik yang didapat akan seperti berikut ini.

In [114]:
```
k = list(range(1,7))
plt.figure(figsize=(20,10))
plt.plot(k,costStore_1,'go--',linewidth=1, markersize=4) # Graph is plotted.
plt.xlabel('Value of k',fontsize = 25) # x-axis is labelled.
plt.ylabel('Square Distance',fontsize = 25) # y-axis is labelled.
```

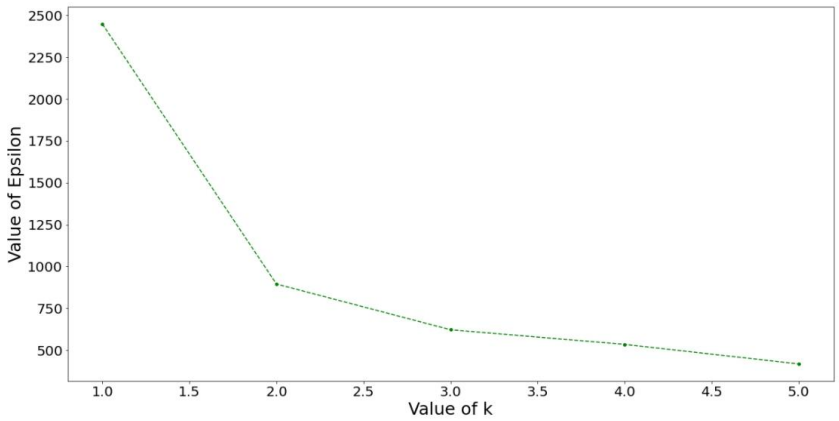
Out[114]: Text(0, 0.5, 'Square Distance')



In [113]:
```
print(costStore_1)
```

```
[1931.1110082055175, 958.7365910811028, 1474.2669732251256, 474.89696100129015, 420.9516034143286, 319.3221515106643]
```

dari gambar ini masih belum bisa kita tentukan, saya mencurigai adanya kurang iterasi untuk pengumpulan nilai SSE menyebabkan hasilnya seperti gambar sebelumnya, dan berikut ini bila dilakukan validasi sebanyak 15 kali maka gambaran grafik elbow dengan algoritma from scratch kurang lebih akan seperti berikut ini (**gambar ini didapatkan dari uji coba yang dilakukan teman dan menggunakan parameter atribut yang sama**).

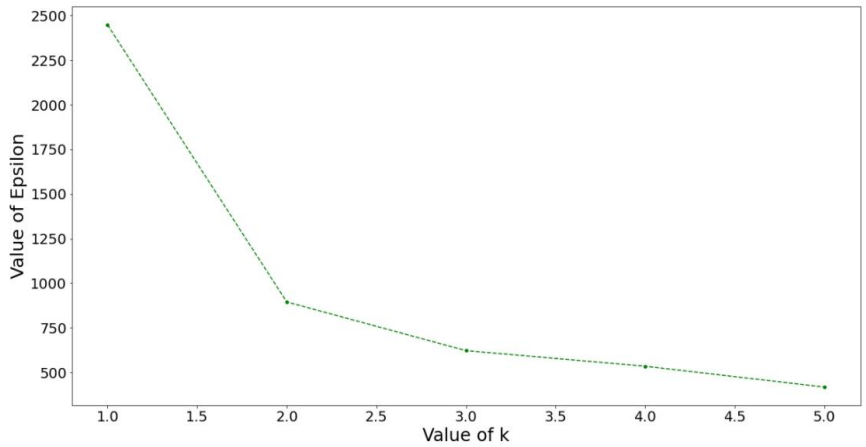


Dikarenakan nilai K yang diuji hanya 5 posisi elbow positionnya kurang terlihat, namun bila diperhatikan kemungkinan ada pada posisi K =3 atau K = 4 , sehingga dapat diputuskan nilai K = 3 atau K = 4 akan diuji pada penelitian ini dan hal ini didukung puka dengan slide pembelajaran di Universitas Telkom Jurusan S1 Informatika **ketika memilih nilai K pilih yang memiliki penurunan SSE dan berada pada posisi *elbow/knee of the curve***[2] dan dapat dilihat nilai SSE semakin mengecil jika K-nya digunakan semakin banyak maka nilai SSEnya akan semakin kecil.

2.2.3 Tahapan Clustering

Pada proses Clustering akan menyelesaikan permasalahan diantaranya menguji mengenai hasil clustering yang dibuat, kemudian membandingkan metode perhitungan distance, dan mengecek antara K=3 dan K=4 mana yang merupakan posisi *elbow* dan apa pengaruh menggunakan cluster melewati posisi *elbow* dengan tahapan yang digunakan kurang lebih akan seperti berikut ini.

- **Langkah 1: Tentukan berapa banyak cluster k dari dataset yang akan dibagi.** Berdasarkan dari Metode Elbow K akan bernilai 3 dan 4, Pada Evaluasi yang saya lakukan tidak jelas karena perlu dilakukan lebih banyak fold validasi baru Metode Elbow terlihat titiknya dan hasilnya akan seperti berikut ini.



Nilai SSE yang didapati setelah 15 fold validasi

```
print(epsilon)
```

```
[2448.384170676462, 895.2641211148489, 739.7736220423453, 461.83815333420637, 433.880524201577]
```

- **Langkah 2: Tetapkan secara acak data k menjadi pusat awal lokasi klaster**.

```
def CentroidsRandom(databaru):
    minKelembaban = min(databaru['Kelembaban3pm'])
    maxKelembaban = max(databaru['Kelembaban3pm'])
    minAwan = min(databaru['Awan3pm'])
    maxAwan = max(databaru['Awan3pm'])
    x_rand = rd.uniform(minKelembaban,maxKelembaban)
    y_rand = rd.uniform(minAwan,maxAwan)
    return x_rand, y_rand
```

```
#Inisialisasi Centroids
LX = []
LY = []
Centroids = []
for i in range(K):
        xrand,yrand = CentroidsRandom(databaru)
        LX.append(xrand)
        LY.append(yrand)
Centroids.append(LX)
Centroids.append(LY)

Centroids = pd.DataFrame(Centroids, columns=['Kelembaban3pm','Awan3pm'])
Centroids.head()
```

- **Langkah 3: Untuk masing-masing data, temukan pusat cluster terdekat. Dengan demikian berarti masing-masing pusat cluster memiliki sebuah subset dari dataset, sehingga mewakili bagian dari dataset. Oleh karena itu, telah terbentuk cluster k: C1, C2, C3, …, Ck .**

```
selisih = 1
iterasi= 0
while(selisih != 0):
    data = databaru
    indexData=1
    for index1,row in Centroids.iterrows(): #Hitung Distance
        distanceList=[]
        for index2,row_1 in data.iterrows():
            distance1 = (row["Kelembaban3pm"]-row_1["Awan3pm"])**2
            distance2 = (row["Kelembaban3pm"]-row_1["Awan3pm"])**2
            distance = np.sqrt(distance1+distance2)
            distanceList.append(distance)
        databaru[indexData]=distanceList #Menyimpan data distance ke dataframe
        indexData += 1

    ClusterList=[]
    for index2,row in databaru.iterrows(): #Looping untuk membuat centroids baru dengan nilai Mean dan perbandingan dengan ni
        min_dist=row[1]
        posisiMinimum=1
        for i in range(K):
            if row[i+1] < min_dist:
                min_dist = row[i+1]
                posisiMinimum=i+1
        ClusterList.append(posisiMinimum)
    databaru["Cluster"]=ClusterList
    tempCentroids = databaru.groupby(["Cluster"]).mean()[["Kelembaban3pm","Awan3pm"]] #Bangkitkan Mean Baru
    if iterasi == 0: #Kondisi looping pertama
        selisih=1
        iterasi = iterasi+1
    else: #kondisi looping kedua setelah ada tempCentroids yang dapat dibandingkan dengan asal
        selisih = (tempCentroids['Kelembaban3pm'] - Centroids['Kelembaban3pm']).sum() + (tempCentroids['Awan3pm'] - Centroids
        print(selisih.sum())
    Centroids = databaru.groupby(["Cluster"]).mean()[["Kelembaban3pm","Awan3pm"]] #Perbaharui Centroids dengan nilai mean
```

```
selisih = 1
iterasi = 0
counterSama = 0
selisihLama = 0

while(selisih != 0 and counterSama <= 4):
    listDistance_1 = []
    listDistance_2 = []
    cluster_1 = []
    cluster_2 = []
    final_cluster = []
    centroidsLama = []
    listDistance_1, listDistance_2 = manhattanDistance(databaru,Centroids,"Kelembaban3pm","Awan3pm")
    cluster_1, cluster_2, final_cluster = Clusterisasi(listDistance_1,listDistance_2)

    databaru["Cluster"]=final_cluster
    #Membuat Centroids Baru
    LX = []
    LY = []
    centroidsLama = Centroids
    centroids1_baru = CentroidsMean(databaru, cluster_1)
    centroids2_baru = CentroidsMean(databaru, cluster_2)
    LX.append(centroids1_baru[0])
    LX.append(centroids2_baru[0])
    LY.append(centroids1_baru[1])
    LY.append(centroids2_baru[1])
    centroidsBaru = []
    centroidsBaru.append(LX)
    centroidsBaru.append(LY)

    Centroids = centroidsBaru
    if iterasi == 0:
        iterasi += 1
        selisih = 1
    else :
        selisih = ((sum(Centroids[0]) - sum(centroidsLama[0]))+(sum(Centroids[1]-sum(centroidsLama[1]))))
        print(selisih)
        if selisih == selisihLama:
            counterSama += 1
        else :
            selisihLama = selisih
```

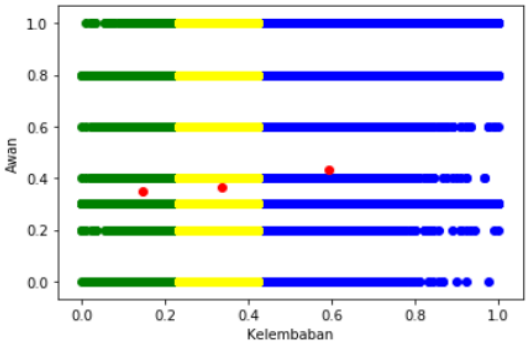
Pada penelitian ini dibuat 2 buah algoritma yang terpikirkan dan akan dilihat hasilnya, kurang lebih sama namun yang membedakan adalah pada Algoritma yang pertama memanfaatkan dataframe, dan pada algoritma kedua mengeluarkan dan mengaksesnya menggunakan list.

- **Langkah 4: Untuk masing-masing cluster k, temukan pusat luasan klaster, dan perbarui lokasi dari masing-masing pusat cluster ke nilai baru dari pusat luasan.**

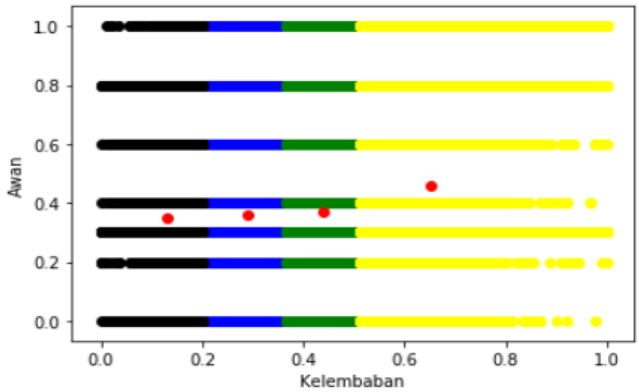
- **Langkah 5: Ulangi langkah ke-3 dan ke-5 hingga data-data pada tiap cluster menjadi terpusat atau selesai**.

2.2.4 Hasil dan Kesimpulan

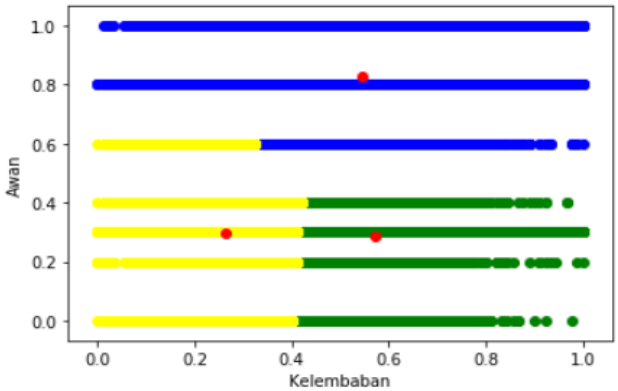
Hasil Clustering K = 3 dengan perhitungan distance Manhattan



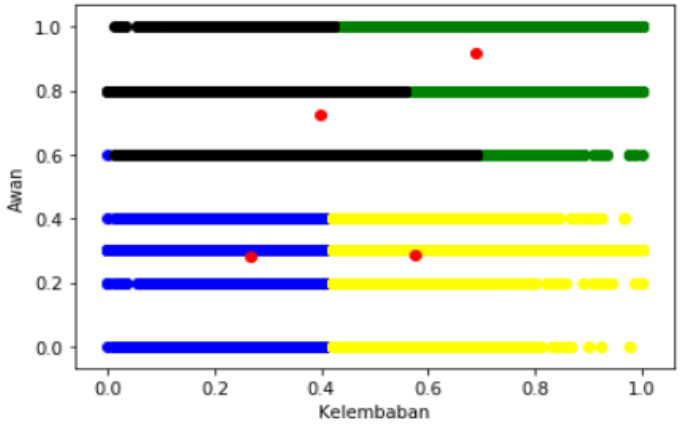
Hasil Clustering K = 4 dengan perhitungan distance Manhattan



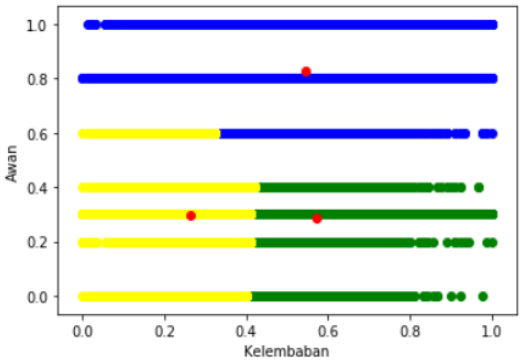
Hasil Clustering K = 3 dengan perhitungan distance Euclidean

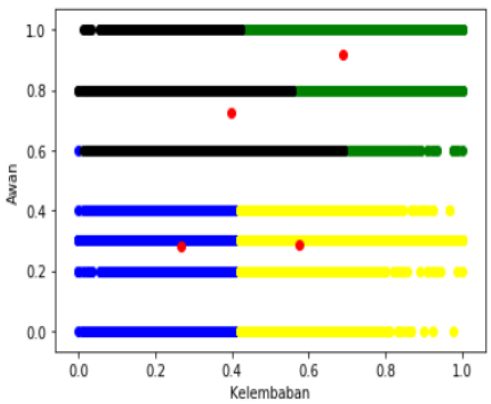


Hasil Clustering K = 4 dengan perhitungan distance Euclidean

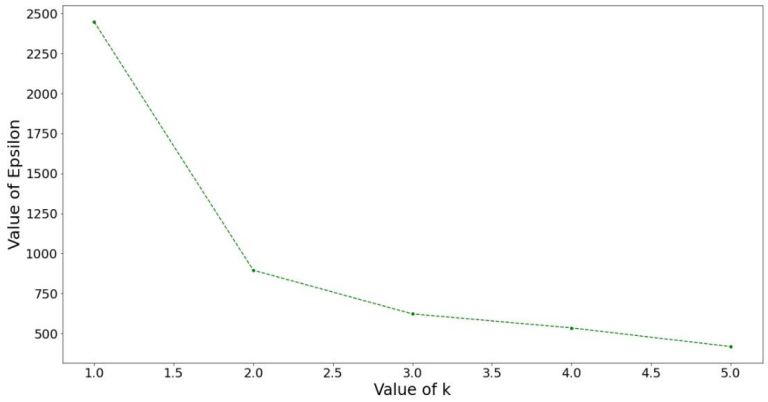


Setelah selesai melakukan itu maka saya melihat bahwa dari penelitian ini, posisi dari centroids diawal nantinya akan mempengaruhi pada hasil akhirnya karena posisinya random dan terus diperbaharui dengan nilai meannya (**jadi pola dari cluster jika dibangkitkan secara random akan berubah-ubah**), kemudian dari hasil validasi yang dilakukan bisa dikatakan semakin banyak cluster maka nilai sum square errornya semakin kecil (**Grafik Elbow Methods**) namun menurut metode elbow layak untuk digunakan saat berada pada posisi Elbow atau *Knee of the Curve* dan nilai tersebut adalah antara nilai K = 3 atau K = 4, dan kalau diperhatikan memang jika semakin banyak pengelompokkan maka nilai dari para cluster semakin mendekat dengan centroid nya seperti yang dilihat pada hasil percobaan.





Dari percobaaan yang saya lakukan pola yang dihasilkan bisa berubah beberapa kali bergantung dari pembangkitan centroids awal, untuk perhitungan distance hasil yang dihasilkan kurang lebih sama namun ada sedikit perbedaan dari posisi dari penggunaan Manhattan dan Euclidean ini sendiri, untuk pengukuran dari performansi dari algoritma distance ini pernah dijelaskan bahwa **Manhattan menghasilkan hasil cluster yang kurang baik jika dibandingkan dari perhitungan Euclidean** [4,5], karena dengan penggunaan Manhattan karena memiliki distorsi pada hasilnya [5], namun bila merujuk kepada berapa K yang memiliki kualitas terbaik berdasarkan evaluasi SSE dengan Elbow Methods K=3 merupakan jumlah cluster yang memiliki kualitas terbaik karena ada penurunan Sum Square Error dengan K=3 dan berada pada posisi Elbow dari kurva yang dihasilkan, kemudian jika membahas mengapa memilih pada posisi elbow karena ketika posisi tersebut terjadi *Diminishing Return* dimana hasilnya bila ditambah jumlah Cluster semisal menjadi 4 (***karena posisi Diminishing pada data ini ada di K = 3***) tetap tidak akan terlalu signifikan perubahannya, berikut gambaran Elbow Method yang didapatkan dari 15 Fold dan pencarian dari K sama dengan 1-5.



Dan setelah pemrosesan Clustering ini data yang sudah di cluster dapat diberikan label, **Ya, Mungkin, Tidak**, Bersalju Besok dan nantinya dapat digunakan untuk tahapan klasifikasi, namun jika melihat database yang disediakan data tersebut sudah memiliki prediktor/class target sehingga seharusnya sudah bisa dilakukan klasifikasi dengan prediktor itu, namun bisa juga digunakan untuk mengukur v score measure, namun tidak digunakan karena beberapa kesulitan ketika mengimplementasikan from scratch.

2.2.5 Saran

Dari percobaan ini mungkin saran yang bisa dilampirkan adalah untuk membuktikan efektifitas dari Manhattan dan Euclidean bisa lebih lagi karena ada perhitungan ditorsi yang tidak dilakukan pada penelitian ini hanya dilakukan Analisis pola dan hasil yang dikeluarkan dari perbedaan perhitungan distance, untuk perhitungan SSE dan Coding dari KMeans mungkin dapat dioptimalkan untuk penelitian berikutnya dikarenakan masih memakan waktu yang cukup lama, dan mungkin dapat ditimbang untuk membandingkan dengan beberapa jenis algoritma clustering lainnya.

DAFTAR PUSTAKA

[1] Proses Terjadinya Salju dan Fakta-Faktanya

[2] Tim Dosen Pembelajaran Mesin Universitas Telkom, *"Unsupervised Learning - Kmeans"* , 2021.

[3] Introduction to Data Mining

[4] Fajriah, R.I., Sutisna, H. and Simpony, B.K., 2019. Perbandingan Distance Space Manhattan Dengan Euclidean Pada K-Means Clustering Dalam Menentukan Promosi. *IJCIT (Indonesian Journal on Computer and Information Technology)*, *4*(1).

[5] Singh, A., Yadav, A. and Rana, A., 2013. K-means with Three different Distance Metrics. *International Journal of Computer Applications*, *67*(10).