Nama : Achmad Alfansyah Nasution

NIM : 1301180006

Kelas : IF-42-04

Laporan Tugas Pembelajaran Mesin

1. Formulasi Masalah

   Diberikan dataset salju yang akan di clustering . Disini saya memakai algoritma **KMeans**.
   K-Means adalah salah satu algoritma clustering yang bersifat Unsupervised Learning. Yaitu berarti inputan dari algoritma ini menerima data tanpa label kelas. Fungsi dari algoritma ini yaitu mengelompokkan data kedalam beberapa cluster.

2. Eksplorasi dan Persiapan Data

   a. Melihat datanya terlebih dahulu dan drop data yang values nya kategorikal, karena clustering membutuhkan data yang bersifat numerik. Karena itu saya drop kolom Tanggal, KodeLokasi, ArahAnginTerkencang, ArahAngin9am, ArahAngin3pm, BersaljuHariIni, BersaljuBesok.
   b. Menangani Missing Values. Terdapat banyak missing values di masing-masing kolomnya, maka untuk data numerical yang tersisa, saya mengisi valuenya dengan nilai mean/rata-rata dari masing-masing kolom. Untuk missing values yang masih terdapat di kolom BersaljuHariIni dan BersaljuBesok saya drop karna jumlah barisnya sedikit(<5%) sehingga tidak terlalu berpengaruh ke modellingnya nanti.
   c. Melihat korelasi dari setiap kolom. Saya memutuskan untuk memilih kolom SuhuMin, SuhuMax, Suhu9am, Suhu3pm. Karena nilai korelasinya cukup

kuat sehingga akan bagus untuk di modelling Kmeans.

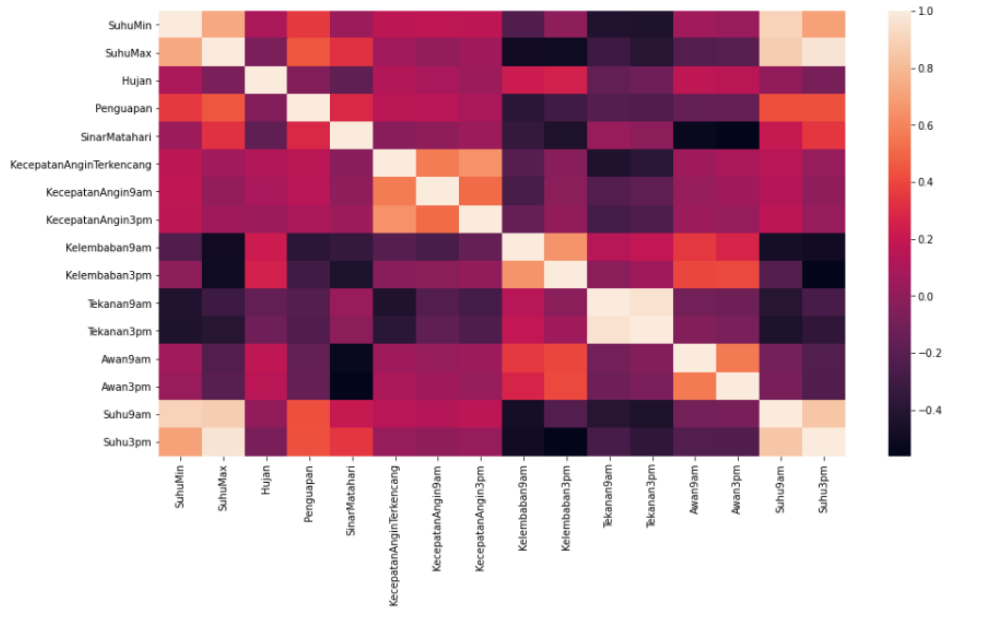
Analisis Data

```
#Melihat kolerasi dari satu variabel ke v
df.corr()
```

	SuhuMin	SuhuMax
SuhuMin	1.000000	0.736220
SuhuMax	0.736220	1.000000
Hujan	0.099211	-0.076857
Penguapan	0.357387	0.446379
SinarMatahari	0.045932	0.325004
KecepatanAnginTerkencang	0.169255	0.062942
KecepatanAngin9am	0.176163	0.016619
KecepatanAngin3pm	0.170409	0.049250
Kelembaban9am	-0.233412	-0.499999
Kelembaban3pm	-0.002248	-0.505629
Tekanan9am	-0.423427	-0.307480
Tekanan3pm	-0.435137	-0.397067
Awan9am	0.066468	-0.220762
Awan3pm	0.030871	-0.202990
Suhu9am	0.897285	0.877804
Suhu3pm	0.703054	0.968470

```
#Visualisasi Korelasinya, Makin terang warnanya, semakin mempunyai korelasi
plt.figure(figsize=(15,8))
sns.heatmap(df.corr())
```

```
<AxesSubplot:>
```



d. Scalling data. Selanjutnya akan dilakukan scalling data yang sudah dipilih tadi. Disini saya menggunakan MinMax Scaler. Tujuan dari scalling ini adalah untuk mempermudah dan mempercepat jalannya algoritma, juga mempermudah dalam perhitungan jarak antar data dan centroid nantinya pada saat modelling.

e. Menggunakan metode Principle Component Analysis. Metode PCA berfungsi untuk mengurangi dimensi dari suatu data, tujuannya agar nantinya pada saat visualisasi hasil cluster akan menjadi lebih bagus daripada tidak memakai metode PCA.

```
In [35]: pca = PCA(n_components=2) #Memakai Principle Component Analysis untuk merub
         principal = pca.fit_transform(df_cluster)
         df_pca = pd.DataFrame(data=principal, columns=['pca1','pca2'])
         df_pca
```

Out[35]:

	pca1	pca2
0	-0.145672	0.047807
1	0.166966	0.002444
2	0.001849	0.148908
3	-0.216837	-0.035422
4	0.581507	-0.080045
...	...	...
18177	0.019778	0.126698
18178	-0.070136	0.067932
18179	0.141939	0.117504
18180	-0.140894	0.042736
18181	0.167230	-0.123168

18182 rows × 2 columns

3. Modelling

K-Means Algorithm

Algoritma KMeans adalah algoritma iterative yang bekerja untuk mempartisi dataset menjadi cluster yang mana setiap point dimiliki pada hanya satu grup. Point tersebut membuat cluster data point yang semirip mungkin dengan point sekitarnya dan sejauh mungkin dari cluster yang lain.

Tahapan dari algoritma ini yaitu :

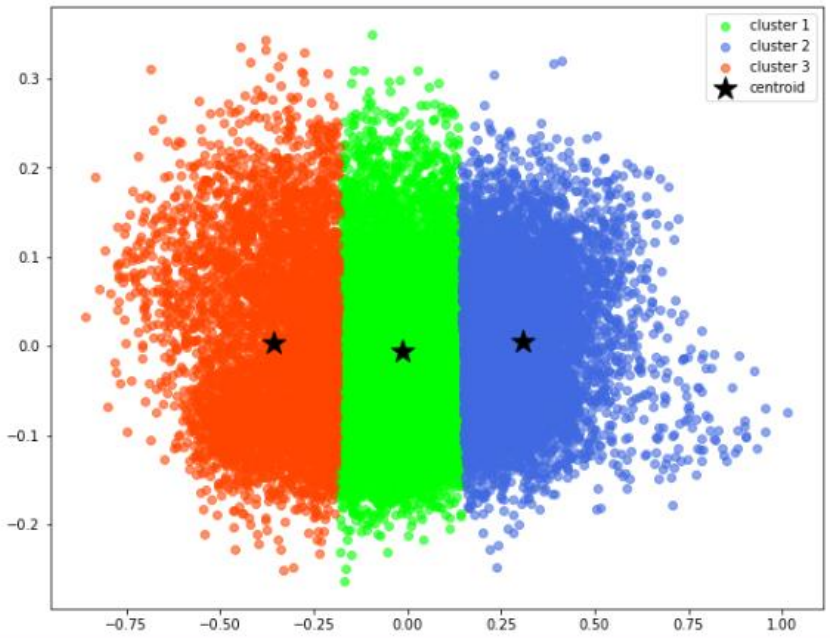
a. Menentukan jumlah K-means, pada kasus ini jumlah cluster sudah ditentukan
b. Alokasikan data ke dalam cluster secara random
c. Hitung centroid/rata-rata dari data yang ada di masing-masing cluster
d. Alokasikan masing-masing data ke centroid/rata-rata terdekat
e. Kembali ke step c, apabila masih ada data yang berpindah cluster atau apabila perubahan nilai centroid.

```
]: class Kmeans:

    def __init__(self, n_clusters, max_iter=100, random_state=123):
        self.n_clusters = n_clusters
        self.max_iter = max_iter
        self.random_state = random_state

    #Inisialisasi Centroids secara Random
    def initializ_centroids(self, X):
        np.random.RandomState(self.random_state)
        random_idx = np.random.permutation(X.shape[0])
        centroids = X[random_idx[:self.n_clusters]]
        return centroids

    #Menghitung rata-rata
    def compute_centroids(self, X, labels):
        centroids = np.zeros((self.n_clusters, X.shape[1]))
        for k in range(self.n_clusters):
            centroids[k, :] = np.mean(X[labels == k, :], axis=0)
        return centroids

    #Menghitung Jarak
    def compute_distance(self, X, centroids):
        distance = np.zeros((X.shape[0], self.n_clusters))
        for k in range(self.n_clusters):
            row_norm = norm(X - centroids[k, :], axis=1)
            distance[:, k] = np.square(row_norm)
        return distance

    def find_closest_cluster(self, distance):
        return np.argmin(distance, axis=1)

    #Menghitung Sum of Square
    def compute_sse(self, X, labels, centroids):
        distance = np.zeros(X.shape[0])
        for k in range(self.n_clusters):
            distance[labels == k] = norm(X[labels == k] - centroids[k], axis=1)
        return np.sum(np.square(distance))

    #Fit Data
    def fit(self, X):
        self.centroids = self.initializ_centroids(X)
        for i in range(self.max_iter):
            old_centroids = self.centroids
            distance = self.compute_distance(X, old_centroids)
            self.labels = self.find_closest_cluster(distance)
            self.centroids = self.compute_centroids(X, self.labels)
            if np.all(old_centroids == self.centroids):
                break
        return self.compute_sse(X, self.labels, self.centroids)
```

Visualisasi

In [38]:
```
#Plotiing data yang sudah di clustering untuk di visualisasi

fig, ax = plt.subplots(figsize=(10, 8))
plt.scatter(z[km.labels == 0, 0], z[km.labels == 0, 1],
            c='lime', label='cluster 1', alpha=0.6)
plt.scatter(z[km.labels == 1, 0], z[km.labels == 1, 1],
            c='royalblue', label='cluster 2', alpha=0.6)
plt.scatter(z[km.labels == 2, 0], z[km.labels == 2, 1],
            c='orangered', label='cluster 3', alpha=0.6)
plt.scatter(centroids[:, 0], centroids[:, 1], marker='*', s=300,
            c='black', label='centroid')
plt.legend()
```

Out[38]:
```
<matplotlib.legend.Legend at 0x121e9224340>
```




```
    def predict(self, X):
        distance = self.compute_distance(X, old_centroids)
        return self.find_closest_cluster(distance)
```

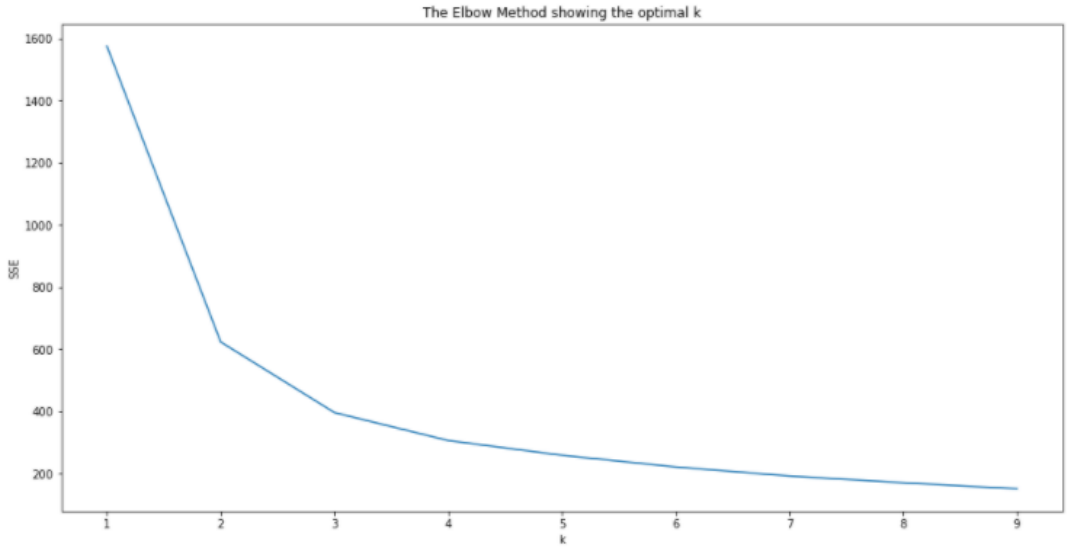
In [37]:
```
z = df_pca.values
K = 3
km = Kmeans(n_clusters=K, max_iter=100) #Modelling data dengan KMeans
km.fit(z)
centroids = km.centroids
```

4. Evaluasi

Evaluasi dari model ini menggunakan Elbow Method. Di tahap ini akan diketahui berapa jumlah cluster yang terbaik dengan mendapatkan perbandingan jumlah cluster dengan cara menghitung SSE (Sum of Suare Error) dari masing-masing nilai cluster.

In [39]:
```
sse = []
K = range(1,10)
for k in K:
    kmeanModel = Kmeans(n_clusters=k)
    sse.append(kmeanModel.fit(z))
```

In [21]:
```
plt.figure(figsize=(16,8))
plt.plot(K, sse)
plt.xlabel('k')
plt.ylabel('SSE')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



In [ ]:

5. Eksperimen

   Sudah dilakukan Eksperimen sebelumnya dengan tidak menggunakan metode PCA, hasil dari cluster kurang optimal

6. Kesimpulan

   Berdasarkan pengolahan data yang dilakukan menggunakan metode clustering KMeans dan optimasi metode Elbow dengan mengetahui nilai SSE(Sum of Square Error) dihasilkan 3 kelompok cluster berdasarkan SuhuMin, SuhuMax, Suhu9am, Suhu3pm.