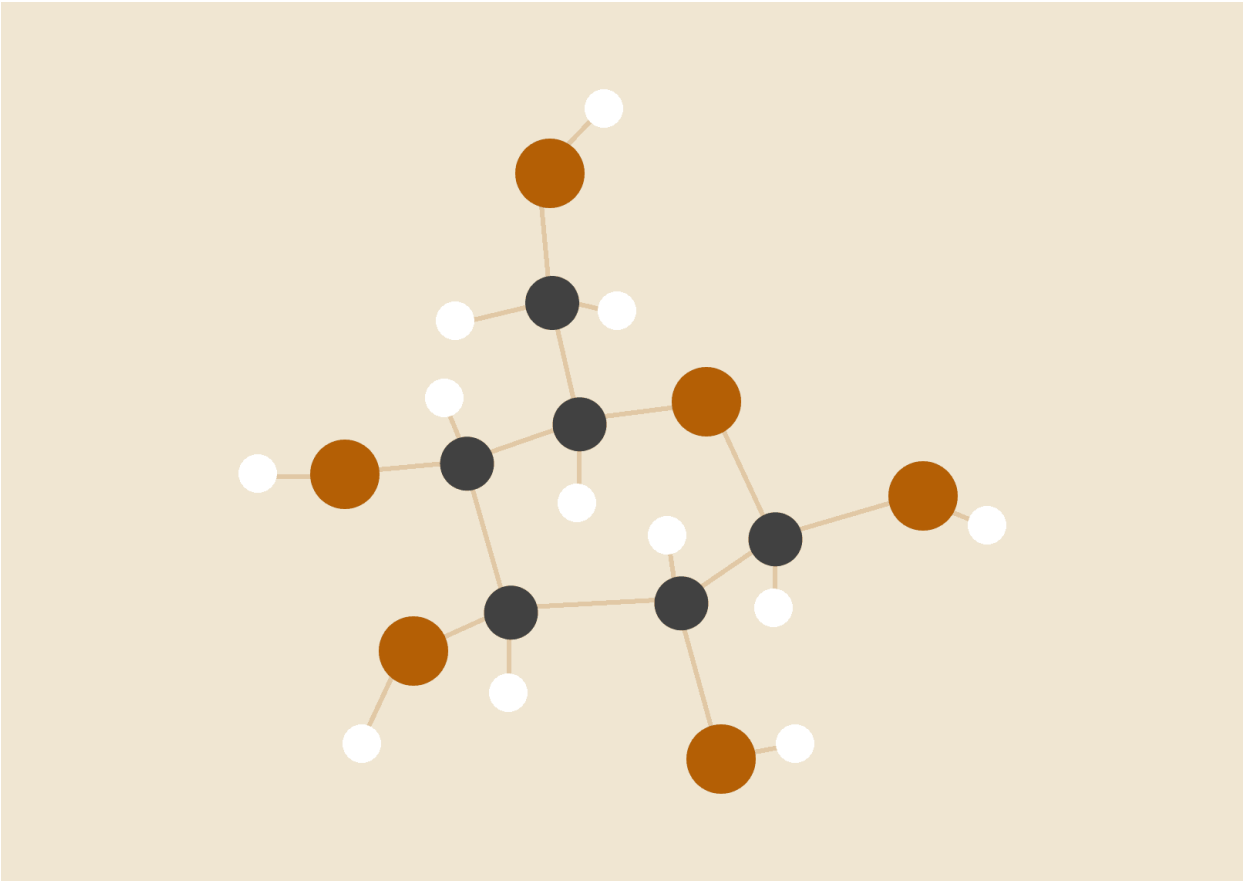
Clustering with K-Means

Lorem ipsum dolor sit amet, consectetuer adipiscing elit



Your Name

09.04.20XX

8TH GRADE SCIENCE

1. Pengantar Analisis Clustering

Dalam bidang ilmu Kecerdasan Buatan terdapat salah satu cabang ilmu *data mining*. Disiplin ilmu ini merupakan irisan dari disiplin ilmu Kecerdasan buatan dan Statistik. *Data mining* sering juga disebut sebagai ilmu tentang data, hal ini dikarenakan data menjadi objek utama dari ilmu ini. Beberapa sumber mempunyai definisi yang berbeda tentang *Data mining*, tetapi semuanya merujuk pada satu tujuan dari *Data mining* yaitu menemukan pola / pengetahuan dari sekumpulan data. Untuk mendapatkan pengetahuan maka dalam *Data mining* dibagi menjadi beberapa Analisis: *Classification / Prediction*, *Clustering* dan *Association.*

Pada *data mining, clustering* merupakan suatu alat yang digunakan untuk mendapatkan informasi dan pengetahuan mengenai distribusi data serta mengamati karakteristik setiap *cluster* untuk dianalisis lebih lanjut. *Clustering* dapat menjadi langkah awal *pre-processing* pada metode lain seperti karakterisasi, pemilihan *subset* atribut dan juga klasifikasi yang kemudian akan beroperasi pada *cluster* yang terdeteksi.

Clustering merupakan sekumpulan objek data yang memiliki kemiripan satu dengan yang lain di dalam *cluster* yang sama dan tidak mirip dengan objek di dalam *cluster* yang lain. Berdasarkan hal tersebut, *clustering* juga dapat disebut sebagai *automatic classification* atau klasifikasi otomatis (Han 2012).

Clustering juga dapat disebut sebagai *data segmentation* di beberapa penerapan lainnya karena *clustering* membagi data besar menjadi kelompok-kelompok kecil. Terdapat berbagai macam metode dalam *clustering,* antara lain:

1. Metode *clustering* berbasis hierarki
2. Metode *clustering* berbasis partisi
3. Metode *clustering* berbasis *density*
4. Metode *clustering* berbasis model
5. Metode *clustering* berbasis *grid-based*

Analisis *Clustering* atau jika dalam Bahasa Indonesia kita sebut sebagai analisis pengelompokkan, merupakan sebuah analisis yang digunakan untuk mengelompokan data yang tidak mempunyai target / label (*unsupervised*). Proses *Clustering* merupakan proses pengelompokkan data berdasarkan kesamaan nilai fitur / atribut. Selain mendapatkan hasil pengelompokkan data, dengan menerapkan analisis *Clustering* ini akan didapatkan titik tengah dari data (*centroid*) dan *cluster* dengan jumlah anggota terbanyak.

Beberapa contoh penerapan dari analisis *Clustering* sebagai berikut:

a. Pada Pengolah Citra Digital terdapat teknik segmentasi. Teknik ini merupakan suatu cara untuk memisahkan objek dengan latar belakangnya. Pemisahan tersebut menggunakan pengelompokan nilai berdasarkan pixel citra. Banyak penelitian yang menggunakan metode-metode *Data Mining* untuk segmentasi citra digital.
b. Untuk menempatkan unit pemadam api maka dibutuhkan tempat yang paling strategis agar dapat mencakup banyak area. Tempat strategis yang dimaksud merupakan titik tengah dari sebuah area dari banyak data. Misalkan dalam kasus kebakaran hutan, dengan mendapatkan data titik api maka kita dapat menganalisis tempat unit pemadam api yang optimal.
c. Hampir sama dengan penentuan titik api, dengan Analisis *Clustering* ini kita dapat menempatkan titik dapur umum atau rumah sakit umum ketika ada suatu kejadian luar biasa. Misalkan terdapat acara pramuka (jamboree) se-Jawa Timur. Pada acara tersebut akan diadakan kegiatan *camping* dan jumlah tenda yang disediakan oleh panitia yakni sebanyak 30 tenda, dengan letak yang tersebar secara acak. Dengan kondisi tenda seperti itu, jumlah dapur umum yang tersedia ada 3 dapur umum. Maka kita perlu menganalisis untuk mengetahui dimana seharusnya panitia mendirikan tenda untuk dapur umum. Analisis *Clustering* dapat dijadikan solusi untuk mendapatkan letak tersebut. Dengan mengelompokkan posisi semua tenda, kita akan mendapatkan posisi titik tengah dari setiap *cluster*.
d. Analisis *clustering* juga dapat digunakan untuk menentukan tren pasar/marketing. Permisalkan contoh sebuah provider kartu meluncurkan produk kartu ABC. Dalam beberapa bulan kartu ABC telah terjual sebanyak 1000 kartu perdana. Untuk mengetahui peminat kartu ABC maka dengan Analisis *clustering* kita dapat mengelompokkan customer kartu ABC dari identitasnya, bisa dari informasi yang dimasukkan waktu registrasi kartu atau dari area pembelian. Dari fitur yang dipilih akhirnya dapat diketahui pengelompokkan data customer, dan dapat mengetahui kelompok customer seperti apa yang mempunyai anggota (peminat) paling banyak.
e. Hampir sama dengan point sebelumnya (d), maka dengan Analisis Custering kita juga dapat mengetahui peminat dari sebuah sekolah, bimbingan belajar, dan kampus. Sehingga dapat menentukan peminatan paling banyak terdapat pada cluster apa. Diharapakan dengan mengetahui peminatanya maka arah kebijakan pemasaran bisa lebih terarah.

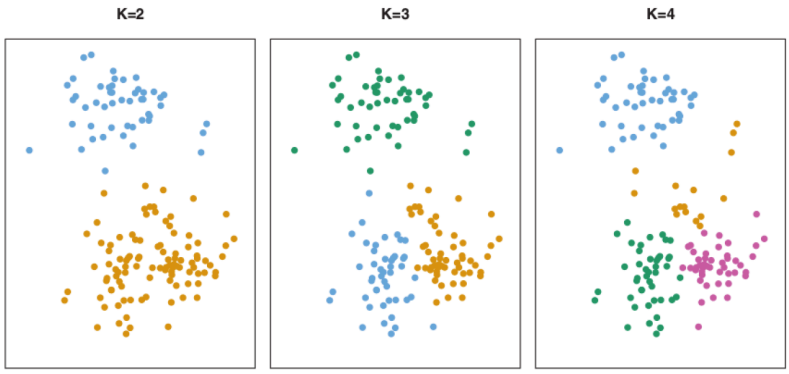
Dalam setiap algoritma atau metode pada ilmu *Data mining* atau ilmu Pendektan (heuristic) kita dapat mengevaluasi kinerja dari metode tersebut, termasuk juga metode-metode yang berada pada Analisis *clustering*. Metode dalam Analisis *Clustering* dievaluasi dengan dua teknik/pengukuran: internal

dan eksternal. Pengukuran internal dilakukan dengan cara menghitung nilai-nilai hasil output dari proses *clustering* tersebut. Pengukuran internal yang paling sering digunakan *Sum Square Error* (SSE), *Mean Square Error* (MSE). Mean *Absolute Percentage Error* (MAPE), *Mean Absolute Deviation* (MAD). Sedangkan untuk pengukuran eksternal nilai yang digunakan untuk pengukuran adalah nilai yang berasal dari luar proses *clustering*. Pengukuran yang sering digunakan dalam pengukuran eksternal adalah pengukuran tingkat akurasi. Tingkat akurasi ini hanya dapat dihitung jika *dataset* yang digunakan adalah *dataset* supervised atau *dataset* dengan label/target.

Selain pengukuran dari internal dan external dalam Analisis *Clustering* terdapat juga pengukuran untuk mendapatkan nilai cluster teroptimal. Untuk menghitung nilai cluster yang paling optimal digunakan dengan Teknik: Silhouette dan Elbow.

K-means

K-Means *Clustering* merupakan salah satu metode sederhana untuk mempartisi sekumpulan data menjadi K *cluster* yang berbeda dan tidak tumpang tindih. Langkah awal yang perlu dilakukan pada metode ini adalah menentukan jumlah *cluster* K yang diinginkan, seperti contoh yang ditunjukkan pada Gambar 3.1.



Gambar 3.1 Contoh Jumlah Cluster K (James et al. 2013)

Sebagai ilustrasi, terdapat sejumlah data seperti ditunjukkan pada Gambar 3.2 berikut.

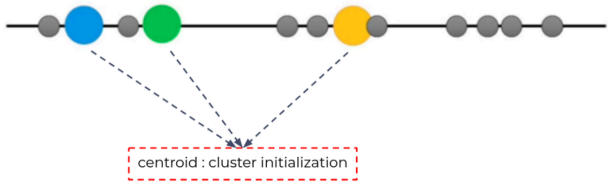
Gambar 3.2 Ilustrasi Data Awal (James et al. 2013)

Langkah-langkah yang perlu dilakukan dalam proses K-Means yakni sebagai berikut:

1. Menentukan nilai *k*, dimana nilai *k* merepresentasikan jumlah *cluster*. Pada studi kasus ini, kami menentukan nilai *k* sebesar 3 yang menunjukkan bahwa terdapat 3 *cluster* pada data tersebut.

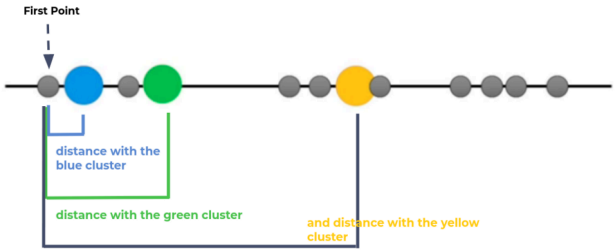
Terdapat berbagai macam cara dalam penentuan nilai k, pada pembahasan kali ini penulis sudah menentukan di awal.

2. Menentukan 3 titik secara acak sebagai *centroid* atau pusat *cluster* untuk inisialisasi *cluster* awal seperti ditunjukkan pada Gambar 3.3. Penentuan 3 titik tersebut berdasarkan dengan nilai k yang telah ditentukan sebelumnya.



Gambar 3.3 Penentuan Cluster Awal (James et al. 2013)

3. Menghitung jarak antara setiap titik dengan *centroid*. Sebagai contoh, penulis menghitung titik pertama dengan ketiga *centroid* yang ada seperti ditunjukkan pada Gambar 3.4.



Gambar 3.4 Perhitungan Jarak Titik dengan Centroid (James et al. 2013)

Rumus yang digunakan penulis untuk menghitung jarak setiap titik dengan *centroid* yakni *Euclidean Distance* sebagai berikut:

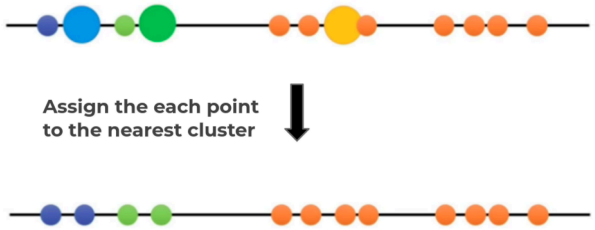
$$d_{(a,b)} = \sqrt{\sum_{i=1}^{n} \left(a_i - b_i\right)^2}$$

Dimana $d_{(a,b)}$ merupakan jarak antara objek a dan b, n merupakan dimensi dari data, a_i merupakan koordinat dari objek a pada dimensi data n dan b_i merupakan koordinat dari objek b pada dimensi data n.

4. Menetapkan setiap titik ke *cluster* dengan jarak terdekat. Sebagai contoh, titik pertama masuk ke dalam *Cluster* pertama karena memiliki jarak paling dekat seperti ditunjukkan pada Gambar 3.5.

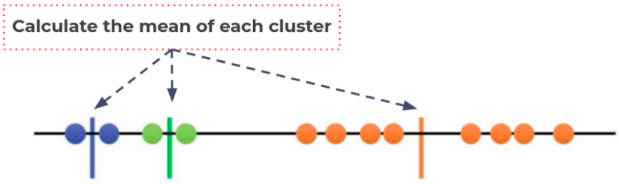
Gambar 3.5 Penetapan Satu Titik pada Cluster (James et al. 2013)

Melakukan hal yang sama untuk semua titik, sehingga tepat setiap titik memiliki *cluster* seperti ditunjukkan Gambar 3.6.



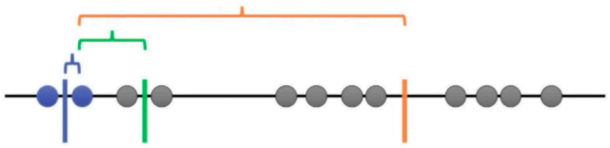
Gambar 3.6 Penetapan Semua Titik pada Cluster (James et al. 2013)

5. Menghitung rata-rata dari setiap *cluster* untuk menjadi *centroid* baru. Sehingga *centroid* awal akan berubah mengikut nilai rata-rata *cluster* tersebut seperti ditunjukkan Gambar 3.7.



Gambar 3.7 Menghitung Nilai Rata-rata Cluster (James et al. 2013)

6. Mengulangi langkah ketiga hingga langkah kelima untuk titik tengah yang baru dari setiap *cluster* seperti ditunjukkan Gambar 3.8.



Gambar 3.8 Pencarian Centroid Baru Tiap Cluster (James et al. 2013)

Proses dalam K-Means ini akan terhenti apabila telah memenuhi kondisi berhenti seperti:

a. Terjadi konvergensi dini (tidak mendapatkan nilai yang lebih baik pada iterasi berikutnya).

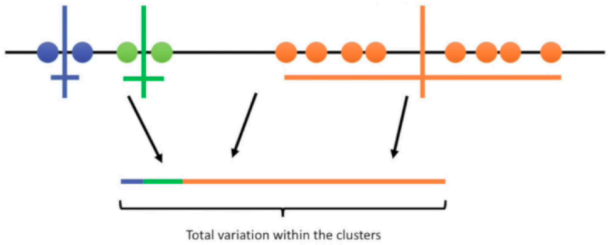
b. Telah mencapai iterasi maksimal yang telah ditentukan sebelumnya.

c. Jika selama proses *clustering* tidak merubah *cluster* data awal, maka proses dapat diakhiri.

Gambar 3.9 Hasil dari Proses Clustering (James et al. 2013)

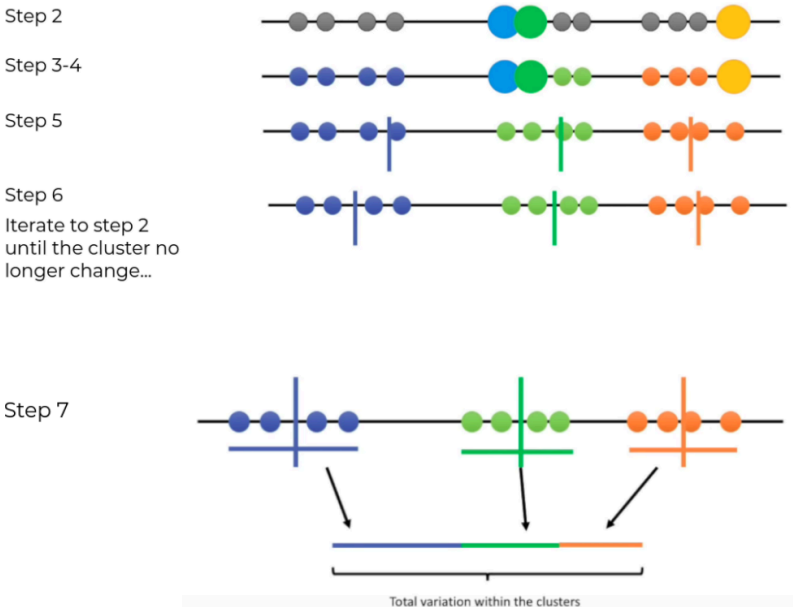
K-Means memiliki tujuan akhir untuk memilih *centroid* yang baru dengan meminimalkan inersia atau kriteria jumlah kuadrat dalam *cluster.* Sehingga proses dalam K-Means dilanjutkan.

7. Menghitung varians dari masing-masing *cluster*. Pada K-Means, tidak dapat terlihat *clustering* terbaik sehingga diperlukan pelacakan pada setiap *cluster* dengan melakukan semua prosesnya dari titik awal yang berbeda.



Gambar 3.10 Perhitungan Varians Tiap Cluster (James et al. 2013)

8. Mengulangi langkah kedua hingga ketujuh sehingga mencapai jumlah varians terendah. Sebagai contoh, penulis mencoba memasukkan dua *centroid* acak yang berbeda ke dalam data.

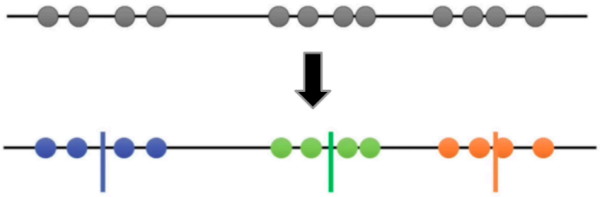


Gambar 3.11 Total Varians Tiap Cluster (James et al. 2013)

9. Langkah-langkah tersebut terus dilakukan hingga mencapai jumlah varians terendah, kemudian memilih *cluster* sebagai hasil akhir.

Gambar 3.12 Pemilihan Cluster (James et al. 2013)

10. Sehingga hasil akhir dari proses *clustering* dengan menggunakan K-Means seperti ditunjukkan Gambar 3.13.



Gambar 3.13 Hasil Akhir dari Proses Clustering (James et al. 2013)

Contoh studi kasus yang dapat diselesaikan dengan metode K-means yakni *clustering* penyakit kelamin. Pada data penyakit kelamin ini terdapat sebanyak 8 faktor resiko, 28 gejala penyakit dan 16 *class* jenis penyakit kelamin.

Implementasi kode program dari metode K-means untuk studi kasus penyakit kelamin yakni sebagai berikut:

Code 3.1 class KMeans

```
1.  package penyakitkelaminkmeans;
2.
3.  import java.io.BufferedReader;
4.  import java.io.FileNotFoundException;
5.  import java.io.FileReader;
6.  import java.io.IOException;
7.  import java.text.DecimalFormat;
8.  import java.util.Random;
9.
10. public class KMeans {
11.
12.     DecimalFormat df = new DecimalFormat("#.####");
13.     int penyakit = 17;
14.     int variabel = 31;
15.     int dataTraining = 109;
16.     double arrDataTraining [][] = new double[dataTraining][variabel];
17.     double arrCenteroid [][] =  new double[penyakit][variabel];
18.     double arrNewCenteroid [][] =  new double[penyakit][variabel];
19.     double arrJarakCenteroid [][] =  new double[dataTraining][penyakit+1];
20.
21.
22.
23.     public  void readData (){
24.         String nama_file = "C:\\Users\\agung\\Documents\\NetBeansProjects\\dataPenyakitKelamin.txt";
25.         int counter=0;
26.         try {
27.             FileReader fr = new FileReader(nama_file);
28.             BufferedReader br = new BufferedReader(fr);
29.
```

```
30.             String text;
31.             String[] nilai = null;
32.             while ((text = br.readLine()) != null && counter<dataTraining){
33.                 nilai = text.split(" ");
34.                 for (int i = 0; i < variabel; i++){
35.                     arrDataTraining[counter][i]= Double.parseDouble(nilai[i]);
36. //                    System.out.print(popSize[counter][i]+" ");
37.                 }
38.                 counter++;
39. //                System.out.println("");
40.             }
41.         }
42.         catch (FileNotFoundException fnfe) {
43.             fnfe.getMessage();
44.         }
45.         catch (IOException ioe) {
46.             ioe.getMessage();
47.         }
48.     }
49.
50.     public void randomCenteroid(){
51.
52. //        for (int i=0;i<penyakit;i++){
53.                 //isi centeroid
54.                 for (int j = 0; j < arrCenteroid[0].length; j++) {
55.                     arrCenteroid[0][j]=arrDataTraining[0][j];
56.                     arrCenteroid[1][j]=arrDataTraining[3][j];
57.                     arrCenteroid[2][j]=arrDataTraining[11][j];
58.                     arrCenteroid[3][j]=arrDataTraining[15][j];
59.                     arrCenteroid[4][j]=arrDataTraining[21][j];
60.                     arrCenteroid[5][j]=arrDataTraining[24][j];
61.                     arrCenteroid[6][j]=arrDataTraining[28][j];
62.                     arrCenteroid[7][j]=arrDataTraining[39][j];
63.                     arrCenteroid[8][j]=arrDataTraining[51][j];
64.                     arrCenteroid[9][j]=arrDataTraining[54][j];
65.                     arrCenteroid[10][j]=arrDataTraining[65][j];
66.                     arrCenteroid[11][j]=arrDataTraining[77][j];
67.                     arrCenteroid[12][j]=arrDataTraining[79][j];
68.                     arrCenteroid[13][j]=arrDataTraining[98][j];
69.                     arrCenteroid[14][j]=arrDataTraining[102][j];
70.                     arrCenteroid[15][j]=arrDataTraining[105][j];
71.                     arrCenteroid[16][j]=arrDataTraining[108][j];
72.                 }
73. //        }
74.     }
75.
76.     public void hitungJarakCenteroid(){
77.         int counter = 0;
78.         double sum = 0;
79.         while (counter<dataTraining){
80.             for (int i = 0; i < arrCenteroid.length; i++) {
81.                 //cek jarak dengan centeroid
82.                 for (int j = 0; j < variabel; j++) {
83.                     sum+=Math.pow(((arrDataTraining[counter][j]-arrCenteroid[i][j])), 2);
84.                 }
85.                 arrJarakCenteroid[counter][i]=Math.sqrt(sum);
86. //                System.out.print(df.format(arrJarakCenteroid[counter][i])+"\t");
87.                 sum=0;
88.             }
89.             counter++;
90. //            System.out.println("");
91.         }
92.
93.     }
94.
95.     public void updateCenteroid(){
```

```
96.         double min=100000;
97.         int index=0;
98.         for (int i = 0; i < arrJarakCenteroid.length; i++) {
99.             for (int j = 0; j < arrJarakCenteroid[0].length-1; j++) {
100.                if (arrJarakCenteroid[i][j]<min) {
101.                    min=arrJarakCenteroid[i][j];
102.                    index=j;
103.                }
104.            }
105.            arrJarakCenteroid[i][penyakit]=index;
106.            min=100000;
107.        }
108.
109.        //new centerooid
110.        int indexNew=0;
111.        int counter1=1;
112.        int counter2=1;
113.        int counter3=1;
114.        int counter4=1;
115.        int counter5=1;
116.        int counter6=1;
117.        int counter7=1;
118.        int counter8=1;
119.        int counter9=1;
120.        int counter10=1;
121.        int counter11=1;
122.        int counter12=1;
123.        int counter13=1;
124.        int counter14=1;
125.        int counter15=1;
126.        int counter16=1;
127.        int counter17=1;
128.
129.        for (int i = 0; i < arrDataTraining.length; i++) {
130.            switch((int)arrJarakCenteroid[i][penyakit]){
131.                case 0:
132.                    indexNew=0;
133.                    for (int j = 0; j < arrNewCenteroid[0].length; j++) {
134.                        arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter1;
135.                    }
136.                    counter1++;
137.                    break;
138.                case 1:
139.                    indexNew=1;
140.                    for (int j = 0; j < arrNewCenteroid[0].length; j++) {
141.                        arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter2;
142.                    }
143.                    counter2++;
144.                    break;
145.                case 2:
146.                    indexNew=2;
147.                    for (int j = 0; j < arrNewCenteroid[0].length; j++) {
148.                        arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter3;
149.                    }
150.                    counter3++;
151.                    break;
152.                case 3:
153.                    indexNew=3;
154.                    for (int j = 0; j < arrNewCenteroid[0].length; j++) {
155.                        arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter4;
156.                    }
157.                    counter4++;
158.                    break;
159.                case 4:
160.                    indexNew=4;
161.                    for (int j = 0; j < arrNewCenteroid[0].length; j++) {
```

```
                arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter5;
            }
            counter5++;
            break;
        case 5:
            indexNew=5;
            for (int j = 0; j < arrNewCenteroid[0].length; j++) {
                arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter1;
            }
            counter6++;
            break;
        case 6:
            indexNew=6;
            for (int j = 0; j < arrNewCenteroid[0].length; j++) {
                arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter2;
            }
            counter7++;
            break;
        case 7:
            indexNew=7;
            for (int j = 0; j < arrNewCenteroid[0].length; j++) {
                arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter3;
            }
            counter8++;
            break;
        case 8:
            indexNew=8;
            for (int j = 0; j < arrNewCenteroid[0].length; j++) {
                arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter4;
            }
            counter9++;
            break;
        case 9:
            indexNew=9;
            for (int j = 0; j < arrNewCenteroid[0].length; j++) {
                arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter5;
            }
            counter10++;
            break;
        case 10:
            indexNew=10;
            for (int j = 0; j < arrNewCenteroid[0].length; j++) {
                arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter1;
            }
            counter11++;
            break;
        case 11:
            indexNew=11;
            for (int j = 0; j < arrNewCenteroid[0].length; j++) {
                arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter2;
            }
            counter12++;
            break;
        case 12:
            indexNew=12;
            for (int j = 0; j < arrNewCenteroid[0].length; j++) {
                arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter3;
            }
            counter13++;
            break;
        case 13:
            indexNew=13;
            for (int j = 0; j < arrNewCenteroid[0].length; j++) {
                arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter4;
            }
            counter14++;
```

```
                break;
            case 14:
                indexNew=14;
                for (int j = 0; j < arrNewCenteroid[0].length; j++) {
                    arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter5;
                }
                counter15++;
                break;
            case 15:
                indexNew=15;
                for (int j = 0; j < arrNewCenteroid[0].length; j++) {
                    arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter4;
                }
                counter16++;
                break;
            case 16:
                indexNew=16;
                for (int j = 0; j < arrNewCenteroid[0].length; j++) {
                    arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew][j]+arrDataTraining[i][j])/counter5;
                }
                counter5++;
                break;
        }
    }
}

public boolean cekCenteroid(){
    boolean cek = false;
    int cekCount = 0;
    for (int i = 0; i < arrCenteroid.length; i++) {
        for (int j = 0; j < arrCenteroid[0].length; j++) {
            if (arrNewCenteroid[i][j]!= arrCenteroid[i][j]) {
                cekCount++;
            }
        }
    }
    if (cekCount<2) {
        cek=true;
    } else{
        //simpan centeroid baru
        for (int i = 0; i < arrCenteroid.length; i++) {
            for (int j = 0; j < arrCenteroid[0].length; j++) {
                arrCenteroid[i][j]=arrNewCenteroid[i][j];
            }
    }
    }

    return cek;
}

public  void printJarakCenteroid(){
    for (int i = 0; i < arrJarakCenteroid.length; i++) {
        for (int j = 0; j < arrJarakCenteroid[0].length; j++) {
            System.out.print(df.format(arrJarakCenteroid[i][j])+"\t");
        }
        System.out.println("");
    }
}

public  void printCenteroid(){
    for (int i = 0; i < arrNewCenteroid.length; i++) {
            for (int j = 0; j < arrNewCenteroid[0].length; j++) {
                System.out.print(df.format(arrNewCenteroid[i][j])+"\t");
            }
            System.out.println("");
        }
```

```
294.         System.out.println("DATA");
295.         for (int i = 0; i < arrDataTraining.length; i++) {
296.                 for (int j = 0; j < arrDataTraining[0].length; j++) {
297.                     System.out.print(df.format(arrDataTraining[i][j])+"\t");
298.                 }
299.                 System.out.println("");
300.             }
301.     }
302.
303. }
```

Code 3.2 class PenyakitKelaminKMeans

```
1.  package penyakitkelaminkmeans;
2.
3.  import java.text.DecimalFormat;
4.
5.  public class PenyakitKelaminKMeans {
6.
7.      public static void main(String[] args) {
8.          // TODO code application logic here
9.
10.         KMeans km = new KMeans();
11.
12.         km.readData();
13.         km.randomCenteroid();
14.         int count =1;
15.         do {
16.             System.out.println("Iterasi"+count);
17.             km.hitungJarakCenteroid();
18.             km.updateCenteroid();
19.             km.printJarakCenteroid();
20.             km.printCenteroid();
21.             System.out.println("");
22.             count++;
23.         } while (count<10);//km.cekCenteroid()==false);
24.
25.     }
26.
27. //        for (int i = 0; i < km.arrCenteroid.length; i++) {
28. //            for (int j = 0; j < km.arrCenteroid[0].length; j++) {
29. //                System.out.print(km.arrCenteroid[i][j]+"\t");
30. //            }
31. //            System.out.println("");
32. //        }
33.
34.
35. }
```

Metode K-means ++

Konsep Metode

K-Means++ *Clustering* merupakan metode pengembangan dari K-Means. Metode ini dinilai dapat mengatasi permasalahan yang ada pada K-Means yakni berkaitan dengan akurasi dan kecepatan. Jika pada K-Means pemilihan *centroid* dilakukan secara acak, sedangkan pada

K-Means++ pemilihan centroid dilakukan dengan menggunakan perhitungan probabilitas (Kumar 2020).

Langkah-langkah yang perlu dilakukan dalam proses K-Means++ yakni sebagai berikut:

1. Memilih titik *centroid* pertama c_1 secara acak.

2. Menghitung jarak semua titik pada *dataset* dari *centroid* yang dipilih. Jarak suatu titik dengan titik *centroid* terjauh dapat dihitung dengan menggunakan rumus sebagai berikut:

$$d_i = \|x_i - c_j\|^2$$

Dimana d_i merupakan jarak dari titik x_i dari titik *centroid* terjauh, sedangkan m merupakan jumlah titik *centroid* yang terpilih.

3. Menjadikan titik x_i sebagai *centroid* baru yang memiliki probabilitas maksimum sebanding dengan d_i.

4. Mengulangi dua langkah sebelumnya hingga menemukan *k centroid*.

1.1.1 Pembuatan Program K-Means++

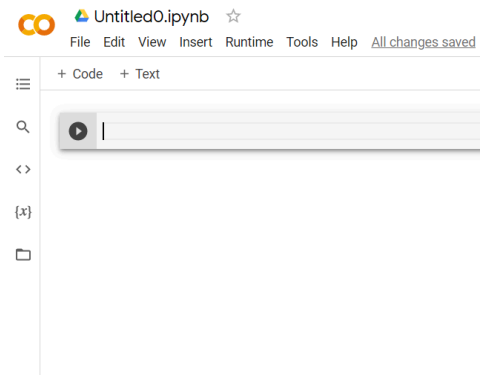
Untuk membuat program K-Means++, pada contoh ini akan menggunakan Google Colab. Langkah-langkah yang perlu dilakukan yakni sebagai berikut:

1. Buat file excel dengan *attribute* (kolom) dan nilai data (baris) seperti pada Gambar 3.14 dan simpan dengan format .xlsx.

	A	B
1	x	y
2	4	5
3	1	6
4	2	7
5	6	8
6	4	8
7	5	4
8	2	3
9	3	7
10	3	3
11	5	9
12	3	4
13	3	5
14	2	8
15	4	7
16	2	1
17	5	6
18	5	5

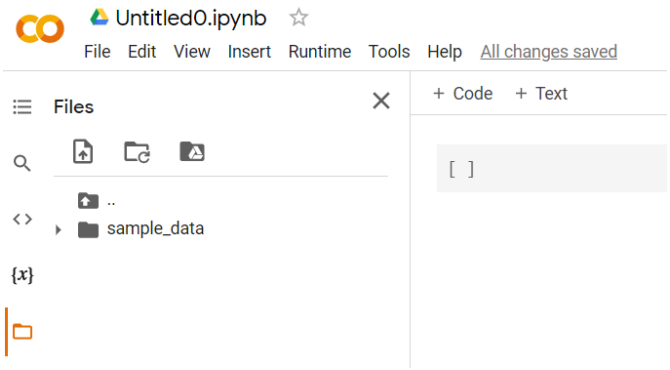
Gambar 3.14 File Excel

2. Buka Google Colab dengan alamat: https://colab.research.google.com/drive/1DvK74PokYRrLKMg0skVahsKzygeEKbMu?usp=sharing.
3. Masuk pada menu *file*, kemudian pilih *new notebook*, maka akan tampil seperti pada Gambar 3.15.



Gambar 3.15 New Notebook

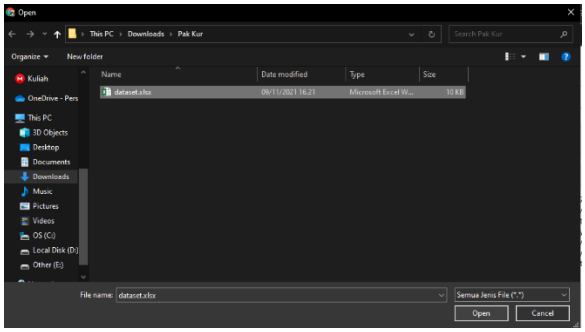
4. Tekan button folder seperti pada Gambar dibawah ini:



Gambar 3.16 Button Folder

5. Upload *file* Excel yang telah dibuat pada langkah pertama dengan menekan *button* klik pada tombol

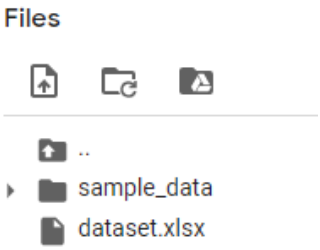
   Setelah menekan *button* maka akan muncul window dialog



Gambar 3.17 Window Dialog

   Pilih file yang telah dibuat sebelumnya dan tekan button open.

6. Klik open dan pastikan sudah terimport



Gambar 3.18 Import Dataset

7. Pada baris pertama kode ketikkan perintah:

   Pertama tulis tulis kode untuk menghilangkan warning pada cell:

   import warnings

   warnings.filterwarnings('ignore')

```
import warnings
warnings.filterwarnings('ignore')
```

Gambar 3.19 Warning

Seletah menuliskan perintah tersebut maka kita melakukan run program dengan menekan tombol run (tombol play).

8. Tuliskan code pada cell selanjutnya dengan perintah:

   !pip install -U scikit-learn

9. Import library-library yang dibutuhkan untuk program K-Means++ ini.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
```

Gambar 3.20 Import Library

Library pandas digunakan untuk mengambil atau mengekstrak nilai dari file excel atau csv kedalam Data Frame Python. Library matplolib dan seaborn digunakan untuk menampilkan plotting dan scatter data. Library numpy untuk membuat data menjadi array. Library KMEANS untuk *clustering* dengan Metode K-Means.

10. Pada cell selanjutkan ketikan perintah untuk mengambil data dari file excel seperti pada code dibawah ini.

```
df = pd.read_excel("dataset.xlsx")
df
```

Gambar 3.21 Perintah Mengambil Data

Setelah di run maka akan muncul hasil nilai seperti pada nilai yang dibuat di excel.

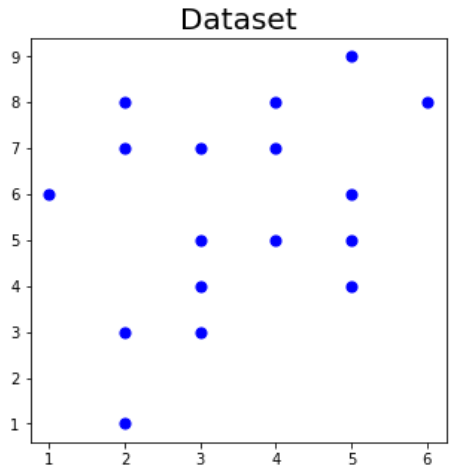
	x	y
0	4	5
1	1	6
2	2	7
3	6	8
4	4	8
5	5	4
6	2	3

Gambar 3.22 Nilai yang Muncul

11. Untuk mendapatkan Gambaran dari data maka kita perlu membuat scatter plotting dari *dataset* yang telah berhasil diambil nilainya pada cell sebelumnya. Ketikan perintah dibawah ini untuk mendapatkan Gambar scatter plotting *dataset*

```
plt.figure(figsize=(5,5))
plt.scatter(df.iloc[:,0],df.iloc[:,1],color='blue',s=50)
plt.title('Dataset',fontsize=20)
plt.show()
```

Gambar 3.23 Perintah Gambar Scatter Plotting Dataset



Gambar 3.24 Gambar Scatter Plotting Dataset

12. Selanjutnya melakukan training model data dengan mendefinisikan jumlah cluster yang kita inginkan pada parameter **n_cluster**. Untuk menggunakan K-Means++, jenis metode inisialisasi didefinisikan pada parameter **init**

```
kmeans = KMeans(n_clusters = 3, init='k-means++')
kmeans.fit(df)
```

Gambar 3.25 Perintah Inisialisi Jumlah Cluster

n_clusters=3 merupakan argument yang digunakan untuk membuat 3 cluster.

13. Untuk melihat label yang terbentuk pada training K-Means++ adalah sebagai berikut:

```
labels = db.labels_
labels
```

Gambar 3.26 Perintah Melihat Label

Hasilnya adalah:

```
array([1, 0, 0, 2, 2, 1, 1, 0, 1, 2, 1, 1, 0, 2, 1, 2, 1], dtype=int32)
```

Gambar 3.27 Hasil Training Data

14. Untuk melihat pusat cluster adalah sebagai berikut:

```
cluster_center = kmeans.cluster_centers_
cluster_center
```

Gambar 3.28 Perintah Melihat Pusat Cluster

Hasilnya adalah:

```
array([[2.   , 7.   ],
       [3.375, 3.75 ],
       [4.8  , 7.6  ]])
```

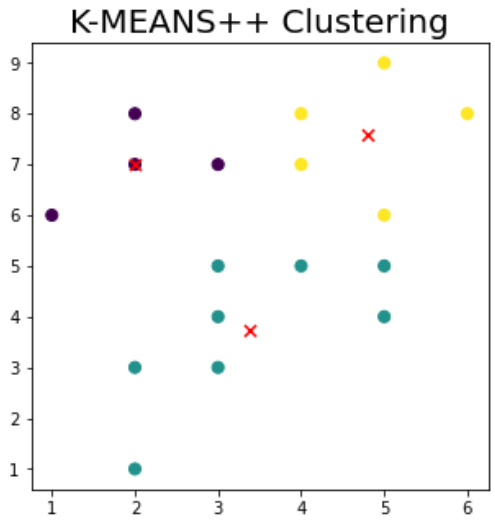
Gambar 3.29 Hasil Pusat Cluster

15. Langkah selanjutnya adalah visualisasi data hasil *clustering* dengan kode sebagai berikut

```
plt.figure(figsize=(5,5))
plt.scatter(df.iloc[:,0],df.iloc[:,1],c=labels,s=50)
plt.scatter(cluster_center[:,0],cluster_center[:,1],marker='x',s=50, c='red')
plt.title('K-MEANS++ Clustering',fontsize=20)
plt.show()
```

Gambar 3.30 Perintah Visualisasi Hasil Clustering

Hasil visualisasi data hasil *clustering* adalah sebagai berikut:



Gambar 3.31 Visualisasi Hasil Clustering

Dapat dilihat bahwa metode K-MEANS++ membentuk 3 cluster dengan pusat cluster yang ditandai dengan simbol x

16. Selanjutnya adalah menghitung SSE dengan jumlah cluster yang diuji adalah 1 sampai 10. berikut adalah kode untuk mendapatkan SSE:

```
sse = []
idx = np.arange(1,11)
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, init = 'k-means++')
    kmeans.fit(df)
    sse.append(kmeans.inertia_)

SSE = pd.DataFrame([idx, sse]).transpose()
SSE.columns = ['n_cluster','SSE']
SSE
```

Gambar 3.32 Perintah Menghitung SSE

Hasilnya adalah:

	n_cluster	SSE
0	1.0	108.117647
1	2.0	53.597222
2	3.0	35.375000
3	4.0	19.700000
4	5.0	15.000000
5	6.0	11.583333
6	7.0	9.083333
7	8.0	6.916667
8	9.0	5.166667
9	10.0	4.166667

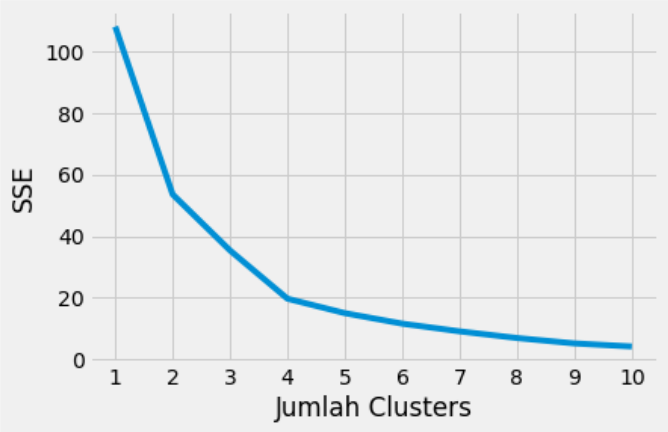
Gambar 3.33 Hasil Perhitungan SSE

17. Langkah terakhir adalah visualisasi SSE untuk setiap cluster dengan kode sebagai berikut:

```
plt.style.use("fivethirtyeight")
plt.plot(range(1, 11), sse)
plt.xticks(range(1, 11))
plt.xlabel("Jumlah Clusters")
plt.ylabel("SSE")
plt.show()
```

Gambar 3.34 Perintah Visualisasi SSE

Hasil visualisasinya adalah:



Gambar 3.35 Visualisasi SSE