KUPAS TUNTAS

Algoritma Clustering

KONSEP, PERHITUNGAN MANUAL, DAN PROGRAM

Rani Rotul Muhima, S.Si., M.T. | Muchamad Kurniawan, S.Kom., M.Kom.
Septiyawan Rosetya Wardhana, S.Kom., M.Kom. | Anton Yudhana, S.T., M.T., Ph.D.
Sunardi, S.T., M.T., Ph.D. | Weny Mistarika Rahmawati, S.Kom., M.Kom., M.Sc.
Gusti Eka Yuliastuti, S.Kom., M.Kom.

Penerbit ANDI

KUPAS TUNTAS ALGORITMA CLUSTERING: KONSEP PERHITUNGAN MANUAL DAN PROGRAM

Oleh: Rani Rotul Muhima, S.Si., M.T.
Muchamad Kurniawan, S.Kom., M.Kom.
Septiyawan Rosetya Wardhana, S.Kom., M.Kom.
Anton Yudhana, S.T., M.T., Ph.D.
Sunardi, S.T., M.T., Ph.D.
Weny Mistarika Rahmawati, S.Kom., M.Kom., M.Sc.
Gusti Eka Yuliastuti, S.Kom., M.Kom.

Editor : Erang Risanto
Desain Cover : Andang Suhana
Setter : Andika Sundoro Aji
Korektor : Rimuru

Diterbitkan oleh Penerbit ANDI (Anggota IKAPI)
Jl. Beo 38-40, Telp. (0274) 561881 (Hunting), Fax. (0274) 588282 Yogyakarta 55281

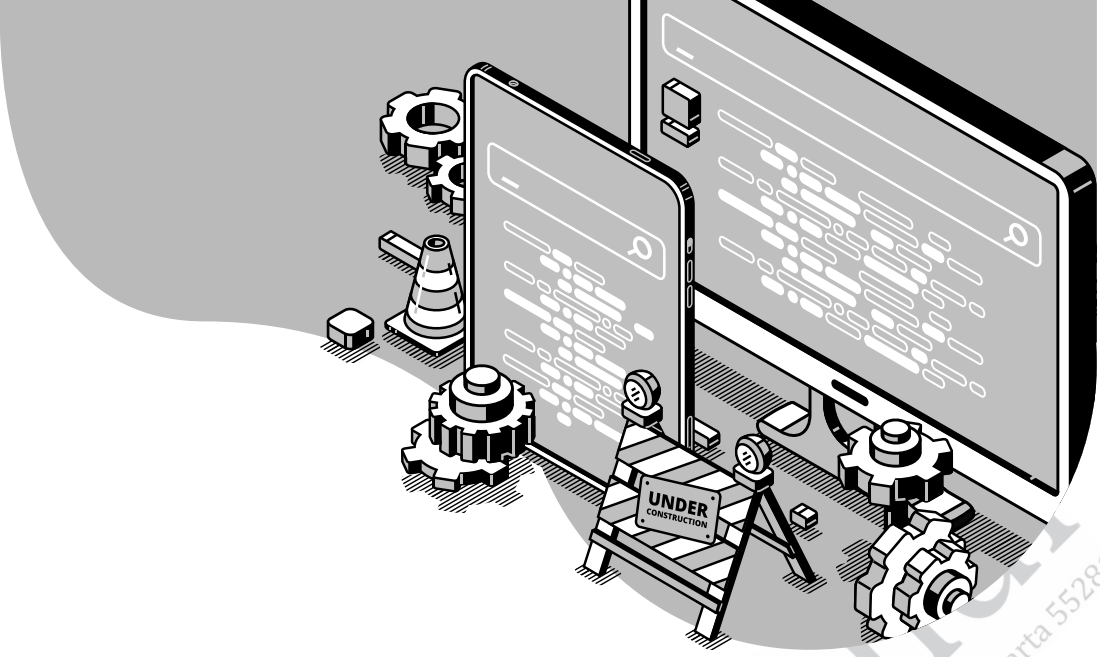
Percetakan: CV ANDI OFFSET
Jl. Beo 38-40, Telp. (0274) 561881 (Hunting), Fax. (0274) 588282 Yogyakarta 55281

Muhima, Rani Rotul
KUPAS TUNTAS ALGORITMA CLUSTERING: KONSEP PERHITUNGAN MANUAL DAN PROGRAM / Rani Rotul Muhima, Muchamad Kurniawan, Septiyawan Rosetya Wardhana, Anton Yudhana, Sunardi, Weny Mistarika Rahmawati, dan Gusti Eka Yuliastuti
– Ed. I. – Yogyakarta: ANDI;

30 – 29 – 28 – 27 – 26 – 25 – 24 – 23 – 22 – 21
hlm xiv + 226; 16 x 23 cm.
10 9 8 7 6 5 4 3 2 1
ISBN: 978-623-01-2564-5
978-623-01-2563-8 (PDF)

I. Judul
1. Data Processing, Computer Science
2. Kurniawan, Muchamad
3. Wardhana, Septiyawan Rosetya
4. Yudhana, Anton
5. Sunardi
6. Rahmawati, Weny Mistarika
7. Yuliastuti, Gusti Eka

DDC'23 : 004



PRAKATA

Puji syukur atas rahmat Allah Yang Mahakuasa sehingga buku *Kupas Tuntas Algoritma Clustering: Konsep Perhitungan Manual dan Program* dapat tersusun dengan baik oleh Tim Penyusun. Buku ajar ini menyajikan konsep dasar dari beberapa metode *clustering*. Penyajian setiap bab disertai perhitungan manual agar pembaca dapat memahami algoritma yang disajikan pada konsep teori dari metode *clustering*. Beberapa metode yang disajikan diberikan contoh program dengan menggunakan bahasa Python, dengan harapan dapat memudahkan pembaca memahami dari bahasa konsep ke dalam bahasa program Python.

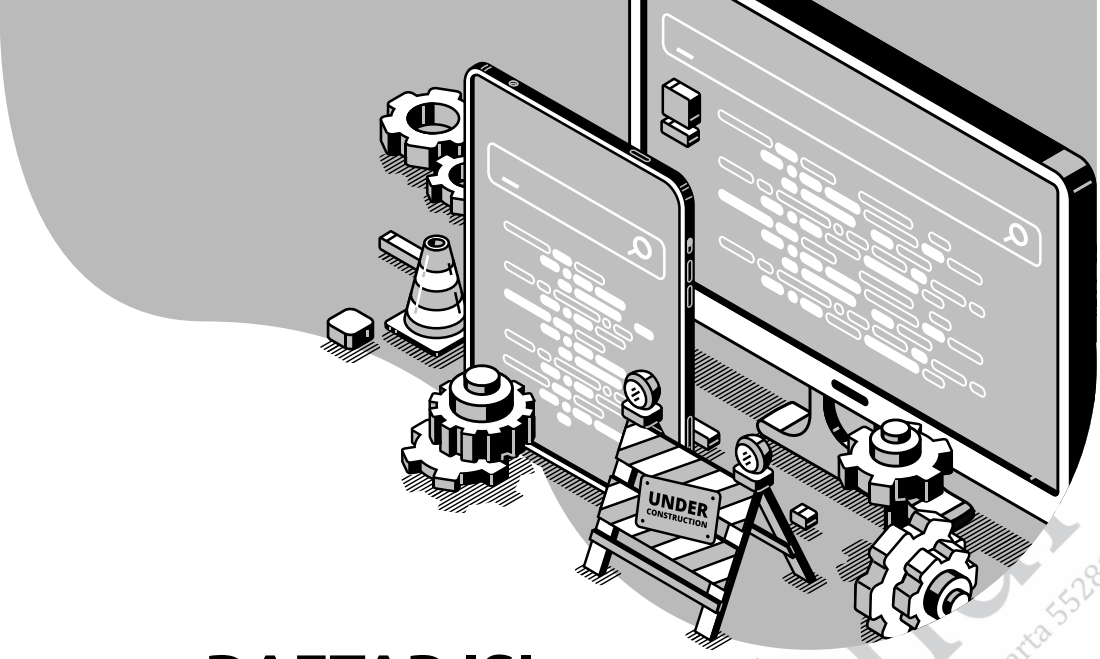
Beberapa materi yang disajikan dalam buku ini, antara lain metode clustering berbasiskan hierarki, metode clustering berbasiskan partisi, metode clustering berbasiskan density, metode clustering berbasiskan model, serta penerapan clustering. Metode clustering berbasiskan hierarki merupakan metode yang mengelompokkan data menjadi kelompok-kelompok secara rekursif. Metode clustering

yang dijelaskan, yaitu metode Agglomerative dan Divisive Hierarki. Metode agglomerative menyajikan beberapa cara untuk menentukan jarak antar-cluster, yaitu Single Linkage, Complete Linkage, Average Linkage, dan Distance Between Centroid. Bab selanjutnya membahas metode clustering berbasiskan partisi. Metode ini merupakan metode pemisahan data satu tingkat pada dataset. Beberapa contoh metode clustering yang dibahas, antara lain metode K-means, metode K-means++, metode K-medoid, dan Fuzzy C-means. Metode Clustering berbasiskan density, metode clustering yang diperkenalkan untuk menentukan cluster pada data spasial yang memiliki noise. Pada bab ini dibahas metode DBSCAN dan DENCLUE, keduanya adalah contoh metode clustering berbasiskan density. Pembahasan selanjutnya adalah metode clustering berbasiskan model. Output dari metode ini berdasarkan model yang dibuat. Beberapa contoh metode yang dibahas adalah Self Organized Maps dan Optimization Clustering. Optimization Clustering sendiri, contoh metode yang dibahas adalah PSO Clustering. Buku ini juga menyajikan contoh-contoh penerapan metode clustering pada data spasial, data teks, dan citra.

Tim Penyusun mengucapkan terima kasih kepada semua pihak yang membantu terselesaikannya penyusunan buku ini khususnya Kementerian Riset Teknologi-Badan Riset dan Inovasi Nasional (RISTEK-BRIN) Indonesia. Buku ini merupakan luaran Hibah Penelitian Kerja Sama Perguruan Tinggi (PKPT) yang didanai oleh RISTEK BRIN. Semoga buku ini dapat menjadi bahan pelengkap ilmu pengetahuan khususnya pendukung mata kuliah Data Mining di Jurusan Teknik Informatika.

Surabaya, November 2021

Tim penyusun



DAFTAR ISI

PRAKATA ..... **iii**
DAFTAR ISI ..... **v**
DAFTAR TABEL ..... **vii**
DAFTAR GAMBAR ..... **ix**

BAB I PENGANTAR ANALISIS CLUSTERING **1**
BAB II METODE CLUSTERING BERBASISKAN HIERARKI **5**
2.1 METODE AGGLOMERATIVE 7
2.1.1 Single Linkage 7
2.1.2 Complete Linkage 25
2.1.3 Average Linkage 36
2.1.4 Distance Between Centroid 46
2.2 DIVISIVE HIERARKI 57
2.3 LATIHAN SOAL CLUSTERING BERBASISKAN HIERARKI 65
BAB III METODE CLUSTERING BERBASISKAN PARTISI **67**
3.1 METODE K-MEANS 68
3.2 METODE K-MEANS ++ 85
3.2.1 Konsep Metode 85
3.2.2 Pembuatan Program K-Means++ 85
3.3 METODE K-MEDOID 93
3.3.1 Konsep Metode 93
3.3.2 Pembuatan Program K-Medoid 100

3.4 FUZZY C-MEANS (FCM) 106
3.4.1 Konsep Metode 106
3.4.2 Perhitungan Manual FCM 108
3.4.3 Pembuatan Program FCM 112
3.5 LATIHAN SOAL CLUSTERING BERBASISKAN PARTISI 118

BAB IV METODE CLUSTERING BERBASISKAN DENSITY 119

4.1 METODE DBSCAN 119
4.1.1 Directly Density Reachable 121
4.1.2 Density Reachable 122
4.1.3 Density Connected 122
4.1.4 Cluster 123
4.1.5 Perhitungan Manual DBSCAN 123
4.1.6 Pembuatan Program DBSCAN 138
4.2 METODE DENCLUE 145
4.2.1 Fungsi Pengaruh 145
4.2.2 Fungsi Densitas 146
4.2.3 Penarik Densitas 146
4.2.4 Algoritma DENCLUE 147
4.3 LATIHAN SOAL CLUSTERING BERBASISKAN DENSITY 148

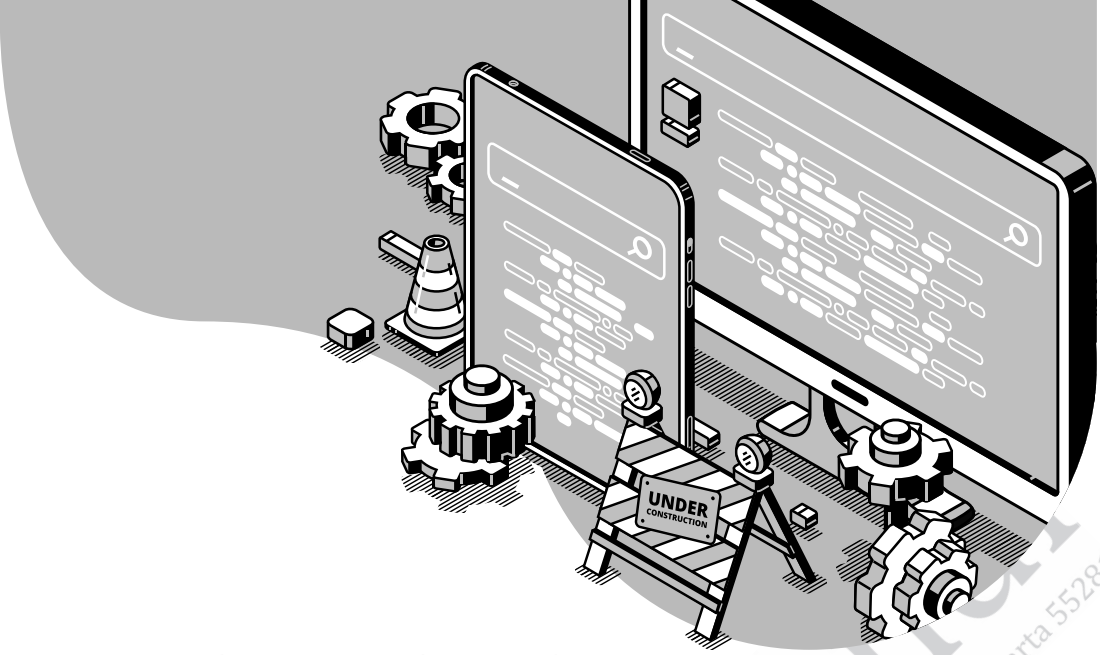
BAB V METODE CLUSTERING BERBASISKAN MODEL 149

5.1 SELF ORGANIZED MAPS 151
5.1.1 Konsep Algoritma 151
5.1.2 Pembuatan Program Self Organized Maps 156
5.2 OPTIMIZATION CLUSTERING 162
5.3 LATIHAN SOAL CLUSTERING BERDASARKAN MODEL 208

BAB VI PENERAPAN CLUSTERING 209

6.1 CLUSTERING PADA ACTIVE FIRE NASA DATASET 209
6.2 CLUSTERING PADA TEXT MINING 212
6.3 CLUSTERING UNTUK STATUS GIZI BALITA 215
6.4 CLUSTERING UNTUK SEGMENTASI CITRA DIGITAL WAJAH MANUSIA 217

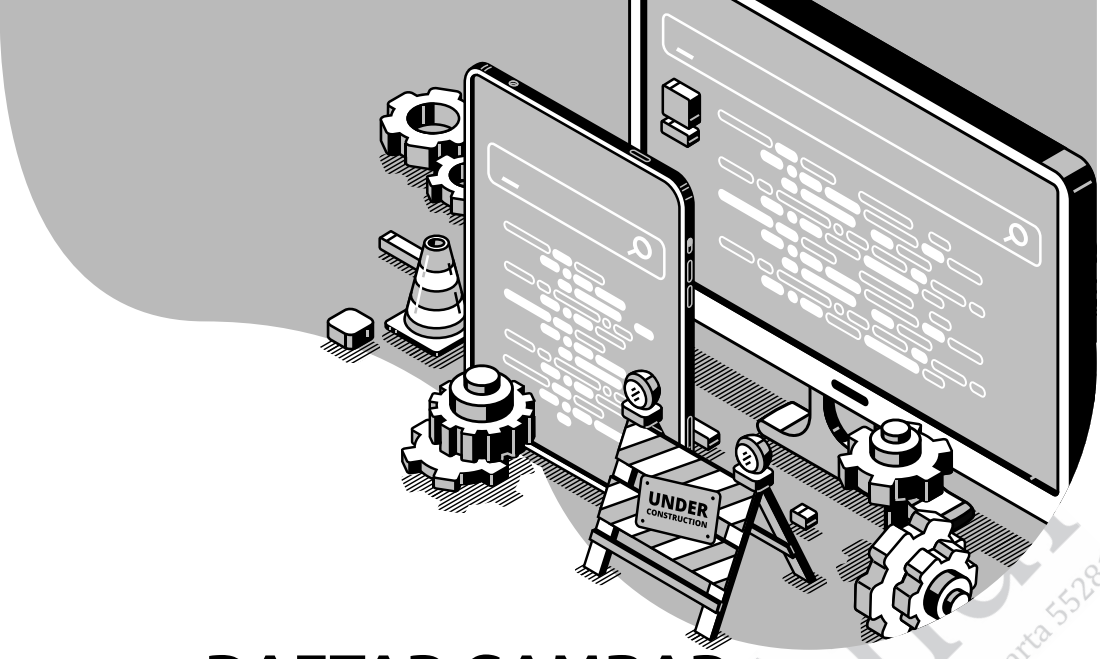
DAFTAR PUSTAKA 219
BIOGRAFI PENULIS 221



DAFTAR TABEL

Tabel 2.1 Contoh Dataset 8
Tabel 2.2 Perhitungan Jarak Antardata 9
Tabel 2.3 Jarak Antar-cluster Single Linkage Iterasi ke-2 10
Tabel 2.4 Jarak Antar-cluster Single Linkage Iterasi ke-3 11
Tabel 2.5 Jarak Antar-cluster Single Linkage Iterasi ke-4 12
Tabel 2.6 Jarak Antar-cluster Single Linkage Iterasi ke-5 13
Tabel 2.7 Jarak Antar-cluster Single Linkage Iterasi ke-6 13
Tabel 2.8 Jarak Antar-cluster Single Linkage Iterasi ke-7 14
Tabel 2.9 Jarak Antar-cluster Single Linkage Iterasi ke-8 14
Tabel 2.10 Jarak Antar-cluster Complete Linkage Iterasi ke-3 27
Tabel 2.11 Jarak Antar-cluster Complete Linkage Iterasi ke-4 28
Tabel 2.12 Jarak Antar-cluster Complete Linkage Iterasi ke-5 28
Tabel 2.13 Jarak Antar-cluster Complete Linkage Iterasi ke-6 28
Tabel 2.14 Jarak Antar-cluster Complete Linkage Iterasi ke-7 29
Tabel 2.15 Jarak Antar-cluster Complete Linkage Iterasi ke-8 29
Tabel 2.16 Jarak Antar-cluster Complete Linkage Iterasi ke-9 29
Tabel 2.17 Jarak Antar-cluster Average Linkage Iterasi ke-3 38
Tabel 2.18 Jarak Antar-cluster Average Linkage Iterasi ke-4 38
Tabel 2.19 Jarak Antar-cluster Average Linkage Iterasi ke-5 38
Tabel 2.20 Jarak Antar-cluster Average Linkage Iterasi ke-6 39
Tabel 2.21 Jarak Antar-cluster Average Linkage Iterasi ke-7 39
Tabel 2.22 Jarak Antar-cluster Average Linkage Iterasi ke-8 39
Tabel 2.23 Jarak Antar-cluster Average Linkage Iterasi ke-9 39
Tabel 2.24 Jarak Antar-cluster Menggunakan Jarak Antar-centroid Iterasi ke-3 48

Tabel 2.25 Jarak Antar-cluster Menggunakan Jarak Antar-centroid Iterasi ke-4 49
Tabel 2.26 Jarak Antar-cluster Menggunakan Jarak Antar-centroid Iterasi ke-5 49
Tabel 2.27 Jarak Antar-cluster Menggunakan Jarak Antar-centroid Iterasi ke-6 50
Tabel 2.28 Jarak Antar-cluster Menggunakan Jarak Antar-centroid Iterasi ke-7 50
Tabel 2.29 Jarak Antar-cluster Menggunakan Jarak Antar-centroid Iterasi ke-8 50
Tabel 2.30 Jarak Antar-cluster Menggunakan Jarak Antar-centroid Iterasi ke-9 50
Tabel 2.31 Jarak Antarobjek dengan Euclidean Distance 59
Tabel 2.32 Rata-Rata Jarak Objek dengan Objek Lain 59
Tabel 2.33 Rata-Rata Jarak dengan Objek Lain selain Splinter 60
Tabel 2.34 Selisih Rata-Rata Jarak Antarobjek dengan Objek pada Splinter 61
Tabel 2.35 Rata-Rata Cluster yang Terbentuk 62
Tabel 2.36 Rata-Rata Jarak Antarobjek Iterasi Kedua 62
Tabel 2.37 Rata-Rata Jarak Objek dengan Objek Lain selain Splinter Iterasi Kedua 63
Tabel 2.38 Selisih Rata-Rata Jarak Antarobjek dengan Objek pada Splinter Iterasi Kedua 63
Tabel 2.39 Rata-Rata Cluster yang Terbentuk Iterasi Kedua 64
Tabel 3.1 Contoh Data Awal 94
Tabel 3.2 Perhitungan Jarak Objek ke Medoid Awal 1 96
Tabel 3.3 Perhitungan Jarak Objek ke Medoid Awal 2 97
Tabel 3.4 Perhitungan Jarak Objek ke Medoid 1 98
Tabel 3.5 Perhitungan Jarak Objek ke Medoid 2 99
Tabel 3.6 Contoh Data Awal 108
Tabel 3.7 Perhitungan Jarak Objek ke Centroid 1 109
Tabel 3.8 Perhitungan Jarak Objek ke Centroid 2 110
Tabel 3.9 Jarak Titik Data pada Tiap Cluster 110
Tabel 3.10 Matriks Keanggotaan Tiap Cluster 111
Tabel 3.11 Data Akhir Hasil Pengolahan 111
Tabel 4.1 Nilai Jarak Setiap Titik terhadap Data ke-15 124
Tabel 4.2 Nilai Jarak Setiap Titik dengan Data ke-7 125
Tabel 4.3 Nilai Jarak Setiap Titik dengan Data ke-11 127
Tabel 4.4 Nilai Jarak Setiap Titik dengan Data ke-6 129
Tabel 4.5 Nilai Jarak Setiap Titik dengan Data ke-6 130
Tabel 4.6 Nilai Jarak Setiap Titik dengan Data ke-14 132
Tabel 4.7 Nilai Jarak Setiap Titik dengan Data ke-3 133
Tabel 4.8 Nilai Jarak Setiap Titik dengan Data ke-2 135
Tabel 4.9 Nilai Jarak Setiap Titik dengan Data ke-4 136



DAFTAR GAMBAR

Gambar 2.1 Simulasi Single Linkage 8
Gambar 2.2 Dendrogram Single Linkage 15
Gambar 2.3 Attribute dan Nilai pada File Excel 15
Gambar 2.4 Ilustrasi Perhitungan Jarak Maksimal Antara Cluster 25
Gambar 2.5 Dendrogram Complete Linkage 30
Gambar 2.6 File Excel 30
Gambar 2.7 New Notebook 31
Gambar 2.8 Button Folder 31
Gambar 2.9 Window Dialog 32
Gambar 2.10 Import Dataset 32
Gambar 2.11 Warning 32
Gambar 2.12 Import Library 33
Gambar 2.13 Perintah Mengambil Data 33
Gambar 2.14 Hasil Nilai 33
Gambar 2.15 Perintah Gambar Scatter Plotting Dataset 34
Gambar 2.16 Gambar Scatter Plotting Dataset 34
Gambar 2.17 Training Model 34
Gambar 2.18 Training Complete Linkage 35
Gambar 2.19 Perintah Visualisasi Hasil Clustering 35
Gambar 2.20 Visualisasi Data Hasil Clustering 35
Gambar 2.21 Perintah Visualisasi Dendrogram 36
Gambar 2.22 Dendrogram Complete Linkage 36

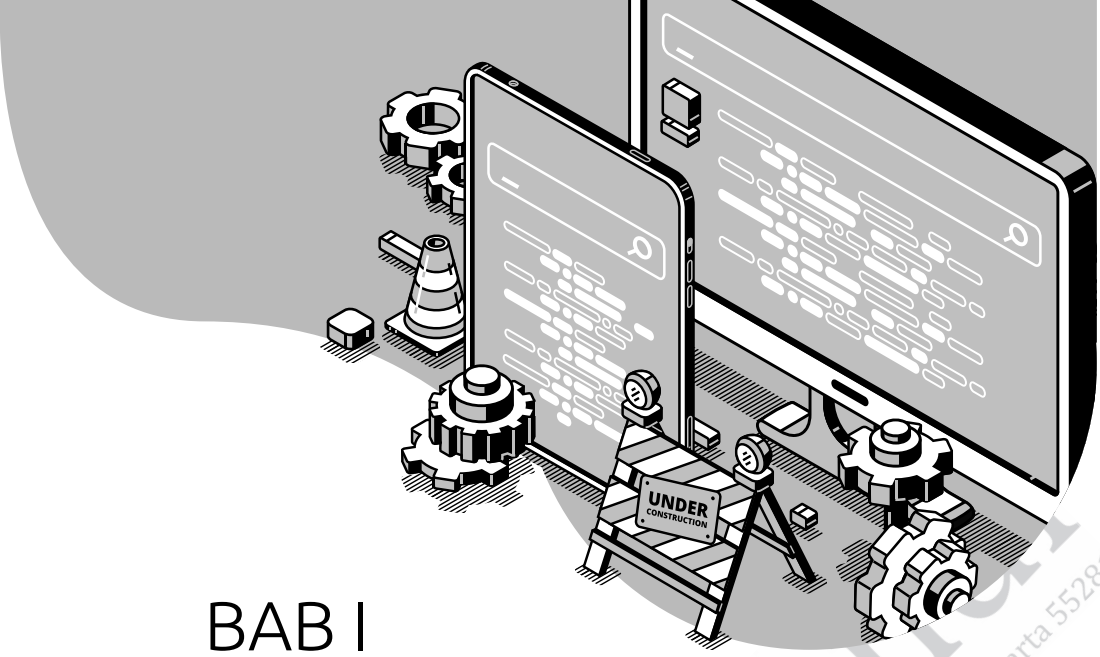
Gambar 2.23 Ilustrasi Average Linkage 37
Gambar 2.24 Dendrogram Average Linkage 40
Gambar 2.25 File Excel 40
Gambar 2.26 New Notebook 41
Gambar 2.27 Button Folder 41
Gambar 2.28 Window Dialog 42
Gambar 2.29 Import Dataset 42
Gambar 2.30 Warning 42
Gambar 2.31 Import Library 43
Gambar 2.32 Perintah Mengambil Data 43
Gambar 2.33 Hasil Nilai 43
Gambar 2.34 Perintah Gambar Scatter Plotting Dataset 44
Gambar 2.35 Gambar Scatter Plotting Dataset 44
Gambar 2.36 Training Model Average Linkage 44
Gambar 2.37 Perintah Melihat Label 45
Gambar 2.38 Hasil Average Linkage 45
Gambar 2.39 Perintah Visualisasi Hasil Clustering 45
Gambar 2.40 Visualisasi Hasil Clustering 45
Gambar 2.41 Perintah Visualisasi Dendrogram Average Linkage 46
Gambar 2.42 Dendrogram Average Linkage 46
Gambar 2.43 Ilustrasi Perhitungan Jarak Antar-cluster Menggunakan Centroid .. 47
Gambar 2.44 Dendrogram Distance Between Centroid 51
Gambar 2.45 File Excel 52
Gambar 2.46 New Notebook 52
Gambar 2.47 Button Folder 53
Gambar 2.48 Window Dialog 53
Gambar 2.49 Import Dataset 53
Gambar 2.50 Warning 54
Gambar 2.51 Import Library 54
Gambar 2.52 Perintah Mengambil Data 54
Gambar 2.53 Hasil Nilai 55
Gambar 2.54 Perintah Gambar Scatter Plotting Dataset 55
Gambar 2.55 Gambar Scatter Plotting Dataset 55
Gambar 2.56 Perintah Plot Dendrogram 56
Gambar 2.57 Dendrogram Distance Between Centroid 56
Gambar 2.58 Perintah Pemodelan 56

Gambar 2.59 Perintah Hasil Perhitungan 56
Gambar 2.60 Perintah Visualisasi Data 57
Gambar 2.61 Hasil Visualisasi Data 57
Gambar 3.1 Contoh Jumlah Cluster K (James *et al.,* 2013) 68
Gambar 3.2 Ilustrasi Data Awal (James *et al.,* 2013) 69
Gambar 3.3 Penentuan Cluster Awal (James *et al.,* 2013) 69
Gambar 3.4 Perhitungan Jarak Titik dengan Centroid (James *et al.,* 2013) ... 70
Gambar 3.5 Penetapan Satu Titik pada Cluster (James *et al.,* 2013) 70
Gambar 3.6 Penetapan Semua Titik pada Cluster (James *et al.,* 2013) 71
Gambar 3.7 Menghitung Nilai Rata-Rata Cluster (James *et al.,* 2013) 71
Gambar 3.8 Pencarian Centroid Baru Tiap Cluster (James *et al.,* 2013) 71
Gambar 3.9 Hasil dari Proses Clustering (James *et al.,* 2013) 72
Gambar 3.10 Perhitungan Varians Tiap Cluster (James *et al.,* 2013) 72
Gambar 3.11 Total Varians Tiap Cluster (James *et al.,* 2013) 73
Gambar 3.12 Pemilihan Cluster (James *et al.,* 2013) 73
Gambar 3.13 Hasil Akhir dari Proses Clustering (James *et al.,* 2013) 73
Gambar 3.14 File Excel 86
Gambar 3.15 New Notebook 86
Gambar 3.16 Button Folder 87
Gambar 3.17 Window Dialog 87
Gambar 3.18 Import Dataset 88
Gambar 3.19 Warning 88
Gambar 3.20 Import Library 88
Gambar 3.21 Perintah Mengambil Data 89
Gambar 3.22 Nilai yang Muncul 89
Gambar 3.23 Perintah Gambar Scatter Plotting Dataset 89
Gambar 3.24 Gambar Scatter Plotting Dataset 90
Gambar 3.25 Perintah Inisialisi Jumlah Cluster 90
Gambar 3.26 Perintah Melihat Label 90
Gambar 3.27 Hasil Training Data 90
Gambar 3.28 Perintah Melihat Pusat Cluster 91
Gambar 3.29 Hasil Pusat Cluster 91
Gambar 3.30 Perintah Visualisasi Hasil Clustering 91
Gambar 3.31 Visualisasi Hasil Clustering 91
Gambar 3.32 Perintah Menghitung SSE 92
Gambar 3.33 Hasil Perhitungan SSE 92

Gambar 3.34 Perintah Visualisasi SSE 93
Gambar 3.35 Visualisasi SSE 93
Gambar 3.36 Contoh Data Awal 95
Gambar 3.37 File Excel 100
Gambar 3.38 New Notebook 101
Gambar 3.39 Button Folder 101
Gambar 3.40 Window Dialog 102
Gambar 3.41 Import Dataset 102
Gambar 3.42 Warning 102
Gambar 3.43 Import Library 103
Gambar 3.44 Pengambilan Data dari Excel 103
Gambar 3.45 Hasil Nilai 103
Gambar 3.46 Perintah Gambar Scatter Plotting Dataset 104
Gambar 3.47 Gambar Scatter Plotting Dataset 104
Gambar 3.48 Inisialisasi Jumlah Cluster 104
Gambar 3.49 Label Hasil dari Training Data 105
Gambar 3.50 Hasil dari Training Data 105
Gambar 3.51 Perintah Melihat Pusat Cluster 105
Gambar 3.52 Hasil Pusat Cluster 105
Gambar 3.53 Perintah Visualisasi Hasil Clustering 105
Gambar 3.54 Visualisasi Hasil Clustering 106
Gambar 3.55 File Excel 112
Gambar 3.56 New Notebook 112
Gambar 3.57 Button Folder 113
Gambar 3.58 Window Dialog 113
Gambar 3.59 Import Dataset 113
Gambar 3.60 Warning 114
Gambar 3.61 Perintah Run Cell 114
Gambar 3.62 Import Library 114
Gambar 3.63 Perintah Mengambil Data 115
Gambar 3.64 Hasil Nilai 115
Gambar 3.65 Perintah Gambar Scatter Plotting Dataset 115
Gambar 3.66 Gambar Scatter Plotting Dataset 116
Gambar 3.67 Perintah Pemodelan Clustering 116
Gambar 3.68 Hasil Centroid 116
Gambar 3.69 Jarak Data dengan Centroid 117

Gambar 3.70 Perintah Visualisasi Hasil Clustering 117
Gambar 3.71 Visualisasi Hasil Clustering 118
Gambar 4.1 Core Point, Border Point, dan Outlier dengan MintPts = 4 dan ε = 1 120
Gambar 4.2 p Directly Density Reachable dengan q 121
Gambar 4.3 p Density Reachable dengan q 122
Gambar 4.4 p Density Connected dengan q Melalui Titik o 122
Gambar 4.5 Plotting Dataset 123
Gambar 4.6 Sebaran Data pada Iterasi I 125
Gambar 4.7 Sebaran Data pada Iterasi II 126
Gambar 4.8 Sebaran Data Titik Iterasi III 128
Gambar 4.9 Sebaran Data pada Iterasi IV 130
Gambar 4.10 Sebaran Titik pada Iterasi V 131
Gambar 4.11 Sebaran Titik pada Iterasi VI 133
Gambar 4.12 Sebaran Titik pada Iterasi VII 134
Gambar 4.13 Sebaran Titik pada Iterasi IX 137
Gambar 4.14 Hubungan Antartitik 138
Gambar 4.15 File Excel 139
Gambar 4.16 New Notebook 139
Gambar 4.17 Button Folder 140
Gambar 4.18 Window Dialog 140
Gambar 4.19 Import Dataset 141
Gambar 4.20 Warning 141
Gambar 4.21 Import Library 141
Gambar 4.22 Perintah Mengambil Data 142
Gambar 4.23 Hasil Nilai 142
Gambar 4.24 Perintah Gambar Scatter Plotting Dataset 142
Gambar 4.25 Gambar Scatter Plotting Dataset 143
Gambar 4.26 Training Model 143
Gambar 4.27 Perintah Melihat Label 143
Gambar 4.28 Hasil Melihat Label 143
Gambar 4.29 Perintah Melihat Jumlah Cluster 144
Gambar 4.30 Perintah Visualisasi Hasil Clustering 144
Gambar 4.31 Visualisasi Hasil Clustering 144
Gambar 5.1 Ilustrasi SOM (Kohonen, 1982) 151
Gambar 5.2 Arsitektur SOM (Kohonen, 1982) 152

Gambar 5.3 Topologi SOM (Larose, 2014) 152
Gambar 5.4 Topologi Tetangga (a) 1 Dimensi, (b) 2 Dimensi 155
Gambar 5.5 File Excel 157
Gambar 5.6 New Notebook 157
Gambar 5.7 Button Folder 158
Gambar 5.8 Window Dialog 158
Gambar 5.9 Import Dataset 158
Gambar 5.10 Warning 159
Gambar 5.11 Import Library SOM 159
Gambar 5.12 Perintah Mengambil Data 159
Gambar 5.13 Hasil Nilai 160
Gambar 5.14 Scatter Plotting Dataset 160
Gambar 5.15 Array Numpy 161
Gambar 5.16 Maksimasi dan Minimasi 163
Gambar 5.17 Klasik dan Modern Optimasi 164
Gambar 5.18 Heuristic Optimization 164
Gambar 5.19 Kawanan Burung 167
Gambar 5.20 Penentuan Burung yang Mempunyai Lokasi Terbaik 167
Gambar 5.21 Ilustrasi Pbest dan Current Position 168
Gambar 5.22 Update Ilustrasi Pbest dan Current Position 188
Gambar 5.23 Hasil Plotting 4 Percobaan 191
Gambar 5.24 Data Contoh Excel 193
Gambar 5.25 Scatter Plotting 195
Gambar 5.26 Hasil Scatter Cluster dan Centroid 204
Gambar 5.27 Hasil Scatter Dataset dengan Centroid dengan 3 Cluster 207
Gambar 5.28 Perubahan Nilai SSE dengan 40 Iterasi 208
Gambar 6.1 Metodologi Penelitian Modified K-Means 210
Gambar 6.2 Algoritma LOF 211
Gambar 6.3 Gambaran Dataset 211
Gambar 6.4 Centroid Data dengan 10 Cluster 212
Gambar 6.5 Hasil Elbow 212
Gambar 6.6 Metodologi Penelitian Pengelompokan Text Mining 214
Gambar 6.7 Hasil Nilai SSE dalam Elbow 215
Gambar 6.8 Nilai SSE untuk Uji Coba K 216
Gambar 6.9 Perbandingan Citra RGB dan l*a*b 218
Gambar 6.10 Hasil Segmentasi K-Means 218



BAB I
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 juga sering disebut sebagai ilmu tentang data karena 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, yaitu Classification/Prediction, Clustering, dan Association.

Dalam 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 (dalam bahasa Indonesia kita sebut sebagai analisis pengelompokan) merupakan sebuah analisis yang digunakan untuk mengelompokkan data yang tidak mempunyai target/label (unsupervised). Proses clustering merupakan proses pengelompokan data berdasarkan kesamaan nilai fitur/atribut. Selain mendapatkan hasil pengelompokan data, dari penerapan analisis clustering akan didapatkan titik tengah dari data (centroid) dan cluster dengan jumlah anggota terbanyak.

Beberapa contoh penerapan analisis clustering, yaitu:

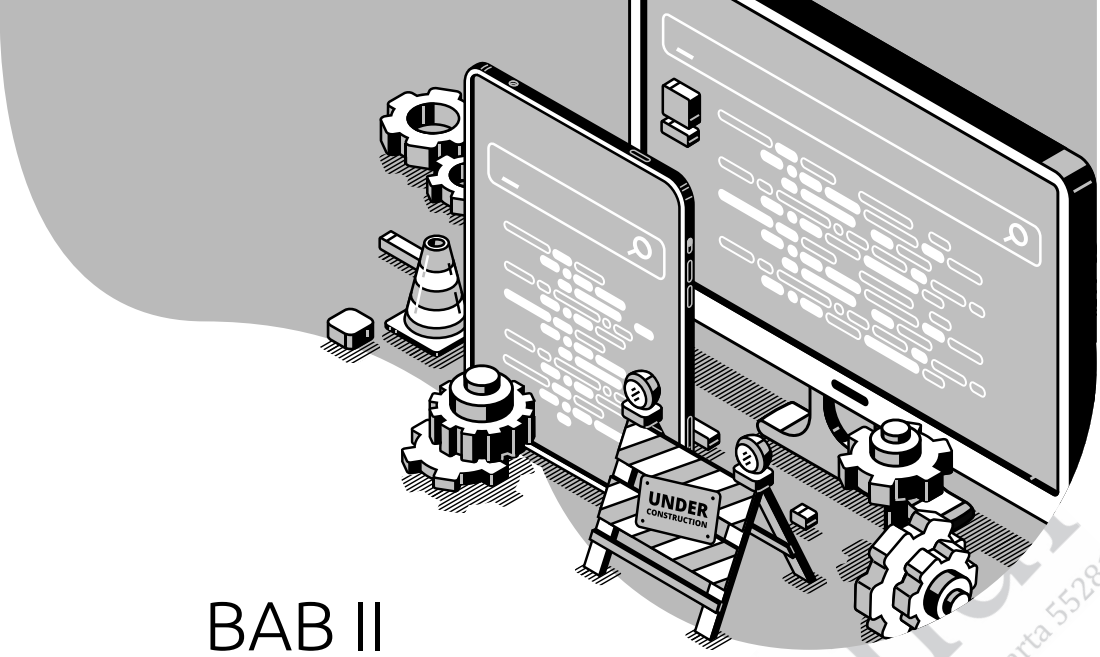
1. Dalam pengolah citra digital terdapat teknik segmentasi. Teknik ini merupakan suatu cara untuk memisahkan objek dengan latar belakangnya. Pemisahan tersebut menggunakan pengelompokan nilai berdasarkan pixel citra. Banyak penelitian yang menggunakan metode-metode data mining untuk segmentasi citra digital.

2. Untuk menempatkan unit pemadam api, 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.
3. Hampir sama dengan penentuan titik api, analisis clustering ini membuat kita menempatkan titik dapur umum atau rumah sakit umum ketika ada suatu kejadian luar biasa. Misalkan, terdapat acara pramuka (jambore) se-Jawa Timur. Pada acara tersebut akan diadakan kegiatan camping dan jumlah tenda yang disediakan oleh panitia sebanyak 30 tenda, dengan letak yang tersebar secara acak. Dengan kondisi tenda seperti itu, jumlah dapur umum yang tersedia ada tiga. Untuk itu, kita perlu menganalisis untuk mengetahui di mana 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.
4. Analisis clustering juga dapat digunakan untuk menentukan tren pasar/marketing. Misalkan, sebuah provider kartu meluncurkan produk kartu ABC. Dalam beberapa bulan, kartu ABC telah terjual sebanyak 1000 kartu perdana. Untuk mengetahui peminat kartu ABC 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 pengelompokan data customer dan dapat mengetahui kelompok customer seperti apa yang mempunyai anggota (peminat) paling banyak.

5. Hampir sama dengan poin sebelumnya (4), dengan analisis clustering kita juga dapat mengetahui peminat dari sebuah sekolah, bimbingan belajar, dan kampus sehingga dapat menentukan peminat paling banyak terdapat pada cluster apa. Diharapkan dengan mengetahui peminatnya maka arah kebijakan pemasaran bisa lebih terarah.

Dalam setiap algoritma atau metode ilmu data mining atau ilmu pendekatan (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, yaitu internal dan eksternal. Pengukuran internal dilakukan dengan 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), dan Mean Absolute Deviation (MAD). Sementara itu, 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 eksternal, dalam analisis clustering juga terdapat juga pengukuran untuk mendapatkan nilai cluster paling optimal. Untuk menghitung nilai cluster yang paling optimal digunakan dengan teknik Silhouette dan Elbow.



BAB II
METODE CLUSTERING BERBASISKAN HIERARKI

Capaian pembelajaran metode clustering berbasis hierarki adalah sebagai berikut.

1. Mampu memahami konsep dari clustering berbasiskan hierarki.
2. Mampu memahami konsep dan algoritma dari model-model clustering berbasiskan hierarki.
3. Mampu menerapkan algoritma dari model-model clustering berbasiskan hierarki untuk memecahkan masalah.

Salah satu cara melakukan cluster yang paling mudah adalah menggunakan metode hierarki. Clustering menggunakan hierarki adalah membagi atau mengelompokkan data menjadi kelompok-kelompok secara rekursif (Cohen-Addad *et al.,* 2018). Metode hierarki dapat diumpamakan, seperti bentuk pohon keluarga. Hierarki sendiri dapat dibentuk dari atas ke bawah (top down) atau dari bawah ke atas (bottom up). Hierarki yang dibentuk dari atas ke bawah biasa disebut hierarki secara divisive. Sementara hierarki yang dibentuk dari bawah ke atas biasa disebut agglomerative.

Sebelum meninjau lebih jauh mengenai teknik clustering secara hierarki, perlu diingat bahwa clustering bertujuan untuk mengelompokkan data-data yang memiliki kesamaan. Kesamaan antara data satu dan data lain ditandai dengan dekatnya jarak antara data-data tersebut. Ada berbagai cara yang dapat digunakan untuk menghitung jarak antardata, beberapa yang biasa digunakan adalah sebagai berikut.

Euclidean Distance:

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

Manhattan Distance:

$$d_{(a,b)} = \sum_{i=1}^{n}\left|a_{i+1} - a_i\right| + \left|b_{i+1} - b_i\right|$$

Minkowski Distance:

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

Dengan d merupakan jarak antara objek a dan objek b, sementara i merupakan penomoran objek.

Proses clusterisasi menggunakan agglomerative, yaitu dengan mengibaratkan pembentukan pohon silsilah dari bawah ke atas (Vijaya, Sharma, and Batra, 2019). Algoritma ini biasa juga disebut sebagai algoritma AHC (Agglomerative Hierarchy Clustering). Di sini kita awalnya akan menganggap bahwa setiap data merupakan satu cluster sehingga kalau kita memiliki 1000 data, berarti kita memiliki 1000 cluster awal. Selanjutnya, cluster-cluster yang memiliki kedekatan akan bergabung membentuk cluster baru. Cluster-cluster yang bergabung akan menurunkan jumlah cluster. Penggabungan cluster terus dilakukan sampai jumlah cluster sesuai yang diinginkan.

2.1 METODE AGGLOMERATIVE

Seperti yang sudah dijelaskan sebelumnya, clusterisasi dengan agglomerative akan menggabungkan cluster-cluster yang jaraknya dekat (Vijaya, Sharma, and Batra, 2019). Untuk menentukan cluster mana yang akan bergabung, kita harus menentukan dulu jarak antar-cluster. Ada beberapa metode yang dapat digunakan untuk menentukan jarak antar-cluster. Dalam hal ini, kita akan membahas empat cara sebagai berikut.

1. Jarak minimal (Single linkage).
2. Jarak maksimal (Complete linkage).
3. Jarak rata-rata (Average linkage).
4. Distance between centroid.

Dari setiap cara tersebut dapat dilakukan hitungan dengan beberapa perhitungan jarak, seperti Manhattan, Euclidean, dan lain sebagainya (sebagaimana telah dijelaskan sebelumnya). Keempat cara tersebut merupakan cara untuk menentukan jarak antara dua cluster, sedangkan untuk menentukan cluster mana yang bergabung, tetap berpedoman pada jarak terdekat. Maksudnya dua cluster yang jaraknya paling dekat adalah yang akan bergabung menjadi satu cluster.

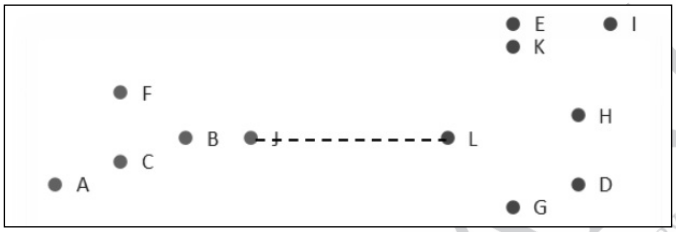
Selanjutnya kita akan bahas lebih detail masing-masing dari keempat cara penentuan jarak antar-cluster di atas.

2.1.1 Single Linkage

1. Konsep Metode

Single linkage merupakan penentuan jarak dua cluster dengan mengambil data dari masing-masing cluster yang memiliki jarak terdekat satu sama lain. Pencarian jarak menggunakan jarak minimal (single linkage) yang diilustrasikan Gambar 2.1. Gambar 2.1

versi warna dapat dilihat di laman https://bit.ly/346NSI5. Gambar 2.1 mengumpamakan adanya 2 cluster, yaitu cluster merah bata dan cluster biru. Apabila menggunakan single linkage, jarak antara cluster merah bata dan cluster biru sama dengan jarak antara data J serta data L karena data J dan data L adalah 2 data yang jaraknya paling dekat dengan cluster lain yang akan dihitung jaraknya.



Gambar 2.1 Simulasi Single Linkage

Sebagai contoh kasus, kita akan melakukan clusterisasi berdasarkan data Tabel 2.1 menggunakan agglomerative cluster menggunakan single linkage dan menjadikannya tiga cluster. Kita akan mencoba perhitungan secara manual dan mengimplementasikan dalam pemrograman bahasa Python. Untuk perhitungan jaraknya akan menggunakan rumus Manhattan distance sebagaimana yang telah dijelaskan sebelumnya.

Tabel 2.1 Contoh Dataset

Data ke	X	Y
1	1	2
2	3	4
3	2	3
4	9	2
5	8	9
6	2	6
7	8	1
8	9	5
9	8	9
10	5	8
11	8	8

Tabel 2.1 memperlihatkan bahwa kita memiliki 11 data, yang berarti sekarang kita memiliki 11 cluster awal. Langkah pertama yang harus kita lakukan adalah menghitung jarak antar-cluster. Karena setiap cluster masih terdiri dari 1 data, kita menghitung jarak pasangan data yang ada. Dengan menggunakan Manhattan distance, jarak antara cluster 1 dengan 2 dan 1 dengan 3 dapat dilihat sebagai berikut.

Jarak cluster {1} ke {2} = $|X_2-X_1|+|Y_2-Y_1|=|3-1|+|4-2|=2+2=4$

Jarak cluster {1} ke {3} = $|X_3-X_1|+|Y_3-Y_1|=|2-1|+|3-2|=1+1=2$

Sementara jarak pasangan cluster yang lain berarti merupakan jarak antara data satu dengan data yang lain dapat dilihat pada Tabel 2.2.

Tabel 2.2 Perhitungan Jarak Antardata

Cluster	1	2	3	4	5	6	7	8	9	10	11
1											
2	4										
3	2	2									
4	8	8	8								
5	14	10	12	8							
6	5	3	3	11	9						
7	8	8	8	2	8	11					
8	11	7	9	3	5	8	5				
9	14	10	12	8	0	9	8	5			
10	10	6	8	10	4	5	10	7	4		
11	13	9	11	7	1	8	7	4	1	3	

Tabel 2.2 memperlihatkan bahwa jarak terpendek yang didapatkan adalah 0, yaitu jarak antara cluster {5} dan cluster {9} maka cluster pertama yang tergabung adalah cluster {5,9}. Karena cluster {5} dan {9} tergabung menjadi 1 cluster maka total cluster kita sekarang ada 10 cluster, yaitu cluster {1}, {2}, {3}, {4}, {5,9}, {6}, {7}, {8}, {10}, {11}.

Selanjutnya, kita cari kembali jarak antar-cluster yang baru. Untuk cluster yang masih terdiri dari 1 data, cara mencari jaraknya sama seperti sebelumnya. Untuk yang sudah terdiri dari 2 data, jarak didapat dari yang minimal. Misalnya, kita ingin menentukan jarak antara cluster {1} dan cluster {5,9} serta cluster {2} dan cluster {5,9} dapat dilihat sebagai berikut.

Jarak {1} ke {5,9} = Min (jarak cluster 1 ke cluster 5, jarak cluster 1 ke data 9)
= min (14,14) (14 didapat dari Tabel 2.1)
= 14

Jarak{2} ke {5,9} = Min (jarak cluster 2 ke data 5, jarak cluster 2 ke data 9)
= min (10,10) (10 didapat dari Tabel 2.1)
= 10

Untuk jarak cluster yang lain dapat dilihat pada Tabel 2.3.

Tabel 2.3 Jarak Antar-cluster Single Linkage Iterasi ke-2

Cluster	1	2	3	4	6	7	8	10	11	5,9
1										
2	4									
3	2	2								
4	8	8	8							
6	5	3	3	11						
7	8	8	8	2	11					
8	11	7	9	3	8	5				
10	10	6	8	10	5	10	7			
11	13	9	11	7	8	7	4	3		
5,9	14	10	12	8	9	8	5	4	1	

Dari Tabel 2.3 kita dapat melihat bahwa jarak terpendek yang dihasilkan adalah 1, yaitu jarak antara cluster {5,9} dan cluster {11}.

Oleh sebab itu, cluster yang tergabung berikutnya adalah cluster {5,9} dan cluster {11} menjadi cluster {5,9,11} sehingga sekarang kita memiliki 9 cluster, yaitu cluster {1}, {2}, {3}, {4}, {6}, {7}, {8}, {10}, {5,9,11}. Selanjutnya, kita cari jarak cluster-cluster yang baru terbentuk. Misalnya, kita mencari jarak cluster {1} ke cluster {5,9,11} serta jarak cluster {2} ke cluster {5,9,11}. Dengan cara yang sama seperti sebelumnya, yaitu mencari minimalnya dan merujuk pada Tabel 2.1 sehingga kita dapatkan sebagai berikut.

Jarak cluster {1} ke cluster {5,9,11} = Min (jarak cluster {1} ke cluster {5,9}, jarak cluster {1} ke cluster {11})
= min (14,13)
= 13

Jarak cluster {2} ke cluster {5,9,11} = Min (jarak cluster {2} ke cluster {5,9}, jarak cluster {2} ke cluster {11})
= min (10,9)
= 9

Dengan cara yang sama, kita dapatkan jarak antar cluster lain sebagaimana dapat dilihat pada Tabel 2.4.

Tabel 2.4 Jarak Antar-cluster Single Linkage Iterasi ke-3

Cluster	1	2	3	4	6	7	8	10	5,9,11
1									
2	4								
3	**2**	**2**							
4	8	8	8						
6	5	3	3	11					
7	8	8	8	**2**	11				
8	11	7	9	3	8	5			
10	10	6	8	10	5	10	7		
5,9,11	13	9	11	7	8	7	4	3	

Dari Tabel 2.4 dapat dilihat bahwa jarak terpendek yang didapatkan sebesar 2, yaitu jarak antara cluster 1 ke cluster 3, cluster 2 ke cluster 3, serta cluster 4 ke cluster 7. Kemungkinan munculnya jarak yang sama seperti ini sering terjadi pada clustering yang menggunakan jarak. Oleh sebab itu, dibutuhkan metode khusus untuk pemilihan cluster mana yang akan tergabung. Salah satu metode yang dapat digunakan adalah memilih secara random. Namun, dapat juga ditentukan dengan memilih titik yang memiliki koordinat terkecil atau terbesar. Misalnya, terdapat jarak yang sama akan dipilih pasangan data yang melibatkan data dengan nomor terkecil. Berarti cluster yang tergabung sekarang adalah cluster {1} dan cluster {3} maka cluster yang terbentuk sekarang ada 8 cluster, yaitu cluster {1,3}, {2}, {4}, {6}, {7}, {8}, {10}, {5,9,11}.

Jarak antar-cluster baru yang dihasilkan dapat dilihat pada Tabel 2.5.

Tabel 2.5 Jarak Antar-cluster Single Linkage Iterasi ke-4

Cluster	2	4	6	7	8	10	5,9,11	1,3
2								
4	8							
6	3	11						
7	8	**2**	11					
8	7	3	8	5				
10	6	10	5	10	7			
5,9,11	9	7	8	7	4	3		
1,3	**2**	8	3	8	9	8	11	

Dari Tabel 2.5, kita mendapatkan 2 jarak yang sama, yaitu cluster {1,3} ke cluster {2} dan jarak cluster {4} ke cluster {7}. Seperti sebelumnya, kita akan memilih yang melibatkan cluster dengan nama terkecil. Berarti yang kita gabung berikutnya adalah cluster

{1,3} dan cluster {2}. Dengan perhitungan jarak seperti yang dijelaskan sebelumnya, dihasilkan Tabel 2.6.

Tabel 2.6 Jarak Antar-cluster Single Linkage Iterasi ke-5

Cluster	4	6	7	8	10	5,9,11	1,2,3
4							
6	11						
7	2	11					
8	3	8	5				
10	10	5	10	7			
5,9,11	7	8	7	4	3		
1,2,3	8	3	8	7	6	9	

Dengan cara yang sama kita, lakukan perulangan penggabungan cluster sampai jumlah cluster memenuhi yang kita inginkan. Hasil penggabungan cluster berikutnya dapat dilihat pada Tabel 2.7 sampai Tabel 2.9. Dari Tabel 2.9 kita sudah mendapatkan 3 cluster sesuai yang kita inginkan. Karena jumlah cluster sudah sesuai yang diinginkan, maka proses penggabungan cluster berhenti. Cluster yang terbentuk adalah cluster {1,2,3,6}, {4,7,8}, dan cluster {5,9,10,11}.

Tabel 2.7 Jarak Antar-cluster Single Linkage Iterasi ke-6

Cluster	6	8	10	5,9,11	1,2,3	4,7
6						
8	8					
10	5	7				
5,9,11	8	4	3			
1,2,3	3	7	6	9		
4,7	11	3	10	7	8	

Tabel 2.8 Jarak Antar-cluster Single Linkage Iterasi ke-7

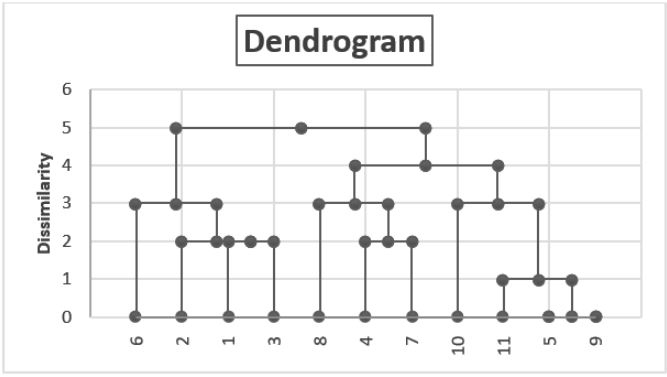
Cluster	10	5,9,11	1,2,3,6	4,7,8
10				
5,9,11	3			
1,2,3,6	5	8		
4,7,8	7	4	7	

Tabel 2.9 Jarak Antar-cluster Single Linkage Iterasi ke-8

Cluster	1,2,3,6	4,7,8	5,9,10,11
1,2,3,6			
4,7,8	7		
5,9,10,11	5	4	

Apabila dilanjutkan, cluster yang terbentuk adalah cluster {4,5,7,8,9,10,11} dan cluster {1,2,3,6} serta terakhir akan menjadi cluster yang berisi gabungan semua data.

Proses pembentukan cluster di atas dapat digambarkan dalam bentuk dendrogram, yaitu diagram yang biasa digunakan untuk merepresentasikan urutan hierarki yang terbentuk. Dendrogram menggambarkan urutan pembentukan cluster yang lebih dulu tergabung dari awal, di mana setiap data menjadi satu cluster sampai semua data menjadi satu cluster. Dendrogram dari clustering menggunakan agglomerative dengan single linkage dapat dilihat pada Gambar 2.2.



Gambar 2.2 Dendrogram Single Linkage

2. Pembuatan Program Single Linkage

Selanjutnya, kita akan mencoba mengimplementasikan single linkage menggunakan bahasa pemrograman Phyton.

a. Membuat data di excel

Langkah pertama membuat program adalah menyiapkan dataset. Dalam program, dataset dapat berupa variable (array atau list), file Excel, file csv, atau database. Dalam contoh program single linkage ini digunakan dataset berupa file Excel. Buatlah atribut beserta nilai seperti pada Gambar 2.3 dan beri nama filenya dengan '**data.xlsx**'.

	A	B
1	**x**	**y**
2	1	2
3	3	4
4	2	3
5	9	2
6	8	9
7	2	6
8	8	1
9	9	5
10	8	9
11	5	8
12	8	8

Gambar 2.3 Attribute dan Nilai pada File Excel

b. Membuat file dan class untuk membaca file Excel

Setelah membuat file Excel, kita perlu membuat code untuk membaca dataset (excel) dari ke attribute class. Pada IDE python (Spyder/PyCharm) buat new project, kemudian buat dua file baru: data.py dan main.py. Pada file data.py ketikan code seperti berikut ini.

import pandas as panda

Program 1

```
1.  class classData:
2.      __data = None
3.
4.      def __init__(self):
5.          self.__readExcel()
6.
7.      def __readExcel(self):
8.          file = panda.read_excel(open('data.xlsx','rb'))
9.          df = panda.DataFrame(file,columns=(['x','y']))
10.         dataset = df.values.tolist()
11.         self.__data = dataset
12.
13.     def getData(self):
14.         print(self.__data)
```

Pada file main.py ketikan perintah seperti program berikut ini. Code yang tertulis pada program tersebut digunakan untuk membuat objek dari class classData dan memanggil metode getData untuk menampilkan hasil pembacaan file Excel. Hasil Run file main.py dapat dilihat pada Output 1. Nilai dataset yang ada di sana sama dengan nilai file Excel yang telah dibuat sebelumnya. Hal ini menandakan program untuk pemanggilan

file Excel telah benar dan nilainya sudah dipindahkan ke attribute pada class classData.

Program 2. Main.py

```
1.  from data import classData
2.
3.  if __name__ == '__main__':
4.      obj = classData()
5.      print("nilai dataset \n", obj.getData())
```

Output 1. Hasil Run Main.py

```
nilai dataset
[[1, 2], [3, 4], [2, 3], [9, 2], [8, 9], [2, 6], [8, 1], [9,
5], [8, 9], [5, 8], [8, 8]]
```

c. Langkah berikutnya adalah membuat code untuk perhitungan jarak. Perhitungan jarak yang digunakan adalah jarak city blok. Pada file data.py tambahkan method pada program berikut ini dengan program pada file dan class yang sama pada langkah poin b.

Program 3. Method perhitugan Jarak

```
1.      def getDist(self,i,j):
2.          # i index data ke-1
3.          # j index data ke-2
4.          data = self.__data
5.          data1 = data[i]
6.          data2 = data[j]
7.          nFitur = len(data1)
8.          dist = 0
9.          for k in range (nFitur):
10.             dist = dist+(abs(data1[k]-data2[k]))
11.         return dist
```

Untuk mencoba apakah method program tersebut benar atau salah, kita perlu melakukan uji coba (testing). Uji coba akan dilakukan dengan memanggil method tersebut pada file main.py dengan nilai argument method tersebut i = 0 dan j = 1, seperti pada contoh perhitungan sebelumnya. Hasil yang diperoleh pada Run file main.py dapat dilihat pada Output 2. Hasil yang diperoleh memiliki nilainya jarak sebesar 4. Nilai tersebut sudah sesuai dengan perhitungan sebelumnya.

Program 4. Main.py

```
1.  from data import classData
2.
3.  if __name__ == '__main__':
4.      obj = classData()
5.      print("nilai dataset \n", obj.getData())
6.      print("jarak cluster 1 dan cluster 2 adalah ",
7.            obj.getDist(0, 1))
```

Output 2. Perhitungan Jarak cluster ke-1 dengan cluster ke-2

```
nilai dataset
[[1, 2], [3, 4], [2, 3], [9, 2], [8, 9], [2, 6], [8, 1], [9,
5], [8, 9], [5, 8], [8, 8]]
```

d. Setelah mendapatkan nilai jarak, tugas file data.py dan class classData telah selesai. Oleh sebab itu, kita perlu membuat file dan class baru untuk menerapkan algoritma dari metode Single Linkage dengan pencarian nilai jarak minimal. Kita create new file dengan nama cluster.py dengan code seperti pada program berikut ini.

Program 5. Code file cluster.py

```
1.  @author: Kurniawan
2.  """
3.  from data import classData
4.
5.  class classCluster:
6.      cData = classData()
7.      __clusterData = []
8.      __clusterIndex = []
9.
10.     def __init__(self):
11.         #initiliasasi data and cluster
12.         data=self.cData.getData()
13.         self.__initClusterIndex(data)
14.
15.     def getClusterData(self):
16.         return self.__clusterData
17.
18.     def getClusterIndex(self):
19.         return self.__clusterIndex
20.
21.     def __initClusterData(self,data):
22.         bar = len(data)
23.         clusters = []
24.         for i in range(bar):
25.             cluster=[]
26.             cluster.append(data[i])
27.             clusters.append(cluster)
28.         self.__clusterData=cluster
29.
30.     def __initClusterIndex(self,data):
31.         bar = len(data)
32.         for i in range(bar):
```

```
33.             cluster = []
34.             cluster.append(i+1)
35.             self.__clusterIndex.append(cluster)
36.
37.     def joinCluster(self,i,j):
38.         # i adalah index data tujuan join
39.         # j adalah index data yang dijoinkan
40.         cluster = self.__clusterIndex
41.         nlist = len(cluster[j])
42.         if nlist == 1:
43.             cluster[i].append(cluster[j][0])
44.         else:
45.             for k in range (nlist):
46.                 cluster[i].append(cluster[j][k])
47.         cluster.pop(j)
48.
49.     def joinClusterData(self,i,j):
50.         # i adalah index data tujuan join
51.         # j adalah index data yang dijoinkan
52.         cluster=self.__clusterData
53.         nlist=len(cluster[j])
54.         if nlist==1:
55.             cluster[i].append(cluster[j][0])
56.         else:
57.             for k in range (nlist):
58.                 cluster[i].append(cluster[j][k])
59.         cluster.pop(j)
60.
61.     def calMinDist(self,i,j):
62.         # i index ke-1 cluster
63.         # j index ke-2 cluster
64.         clust=self.__clusterIndex
65.         m=len(clust[i])
66.         n=len(clust[j])
```

```
67.           min0=self.cData.getDist(clust[i][0]-1,clust[j]
    [0]-1)
68.         for t in range(m):
69.             for r in range (n):
70.                 min1=self.cData.getDist(clust[i][t]-1,
71.                                         clust[j][r]-1)
72.                 if min1<min0:
73.                     min0=min1
74.         return min0
75.
76.     def findMinIndxCluster(self):
77.         cs=self.__clusterIndex
78.         n=len(cs)
79.         minVal1=self.calMinDist(0, 1)
80.         indx1=0
81.         indx2=1
82.         for i in range (1,n):
83.             for j in range(i+1,n):
84.                 minVal2=self.calMinDist(i, j)
85.                 if minVal1>minVal2:
86.                     minVal1=minVal2
87.                     indx1=i
88.                     indx2=j
89.         return indx1,indx2
```

Tujuan utama dari program berikut adalah mencari gabungan antara dua buah cluster dengan nilai terkecil (seperti pada konsep di atas). Class ini mengambil atau memanfaatkan class yang telah dibuat sebelumnya (classData). Hal ini ditunjukkan pada baris atas program untuk impor classData dan terdapat pembuatan object dari class ini pada attribute classCluster.

Cara kerja singkatnya adalah menghitung jarak cluster A dengan semua cluster yang ada jika ada cluster yang mempunyai anggota lebih dari satu nilai maka akan dicari nilai

terkecil dari semua anggota cluster. Cluster A akan bergabung dengan cluster yang mempunyai jarak terkecil dari semua perhitungan. Dalam program untuk mencari nilai terkecil dari semua cluster terdapat pada method CalMinDist( ), yang kemudian hasil dari method tersebut digunakan pada method findMinIndxCluster( ). Proses penggabungan dua cluster terdapat pada method joinClusterData(i, j).

Class ini merupakan proses utama dalam program Single linkage, tetapi pada class ini tidak terdapat perangkaian dari method-method yang telah dibuat. Perangkaian method-method-nya ada pada file dan class pada proses selanjutnya.

e. Membuat file algomerative.py dan class algometarive. Apabila proses sebelumnya mengoneksikan classData dengan classCluster, class ini mengoneksikan classCluster dengan class algomerative. Code class ini dapat dilihat pada program berikut.

Program 6. Class algomerative

```
1.  from cluster import classCluster
2.  class algomerative:
3.      cs=classCluster()
4.      tipe = 'min'
5.
6.      def __init__(self,nCluster):
7.          self.nCluster = nCluster
8.          self.iterative()
9.
10.         def iterative(self):
11.          cluster = self.cs.getClusterIndex()
12.          n = len(cluster)
13.          print("proses iterasi ke - ", 0)
```

```
14.         print(cluster)
15.         print(“”)
16.
17.         #loop all cluster
18.         for k in range (n-self.nCluster):
19.             if self.tipe == ‘min’:
20.                 i,j = self.cs.findMinIndxCluster()
21.             self.cs.joinCluster(i, j)
22.             print(“proses iterasi ke - “,k+1)
23.             print(cluster)
24.             print(“”)
```

Program di atas merupakan tempat perangkaian dari proses-proses yang telah dikerjakan pada class sebelumnya. Method iterative() merupakan method yang digunakan untuk merangkai keseluruhan proses. Untuk mencoba file yang telah kita buat benar atau tidak maka kita perlu testing pada file main.py. Rubah code yang ada pada file tersebut dengan program 7. Hasil yang terjadi dapat dilihat pada Output 3. Hasil tersebut merupakan hasil akhir dari program ini. Setiap iterasi akan menghasilkan cluster. Pada iterasi ke-0 akan mempunyai 11 cluster sesuai dengan banyak data. Setiap iterasi berjalan maka akan terdapat penggabungan cluster, contoh pada iterasi ke-1 ada penggabungan cluster ke-5 dengan cluster ke-9 hal ini ditandai dengan penggabungan pada list [5, 9]. Sampai pada akhirnya diiterasi ke-8 maka hasil akhirnya akan terbentuk 3 buah cluster, seperti yang diinginkan pada argument baris ke-5 main.py.

Program 7. Main.py

```
1.  from algomerative import algomerative
2.
3.  if __name__ == '__main__':
4.      nCluster = 3
5.      algo = algomerative(nCluster)
```

Output 3. Hasil Run file main.py

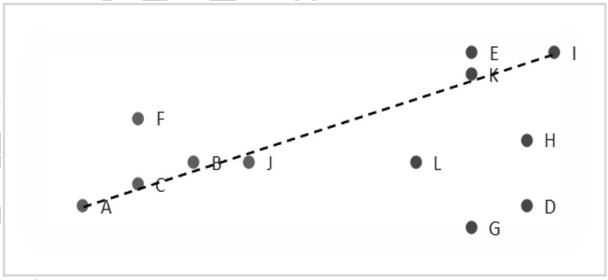
```
proses iterasi ke -  0
[[1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]]

proses iterasi ke -  1
[[1], [2], [3], [4], [5, 9], [6], [7], [8], [10], [11]]

proses iterasi ke -  2
[[1], [2], [3], [4], [5, 9, 11], [6], [7], [8], [10]]

proses iterasi ke -  3
[[1], [2, 3], [4], [5, 9, 11], [6], [7], [8], [10]]

proses iterasi ke -  4
[[1, 2, 3], [4], [5, 9, 11], [6], [7], [8], [10]]

proses iterasi ke -  5
[[1, 2, 3], [4, 7], [5, 9, 11], [6], [8], [10]]

proses iterasi ke -  6
[[1, 2, 3], [4, 7, 8], [5, 9, 11], [6], [10]]

proses iterasi ke -  7
[[1, 2, 3], [4, 7, 8], [5, 9, 11, 10], [6]]

proses iterasi ke -  8
[[1, 2, 3], [4, 7, 8, 5, 9, 11, 10], [6]]

Process finished with exit code 0
```

2.1.2 Complete Linkage

1. Konsep Metode

Complete linkage menentukan jarak dua cluster dengan mengambil data dari masing-masing cluster yang memiliki jarak terjauh satu sama lain. Pencarian jarak menggunakan jarak maksimal dapat diilustrasikan pada Gambar 2.4. Gambar 2.4 versi warna dapat dilihat di laman https://bit.ly/346NSI5. Dari Gambar 2.4 dapat dilihat bahwa terdapat dua cluster, yaitu cluster merah bata dan cluster biru. Untuk menentukan jarak antara dua cluster yang telah terbentuk tersebut adalah dengan memilih jarak data yang letaknya paling jauh satu sama lain dari kedua cluster. Dari gambar 2.4 dapat dilihat bahwa data terjauh dari kedua cluster adalah data A dari cluster merah bata dan data I dari cluster biru maka jarak antara cluster biru serta cluster merah bata sama dengan jarak data A ke data I.

Sementara itu, untuk menentukan cluster yang tergabung tetap memilih dua cluster yang memiliki jarak terpendek.



Gambar 2.4 Ilustrasi Perhitungan Jarak Maksimal Antara Cluster

Sebagai contoh, perhatikan kembali contoh data yang ada pada Tabel 2.1 beserta jarak antardata pada Tabel 2.2. Dari Tabel 2.2 dapat dilihat cluster yang tergabung terlebih dahulu adalah cluster yang memiliki jarak terpendek. Pada Langkah pertama ini sama seperti single linkage, cluster yang tergabung pertama kali adalah

cluster {5} dan {9}. Dari penggabungan pertama ini menyisakan 10 cluster, yaitu cluster {1}, {2}, {3}, {4}, {5,9}, {6}, {7}, {8}, {10}, {11}.

Selanjutnya, kita cari jarak antar-cluster. Untuk cluster yang sudah terdiri dari 2 data atau lebih maka jarak didapat dari jarak yang maksimal. Misalnya, kita ingin menentukan jarak antara cluster {1} dan cluster {5,9} serta cluster {2} dan cluster {5,9} dapat dilihat sebagai berikut.

Jarak {1} ke {5,9} = Max (jarak cluster1 ke cluster 5, jarak cluster 1 ke data 9)
= max (14,14) (14 didapat dari Tabel 2.2)
= 14

Jarak {2} ke {5,9} = Max (jarak cluster 2 ke data 5, jarak cluster 2 ke data 9)
= max (10,10) (10 didapat dari Tabel 2.2)
= 10

Bila kita perhatikan karena jarak antara cluster 5 dan cluster 9 adalah 0, jarak antara cluster lain terhadap cluster 5 dan cluster 9 pasti sama sehingga untuk jarak baru cluster-cluster yang terbentuk masih sama, seperti single linkage, yaitu sebagaimana dilihat pada Tabel 2.3. Dari Tabel 2.3, dapat dilihat bahwa jarak terpendek adalah jarak cluster {5,9} dan cluster {11} maka kita gabungkan kedua cluster tersebut sehingga sekarang kita dapat 9 cluster, yaitu cluster, {1}, {2}, {3}, {4}, {6}, {7}, {8}, {10}, {5,9,11}.

Dari 9 cluster baru yang terbentuk kita hitung kembali jarak cluster-cluster yang baru. Misalnya, kita mencari jarak cluster {1} ke cluster {5,9,11} serta jarak cluster {2} ke cluster {5,9,11}. Dengan cara yang sama seperti sebelumnya, yaitu mencari nilai maksimalnya maka kita dapatkan sebagai berikut.

Jarak cluster {1} ke cluster {5,9,11} = Max (jarak cluster {1} ke cluster {5,9}, jarak cluster {1} ke cluster {11})
= max (14,13)
= 14

Jarak cluster {2} ke cluster {5,9,11} = Max (jarak cluster {2} ke cluster {5,9}, jarak cluster {2} ke cluster {11})
= max (10,9)
= 10

Dengan cara yang sama kita hitung jarak untuk cluster yang lain di mana hasilnya dapat kita lihat sebagaimana Tabel 2.10.

Tabel 2.10 Jarak Antar-cluster Complete Linkage Iterasi ke-3

Cluster	1	2	3	4	6	7	8	10	5,9,11
1									
2	4								
3	**2**	**2**							
4	8	8	8						
6	5	3	3	11					
7	8	8	8	**2**	11				
8	11	7	9	3	8	5			
10	10	6	8	10	5	10	7		
5,9,11	14	10	12	8	9	8	5	4	

Dari Tabel 2.10, dapat kita lihat bahwa jarak terpendek adalah 2, yaitu jarak antara cluster {1} ke {3}, cluster {2} ke {3}, serta cluster {4} ke {7}. Dengan alasan yang sama seperti pembahasan single linkage, kita akan memilih cluster yang terbentuk berikutnya adalah cluster {1} dan cluster {3} maka cluster yang terbentuk sekarang ada 8 cluster, yaitu cluster {1,3}, {2}, {4}, {6}, {7}, {8}, {10}, {5,9,11}. Untuk langkah berikutnya, dengan cara yang sama kita lakukan perhitungan jarak dengan nilai maksimal dan menggabung cluster

yang jaraknya terdekat diulangi terus sampai jumlah cluster, yaitu 3. Hasil proses setiap perhitungan jarak dan penggabungan cluster dapat dilihat pada Tabel 2.11 sampai Tabel 2.16.

Tabel 2.11 Jarak Antar-cluster Complete Linkage Iterasi ke-4

Cluster	2	4	6	7	8	10	5,9,11	1,3
2								
4	8							
6	3	11						
7	8	2	11					
8	7	3	8	5				
10	6	10	5	10	7			
5,9,11	10	8	9	8	5	4		
1,3	4	8	5	8	11	10	14	

Tabel 2.12 Jarak Antar-cluster Complete Linkage Iterasi ke-5

Cluster	2	6	8	10	5,9,11	1,3	4,7
2							
6	3						
8	7	8					
10	6	5	7				
5,9,11	10	9	5	4			
1,3	4	5	11	10	14		
4,7	8	11	5	10	8	8	

Tabel 2.13 Jarak Antar-cluster Complete Linkage Iterasi ke-6

Cluster	8	10	5,9,11	1,3	4,7	2,6
8						
10	7					
5,9,11	5	4				
1,3	11	10	14			
4,7	5	10	8	8		
2,6	8	6	10	5	11	

Tabel 2.14 Jarak Antar-cluster Complete Linkage Iterasi ke-7

Cluster	8	1,3	4,7	2,6	5,9,10,11
8					
1,3	11				
4,7	5	8			
2,6	8	5	11		
5,9,10,11	7	14	10	10	

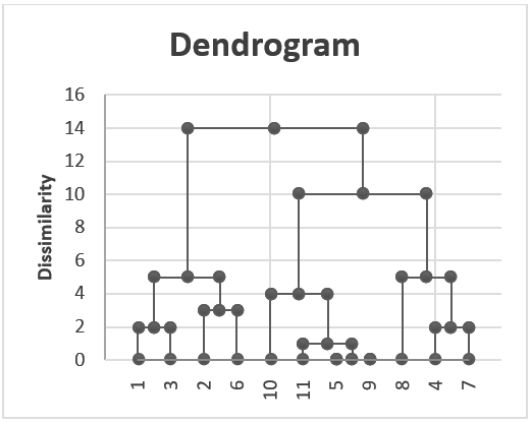
Tabel 2.15 Jarak Antar-cluster Complete Linkage Iterasi ke-8

Cluster	8	4,7	5,9,10,11	1,2,3,6
8				
4,7	5			
5,9,10,11	7	10		
1,2,3,6	11	11	14	

Tabel 2.16 Jarak Antar-cluster Complete Linkage Iterasi ke-9

Cluster	5,9,10,11	1,2,3,6	4,7,8
5,9,10,11			
1,2,3,6	14		
4,7,8	10	11	

Dari Tabel 2.16 telah kita dapatkan cluster yang terbentuk. Cluster akhir yang terbentuk dari clustering menggunakan complete linkage adalah cluster {5,9,10,11}, {1,2,3,6}, dan cluster {4,7,8}. Pada kasus ini, hasil cluster yang dihasilkan oleh single linkage dan complete linkage adalah sama. Walaupun hasilnya sama, tetapi proses pembentukan cluster yang tergabung berbeda. Hal ini dapat dilihat dari dendrogram complete linkage pada Gambar 2.5.



Gambar 2.5 Dendrogram Complete Linkage

2. Pembuatan Program Complete Linkage

Pada contoh ini, untuk membuat program complete linkage, akan menggunakan Google Colab. Langkah-langkah yang dapat digunakan, antara lain sebagai berikut.

a. Buat file Excel dengan attribute (kolom) dan nilai data (baris) seperti gambar berikut ini dan simpan dengan format .xlsx.

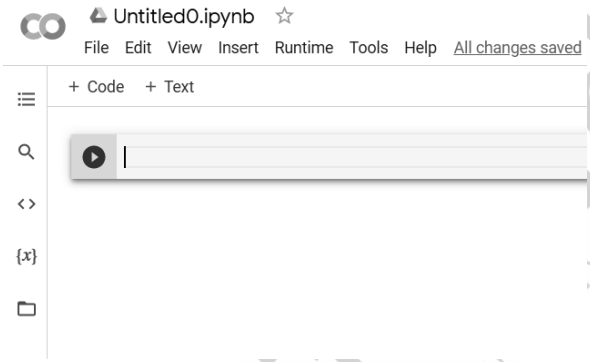
	A	B
1	**x**	**y**
2	4	5
3	1	6
4	2	7
5	6	8
6	4	8
7	5	4
8	2	3
9	3	7
10	3	3
11	5	9
12	3	4
13	3	5
14	2	8
15	4	7
16	2	1
17	5	6
18	5	5

Gambar 2.6 File Excel

b. Buka Google Colab dengan alamat:

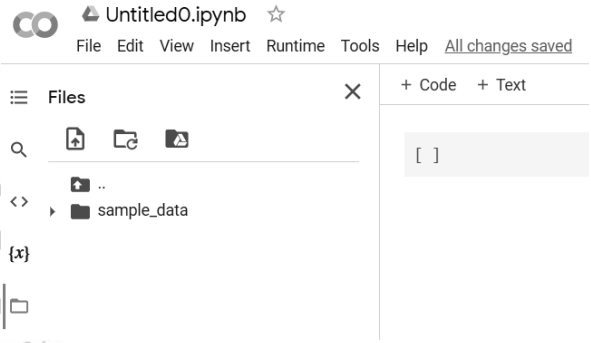
   https://colab.research.google.com/drive/1kqyxTlSWLfH7en-fEUFUbcrlQp8Td7M8C?usp=sharing.

c. Masuk pada menu file, kemudian pilih new notebook maka akan tampil seperti pada gambar berikut ini.



Gambar 2.7 New Notebook

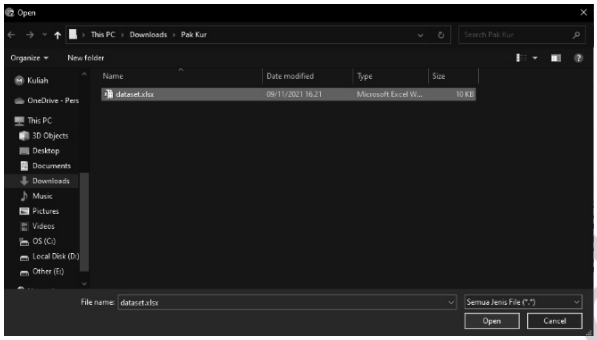
d. Tekan button folder seperti pada gambar berikut ini.



Gambar 2.8 Button Folder

e. Upload file Excel yang telah dibuat pada point pertama dengan menekan button klik pada tombol .

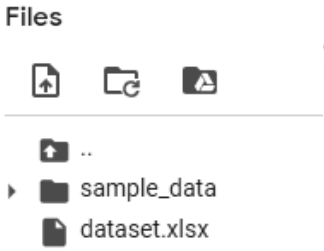
Setelah menekan button maka akan muncul window dialog.



Gambar 2.9 Window Dialog

Pilih file yang telah dibuat sebelumnya dan tekan button open.

f. Klik open dan pastikan sudah ter-import.

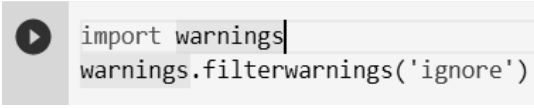


Gambar 2.10 Import Dataset

g. Pada baris pertama kode ketikan perintah:

Pertama tulis kode untuk menghilangkan warning pada cell:

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



Gambar 2.11 Warning

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

h. Tuliskan code pada cell selanjutnya dengan perintah:

!pip install -U scikit-learn

i. Import library-library yang dibutuhkan untuk program FCM ini.

```
import pandas as pd
import matplotlib.pyplot as plt
import scipy.cluster.hierarchy as shc
from sklearn.cluster import AgglomerativeClustering
```

Gambar 2.12 Import Library

Library pandas digunakan untuk mengambil atau mengekstrak nilai dari file excel atau csv ke dalam Data Frame Python. Library matplolib dan seaborn digunakan untuk menampilkan plotting dan scatter data. Library scipy hierarchy digunakan untuk menampilkan dendrogram. Library Agglomerative Clustering untuk clustering dengan metode Agglomerative.

j. Pada cell selanjutnya ketikan perintah untuk mengambil data dari file Excel, seperti pada code berikut ini.

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

Gambar 2.13 Perintah Mengambil Data

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

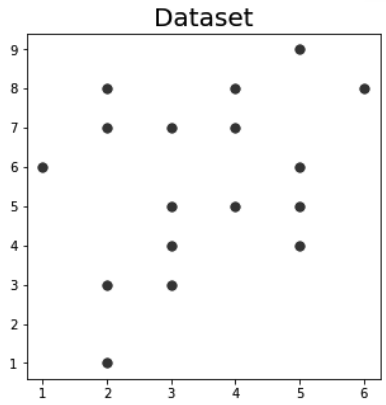
	x	y
0	4	5
1	1	6
2	2	7
3	6	8
4	4	8
5	5	4
6	2	3

Gambar 2.14 Hasil Nilai

k. Untuk mendapatkan gambaran dari data maka kita perlu membuat scatter plotting dari dataset yang telah berhasil diambil nilainya pada cell sebelumnya. Ketikan perintah berikut 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 2.15 Perintah Gambar Scatter Plotting Dataset



Gambar 2.16 Gambar Scatter Plotting Dataset

l. Selanjutnya melakukan training model terhadap data dengan mendefinisikan jumlah cluster yang kita inginkan pada parameter **n_cluster**. Untuk menggunakan metode Complete Linkage, didefinisikan pada parameter **linkage.**

```
complete_linkage = AgglomerativeClustering(n_clusters = 3, linkage = 'complete')
complete_linkage.fit(df)
```

Gambar 2.17 Training Model

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

m. Untuk melihat label yang terbentuk pada training complete linkage adalah sebagai berikut.

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

Gambar 2.18 Training Complete Linkage

Hasilnya adalah:

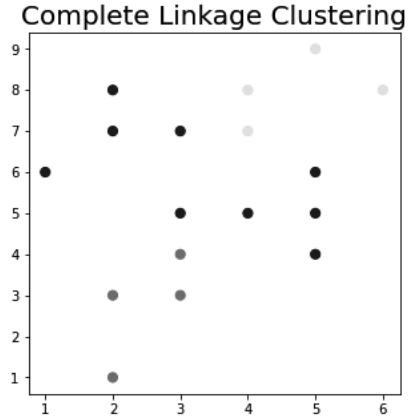
```
array([0, 0, 0, 2, 2, 0, 1, 0, 1, 2, 1, 0, 0, 2, 1, 0, 0])
```

n. 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.title('Complete Linkage Clustering',fontsize=20)
plt.show()
```

Gambar 2.19 Perintah Visualisasi Hasil Clustering

Hasil visualisasi data hasil clustering adalah sebagai berikut.



Gambar 2.20 Visualisasi Data Hasil Clustering

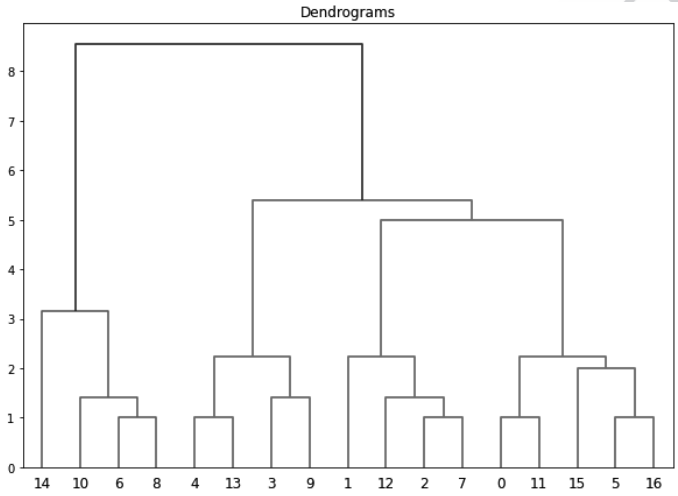
Dapat dilihat bahwa metode Complete Linkage membentuk 3 cluster.

o. Langkah terakhir adalah visualisasi dendrogram untuk setiap cluster dengan kode sebagai berikut.

```
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dend = shc.dendrogram(shc.linkage(df, method='complete'))
```

Gambar 2.21 Perintah Visualisasi Dendrogram

Hasil visualisasinya adalah:

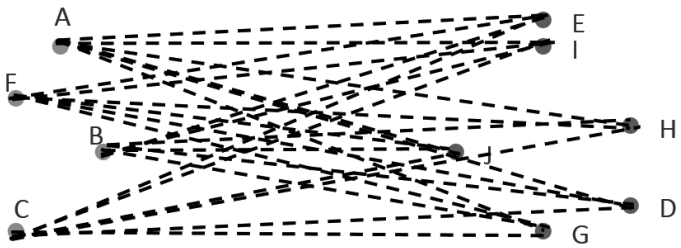


Gambar 2.22 Dendrogram Complete Linkage

2.1.3 Average Linkage

1. Konsep Metode

Average linkage menentukan jarak antar-cluster dengan cara mencari rata-rata jarak antardata-data dalam cluster. Misalnya, terdapat cluster X berisi data A dan data B, cluster Y berisi data C serta data D. Oleh sebab itu, jarak antara cluster X dan cluster Y adalah jarak data (AC+AD+BC+BD)/4. Ilustrasi average linkage dapat dilihat pada Gambar 2.23.



Gambar 2.23 Ilustrasi Average Linkage

Kembali kita lihat contoh sebelumnya. Sekarang kita akan menyelesaikan kasusnya menggunakan average linkage. Seperti sebelumnya mula-mula kita memiliki 11 cluster awal. Lalu karena cluster {9} dan cluster {5} memiliki jarak 0 maka cluster {9} dan cluster {5} bergabung terlebih dahulu. Selanjutnya, kita memiliki 10 cluster yang jarak masing-masing cluster dapat dilihat pada Tabel 2.2, kemudian karena jarak cluster {5} dan {9} adalah 0 makanya jaraknya terhadap cluster lain adalah sama, yaitu sebagaimana yang dapat dilihat pada Tabel 2.3. Dari Tabel 2.3, cluster berikutnya yang terbentuk adalah cluster {5,9,11}. Selanjutnya, kita hitung jarak cluster baru yang terbentuk dengan cluster lain. Misalnya, kita akan menghitung jarak antara cluster {5,9,11} dengan cluster {1} dan jarak antara cluster {5,9,11} dengan cluster 2.

Jarak cluster {1} ke cluster {5,9,11} = (jarak cluster {1} ke cluster {5} + jarak cluster {1} ke cluster {9} + jarak cluster {1} ke cluster {11})/3)
= (14+14+13)/3
= 13,67

Jarak cluster {2} ke cluster {5,9,11} = (jarak cluster {2} ke cluster {5} + jarak cluster {2} ke cluster {9} + jarak cluster {2} ke cluster {11})/3)
= (10+10+9)/3
= 9,67

Untuk jarak cluster yang lain, dihitung menggunakan cara yang sama. Perhitungan jarak dan hasil clustering menggunakan average linkage dapat dilihat pada Tabel 2.17 sampai Tabel 2.23.

Tabel 2.17 Jarak Antar-cluster Average Linkage Iterasi ke-3

Cluster	1	2	3		4	6	7	8	10	5,9,11
1										
2	4									
3	2	2								
4	8	8	8							
6	5	3	3		11					
7	8	8	8		2	11				
8	11	7	9		3	8	5			
10	10	6	8		10	5	10	7		
5,9,11	13. 7	9.7	11.7		7.7	8.7	7.7	4.7	3.7	

Tabel 2.18 Jarak Antar-cluster Average Linkage Iterasi ke-4

Cluster	2	4	6	7	8	10	5,9,11	1,3
2								
4	8							
6	3	11						
7	8	2	11					
8	7	3	8	5				
10	6	10	5	10	7			
5,9,11	9.7	7.7	8.7	7.7	4.7	3.7		
1,3	3	5	4	8	10	9	12.7	

Tabel 2.19 Jarak Antar-cluster Average Linkage Iterasi ke-5

Cluster	2	6	8	10	5,9,11	1,3	4,7
2							
6	3						
8	7	8					
10	6	5	7				
5,9,11	9.7	8.7	4.7	3.7			
1,3	3	4	10	9	12.7		
4,7	8	11	4	10	7.7	8	

Tabel 2.20 Jarak Antar-cluster Average Linkage Iterasi ke-6

Cluster	6	8	10	5,9,11	4,7	1,2,3
6						
8	8					
10	5	7				
5,9,11	8.67	4.67	3.67			
4,7	11	4	10	7.67		
1,2,3	3.67	9	8	11.67	8	

Tabel 2.21 Jarak Antar-cluster Average Linkage Iterasi ke-7

Cluster	8	10	5,9,11	4,7	1,2,3,6
8					
10	7				
5,9,11	4.67	3.67			
4,7	4	10	7. 67		
1,2,3,6	8.75	7.25	10.92	8.75	

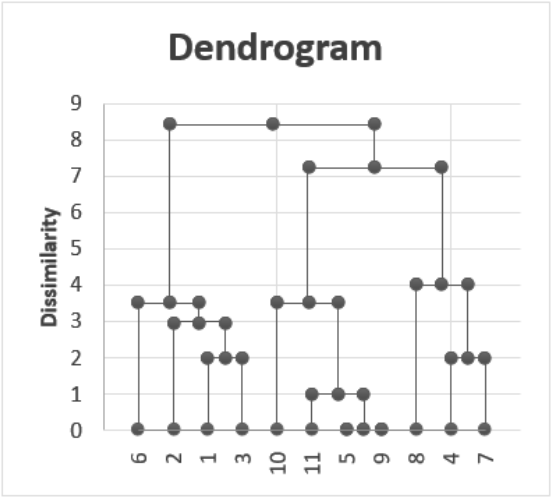
Tabel 2.22 Jarak Antar-cluster Average Linkage Iterasi ke-8

Cluster	8	4,7	1,2,3,6	5,9,10,11
8				
4,7	4			
1,2,3,6	8.75	8.75		
5,9,10,11	5.25	8.25	10	

Tabel 2.23 Jarak Antar-cluster Average Linkage Iterasi ke-9

Cluster	1,2,3,6	5,9,10,11	4,7,8
1,2,3,6			
5,9,10,11	10		
4,7,8	8.75	7.25	

Hasil clustering menggunakan agglomerative dengan average linkage menghasilkan 3 cluster, yaitu cluster {1,2,3,6}, cluster {5,9,10,11}, serta cluster {4,7,8}. Pada kasus ini, cluster yang dihasilkan sama dengan cluster dengan metode sebelumnya, tetapi sekali lagi berbeda dari urutan pembentukannya atau bisa dilihat dari perbedaan dendrogram yang dihasilkan pada Gambar 2.24.



Gambar 2.24 Dendrogram Average Linkage

2. Pembuatan Program Average Linkage

Untuk membuat program Average Linkage, pada contoh ini akan menggunakan Google Colab. Langkah-langkah yang dapat digunakan:

a. Buat file Excel dengan attribute (kolom) dan nilai data (baris) seperti pada Gambar 2.25, serta simpan dengan format .xlsx.

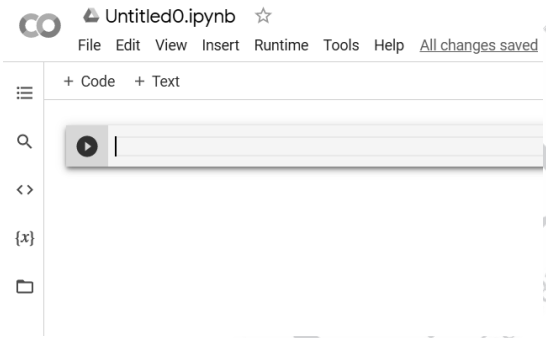
	A	B
1	x	y
2	4	5
3	1	6
4	2	7
5	6	8
6	4	8
7	5	4
8	2	3
9	3	7
10	3	3
11	5	9
12	3	4
13	3	5
14	2	8
15	4	7
16	2	1
17	5	6
18	5	5

Gambar 2.25 File Excel

b. Buka Google Colab dengan alamat:

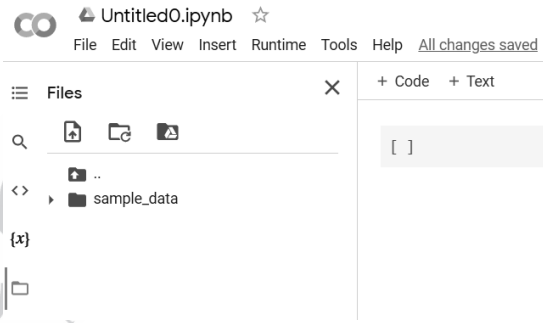
https://colab.research.google.com/drive/1kqyxTlSWLfH7en-fEUFUbcrlQp8Td7M8C?usp=sharing.

c. Masuk pada menu file, kemudian pilih new notebook akan tampil seperti pada Gambar 2.26.



Gambar 2.26 New Notebook

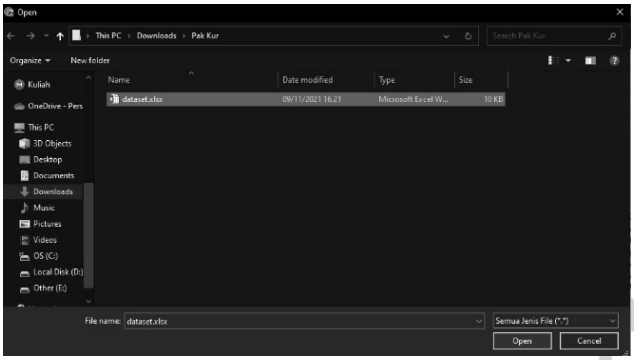
d. Tekan button folder seperti pada Gambar 2.27.



Gambar 2.27 Button Folder

e. Upload file Excel yang telah dibuat pada point pertama dengan menekan button klik pada tombol .

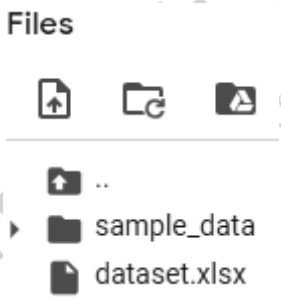
Setelah menekan button maka akan muncul window dialog sebagaimana Gambar 2.28.



Gambar 2.28 Window Dialog

Pilih file yang telah dibuat sebelumnya dan tekan button open.

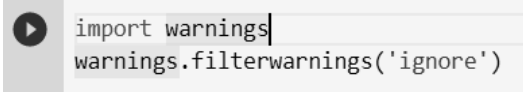
f. Klik open dan pastikan sudah ter-import, seperti pada Gambar 2.29.



Gambar 2.29 Import Dataset

g. Pada baris pertama kode ketikan perintah berikut.

Pertama tulis kode untuk menghilangkan warning pada cell:
import warnings
warnings.filterwarnings('ignore')



Gambar 2.30 Warning

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

h. Tuliskan code pada cell selanjutnya dengan perintah:

!pip install -U scikit-learn

i. Import library-library yang dibutuhkan untuk program Average linkage ini.

```
from scipy.cluster.hierarchy import dendrogram, linkage
from matplotlib import pyplot as plt
import pandas as pd
from sklearn.cluster import AgglomerativeClustering
from seaborn import scatterplot as scatter
```

Gambar 2.31 Import Library

Library pandas digunakan untuk mengambil atau mengekstrak nilai dari file Excel atau csv ke dalam Data Frame Python. Library matplolib dan seaborn digunakan untuk menampilkan plotting serta scatter data. Library scipy hierarchy digunakan untuk menampilkan dendrogram. Library Agglomerative Clustering untuk clustering dengan metode Agglomerative.

j. Pada cell selanjutnya ketikan perintah untuk mengambil data dari file Excel, seperti pada code berikut ini.

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

Gambar 2.32 Perintah Mengambil Data

Setelah di-run akan muncul hasil nilai seperti pada nilai yang dibuat di Excel.

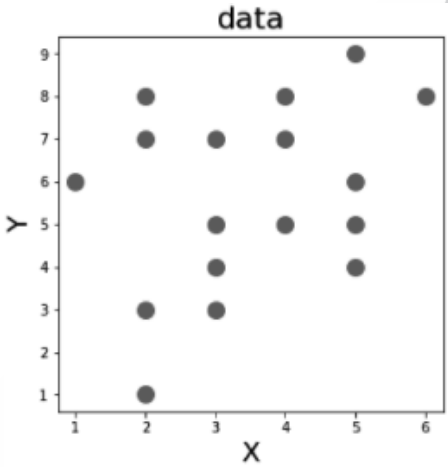
	x	y
0	4	5
1	1	6
2	2	7
3	6	8
4	4	8
5	5	4
6	2	3

Gambar 2.33 Hasil Nilai

k. Untuk mendapatkan gambaran dari data maka kita perlu membuat scatter plotting dari dataset yang telah berhasil diambil nilainya pada cell sebelumnya. Ketikan perintah berikut ini untuk mendapatkan gambar scatter plotting dataset.

```
plt.figure(figsize=(5,5))
scatter(df.iloc[:,0].values, df.iloc[:,1].values, marker='o', s=200)
plt.xlabel('X', fontsize=20)
plt.ylabel('Y', fontsize=20)
plt.title('data', fontsize=22)
plt.show()
```

Gambar 2.34 Perintah Gambar Scatter Plotting Dataset



Gambar 2.35 Gambar Scatter Plotting Dataset

l. Selanjutnya melakukan training model terhadap data dengan mendefinisikan jumlah cluster yang kita inginkan pada parameter **n_cluster**. Untuk menggunakan metode Average linkage, didefinisikan pada parameter **linkage**.

```
cluster = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='average')
cluster.fit(df)
```

Gambar 2.36 Training Model Average Linkage

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

m. Untuk melihat label yang terbentuk pada training Average Linkage adalah sebagai berikut.

```
cluster.labels_
```

Gambar 2.37 Perintah Melihat Label

Hasilnya adalah:

```
array([0, 1, 1, 2, 1, 0, 0, 1, 0, 2, 0, 0, 1, 1, 0, 0, 0])
```

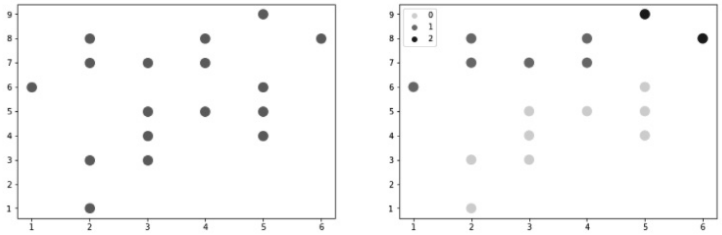
Gambar 2.38 Hasil Average Linkage

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

```
f, axes = plt.subplots(1, 2, figsize=(16,5))
scatter(df.iloc[:,0].values, df.iloc[:,1].values, ax=axes[0], s=200)
scatter(df.iloc[:,0].values, df.iloc[:,1].values, ax=axes[1], hue=cluster.labels_, s=200)
plt.show()
```

Gambar 2.39 Perintah Visualisasi Hasil Clustering

Hasil visualisasi data hasil clustering adalah sebagai berikut.



Gambar 2.40 Visualisasi Hasil Clustering

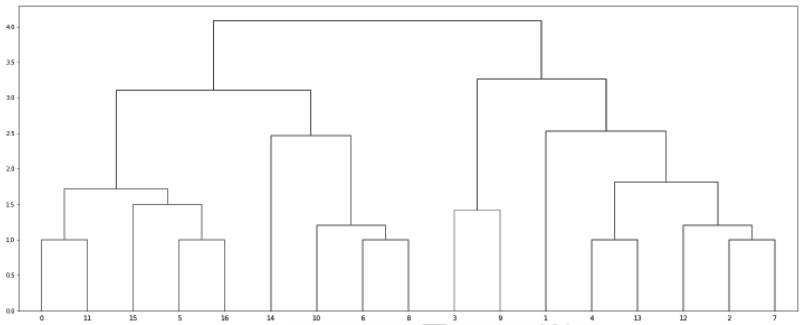
Grafik sebelah kiri merupakan sebaran awal dan grafik sebelah kanan merupakan hasil cluster yang membentuk 3 cluster.

o. Langkah terakhir adalah visualisasi dendrogram untuk setiap cluster dengan kode sebagai berikut.

```
Z = linkage(df, 'average')
# Z
fig = plt.figure(figsize=(25, 10))
dn = dendrogram(Z)
plt.show()
```

Gambar 2.41 Perintah Visualisasi Dendrogram Average Linkage

Hasil dendrogram dapat dilihat sebagaimana Gambar 2.42.

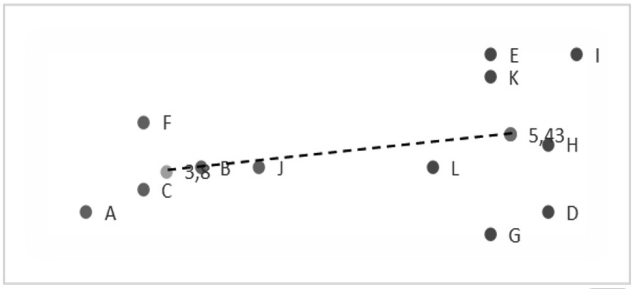


Gambar 2.42 Dendrogram Average Linkage

2.1.4 Distance Between Centroid

1. Konsep Metode

Metode berikutnya, yaitu penghitungan jarak cluster menggunakan metode Distance Between Centroid atau jarak antar-centroid. Dalam hal ini artinya kita harus menentukan letak centroid setiap cluster sebelum melakukan perhitungan jarak antar-cluster. Perhitungan jarak menggunakan centroid dapat diilustrasikan pada Gambar 2.43. Centroid dari cluster merah bata dilambangkan dengan titik warna orange, sedangkan centroid cluster biru dilambangkan dengan titik warna merah maka jarak antara cluster merah bata dan cluster biru sama dengan jarak titik orange dan titik merah. Gambar 2.43 versi warna dapat dilihar di laman https://bit.ly/346NSI5.



Gambar 2.43 Ilustrasi Perhitungan Jarak Antar-cluster Menggunakan Centroid

Masih dengan studi kasus sebelumnya, kita akan membuat tiga cluster dari 11 data menggunakan agglomerative dengan pencarian jarak menggunakan jarak antar-centroid. Sama seperti sebelumnya, mula-mula setiap data menjadi satu cluster. Karena setiap cluster masih terdiri dari satu data, centroid setiap cluster adalah data itu sendiri. Untuk perhitungan antar-cluster, hasilnya masih sama, seperti Tabel 2.2. Dari Tabel 2.2 yang memiliki nilai terkecil adalah jarak cluster {5} dan cluster {9}. Karena jarak keduanya 0, centroid dari cluster baru {5,9} sama dengan data itu sendiri, jarak antara cluster {5,9} dengan cluster lain dapat dilihat sebagaimana Tabel 2.3. Dari Tabel 2.3, cluster berikutnya yang tergabung adalah cluster {5,9,11}. Karena di sini terdapat 3 data dalam 1 cluster, kita bisa cari centroid-nya. Untuk mencari centroid, kita harus melihat koordinat setiap data yang ada dalam cluster tersebut. Artinya kita harus melihat Tabel 2.1, untuk menentukan centroid cluster. Centroid cluster {5,9,11} dapat dicari dengan cara berikut.

Centroid {5,9,11} variable x = (8+8+8)/3 = 8
Centroid {5,9,11} variable y = (9+9+8)/3 = 8.67

Setelah diketahui centroid dari cluster {5,9,11}, baru kita dapat menghitung jarak cluster tersebut dengan cluster yang lain. Misalnya, kita ingin mencari jarak cluster {5,9,11} dengan cluster {1}

dan jarak antara cluster {5,9,11} dengan cluster {2}. Penghitungan sama seperti penghitungan jarak yang lain. Dalam kasus ini, yaitu menggunakan Manhattan Distance. Karena perhitungan dilakukan berdasarkan jarak yang melibatkan koordinat, Tabel 2.1 tetap harus diperhatikan.

Jarak cluster {1} ke cluster {5,9,11} = |1-8|+|2-8.67|
= 7+6.67
= 13,67

Jarak cluster {2} ke cluster {5,9,11} = |3-8|+|4-8.67|
= 5+4.67
= 9,67

Untuk jarak antar-cluster lain dapat dilihat pada Tabel 2.24.

Tabel 2.24 Jarak Antar-cluster Menggunakan Jarak Antar-centroid Iterasi ke-3

Cluster	1	2	3	4	6	7	8	10	5,9,11
1									
2	4								
3	2	2							
4	8	8	8						
6	5	3	3	11					
7	8	8	8	2	11				
8	11	7	9	3	8	5			
10	10	6	8	10	5	10	7		
5,9,11	13.7	9.7	11.7	7.7	8.7	7.7	4.7	3.7	

Dari Tabel 2.24, cluster yang tergabung berikutnya adalah cluster {1,3}. Selanjutnya, kita cari centroid dari cluster {1,3} menggunakan cara sebelumnya.

Centroid {1,3} variable x = (1+2)/2 = 1.5
Centroid {1,3} variable y = (2+3)/2 = 2.5

Misalnya, kita akan menghitung jarak cluster {1,3} dan cluster {5,9,11}, jarak kedua cluster tersebut merupakan jarak centroid dari kedua cluster, yaitu:

Jarak cluster {1,3} ke cluster {5,9,11} = |1.5-8|+|2.5-8.67|
= 6.5+6.17
= 12.67

Berikutnya kembali menghitung jarak setiap cluster yang ada terhadap centroid tersebut. Begitu seterusnya sampai cluster-cluster saling tergabung dan membentuk 3 cluster sebagaimana yang diinginkan. Proses pembentukan cluster dapat dilihat pada Tabel 2.25 sampai Tabel 2.30 berikut.

Tabel 2.25 Jarak Antar-cluster Menggunakan Jarak Antar-centroid Iterasi ke-4

Cluster	2	4	6	7	8	10	5,9,11	1,3
2								
4	8							
6	3	11						
7	8	2	11					
8	7	3	8	5				
10	6	10	5	10	7			
5,9,11	9.7	7.7	8.7	7.7	4.7	3.7		
1,3	3	8	4	8	10	9	12.7	

Tabel 2.26 Jarak Antar-cluster Menggunakan Jarak Antar-centroid Iterasi ke-5

Cluster	2	6	8	10	5,9,11	1,3	4,7
2							
6	3						
8	7	8					
10	6	5	7				
5,9,11	9.67	8.67	4.67	3.67			
1,3	3	4	10	9	12.67		
4,7	8	11	4	10	7.67	8	

Tabel 2.27 Jarak Antar-cluster Menggunakan Jarak Antar-centroid Iterasi ke-6

Cluster	6	8	10	5,9,11	4,7	1,2,3
6						
8	8					
10	5	7				
5,9,11	8.67	4.67	3.67			
4,7	11	4	10	7.67		
1,2,3	3	9	8	11.67	8	

Tabel 2.28 Jarak Antar-cluster Menggunakan Jarak Antar-centroid Iterasi ke-7

Cluster	8	10	5,9,11	4,7	1,2,3,6
8					
10	7				
5,9,11	4.67	3.67			
4,7	4	10	7.67		
1,2,3,6	8.25	7.25	10.92	8.75	

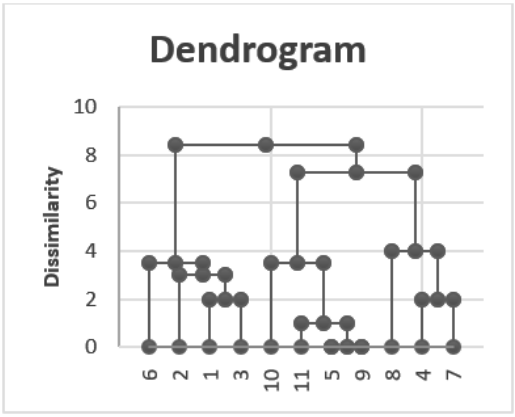
Tabel 2.29 Jarak Antar-cluster Menggunakan Jarak Antar-centroid Iterasi ke-8

Cluster	8	4,7	1,2,3,6	5,9,10,11
8				
4,7	4			
1,2,3,6	8.25	8.75		
5,9,10,11	5.25	8.25	10	

Tabel 2.30 Jarak Antar-cluster Menggunakan Jarak Antar-centroid Iterasi ke-9

Cluster	1,2,3,6	5,9,10,11	4,7,8
1,2,3,6			
5,9,10,11	10		
4,7,8	7.75	7.25	

Hasil clustering menggunakan agglomerative dengan jarak antar-centroid menghasilkan 3 cluster, yaitu cluster {1,2,3,6}, cluster {5,9,10,11}, serta cluster {4,7,8}. Dendrogram hasil cluster menggunakan Distance Between Centroid dapat dilihat pada Gambar 2.44.



Gambar 2.44 Dendrogram Distance Between Centroid

2. Contoh Implementasi Clustering pada Python

Berikut implementasi clustering menggunakan Distance Between Centroid (Agglomerative) pada Phyton.

Pembuatan Program Distance Between Centroid (Agglomerative)

Dalam contoh ini, untuk membuat program Distance Between Centroid menggunakan Google Colab. Langkah-langkahnya adalah sebagai berikut.

a. Buat file Excel dengan attribute (kolom) dan nilai data (baris), seperti gambar berikut ini serta simpan dengan format .xlsx.

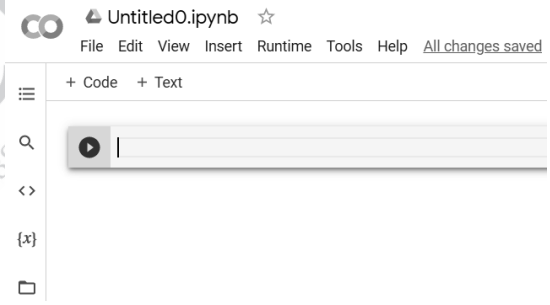
	A	B
1	x	y
2	4	5
3	1	6
4	2	7
5	6	8
6	4	8
7	5	4
8	2	3
9	3	7
10	3	3
11	5	9
12	3	4
13	3	5
14	2	8
15	4	7
16	2	1
17	5	6
18	5	5

Gambar 2.45 File Excel

b. Buka Google Colab dengan alamat:

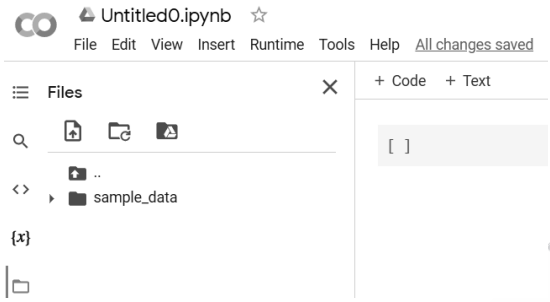
https://colab.research.google.com

c. Masuk pada menu file, pilih new notebook dan akan tampil seperti gambar berikut ini.



Gambar 2.46 New Notebook

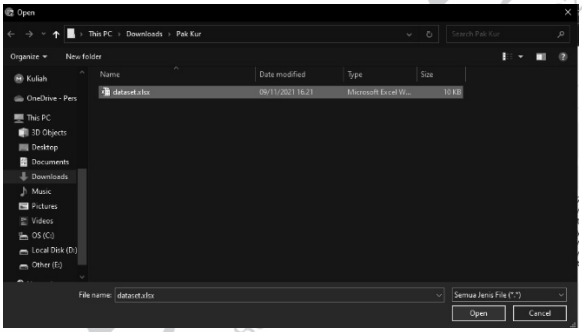
d. Tekan button folder seperti pada gambar berikut ini.



Gambar 2.47 Button Folder

e. Upload file Excel yang telah dibuat pada point pertama dengan menekan button klik pada tombol .

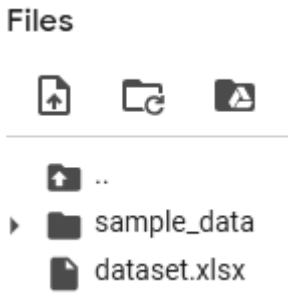
Setelah menekan button akan muncul window dialog



Gambar 2.48 Window Dialog

Pilih file yang telah dibuat sebelumnya dan tekan button open.

f. Klik open dan pastikan sudah ter-import.



Gambar 2.49 Import Dataset

g. Pada baris pertama kode ketikan perintah.

Pertama tulis kode untuk menghilangkan warning pada cell:
import warnings
warnings.filterwarnings('ignore')

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

Gambar 2.50 Warning

Setelah menuliskan perintah tersebut kita akan melakukan run program dengan menekan tombol run (tombol play).

h. Import library-library yang dibutuhkan untuk program Agglomerative Clustering ini.

```
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import AgglomerativeClustering
from seaborn import scatterplot as scatter
import scipy.cluster.hierarchy as sch
```

Gambar 2.51 Import Library

Library pandas digunakan untuk mengambil atau mengekstrak nilai dari file Excel atau csv ke dalam Data Frame Python. Library matplolib dan seaborn digunakan untuk menampilkan plotting dan scatter data. Library skearn digunakan untuk menghitung plotting.

i. Pada cell selanjutnya ketikan perintah untuk mengambil data dari file Excel, seperti pada code berikut ini.

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

Gambar 2.52 Perintah Mengambil Data

Setelah di-run akan muncul hasil nilai seperti pada nilai yang dibuat di Excel.

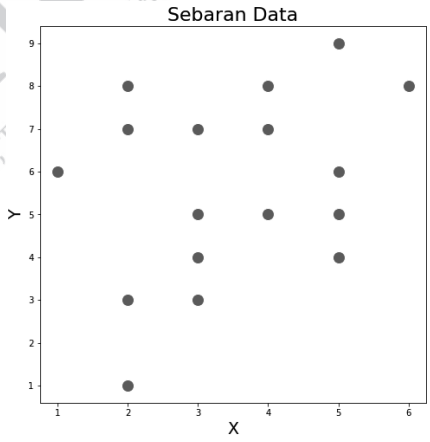
	x	y
0	4	5
1	1	6
2	2	7
3	6	8
4	4	8
5	5	4
6	2	3

Gambar 2.53 Hasil Nilai

j. Untuk mendapatkan gambaran dari data kita perlu membuat scatter plotting dari dataset yang telah berhasil diambil nilainya pada cell sebelumnya. Ketikan perintah berikut ini untuk mendapatkan gambar scatter plotting dataset.

```
plt.figure(figsize=(8,8))
scatter(df.iloc[:,0].values, df.iloc[:,1].values, marker='o', s=200)
plt.xlabel('X', fontsize=19)
plt.ylabel('Y', fontsize=19)
plt.title('Sebaran Data', fontsize=22)
plt.show()
```

Gambar 2.54 Perintah Gambar Scatter Plotting Dataset

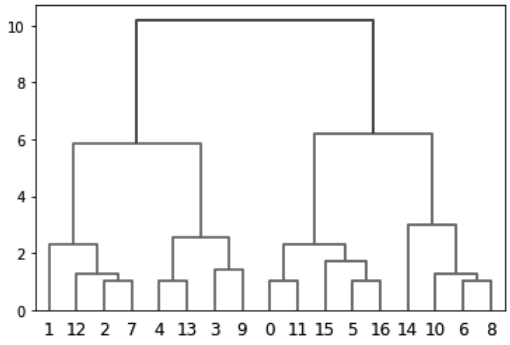


Gambar 2.55 Gambar Scatter Plotting Dataset

k. Selanjutnya untuk melihat plot dendrogram ketik perintah berikut ini.

```
dendrogram = sch.dendrogram(sch.linkage(df.values, method='ward'))
```

Gambar 2.56 Perintah Plot Dendrogram



Gambar 2.57 Dendrogram Distance Between Centroid

l. Buat sebuah model clustering dengan metode Agglomerative dengan membuat code berikut.

```
model = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward')
model.fit(df.values)
```

Gambar 2.58 Perintah Pemodelan

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

m. Untuk mendapatkan hasil perhitungan dari cluster. Kita dapat melalukan perintah sebagai berikut.

```
labels = model.labels_
labels
```

Gambar 2.59 Perintah Hasil Perhitungan

Maka akan keluar output seperti berikut.

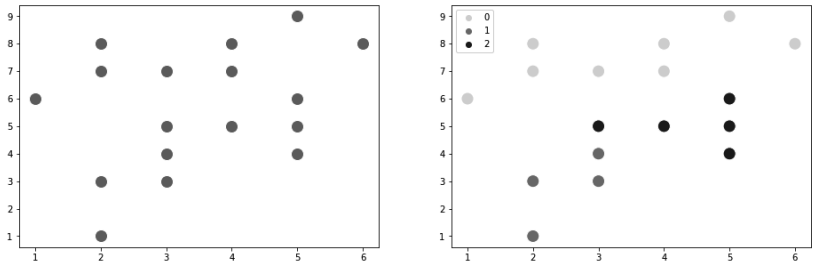
```
array([2, 0, 0, 0, 0, 2, 1, 0, 1, 0, 1, 2, 0, 0, 1, 2, 2])
```

n. Langkah terakhir adalah kita visualisasikan data awal dan hasil clustering dengan kode sebagai berikut.

```
f, axes = plt.subplots(1, 2, figsize=(16,5))

scatter(df.iloc[:,0].values, df.iloc[:,1].values, ax=axes[0], s=200)

scatter(df.iloc[:,0].values, df.iloc[:,1].values, ax=axes[1], hue=labels, s=200)

plt.show()
```

Gambar 2.60 Perintah Visualisasi Data

Maka hasilnya dapat dilihat dalam gambar berikut.



Gambar 2.61 Hasil Visualisasi Data

2.2 DIVISIVE HIERARKI

Divisive hierarki clustering atau bisa disebut Divisive Analysis (DIANA) merupakan algoritma clustering yang merupakan kebalikan dari bentuk agglomerative, yaitu dibentuk dari atas ke bawah (top down) pada sebuah pohon hierarki (Nietto and Do Carmo Nicoletti, 2017). Maksudnya dari satu cluster dipecah-pecah menjadi beberapa cluster. Mula-mula semua data yang ada dianggap sebagai 1 cluster besar. Berikutnya data yang memiliki jarak terjauh dari data lainnya akan

dipisah menjadi cluster baru. Algoritma untuk melakukan clustering menggunakan divisive secara detail dijelaskan sebagai berikut.

1. Anggap semua data menjadi 1 cluster. Hitung jarak antardata dalam cluster tersebut dan membentuknya dalam bentuk matriks.
2. Selanjutnya mencari rata-rata jarak setiap data/objek dengan data/objek lain.
3. Dari hasil langkah 2, dicari data/objek dengan nilai rata-rata terbesar dan akan menjadi splinter group.
4. Hitung jarak nilai elemen objek splinter group dengan nilai rata-rata objek yang tersisa.
5. Dari hasil nomor 4, temukan nilai jarak terbesar. Apabila nilai tersebut positif, objek yang memiliki jarak terbesar tadi akan tergabung dengan splinter group.
6. Ulangi langkah 1-5 sampai semua nilai selisih elemen splinter matriks bernilai negatif.

Untuk memudahkan pemahaman, kita akan melakukan clustering pada data pada yang telah digunakan pada contoh sebelumnya, yaitu data pada Tabel 2.1. Langkah pertama dilakukan dengan membentuk matriks jarak. Untuk membedakan dengan jarak yang sebelumnya, jarak yang akan digunakan di sini adalah menggunakan Euclidean Distance. Dengan menggunakan Euclidean Distance, matriks yang terbentuk dapat dilihat pada Tabel 2.31.

Tabel 2.31 Jarak Antarobjek dengan Euclidean Distance

	1	2	3	4	5	6	7	8	9	10	11
1	0.00										
2	2.83	0.00									
3	1.41	1.41	0.00								
4	8.00	6.32	7.07	0.00							
5	9.90	7.07	8.49	7.07	0.00						
6	4.12	2.24	3.00	8.06	6.71	0.00					
7	7.07	5.83	6.32	1.41	8.00	7.81	0.00				
8	8.54	6.08	7.28	3.00	4.12	7.07	4.12	0.00			
9	9.90	7.07	8.49	7.07	0.00	6.71	8.00	4.12	0.00		
10	7.21	4.47	5.83	7.21	3.16	3.61	7.62	5.00	3.16	0.00	
11	9.22	6.40	7.81	6.08	1.00	6.32	7.00	3.16	1.00	3.00	0.00

Selanjutnya, langkah kedua adalah mencari rata-rata jarak setiap data/objek. Pada tahap ini setiap data membentuk satu objek sehingga kita memiliki 11 objek. Hasil rata-rata setiap objek terhadap objek lain dapat dilihat pada Tabel 2.32.

Tabel 2.32 Rata-Rata Jarak Objek dengan Objek Lain

Objek	Rata-rata jarak objek dengan objek lain
1	(2.83+1.41+8+9.9+4.12+7.07+8.54+9.9+7.21+9.22)/10 =6.82
2	(2.83+1.41+6.32+7.07+2.24+5.83+6.08+7.07+4.47+6.4)/10=4.97
3	(1.41+1.41+7.07+8.49+3+6.32+7.28+8.49+5.83+7.81)/10 =5.7
4	(8+6.32+7.07+7.07+8.06+1.41+3+7.07+7.21+6.08)/10=6.13
5	(9.9+7.07+8.49+7.07+6.71+8+4.12+0+3.16+1)/10=5.55
6	(4.12+2.24+3+8.06+6.71+7.81+7.07+6.71+3.61+6.32)/10 =5.56
7	(7.07+5.83+6.32+1.41+8+7.81+4.12+8+7.62+7)/10=6.32
8	(8.54+6.08+7.28+3+4.12+7.07+4.12+4.12+5+3.16)/10=5.25
9	(9.9+7.07+8.49+7.07+0+6.71+8+4.12+3.16+1)/10=5.55
10	(7.21+4.47+5.83+7.21+3.16+3.61+7.62+5+3.16+3)/10=5.03
11	(9.22+6.4+7.81+6.08+1+6.32+7+3.16+1+3)/10=5.1

Setelah diketahui hasil rata-rata setiap objek terhadap objek lain, langkah ketiga adalah mencari objek yang memiliki nilai rata-rata paling besar. Hasil dari Tabel 2.32 dapat dilihat bahwa nilai paling besar dimiliki oleh objek 1, yaitu dengan nilai rata-rata 6.82. Oleh sebab itu, objek 1 dijadikan splinter group. Dari langkah ketiga ini kita telah memiliki 2 cluster, yaitu cluster {1} dan cluster {2,3,4,5,6,7,8,9,10,11}.

Langkah keempat adalah mencari jarak nilai elemen objek splinter group dengan nilai rata-rata objek yang tersisa. Sebelum mencari jaraknya, terlebih dahulu dicari rata-rata objek yang tersisa yang dapat dilihat pada Tabel 2.33.

Tabel 2.33 Rata-Rata Jarak dengan Objek Lain selain Splinter

Objek	Rata-rata jarak objek dengan objek lain
2	(1.41+6.32+7.07+2.24+5.83+6.08+7.07+4.47+6.4)/9=5.21
3	(1.41+7.07+8.49+3+6.32+7.28+8.49+5.83+7.81)/9=6.12
4	(6.32+7.07+7.07+8.06+1.41+3+7.07+7.21+6.08)/9=5.92
5	(7.07+8.49+7.07+6.71+8+4.12+0+3.16+1)/9=5.07
6	(2.24+3+8.06+6.71+7.81+7.07+6.71+3.61+6.32)/9=5.73
7	(5.83+6.32+1.41+8+7.81+4.12+8+7.62+7)/9=6.23
8	(6.08+7.28+3+4.12+7.07+4.12+4.12+5+3.16)/9=4.89
9	(7.07+8.49+7.07+0+6.71+8+4.12+3.16+1)/9=5.07
10	(4.47+5.83+7.21+3.16+3.61+7.62+5+3.16+3)/9=4.78
11	(6.4+7.81+6.08+1+6.32+7+3.16+1+3)/9=4.64

Setelah didapat rata-rata jarak objek, kemudian baru dihitung jarak objek splinter dengan rata-rata jarak objek. Hasilnya dapat dilihat pada Tabel 2.34.

Tabel 2.34 Selisih Rata-Rata Jarak Antarobjek dengan Objek pada Splinter

Objek	Rata-rata jarak objek dengan objek lain	Jarak objek dengan objek pada splinter group	Selisih rata-rata jarak objek dengan objek lain dengan Jarak objek dengan objek pada splinter group
2	5.21	2.83	2.38
3	6.12	1.41	4.77
4	5.92	8	-2.08
5	5.07	9.9	-4.83
6	5.73	4.12	1.6
7	6.23	7.07	-0.84
8	4.89	8.54	-3.66
9	5.07	9.9	-4.83
10	4.78	7.21	-2.43
11	4.64	9.22	-4.58

Langkah kelima, dari Tabel 2.34 dapat dilihat bahwa selisih rata-rata jarak objek dengan objek lain dan jarak objek dengan objek pada splinter group yang memiliki nilai terbesar adalah data 3 dengan nilai 4.77. Karena nilai 4.77 adalah nilai positif, data 3 bergabung dengan splinter group sehingga dari putaran/iterasi pertama ini didapatkan dua cluster, yaitu cluster {1,3} dan cluster {2,4,5,6,7,8,9,10,11}. Karena telah mempunyai 2 cluster, berikutnya menentukan cluster mana yang akan dipecah. Salah satu cara untuk menentukan cluster mana yang akan dipecah adalah dengan mencari cluster yang memiliki rata-rata terbesar. Karena kita memiliki 2 cluster maka rata-rata setiap cluster dapat dilihat sebagaimana Tabel 2.35.

Tabel 2.35 Rata-Rata Cluster yang Terbentuk

Cluster	Rata-Rata Cluster
{1,3}	1.41/1=1.41
{2,4,5,6,7,8,9,10,11}	(6.32+7.07+2.24+5.83+6.08+7.07+4.47+6.40 + 7.07+8.06+1.41+3.00+7.07+7.21+6.08+6.7 1+ 8.00+4.12+0.00+3.16+1.00+7.81+7.07+6. 71+ 3.61+6.32+4.12+8.00+7.62+7.00+4.12+ 5.00+ 3.16+3.16+1.00+3.00)/36=5.17

Karena cluster yang memiliki rata-rata terbesar adalah cluster {2,4,5,6,7,8,9,10,11}, cluster ini yang selanjutnya akan dipisah. Caranya seperti iterasi sebelumnya dimulai dari langkah 1, yaitu menghitung jarak antardata dalam cluster tersebut. Hasilnya sama seperti hasil langkah 1 pada iterasi sebelumnya. Oleh sebab itu, bisa dilanjutkan ke langkah 2. Langkah 2 adalah menghitung rata-rata jarak pada cluster. Hasilnya dapat dilihat pada Tabel 2.36.

Tabel 2.36 Rata-Rata Jarak Antarobjek Iterasi Kedua

Objek	Rata-rata jarak objek dengan objek lain
2	(6.32+7.07+2.24+5.83+6.08+7.07+4.47+6.4)/8=5.69
4	(6.32+7.07+8.06+1.41+3+7.07+7.21+6.08)/8=5.78
5	(7.07+7.07+6.71+8+4.12+0+3.16+1)/8=4.64
6	(2.24+8.06+6.71+7.81+7.07+6.71+3.61+6.32)/8=6.07
7	(5.83+1.41+8+7.81+4.12+8+7.62+7)/8=6.22
8	(6.08+3+4.12+7.07+4.12+4.12+5+3.16)/8=4.59
9	(7.07+7.07+0+6.71+8+4.12+3.16+1)/8=4.64
10	(4.47+7.21+3.16+3.61+7.62+5+3.16+3)/8=4.65
11	(6.4+6.08+1+6.32+7+3.16+1+3)/8=4.25

Langkah ketiga adalah menentukan objek yang memiliki nilai rata-rata objek terbesar dari langkah sebelumnya, yaitu objek 7. Oleh sebab itu, objek 7 menjadi splinter grup sehingga cluster yang tersisa adalah cluster {2,4,5,6,8,9,10,11}. Jarak rata-rata dari cluster yang tersisa dan splinter group kita cari selisihnya sebagai langkah 4. Sebelum mencari selisihnya, kita cari dulu rata-rata jarak dari cluster yang tersisa. Hasilnya dapat dilihat pada Tabel 2.37.

Tabel 2.37 Rata-Rata Jarak Objek dengan Objek Lain selain Splinter Iterasi Kedua

Objek	Rata-rata jarak objek dengan objek lain
2	(6.32+7.07+2.24+6.08+7.07+4.47+6.4)/7=5.67
4	(6.32+7.07+8.06+3+7.07+7.21+6.08)/7=6.4
5	(7.07+7.07+6.71+4.12+0+3.16+1)/7=4.16
6	(2.24+8.06+6.71+7.07+6.71+3.61+6.32)/7=5.82
8	(6.08+3+4.12+7.07+4.12+5+3.16)/7=4.65
9	(7.07+7.07+0+6.71+4.12+3.16+1)/7=4.16
10	(4.47+7.21+3.16+3.61+5+3.16+3)/7=4.23
11	(6.4+6.08+1+6.32+3.16+1+3)/7=3.85

Setelah itu baru dicari selisihnya terhadap objek splinter sebagaimana Tabel 2.38.

Tabel 2.38 Selisih Rata-Rata Jarak Antarobjek dengan Objek pada Splinter Iterasi Kedua

Objek	Rata-rata jarak objek dengan objek lain	Jarak objek dengan objek pada splinter group	Selisih rata-rata jarak objek dengan objek lain dengan Jarak objek dengan objek pada splinter group
2	5.67	2.53	3.14
4	6.4	1.41	4.99
5	4.16	8	-3.84
6	5.82	7.81	-1.99

Objek	Rata-rata jarak objek dengan objek lain	Jarak objek dengan objek pada splinter group	Selisih rata-rata jarak objek dengan objek lain dengan Jarak objek dengan objek pada splinter group
8	4.65	4.12	0.53
9	4.17	8	-3.83
10	4.23	7.62	-3.39
11	3.85	7	-3.15

Langkah kelima adalah dengan melihat dari Tabel 2.38 yang memiliki nilai tertinggi adalah objek 4. Karena nilai selisih tertinggi merupakan bilangan positif maka objek 4 bergabung dengan objek splinter. Dari iterasi kedua ini, kita mendapat 3 cluster, yaitu cluster {1,3}, {4,7}, dan {2,5,6,8,9,10,11}. Selanjutnya, kembali menghitung rata-rata setiap cluster. Cluster yang memiliki rata-rata terbesar akan menjadi cluster yang akan di-split atau dipisah. Hasil rata-rata cluster dapat dilihat sebagaimana Tabel 2.39.

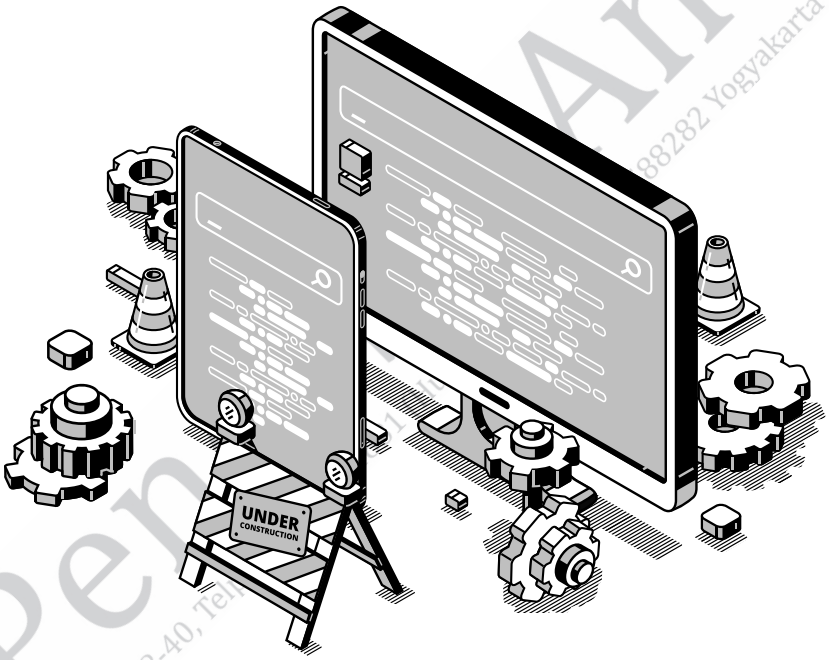
Tabel 2.39 Rata-Rata Cluster yang Terbentuk Iterasi Kedua

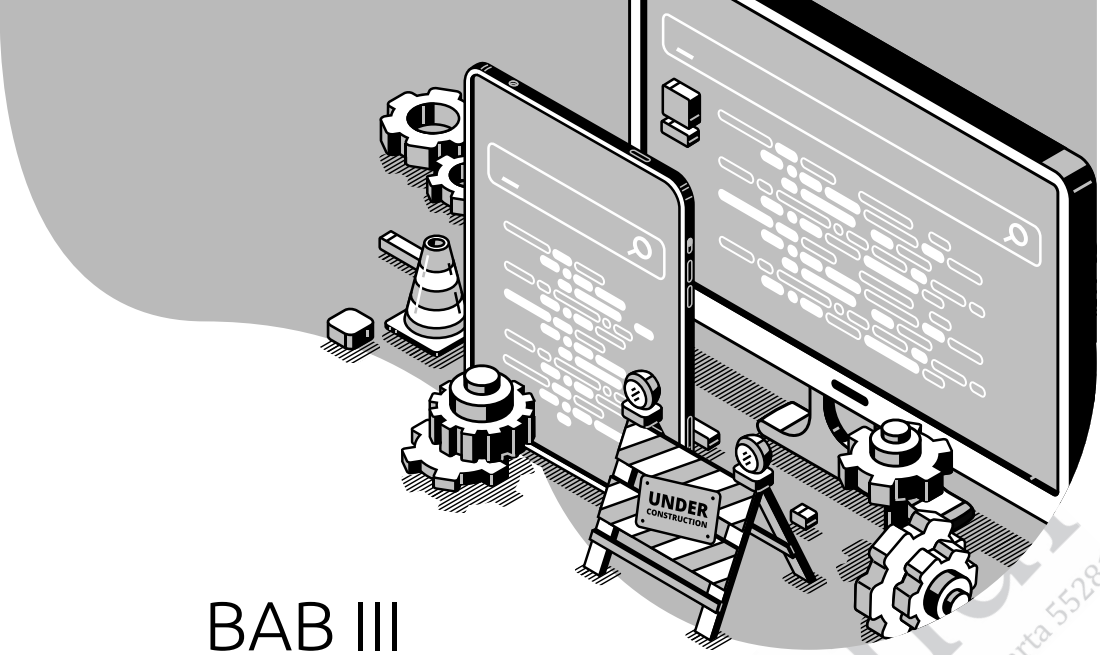
Cluster	Rata-Rata cluster
{1,3}	1.41/1=1.41
{4,7}	1.41/1=1.41
{2,5,6,8,9,10,11}	(7.07+2.24+6.08+7.07+4.5+6.4+6.71+4.12+0+3.16+1+7.07+6.71+3.61+6.32+4.12+5+3.16+3.16+1+3)/21=4.36

Karena cluster {2,5,6,8,9,10,11} memiliki rata-rata cluster paling besar, cluster ini yang akan dipisah. Lakukan iterasi berikutnya sampai selisih rata-rata dalam objek dengan splinter group bernilai negatif semua. Apabila nilai selisih bernilai negatif semua, iterasi berhenti.

2.3 LATIHAN SOAL CLUSTERING BERBASISKAN HIERARKI

1. Jelaskan perbedaan agglomerative dan divisive pada hierarki clustering!
2. Jelaskan macam-macam cara yang dapat digunakan untuk menghitung jarak antardata!
3. Jelaskan kelebihan dan kekurangan single linkage!
4. Jelaskan kelebihan dan kekurangan complete linkage!
5. Jelaskan apa yang Anda ketahui tentang dendrogram dan manfaatnya pada agglomerative clustering!





BAB III
METODE CLUSTERING BERBASISKAN PARTISI

Capaian Pembelajaran:

1. Mampu memahami konsep dari clustering berbasiskan partisi.
2. Mampu memahami konsep dan algoritma dari model-model clustering berbasiskan partisi.
3. Mampu menerapkan algoritma dari model-model clustering berbasiskan partisi untuk memecahkan masalah.

Bab ini akan membahas metode clustering berbasis partisi. Metode clustering berbasis partisi (partitioning-based clustering algorithm) merupakan metode yang cukup populer dan banyak digunakan untuk menyelesaikan berbagai permasalahan data mining. Sesuai dengan namanya, metode ini akan melakukan pemisahan atau partisi data satu tingkat pada dataset yang ada.

Menurut Kaufman dan Rousseeuw, terdapat dua prinsip dasar dalam metode clustering berbasis partisi ini antara lain (Kaufman and Rousseeuw, 1990).

1. Setiap cluster harus memiliki setidaknya satu objek.
2. Setiap objek harus tepat milik satu cluster.

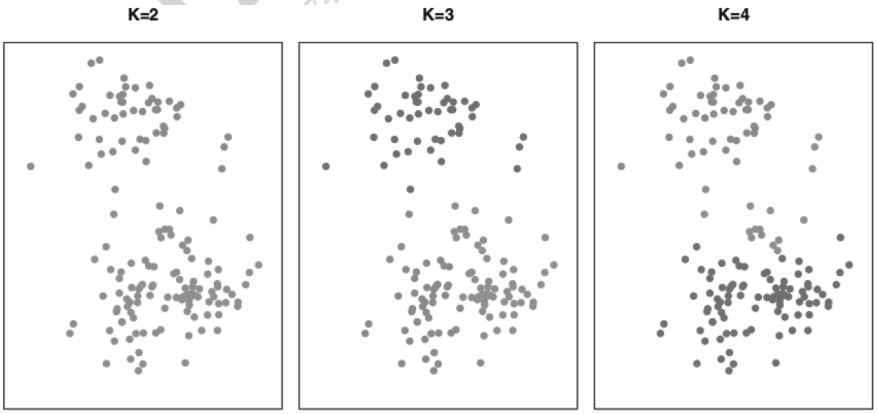
Metode ini berupa sekumpulan n objek yang kemudian membentuk partisi data sebanyak k, di mana setiap partisi mewakili sebuah cluster dan k ≤ n. Metode clustering berbasis partisi ini biasanya mengadopsi konsep pemisahan cluster secara eksklusif. (Kaufman and Rousseeuw, 1990).

Metode clustering yang baik akan menghasilkan cluster berkualitas dengan kesamaan intra-class yang tinggi dan kesamaan inter-class yang rendah. Kualitas hasil clustering juga ditentukan oleh (Han, Kamber, and Pei, 2014).

1. Ukuran kesamaan yang digunakan beserta metode dan implementasinya.
2. Kemampuannya untuk menemukan beberapa atau semua pola yang tersembunyi.

3.1 METODE K-MEANS

K-Means Clustering merupakan salah satu metode sederhana untuk melakukan partisi 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 Gambar 3.1.



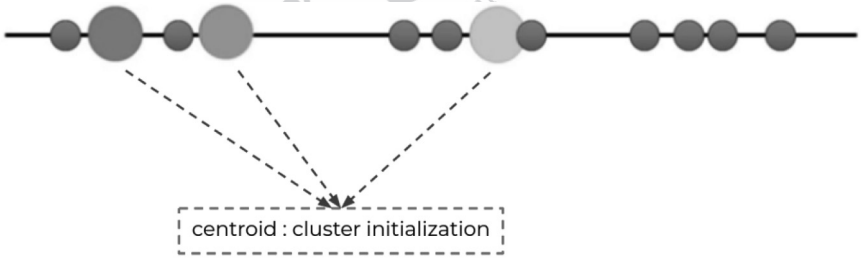
Gambar 3.1 Contoh Jumlah Cluster K (James *et al.*, 2013)

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

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

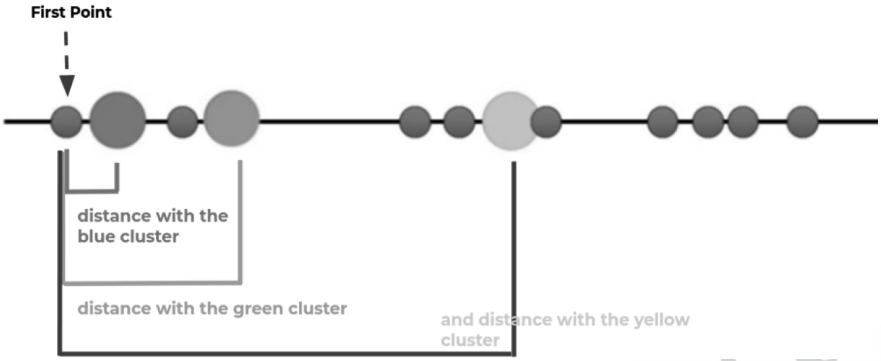
Langkah-langkah yang dilakukan dalam proses K-Means adalah sebagai berikut.

1. Menentukan nilai k, di mana nilai k merepresentasikan jumlah cluster. Dalam 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, yang dalam pembahasan kali ini penulis sudah menentukan di awal pembahasan.
2. Menentukan tiga titik secara acak sebagai centroid atau pusat cluster untuk inisialisasi cluster awal, seperti ditunjukkan pada Gambar 3.3. Penentuan tiga titik tersebut berdasarkan dengan nilai k yang telah ditentukan sebelumnya.



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

3. Menghitung jarak 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, yakni sebagai berikut.

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

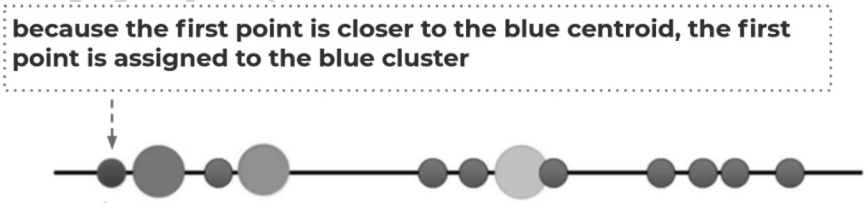
$d_{(a,b)}$ merupakan jarak antara objek a dan b;

n merupakan dimensi dari data;

a_i merupakan koordinat dari objek a pada dimensi data n; dan

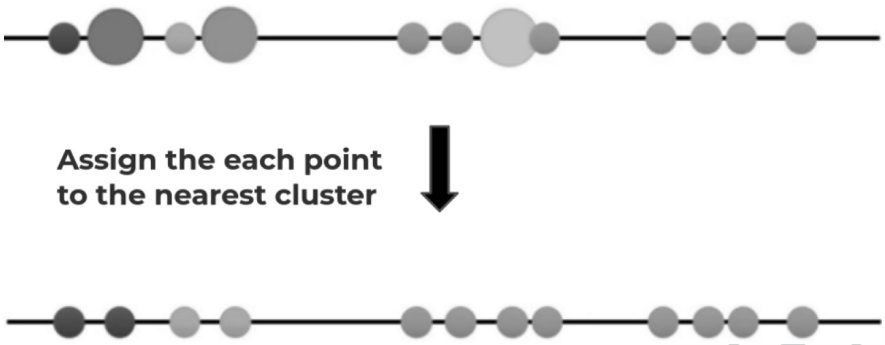
b_i merupakan koordinat dari objek b pada dimensi data n.

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 Gambar 3.5.



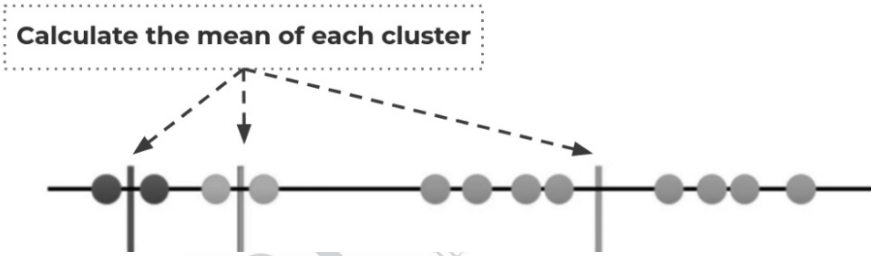
Gambar 3.5 Penetapan Satu Titik pada Cluster (James *et al.*, 2013)

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



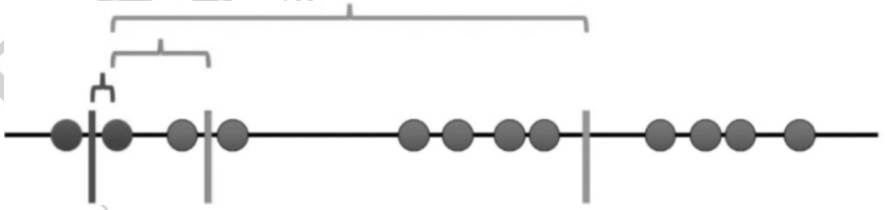
Gambar 3.6 Penetapan Semua Titik pada Cluster (James *et al.*, 2013)

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)

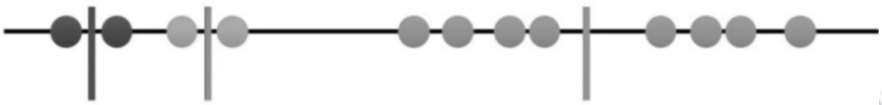
6. Mengulangi langkah ketiga hingga langkah kelima untuk titik tengah yang baru dari setiap cluster, seperti ditunjukkan Gambar 3.8.



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

Proses K-Means ini akan terhenti jika telah memenuhi kondisi berhenti sebagai berikut.

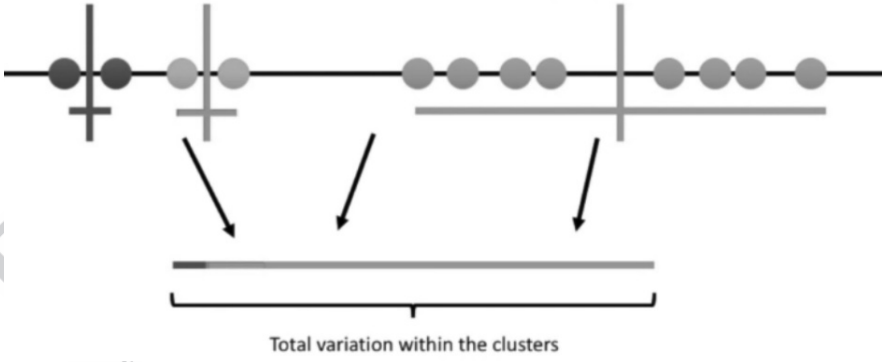
a. Terjadi konvergensi dini (tidak mendapatkan nilai yang lebih baik pada iterasi berikutnya).
b. Telah mencapai iterasi maksimal yang telah ditentukan sebelumnya.
c. Apabila selama proses clustering tidak mengubah cluster data awal, proses dapat diakhiri.



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

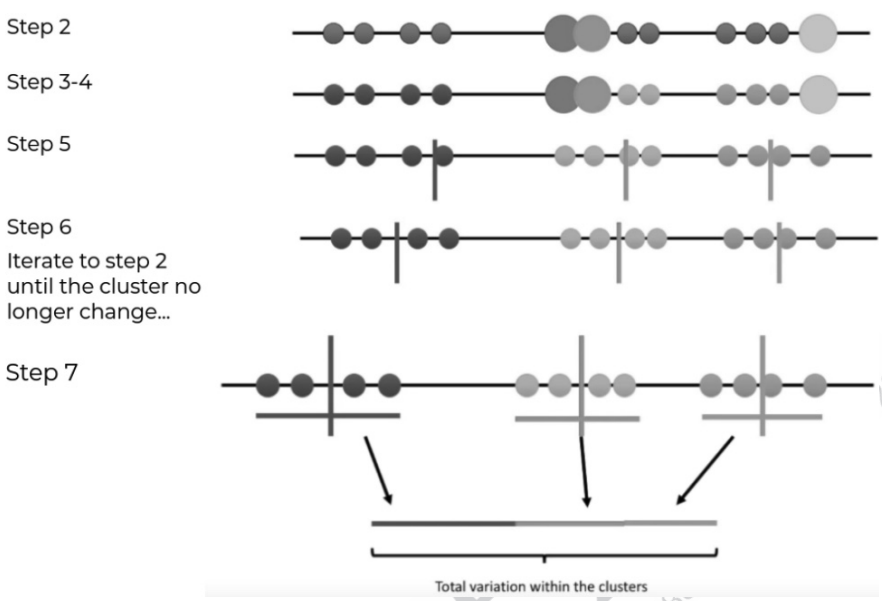
K-Means memiliki tujuan akhir untuk memilih centroid yang baru dengan meminimalkan inersia atau kriteria jumlah kuadrat dalam clusters sehingga proses dalam K-Means dilanjutkan.

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.

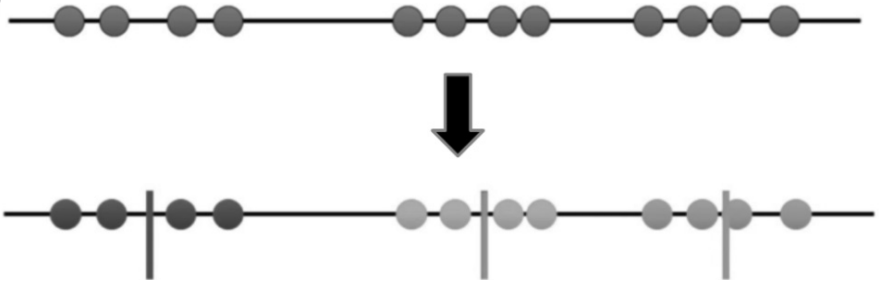


Gambar 3.11 Total Varians Tiap Cluster (James *et al.*, 2013)

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 risiko, 28 gejala penyakit dan 16 class jenis penyakit kelamin.

Implementasi kode program dari metode K-Means untuk studi kasus penyakit kelamin adalah 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++) {
162. arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew]
     [j]+arrDataTraining[i][j])/counter5;
163.                     }
164.                     counter5++;
165.                     break;
166.                 case 5:
167.                     indexNew=5;
168. for (int j = 0; j < arrNewCenteroid[0].length; j++) {
169. arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew]
     [j]+arrDataTraining[i][j])/counter1;
170.                     }
171.                     counter6++;
172.                     break;
173.                 case 6:
174.                     indexNew=6;
175. for (int j = 0; j < arrNewCenteroid[0].length; j++) {
176. arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew]
     [j]+arrDataTraining[i][j])/counter2;
177.                     }
178.                     counter7++;
179.                     break;
```

```
180.                 case 7:
181.                     indexNew=7;
182. for (int j = 0; j < arrNewCenteroid[0].length; j++) {
183. arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew]
     [j]+arrDataTraining[i][j])/counter3;
184.                     }
185.                     counter8++;
186.                     break;
187.                 case 8:
188.                     indexNew=8;
189. for (int j = 0; j < arrNewCenteroid[0].length; j++) {
190. arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew]
     [j]+arrDataTraining[i][j])/counter4;
191.                     }
192.                     counter9++;
193.                     break;
194.                 case 9:
195.                     indexNew=9;
196. for (int j = 0; j < arrNewCenteroid[0].length; j++) {
197. arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew]
     [j]+arrDataTraining[i][j])/counter5;
198.                     }
199.                     counter10++;
200.                     break;
201.                 case 10:
202.                     indexNew=10;
203. for (int j = 0; j < arrNewCenteroid[0].length; j++) {
204. arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew]
     [j]+arrDataTraining[i][j])/counter1;
205.                     }
206.                     counter11++;
207.                     break;
208.                 case 11:
209.                     indexNew=11;
```

```
210. for (int j = 0; j < arrNewCenteroid[0].length; j++) {
211. arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew]
     [j]+arrDataTraining[i][j])/counter2;
212.                     }
213.                     counter12++;
214.                     break;
215.                 case 12:
216.                     indexNew=12;
217. for (int j = 0; j < arrNewCenteroid[0].length; j++) {
218. arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew]
     [j]+arrDataTraining[i][j])/counter3;
219.                     }
220.                     counter13++;
221.                     break;
222.                 case 13:
223.                     indexNew=13;
224. for (int j = 0; j < arrNewCenteroid[0].length; j++) {
225. arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew]
     [j]+arrDataTraining[i][j])/counter4;
226.                     }
227.                     counter14++;
228.                     break;
229.                 case 14:
230.                     indexNew=14;
231. for (int j = 0; j < arrNewCenteroid[0].length; j++) {
232. arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew]
     [j]+arrDataTraining[i][j])/counter5;
233.                     }
234.                     counter15++;
235.                     break;
236.                 case 15:
237.                     indexNew=15;
238. for (int j = 0; j < arrNewCenteroid[0].length; j++) {
```

```
239. arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew]
     [j]+arrDataTraining[i][j])/counter4;
240.                     }
241.                     counter16++;
242.                     break;
243.                 case 16:
244.                     indexNew=16;
245. for (int j = 0; j < arrNewCenteroid[0].length; j++) {
246. arrNewCenteroid[indexNew][j]=(arrNewCenteroid[indexNew]
     [j]+arrDataTraining[i][j])/counter5;
247.                     }
248.                     counter5++;
249.                     break;
250.             }
251.         }
252.     }
253.
254.     public boolean cekCenteroid(){
255.       boolean cek = false;
256.       int cekCount = 0;
257.       for (int i = 0; i < arrCenteroid.length; i++) {
258.       for (int j = 0; j < arrCenteroid[0].length; j++) {
259. if (arrNewCenteroid[i][j]!= arrCenteroid[i][j]) {
260.                     cekCount++;
261.                 }
262.             }
263.         }
264.         if (cekCount<2) {
265.             cek=true;
266.         } else{
267.             //simpan centeroid baru
268.           for (int i = 0; i < arrCenteroid.length; i++) {
269. for (int j = 0; j < arrCenteroid[0].length; j++) {
270. arrCenteroid[i][j]=arrNewCenteroid[i][j];
271.                 }
```

```
272.         }
273.         }
274.
275.         return cek;
276.     }
277.
278.     public  void printJarakCenteroid(){
279. for (int i = 0; i < arrJarakCenteroid.length; i++) {
280. for (int j = 0; j < arrJarakCenteroid[0].length; j++) {
281. System.out.print(df.format(arrJarakCenteroid[i]
     [j])+"\t");
282.             }
283.             System.out.println("");
284.         }
285.     }
286.
287.     public  void printCenteroid(){
288.        for (int i = 0; i < arrNewCenteroid.length; i++) {
289. for (int j = 0; j < arrNewCenteroid[0].length; j++) {
290. System.out.print(df.format(arrNewCenteroid[i]
     [j])+"\t");
291.             }
292.             System.out.println("");
293.         }
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. }
```

3.2 METODE K-MEANS ++

3.2.1 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. Apabila 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:

   $$d_i = \max_{(j:1 \to m)} ||x_i - c_j||^2$$

   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.

3.2.2 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.

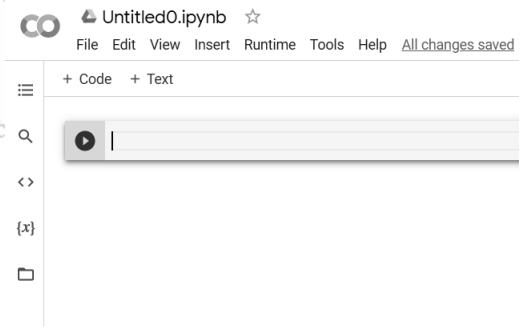
	A	B
1	x	y
2	4	5
3	1	6
4	2	7
5	6	8
6	4	8
7	5	4
8	2	3
9	3	7
10	3	3
11	5	9
12	3	4
13	3	5
14	2	8
15	4	7
16	2	1
17	5	6
18	5	5

Gambar 3.14 File Excel

2. Buka Google Colab dengan alamat:

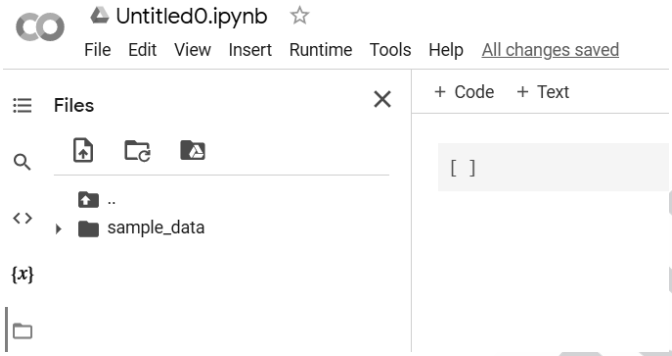
   https://colab.research.google.com/drive/1DvK74PokYRrLKMg0sk-VahsKzygeEKbMu?usp=sharing.

3. Masuk pada menu file, kemudian pilih new notebook maka akan tampil seperti Gambar 3.15.



Gambar 3.15 New Notebook

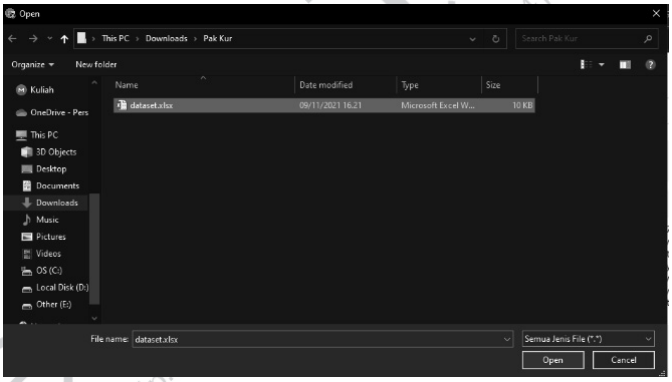
4. Tekan button folder seperti gambar berikut ini.



Gambar 3.16 Button Folder

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

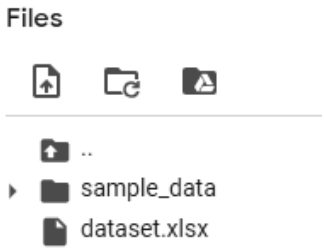
   Setelah menekan button akan muncul window dialog.



Gambar 3.17 Window Dialog

Pilih file yang telah dibuat sebelumnya dan tekan button open.

6. Klik open dan pastikan sudah ter-import.



Gambar 3.18 Import Dataset

7. Pada baris pertama kode ketikan perintah:

   pertama tulis kode untuk menghilangkan warning pada cell.
   import warnings
   warnings.filterwarnings('ignore')

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

Gambar 3.19 Warning

   Setelah menuliskan perintah tersebut, 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 ke dalam Data Frame Python. Library matplolib dan seaborn digunakan untuk menampilkan plotting serta scatter data. Library numpy untuk membuat data menjadi array. Library KMEANS untuk clustering dengan Metode K-Means.

10. Pada cell selanjutnya ketikan perintah untuk mengambil data dari file Excel, seperti dalam code berikut ini.

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

Gambar 3.21 Perintah Mengambil Data

Setelah di-run akan muncul hasil nilai, seperti pada nilai yang dibuat di Excel.

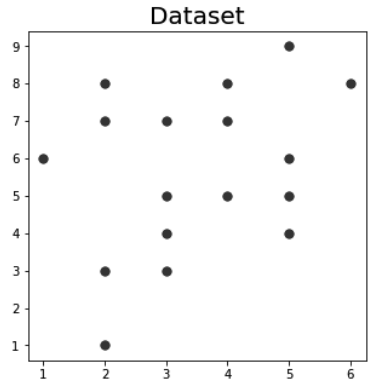
	x	y
0	4	5
1	1	6
2	2	7
3	6	8
4	4	8
5	5	4
6	2	3

Gambar 3.22 Nilai yang Muncul

11. Untuk mendapatkan gambaran dari data kita perlu membuat scatter plotting dari dataset yang telah berhasil diambil nilainya pada cell sebelumnya. Ketikan perintah berikut 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 sebagai berikut.

```
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 sebagai berikut

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

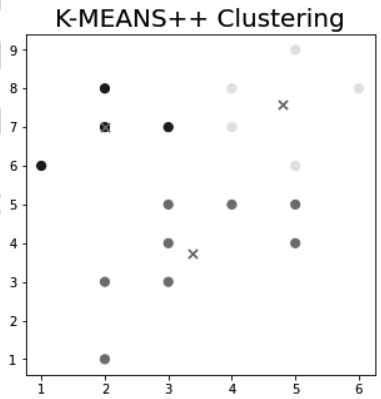
Gambar 3.29 Hasil Pusat Cluster

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 tiga 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 sebagai berikut.

	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

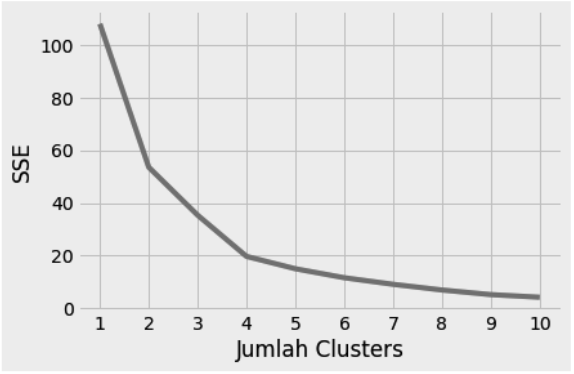
Gambar 3.33 Hasil Perhitungan SSE

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 sebagai berikut.



Gambar 3.35 Visualisasi SSE

3.3 METODE K-MEDOID

3.3.1 Konsep Metode

K-Medoids atau juga disebut Partitioning Around Method (PAM) merupakan suatu metode clustering varian dari metode K-Means. K-Medoids ini mampu menangani kelemahan yang ada pada K-Means yang sensitive terhadap outlier.

Adapun langkah-langkah yang perlu dilakukan dalam proses K-Medoids, yakni sebagai berikut.

1. Menentukan nilai k, di mana nilai k merepresentasikan jumlah cluster.
2. Memilih secara acak medoid awal sebanyak k dari n data.
3. Menghitung jarak masing-masing objek ke medoid sementara, kemudian menandai jarak terdekat objek ke medoid tersebut dengan menghitung totalnya.
4. Mengulangi langkah pemilihan medoid.
5. Menghitung total simpangan (S) dengan ketentuan jika a adalah jumlah jarak terdekat antara objek ke medoid awal dan b adalah jumlah jarak terdekat antara objek ke medoid baru maka total simpangannya adalah $\boldsymbol{S = b - a}$. Apabila $S < 0$, tukar objek dengan data untuk membentuk sekumpulan k baru sebagai medoids.
6. Mengulangi langkah 3 sampai 5 kemudian proses dapat dihentikan jika sudah tidak terjadi perubahan anggota medoid.

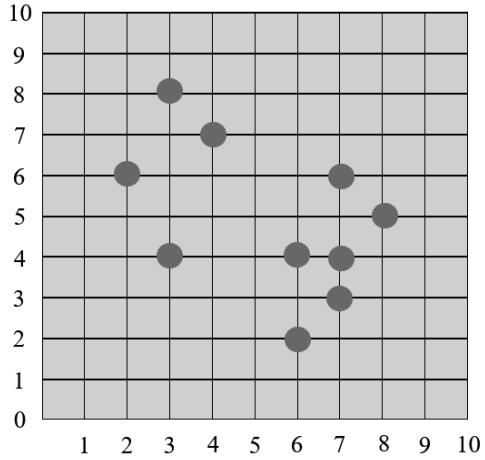
Untuk contoh perhitungan dengan menggunakan K-Medoids seperti berikut.

a. Menentukan nilai k sebesar 2, di mana artinya terdapat 2 cluster.

Tabel 3.1 Contoh Data Awal

Objek	*x*	*y*
A	2	6
B	3	4
C	3	8
D	4	7
E	6	2
F	6	4
G	7	3
H	7	4
I	8	5
J	7	6

Ilustrasi dari contoh data, seperti ditunjukkan pada Gambar 3.36.



Gambar 3.36 Contoh Data Awal

b. Memilih secara acak medoid awal sebanyak k dari n data. Penulis memilih objek B (3,4) sebagai medoid pertama $M1$ dan objek H (7,4) sebagai medoid kedua $M2$.

c. Menghitung jarak masing-masing objek ke medoid sementara, kemudian menandai jarak terdekat objek ke medoid tersebut dengan menghitung totalnya. Untuk menghitung jarak, penulis menggunakan rumus sederhana Euclidean Distance sebagai berikut.

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

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

Tabel 3.2 Perhitungan Jarak Objek ke Medoid Awal 1

Objek	*x*	*y*	Jarak Objek ke Medoid 1
A	2	6	$d_{(A,M1)} = \sqrt{\left((2-3)^2 + (6-4)^2\right)} = 2{,}23$
B	3	4	$d_{(B,M1)} = \sqrt{\left((3-3)^2 + (4-4)^2\right)} = 0$
C	3	8	$d_{(C,M1)} = \sqrt{\left((3-3)^2 + (8-4)^2\right)} = 4$
D	4	7	$d_{(D,M1)} = \sqrt{\left((4-3)^2 + (7-4)^2\right)} = 3{,}16$
E	6	2	$d_{(E,M1)} = \sqrt{\left((6-3)^2 + (2-4)^2\right)} = 3{,}6$
F	6	4	$d_{(F,M1)} = \sqrt{\left((6-3)^2 + (4-4)^2\right)} = 3$
G	7	3	$d_{(G,M1)} = \sqrt{\left((7-3)^2 + (3-4)^2\right)} = 4{,}12$
H	7	4	$d_{(H,M1)} = \sqrt{\left((7-3)^2 + (4-4)^2\right)} = 4$
I	8	5	$d_{(I,M1)} = \sqrt{\left((8-3)^2 + (5-4)^2\right)} = 5{,}1$
J	7	6	$d_{(J,M1)} = \sqrt{\left((7-3)^2 + (6-4)^2\right)} = 4{,}47$

Objek terdekat dengan medoid awal cluster 1, antara lain A dengan jarak 2,23, B dengan jarak 0, C dengan jarak 4, dan D dengan jarak 3,16.

Tabel 3.3 Perhitungan Jarak Objek ke Medoid Awal 2

Objek	*x*	*y*	Jarak Objek ke Medoid 2
A	2	6	$d_{(A,M2)} = \sqrt{\left((2-7)^2 + (6-4)^2\right)} = 5,38$
B	3	4	$d_{(B,M2)} = \sqrt{\left((3-7)^2 + (4-4)^2\right)} = 4$
C	3	8	$d_{(C,M2)} = \sqrt{\left((3-7)^2 + (8-4)^2\right)} = 5,65$
D	4	7	$d_{(D,M2)} = \sqrt{\left((4-7)^2 + (7-4)^2\right)} = 4,24$
E	6	2	$d_{(E,M2)} = \sqrt{\left((6-7)^2 + (2-4)^2\right)} = 2,23$
F	6	4	$d_{(F,M2)} = \sqrt{\left((6-7)^2 + (4-4)^2\right)} = 1$
G	7	3	$d_{(G,M2)} = \sqrt{\left((7-7)^2 + (3-4)^2\right)} = 1$
H	7	4	$d_{(H,M2)} = \sqrt{\left((7-7)^2 + (4-4)^2\right)} = 0$
I	8	5	$d_{(I,M2)} = \sqrt{\left((8-7)^2 + (5-4)^2\right)} = 1,41$
J	7	6	$d_{(J,M2)} = \sqrt{\left((7-7)^2 + (6-4)^2\right)} = 2$

Objek terdekat dengan medoid awal cluster 2, antara lain E dengan jarak 2,23, F dengan jarak 1, G dengan jarak 1, H dengan jarak 0, I dengan jarak 1,41, dan J dengan jarak 2 sehingga total jarak terdekat pada medoid awal adalah 2,23+0+4+3,16+2,23+ 1+1+0+1,41+2= 17,03 dengan anggota cluster 1 adalah A, B, C, dan D, sedangkan anggota cluster 2 adalah E, F, G, H, I, serta J.

d. Mengulangi langkah pemilihan medoid. Pada iterasi kedua ini penulis memilih objek B (3,4) sebagai medoid pertama $M1$ dan objek G (7,3) sebagai medoid kedua $M2$.

e. Menghitung jarak masing-masing objek ke medoid.

Tabel 3.4 Perhitungan Jarak Objek ke Medoid 1

Objek	*x*	*y*	Jarak Objek ke Medoid 1
A	2	6	$d_{(A,M1)} = \sqrt{\left((2-3)^2 + (6-4)^2\right)} = 2{,}23$
B	3	4	$d_{(B,M1)} = \sqrt{\left((3-3)^2 + (4-4)^2\right)} = 0$
C	3	8	$d_{(C,M1)} = \sqrt{\left((3-3)^2 + (8-4)^2\right)} = 4$
D	4	7	$d_{(D,M1)} = \sqrt{\left((4-3)^2 + (7-4)^2\right)} = 3{,}16$
E	6	2	$d_{(E,M1)} = \sqrt{\left((6-3)^2 + (2-4)^2\right)} = 3{,}6$
F	6	4	$d_{(F,M1)} = \sqrt{\left((6-3)^2 + (4-4)^2\right)} = 3$
G	7	3	$d_{(G,M1)} = \sqrt{\left((7-3)^2 + (3-4)^2\right)} = 4{,}12$
H	7	4	$d_{(H,M1)} = \sqrt{\left((7-3)^2 + (4-4)^2\right)} = 4$
I	8	5	$d_{(I,M1)} = \sqrt{\left((8-3)^2 + (5-4)^2\right)} = 5{,}1$
J	7	6	$d_{(J,M1)} = \sqrt{\left((7-3)^2 + (6-4)^2\right)} = 4{,}47$

Objek terdekat dengan medoid baru cluster 1, antara lain A dengan jarak 2,23, B dengan jarak 0, C dengan jarak 4, dan D dengan jarak 3,16.

Tabel 3.5 Perhitungan Jarak Objek ke Medoid 2

Objek	*x*	*y*	Jarak Objek ke Medoid 2
A	2	6	$d_{(A,M2)} = \sqrt{\left((2-7)^2 + (6-3)^2\right)} = 5,83$
B	3	4	$d_{(B,M2)} = \sqrt{\left((3-7)^2 + (4-3)^2\right)} = 4,12$
C	3	8	$d_{(C,M2)} = \sqrt{\left((3-7)^2 + (8-3)^2\right)} = 6,4$
D	4	7	$d_{(D,M2)} = \sqrt{\left((4-7)^2 + (7-3)^2\right)} = 5$
E	6	2	$d_{(E,M2)} = \sqrt{\left((6-7)^2 + (2-3)^2\right)} = 1,41$
F	6	4	$d_{(F,M2)} = \sqrt{\left((6-7)^2 + (4-3)^2\right)} = 1,41$
G	7	3	$d_{(G,M2)} = \sqrt{\left((7-7)^2 + (3-3)^2\right)} = 0$
H	7	4	$d_{(H,M2)} = \sqrt{\left((7-7)^2 + (4-3)^2\right)} = 1$
I	8	5	$d_{(I,M2)} = \sqrt{\left((8-7)^2 + (5-3)^2\right)} = 2,23$
J	7	6	$d_{(J,M2)} = \sqrt{\left((7-7)^2 + (6-3)^2\right)} = 3$

Objek terdekat dengan medoid baru cluster 2, antara lain E dengan jarak 1,41, F dengan jarak 1,41, G dengan jarak 0, H dengan jarak 1, I dengan jarak 2,23, dan J dengan jarak 3 sehingga total jarak terdekat pada medoid sekarang adalah 2,23+0+4+3,16+1,41+1,41 +0+1+2,23+3= 18,45 dengan anggota cluster 1 adalah A, B, C dan D, sedangkan anggota cluster 2 adalah E, F, G, H, I, serta J.

f. Menghitung total simpangan (S) dengan menggunakan rumus $S = b - a$. Total simpangan $S = 18,45 - 17,03 = 1,42$.

g. Iterasi dihentikan karena $S > 0$ sehingga anggota cluster yang terbentuk pada masing-masing medoid yakni anggota cluster 1 adalah A, B, C, dan D, sedangkan anggota cluster 2 adalah E, F, G, H, I, serta J.

3.3.2 Pembuatan Program K-Medoid

Dalam membuat program K-Medoid, contoh ini akan menggunakan Google Colab. Langkah-langkah yang dapat digunakan, yaitu sebagai berikut.

1. Buat file Excel dengan attribute (kolom) dan nilai data (baris), seperti gambar berikut ini serta simpan dengan format .xlsx.

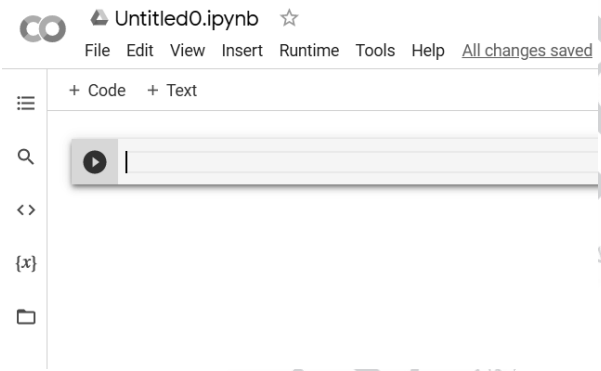
	A	B
1	**x**	**y**
2	4	5
3	1	6
4	2	7
5	6	8
6	4	8
7	5	4
8	2	3
9	3	7
10	3	3
11	5	9
12	3	4
13	3	5
14	2	8
15	4	7
16	2	1
17	5	6
18	5	5

Gambar 3.37 File Excel

2. Buka Google Colab dengan alamat.

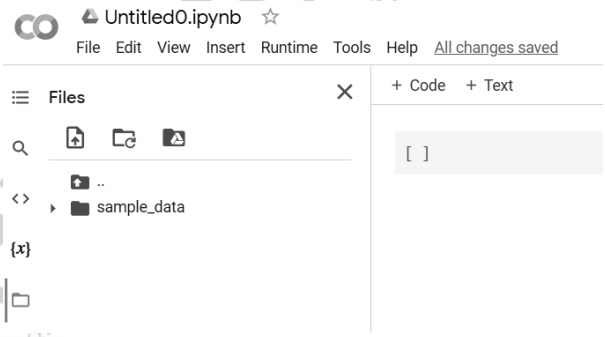
   https://colab.research.google.com/drive/1fWk3pWtF1pYyCJWxfyP6JWVRPQ-gNfSZ?usp=sharing.

3. Masuk pada menu file, kemudian pilih new notebook akan tampil seperti gambar berikut ini.



Gambar 3.38 New Notebook

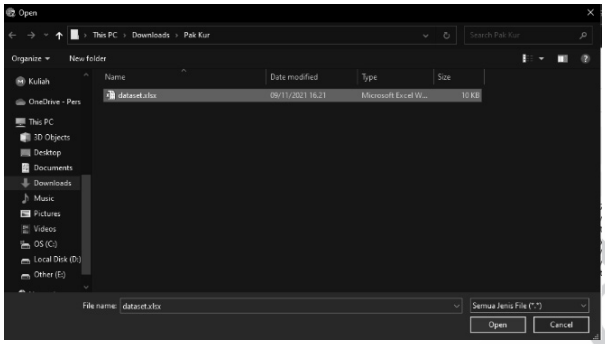
4. Tekan button folder, seperti gambar berikut ini.



Gambar 3.39 Button Folder

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

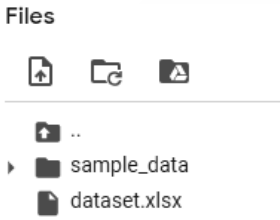
Setelah menekan button akan muncul window dialog.



Gambar 3.40 Window Dialog

Pilih file yang telah dibuat sebelumnya dan tekan button open.

6. Klik open dan pastikan sudah ter-import.



Gambar 3.41 Import Dataset

7. Pada baris pertama kode ketikan perintah:

   pertama tulis kode untuk menghilangkan warning pada cell:
   import warnings
   warnings.filterwarnings(‘ignore’)

Gambar 3.42 Warning

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

8. Tuliskan code pada cell selanjutnya dengan perintah:

   !pip install scikit-learn-extra

9. Import library-library yang dibutuhkan untuk program K-Medoid ini.

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn_extra.cluster import KMedoids
from seaborn import scatterplot as scatter
```

Gambar 3.43 Import Library

Library pandas digunakan untuk mengambil atau mengekstrak nilai dari file Excel atau csv ke dalam Data Frame Python. Library matplolib dan seaborn digunakan untuk menampilkan plotting dan scatter data. Library numpy untuk membuat data menjadi array. Library KMedoids untuk clustering dengan Metode K-Medoid.

10. Pada cell selanjutnya, ketikan perintah untuk mengambil data dari file Excel, seperti code berikut ini.

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

Gambar 3.44 Pengambilan Data dari Excel

Setelah di-run 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

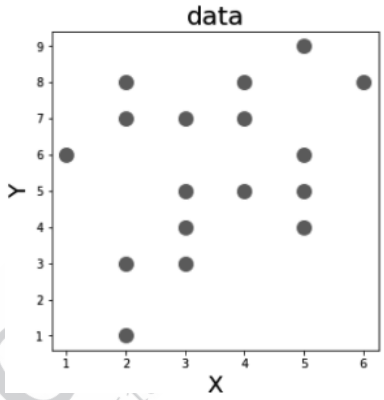
Gambar 3.45 Hasil Nilai

11. Selanjutnya, kita perlu membuat scatter plotting dari dataset yang nilainya telah berhasil diambil dari cell sebelumnya. Ketikan perintah berikut ini untuk mendapatkan gambar scatter plotting dataset.

```
plt.figure(figsize=(5,5))
scatter(df.iloc[:,0].values, df.iloc[:,1].values, marker='o', s=200)
plt.xlabel('X', fontsize=20)
plt.ylabel('Y', fontsize=20)
plt.title('data', fontsize=22)
plt.show()
```

Gambar 3.46 Perintah Gambar Scatter Plotting Dataset

Hasil Ploting:



Gambar 3.47 Gambar Scatter Plotting Dataset

12. Selanjutnya, lakukan training model data dengan mendefinisikan jumlah cluster yang kita inginkan pada parameter **n_cluster** dengan nilai 3.

```
kmedoids = KMedoids(n_clusters=3, random_state=0).fit(df)
```

Gambar 3.48 Inisialisasi Jumlah Cluster

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

13. Untuk melihat label yang terbentuk pada training K-Medoid adalah sebagai berikut.

```
kmedoids.labels_
```

Gambar 3.49 Label Hasil dari Training Data

Hasilnya adalah sebagai berikut.

```
array([0, 1, 2, 2, 2, 0, 1, 2, 1, 2, 1, 1, 2, 2, 1, 0, 0])
```

Gambar 3.50 Hasil dari Training Data

14. Untuk melihat pusat cluster adalah sebagai berikut.

```
kmedoids.cluster_centers_
```

Gambar 3.51 Perintah Melihat Pusat Cluster

Hasilnya adalah sebagai berikut.

```
array([[5, 5],
       [3, 4],
       [4, 8]])
```

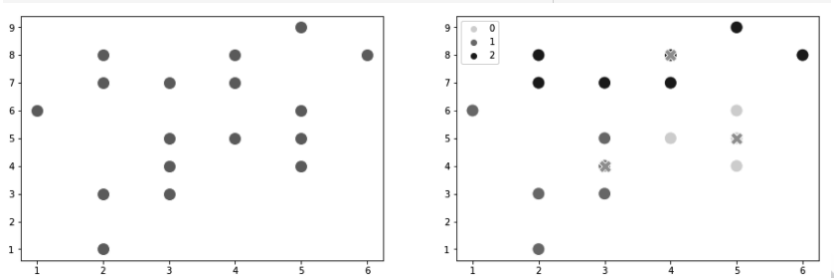
Gambar 3.52 Hasil Pusat Cluster

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

```
f, axes = plt.subplots(1, 2, figsize=(16,5))

scatter(df.iloc[:,0].values, df.iloc[:,1].values, ax=axes[0], s=200)

scatter(df.iloc[:,0].values, df.iloc[:,1].values, ax=axes[1], hue=kmedoids.labels_, s=200)
scatter(kmedoids.cluster_centers_[:,0], kmedoids.cluster_centers_[:,1], ax=axes[1],marker="X",s=200)

plt.show()
```

Gambar 3.53 Perintah Visualisasi Hasil Clustering

Hasil visualisasi data hasil clustering adalah sebagai berikut.



Gambar 3.54 Visualisasi Hasil Clustering

Grafik sebelah kiri merupakan sebaran awal dan grafik sebelah kanan merupakan hasil cluster dengan pusat X dan terdapat 3 cluster.

3.4 FUZZY C-MEANS (FCM)

3.4.1 Konsep Metode

Berbagai macam metode clustering sebelumnya merupakan crisp clustering yang dapat digunakan saat data terdistribusi ke dalam beberapa cluster dengan baik. Namun, kalau data tidak terdistribusi ke dalam beberapa cluster dengan baik maka diperlukan metode fuzzy clustering (Bezdek, Ehrlich, and Full, 1984).

Prinsip logika fuzzy dapat digunakan untuk mengelompokkan data multidimensi, menetapkan setiap titik keanggotaan di setiap pusat cluster dengan derajat keanggotaan 0 hingga 1. Ini yang membedakan fuzzy clustering dengan hard-threshold clustering, di mana setiap titiknya diberi label dengan tepat. Fuzzy clustering bekerja dengan cara menetapkan keanggotaan pada setiap titik data yang sesuai dengan setiap pusat cluster berdasarkan jarak antara pusat cluster dan titik data tersebut. Semakin banyak data yang dekat dengan pusat cluster maka

semakin banyak pula keanggotaannya menuju pusat cluster tertentu. Satu hal yang perlu diperhatikan adalah penjumlahan keanggotaan setiap titik data harus sama dengan satu (Valente de Oliveira and Pedrycz, 2007).

Metode ini termasuk pengelompokan tanpa pengawasan atau biasa disebut dengan unsupervised clustering, dengan begitu memungkinkan kita untuk membangun partisi fuzzy dari data. Adapun proses dari fuzzy clustering, yakni sebagai berikut.

1. Menentukan nilai k, di mana nilai k merepresentasikan jumlah cluster.
2. Menginisialisasi matriks keanggotaan secara acak menggunakan persamaan berikut.

$$\sum_{j=1}^{C} \mu_j(x_i) = 1$$

3. Menghitung centroid dengan menggunakan persamaan berikut.

$$C_j = \frac{\sum_i \left[\mu_j(x_i)\right]^m x_i}{\sum_i \left[\mu_j(x_i)\right]^m}$$

4. Menghitung jarak antara setiap titik data dengan centroid menggunakan Euclidean Distance.

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

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

5. Memperbarui matriks keanggotaan baru menggunakan persamaan berikut.

$$\mu_j(x_i) = \frac{\left[\frac{1}{d_{ji}}\right]^{\frac{1}{m-1}}}{\sum_{k=1}^{C}\left[\frac{1}{d_{ki}}\right]^{\frac{1}{m-1}}}$$

Di mana m merupakan parameter fuzzifikasi.

6. Mengulangi kembali ke langkah ketiga, kecuali centroid tidak berubah.

3.4.2 Perhitungan Manual FCM

Untuk contoh perhitungan dengan menggunakan Fuzzy C-Means seperti berikut.

1. Menentukan nilai k sebesar 2, yang artinya terdapat 2 cluster.
2. Menginisialisasi matriks keanggotaan secara acak.

Tabel 3.6 Contoh Data Awal

Objek	*x*	*y*	C_1	C_2
A	1	6	0,8	0,2
B	2	5	0,9	0,1
C	3	8	0,7	0,3
D	4	4	0,3	0,7
E	5	7	0,5	0,5
F	6	9	0,2	0,8

3. Menghitung centroid dari setiap cluster seperti berikut.

$$C_j = \frac{\sum_i \left[\mu_j(x_i)\right]^m x_i}{\sum_i \left[\mu_j(x_i)\right]^m}, \frac{\sum_i \left[\mu_j(y_i)\right]^m y_i}{\sum_i \left[\mu_j(y_i)\right]^m}$$

$$C_1 = \left[\frac{(1\times0,8)^2+(2\times0,9)^2+(3\times0,7)^2+(4\times0,3)^2+(5\times0,5)^2+(6\times0,2)^2+}{(0,8)^2+(0,9)^2+(0,7)^2+(0,3)^2+(0,5)^2+(0,2)^2}, \right.$$

$$\left. \frac{(6\times0,8)^2+(5\times0,9)^2+(8\times0,7)^2+(4\times0,3)^2+(7\times0,5)^2+(9\times0,2)^2+}{(0,8)^2+(0,9)^2+(0,7)^2+(0,3)^2+(0,5)^2+(0,2)^2} \right]$$

$$C_1 = \frac{5,58}{2,32}, \frac{14,28}{2,32}$$

$$C_1 = (2,4;6,1)$$

$$C_2 = \left[\frac{(1\times0,2)^2+(2\times0,1)^2+(3\times0,3)^2+(4\times0,7)^2+(5\times0,5)^2+(6\times0,8)^2+}{(0,2)^2+(0,1)^2+(0,3)^2+(0,7)^2+(0,5)^2+(0,8)^2}, \right.$$

$$\left. \frac{(6\times0,2)^2+(5\times0,1)^2+(8\times0,3)^2+(4\times0,7)^2+(7\times0,5)^2+(9\times0,8)^2+}{(0,2)^2+(0,1)^2+(0,3)^2+(0,7)^2+(0,5)^2+(0,8)^2}, \right]$$

$$C_2 = \frac{7,38}{1,52}, \frac{10,48}{1,52}$$

$$C_2 = (4,8;6,8)$$

4. Menghitung jarak antara setiap titik data dengan centroid menggunakan Euclidean Distance.

Tabel 3.7 Perhitungan Jarak Objek ke Centroid 1

Objek	*x*	*y*	Jarak Objek ke Centroid 1
A	1	6	$d_{(A,C1)} = \sqrt{\left((2,4-1)^2+(6,1-6)^2\right)} = 1,40$
B	2	5	$d_{(B,C1)} = \sqrt{\left((2,4-2)^2+(6,1-5)^2\right)} = 1,17$
C	3	8	$d_{(C,C1)} = \sqrt{\left((2,4-3)^2+(6,1-8)^2\right)} = 1,99$
D	4	4	$d_{(D,C1)} = \sqrt{\left((2,4-4)^2+(6,1-4)^2\right)} = 2,64$

Objek	*x*	*y*	Jarak Objek ke Centroid 1
E	5	7	$d_{(E,C1)} = \sqrt{\left((2,4-5)^2 + (6,1-7)^2\right)} = 2,75$
F	6	9	$d_{(F,C1)} = \sqrt{\left((2,4-6)^2 + (6,1-9)^2\right)} = 1,40$

Tabel 3.8 Perhitungan Jarak Objek ke Centroid 2

Objek	*x*	*y*	Jarak Objek ke Centroid 2
A	1	6	$d_{(A,C2)} = \sqrt{\left((4,8-1)^2 + (6,8-6)^2\right)} = 3,88$
B	2	5	$d_{(B,C2)} = \sqrt{\left((4,8-2)^2 + (6,8-5)^2\right)} = 3,32$
C	3	8	$d_{(C,C2)} = \sqrt{\left((4,8-3)^2 + (6,8-8)^2\right)} = 2,16$
D	4	4	$d_{(D,C2)} = \sqrt{\left((4,8-4)^2 + (6,8-4)^2\right)} = 2,91$
E	5	7	$d_{(E,C2)} = \sqrt{\left((4,8-5)^2 + (6,8-7)^2\right)} = 0,28$
F	6	9	$d_{(F,C2)} = \sqrt{\left((4,8-6)^2 + (6,8-9)^2\right)} = 2,50$

Tabel 3.9 Jarak Titik Data pada Tiap Cluster

Objek	Cluster 1		Cluster 2	
	(x, y)	Jarak	(x, y)	Jarak
A	(1,6)	1,40	(1,6)	3,88
B	(2,5)	1,17	(2,5)	3,32
C	(3,8)	1,99	(3,8)	2,16
D	(4,4)	2,64	(4,4)	2,91
E	(5,7)	2,75	(5,7)	0,28
F	(6,9)	4,62	(6,9)	2,50

5. Memperbarui matriks keanggotaan baru menggunakan persamaan berikut.

$$\mu_j(x_i) = \frac{\left[\frac{1}{d_{ji}}\right]^{\frac{1}{m-1}}}{\sum_{k=1}^{C}\left[\frac{1}{d_{ki}}\right]^{\frac{1}{m-1}}}$$

Tabel 3.10 Matriks Keanggotaan Tiap Cluster

Cluster 1	Cluster 2
$\mu_{11} = \frac{0,71}{0,96} = 0,7$	$\mu_{21} = \frac{0,25}{0,96} = 0,3$
$\mu_{12} = \frac{0,56}{0,86} = 0,6$	$\mu_{22} = \frac{0,30}{0,86} = 0,4$
$\mu_{13} = \frac{0,50}{0,96} = 0,5$	$\mu_{23} = \frac{0,46}{0,96} = 0,5$
$\mu_{14} = \frac{0,37}{0,71} = 0,5$	$\mu_{24} = \frac{0,34}{0,71} = 0,5$
$\mu_{15} = \frac{0,36}{3,93} = 0,1$	$\mu_{25} = \frac{3,57}{3,93} = 0,9$
$\mu_{16} = \frac{0,21}{0,61} = 0,3$	$\mu_{26} = \frac{0,40}{0,61} = 0,7$

Selanjutnya contoh data awal akan menjadi seperti berikut.

Tabel 3.11 Data Akhir Hasil Pengolahan

Objek	*x*	*y*	C_1	C_2
A	1	6	0,7	0,3
B	2	5	0,6	0,4
C	3	8	0,5	0,5
D	4	4	0,5	0,5
E	5	7	0,1	0,9
F	6	9	0,3	0,7

Melanjutkan proses hingga didapatkan centroid yang sama.

3.4.3 Pembuatan Program FCM

Untuk membuat program FCM, contoh ini akan menggunakan Google Colab. Langkah-langkah yang dapat digunakan, yakni sebagai berikut.

1. Buat file Excel dengan attribute (kolom) dan nilai data (baris), seperti gambar berikut ini, serta simpan dengan format .xlsx.

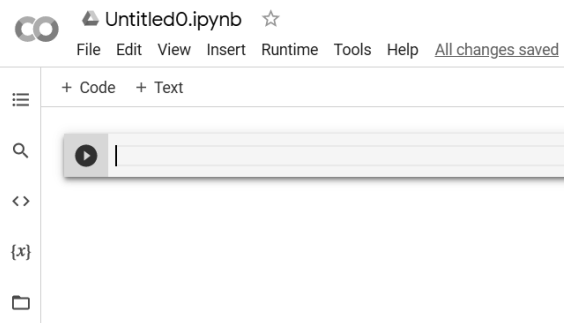
	A	B
1	**x**	**y**
2	4	5
3	1	6
4	2	7
5	6	8
6	4	8
7	5	4
8	2	3
9	3	7
10	3	3
11	5	9
12	3	4
13	3	5
14	2	8
15	4	7
16	2	1
17	5	6
18	5	5

Gambar 3.55 File Excel

2. Buka Google Colab dengan alamat:

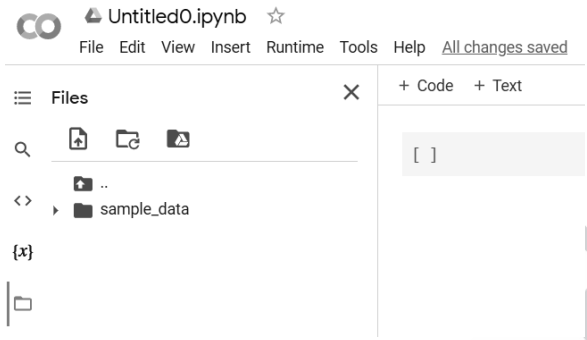
   https://colab.research.google.com/#scrollTo=5fCEDCU_qrC0

3. Masuk pada menu file, kemudian pilih new notebook sehingga akan tampil seperti gambar berikut ini.



Gambar 3.56 New Notebook

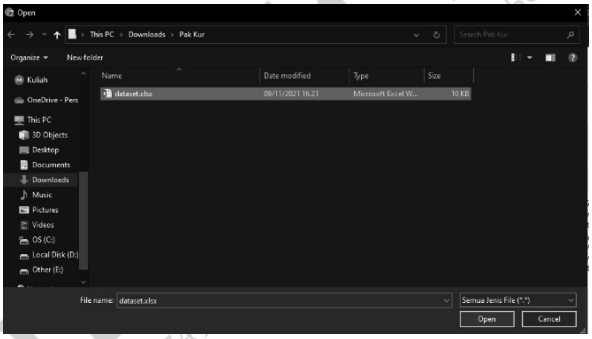
4. Tekan button folder, seperti pada gambar berikut ini.



Gambar 3.57 Button Folder

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

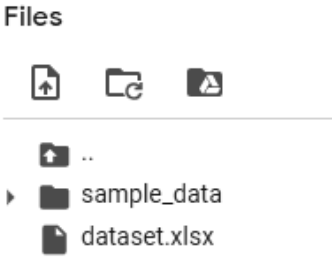
   Setelah menekan button maka akan muncul window dialog



Gambar 3.58 Window Dialog

   Pilih file yang telah dibuat sebelumnya dan tekan button open.

6. Klik open dan pastikan sudah ter-import.



Gambar 3.59 Import Dataset

7. Pada baris pertama kode ketikan perintah.

   Pertama tulis kode untuk menghilangkan warning pada cell.
   import warnings
   warnings.filterwarnings('ignore')

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

Gambar 3.60 Warning

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

8. Tuliskan code pada cell selanjutnya dengan perintah.

   !pip install fuzzy-c-means

   Setelah itu run cell tersebut maka jika tidak masalah maka akan muncul perintah sebagai berikut.

```
!pip install fuzzy-c-means

Requirement already satisfied: fuzzy-c-means in /usr/local/lib/python3.7/dist-packages (1.6.3)
Requirement already satisfied: typer<0.4.0,>=0.3.2 in /usr/local/lib/python3.7/dist-packages (from fuzzy-c-means) (0.3.2)
Requirement already satisfied: tabulate<0.9.0,>=0.8.9 in /usr/local/lib/python3.7/dist-packages (from fuzzy-c-means) (0.8.9)
Requirement already satisfied: pydantic<2.0.0,>=1.8.2 in /usr/local/lib/python3.7/dist-packages (from fuzzy-c-means) (1.8.2)
Requirement already satisfied: numpy<2.0.0,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from fuzzy-c-means) (1.21.4)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/dist-packages (from pydantic<2.0.0,>=1.8
Requirement already satisfied: click<7.2.0,>=7.1.1 in /usr/local/lib/python3.7/dist-packages (from typer<0.4.0,>=0.3.2->fuzzy-
```

Gambar 3.61 Perintah Run Cell

9. Import library-library yang dibutuhkan untuk program FCM ini.

```
import pandas as pd
import matplotlib.pyplot as plt
from seaborn import scatterplot as scatter
from fcmeans import FCM
```

Gambar 3.62 Import Library

Library pandas digunakan untuk mengambil atau mengekstrak nilai dari file Excel atau csv ke dalam Data Frame Python. Library matplolib dan seaborn digunakan untuk menampilkan plotting dan scatter data. Library FCM digunakan untuk menghitung plotting.

10. Pada cell selanjutnya ketikan perintah untuk mengambil data dari file Excel, seperti pada code berikut ini.

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

Gambar 3.63 Perintah Mengambil Data

Setelah di-run akan muncul hasil nilai seperti pada nilai yang dibuat di Excel.

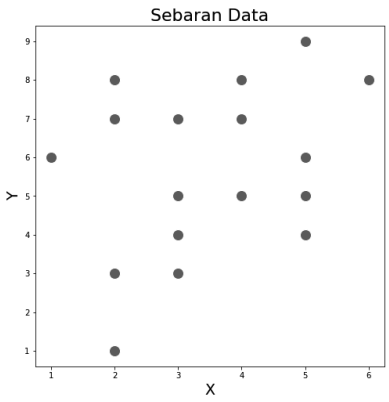
	x	y
0	4	5
1	1	6
2	2	7
3	6	8
4	4	8
5	5	4
6	2	3

Gambar 3.64 Hasil Nilai

11. Untuk mendapatkan gambaran dari data, kita perlu membuat scatter plotting dari dataset yang telah berhasil diambil nilainya pada cell sebelumnya. Ketikan perintah berikut ini untuk mendapatkan gambar scatter plotting dataset.

```
plt.figure(figsize=(8,8))
scatter(df.iloc[:,0].values, df.iloc[:,1].values, marker='o', s=200)
plt.xlabel('X', fontsize=19)
plt.ylabel('Y', fontsize=19)
plt.title('Sebaran Data', fontsize=22)
plt.show()
```

Gambar 3.65 Perintah Gambar Scatter Plotting Dataset



Gambar 3.66 Gambar Scatter Plotting Dataset

12. Buat sebuah model clustering dengan metode FCM dengan membuat code:

```
fcm = FCM(n_clusters=3)
fcm.fit(df.values)
```

Gambar 3.67 Perintah Pemodelan Clustering

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

13. Untuk mendapatkan centroid dari cluster kita dapat melalukan perintah fcm.centers, akan keluar output seperti berikut.

```
fcm.centers

array([[4.75715068, 7.24943715],
       [2.88280857, 3.26329196],
       [2.35300563, 6.8449052 ]])
```

Gambar 3.68 Hasil Centroid

14. Untuk mendapatkan jarak setiap data dengan semua centroid, bisa membuat perintah dengan codeL fcm.u maka akan muncul hasil, seperti pada gambar berikut ini.

```
array([[0.30844585, 0.40746724, 0.2840869 ],
       [0.11652279, 0.16554926, 0.71792795],
       [0.01884238, 0.00979559, 0.97136204],
       [0.82671755, 0.05420088, 0.11908157],
       [0.75252645, 0.0361138 , 0.21135975],
       [0.26205025, 0.55368553, 0.18426422],
       [0.03034335, 0.91742946, 0.05222719],
       [0.11988918, 0.02701826, 0.85309256],
       [0.00389135, 0.99069592, 0.00541273],
       [0.73985543, 0.06180142, 0.19834315],
       [0.03686465, 0.90403463, 0.05910072],
       [0.17180204, 0.46198562, 0.36621234],
       [0.14391157, 0.05061521, 0.80547323],
       [0.78493204, 0.03279323, 0.18227473],
       [0.09740585, 0.77004748, 0.13254668],
       [0.74340481, 0.10059806, 0.15599713],
       [0.45990343, 0.31395228, 0.22614429]])
```

Gambar 3.69 Jarak Data dengan Centroid

15. Selanjutnya, kita definisikan variabel untuk pusat centroid dan label yang didapatkan tiap data dengan kode sebagai berikut.

    fcm_centers = fcm.centers

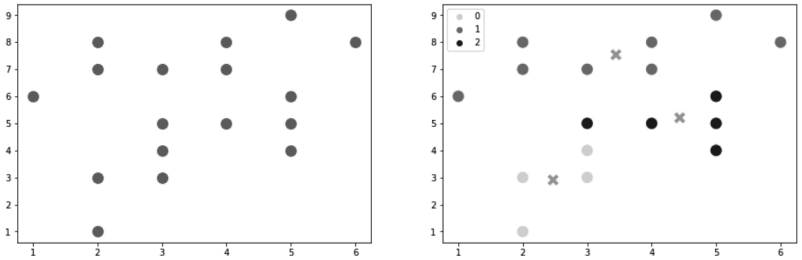
    fcm_labels = fcm.u.argmax(axis=1)

16. Langkah terakhir adalah kita visualisasikan data awal dan hasil clustering dengan kode sebagai berikut.

```
f, axes = plt.subplots(1, 2, figsize=(16,5))

scatter(df.iloc[:,0].values, df.iloc[:,1].values, ax=axes[0], s=200)

scatter(df.iloc[:,0].values, df.iloc[:,1].values, ax=axes[1], hue=fcm_labels, s=200)
scatter(fcm_centers[:,0], fcm_centers[:,1], ax=axes[1],marker="X",s=200)

plt.show()
```

Gambar 3.70 Perintah Visualisasi Hasil Clustering

Maka hasilnya dapat dilihat pada gambar berikut.

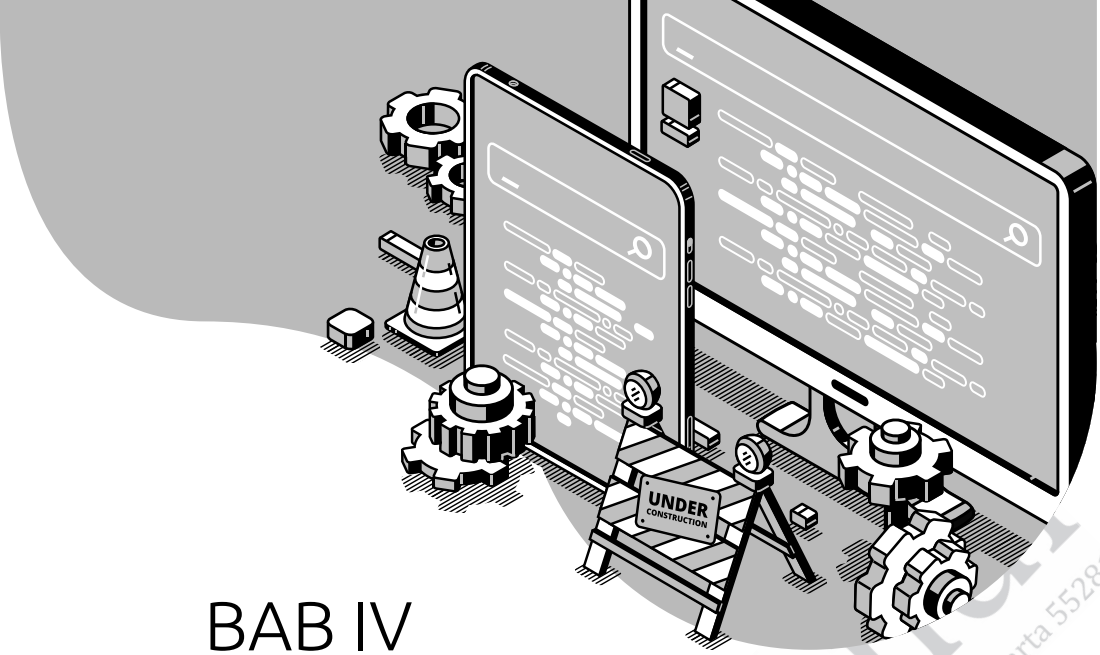


Gambar 3.71 Visualisasi Hasil Clustering

3.5 LATIHAN SOAL CLUSTERING BERBASISKAN PARTISI

1. Jelaskan perbedaan Metode K-Means dan K-Means++!
2. Jelaskan perbedaan Metode K-Means dan K-Medoid!
3. Jelaskan langkah-langkah yang dilakukan pada metode K-Medoid!
4. Jelaskan langkah-langkah yang dilakukan pada fuzzy clustering!
5. Lakukan proses clustering data berikut dengan menggunakan metode fuzzy c-means!

No.	x	y
1	1	2
2	3	4
3	2	3
4	9	2
5	8	9
6	2	6
7	8	1
8	9	5
9	8	9
10	5	8
11	8	8



BAB IV
METODE CLUSTERING BERBASISKAN DENSITY

Capaian Pembelajaran:

1. Mampu memahami konsep dari clustering berbasiskan density (kepadatan).
2. Mampu memahami konsep dan algoritma dari model-model clustering berbasiskan density.
3. Mampu menerapkan algoritma dari model-model clustering berbasiskan density untuk memecahkan masalah.

Clustering berbasis densitas diperkenalkan untuk menentukan cluster pada data spasial yang memiliki noise (Fong *et al.,* 2014). Ada beberapa contoh metode clustering berbasis densitas, antara lain Density Based Spatial Clustering of Applications with Noise (DBSCAN), DBCLASD, dan Density Based Clustering (DNCLUE).

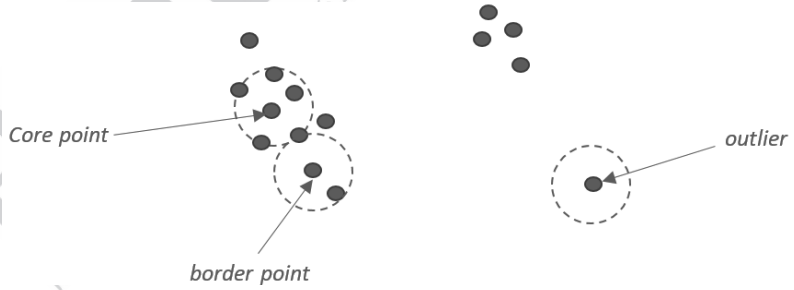
4.1 METODE DBSCAN

DBSCAN merupakan metode pengelompokan data yang memiliki prinsip pengelompokan data yang berdekatan ke dalam satu kelompok. DBSCAN sering digunakan untuk mengelompokkan data yang banyak

mengandung outlier atau noise karena DBSCAN tidak akan memasukkan data yang dianggap outlier ke dalam kelompok mana pun. Algoritma ini pertama kali diperkenalkan oleh Ester dkk. (1996) pada seminar internasional "Knowledge Discovery and Data Mining".

Parameter input yang digunakan pada metode DBSCAN ini ada dua, yaitu epsilon (ε) dan titik minimum atau penyebutannya sering menggunakan MinPts. Ada beberapa istilah pada metode DBSCAN.

1. Epsilon adalah batasan maksimal untuk jarak antarobjek dalam satu kelompok sehingga suatu objek dapat dikatakan tetangga dari objek lain. Parameter ini menjadi landasan jumlah objek yang merupakan tetangga (neighborhood) suatu objek.
2. MinPts adalah batas minimal jumlah objek tetangga yang dimiliki suatu objek. Parameter ini akan memengaruhi suatu objek termasuk core point, border point, atau outlier.
3. Core Point adalah objek dalam suatu kelompok yang memiliki jumlah tetangga minimal sama dengan MinPts.
4. Border Point adalah objek dalam suatu kelompok yang memiliki tetangga tetapi jumlahnya kurang dari MintPts.
5. Outlier adalah objek yang tidak memiliki tetangga.



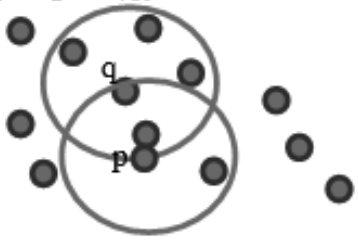
Gambar 4.1 Core Point, Border Point, dan Outlier dengan MintPts = 4 dan ε = 1

Gambar 4.1 menunjukkan gambaran core point, border point, dan outlier dari sekumpulan data dengan Mint Pts.= 4 serta ε = 1. Perhitungan jarak antarobjek menggunakan jarak Euclidean.

Lain halnya dengan K-Means yang ditentukan jumlah kelompoknya di awal, algoritma ini tidak perlu masukan berupa jumlah kelompok karena akan menentukan sendiri jumlah kelompoknya. DBSCAN menemukan cluster dengan cara memeriksa eps-neighborhood (tetangga dalam radius eps) dari setiap poin. Artinya jika tetangga dalam radius eps suatu objek p jumlahnya lebih dari sama dengan nilai MinPts maka cluster baru terbentuk dengan p sebagai core point. Selanjutnya, DBSCAN secara iteratif mengumpulkan objek-objek density reachable (densitas yang terjangkau) dari suatu core point, baik secara langsung atau menggabungkan beberapa cluster yang merupakan densitas terjangkaunya.

4.1.1 Directly Density Reachable

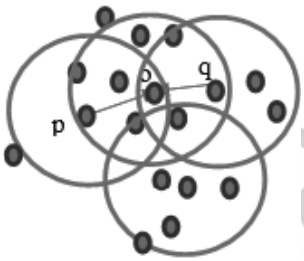
Suatu objek disebut directly density reachable (densitas terjangkau secara langsung) dari suatu objek lainnya jika jarak antara keduanya tidak melebihi nilai eps. Misalkan, titik q densitas terjangkau dari titik p, di mana titik p merupakan core point maka jarak p dan q tidak boleh melebihi eps. Apabila digambarkan keadaan titik p dan q, seperti Gambar 4.2.



Gambar 4.2 p Directly Density Reachable dengan q

4.1.2 Density Reachable

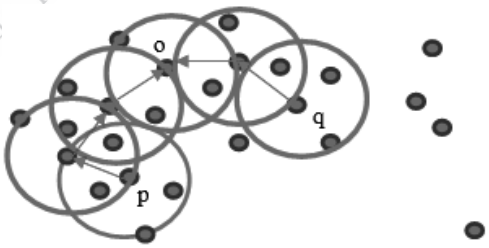
Suatu objek disebut density reachable (densitas terjangkau) dari objek yang lain jika kedua objek terhubung oleh suatu rantai yang berisi objek-objek yang directly density reachable. Misal, ada titik p density reachable dengan titik q maka ada rantai antara titik p_1, p_2,…,p_n, p_1 merupakan p dan p_n merupakan q serta p_{i+1} directly density reachable dengan p_i (Ester *et al.,* 1996).



Gambar 4.3 p Density Reachable dengan q

4.1.3 Density Connected

Titik p dikatakan density connected dengan titik q jika ada titik o di antara keduanya maka titik o ini density reachable terhadap titik p maupun titik q. Density connectivity merupakan relasi simetri dan untuk titik-titik yang density reachable maka relasi tersebut juga bersifat reflektif (Ester *et al.,* 1996). Gambaran titik p yang density connected dengan titik q melalui titik o dijelaskan pada Gambar 4.4.



Gambar 4.4 p Density Connected dengan q Melalui Titik o

4.1.4 Cluster

Dalam suatu database spasial D, suatu cluster C dengan eps dan MinPts tertentu merupakan himpunan bagian tak kosong dari D dengan memenuhi syarat sebagai berikut (Ester *et al.,* 1996).

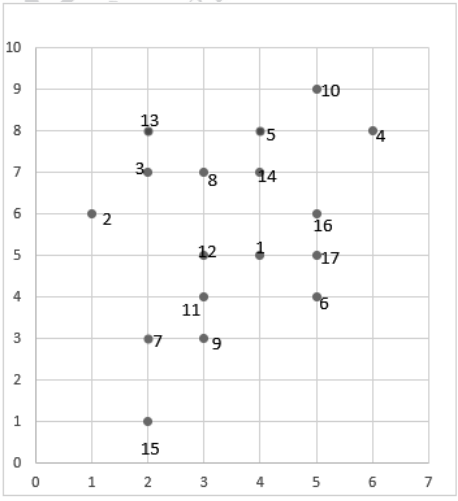
1. Maximality, ∀ p, q: jika p ∈ C dan q adalah density reachable dari p maka q ∈ C.
2. Connectivity, ∀ p, q ∈ C: p density connected dengan q.

Adapun alur dari algoritma DBSCAN adalah sebagai berikut.

1. Memilih sembarang titik p awal.
2. Pilih semua poin yang density reachable dengan titik p.
3. Jika p merupakan core point maka terbentuk cluster.
4. Jika p merupakan border point maka tidak ada density reachable dan DBSCAN mengunjungi titik yang lain dalam database.
5. Proses dilanjutkan hingga semua titik diproses.

4.1.5 Perhitungan Manual DBSCAN

Dari dataset, hasil plot dataset disajikan pada Gambar 4.5.



Gambar 4.5 Plotting Dataset

Iterasi I

Pada contoh perhitungan manual pengelompokan data menggunakan DBSCAN ini digunakan nilai MinPts=3 dan nilai eps=2 satuan. Langkah pertama, yaitu memilih data secara acak, pada perhitungan ini dipilih data 15. Langkah selanjutnya menghitung jarak setiap titik dengan data 15 menggunakan persamaan.

$$\text{distance}\left(d_{i-j}\right)=\sqrt[2]{\left(x_{i,1}-c_{j,1}\right)^2+\left(x_{i,2}-c_{j,2}\right)^2} \quad (4.1)$$

Jarak data 15 dengan data 7 maka:

$$d_{15\text{-}7}=\sqrt[2]{\left(x_{7,1}-c_{15,1}\right)^2+\left(x_{7,2}-c_{15,2}\right)^2}$$

$$=\sqrt[2]{\left(2-2\right)^2+\left(3-1\right)^2}=2$$

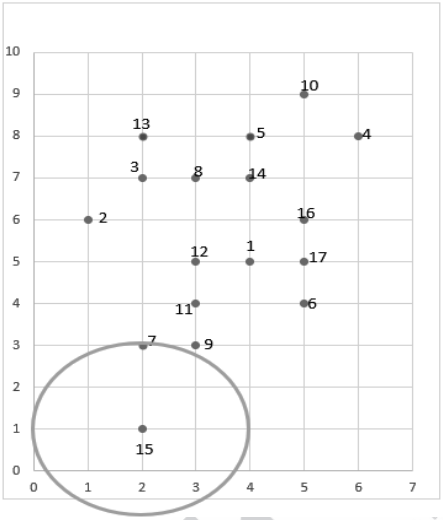
dengan cara yang sama, diperoleh nilai jarak semua titik terhadap data ke-15, seperti ditunjukkan Tabel 4.1.

Tabel 4.1 Nilai Jarak Setiap Titik terhadap Data ke-15

Jarak	Nilai	Jarak	Nilai
$d_{15\text{-}1}$	4,472136	$d_{15\text{-}10}$	8,544004
$d_{15\text{-}2}$	5,09902	$d_{15\text{-}11}$	3,162278
$d_{15\text{-}3}$	6	$d_{15\text{-}12}$	4,123106
$d_{15\text{-}4}$	8,062258	$d_{15\text{-}13}$	7
$d_{15\text{-}5}$	7,28011	$d_{15\text{-}14}$	6,324555
$d_{15\text{-}6}$	4,242641	$d_{15\text{-}15}$	0
$d_{15\text{-}7}$	2	$d_{15\text{-}16}$	5,830952
$d_{15\text{-}8}$	6,082763	$d_{15\text{-}17}$	5
$d_{15\text{-}9}$	2,236068		

Dari hasil jarak tersebut, nilai yang memenuhi d ≤ eps atau density reachable (densitas yang terjangkau) adalah $d_{15\text{-}7}$ dan $d_{15\text{-}15}$. Jumlah densitas yang terjangkau adalah 2, jumlah ini masih kurang dari nilai

MinPts sehingga data 15 bukan core point. Langkah selanjutnya, karena titik tersebut bukan core point maka kembali ke langkah awal.



Gambar 4.6 Sebaran Data pada Iterasi I

Iterasi II

Pada iterasi II, dimulai dengan memilih data lain yang belum menjadi border point maupun core point, di sini data yang dipilih adalah data ke-7 sebagai titik pusatnya. Dengan cara yang sama pada iterasi sebelumnya, yaitu menghitung jarak setiap titik terhadap data ke-7 menggunakan persamaan (4.1). Hasil perhitungan jarak setiap titik terhadap data ke-7 disajikan pada Tabel 4.2.

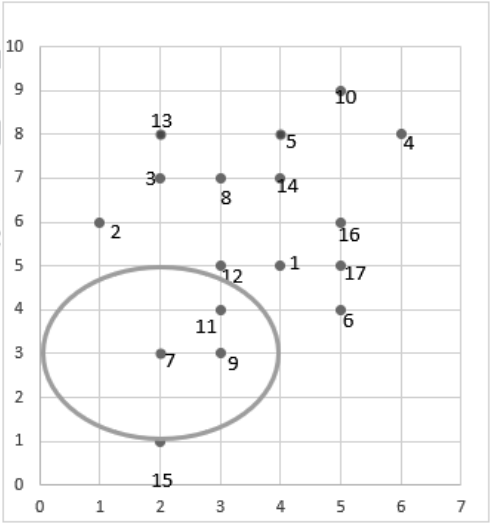
Tabel 4.2 Nilai Jarak Setiap Titik dengan Data ke-7

Jarak	Nilai	Jarak	Nilai
d_{7-1}	2,828427	d_{7-10}	6,708204
d_{7-2}	3,162278	d_{7-11}	1,414214
d_{7-3}	4	d_{7-12}	2,236068
d_{7-4}	6,403124	d_{7-13}	5
d_{7-5}	5,385165	d_{7-14}	4,472136

Jarak	Nilai	Jarak	Nilai
d_{7-6}	3,162278	d_{7-15}	2
d_{7-7}	0	d_{7-16}	4,242641
d_{7-8}	4,123106	d_{7-17}	3,605551
d_{7-9}	1		

Berdasarkan Tabel 4.2 maka nilai d yang memenuhi d ≤ eps atau densitas terjangkau adalah titik 7, titik 9, titik 11, dan titik 15. Ada 4 titik densitas terjangkau sehingga nilai tersebut memenuhi untuk menjadikan titik 7 sebagai core point karena nilai densitas terjangkau ≥ MintPts. Titik 9,11, dan 15 merupakan border point dari titik 7.

Sebaran titik pada iterasi kedua ditunjukkan pada Gambar 4.7. Langkah selanjutnya adalah menentukan core point berikutnya berdasarkan perhitungan iterasi kedua. Dari border point titik 7 yang terjauh adalah titik 15, tetapi pada iterasi sebelumnya titik 15 ini sudah dilakukan perhitungan dan tidak memenuhi syarat menjadi core point maka pilihan core point selanjutnya adalah border point titik 7 yang merupakan titik terjauh berikutnya, yaitu titik 11.



Gambar 4.7 Sebaran Data pada Iterasi II

Iterasi III

Langkah berikutnya seperti halnya langkah sebelumnya, yaitu menghitung jarak semua titik terhadap titik pusatnya. Dalam hal ini titik pusat yang dipilih adalah titik 11. Jarak titik 11 dengan titik 1.

$$d_{11\text{-}1} = \sqrt[2]{\left(x_{1,1} - c_{11,1}\right)^2 + \left(x_{1,2} - c_{11,2}\right)^2}$$

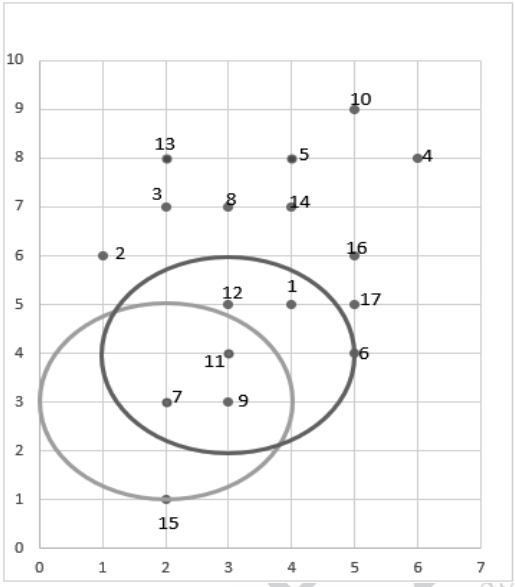
$$= \sqrt[2]{\left(4-3\right)^2 + \left(5-4\right)^2} = 1{,}414214$$

Hasil perhitungan jarak setiap titik terhadap titik 11 secara lengkap disajikan pada Tabel 4.3.

Tabel 4.3 Nilai Jarak Setiap Titik dengan Data ke-11

Jarak	Nilai	Jarak	Nilai
$d_{11\text{-}1}$	1,414214	$D_{11\text{-}10}$	5,385165
$D_{11\text{-}2}$	2,828427	$D_{11\text{-}11}$	0
$D_{11\text{-}3}$	3,162278	$D_{11\text{-}12}$	1
$D_{11\text{-}4}$	5	$D_{11\text{-}13}$	4,123106
$D_{11\text{-}5}$	4,123106	$D_{11\text{-}14}$	3,162278
$D_{11\text{-}6}$	2	$D_{11\text{-}15}$	3,162278
$D_{11\text{-}7}$	1,414214	$D_{11\text{-}16}$	2,828427
$D_{11\text{-}8}$	3	$D_{11\text{-}17}$	2,236068

Berdasarkan Tabel 4.3, titik-titik yang jaraknya memenuhi d ≤ eps adalah titik 1, titik 6, titik 7, titik 9, titik 11, dan titik 12. Ada 6 titik yang merupakan densitas terjangkau dari titik 11. Hal ini menunjukkan bahwa titik 11 memenuhi sebagai core point karena titik densitas terjangkau ≥ MintPts. Sebaran titik pada iterasi ke-3 dapat dilihat pada Gambar 4.8.



Gambar 4.8 Sebaran Data Titik Iterasi III

Titik-titik pada area warna orange merupakan objek ketetanggaan dengan pusat titik 11. Untuk menentukan titik pusat pada iterasi selanjutnya adalah dengan memilih titik border point dari titik 11 yang terjauh dan bukan merupakan border point pada iterasi sebelumnya maka core point pada itersi selanjutnya adalah titik 6.

Iterasi IV

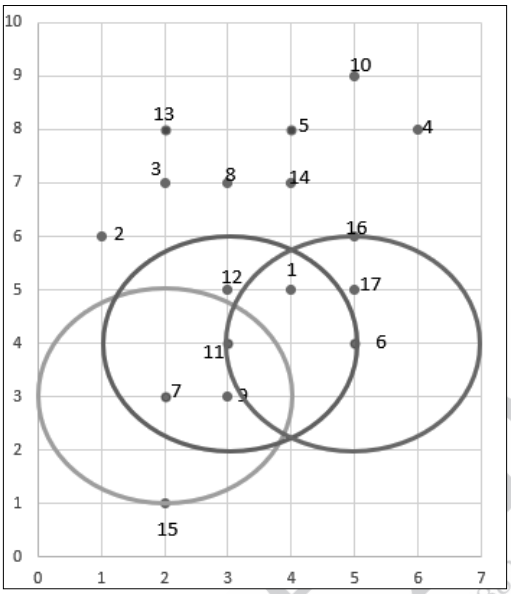
Pada iterasi ini, sama dengan iterasi sebelumnya, yaitu menghitung jarak setiap titik terhadap core point. Hasil perhitungan jarak setiap titik terhadap core point, di mana core point = 6 disajikan pada Tabel 4.4. Gambar versi warna dari Gambar 4.6 sampai Gambar 4.14 serta Gambar 4.31 dapat dilihat di laman https://docs.google.com/document/d/11ExOHb1Lkx5Iyv_5OB1UmUmSiFPjdvG5/edit?usp=sharing&ouid=111285027533853666762&rtpof=true&sd=true (atau https://bit.ly/346NSI5).

Tabel 4.4 Nilai Jarak Setiap Titik dengan Data ke-6

Jarak	Nilai	Jarak	Nilai
d_{6-1}	1,414214	d_{6-10}	5
d_{6-2}	4,472136	d_{6-11}	2
d_{6-3}	4,242641	d_{6-12}	2,236068
d_{6-4}	4,123106	d_{6-13}	5
d_{6-5}	4,123106	d_{6-14}	3,162278
d_{6-6}	0	d_{6-15}	4,242641
d_{6-7}	3,162278	d_{6-16}	2
d_{6-8}	3,605551	d_{6-17}	1
d_{6-9}	2,236068		

Langkah berikutnya adalah menentukan titik-titik yang menjadi densitas terjangkau ($d \leq eps$) dari titik pusat 6. Berdasarkan Tabel 4.4, titik-titik tersebut adalah titik 1, titik 6, titik 11, titik 16, dan titik 17. Ada 5 titik yang merupakan densitas terjangkau dari titik 6 sehingga titik 6 memenuhi sebagai core point. Sebaran titik pada iterasi ke 4 dapat dilihat pada Gambar 4.8.

Langkah berikutnya adalah menentukan core point untuk iterasi selanjutnya. Dari ke-5 titik tersebut dipilih titik yang memiliki jarak terjauh. Jarak terjauh terhadap titik 6 adalah titik 11 dan titik 16. Dari dua pilihan tersebut yang terpilih adalah titik 16 karena titik 11 merupakan core point pada iterasi III. Pemilihan core point tidak hanya berdasarkan jarak terjauh, tetapi titik tersebut bukan termasuk border point dari core point iterasi sebelumnya atau juga bukan core point iterasi sebelumnya.



Gambar 4.9 Sebaran Data pada Iterasi IV

Iterasi V

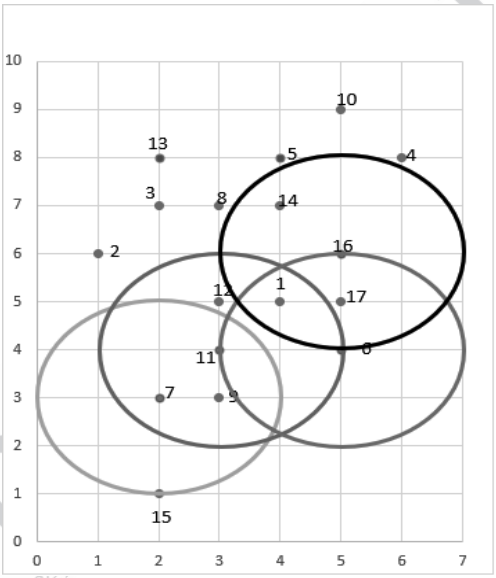
Pada iterasi ini titik yang menjadi titik pusat adalah titik 16. Langkah DBSCAN selanjutnya adalah menghitung semua jarak setiap titik terhadap titik pusat 16. Hasil perhitungan jarak setiap titik terhadap titik pusat 16 disajikan pada Tabel 4.5.

Tabel 4.5 Nilai Jarak Setiap Titik dengan Data ke-6

Jarak	Nilai	Jarak	Nilai
$d_{16\text{-}1}$	1,414214	$d_{16\text{-}10}$	3
$d_{16\text{-}2}$	4	$d_{16\text{-}11}$	2,828427
$d_{16\text{-}3}$	3,162278	$d_{16\text{-}12}$	2,236068
$d_{16\text{-}4}$	2,236068	$d_{16\text{-}13}$	3,605551
$d_{16\text{-}5}$	2,236068	$d_{16\text{-}14}$	1,414214
$d_{16\text{-}6}$	2	$d_{16\text{-}15}$	5,830952

Jarak	Nilai	Jarak	Nilai
$d_{16\text{-}7}$	4,242641	$d_{16\text{-}16}$	0
$d_{16\text{-}8}$	2,236068	$d_{16\text{-}17}$	1
$d_{16\text{-}9}$	3,605551		

Hasil perhitungan jarak semua titik dengan titik pusat 16 sesuai Tabel 4.5, dapat diketahui densitas terjangkau dari titik 16, yaitu titik yang memiliki d ≤ eps. Titik tersebut adalah titik 1, titik 6, titik 14, titik 16, dan titik 17. Ada 5 titik yang merupakan objek ketetanggaan dari titik 16 sehingga titik 16 memenuhi sebagai core point.



Gambar 4.10 Sebaran Titik pada Iterasi V

Berdasarkan Gambar 4.10, titik-titik pada area warna hitam merupakan objek ketetanggaan dari titik 16. Langkah DBSCAN selanjutnya adalah menentukan titik sebagai titik pusat pada iterasi selanjutnya. Dari 5 titik yang menjadi densitas terjangkau titik 16, titik terjauhnya adalah titik 6, tetapi titik 6 tidak bisa dipilih karena

merupakan core point dari iterasi sebelumnya. Titik terjauh selanjutnya adalah titik 1 dan titik 14 yang memiliki nilai yang sama besar. Namun, titik 1 tidak dapat dipilih karena merupakan border point pada iterasi sebelumnya juga sehingga titik pusat pada iterasi VI adalah titik 14.

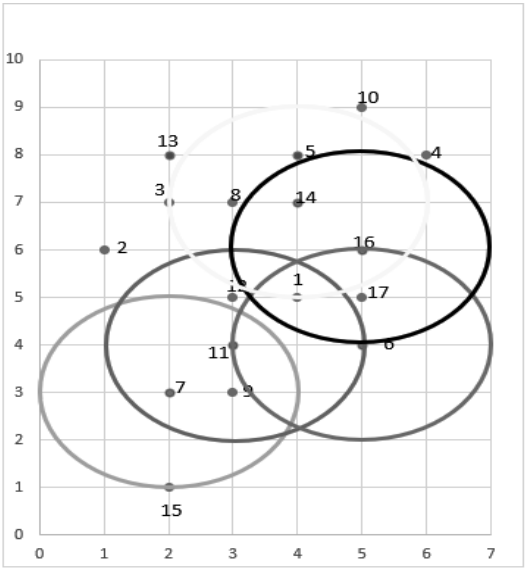
Iterasi VI

Pada iterasi VI yang menjadi titik pusat adalah titik 14. Langkah DBSCAN selanjutnya adalah menghitung jarak semua titik terhadap titik pusat 14. Hasil perhitungan jarak semua titik terhadap titik 14 disajikan pada Tabel 4.6.

Tabel 4.6 Nilai Jarak Setiap Titik dengan Data ke-14

Jarak	Nilai	Jarak	Nilai
$d_{14\text{-}1}$	2	$d_{14\text{-}10}$	2,236068
$d_{14\text{-}2}$	3,162278	$d_{14\text{-}11}$	3,162278
$d_{14\text{-}3}$	2	$d_{14\text{-}12}$	2,236068
$d_{14\text{-}4}$	2,236068	$d_{14\text{-}13}$	2,236068
$d_{14\text{-}5}$	1	$d_{14\text{-}14}$	0
$d_{14\text{-}6}$	3,162278	$d_{14\text{-}15}$	6,324555
$d_{14\text{-}7}$	4,472136	$d_{14\text{-}16}$	1,414214
$d_{14\text{-}8}$	1	$d_{14\text{-}17}$	2,236068
$d_{14\text{-}9}$	4,123106		

Langkah selanjutnya adalah menentukan titik-titik yang menjadi densitas terjangkau dari titik 14, yaitu titik yang memiliki d ≤ eps. Titik-titik tersebut adalah titik 1, titik 3, titik 5, titik 8, titik 14, dan titik 16. Ada 6 titik sehingga titik 14 merupakan core point karena densitas terjangkaunya memenuhi syarat, yaitu lebih besar dari MintPts. Titik-titik yang menjadi densitas terjangkau, seperti Gambar 4.11 di area warna kuning.



Gambar 4.11 Sebaran Titik pada Iterasi VI

Langkah selanjutnya adalah menentukan titik pusat untuk iterasi selanjutnya. Dari titik yang menjadi densitas terjangkau titik 14, titik yang terjauh adalah titik 1 dan titik 3. Titik 1 tidak dapat dipilih karena merupakan border point dari core point pada itrasi sebelumnya sehingga titik pusat untuk iterasi selanjutnya adalah titik 3.

Iterasi VII

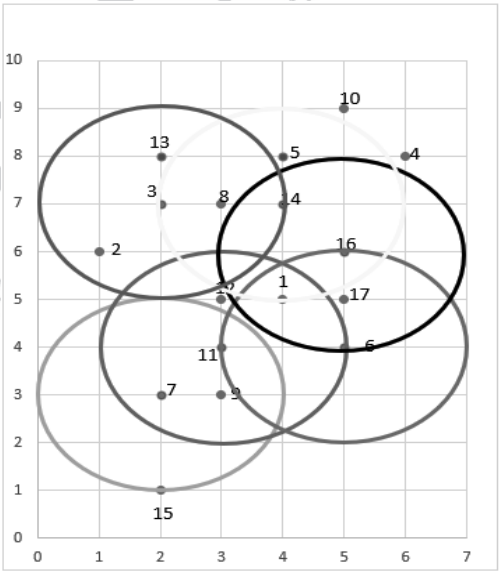
Seperti iterasi sebelumnya, langkah DBSCAN dimulai dengan menghitung jarak semua titik terhadap titik pusatnya. Titik pusat pada iterasi ini adalah titik 3. Hasil perhitungan jarak semua titik terhadap titik 3 disajikan pada Tabel 4.7.

Tabel 4.7 Nilai Jarak Setiap Titik dengan Data ke-3

Jarak	Nilai	Jarak	Nilai
d_{3-1}	2,828427	d_{3-10}	3,605551
d_{3-2}	1,414214	d_{3-11}	3,162278

Jarak	Nilai	Jarak	Nilai
d_{3-3}	0	d_{3-12}	2,236068
d_{3-4}	4,123106	d_{3-13}	1
d_{3-5}	2,236068	d_{3-14}	2
d_{3-6}	4,242641	d_{3-15}	6
d_{3-7}	4	d_{3-16}	3,162278
d_{3-8}	1	d_{3-17}	3,605551
d_{3-9}	4,123106		

Sesuai langkah DBSCAN, langkah selanjutnya adalah menentukan titik-titik yang menjadi densitas terjangkau dari titik 3. Titik-titik yang menjadi densitas terjangkau adalah yang memiliki d ≤ eps, yaitu titik 3, titik 8, titik 13, dan titik 14. Ada 4 titik yang berada pada daerah yang menjadi densitas terjangkau dari titik 3. Hal ini cukup untuk menjadikan titik 3 sebagai core point karena densitas terjangkaunya memenuhi syarat, yaitu lebih besar dari MintPts. Titik-titik yang menjadi densitas terjangkau pada iterasi VII, seperti Gambar 4.12 di area warna ungu.



Gambar 4.12 Sebaran Titik pada Iterasi VII

Langkah berikutnya adalah menentukan titik pusat untuk iterasi selanjutnya. Titik pusat selanjutnya ditentukan dari border point dari iterasi VII yang terjauh. Titik tersebut adalah titik 14. Namun, titik ini tidak bisa dipilih karena sudah menjadi core point dari iterasi sebelumnya. Pilihan selanjutnya adalah titik 2 yang merupakan titik terjauh kedua. Titik 2 bukan border point dari core point sebelumnya.

Iterasi VIII

Pada iterasi ini, yang menjadi titik pusat adalah titik 2. Langkah selanjutnya adalah menghitung jarak semua titik terhadap titik 2. Hasil perhitungan jarak semua titik terhadap titik 2 disajikan pada Tabel 4.8.

Tabel 4.8 Nilai Jarak Setiap Titik dengan Data ke-2

Jarak	Nilai	Jarak	Nilai
d_{3-1}	3,162278	d_{3-10}	5
d_{3-2}	0	d_{3-11}	2,828427
d_{3-3}	1,414214	d_{3-12}	2,236068
d_{3-4}	5,385165	d_{3-13}	2,236068
d_{3-5}	3,605551	d_{3-14}	3,162278
d_{3-6}	4,472136	d_{3-15}	5,09902
d_{3-7}	3,162278	d_{3-16}	4
d_{3-8}	2,236068	d_{3-17}	4,123106
d_{3-9}	3,605551		

Dari Tabel 4.8, dapat dilihat bahwa titik yang merupakan densitas terjangkau dari titik 2 adalah titik 3 dan titik 2. Hasil tersebut menunjukkan bahwa titik 2 bukan core point karena jumlah titik dalam radius eps tidak memenuhi MintPts = 3. Oleh sebab itu, tidak

ada lagi yang menjadi core point dalam densitas terjangkau. Langkah selanjutnya adalah memilih titik yang belum pernah menjadi border point atau titik yang belum pernah dikunjungi atau masih bebas. Titik tersebut akan menjadi titik pusat pada iterasi selanjutnya. Titik yang dipilih untuk menjadi titik pusat adalah titik 4.

Iterasi IX

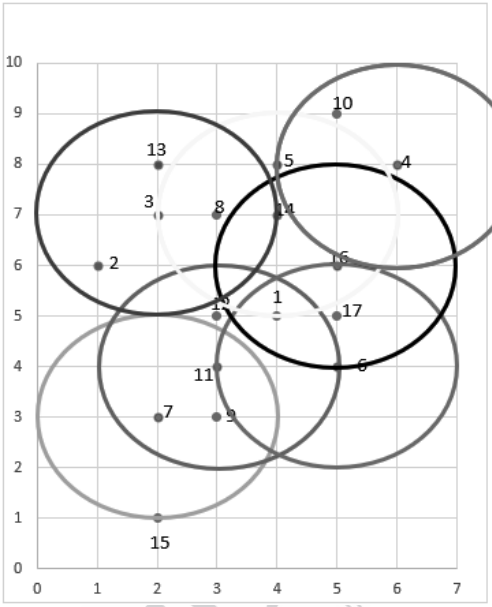
Pada iterasi ini, yang menjadi titik pusat adalah titik 4. Langkah DBSCAN selanjutnya menghitung jarak setiap titik terhadap titik pusat. Hasil perhitungan jarak setiap titik terhadap titik 4 disajikan pada Tabel 4.9.

Tabel 4.9 Nilai Jarak Setiap Titik dengan Data ke-4

Jarak	Nilai	Jarak	Nilai
d_{3-1}	3,605551	d_{3-10}	1,414214
d_{3-2}	5,385165	d_{3-11}	5
d_{3-3}	4,123106	d_{3-12}	4,242641
d_{3-4}	0	d_{3-13}	4
d_{3-5}	2	d_{3-14}	2,236068
d_{3-6}	4,123106	d_{3-15}	8,062258
d_{3-7}	6,403124	d_{3-16}	2,236068
d_{3-8}	3,162278	d_{3-17}	3,162278
d_{3-9}	5,830952		

Dari hasil perhitungan jarak setiap titik terhadap titik pusat, selanjutnya adalah menentukan titik-titik yang menjadi densitas terjangkau dari titik pusatnya. Titik-titik tersebut adalah titik yang memiliki d ≤ eps, yaitu titik 4, titik 5, dan titik 10. Jumlah titik dalam densitas terjangkau adalah 3, di mana nilai tersebut ≥ MintPts. Hal ini

menunjukkan bahwa titik 4 merupakan core point. Daerah densitas terjangkau dari titik 4 digambarkan dengan area warna merah pada Gambar 4.13.

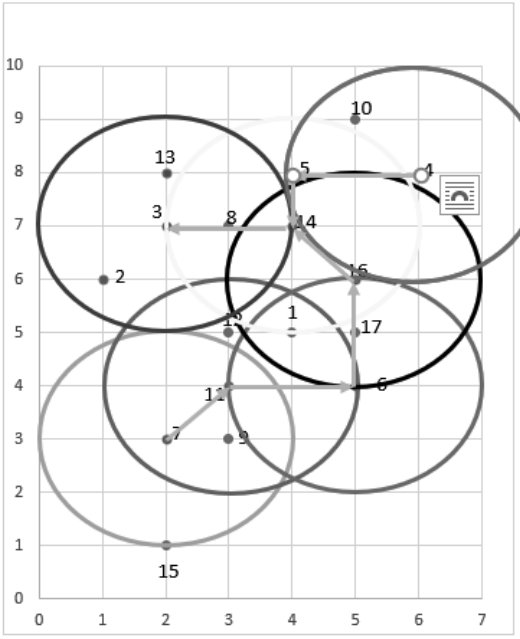


Gambar 4.13 Sebaran Titik pada Iterasi IX

Berdasarkan Gambar 4.13 sudah tidak ada titik yang dapat dijadikan core point lagi. Hal ini ditunjukkan dari nilai titik terjauh adalah titik 5, sedangkan titik 5 merupakan border point dari iterasi sebelumnya. Titik terjauh berikutnya adalah titik 10 jika dipilih menjadi titik pusat maka densitas terjangkaunya sama dengan densitas terjangkau titik 4 sehingga iterasi selesai.

Langkah selanjutnya adalah menandai hubungan antar-core point. Apabila antardensitas terjangkau suatu core ada keterhubungan atau terdapat density connected maka didefinisikan dalam satu cluster. Hubungan antardensitas terjangkau suatu core point dapat dilihat pada Gambar 4.14. Dari gambar tersebut ditunjukkan bahwa ada hubungan

core point satu dengan yang lain baik secara langsung maupun tidak langsung. Pada core point 4 terhubung dengan core point 14 melalui border point 5.



Gambar 4.14 Hubungan Antartitik

Hasil tersebut menunjukkan bahwa pengelompokan data dengan metode DBSCAN dengan nilai parameter eps = 2 dan MintPts = 3 menghasilkan 1 kelompok. Hasil pengelompokan menggunakan DBSCAN sangat dipengaruhi oleh parameter eps dan MintPts.

4.1.6 Pembuatan Program DBSCAN

Untuk membuat program DBSCAN, pada contoh ini akan menggunakan Google Colab. Langkah-langkah yang dapat digunakan, yakni sebagai berikut.

1. Buat file Excel dengan attribute (kolom) dan nilai data (baris), seperti gambar berikut ini, serta 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 4.15 File Excel

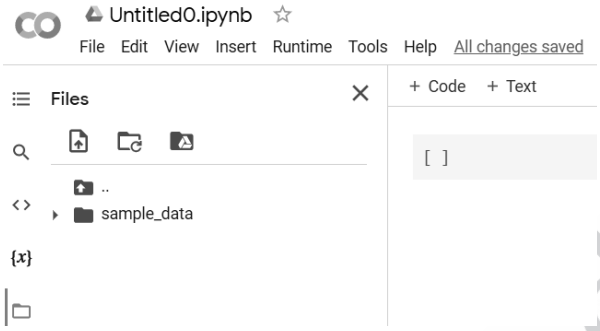
2. Buka Google Colab dengan alamat:

   https://colab.research.google.com/drive/11U77U66E0I5T-jAOO7CPVM7-84e-w0C9y?usp=sharing.

3. Masuk pada menu file, kemudian pilih new notebook dan akan tampil seperti gambar berikut.

Gambar 4.16 New Notebook

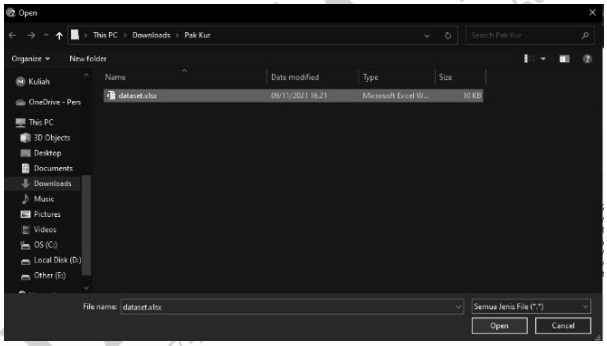
4. Tekan button folder, seperti pada gambar berikut.



Gambar 4.17 Button Folder

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

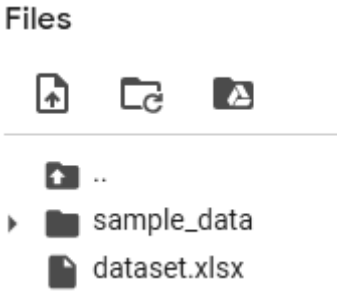
   Setelah menekan button maka akan muncul window dialog.



Gambar 4.18 Window Dialog

Pilih file yang telah dibuat sebelumnya dan tekan button open.

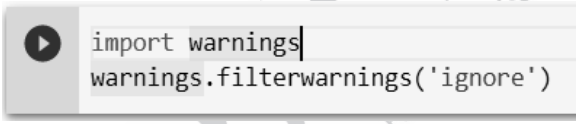
6. Klik open dan pastikan sudah ter-import.



Gambar 4.19 Import Dataset

7. Pada baris pertama kode ketikan perintah:

   Pertama tulis kode untuk menghilangkan warning pada cell:
   import warnings
   warnings.filterwarnings('ignore')



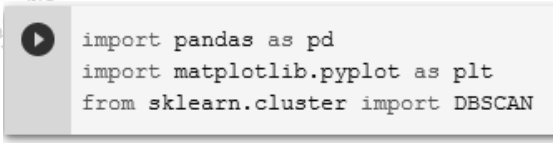
Gambar 4.20 Warning

   Setelah menuliskan perintah tersebut, 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 DBSCAN ini.



Gambar 4.21 Import Library

   Library pandas digunakan untuk mengambil atau mengekstrak nilai dari file Excel atau csv ke dalam Data Frame Python. Library

matplolib dan seaborn digunakan untuk menampilkan plotting serta scatter data. Library DBSCAN digunakan untuk menghitung plotting.

10. Pada cell selanjutnya, ketik perintah untuk mengambil data dari file Excel, seperti pada code berikut ini.

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

Gambar 4.22 Perintah Mengambil Data

Setelah di-run akan muncul hasil nilai seperti pada nilai yang dibuat di Excel.

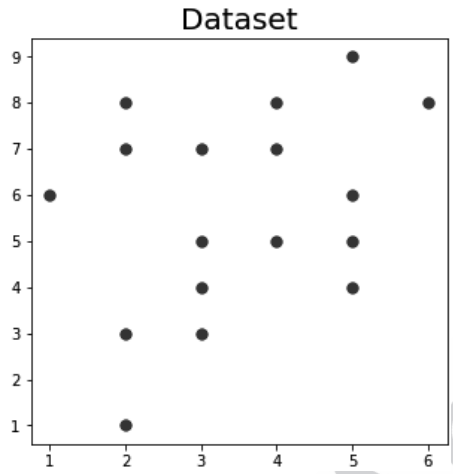
	x	y
0	4	5
1	1	6
2	2	7
3	6	8
4	4	8
5	5	4
6	2	3

Gambar 4.23 Hasil Nilai

11. Untuk mendapatkan gambaran dari data, kita perlu membuat scatter plotting dari dataset yang telah berhasil diambil nilainya pada cell sebelumnya. Ketikan perintah berikut 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 4.24 Perintah Gambar Scatter Plotting Dataset



Gambar 4.25 Gambar Scatter Plotting Dataset

12. Selanjutnya melakukan training model terhadap data dengan mendefinisikan nilai epsilon yang kita inginkan pada parameter eps dan nilai core point pada sample pada parameter min_samples.

```
db = DBSCAN(eps = 1.1, min_samples = 3)
db.fit(df)
```

Gambar 4.26 Training Model

13. Untuk melihat label yang terbentuk pada training DBSCAN adalah sebagai berikut.

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

Gambar 4.27 Perintah Melihat Label

Hasilnya adalah sebagai berikut.

```
array([ 0, -1,  1, -1,  1,  0,  0,  1,  0, -1,  0,  0,  1,  1, -1,  0,  0])
```

Gambar 4.28 Hasil Melihat Label

14. Untuk melihat jumlah cluster yang terbentuk adalah sebagai berikut.

```
n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
n_clusters
```

Gambar 4.29 Perintah Melihat Jumlah Cluster

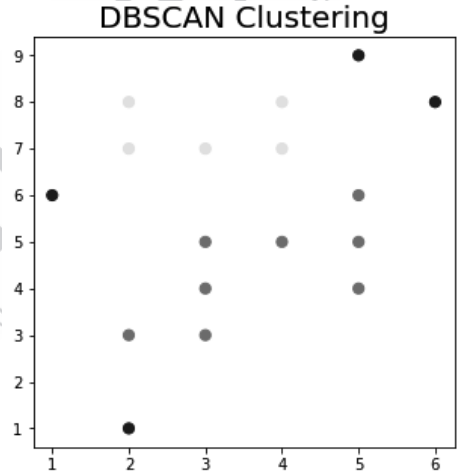
(-1) tidak termasuk ke dalam cluster karena merupakan noise. Dari kode tersebut, cluster yang terbentuk adalah 2 cluster.

15. Langkah terakhir 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.title('DBSCAN Clustering',fontsize=20)
plt.show()
```

Gambar 4.30 Perintah Visualisasi Hasil Clustering

Hasil visualisasi data hasil clustering adalah sebagai berikut.



Gambar 4.31 Visualisasi Hasil Clustering

Dapat dilihat bahwa metode DBSCAN membentuk 2 cluster dan terdapat noise yang ditandai dengan point berwarna ungu.

4.2 METODE DENCLUE

Algoritma DENCLUE (Density of Clustering) diperkenalkan pertama kali oleh Hinneburg dan Keim pada tahun 1998 pada seminar internasional "Knowledge Discovery and Data Mining". Algoritma ini dapat mengelompokkan data dengan jumlah data noise yang besar dan menunjukkan beberapa keunggulan dibandingkan DBSCAN. Algoritma DENCLUE memiliki dasar matematika yang kuat dan memiliki kemampuan lebih cepat dibandingkan algoritma yang sudah ada. Selain itu, juga memungkinkan deskripsi matematis yang lebih ringkas terkait pengelompokan data bagaimanapun bentuk datanya (Hinneburg and Keim, 1998).

Cluster dapat diidentifikasi secara matematis dengan menentukan density attractors. Density attractors merupakan local maxima dari semua fungsi densitas. Setiap data dimodelkan fungsi matematisnya yang disebut "fungsi pengaruh". Fungsi pengaruh juga dapat dilihat sebagai fungsi yang menggambarkan dampak dari titik data beserta tetangganya. Fungsi pengaruh ini merupakan fungsi sembarang (Hinneburg and Keim, 1998).

4.2.1 Fungsi Pengaruh

Seperti yang dijelaskan sebelumnya, fungsi pengaruh merupakan fungsi sembarang. Contoh fungsi pengaruh dapat merupakan fungsi parabola, fungsi square, atau fungsi Gaussian. Berikut beberapa contoh fungsi pengaruh.

1. Fungsi pengaruh square wave:

$$f_{square}(x,y)=\begin{cases}0 & jika\ d(x,y)>\sigma\\ 1 & jika\ d(x,y)<\sigma\end{cases}$$

2. Fungsi pengaruh Gaussian:

$$f_{Gauss}(x,y)=e^{-\frac{d(x,y)^2}{2\sigma^2}}$$

4.2.2 Fungsi Densitas

Fungsi densitas merupakan jumlah fungsi pengaruh dari semua data. Misalkan, fungsi pengaruh didefinisikan:

$$f_B^y(x)=f_B(x,y) \tag{4.2}$$

jika terdapat N objek data yang dijelaskan dalam satu himpunan $D=\{x1,\ldots,xn\}\subset F^d$ maka densitas fungsi didefinisikan (Hinneburg and Keim, 1998).

$$f_B^D(x)=\sum_{i=1}^{N} f_B^{xi}(x) \tag{4.3}$$

4.2.3 Penarik Densitas

Suatu titik x^* disebut penarik densitas dari fungsi pengaruh jika x^* merupakan lokal maksimal dari fungsi densitas $f_B^D(x)$. Titik $x\in F^d$ disebut densitas yang ditarik oleh penarik densitas x^*. x^* dapat didefinisikan dengan (Hinneburg and Keim, 1998):

$$x^0=x, x^i=x^{i-1}+\delta.\frac{\nabla f_B^D(x^{i-1})}{||\nabla f_B^D(x^{i-1})||} \tag{4.4}$$

jika $\exists k\in N: d(x^k,x^*)\le\in$, dan

$$\nabla f_B^D(x)=\sum_{i=1}^{N}(x_i-x).f_B^{x_i}(x) \tag{4.5}$$

Sebuah cluster dengan parameter σ dan ξ ditentukan oleh pusatnya. Untuk pusat cluster x^* dari himpunan data C, di mana C bagian dari data D maka cluster dibentuk oleh x yang merupakan anggota C dan merupakan densitas yang ditarik oleh x^* serta nilai $f_B^D\left(x^*\right) \geq \xi$. Sementara itu, outlier didefinisikan sebagai titik x yang merupakan anggota dari data D yang merupakan densitas yang ditarik oleh suatu penarik densitas x^*, tetapi $f_B^D\left(x^*\right) < \xi$.

4.2.4 Algoritma DENCLUE

Ada dua tahapan dalam algoritma DENCLUE dan dua parameter masukan yang ditetapkan parameter σ dan ξ (Awalludin *et al.*, 2018). Adapun dua tahapan dari algoritma, yaitu sebagai berikut.

1. Membangun map dengan menghubungkan populasi.

   Untuk membangun sebuah map dengan menghubungkan suatu populasi dengan menghitung jarak setiap data dengan data lainnya. Penghubungan dari setiap area populasi kubus dengan $d\left(x_j, x_i\right) \leq 4\sigma$.

2. Tahap pengelompokan.

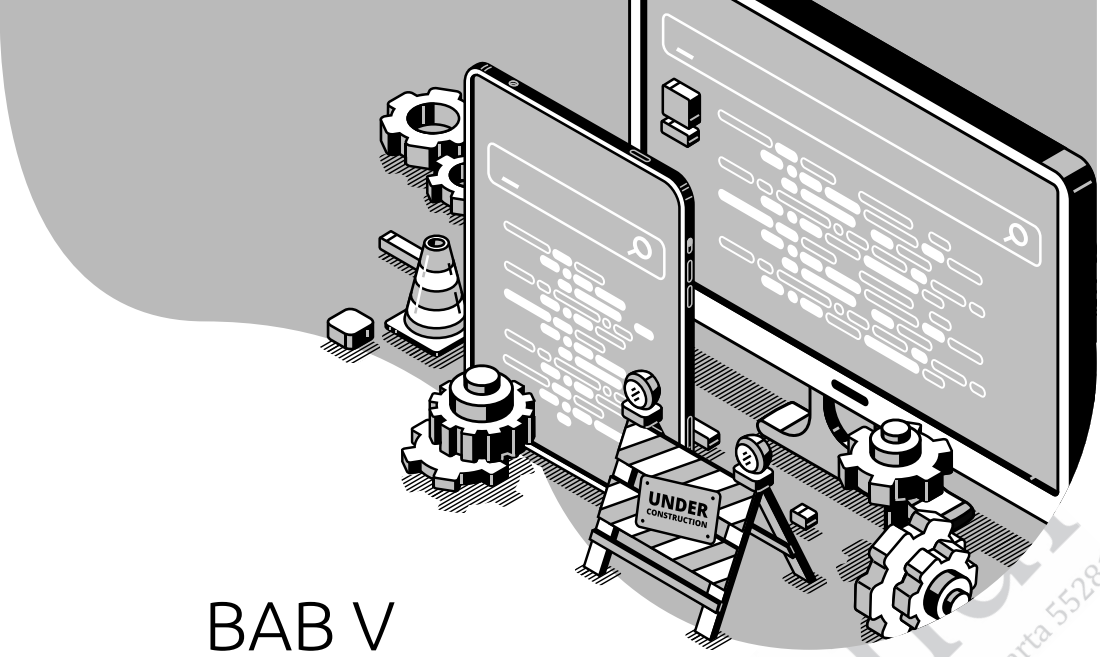
   Hanya populasi kubus yang memiliki densitas padat yang dianggap sebagai cluster. Selanjutnya, menentukan nilai penarik densitas x^* sesuai dengan persamaan (4.4) dan perhitungan berhenti ketika $f^D\left(x^{k+1}\right) < f^D\left(x^k\right)$ maka $x^* = x^k$.

4.3 LATIHAN SOAL CLUSTERING BERBASISKAN DENSITY

1. Jelaskan perbedaan antara metode DBSCAN dengan DENCLUE dalam menentukan outlier!
2. Kelompokan data berikut ini menggunakan metode DBSCAN dengan MinPts=3 dan eps=1,5!

Data ke-	x	y
1	1	1
2	1	5
3	2	1
4	2	2
5	2	5
6	2	6
7	3	2
8	3	4
9	3	6
10	4	7
11	5	6
12	6	5
13	7	4
14	8	5
15	9	2

3. Dalam pengelompokan menggunakan DBSCAN, apa pengaruh parameter MinPts dan eps terhadap hasil pengelompokan?
4. Dalam pengelompokan menggunakan DENCLUE, terdapat dua tahap. Jelaskan dua tahap tersebut!
5. Apa peranan parameter σ dan ξ dalam proses pengelompokkan menggunakan metode DENCLUE?



BAB V
METODE CLUSTERING BERBASISKAN MODEL

Capaian Pembelajaran:

1. Mampu memahami konsep dari clustering berbasiskan model.
2. Mampu memahami konsep dan algoritma dari model-model clustering berbasiskan model.
3. Mampu menerapkan algoritma dari model-model clustering berbasiskan model untuk memecahkan masalah.

Clustering based on model (model clustering) sering disebut dengan Soft Clustering atau Heuristic Clustering merupakan salah satu jenis Analisis Clustering dari materi Data Mining. Seperti yang disampaikan sebelumnya, jenis clustering ini merupakan jenis yang memiliki banyak nama, hal ini karena cakupan jangkauan yang cukup besar. Disebut clustering berdasarkan model karena output clustering yang dihasilkan berdasarkan model yang dibuat. Dikatakan sebagai Soft Clustering karena dalam menemukan solusinya, metode-metode yang termasuk dalam jenis ini didapatkan secara perlahan atau “soft”.

Hal tersebut berbeda dengan yang dikerjakan oleh partial Clustering (seperti K-Means, K-Means termasuk Hard Clustering) karena penemuan centroid-nya di-adjust secara langsung pada mean data. Nama lain dari model clustering adalah Heuristic Clustering. Arti kata Heuristic sendiri adalah pendekatan, jadi yang dikerjakan metode ini adalah dengan teknik pendekatan atau terdapat selisih error yang selalu diperkecil dengan berjalannya iterasi. Pengertian yang terakhir sangat dekat dengan pengertian Soft Clustering.

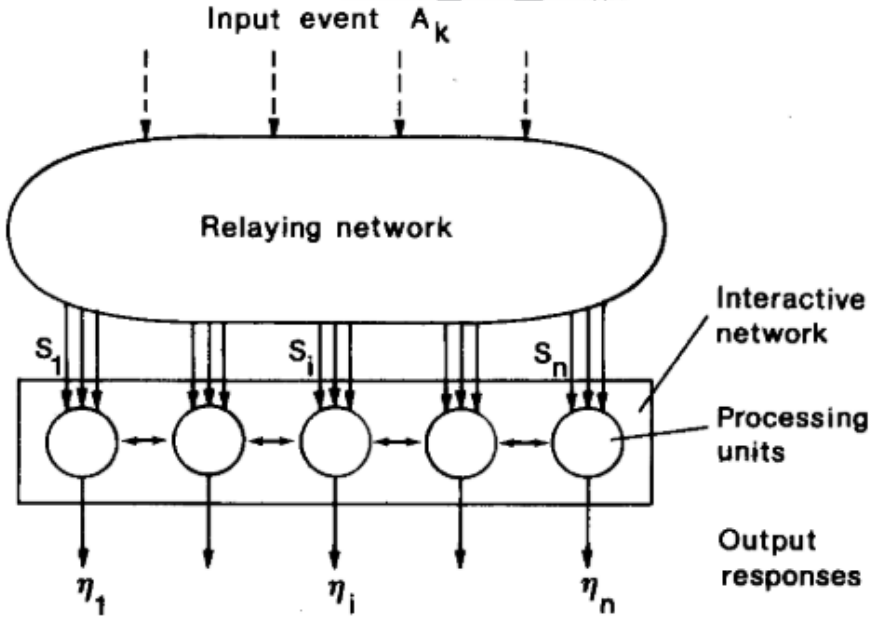
Apabila dilihat dari definisi model clustering, terdapat beberapa metode yang dapat masuk dalam jenis ini. Self Organized Map (SOM) merupakan metode yang paling populer dari tipe clustering ini. SOM merupakan metode yang memiliki arsitektur sama dengan Artificial Neural Network, sedangkan Expectation Maximization (EM) merupakan metode clustering yang mempunyai dasar dari disiplin ilmu Statistik. Distribusi Gaussian menjadi kunci utama dalam metode ini. Metode yang paling baru dalam jenis ini adalah optimization clustering. Metode ini merupakan metode optimasi yang digunakan untuk analisis clustering.

Apabila pada umumnya metode optimasi digunakan untuk mengoptimalkan sebuah fungsi dan kombinasi, dalam pembahasan ini metode optimasi digunakan untuk clustering. Karena hal ini merupakan sesuatu yang cukup baru, buku ini akan membahas detail tentang konsep, hitungan manual, dan program pada metode Optimasi untuk clustering.

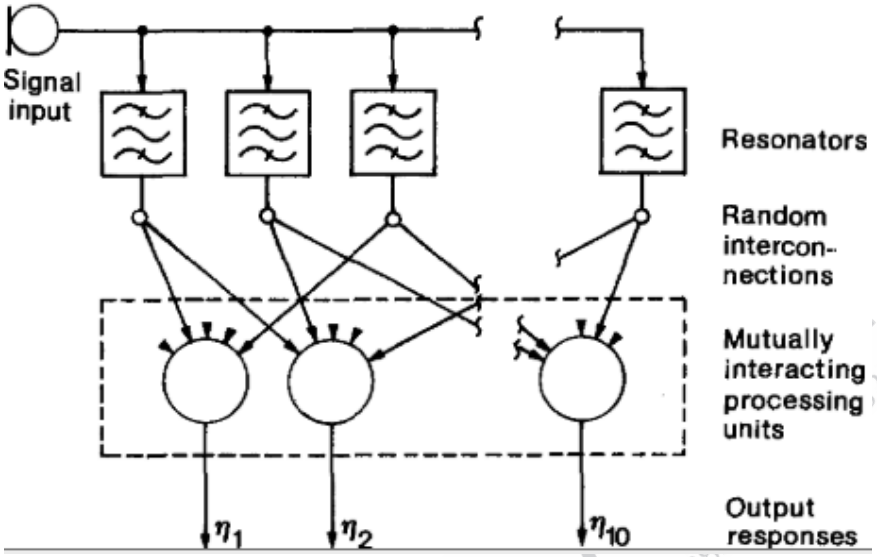
5.1 SELF ORGANIZED MAPS

5.1.1 Konsep Algoritma

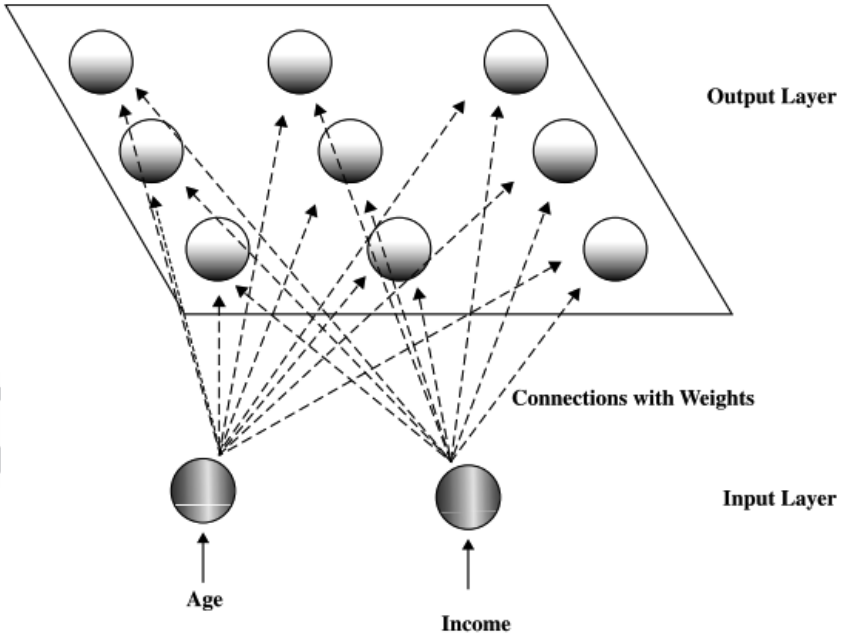
Self Organized Map berawal dari manuskrip berjudul *Self-Organized Formation of Topologically Correct Feature Maps* dari jurnal *Biological Cybernetics* yang diterbitkan oleh Publisher Springer–Verlag pada tahun 1982 dan ditulis oleh Teuvo Kohonen (Kohonen, 1982). Manuskrip tersebut menerangkan bahwa SOM merupakan jaringan sederhana yang memiliki input dan signal yang merepresentasikan secara otomatis ke dalam maps. Secara singkat, hal ini menciptakan sebuah formasi map yang mempresentasikan input data. Arsitektur SOM pada manuskrip aslinya, ilustrasi kerangka kerja dari SOM dapat dilihat pada Gambar 5.1 dan Arsitektur pada array 1D dapat dilihat pada Gambar 5.2.



Gambar 5.1 Ilustrasi SOM (Kohonen, 1982)



Gambar 5.2 Arsitektur SOM (Kohonen, 1982)



Gambar 5.3 Topologi SOM (Larose, 2014)

Gambar 5.3 adalah contoh topologi yang diterapkan pada dua buah fitur data (age dan income) dengan 9 buah unit (centroid). Setiap fitur terhubung dengan semua unit yang ada sehingga total nilai bobot yang terjadi adalah unit x fitur = 9 x 2 = 18 nilai bobot.

Secara garis besar, konsep dari SOM adalah menggunakan jaringan Perceptron sebagai arsitekturnya. Namun, tanpa adanya target data, metode ini sering disebut sebagai Artificial Neural Network Unsupervised. Tentu saja sebuah jaringan syarat tiruan tidak akan dapat bekerja tanpa adanya target data, oleh sebab itu target data dalam SOM digantikan dengan Proses competition. Setelah proses Competition selesai dilanjutkan dengan proses cooperation dan proses Adaptation (Larose, 2014). Berikut adalah penjelasan untuk setiap proses utama dalam SOM.

Competition

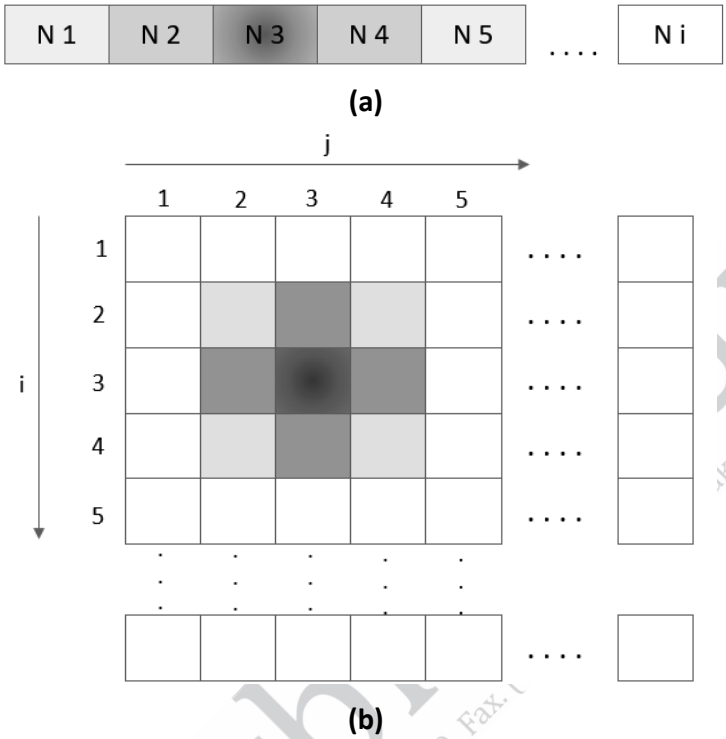
Proses ini merupakan pengganti dari target data. Dalam proses ini akan terjadi proses kompetisi di antara bobot-bobot (Processing Units). Unit yang dipilih menjadi pemenang adalah unit dengan jarak terkecil dari input yang diberikan, sedangkan jarak yang digunakan adalah Euclidean Distance. Unit yang terpilih akan mendapatkan keistimewaan, yaitu mengubah nilai unitnya (update bobot). Persamaan Euclidean Distance dapat dilihat dalam persamaan 5.1, Persamaan tersebut merupakan persamaan untuk mencari nilai Distance (Jarak) untuk satu cluster C, terhadap satu baris i Data X terhadap semua fitur j yang digunakan.

$$\text{Distance}_k = \sqrt{\sum_{j=1}^{n} \left(X_{ij} - C_{kj}\right)^2} \qquad (5.1)$$

Cooperation

SOM ini dapat dikatakan sebagai sesuatu yang spesial dan berbeda dengan metode clustering lainnya karena adanya proses cooperation atau lebih umum kata "sharing" (berbagi). Unit yang menjadi pemenang akan membagikan kemenangannya dengan tetangganya sehingga akan terdapat hyperparameter dalam proses ini, yaitu topologi tetangga dan berapa tetangga yang dibagikan (topologi tetangga dapat dilihat pada Gambar 5.4). Dalam contoh gambar tersebut, N 3 yang menjadi pemenang dan jika besar tetangga yang diterapkan adalah 2, tetangga yang lain (N1,N2,N4,N5) akan juga mengalami perubahan nilai. Tetangga paling dekat (N2,N4) akan mendapatkan nilai lebih besar dari tetangga yang lebih jauh (N1,N5) jika terdapat tetangga yang nilainya terlalu jauh dan di luar radius tetangga yang telah ditetapkan, nilai unit tersebut tidak akan di-update. Nilai ini digunakan untuk meng-update nilai unit tersebut.

Selain konsep sharing untuk 1 dimensi, konsep sharing juga dapat digunakan model 2 dimensi (Gambar 5.4 (b)). Jika dalam 1 dimensi konsep tetangga yang dimiliki hanya kanan dan kiri, konsep tetangga 2 dimensi mempunyai posisi sumbu x serta sumbu y, atau jika diterapkan pada array punyai nilai i dan j.



Gambar 5.4 Topologi Tetangga (a) 1 Dimensi, (b) 2 Dimensi

Adaptation

Seperti dalam Proses Cooperation sebelumya, proses ini merupakan proses update nilai unit. Persamaan untuk update unit terlihat dalam persamaan 5.2.

$$W_{baru} = W_{lama} + \theta . \alpha_t . (X_t - W_{lama}) \quad (5.2)$$

Di mana:

W_{lama} = bobot / nilai unit

θ = merupakan hasil dari fungsi tetangga

α_t = nilai learning rate pada iterasi ke t

X_t = nilai Data yang pada waktu diproses

Variabel θ merupakan hasil dari koefisien fungsi ketetanggaan. Fungsi yang digunakan berdasarkan fungsi Gaussian (seperti pada persamaan 5.3), di mana $||r_1 - r_2||^2$ merupakan jarak kuadrat antara dua nilai unit, r_1 merupakan posisi unit pemenang dan r_2 merupakan posisi unit tetangga, sedangkan δ_t merupakan nilai parameter sigma. Nilai ini merupakan hyperparameter dari algoritma SOM dan nilainya terus berkurang mengikuti iterasi yang telah dilakukan. Untuk mendapatkan nilai δ_t yang adapatif dengan iterasi dapat dilihat dalam persamaan 5.4. Dalam persamaan 5.4 variabel δ merupakan inisialisasi radius awal tetangga dan t merupakan iterasi waktu itu dan T merupakan total iterasi.

$$\theta = e^{-\frac{||r_1 - r_2||^2}{2\delta_t}} \quad (5.3)$$

$$\delta_t = \delta\left(1 - \frac{t}{T}\right) \quad (5.4)$$

Perubahan nilai learning rate akan terus berkurang sesuai dengan total iterasi. Persamaan yang digunakan hampir sama dengan persamaan 5.4 dengan pergantian variabel δ diganti dengan α.

5.1.2 Pembuatan Program Self Organized Maps

Algoritma Self Organized Maps:

1. Inisialisasi parameter jumlah unit, sigma, maximum iterasi, alfa, radius tetangga, topologi ketetanggaan.
2. Generate nilai dari masing-masing unit dengan nilai random.
3. Lakukan perulangan sampai dengan batas maximum iterasi.
   a. Update parameter sigma, maximum tetangga, dan alfa.
   b. Lakukan perulangan sampai dengan jumlah Data.
      1) Cari unit pemenang.
      2) Update nilai unit pemenang dan unit yang dianggap sebagai tetangga.

Untuk membuat program Self Organized Maps, pada contoh ini akan menggunakan Google Colabs. Berikut adalah langkah-langkah yang dapat digunakan.

1. Buat file Excel dengan attribute (kolom) dan nilai data (baris), seperti gambar berikut ini, serta simpan dengan format .xlsx.

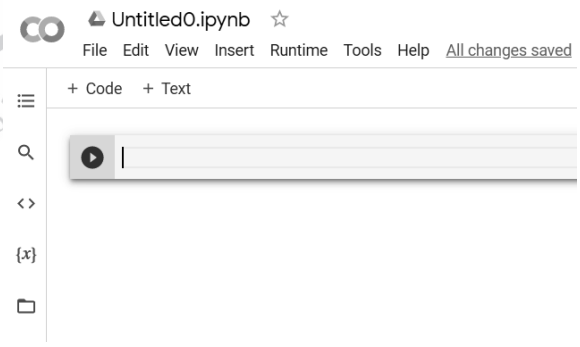
	A	B
1	x	y
2	4	5
3	1	6
4	2	7
5	6	8
6	4	8
7	5	4
8	2	3
9	3	7
10	3	3
11	5	9
12	3	4
13	3	5
14	2	8
15	4	7
16	2	1
17	5	6
18	5	5

Gambar 5.5 File Excel

2. Buka Google Colab dengan alamat:

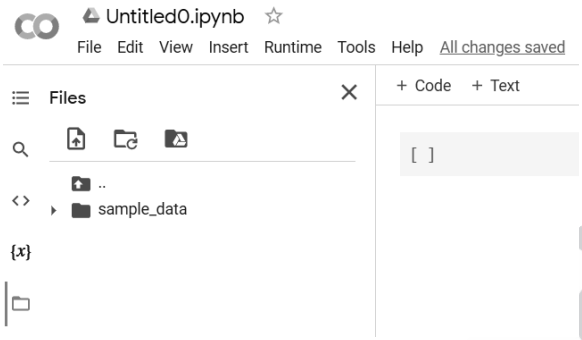
   https://colab.research.google.com/drive/1Q26epc2MJ5fGNa3d-Qs-tF7gJM_2N5aWZ?usp=sharing.

3. Masuk menu file, kemudian pilih new notebook maka akan tampil, seperti pada gambar berikut ini.



Gambar 5.6 New Notebook

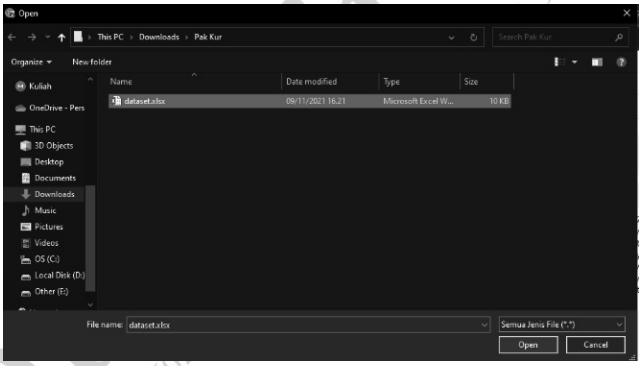
4. Tekan button folder seperti gambar berikut.



Gambar 5.7 Button Folder

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

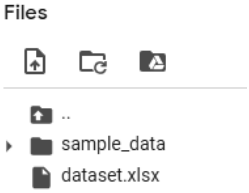
   Setelah menekan button 4 akan muncul window dialog.



Gambar 5.8 Window Dialog

   Pilih file yang telah dibuat sebelumnya dan tekan button open.

6. Klik open dan pastikan sudah di-import



Gambar 5.9 Import Dataset

7. Pada baris pertama kode ketikan perintah:

   Pertama tulis kode untuk menghilangkan warning pada cell:

   import warnings

   warnings.filterwarnings('ignore')

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

Gambar 5.10 Warning

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

8. Tuliskan code pada cell selanjutnya dengan perintah:

   !pip install sklearn-som

9. Import library-library yang dibutuhkan untuk program Self Organized Map ini.

```
from matplotlib import pyplot as plt
import pandas as pd
from sklearn_som.som import SOM
from seaborn import scatterplot as scatter
```

Gambar 5.11 Import Library SOM

   Library pandas digunakan untuk mengambil atau mengekstrak nilai dari file Excel atau csv ke dalam Data Frame Python. Library matplolib dan seaborn digunakan untuk menampilkan plotting dan scatter data. Library numpy untuk membuat data menjadi array. Library SOM untuk clustering dengan Metode Self Organized Maps.

10. Pada cell selanjutnya ketikan perintah untuk mengambil data dari file Excel, seperti code berikut ini.

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

Gambar 5.12 Perintah Mengambil Data

Setelah di-run 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

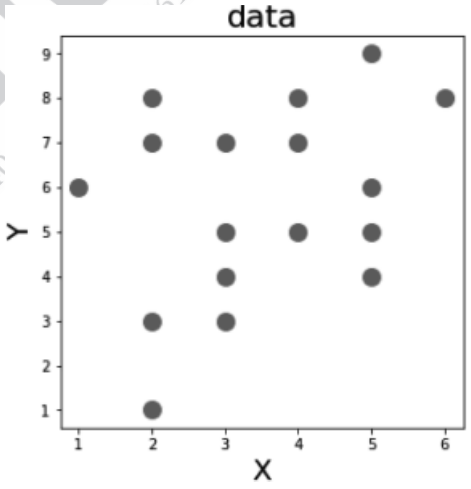
Gambar 5.13 Hasil Nilai

11. Selanjutnya, kita perlu membuat scatter plotting dari dataset yang telah berhasil diambil nilainya pada cell sebelumnya. Ketikan perintah berikut ini untuk mendapatkan gambar scatter plotting dataset.

```
plt.figure(figsize=(5,5))
scatter(df.iloc[:,0].values, df.iloc[:,1].values, marker='o', s=200)
plt.xlabel('X', fontsize=20)
plt.ylabel('Y', fontsize=20)
plt.title('data', fontsize=22)
plt.show()
```

Gambar 5.14 Scatter Plotting Dataset

Hasil Plotting:



12. Ubah dataframe menjadi array numpy

```
df_array= df.to_numpy()
df_array
```

Gambar 5.15 Array Numpy

Hasilnya:

```
array([[4, 5],
       [1, 6],
       [2, 7],
       [6, 8],
       [4, 8],
       [5, 4],
       [2, 3],
       [3, 7],
       [3, 3],
       [5, 9],
       [3, 4],
       [3, 5],
       [2, 8],
       [4, 7],
       [2, 1],
       [5, 6],
       [5, 5]])
```

13. Selanjutnya melakukan training model data dari array tadi dengan cara sebagai berikut.

```
som = SOM(m=3, n=1, dim=2)
som = som.fit_predict(df_array)
som
```

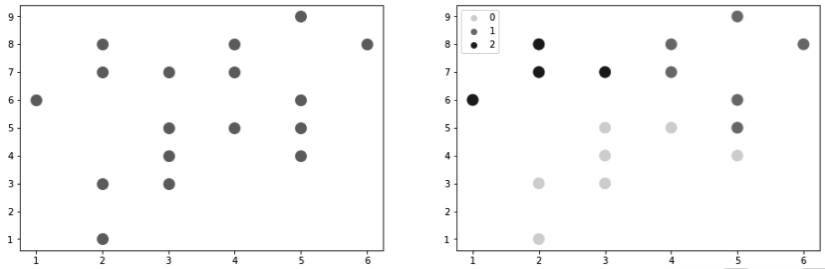
Hasil berupa label:

```
array([0, 2, 2, 1, 1, 0, 0, 2, 0, 1, 0, 0, 2, 1, 0, 1, 1])
```

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

```
f, axes = plt.subplots(1, 2, figsize=(16,5))

scatter(df.iloc[:,0].values, df.iloc[:,1].values, ax=axes[0], s=200)

scatter(df.iloc[:,0].values, df.iloc[:,1].values, ax=axes[1], hue=som, s=200)

plt.show()
```

Hasil visualisasi data hasil clustering adalah sebagai berikut.



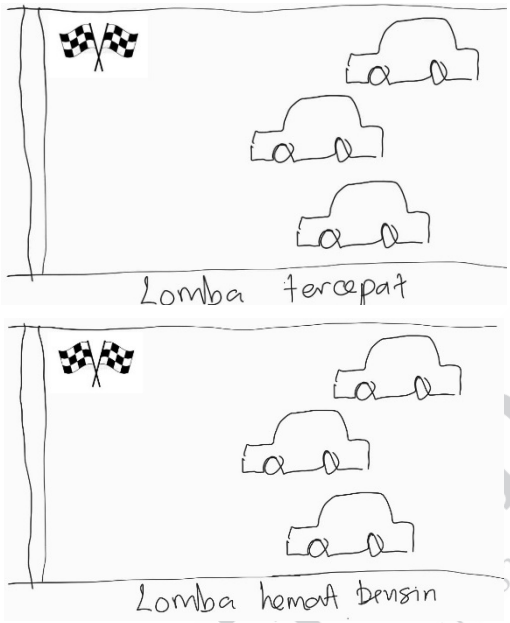
Grafik sebelah kiri merupakan sebaran awal dan grafik sebelah kanan merupakan hasil cluster serta terdapat 3 cluster.

5.2 OPTIMIZATION CLUSTERING

Seperti penjelasan sebelumnya mengenai jenis clustering dengan pemodelan, pendekatan ini menggunakan metode optimasi. Oleh sebab itu, ada beberapa penjelasan pendahuluan sebelum menuju ke analisis clustering. Pembahasan pertama mengenai metode optimasi secara umum, setelah itu, pembahasan metode optimasi metaheuristik, pembahan salah satu metode optimasi metaheuristik, dan yang terakhir adalah implementasi metode optimasi metaheuristik dalam clustering.

Optimasi/Optimization

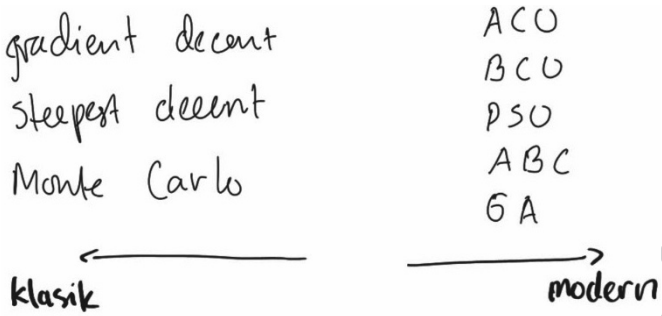
Optimasi merupakan metode yang digunakan untuk mengoptimalkan sebuah fungsi. Fungsi ini biasa disebut dengan Objective Function atau Fitness Function. Kata mengoptimalkan dapat diartikan dalam dua definisi. Dalam arti populer, memaksimalkan atau meminimalkan memiliki arti maksimasi dan minimasi. Maksimasi dapat diartikan semakin tinggi nilai yang didapatkan dari sebuah fungsi maka semakin optimal nilai tersebut. Demikian juga dengan minimasi, semakin kecil nilai yang dihasilkan oleh fungsi maka nilai tersebut akan semakin optimal.



Gambar 5.16 Maksimasi dan Minimasi

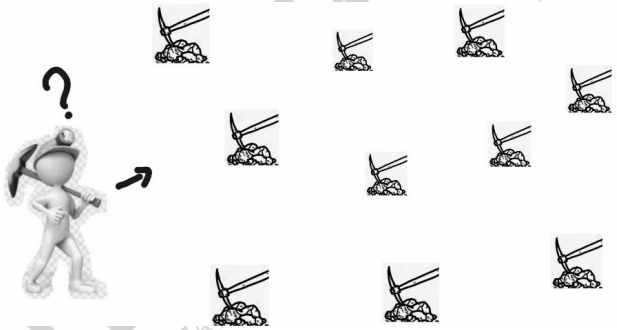
Dalam proses pengoptimalan terkadang terdapat batasan sebuah kondisi atau nilai, batasan ini biasa disebut dengan fungsi penalti atau constraint. Namun demikian, constraint ini tidak harus ada dalam pencarian sebuah nilai yang optimal.

Dilihat dari waktunya, metode optimasi dibagi menjadi dua, yaitu klasik dan modern. Metode-metode optimasi klasik umumnya ditandai dengan pencarian tunggal. Metode optimasi klasik yang paling sering kita dengar, seperti metode Monte Carlo, Gradient Descent, Steepest Descent, Golden Section Search, dan masih banyak yang lain. Sementara itu, metode optimasi modern memiliki ciri utamanya dengan menggunakan multisearch.



Gambar 5.17 Klasik dan Modern Optimasi

Metaheuristik merupakan gabungan dari Heuristik dan Meta. Heuristik dapat diartikan sebagai pendekatan dan Meta dapat diartikan advance. Heuristik memiliki arti pendekatan. Pertanyaannya adalah kenapa dilakukan pendekatan dan apa yang harus dilakukan pendekatan?



Gambar 5.18 Heuristic Optimization

Ilustrasi Gambar 5.18 menunjukkan seorang penambang yang mencari minyak dalam kurun waktu tertentu. Penambangan harus mencari tempat minyak terbanyak untuk menghasilkan nilai terbaik, tetapi waktu yang diberikan oleh investor hanya 3 hari. Sementara itu, waktu untuk menggali satu tempat adalah 12 jam, sedangkan jumlah tempat penambangan ada 10. Secara matematis, waktu yang ada tidak akan cukup untuk melihat semua kandungan minyak di semua

tempat sehingga pendekatan nilai terbaik atau nilai sejati tidak dapat dilakukan dalam ilustrasi tersebut. Dibutuhkan sebuah metode untuk mendapatkan minyak terbanyak dengan kurun waktu yang terbatas. Meskipun tidak menemukan solusi sejati, tetapi dapat menemukan solusi terbaik.

Ilustrasi Gambar 5.18 dapat diartikan bahwa pendekatan heuristik adalah teknik dalam menemukan solusi terbaik dengan waktu yang dapat diterima "acceptable". Teknik ini mungkin mengorbankan solusi sejati, tetapi digantikan dengan waktu yang cukup. Semua metode optimasi dapat dilandasi dengan cara ini, baik yang klasik, modern, atau heuristik, dan metaheuristik.

Setelah kita membahas sedikit tentang heuristik, kemudian akan membahas mengenai metaheuristik? Metaheuristik merupakan metode optimasi heuristik yang dilakukan dengan cara yang lebih "advance" sehingga hasil yang diperoleh dari metode ini pada umumnya lebih baik dari metode heuristik. Lantas, bagaimana cara yang lebih "advance" bisa mengubah hasil menjadi lebih baik? Pertanyaan tersebut berhubungan dengan pertanyaan utama dalam pembahasan ini, yaitu apa/bagaimana yang dimaksud cara lebih "advance"? Cara yang lebih "advance" memiliki berbagai ciri, antara lain agen dalam pencarian tidak satu (single searching), melainkan multiagent (multi-searching) dan agen-agen tersebut saling berkomunikasi atau bekerja sama untuk mendapatkan hasil yang paling optimal. Hal-hal tersebut menjadi ciri utama dalam metode optimasi metaheuristik. Ciri lain yang harus ada adalah bekerja dengan nilai acak (random/stochastic). Metode optimasi metaheuristik yang paling sering kita dengar adalah Genetic Algorithm (GA) yang didasari dari teori evolusi dari biologist Charles Darwin. Metode GA memiliki tiga proses utama, yaitu Elitism, Crossover, dan Mutation. Elitisme (elitism) merupakan proses yang

akan mempertahankan individu sebagai menjadi pemenang, sedangkan crossover merupakan proses kawin silang untuk mendapatkan individu yang lebih baik. Sementara itu, proses terakhir adalah mutasi individu yang bertujuan mendapatkan keragaman individu.

Selain GA, terdapat metode-metode berdasarkan makhluk hidup atau kejadian sekitar kita (nature inspire). Misalnya, terdapat metode berdasarkan lebah, seperti Bee Colony Optimization dan Artificial Bee Colony. Metode optimasi berdasarkan semut, yaitu metode Ant Colony Optimization dan optimasi berdasarkan kawanan atau Particle Swarm Optimization (PSO) yang menjadi metode favorit. Metode PSO merupakan metode yang termasuk dalam Swarm Intelligence selain ACO, BCO, ABC, dan masih banyak yang lain.

Dalam bab ini, metode optimasi metaheuristik yang akan dipilih adalah Particle Swarm Optimization yang digunakan untuk clustering. Pertimbangannya adalah metode PSO menjadi sangat populer dalam 10 tahun terakhir ini dalam dunia penelitian. PSO merupakan metode yang memiliki cara kerja paling sederhana, tetapi memiliki akurasi atau hasil yang cukup bagus jika dibandingkan dengan metode yang mempunyai komputasi lebih berat dari metode PSO. Alasan terakhir yang menjadi alasan utama adalah metode ini memiliki kesederhanaan logika (cara kerja). Kesederhanaan tersebut akan memudahkan dalam melakukan modifikasi/menerapkan dalam studi kasus atau objek-objek yang berbeda, termasuk dalam analisis clustering.

Particle Swarm Optimization (PSO)

Konsep logika dalam metode ini adalah menirukan kawanan burung yang mencari makan (Kurniawan and Suciati 2018), seperti terlihat di Gambar 5.19. Gambar tersebut adalah ilustrasi sebuah kawanan burung yang saling bekerja sama.

Gambar 5.19 Kawanan Burung

Ketika ada seekor burung yang mendapatkan makanan terbanyak, burung tersebut akan membagikan lokasinya terhadap seluruh kawanan sehingga nantinya kawanan yang lain akan menuju pada posisi yang sama dengan burung pertama tadi.

Gambar 5.20 Penentuan Burung yang Mempunyai Lokasi Terbaik

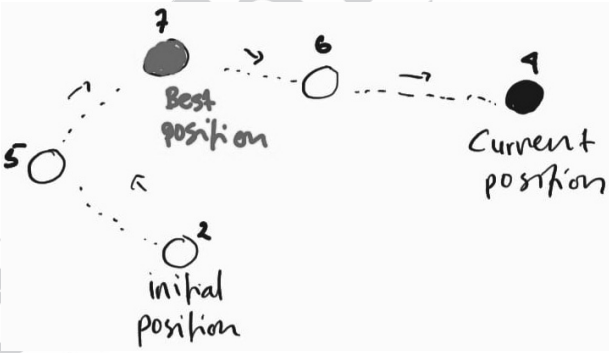
Bagaimana jika dalam perjalanan terdapat burung lain yang mendapatkan lokasi lebih baik (sumber makanan lebih banyak)? Burung tersebut akan membagikan lokasi tempat lebih banyak sumber makanan dan kawanan burung-burung yang lain akan melakukan proses yang sama, yaitu terbang menuju burung dengan lokasi lebih baik. Ketika semua burung telah terkumpul dan makan, selesailah proses kawanan burung tersebut mencari makanan.

Terdapat dua proses utama yang harus dikerjakan dalam metode ini, yaitu proses perhitungan Kecepatan (Velocity) dan perubahan Posisi (update Position). Formula untuk menghitung dua proses tersebut dapat dilihat dalam persamaan berikut.

$$V_i(t) = V_i(t-1) + c_1 r_1 \left(X_i^P - X_i(t-1)\right) + c_2 r_2 \left(X^G - X_i(t-1)\right)$$

$$X_i(t) = V_i(t) + X_i(t-1)$$

Selanjutnya akan dijabarkan satu per satu notasi persamaan di atas. Namun, sebelumnya perhatikan ilustrasi Gambar 5.21. Ilustrasi tersebut menggambarkan setiap burung/particle akan memiliki tiga posisi. Posisi pertama adalah posisi awal (initial position), kemudian posisi sekarang (current position), dan posisi terbaik (best position). Initial position merupakan posisi awal dari sebuah burung untuk memulai perjalanannya. Posisi awal ini dibangkitkan oleh bilangan random pada luas area yang telah ditentukan.



Gambar 5.21 Ilustrasi Pbest dan Current Position

X merupakan posisi dari setiap burung/particle dan X_i merupakan current position (posisi sekarang). Berdasarkan ilustrasi Gambar 5.21, current position adalah node yang berwarna hitam, sedangkan nilai di atas node merupakan contoh nilai fitness yang diperoleh. Sementara itu, notasi i merupakan individu/particle/burung ke-i.

Jika kita perhatikan Gambar 5.21, setiap node akan melakukan perjalanan dari initial position ke current position dan saat perpindahan posisi, setiap particle akan menghitung nilai fitness-nya. Hal ini ditandai dengan nilai di atas node tersebut. Contohnya ilustrasi di atas 5 kali perpindahan posisi dari satu node dengan nilai [2, 5, 7, 6, dan 4], setiap particle/node akan menyimpan memory kapan posisi terbaiknya. Keadaan posisi terbaik digambarkan dengan node dengan warna merah dengan nilai fitness 7 (yang berarti paling maksimal) dan pada persamaan untuk mencari Velocity dengan notasi X_i^P (notasi ^{P} diartikan sebagai Pbest (position best)).

Lantas, apa makna persamaan untuk mencari velocity tersebut? Velocity atau kecepatan diperoleh dengan mencari selisih nilai Gbest (Global best) dengan posisi sekarang ditambahkan posisi terbaik dari particle (Pbest), kemudian dikurangi posisi sekarang sehingga akan didapat nilai untuk arah dan kecepatan. Sementara itu, untuk perpindahan posisi (update position) cukup hanya menambahkan posisi lama $X_i(t-1)$ dengan hasil perhitungan velocity $V_i(t)$.

Untuk lebih memahami Metode PSO, kita harus mencoba menghitung secara manual. Langkah pertama yang harus dilakukan adalah menentukan Objective Function terlebih dahulu dengan menggunakan fungsi maksimasi atau minimasi berapa dimensi attribute particle yang dibutuhkan untuk menyelesaikan persamaan tersebut.

Rosenbrock Function (minimasi):

$$f = 100*((x_1^2 - x_2)^2 - (x_1 - 1)^2$$

Rosenbrock ini merupakan fungsi standar yang digunakan untuk menunjukkan metode optimasi. Nilai optimal dari fungsi ini bernilai 0, dengan range nilai x berada pada range [-5 10]. Contoh perhitungan manual dari fungsi di atas sebagai berikut.

Semisal, nilai x yang kita masukan adalah [-2 1] atau $x_1 = -2$ dan $x_2 = 2$:

$$f = 100*\left(x_1^{\,2} - x_2 \right)^2 - \left(x_1 - 1 \right)^2$$

$$f = 100*\left(-2^2 - 2 \right)^2 - \left(-2 - 1 \right)^2$$

$$f = 100*\left(4 - 2 \right)^2 - \left(-2 - 1 \right)^2$$

$$f = 100*\left(2 \right)^2 - \left(-3 \right)^2 = 100*4 - 9 = 400 - 9$$

$$f = 391$$

Selanjutnya, bagaimana implementasikan pada code Pyhton? Sampai dengan perhitungan atau rumus untuk menghitung posisi, kita bisa melihat programnya pada file pso.py (Code x.17). Lantas bagaimana membangun program tersebut secara perlahan atau step by step dari paling bottom to up? Kita harus memperhatikan rangkaian pembuatan program dari Code 5.1 sampai Code 5.16.

Code berikut adalah langkah pertama yang harus dibuat untuk membangun program PSO. Langkah pertama adalah menyiapkan sebuah class untuk Objective dan setelah selesai membuat class PSO.

Code 5.1 class Objective

```
1.  import math
2.  class Objective:
3.      result = None
4.      def __init__(self):
5.          pass
6.
7.      def Rosenbrock(self, x):
8.          x1 = x[0]
9.          x2 = x[1]
10.         temp1 = math.pow(math.pow(x1, 2) - x2, 2)
11.         temp2 = math.pow(x1-1, 2)
12.         return 100*temp1-temp2
```

Code 5.1 diberi nama class Objective, di mana dalam class tersebut terdapat method Resenbrock. Karena hanya bekerja pada satu fungsi, yang perlu buatkan code hanya fungsi ini. Namun, kalau nanti kita mau menguji dengan fungsi yang lain kita sudah membuatkan class Objective tinggal menambahkan fungsi yang lain.

Untuk mengujikan fungsi tersebut kita perlu mengeksekusi class dan method yang telah dibuat. Pada contoh ini penulis membuat file sendiri dengan nama main.py dengan isi code seperti berikut ini.

Code 5.2 main.py

```
1.  from objective_function import Objective
2.  if __name__ == '__main__':
3.      obj=Objective()
4.      print("hasil ",obj.Rosenbrock([-2, 2]))
```

Output Program 5.1

```
C:\Users\Kurniawan\anaconda3\python.exe  "C:/Users/Kurniawan
Google Drive/program Python/pythonProject/main.py"
hasil  391.0
```

Apabila kita RUN Hasil main.py, hasilnya terdapat pada Output Program 5.1, di mana hasil yang diperoleh sama dengan hasil perhitungan manual. Kita bisa mencoba-coba nilainya dalam line 4 code 5.2 untuk mendapatkan hasil yang lain. Karena hasilnya sudah sesuai, kita bisa pergi ke langkah selanjutnya, yaitu ke inti dari Metode PSO.

Algoritma PSO diawali dengan menentukan berapa banyak n burung (particle) dalam satu populasi (swarm). Setelah menentukan n particle, langkah berikutnya adalah melakukan generate nilai awal (random) dari masing-masing particle. Sebagai contoh, jika kita anggap n = 5, kita

akan mendapatkan posisi awal (x_1 dan x_2) dengan batasan range nilai $x \epsilon \{-5..10\}$ sebanyak n: (misalkan, yang degenerate di bawah ini).

p={[1;1],[1,4;2,1],[3;-1],[5;-2],[1;1]}

Code 5.3 class PSO dan Initial Method

```
1.  import random
2.  class PSO:
3.      nParticle = 0
4.      ndim = 0
5.      particles = []
6.
7.      def __init__(self, nBird,nDim):
8.          self.nParticle = nBird
9.          self.ndim = nDim
10.         self.initPosition()
11.
12.     def initPosition(self):
13.         min = -5
14.         max = 10
15.         n = self.nParticle
16.         particle = []
17.         for i in range(n):
18.             particle.append(
19.                 random.sample(range(min, max), self.ndim))
20.         self.particles = particle
```

Sebelum membuat method untuk melakukan generate nilai particle, langkah pertama adalah membuat file dan class terlebih dahulu. Nama file yang dipilih dalam pembahasan ini adalah pso. py dan nama class PSO. Dalam class ini terdapat constructor dengan satu argumen (sementara), yaitu nBird. Argumen ini digunakan untuk

inisialisasi berapa particle dalam kawanan. Pada class ini juga terdapat method initPosition (line 11), method ini digunakan untuk melakukan generate bilangan acak dengan range nilai yang telah ditentukan sebelumnya. Output method ini akan dikembalikan pada attribute class, yaitu particles. Selanjutnya, untuk mencoba hasil dari method ini yang harus dilakukan adalah memanggil method ini dalam constructor class (line 8).

Seperti code sebelumnya untuk mencoba, kita perlu memanggil class pada main.py seperti code berikut ini.

Code 5.4 testing class PSO

```
1.  from objective_function import Objective
2.  from pso import PSO
3.
4.  if __name__ == '__main__':
5.      # obj=Objective()
6.      # print("hasil ",obj.Rosenbrock([-2, 2]))
7.      obj = PSO(5, 2)
8.      print("particles :\n",obj.particles)
```

Output Program 5.2 generate particles

```
particles :
 [[-1, -3], [7, 8], [-4, -3], [1, -3], [7, 1]]
```

Setelah berhasil melakukan generate particle (swarm), hal yang harus dilakukan selanjutnya adalah menghitung nilai fitness (objective function) dari masing-masing particle. Code Python untuk menghitung nilai fitness dapat dilihat seperti code berikut.

Code 5.5 perhitungan Fitness

```
1.  import random
2.  from objective_function import Objective
3.
4.  class PSO:
5.      obj = Objective()
6.      fit = []
7.
8.      def __init__(self, nBird,nDim):
9.          self.nParticle = nBird
10.         self.ndim = nDim
11.         self.initPosition()
12.         self.calculateFitness()
13.
14.     def calculateFitness(self):
15.         n = self.nParticle
16.         fit = []
17.         for i in range(n):
18.             fit.append(self.obj.Rosenbrock(
19.                 self.particles[i]))
20.         self.fit = fit
```

Pembahasan ini menambahkan method calculateFitness (Code tersebut) dalam class pso.py. Hal ini karena dalam perhitungan fitness kita perlu memanggil method Rosenbrock pada class objective_ function.py sehingga perlu dilakukan pemanggilan class, seperti line 2. Hasil dari method calculateFitness akan ditampung ke attribute class sehingga perlu ditambahkan attribute fit = [ ], seperti pada line 6.

Code 5.6 testing fitness

```
1.  from pso import PSO
2.
3.  if __name__ == '__main__':
4.      obj = PSO(5, 2)
5.      print("particles position :\n", obj.particles)
6.      print("fitness :\n", obj.fit)
```

Untuk mendapatkan hasil yang sama dengan testing-testing fungsi sebelumnya, kita perlu RUN program di main.py (Code 5.6). Dalam file Code tersebut terdapat beberapa perubahan syntax dari sebelumnya. Code ini sudah tidak memiliki import untuk file objective_function. py karena sudah tidak digunakan. Ketika kita RUN file ini, hasil yang diperoleh dapat dilihat pada Output berikut.

Output Program 5.3 output fitness

```
particles position :
 [[-1, 0], [1, -3], [9, 0], [9, -5], [8, 6]]
fitness :
 [96.0, 1600.0, 656036.0, 739536.0, 336351.0]
```

Proses selanjutnya adalah menghitung Velocity. Untuk menghitung Velocity, langkah awal yang harus ditemukan adalah mencari nilai Gbest. Gbest, seperti yang dijelaskan sebelumnya adalah nilai paling optimal. Dalam studi kasus fungsi Rosenbrock, fungsi optimal adalah fungsi minimasi. Nilai terbaik yang akan dicari adalah nilai fitness-nya dan index particle terbaiknya.

Code 5.7 Method pencarian Gbest

```
1.  import random
2.  import numpy as np
3.  from objective_function import Objective
4.
5.  class PSO:
6.      nParticle = 0
7.      ndim = 2
8.      particles = []
9.      obj = Objective()
10.     fit = []
11.     GbestInd = None
12.     GbestVal = 0
13.
14.     def __init__(self, nBird, nDim, maxmin):
15.         self.nParticle = nBird
16.         self.ndim = nDim
17.         self.initPosition()
18.         self.calculateFitness()
19.         self.findGbest(maxmin)
20.
21.     def findGbest(self, maxmin):
22.         if maxmin == 'minimasi':
23.             fitArr = np.array(self.fit)
24.             self.GbestInd = np.argmin(fitArr)
25.             self.GbestVal = np.min(fitArr)
26.         else:
27.             fitArr = np.array(self.fit)
28.             self.GbestInd = np.argmax(fitArr)
29.             self.GbestVal = np.max(fitArr)
```

Code program untuk mendapatkan Gbest dapat dilihat di Code 5.7 dalam method findGbest. Dalam method tersebut terdapat beberapa

komponen yang harus ditambahkan dalam import library numpy (line 2). Pada attribute class PSO terdapat penambahan attribute GbestInd dan GbestVal yang nilainya diupdate pada method findGbest.

Code 5.8 testing method findGbest

```
1.  if __name__ == '__main__':
2.       obj = PSO(5, 2,'minimasi')
3.       print("particles position :\n", obj.particles)
4.       print("fitness :\n", obj.fit)
5.       print("Gbest Value :",obj.GbestVal)
6.       print("Gbest Index :", obj.GbestInd)
7.      print("Gbest Position :",obj.particles[obj.GbestInd])
```

Code di atas merupakan main program untuk testing method findGbest. Output yang dapat diperoleh dari method tersebut berupa 3 nilai, yaitu nilai Gbset Value (line 3), nilai Gbest index (line 4), dan nilai Gbest position (line 5). Hasil dari RUN program ini dapat dilihat dalam output program berikut.

```
particles position :
 [[-4, 2], [-3, 1], [2, 3], [5, 7], [2, -5]]
fitness :
 [19575.0, 6384.0, 99.0, 32384.0, 8099.0]
Gbest Value : 99.0
Gbest Index : 2
Gbest Position : [2, 3]
```

Setelah mendapatkan Gbest, proses selanjutnya adalah mencari Pbest. Karena proses ini adalah iterasi pertama (ke-0), nilai Pbest setiap particle akan sama dengan current position. Untuk itu harus membuat code untuk initial Pbest.

Code 5.9 init Pbest dan Pbestfit

```
1.  import random
2.  import numpy as np
3.  from objective_function import Objective
4.
5.  class PSO:
6.      nParticle = 0
7.      ndim = 0
8.      particles = []
9.      obj = Objective()
10.     fit = []
11.     GbestInd = None
12.     GbestVal = 0
13.     Pbest = []
14.     Pbestfit = []
15.
16.     def __init__(self, nBird, nDim, maxmin):
17.         self.nParticle = nBird
18.         self.ndim = nDim
19.         self.initPosition()
20.         self.calculateFitness()
21.         self.findGbest(maxmin)
22.         self.initPbest()
23.
24.     def initPbest(self):
25.         pbest=[]
26.         Pbestfit=[]
27.         for i in range(self.nParticle):
28.             pbest.append(self.particles[i])
29.             pbestfit.append(self.fit[i])
30.         self.Pbest=pbest
31.         self.Pbestfit=pbestfit
```

Untuk membuat initial Pbest dan Pbestfit, hal yang harus dilakukan adalah membuat attribute Pbest=[] dan Pbestfit=[] (line 13 dan 14). Setelah menyiapkan attribute, kemudian membuat method initPbest (line 24). Panggil method tersebut dalam constructor (line 22) untuk mengaktifkan/mengeksekusi method initPbest. Hasil dan main program dapat dilihat pada Code dan output program berikut.

Code 5.10 testing method initPbest

```
1.  if __name__ == '__main__':
2.      obj = PSO(5, 2, 'minimasi')
3.      print("particles position :\n", obj.particles)
4.      print("fitness :\n", obj.fit)
5.      print("Gbest Value :",obj.GbestVal)
6.      print("Gbest Index :", obj.GbestInd)
7.     print("Gbest Position :", obj.particles[obj.GbestInd])
8.      print("Pbest Position :", obj.Pbest)
```

Output Program 5.4 output initiPbest

```
particles position :
 [[-5, -3], [0, 2], [-1, -3], [3, 6], [3, -3]]
fitness :
 [78364.0, 399.0, 1596.0, 896.0, 14396.0]
Gbest Value : 399.0
Gbest Index : 1
Gbest Position : [0, 2]
Pbest Position : [[-5, -3], [0, 2], [-1, -3], [3, 6], [3, -3]]
```

Proses utama dari algoritma PSO adalah mencari kecepatan dan arah (velocity). Menghitung velocity dapat dilakukan dengan memperhatikan Code 5.11. Pada code tersebut terdapat beberapa penambahan (tambahkan code pada contoh, yang tidak terdapat pada code sebelumnya tidak perlu menghapus jika tidak ada instruksi pada

buku). Penambahan pertama berada pada attribute velocity =[] (line 6). Update berikutnya adalah method untuk inisialisasi velocity dengan pemberian nilai 0 untuk attribute velocity=[], yang telah dituliskan dalam method initVelo (line 11). Setelah membuat method, kita harus panggil dalam constructor (line 6).

Langkah terakhir adalah membuat method utama, yaitu calVelo (calculation velocity). Untuk memendek baris code-nya dipisahkan menjadi beberapa bagian. Velo1 adalah bagian untuk menghitung selisih particle dengan Pbest, velo2 untuk menghitung particle dengan Gbest, dan variabel result untuk menjumlahkan velocity awal (0) dengan velo1 dan velo2. Hasil dari method ini berupa return value (return result). Hal ini karena nantinya method ini akan dipanggil ke method yang lain.

Code 5.11 inisialisasi velocity dan menghitung velocity

```
1.  import random
2.  import numpy as np
3.  from objective_function import Objective
4.
5.  class PSO:
6.      velocity = []
7.
8.      def __init__(self, nBird, nDim, maxmin):
9.          self.initVelo()
10.
11.     def initVelo(self):
12.         self.velocity = [0]*self.nParticle
13.
14.     def calVelo(self, index):
15.         gbest = np.array(self.particles[self.Gbest])
16.         particle = np.array(self.particles[index])
17.         pbest = np.array(self.Pbest[index])
18.         velocit = np.array(self.velocity[index])
```

```
19.         velo1 = random.random()*(
20.             pbest-particle)
21.         velo2 = random.random()*(
22.             gbest-particle)
23.         result = velocit+velo1+velo2
24.         return result
```

Untuk mencoba hasil dari method yang telah dibuat, kita perlu menambahkan pemanggilan method pada main.py (seperti Code 5.12), yang hasilnya dapat dilihat pada Output Program dengan hasil velocity = [1.529 1.529].

Catatan: Jika hasil program yang Anda buat tidak sama, hal ini karena terdapat nilai random pada perhitungan tersebut.

Code 5.12 testing calculate velocity

```
1.  if __name__ == '__main__':
2.      obj = PSO(5, 2, 'minimasi')
3.      print("velocity :",obj.calVelo(4))
```

Output Program 5.5 output velocity pada particle dengan index ke-4

```
velocity : [-0.63901152  3.8340691 ]
```

Setelah berhasil mendapatkan nilai velocity untuk satu individu, langkah berikutnya adalah menghitung seluruh nilai velocity untuk semua individu. Tambahkan Code 5.13 pada class pso, tanpa menghapus method yang lainnya. Selanjutnya, tambahkan pemanggil method calVeloAll pada constructor class (line 2).

Code 5.13 method menghitung velocity semua particle

```
1.  def __init__(self, nBird, nDim, maxmin):
2.      self.calVeloAll()
3.
4.  def calVeloAll(self):
5.     velo = []
6.     for i in range(self.nParticle):
7.           velo.append(self.calVelo(i))
8.     self.velocity = velo
```

Code 5.14 testing calVeloAll

```
1.  from pso import PSO
2.  if __name__ == '__main__':
3.      obj = PSO(5, 2 ,'minimasi')
4.      print("velocity all:", obj.velocity)
```

Output Program 5.6 output velocity semua particle

```
velocity all: [array([-5.37279627, -5.37279627]), array([0.,
0.]), array([-5.70656193, -4.75546828]), array([-0.63328106,
0.21109369]), array([ 5.34002366, -3.56001577])]
```

Code 5.15 merupakan code yang digunakan untuk mencoba menampilkan attribute velocity yang di-update nilainya dengan method sebelumnya. Tampak dari hasil output program bahwa nilai yang tertera sudah berubah (bukan 0 lagi). Nilai yang diperoleh jika kita RUN akan berbeda, hal ini wajar karena inisialisasi dan perhitungan velocity terdapat nilai random.

Proses berikutnya dari rangkaian algoritma PSO adalah update position. Untuk perhitungan update posisi ini cukup mudah, yaitu hanya dengan menambahkan posisi lama dengan velocity setiap individu.

Code 5.15 update position

```
1.  import random
2.  import numpy as np
3.  from objective_function import Objective
4.
5.  class PSO:
6.
7.      def __init__(self, nBird, nDim, maxmin):
8.          self.updatePosition()
9.
10.     def updatePosition(self):
11.         oldParticle = self.particles
12.         newPos=[]
13.         for i in range(self.nParticle):
14.             pos = self.particles[i]+self.velocity[i]
15.             newPos.append(pos)
16.         self.particles = newPos
```

Code 5.15 digunakan untuk membuat update posisi baru dari particle. Pada code tersebut penulis hanya mencantumkan hal-hal yang perlu ditambahkan pada code sebelumnya. Misalkan, pada constructor terdapat pemanggilan method updatePosition dan pada line 10 terdapat pembuatan method updatePosition. Tambahkan Code 5.16 pada main.py untuk mendapatkan hasil dari attribute perubahan nilai posisi dari particle.

Code 5.16 testing nilai tempPos

```
from pso import PSO
if __name__ == '__main__':
    obj = PSO(5, 2 ,'minimasi')
    print("particle position :", obj.particles)
```

Output Program 5.7 particle lama dan particle baru

```
particle position : [array([1.75493541, 1.32017225]),
array([4.01299691, 1.98700309]), array([0., 6.]),
array([-1.93370694, 4.45303445]), array([ 2.71020818,
-0.32381909])]
```

Semua code yang telah dituliskan merupakan code yang digunakan untuk inisialisasi saja, baik inisialisasi untuk atribut atau untuk perhitungan awal. PSO membutuhkan perulangan untuk menemukan solusi terbaik. Sebelum masuk dalam pembuatan perulangan (main loop) kita perlu memperhatikan seluruh Code X (5.17). Hal ini akan membantu kita melihat apakah code yang telah dibuat secara parsial (terpisah) sudah sesuai atau belum.

Code 5.17 file pso.py tanpa perulangan

```
1.  import random
2.  import numpy as np
3.  from objective_function import Objective
4.  import matplotlib.pyplot as plt
5.
6.  class PSO:
7.      nParticle = 0
8.      ndim = 0
9.      particles = []
10.     obj = Objective()
11.     fit = []
12.     Gbest = None
13.     Pbest = []
14.     Pbestfit = []
15.     velocity = []
16.
17.     def __init__(self, nBird, nDim, maxmin):
18.         self.nParticle = nBird
```

```
19.         self.ndim = nDim
20.         self.initPosition()
21.         self.calculateFitness()
22.         self.initPbest()
23.         self.findGbest(maxmin)
24.         self.initVelo()
25.         self.calVeloAll()
26.         self.updatePosition()
27.
28.     def initPosition(self):
29.         min = -5
30.         max = 10
31.         n = self.nParticle
32.         particle = []
33.         for i in range(n):
34.             particle.append(
35.                random.sample(range(min, max), self.ndim))
36.         self.particles = particle
37.
38.     def calculateFitness(self):
39.         n = self.nParticle
40.         fit = []
41.         for i in range(n):
42.             fit.append(self.obj.Rosenbrock(
43.                 self.particles[i]))
44.         self.fit = fit
45.
46.     def findGbest(self, maxmin):
47.         if maxmin == 'minimasi':
48.             fitArr = np.array(self.Pbestfit)
49.             self.Gbest = np.argmin(fitArr)
50.         else:
51.             fitArr = np.array(self.fit)
```

```
52.             self.Gbest = np.argmax(fitArr)
53.
54.     def initPbest(self):
55.         pbest = []
56.         pbestfit = []
57.         for i in range(self.nParticle):
58.             pbest.append(self.particles[ i])
59.             pbestfit.append(self.fit[i])
60.         self.Pbest = pbest
61.         self.Pbestfit = pbestfit
62.
63.     def initVelo(self):
64.         self.velocity = [0]*self.nParticle
65.
66.     def calVelo(self, index):
67.         gbest = np.array(self.Pbest[self.Gbest])
68.         particle = np.array(self.particles[index])
69.         pbest = np.array(self.Pbest[index])
70.         velocit = np.array(self.velocity[index])
71.         velo1 = random.random()*(
72.             pbest-particle)
73.         velo2 = random.random()*(
74.             gbest-particle)
75.         result = 0.7*velocit+velo1+velo2
76.         return result
77.
78.     def calVeloAll(self):
79.         velo = []
80.         for i in range(self.nParticle):
81.             velo.append(self.calVelo(i))
82.         self.velocity = velo
83.
84.     def updatePosition(self):
```

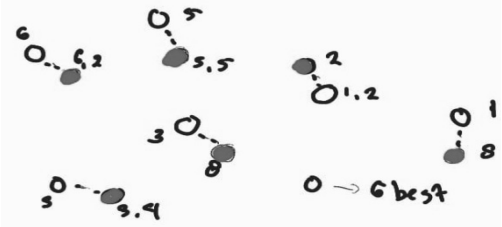
```
85.         newPos = []
86.         for i in range(self.nParticle):
87.             pos = self.particles[i]+self.velocity[i]
88.             newPos.append(pos)
89.         self.particles = newPos
```

Seluruh proses perhitungan (parsial/terpisah) pada algoritma PSO telah selesai dijelaskan dan dengan code program. Langkah yang dikerjakan selanjutnya adalah menyusun langkah-langkah dari algoritma PSO.

1. Menghitung Fitness.
2. Update Pbest.
3. Find Gbest.
4. Update velocity.
5. Update position particle.

Dalam pembahasan ini, code program langkah kedua (update Pbest) belum dibuat. Code sebelumnya yang sudah dituliskan adalah code untuk inisialisasi awal Pbest. Oleh sebab itu, sebelum kita membuat looping untuk kelima proses tersebut, kita siapkan code untuk update Pbest terlebih dahulu.

Gambar 5.22 merupakan ilustrasi untuk meng-update Pbest (kasus: maximasi). Gambar 5.22 versi warna dapat dilihat pada laman https://bit.ly/346NSI5. Setiap akan terjadi perpindahan posisi node/particle (kecuali node sebagai Gbest), nilai fitness-nya akan dicek ulang. Seperti pada node paling atas (node dengan nilai 6). Setelah update posisi, nilainya berubah menjadi 6.2. Nilai Pbest (warna merah) node tersebut berada pada posisi yang baru (current node == Pbest). Berbeda dengan node dengan nilai 2 (nilai awal), ketika berubah posisi nilainya lebih kecil sehingga Pbest akan berapa pada nilai posisi lama.



Gambar 5.22 Update Ilustrasi Pbest dan Current Position

Code untuk menghitung update Pbest dapat dilihat pada Code 5.18. Tambahkan code ini pada class pso (file pso.py) pada kumpulan method-method yang telah dibuat sebelumnya. Hasil pada code tersebut akan meng-update nilai Pbest yang lama sehingga nilai Pbest lama akan ter-update setelah method ini dijalankan.

Code 5.18 update Pbest dan PbestFit

```
1.  def updatePbest(self, maxmin):
2.      pbest=[]
3.      pbestfit=[]
4.      for i in range(self.nParticle):
5.            if maxmin == 'minimasi':
6.                  if self.Pbestfit[i] <= self.fit[i]:
7.                        pbest.append(np.array(
8.                              self.Pbest[i]))
9.                        pbestfit.append(self.Pbestfit[i])
10.                 else:
11.                       pbest.append(self.particles[i])
12.                       pbestfit.append(self.fit[i])
13.     self.Pbest=pbest
14.     self.Pbestfit=pbestfit
```

Kembali pada proses perulangan algoritma PSO, dengan dibuatnya method untuk update Pbest, lengkap sudah method yang dibutuhkan.

Langkah selanjutnya adalah membuat method untuk main loop. Method tersebut dapat dilihat pada Code 5.19.

Code 5.19 mainloop dan plot

```
1.  def mainloop(self, maxmin,maxloop):
2.      gvalue = []
3.      for i in range(maxloop):
4.              self.calculateFitness()
5.              self.updatePbest(maxmin)
6.              self.findGbest(maxmin)
7.              self.calVeloAll()
8.              self.updatePosition()
9.              gvalue.append(self.Pbestfit[self.Gbest])
10.     self.plot(gvalue)
11.
12. def plot(self, val):
13.     plt.plot(val)
14.     plt.show()
```

Pada Code 5.19 terdapat dua method, yaitu mailoop dan plot. Method plot digunakan untuk menampilkan grafik hasil perubahan nilai Gbest fitness. Pada method mainloop terdapat pemanggilan method updatePbest (line 5) yang telah dibuat sebelumnya. Hal ini karena method mainloop membutuhkan argumen tambahan, yaitu maxloop, yang pada constructor class pso juga harus ditambahkan. Hal ini memudahkan kita nantinya dalam uji coba program. Constructor class pso sekarang menjadi seperti berikut.

Code 5.20 constructor pso setelah penambahan maxloop

```
1.  def __init__(self, nBird, nDim, maxmin,maxloop):
2.      self.nParticle = nBird
3.      self.ndim = nDim
4.      self.initPosition()
```

```
5.      self.calculateFitness()
6.      self.initPbest()
7.      self.findGbest(maxmin)
8.      self.initVelo()
9.      self.calVeloAll()
10.     self.updatePosition()
11.     self.mainloop(maxmin, maxloop)
```

Karena terjadi perubahan di bagian constructor, pada file main.py juga akan terdapat perbuahan pemanggilan class dalam objek. Jika kita perhatikan code di bawah, terdapat variabel yang bisa diubah (kecepatan dimensi, dikarenakan fungsi objective Rosenbrock membutuhkan 2 dimensi) untuk mendapatkan hasil yang berbeda.

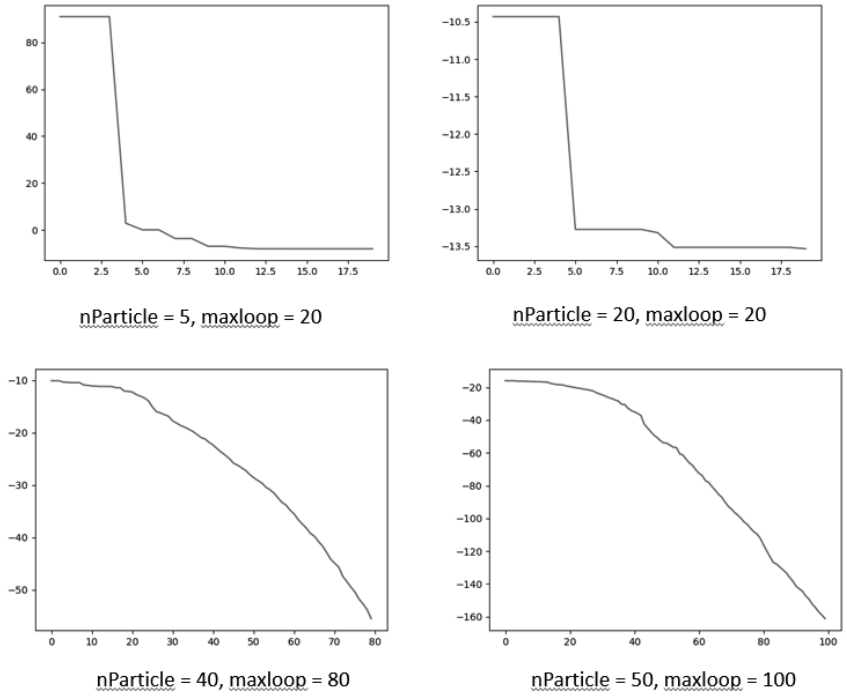
Code 5.21 main.py setelah penambahan

```
1.  from pso import PSO
2.  if __name__ == '__main__':
3.      nParticle = 5
4.      dimensi = 2
5.      maxloop = 10
6.      fungsi = 'minimasi'
7.      obj = PSO(nParticle, dimensi,fungsi, maxloop)
8.      print("nilai fitness terbaik :", obj.Pbestfit[obj.
        Gbest])
9.      print("posisi Terbaik :", obj.Pbest[obj.Gbest])
```

Run program akan menghasilkan 2 hal. Pertama berupa angka, seperti yang kita lihat sebelumnya dengan hasil berupa plotting. Untuk hasil angka kita bisa lihat seperti Output Program di bawah. Hasil yang tertera merupakan hasil percobaan dengan nilai nParticle = 50 dan maxloop = 100. Sementara itu, untuk plotting, sudah dicoba dengan 4 kombinasi yang berbeda dan dapat dilihat pada Gambar 5.23.

Output Program 5.8 fitness terbaik dan posisi terbaik

```
nilai fitness terbaik : -161.16803069143344
posisi Terbaik : [-11.74380208 137.80569111]
```



Gambar 5.23 Hasil Plotting 4 Percobaan

Clustering

Clusterting dalam pemodelan ini adalah pengelompokan data berdasarkan kesamaan attribute. Dalam soft-clustering, model yang digunakan bertujuan membuat pusat cluster (centroid) tepat berada di nilai pusat cluster secara bertahap. Pemodelan dengan optimasi bisa diartikan sebagai perubahan posisi centroid melakukan update nilai berdasarkan algoritma optimasi yang digunakan. Fungsi objektif yang digunakan dilakukan dengan menghitung total jarak centroid

pada masing-masing cluster data (SSE) atau jika pengertian tersebut dibalik, menghitung nilai jarak data dengan pusat centroid pada cluster tersebut.

$$SSE = \sum_{i=1}^{C}\sum_{j=1}^{n}\sqrt[2]{\left(centroid_i - data_j\right)^2}$$

di mana:

C	=	merupakan jumlah Centroid yang terbentuk
n	=	merupakan jumlah data yang terdapat pada cluster ke – i
centroid	=	merupakan titik/posisi Centroid
data	=	merupakan titik/posisi Data

Dari definisi tersebut, langkah pertama untuk pemodelan optimization clustering adalah membuat fungsi objektif dengan fungsi minimasi total jarak (SSE). Untuk menghitung SSE kita harus bisa menghitung terlebih dahulu Distance (jarak) satu data dengan titik Centroid.

$$distance = \sum_{i=1}^{dim}\sqrt{\left(centroid_i - data_i\right)}$$

dim merupakan dimensi dari data.

Sebagai contoh perhitungan manual, misalkan, titik centroid dan data berada pada titik [1 2] serta [3 1], dimensi yang terbentuk (dim) adalah 2. Cara menghitungnya bisa dilihat seperti berikut ini.

$$distance = \sqrt{\left(1-3\right)^2 + \left(2-1\right)^2}$$

$$distance = \sqrt{4+1} = \sqrt{5} = 2{,}23606797749979$$

Dalam pembuatan program, kita asumsikan titik centroid telah didapatkan dari class yang lain. Data yang akan diolah adalah data yang diperoleh dari file Excel dengan data sebagai berikut.

	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 5.24 Data Contoh Excel

Buat data seperti Gambar 5.24 dan beri nama dengan "dataset.xlsx". Supaya tidak terjadi error, buat data yang sama persis beserta dengan nama kolom yang sama. Simpan file Excel tersebut pada folder yang sama dengan program yang telah dibuat.

Setelah membuat data, pekerjaan berikutnya dalam membuat file .py baru dengan nama Dataset.py. Dalam file tersebut buat class dengan nama data, seperti Code 5.22. Terdapat dua library pada class data: numpy dan pandas. Selain itu, terdapat satu attribute dan satu method untuk membaca data dari Excel yang akan dipanggil pada constructor class.

Code 5.22 pembacaan data dari Excel

```
1.  import pandas as pd
2.  import numpy as np
3.
4.  class data:
5.      dset = []
6.      def __init__(self):
```

```
7.          self.readDataExcel()
8.
9.      def readDataExcel(self):
10.         file = pd.read_excel(open('dataset.xlsx', 'rb'))
11.         dframe = pd.DataFrame(file, columns=(['x', 'y']))
12.         self.dset = np.array(dframe)
```

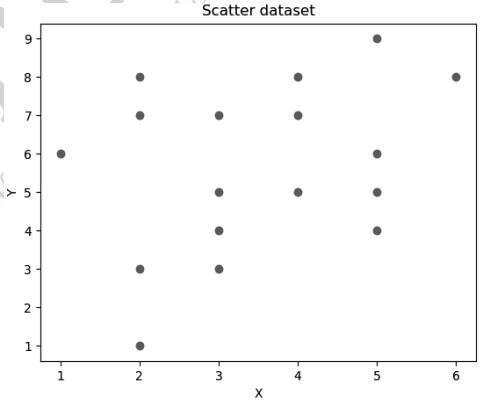
Code 5.23 testing read data from Excel

```
1.  from pso import PSO
2.  from Dataset import data
3.  if __name__ == '__main__':
4.      pop = data()
5.      print("dataset: ",pop.dset)
6.  Output Program B.9 output read data from Excel

dataset:  [[4 5]
 [1 6]
 [2 7]
 [6 8]
 [4 8]
 [5 4]
 [2 3]
 [3 7]
 [3 3]
 [5 9]
 [3 4]
 [3 5]
 [2 8]
 [4 7]
 [2 1]
 [5 6]
 [5 5]]
```

Untuk mempercantik tampilan dari data yang telah berhasil diambil dari Excel, kita buat scatter plotting terlebih dahulu. Pembuatan code untuk plotting dapat dilihat pada Code 5.24. Jika RUN (tidak ada perbedaan pada main.py), hasil yang dapat dilihat pada plotting editor seperti pada Gambar 5.25.

Code 5.24 scatter plotting dataset

```
1.  class data:
2.      dset = []
3.      def __init__(self):
4.          self.readDataExcel()
5.          self.plotScatter()
6.
7.      def plotScatter(self):
8.          fig, ax = plt.subplots()
9.          tmp = self.dset
10.         ax.scatter(tmp[:, 0], tmp[:, 1], marker='o')
11.         plt.title("Scatter dataset")
12.         plt.xlabel('X')
13.         plt.ylabel('Y')
14.         plt.show()
```



Gambar 5.25 Scatter Plotting

Cara paling sederhana untuk menemukan total jarak (SSE) adalah dengan menghitung distance terlebih dahulu, seperti persamaan sebelumnya. Code 5.25 digunakan untuk menghitung Distance (jarak) Euclidean dari satu point ke point yang lain atau jika dimasukkan dengan konteks ini adalah centroid dengan satu data.

Code 5.25 menghitung distance

```
1.  def calDistance(self, centroid, data):
2.      n = centroid.size
3.      dis = 0
4.      for i in range(n):
5.            dis = dis+math.pow(centroid[i]-data[i], 2)
6.      return math.sqrt(dis)
```

Output dari Code 5.25 adalah return value dan tidak ada update nilai pada attribute karena nantinya code tersebut akan digunakan oleh method yang lain. Namun, kita harus menguji setiap method yang telah dibuat. Oleh sebab itu, kita perlu memanggil method ini pada main.py, seperti pada Code 5.26 dan sekalian hasilnya ditunjukkan pada Output Program 5.10.

Code 5.26 testing method calDistrance

```
1.  from pso import PSO
2.  from Dataset import data
3.  import numpy as np
4.  if __name__ == '__main__':
5.      pop = data()
6.      centroid = np.array([5, 5])
7.      data = pop.dset[0]
8.      print("centroid :", centroid)
9.      print("data :", data)
10.     print("tes  distance  :",  pop.calDistance(centroid,
        data))
```

Output Program 5.10 hasil testing method calDistance

```
centroid : [5 5]
data : [4 5]
tes distance : 1.0
```

Proses selanjutnya adalah menghitung jarak satu data dengan semua centroid yang ditentukan atau nilainya diberikan oleh metode Optimasi. Code untuk membuat programnya dapat dilihat dalam Code berikut.

Code 5.27 perhitungan jarak satu data dengan semua centroid

```
1.  def calData2Centroids(self, centroids, data):
2.      n, m = centroids.shape
3.      minimal = self.calDistance(centroids[0], data)
4.      indMin=0
5.      for i in range(n):
6.            minimal1 = self.calDistance(centroids[i], data)
7.            if minimal1 < minimal:
8.                  minimal = minimal1
9.                  indMin = i
10.     return minimal, indMin
```

Code 5.27 memiliki dua return value, yaitu nilai minimum distance dan index centroid dengan minimum distance. Pada method ini terdapat panggilan method yang telah dibuat sebelumnya (calDistance), kemudian dilakukan perulangan sebanyak jumlah centroid yang ada (line 5). Uji coba method tersebut dapat dilihat pada Code main.py dan hasilnya ada di Output Program.

Code 5.28 uji coba (main.py) method calData2Centroids

```
1.  from Dataset import data
2.  import numpy as np
3.  if __name__ == '__main__':
4.      pop = data()
5.      centroids = np.array([[5, 5], [3,1]])
6.      data = pop.dset[0]
7.      datas = pop.dset[1:5]
8.      print("centroids :\n", centroids)
9.      print("data :", data)
10.     dist, ind = pop.calData2Centroids(centroids, data)
11.     print("distance minimal to centroid:", dist)
12.     print("index centroid minimum distance: ", ind)
```

Output Program 5.11

```
centroids :
 [[5 5]
 [3 1]]
data : [4 5]
distance minimal to centroid: 1.0
index centroid minimum distance:  0
```

Rangkaian terakhir dari perhitungan jarak adalah dengan menghitung jarak semua data yang ada ke semua titik centroid. Perhitungan ini akan menghasilkan dua nilai, yaitu nilai index data dan jarak minimal. Nilai index data diartikan sebagai label atau cluster data tersebut masuk ke dalam index atau cluster berapa. Untuk jarak minimal adalah jarak paling kecil dari satu data ke semua centroid.

Code 5.29 perhitungan jarak semua data dengan semua centroid

```
1.  import pandas as pd
2.  import numpy as np
3.  import matplotlib.pyplot as plt
4.  import math
5.
6.  class data:
7.      dset = []
8.      distMin = []
9.      index = []
10.     def __init__(self, centroids):
11.         self.readDataExcel()
12.         self.plotScatter()
13.         self.calDatas2centroids(centroids)
14.
15.     def calDatas2centroids(self,centroids):
16.         n, m = self.dset.shape
17.         distMin = []
18.         index = []
19.         for i in range(n):
20.             dist, ind = self.calData2Centroids(
21.                 centroids, self.dset[i])
22.             distMin.append(dist)
23.             index.append(ind)
24.         self.distMin = distMin
25.         self.index = index
```

Code 5.30 testing method calDatas2centroids

```
1.  from Dataset import data
2.  import numpy as np
3.  if __name__ == '__main__':
4.      centroids = np.array([[5, 5], [3, 1], [6, 6]])
```

```
5.      pop = data(centroids)
6.      data = pop.dset[0]
7.      datas = pop.dset[1:5]
8.      print("centroids :\n", centroids)
9.      print("distance minimal :\n", pop.distMin)
10.     print("class data :\n", pop.index)
```

Output Program 5.12 output index dan distance minimal

```
centroids :
 [[5 5]
 [3 1]
 [6 6]]
distance minimal :
[1.0,  4.123105625617661,  3.605551275463989,  2.0,
2.8284271247461903, 1.0, 2.23606797749979, 2.8284271247461903,
2.0, 3.1622776601683795, 2.23606797749979, 2.0, 4.242640687119285,
2.23606797749979, 1.0, 1.0, 0.0]
class data :
 [0, 0, 0, 2, 2, 0, 1, 0, 1, 2, 0, 0, 0, 0, 1, 0, 0]
```

Proses terakhir dari class ini adalah perhitungan nilai SSE dan plotting scatter cluster, beserta dengan titik centroid-nya. Code 5.31 merupakan keseluruhan code yang terbentuk dalam class data. Dalam Code ini terdapat penambahan method untuk perhitungan SSE (calSSE) dan method untuk plotting scatter dataset (scatterDataCentroid dan getCluster).

Code 5.31 perhitungan SSE dan plotting

```
1.  import pandas as pd
2.  import numpy as np
3.  import matplotlib.pyplot as plt
4.  import math
5.
```

```
6.  class data:
7.      dset = []
8.      distMin = []
9.      index = []
10.     def __init__(self, centroids):
11.         self.readDataExcel()
12.         self.calDatas2centroids(centroids)
13.
14.     def readDataExcel(self):
15.         file = pd.read_excel(open('dataset.xlsx', 'rb'))
16.         dframe = pd.DataFrame(file, columns=(['x', 'y']))
17.         self.dset = np.array(dframe)
18.
19.     def plotScatter(self):
20.         fig, ax = plt.subplots()
21.         tmp = self.dset
22.         ax.scatter(tmp[:, 0], tmp[:, 1], marker='o')
23.         plt.title("Scatter dataset")
24.         plt.xlabel('X')
25.         plt.ylabel('Y')
26.         plt.show()
27.
28.     def calDistance(self, centroid, data):
29.         n = centroid.size
30.         dis = 0
31.         for i in range(n):
32.             dis = dis+math.pow(centroid[i]-data[i], 2)
33.         return math.sqrt(dis)
34.
35.     def calData2Centroids(self, centroids, data):
36.         n, m = centroids.shape
37.         minimal = self.calDistance(centroids[0], data)
38.         indMin=0
39.         for i in range(n):
```

```
40.           minimal1 = self.calDistance(centroids[i], data)
41.             if minimal1 < minimal:
42.                 minimal = minimal1
43.                 indMin = i
44.         return minimal, indMin
45.
46.     def calDatas2centroids(self,centroids):
47.         n, m = self.dset.shape
48.         distMin = []
49.         index = []
50.         for i in range(n):
51.             dist, ind = self.calData2Centroids(
52.                 centroids, self.dset[i])
53.             distMin.append(dist)
54.             index.append(ind)
55.         self.distMin = distMin
56.         self.index = index
57.
58.     def calSSE(self):
59.         return sum(self.distMin)
60.
61.     def getCluster(self, centroids):
62.         n, m = self.dset.shape
63.         j, k = centroids.shape
64.         temp1 = []
65.         for i in range(k+1):
66.             temp2 = []
67.             for j in range(n):
68.                 if i == self.index[j]:
69.                     temp2.append(self.dset[j])
70.             temp1.append(temp2)
71.         return temp1
72.
73.     def scatterDataCentroid(self, centroids):
```

```
74.         fig, ax = plt.subplots()
75.         x = self.getCluster(centroids)
76.         tes = []
77.         for i in range(len(x)):
78.             tes.append(i)
79.             temp = np.array(x[i])
80.             ax.scatter(temp[:, 0], temp[:, 1])
81.         cent = np.array(centroids)
82.         plt.legend(tes)
83.         ax.scatter(cent[:, 0], cent[:, 1], marker='x')
84.         plt.title('Scatter')
85.         plt.xlabel('X')
86.         plt.ylabel('Y')
87.         plt.show()
```

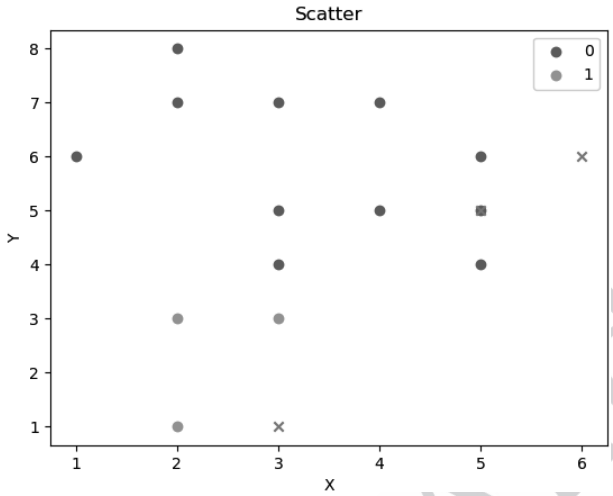
Untuk uji coba hasil Code dari class data, kita perlu menuliskan code di main.py, seperti pada Code 5.32. Hasil yang didapatkan terdapat dua hasil, hasil pada Output Program 5.13 dan Gambar 5.26. Dua hasil tersebut didapatkan dari hasil centroid yang telah ditetapkan, seperti pada line 4 pada file main.py. Nantinya dari hasil-hasil ini akan diperoleh hasil cluster terbaik berdasarkan algoritma PSO.

Code 5.32 testing

```
1.  from Dataset import data
2.  import numpy as np
3.  if __name__ == '__main__':
4.      centroids = np.array([[5, 5], [3, 1], [6, 6]])
5.      pop = data(centroids)
6.      print("SSE :", pop.calSSE())
7.      pop.scatterDataCentroid(centroids)
```

Output Program 5.13

```
SSE: 37.49863343036107
```



Gambar 5.26 Hasil Scatter Cluster dan Centroid

PSO – Clustering

Setelah mendapatkan nilai SSE dari class data artinya tugas class tersebut sudah selesai. Langkah berikutnya adalah mengubah integrasi class data dan PSO yang awalnya class PSO diintegrasikan dengan class objective.

Terdapat beberapa perubahan dan penambahan attribute pada class PSO untuk menyesuaikan hasil objective function dari clustering.

Penambahan pada class PSO, yakni sebagai berikut.

1. Pertama adalah penambahan attribute class ncluster = [] (seperti pada line 14).
2. Penambahan argument constructor dengan menambahkan satu argument nCluster.
3. Penambahan syntax dalam contructor yang bertugas untuk mengisi data ncluster dengan argument nCluster dengan syntax self.ncluster = nCluster (line 17).

4. Penambahan method plotResult. Method ini berfungsi untuk menambahkan plotting data yang diubah menjadi cluster beserta centroid-nya. Hasil yang terdapat di sini adalah hasil akhir dari proses iterasi PSO.

Perubahan yang terjadi terdapat pada beberapa hal, seperti berikut ini.

1. Perubahan utama terjadi pada method initPosition. Yang awalnya output dari method ini adalah nilai dengan 2 dimensi, sekarang output nilai ini menjadi 3 dimensi. Dimensi pertama untuk banykanya nParticle, dimensi kedua untuk jumlah ncluster, dan dimensi terakhir untuk banyak attribute data setiap cluster. Perubahan detailnya dapat dilihat pada line 34 sampai habis.
2. Perubahan penting juga terdapat pada method calculateFitness (line 47). Perubahan ini merupakan integrasi dari class data ke class PSO (line 51). Pada method awalnya perhitungan fitness dari class objective, pada perubahan ini perhitungan fitness diperoleh dari nilai SSE (line 52).

Code 5.33 update PSO class

```
1.  import random
2.  import numpy as np
3.  from Dataset import data
4.  import matplotlib.pyplot as plt
5.
6.  class PSO:
7.      ncluster = []
8.
9.      def __init__(self, nBird, nDim, nCluster, maxmin,
        maxloop):
10.         self.ncluster = nCluster
11.         self.nParticle = nBird
12.         self.ndim = nDim
```

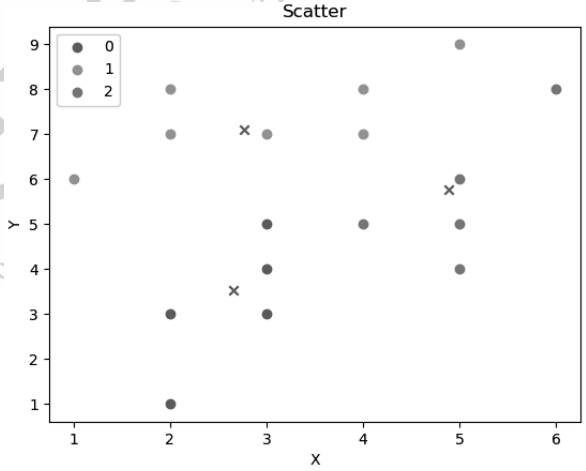
```
13.         self.initPosition()
14.         self.calculateFitness()
15.         self.initPbest()
16.         self.findGbest(maxmin)
17.         self.initVelo()
18.         self.calVeloAll()
19.         self.updatePosition()
20.         self.mainloop(maxmin, maxloop)
21.
22.     def plotResult(self):
23.         centroid = self.Pbest[self.Gbest]
24.         dt = data(centroid)
25.         dt.scatterDataCentroid(centroid)
26.
27.     def initPosition(self):
28.         dt = data()
29.         max, min = dt.dset.shape
30.         n = self.nParticle
31.         particles = []
32.         for j in range(n):
33.             particle = []
34.             for i in range(self.ncluster):
35.                 ind = random.randint(0, max-1)
36.                 particle.append(dt.dset[ind])
37.             particles.append(particle)
38.         self.particles = np.array(particles)
39.
40.     def calculateFitness(self):
41.         n = self.nParticle
42.         fit = []
43.         for i in range(n):
44.             obj = data(self.particles[i])
45.             fit.append(obj.calSSE())
46.         self.fit = fit
```

Code 5.34 testing PSO-Clustering.

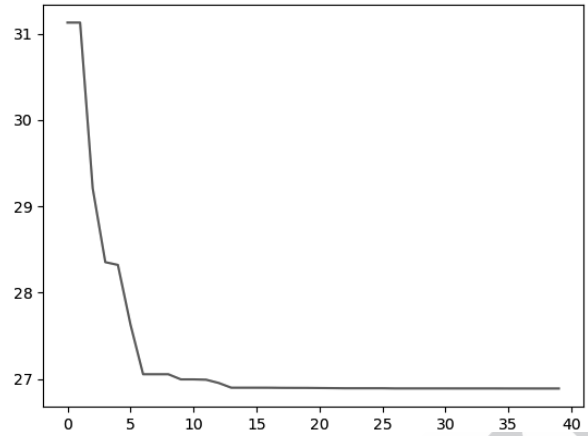
```
1.  from pso import PSO
2.  if __name__ == '__main__':
3.      nParticle = 5
4.      dimensi = 2
5.      ncluster = 3
6.      maxloop = 4
7.      fungsi = 'minimasi'
8.      obj = PSO(nParticle, dimensi, ncluster, fungsi,
        maxloop)
9.      print("nilai fitness terbaik :", obj.Pbestfit[obj.
        Gbest])
10.     print("posisi Terbaik :", obj.Pbest[obj.Gbest])
11.     obj.plotResult()
```

Output Program 5.14

```
nilai fitness terbaik : 22.635807340960408
posisi Terbaik : [[2.65627957 3.5417061 ]
 [2.77085305 7.11457348]
 [4.88542652 5.77085305]]
```



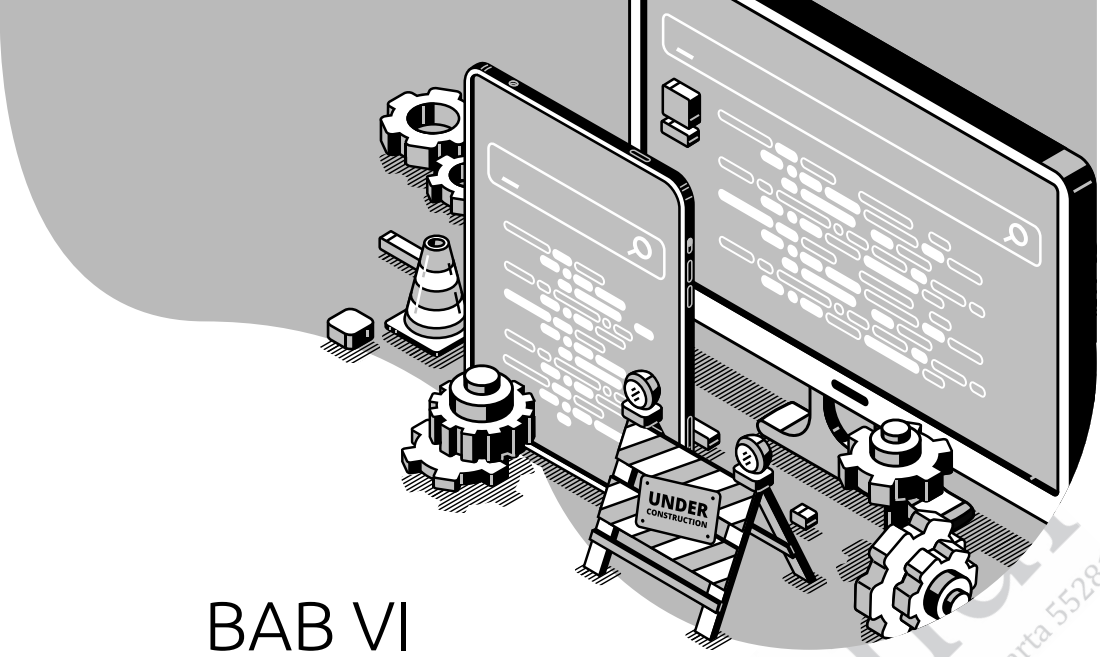
Gambar 5.27 Hasil Scatter Dataset dengan Centroid dengan 3 Cluster



Gambar 5.28 Perubahan Nilai SSE dengan 40 Iterasi

5.3 LATIHAN SOAL CLUSTERING BERDASARKAN MODEL

1. Sebutkan dan jelaskan apa ciri utama atau pembeda teknik clustering berbasiskan model dengan teknik clustering yang lain!
2. Sebutkan metode-metode yang termasuk dalam teknik clustering berbasis model!
3. Berapa layer jaringan saraf yang digunakan pada metode Self Organized Maps?
4. Pada optimasi clustering, nilai apa yang menjadi fungsi fitness/ fungsi objective?
5. Metode Particle Swarm Optimiziation dapat digantikan dengan metode apa saja untuk mendapatkan clustering?



BAB VI
PENERAPAN CLUSTERING

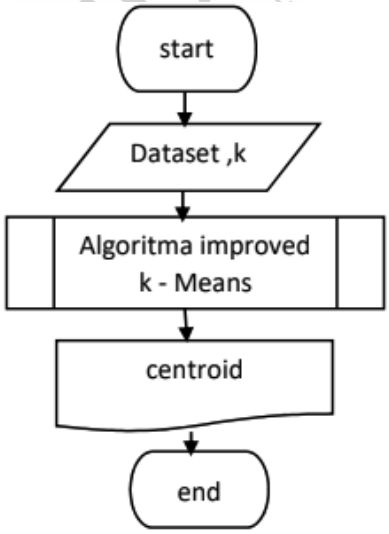
6.1 CLUSTERING PADA ACTIVE FIRE NASA DATASET

Dalam pembahasan ini, contoh studi kasus penerapan analisis clustering diambil dari penelitian dengan topik *Penerapan Modified K-Means untuk Dataset Nasa Active Fire Dataset*. Dataset diperoleh dari halaman website: *https://firms.modaps.eosdis.nasa.gov/active_fire/*. Tujuan utama penelitian adalah mengelompokkan titik panas (hotspot). Manfaat mengelompokkan dataset tersebut untuk simulasi tempat yang paling optimal, di mana tempat paling optimal dapat diartikan berada di pusat data.

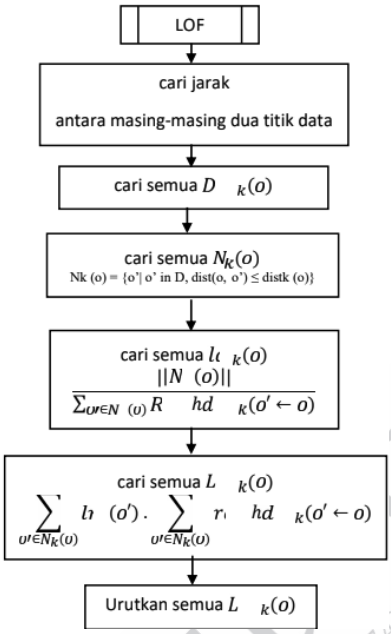
Berdasarkan pemaparan tersebut, penelitian ini merancang sebuah sistem pengelompokan data titik panas bumi dengan menggunakan metode clustering dengan Algoritma Improved K-Means. Algoritma

Improved K-Means merupakan algoritma K-Means dengan penambahan teknik LOF untuk menghilangkan outlier. Pengelompokan data hotspot diharapkan dapat membantu untuk mengetahui lokasi pemadaman paling efisien.

Penelitian ini akan mengembangkan sistem yang dapat mengelompokkan titik panas bumi berdasarkan dataset tabel deteksi titik panas bumi yang didapat dari web resmi NASA. Dataset tersebut merupakan data titik panas di wilayah Asia Tenggara selama 7 hari melalui satelit resmi milik NASA. Dari data tersebut terdapat 11 fitur, yaitu latitude, longitude, brigt_ti4, scan, track, aq_date, acq_time, confidence, bright_ti5, frp, dan daynight. Yang digunakan untuk pengelompokan ini adalah atribut longitude, latitude, dan Bright_ ti4. Pemilihan atribut latitude dan longitude digunakan karena fitur tersebut merupakan koordinat dari titik panas (hotspot) sehingga atribut tersebut dapat di gunakan untuk menyatakan letak dari titik panas (hotspot).

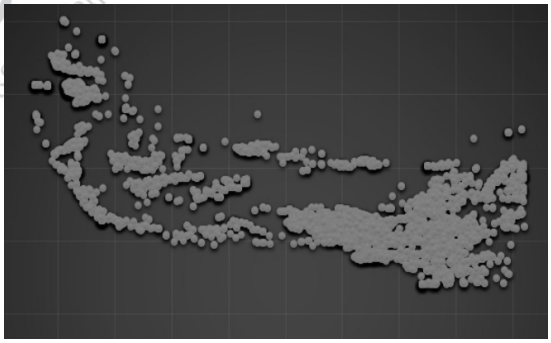


Gambar 6.1 Metodologi Penelitian Modified K-Means

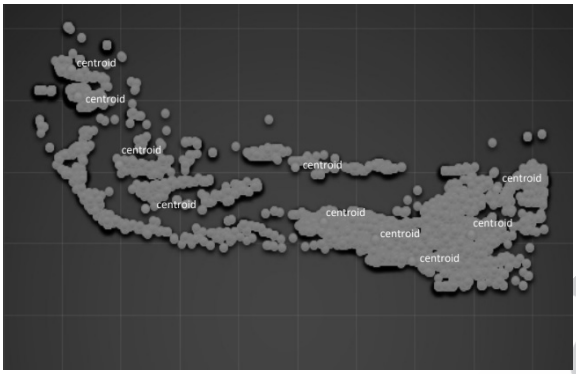


Gambar 6.2 Algoritma LOF

Gambaran dataset (scatter plotting) dari penelitian ini dapat dilihat pada Gambar 6.3. Setelah dilakukan proses clustering dengan modified K-Means, hasil centroid data dapat dilihat pada Gambar 6.4. Gambar 6.3 dan 6.4 versi warna dapat dilihat pada laman https://bit.ly/346NSI5. Titik yang berwarna orange merupakan centroid data. Terdapat 10 titik data (cenroid), dikarenakan jumlah cluster yang ditentukan ada 10 cluster.

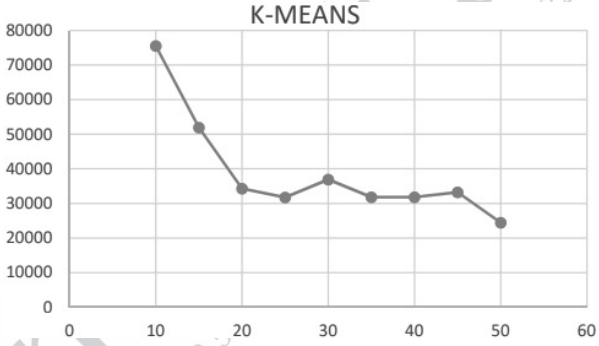


Gambar 6.3 Gambaran Dataset



Gambar 6.4 Centroid Data dengan 10 Cluster

Dari hasil pengukuran cluster yang paling optimal dengan teknik Elbow, hasil terbaik yang diperoleh adalah K=10 (10 cluster). Analisis ini dapat dilihat pada hasil yang ditunjukkan pada Gambar 6.5.



Gambar 6.5 Hasil Elbow

6.2 CLUSTERING PADA TEXT MINING

Dalam contoh studi kasus ini, analisis clustering digunakan untuk mengelompokkan dokumen. Dokumen yang dikelompokkan adalah dokumen abstrak dari skripsi. Contoh sebelumnya menggunakan data numerik untuk clustering, sedangkan dalam contoh ini data yang digunakan adalah data text (text mining).

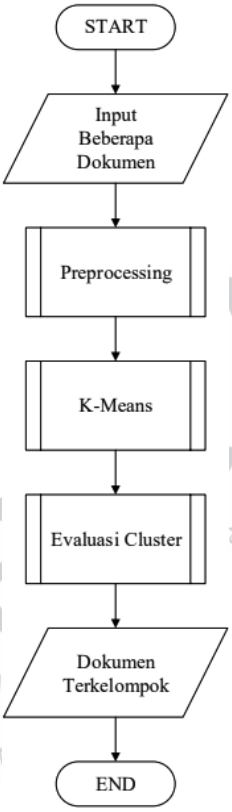
Saat ini dokumen abstrak skripsi jurusan Teknik Informatika dalam *library.itats.ac.id* hanya sebatas melakukan pencarian sesuai data yang diinginkan dan tidak dapat mengetahui asal fakultas dan jurusan dari dokumen abstrak skripsi dua jurusan Teknik Informatika tersebut. Dengan kata lain, "apa yang dicari, hal itu yang didapat". Hal ini belum bisa mengelompokkan dokumen abstrak penelitian berdasarkan topik yang sudah dikerjakan.

E-library ITATS sangat penting karena digunakan oleh dosen dan mahasiswa sebagai sumber referensi mencari ilmu pengetahuan. Oleh sebab itu, penataan dokumen-dokumen abstrak pada e-library juga sangat penting, termasuk dalam dokumen abstrak skripsi. Dokumen-dokumen abstrak skripsi yang ditata sesuai dengan kesamaannya akan memudahkan mahasiswa dan dosen dalam mencari referensi ketika akan mengerjakan atau hanya melihat topik-topik yang sudah dikerjakan oleh mahasiswa sebelumnya.

Pengelompokan dokumen abstrak skripsi diharapkan memudahkan pencarian yang spesifik dari setiap konsentrasi dalam jurusan. Dengan demikian, pencarian menjadi lebih terarah pada hal-hal yang ingin diketahui maupun hal-hal yang tidak ingin diketahui, serta memudahkan penggalian sumber ilmu pengetahuan yang berasal dari dokumen skripsi. Dokumen penelitian dapat dikelompokkan berdasarkan dengan kemiripan topik, objek, maupun dari metode penelitian. Hasil dari pengelompokan data penelitian tersebut dapat memperlihatkan adanya suatu pola. Dari pola-pola tersebut muncul sebuah informasi yang dapat digali. Pengelompokan dokumen abstrak skripsi ini umumnya berbentuk teks yang dapat dilakukan dengan text mining.

Metodologi penelitian yang digunakan dapat dilihat pada Gambar 6.6. Penelitian ini akan mengembangkan sistem yang dapat mengelompokkan dokumen abstrak skripsi yang sudah dikerjakan oleh

mahasiswa jurusan Teknik Informatika ITATS. Data yang diambil berupa abstrak skripsi 21 dengan format PDF yang kemudian akan diubah menjadi format plain text atau TXT. Pengubahan data menjadi plain text ini berfungsi untuk memudahkan program komputer.

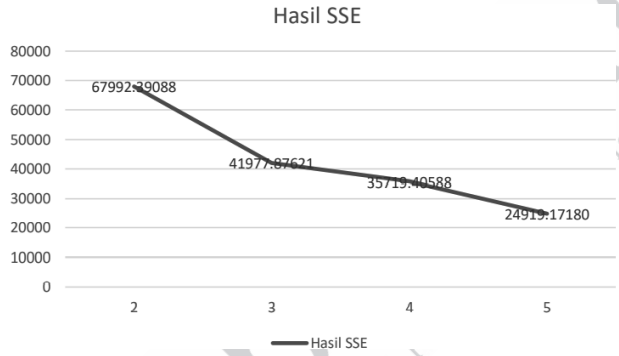


Gambar 6.6 Metodologi Penelitian Pengelompokan Text Mining

Setelah mengubah format, data siap diolah ke tahap pre-processing. Tahap ini terdiri dari beberapa tahap, yaitu case folding, tokenisasi, eliminasi stopword, stemming, term-frequency (TF), document frequency (DF), inverse document frequency (IDF), dan cosine similarity. Tahap setelah pre-processing adalah K-Means Clustering. Langkah pertama yang harus dilakukan adalah menetapkan nilai K atau jumlah cluster yang akan dibentuk. Penelitian ini melakukan uji coba

dengan nilai K=2, K=3, K=4 dan K=5. Kemudian menghitung x jarak data ke centroid dengan menggunakan Cosine Similarity. Cari jarak terdekat antara semua dokumen dengan dokumen yang dijadikan centroid.

Dokumen dengan jarak minimal terhadap centroid akan berkelompok membentuk cluster. Selanjutnya, tahap terakhir dari penelitian ini adalah evaluasi cluster menggunakan metode Elbow. Metode ini berfungsi untuk mengukur nilai K paling optimal pada metode K-Means.



Gambar 6.7 Hasil Nilai SSE dalam Elbow

Hasil yang diperoleh setelah dilakukan proses Text Mining dan Analisis Clustering dapat dilihat dalam Gambar 6.7. Dalam gambar tersebut, nilai yang optimal dari Teknik elbow diperoleh nilai K=2 dengan nilai SSE 4197. 2908.

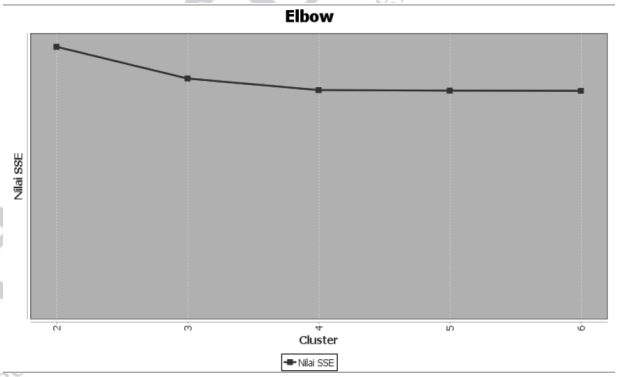
6.3 CLUSTERING UNTUK STATUS GIZI BALITA

Contoh dalam studi kasus berikutnya adalah pengelompokan data berdasarkan gizi balita. Pengukuran gizi balita diukur dari nilai umur, berat badan, dan tinggi badan. Metode clustering yang digunakan dalam contoh ini merupakan kombinasi dari dua jenis clustering, yaitu hierarki dengan metode Single Linkage dan partisi dengan metode K-Means.

Metodologi yang digunakan pada penelitian ini:

1. Input data.
2. Penentuan berapa cluster yang diharapkan.
3. Normalisasi Data.
4. Implementasi Single Linkage.
5. Implementasi K-Means.
6. Performa Hasil Cluster.

Seperti yang dijelaskan sebelumnya, input data penelitian ini memiliki tiga buat fitur, yaitu umur balita, berat badan, dan tinggi badan. Setelah pengambilan data, langkah berikutnya adalah inisialisasi cluster. Normalisasi digunakan untuk mengubah nilai dengan range 0-1. Untuk mendapatkan nilai awal metode K-Means, penelitian ini menggunakan algoritma single linkage (pada K-Means digunakan nilai random). Setelah mengimplementasi metode K-Means, hal terakhir adalah melakukan uji performa pada setiap nilai K yang di-input. Hasil yang diperoleh dari uji coba dapat dilihat pada Gambar 6.8.



Gambar 6.8 Nilai SSE untuk Uji Coba K

6.4 CLUSTERING UNTUK SEGMENTASI CITRA DIGITAL WAJAH MANUSIA

Apabila data yang digunakan sebelumnya adalah data nilai dan text, contoh kali ini clustering akan digunakan pada citra digital. Analisis clustering akan digunakan pada teknik segmentasi citra wajah manusia. Segmentasi citra adalah proses pemisahan objek (foreground) dan bukan objek (background).

Penelitian ini mengembangkan sebuah sistem yang mengimplementasikan segmentasi wajah manusia menggunakan metode K-Means dengan color space L*a*b. Lab yang digunakan adalah CIE L*a*b terdiri dari beberapa proses yang dimulai dari input, proses, dan output pada wajah manusia untuk membedakan antara objek wajah dengan background.

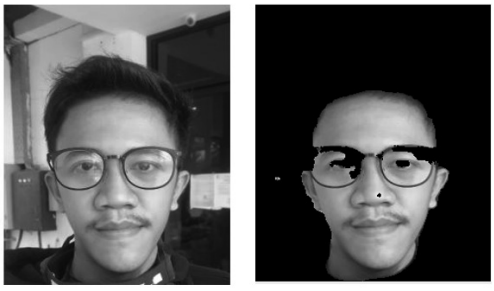
Metodologi penelitian ini, citra yang di-input terlebih dahulu telah dilakukan resize. Setelah resize, citra yang mempunyai nilai RGB diubah color space-nya dengan nilai l*a*b. Dari nilai l, a, dan b diambil nilai a serta b. Nilai ini yang akan menjadi fitur data untuk setiap pixel yang akan dilakukan proses clustering menggunakan K-Means. Proses clustering ini akan menghasilkan dua cluster, yaitu cluster pertama untuk wajah manusia dan cluster yang lain untuk background.

Pengujian penelitian ini menggunakan citra ground truth, yaitu membandingkan hasil segmentasi dengan citra hasil ground truth. Citra yang akan diuji sebanyak 40 gambar wajah manusia. Pengambilan gambar diambil dalam dua kondisi, yaitu pada siang hari di dalam ruangan dan siang hari di luar ruangan. Peneliti ingin menguji dengan cara membandingkan antara citra ground truth dengan hasil citra dari segmentasi K-Means.

Hasil dari konversi citra RGB ke l*a*b dapat dilihat pada Gambar 6.9 dan hasil segmentasi pada citra wajah manusia dengan clustering (K-Means) dapat dilihat pada Gambar 6.10.

CitraOriginal Citra L*a*b

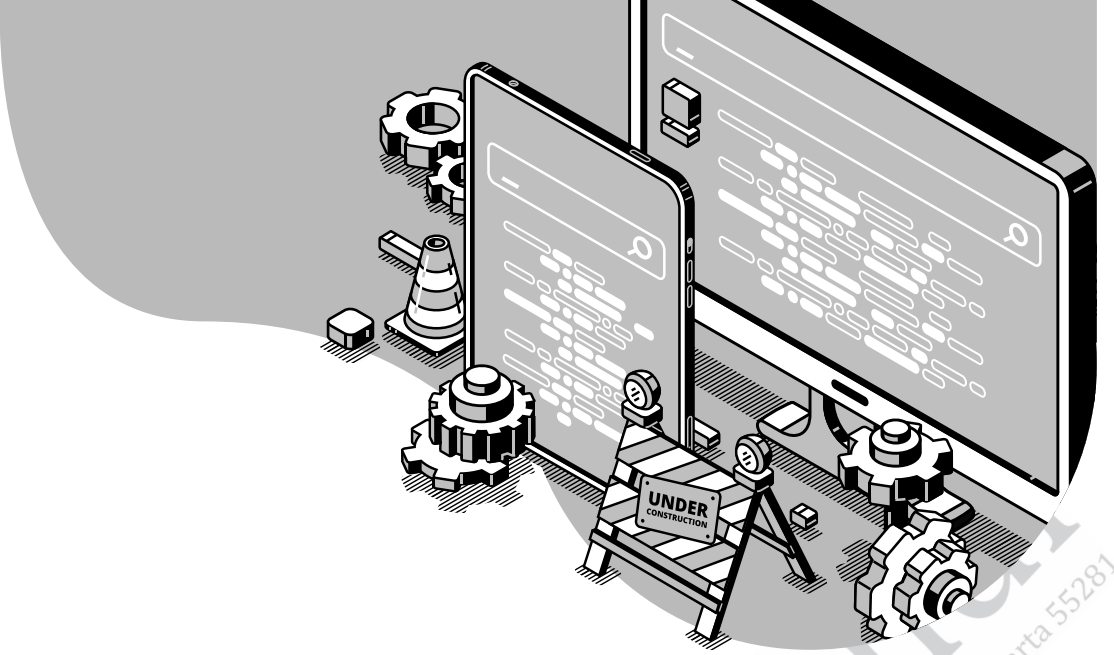
Gambar 6.9 Perbandingan Citra RGB dan l*a*b



CitraOriginal Citra L*a*b

Gambar 6.10 Hasil Segmentasi K-Means

Hasil pengujian K-Means menggunakan metode K-Means – L*a*b terhadap citra indoor dan outdoor dengan total dataset 40 citra yang terdiri dari citra indoor sebanyak 20 serta citra outdoor sebanyak 20 menghasilkan segmentasi wajah K-Means yang baik dan memberikan hasil dengan tingkat akurasi yang tinggi, yaitu citra di dalam ruangan (indoor) sebesar 99,64% serta citra di luar ruangan (outdoor) sebesar 99,29%.



DAFTAR PUSTAKA

Bezdek, James C., Robert Ehrlich, and William Full. 1984. "FCM: The Fuzzy C-Means Clustering Algorithm." *Computers and Geosciences* 10(2–3): 191–203.

Cohen-Addad, Vincent, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. 2018. "Hierarchical Clustering: Objective Functions and Algorithms." *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms* 66(4): 378–397.

Ester, M., H.P. Kriegel, J. Sander, and X. Xu. 1996. "A *Density*-Based Algorithm for Discovering Clusters." *KDD-96 Proceedings* 2: 226–231.

Fong, Simon, Saif Ur Rehman, Kamran Aziz, and Information Science. 2014. "DBSCAN: Past, Present, and Future." ICADIWIT. 232–238.

Han, Jiawei. 2012. "Cluster Analysis: Basic Concepts and Methods." In *Data Mining: Concepts and Techniques*, 443–495.

Han, Jiawei, Micheline Kamber, and Jian Pei. 2014. Elsevier *Data Mining Concepts and Techniques (Third Edition)*.

Hinneburg, Alexander, and Daniel A. Keim. 1998. "An Efficient Approach to Clustering in Large Multimedia Databases with Noise-Based

Clustering, Clustering of High-Dimensional Data, Clustering in Multimedia Databases, Clustering in the Presence of Noise." *Proceeding of the 4th International Conference on Knowledge Discovery and Data Mining (KDD)* (c): 58–65.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning with Application in R*.

Kaufman, Leonard, and Peter J. Rousseeuw. 1990. *Finding Groups in Data (An Introduction to Cluster Analysis)*.

Kohonen, Teuvo. 1982. "Self-Organized Formation of Topologically Correct Feature Maps." *Biological Cybernetics* 43(1): 59–69.

Kumar, Satyam. 2020. "Understanding K-Means, K-Means++ and K-Medoids Clustering Algorithms." *Medium*: 1–11.

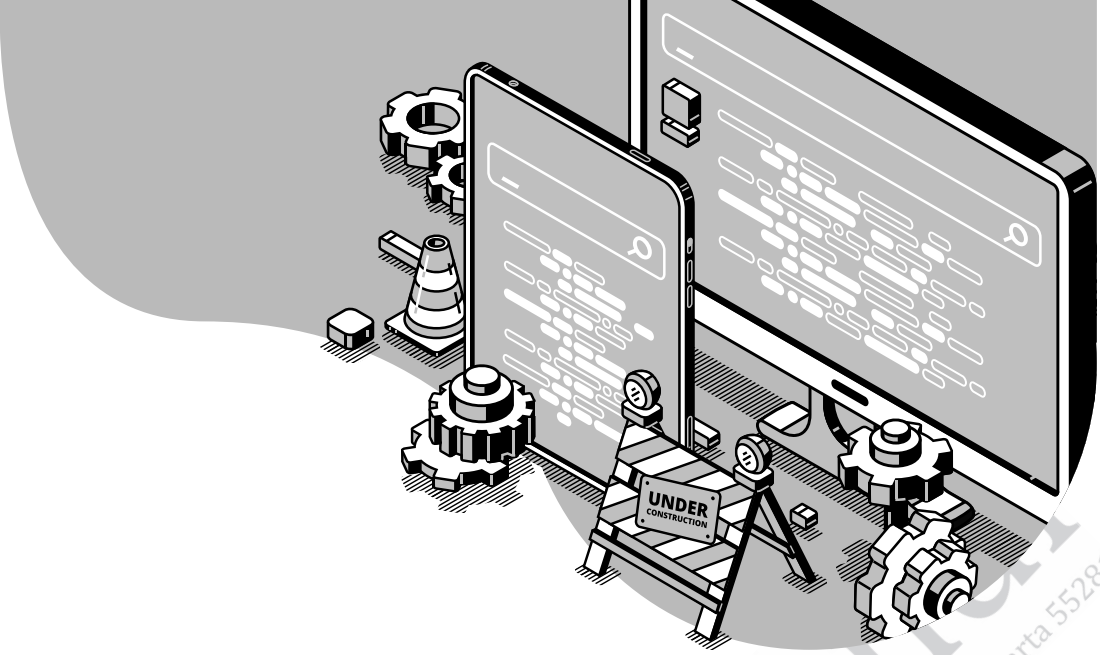
Kurniawan, Muchamad and Nanik Suciati. 2018. "Premise PArameter Optimization on Adaptive Network Based FuzzyInference System Using Modification Hybrid Particle Swarm Optimization Ad Genetic Algorithm." *Jurnal IPTEK*: 45–52.

Larose, Daniel T. 2014. *Discovering Knowledge in Data: An Introduction to Data Mining (Second Edition).*

Nietto, Paulo Rogério, and Maria Do Carmo Nicoletti. 2017. "Case Studies in Divisive Hierarchical Clustering." *International Journal of Innovative Computing and Applications* 8(2): 102–12.

Valente de Oliveira, J., and W. Pedrycz. 2007. Advances in Fuzzy Clustering and its Applications *Advances in Fuzzy Clustering and Its Applications*.

Vijaya, V., Shweta Sharma, and Neha Batra. 2019. "Comparative Study of Single Linkage, Complete Linkage, and Ward Method of Agglomerative Clustering." *Proceedings of the International Conference on Machine Learning, Big Data, Cloud and Parallel Computing: Trends, Prespectives and Prospects, COMITCon 2019*: 568–73.



BIOGRAFI PENULIS

Rani Rotul Muhima, S.Si., M.T., lahir pada tahun 1985 di Ponorogo, Jawa Timur. Menyelesaikan Sekolah Menengah Atas di SMAN 2 Ponorogo dan menyelesaikan studi S-1 Fisika di Universitas Airlangga, serta menyelesaikan S-2 Teknik Elektro di Institut Teknologi Sepuluh Nopember .

Sejak tahun 2015, penulis menjadi staf pengajar di Jurusan Teknik Informatika ITATS. Selain itu, penulis juga melakukan penelitian dan memublikasikan beberapa hasil penelitiannya di jurnal dan seminar, baik nasional maupun internasional. Peneliti saat ini berfokus pada aplikasi untuk membantu penanganan kebakaran hutan.

Muchamad Kurniawan, S.Kom., M.Kom. Penulis lahir di Surabaya pada tahun 1986. Penulis merupakan lulusan Sarjana dari Teknik Informatika kampus Institut Teknologi Adhi Tama Surabaya. Penulis menyelesaikan program Magister di Teknik Informatika (ITS Surabaya).

Penulis merupakan dosen di Jurusan Teknik Informatika ITATS sejak tahun 2015. Penulis juga aktif sebagai peneliti dan telah menghasilkan karya ilmiah yang diterbitkan pada jurnal serta seminar nasional dan internasional. Penulis juga telah menerbitkan buku *Perancangan Framework Machine Learning* pada tahun 2021 dan dua *book chapter* pada tahun yang sama.

Septiyawan Rosetya Wardhana, S.Kom., M.Kom. Penulis lahir di Surabaya pada tahun 1991. Penulis merupakan lulusan Sarjana dari Teknik Informatika kampus Institut Teknologi Adhi Tama Surabaya. Penulis menyelesaikan program Magister di Teknik Informatika (ITS Surabaya) dan saat ini merupakan dosen di Jurusan Teknik Informatika ITATS sejak tahun 2017. Penulis juga aktif sebagai peneliti dan telah menghasilkan karya ilmiah yang diterbitkan pada jurnal serta seminar nasional dan internasional.

Anton Yudhana, S.T., M.T., Ph.D. Penulis lahir di Purworejo pada tanggal 8 Agustus 1976. Telah menyelesaikan studi S-1 Teknik Elektro di ITS Surabaya (2001) dengan konsentrasi Telekomunikasi Multimedia, S-2 Teknik Elektro UGM (2005) dengan konsentrasi Pengolahan Sinyal, dan S-3 untuk Jurusan Teknik Elektro dengan konsentrasi *Radio Communication* dan *Signal Processing*.

Sejak tahun 2001 mengajar di Prodi Teknik Elektro Universitas Ahmad Dahlan, Yogyakarta. Saat ini mengajar S-2 Teknik Informatika di kampus yang sama untuk mata kuliah Komunikasi Data dan Jaringan Komputer.

Penelitian-penelitian yang dilakukan selama lima tahun terakhir berfokus kepada aplikasi-aplikasi automasi serta komunikasi di bidang pertanian dan kesehatan. Penelitian-penelitian tersebut telah mengantarkannya dalam program-program *research collaboration* dengan universitas di luar negeri, di antaranya *Scheme for Academic Mobility Exchange* (SAME) yang diselenggarakan Dikti Kemendikbud pada tahun 2013 di Massey University, New Zealand untuk topik penelitian *Wireless Sensor Network*.

Penelitian terbarunya adalah bertugas di Iwate Prefectural University Japan sebagai *visiting lecturer* dan mengerjakan kolaborasi riset di bidang *smart sensor* untuk difabel serta penderita stroke. Penulis dapat dihubungi melalui e-mail eyudhana@ee.uad.ac.id.

Penulis telah mendapat berbagai penghargaan, aktivitas penelitian, pengembangan dan inovasi, antara lain sebagai berikut.

1. Kepala LPPM UAD 2020-2024.
2. Wakil Direktur bidang Inovasi KUBI UAD 2019-2020.

3. Post Doctoral Program Massey University, New Zealand, tahun 2013.
4. Visiting Researcher di Iwate Prefectural University, Japan, tahun 2017.
5. Visiting Researcher di Sona College, Saleem, Tamil Nadu, India, tahun 2018.
6. Peneliti Berprestasi Universitas Ahmad Dahlan, tahun 2018.
7. Pembimbing bidang Penalaran Terbaik 1, Bimawa Awards 2019.
8. Visiting Researcher di Karabuk Bniversity, Turkey, tahun 2019.
9. Konsultan Ahli Revitalisasi SMK, Kemendikbud (2019).
10. Pengurus HKTI Bantul bidang Inovasi dan Teknologi.
11. Inventor dan Founder Produk AgriPrecision (Simonkori, PuDang, E-Kalpin).
12. Penghargaan sebagai Dosen dengan Publikasi Terbaik FTI tahun 2020.
13. Inovator UAD terkait Produk Tepat Guna Penanggulangan Covid-19 pada Milad ke-60 UAD.
14. Pembimbing PKM dan Inovator Mahasiswa tingkat nasional serta internasional (Dubai, Shanghai, Thailand, dan Malaysia).
15. Keynote Speaker di beragam International Conference.
16. Pembina Ramada FM.

Sunardi, S.T., M.T., Ph.D. Penulis lahir di Sragen pada tahun 1974. Penulis menyelesaikan S-1 di Universitas Gadjah Mada tahun 1999, S-2 di Institut Teknologi Bandung tahun 2003, dan S-3 di Universiti Teknologi Malaysia tahun 2011.

Penulis merupakan dekan Fakultas Teknologi Industri Universitas Ahmad Dahlan Yogyakarta.

Penulis memiliki pengalaman penelitian dalam bidang *Geographical Information System, Forensic Analysis, Steganography, Internet of Things,* dan lainnya.

Selain aktif di bidang penelitian, penulis juga aktif dan memiliki pengalaman sebagai pembicara pada suatu kegiatan, antara lain "Webinar Kurikulum Berbasis MBKM: Peninjauan dan Evaluasi Kurikulum Perguruan Tinggi untuk Mengakomodasi", "Mengenal Big Data", "Seminar Lokakarya Kurikulum Berbasis MBKM", dan lainnya.

Weny Mistarika Rahmawati, S.Kom., M.Kom., M.Sc. Penulis lahir di Magetan pada tanggal 3 April 1991. Penulis mendapat gelar Sarjana dari Teknik Informatika Institut Teknologi Sepuluh Nopember (2013). Mendapat gelar Magister dari program *Double Degree* dari Institut Teknologi Sepuluh November dan Asian Institute of Technology, Thailand (2015)

Saat ini penulis berstatus sebagai dosen tetap jurusan Teknik Informatika Institut Teknologi Adhi Tama Surabaya dengan bidang sistem cerdas.

Selama ini penulis telah menghasilkan beberapa *book chapter* berkaitan dengan teknologi informasi yang merupakan buku pertama yang dibuat oleh penulis. Penulis dapat dihubungi pada e-mail wenymistarika@gmail.com.

Gusti Eka Yuliastuti, S.Kom., M.Kom. Penulis lahir di Surabaya pada tahun 1994. Penulis memperoleh gelar Sarjana pada tahun 2016 dan gelar Magister pada tahun 2018 dari Fakultas Ilmu Komputer Universitas Brawijaya Malang.

Penulis merupakan dosen tetap Jurusan Teknik Informatika Institut Teknologi Adhi Tama Surabaya sejak 2019. Selain aktif dalam bidang pendidikan, penulis juga aktif dalam bidang penelitian dan pengabdian. Beberapa hasil penelitian penulis dimuat pada jurnal nasional terakreditasi dan jurnal internasional bereputasi. Bidang penelitian penulis yakni Algoritma Evolusi.