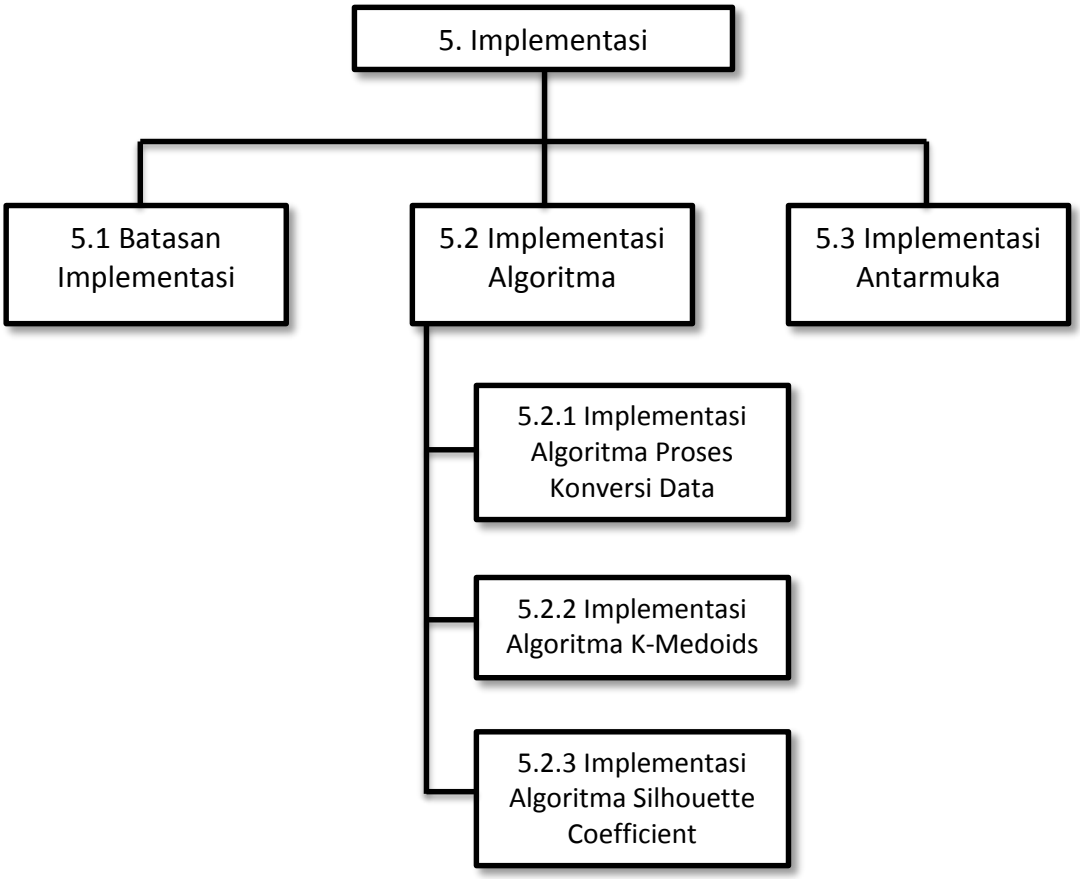
BAB 5 IMPLEMENTASI

Pada bab ini membahas tentang implementasi "Clustering Pasien Kanker Berdasarkan Struktur Protein Dalam Tubuh Menggunakan Metode K-Medoids". Implementasi yang dilakukan telah disesuaikan dengan tahap perancangan yang telah dibahas pada bab sebelumnya. Pohon implementasi pada sistem ini meliputi tiga tahap yaitu batasan implementasi, implementasi algoritma, dan implementasi antarmuka. Implementasi algoritma terdiri dari implementasi algoritma proses konversi data, implementasi algoritma K-Medoids dan implementasi algoritma Silhouette Coefficient. Pohon implementasi pada sistem ini ditunjukkan pada Gambar 5.1 berikut.



Gambar 5.1 Pohon Implementasi

5.1 Batasan Implementasi

Dalam proses pembangunan sistem clustering pasien kanker berdasarkan struktur protein dalam tubuh menggunakan metode K-Medoids ini terdapat beberapa batasan implementasi yang diberikan.

1. Sistem yang dibangun adalah aplikasi berbasis desktop yang menggunakan bahasa pemrograman java.

2. Inputan yang dilakukan oleh pengguna adalah data struktur protein dalam tubuh yang telah mengalami proses mutasi gen dan data disimpan dalam bentuk file xml.
3. Data struktur protein dalam bentuk xml harus memiliki panjang data yang sama dengan panjang data *wild* yaitu 393 karakter.
4. Pengguna hanya menginputkan 2 file data dalam bentuk xml, yaitu dataset protein dan data *wild*, juga menginputkan nilai *k* atau jumlah cluster yang diinginkan.
5. Output yang diberikan oleh sistem adalah Tabel dataset protein, hasil clustering, dan hasil kualitas cluster.
6. Semua output ditampilkan dalam satu halaman.

Batasan – batasan implementasi ini diberikan dengan harapan agar hasil clustering bisa menjadi lebih baik.

5.2 Implementasi Algoritma

Dalam sistem clustering pasien kanker menggunakan metode K-Medoids ini terdapat tiga proses utama yaitu proses konversi data yang diawali dengan mengambil data yang akan diolah dari file berbentuk xml, data yang diambil adalah data bertipe String yang akan dikonversi ke dalam bentuk integer melalui Tabel PAM. Proses yang kedua adalah perhitungan oleh metode K-Medoids, setelah data dikonversi ke dalam bentuk integer maka perhitungan K-Medoids bisa dilakukan sesuai dengan tahap-tahapannya dan akan menghasilkan cluster. Dan proses yang ketiga adalah perhitungan Silhouette Coefficient, setelah cluster telah terbentuk dan semua data sudah masuk dalam clusternya masing-masing maka akan dilakukan proses untuk mengetahui kualitas cluster.

5.2.1 Implementasi Algoritma Proses Konversi Data

Proses konversi data dilakukan dengan tujuan untuk mengubah data asli yang bertipe String menjadi data virtual yang bertipe integer agar bisa dilakukan proses perhitungan pada tahapan selanjutnya. Algoritma proses konversi data akan ditampilkan pada source code 5.1.

```
1.      private ListData listData = new ListData();
2.      private int[][] dataPAM = {
            {2, -2, 0, 0, -2, 0, 0, 1, -1, 0, -2, -1, -1, -3, 1, 1, 1, -6, -4, 0},
            {-2, 6, 0, -1, -4, 1, -1, -3, 2, -2, -3, 3, 0, -4, 0, 0, -1, 2, -5, -2},
            {0, 0, 2, 2, -3, 1, 2, 1, 2, -2, -3, 1, -2, -3, 0, 1, 1, -4, -2, -2},
            {0, -1, 2, 4, -5, 2, 3, 1, 1, -2, -4, 0, -3, -5, -1, 0, 0, -7, -4, -2},
            {-2, -3, -4, -5, 12, -5, -5, -4, -3, -3, -6, -5, -5, -4, -2, 0, -2, -8, 0, -2},
```

	{0, 1, 1, 2, -5, 4, 2, -1, 3, -2, -2, 1, -1, -4, 0, -1, -1, -5, -4, -2}, {0, -1, 1, 3, -5, 2, 4, 0, 1, -2, -3, 0, -2, -5, 0, 0, 0, -7, -4, -2}, {1, -3, 0, 1, -3, -1, 0, 5, -2, -2, -4, -2, -3, -5, 0, 1, 0, -7, -5, -1}, {-1, 1, 1, 1, -3, 3, 0, -3, 6, -3, -3, 0, -3, -2, 0, -1, -1, -3, 0, -3}, {-1, -2, -2, -2, -2, -2, -2, -3, -3, 4, 2, -2, 2, 1, -2, -1, 0, -5, -1, 4}, {-2, -3, -3, -4, -6, -2, -3, -4, -2, 2, 6, -2, 4, 2, -2, -3, -2, -2, -1, 2}, {-1, 3, 1, 0, -5, 1, 0, -2, 0, -2, -3, 5, 0, -5, -1, 0, 0, -4, -5, -2}, {-1, -1, -2, -3, -5, -1, -2, -3, -2, 2, 4, 1, 6, 0, -2, -2, 0, -4, -3, 2}, {-3, -4, -3, -5, -4, -4, -5, -4, -2, 1, 2, -5, 0, 9, -5, -3, -3, 0, 7, -1}, {1, 0, 0, -1, -3, 0, 0, 0, 0, -2, -2, -1, -2, -4, 6, 1, 0, -6, -5, -1}, {2, 1, 2, 1, 1, 0, 1, 2, 0, -1, -2, 1, -1, -2, 2, 2, 2, -2, -2, 0}, {0, -2, 0, -1, -3, -2, -1, -1, -2, -1, -2, -1, -1, -4, 0, 1, 2, -6, -4, 0}, {-6, 2, -5, -7, -7, -6, -7, -7, -5, -6, -7, -4, -6, 1, -6, -2, -5, 17, 1, -8}, {-3, -5, -2, -4, 1, -4, -4, -5, 0, -1, -1, -5, -2, 7, -5, -3, -3, 0, 10, -2}, {0, -2, -2, -2, -2, -2, -2, -2, -2, 4, 2, -2, 2, -1, -1, -1, 0, -6, -3, 4} };
3.	private String[] data*Wild*, labelDataPAM = {"A", "R", "N", "D", "C", "Q", "E", "G", "H", "I", "L", "K", "M", "F", "P", "S", "T", "W", "Y", "V"};
4.	private String[][] data*Wild*_STR;
5.	private String TR = "TR", WL = "WL",
6.	srcTR = "latih.xml", srcTS = "uji.xml", srcWL = "*wild*.xml";
7.	public void setDataMaster(String code) throws ParserConfigurationException, SAXException, IOException {
8.	String[][] data2;
9.	int dataKelas2[];
10.	File path;
11.	Document document;
12.	DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
13.	DocumentBuilder builder = builderFactory.newDocumentBuilder();

```
14.         if (code.equals(this.TR)) {
15.             path = new File(this.srcTR);
16.             document = builder.parse(path);
17.         } else {
18.             path = new File(this.srcWL);
19.             document = builder.parse(path);
20.         }
21.         Element hasil = (Element)
document.getElementsByTagName("hasil").item(0);
22.         NodeList list = hasil.getElementsByTagName("protein");
23.         Element protein1 = (Element) list.item(0);
24.         Node a = protein1.getElementsByTagName("isi").item(0);
25.         data2 = new
String[list.getLength()][a.getTextContent().length()];
26.         dataKelas2 = new int[list.getLength()];
27.         for (int i = 0; i < list.getLength(); i++) {
28.             Element protein = (Element) list.item(i);
29.             Node isi = protein.getElementsByTagName("isi").item(0);
30.             data2[i] = isi.getTextContent().split("");
31.             Node kelas =
protein.getElementsByTagName("kelas").item(0);
32.             dataKelas2[i] =
Integer.valueOf(kelas.getTextContent());
33.         }
34.         if (code.equals(TR)) {
35.             this.listData.setJumlahData(data2.length);
36.             this.listData.setPanjangGen(data2[0].length);
37.             ArrayList<Data> listDt = new ArrayList<Data>();
38.             for (int i = 0; i < data2.length; i++) {
39.                 Data data = new Data();
40.                 String[] dataSTR = new String[data2[0].length];
41.                 for (int j = 0; j < data2[0].length; j++) {
42.                     dataSTR[j] = data2[i][j];
43.                 }
44.                 int kelasINT = dataKelas2[i];
45.                 data.setDataSTR(dataSTR);
46.                 data.setKelasINT(kelasINT);
47.                 data.setIsEverMedoid(false);
48.                 data.setIdData(i);
49.                 listDt.add(data);
50.             }
```

```
51.             this.listData.setListData(listDt);
52.         } else {
53.             data_Wild_STR = new String[data2.length][data2[0].length
    - 1];
54.             data_Wild = new String[data2[0].length - 1];
55.             for (int i = 0; i < data_Wild_STR.length; i++) {
56.                 for (int j = 0; j < data_Wild_STR[0].length; j++) {
57.                     data_Wild_STR[i][j] = data2[i][j + 1];
58.                     data_Wild[j] = data_Wild_STR[i][j];
59.                 }
60.             }
61.         }
62.     }
63.     public void setData() {
64.         int idxBrs, idxKlm;
65.         for (int i = 0; i < this.listData.getJumlahData(); i++) {
66.             double[] dataDOU = new
    double[this.listData.getPanjangGen()];
67.             for (int j = 0; j < this.listData.getPanjangGen(); j++)
    {
68.                 idxBrs = -1;
69.                 idxKlm = -1;
70.                 for (int k = 0; k < labelDataPAM.length; k++) {
71.                     String sLabel = labelDataPAM[k];
72.                     if
    (this.listData.getListData().get(i).getDataSTR()[j].equals(sLabel))
    {
73.                         idxBrs = k;
74.                     }
75.                     if (data_Wild_[j].equals(sLabel)) {
76.                         idxKlm = k;
77.                     }
78.                 }
79.                 dataDOU[j] = dataPAM[idxBrs][idxKlm];
80.             }
81.             this.listData.getListData().get(i).setDataDOU(dataDOU);
82.         }
83.     }
```

Kode Program 5.1 Proses Konversi Data

Penjelasan fungsi dari Kode Program 5.1 :

1. Baris ke-3 sampai baris ke-22 adalah Tabel pam yang digunakan untuk konversi data dari String menjadi integer dengan menggunakan array dua dimensi.
2. Baris ke-34 sampai baris ke-43 merupakan konversi file xml agar dapat terbaca oleh java.
3. Baris ke-44 merupakan fungsi untuk mengambil tag “hasil”. Hasil ini berisi semua data yang dibutuhkan.
4. Baris ke-45 dan baris ke-46 merupakan fungsi untuk mengambil tag “protein”. Protein ini berisi dua jenis data yaitu isi protein atau data asli yang akan dilakukan proses perhitungan dan data kelas untuk masing-masing isi protein.
5. Baris ke-50 sampai baris ke-56 merupakan proses perulangan untuk mengambil “isi” dan “kelas”. Isi berisi data asli protein yang akan dilakukan proses konversi dan perhitungan data yang memiliki panjang 393 karakter. Kelas berisi data kelas untuk setiap data yaitu 0 untuk *non-cancer* (NC), 1 untuk *breast cancer* (BC), 2 untuk *colorectal cancer* (CC), dan 3 untuk *lung cancer* (LC). Proses perulangan ini akan berlangsung sebanyak jumlah data yang digunakan.
6. Baris ke-84 sampai baris ke-104 merupakan proses konversi oleh Tabel PAM dari karakter yang bertipe String menjadi angka-angka yang bertipe integer dan siap untuk dilakukan proses perhitungan menggunakan metode K-Medoid.

5.2.2 Implementasi Algoritma K-Medoids

Setelah data berhasil dikonversikan ke dalam bentuk integer maka data akan diolah menggunakan metode K-Medoids sehingga proses clusterisasi bisa dilakukan. Ada beberapa tahap yang harus dilakukan dalam metode K-Medoids sampai terbentuknya beberapa cluster yang diinginkan. Untuk lebih jelasnya algoritma K-Medoids akan ditampilkan pada Kode Program 5.2.

```
1.      private ListCluster listCluster = new ListCluster(),
listClusterAwal = new ListCluster();
2.      private ListData listData;
3.      private boolean isStop = false;
4.      public KMedoid(int k, ListData listData) {
5.          this.listCluster.setK(k);
6.          this.listClusterAwal.setK(k);
7.          this.listData = listData;
8.      }
9.      private boolean isPusatCluster(int id) {
10.         ` boolean isPusatCluster = false;
11.         for (int i = 0; i < this.listCluster.getK(); i++) {
```

```
12.             if (this.listCluster.getListCluster().get(i).isChange()) {
13.                 if (id == this.listCluster.getListCluster().get(i).getPusatCluster2()) {
14.                     return true;
15.                 }
16.             } else {
17.                 if (id == this.listCluster.getListCluster().get(i).getPusatCluster1()) {
18.                     return true;
19.                 }
20.             }
21.         }
22.         return false;
23.     }
24.     private double[] hitungJarak(int pusatCluster) {
25.         double[] cost = new double[this.listData.getJumlahData()];
26.         double[] dataPusatCluster = this.listData.getListData().get(pusatCluster).getDataDOU();
27.         for (int i = 0; i < dataPusatCluster.length; i++) {
28.         }
29.         for (int i = 0; i < this.listData.getJumlahData(); i++) {
30.             if (!this.isPusatCluster(i)) {
31.                 double[] dataDOU = this.listData.getListData().get(i).getDataDOU();
32.                 for (int j = 0; j < this.listData.getPanjangGen(); j++) {
33.                     cost[i] += Math.abs(dataDOU[j] - dataPusatCluster[j]);
34.                 }
35.             }
36.         }
37.         return cost;
38.     }
39.     private double[][] cariClusterData(double[][] cost) {
40.         double[][] clusterData = new double[this.listData.getJumlahData()][2];
41.         for (int i = 0; i < this.listData.getJumlahData(); i++) {
42.             double minCost = 9999;
43.             for (int j = 0; j < this.listCluster.getK(); j++) {
44.                 if (!this.isPusatCluster(i)) {
45.                     if (cost[i][j] < minCost) {
46.                         minCost = cost[i][j];
```

```
47.                             clusterData[i][0] = minCost;
48.                             clusterData[i][1] = j;
49.                         }
50.                     }
51.                 }
52.             }
53.             return clusterData;
54.         }
55.         private double[][] hitungCost(int itr) {
56.             double[][] cost = new
    double[this.listData.getJumlahData()][this.listCluster.getK()];
57.             for (int i = 0; i < this.listCluster.getK(); i++) {
58.                 int pusatCluster;
59.                 if (itr == 0) {
60.                     pusatCluster =
    this.listCluster.getListCluster().get(i).getPusatCluster1();
61.                 } else if
    (this.listCluster.getListCluster().get(i).isChange()) {
62.                     pusatCluster =
    this.listCluster.getListCluster().get(i).getPusatCluster2();
63.                 } else {
64.                     pusatCluster =
    this.listCluster.getListCluster().get(i).getPusatCluster1();
65.                 }
66.                 double[] tempCost = this.hitungJarak(pusatCluster);
67.                 for (int j = 0; j < this.listData.getJumlahData(); j++)
    {
68.                     cost[j][i] = tempCost[j];
69.                 }
70.             }
71.             return this.cariClusterData(cost);
72.         }
73.         private void setUrutanCluster() {
74.             int cluster[][] = new int[this.listCluster.getK()][2];
75.             for (int i = 0; i < this.listCluster.getK(); i++) {
76.                 cluster[i][0] =
    this.listCluster.getListCluster().get(i).getAnggotaCluster().size()
    ;
77.                 cluster[i][1] = i;
78.             }
79.                 public int compare(int[] a, int[] b) {
80.                     return Integer.compare(a[0], b[0]);
81.                 }
```

```
82.         });
83.         ArrayList<Cluster> listCluster = new ArrayList<Cluster>();
84.         for (int i = 0; i < this.listCluster.getK(); i++) {
85.
listCluster.add(this.listCluster.getListCluster().get(cluster[i][1]
));
86.         }
87.         this.listCluster.setListCluster(listCluster);
88.         this.listClusterAwal.setListCluster(listCluster);
89.     }
90.     private void setAnggotaCluster(double[][] clusterData, int itr)
{
91.         ArrayList<Cluster> listCluster = new ArrayList<Cluster>();
92.         for (int i = 0; i < this.listCluster.getK(); i++) {
93.             int pusatCluster;
94.             if
(this.listCluster.getListCluster().get(i).isChange()) {
95.                 pusatCluster =
this.listCluster.getListCluster().get(i).getPusatCluster2();
96.             } else {
97.                 pusatCluster =
this.listCluster.getListCluster().get(i).getPusatCluster1();
98.             }
99.             ArrayList<Data> anggotaCluster = new ArrayList<Data>();
100.
anggotaCluster.add(this.listData.getListData().get(pusatCluster));
101.            Cluster cluster = new Cluster();
102.
cluster.setIdCluster(this.listCluster.getListCluster().get(i).getId
Cluster());
103.
cluster.setIsChange(this.listCluster.getListCluster().get(i).isChan
ge());
104.
cluster.setPusatCluster1(this.listCluster.getListCluster().get(i).g
etPusatCluster1());
105.
cluster.setPusatCluster2(this.listCluster.getListCluster().get(i).g
etPusatCluster2());
106.
cluster.setS(this.listCluster.getListCluster().get(i).getS());
107.            for (int j = 0; j < this.listData.getJumlahData(); j++)
{
108.                if (!this.isPusatCluster(j) && ((int)
clusterData[j][1] == i)) {
109.
anggotaCluster.add(this.listData.getListData().get(j));
```

```
110.                 }
111.             }
112.             cluster.setAnggotaCluster(anggotaCluster);
113.             cluster.setPusatCluster1(pusatCluster);
114.             listCluster.add(cluster);
115.         }
116.         this.listCluster.setListCluster(listCluster);
117.         if (itr == 0) {
118.             this.listClusterAwal.setListCluster(listCluster);
119.             this.setUrutanCluster();
120.         }
121.     }
122.     private boolean isClusterCompleted(int idCluster) {
123.         for (int i = 0; i < this.listClusterAwal.getK(); i++) {
124.             Cluster cluster =
this.listClusterAwal.getListCluster().get(i);
125.             if (cluster.getIdCluster() == idCluster) {
126.                 int itr = 0;
127.                 for (int j = 0; j <
cluster.getAnggotaCluster().size(); j++) {
128.                     int idData =
cluster.getAnggotaCluster().get(j).getIdData();
129.                     if
(this.listData.getListData().get(idData).isIsEverMedoid()) {
130.                         itr++;
131.                     }
132.                 }
133.                 if (itr == cluster.getAnggotaCluster().size()) {
134.                     return true;
135.                 } else {
136.                     return false;
137.                 }
138.             }
139.         }
140.         return false;
141.     }
142.     private int[] cariAnggotaCluster(int idData, int idCluster) {
143.         int[] anggotaCluster = new int[2];
144.         for (int i = 0; i < this.listCluster.getK(); i++) {
145.             Cluster cluster =
this.listCluster.getListCluster().get(i);
```

```
146.            for (int j = 0; j < cluster.getAnggotaCluster().size();
        j++) {
147.                if (idData ==
        cluster.getAnggotaCluster().get(j).getIdData()) {
148.                    if (isClusterCompleted(cluster.getIdCluster()))
        {
149.                        Cluster clstr1 = new Cluster();
150.
        clstr1.setIdCluster(cluster.getIdCluster());
151.                        clstr1.setIsChange(cluster.isChange());
152.
        clstr1.setPusatCluster1(cluster.getPusatCluster1());
153.
        clstr1.setPusatCluster2(cluster.getPusatCluster2());
154.                        clstr1.setS(cluster.getS());
155.                        ArrayList<Data> anggotaClstr1 = new
        ArrayList<Data>();
156.                        int idData1 = 0;
157.                        for (int k = 0; k <
        cluster.getAnggotaCluster().size(); k++) {
158.                            if
        (cluster.getAnggotaCluster().get(k).getIdData() != idData) {
159.
        anggotaClstr1.add(cluster.getAnggotaCluster().get(k));
160.                            } else {
161.                                idData1 = k;
162.                            }
163.                        }
164.                        clstr1.setAnggotaCluster(anggotaClstr1);
165.                        this.listCluster.getListCluster().set(i,
        clstr1);
166.
        this.listCluster.getListCluster().get(i).setIsKurang(true);
167.
        this.listCluster.getListCluster().get(i).setIdKurang(idData1);
168.
        this.listCluster.getListCluster().get(i).setValKurang(idData);
169.                        Cluster cluster2 = null;
170.                        int namaCluster = 0;
171.                        for (int k = 0; k <
        this.listCluster.getK(); k++) {
172.                            if
        (this.listCluster.getListCluster().get(k).getIdCluster() ==
        idCluster) {
173.                                cluster2 =
        this.listCluster.getListCluster().get(k);
174.                                namaCluster = k;
```

```
175.                                    break;
176.                                }
177.                            }
178.                            Cluster clstr2 = new Cluster();
179.    clstr2.setIdCluster(cluster2.getIdCluster());
180.                            clstr2.setIsChange(cluster2.isChange());
181.    clstr2.setPusatCluster1(cluster2.getPusatCluster1());
182.    clstr2.setPusatCluster2(cluster2.getPusatCluster2());
183.                            clstr2.setS(cluster2.getS());
184.                            ArrayList<Data> anggotaClstr2 = new
        ArrayList<Data>();
185.                            int idData2 = 0;
186.                            boolean isAdded = false;
187.                            for (int k = 0; k <
        cluster2.getAnggotaCluster().size(); k++) {
188.                                if
        (cluster2.getAnggotaCluster().get(k).getIdData() > idData) {
189.    anggotaClstr2.add(this.listData.getListData().get(idData));
190.    anggotaClstr2.add(this.listData.getListData().get(k));
191.                                    if (!isAdded) {
192.                                        idData2 = k;
193.                                        isAdded = true;
194.                                    }
195.                                } else {
196.    anggotaClstr2.add(cluster2.getAnggotaCluster().get(k));
197.                                    if ((k + 1) ==
        cluster2.getAnggotaCluster().size()) {
198.    anggotaClstr2.add(this.listData.getListData().get(idData));
199.                                        idData2 = k + 1;
200.                                    }
201.                                }
202.                            }
203.                            clstr2.setAnggotaCluster(anggotaClstr2);
204.    this.listCluster.getListCluster().set(namaCluster, clstr2);
205.    this.listCluster.getListCluster().get(namaCluster).setIsTambah(true
        );
```

```
206.
this.listCluster.getListCluster().get(namaCluster).setIdTambah(idDa
ta2);
207.                            anggotaCluster[0] = namaCluster;
208.                            anggotaCluster[1] = idData2;
209.                            return anggotaCluster;
210.                        } else {
211.                            if (idData ==
cluster.getAnggotaCluster().get(j).getIdData()) {
212.                                anggotaCluster[0] = i;
213.                                anggotaCluster[1] = j;
214.                                return anggotaCluster;
215.                            }
216.                        }
217.                    }
218.                }
219.            }
220.            return anggotaCluster;
221.        }
222.        private int setPusatCluster() {
223.            for (int i = 0; i < this.listClusterAwal.getK(); i++) {
224.                Cluster cluster =
this.listClusterAwal.getListCluster().get(i);
225.                for (int j = 0; j < cluster.getAnggotaCluster().size();
j++) {
226.                    if
(!cluster.getAnggotaCluster().get(j).isIsEverMedoid()) {
227.                        int pusatCluster =
cluster.getAnggotaCluster().get(j).getIdData();
228.                        int[] anggotaCluster =
this.cariAnggotaCluster(pusatCluster, cluster.getIdCluster());
229.
this.listData.getListData().get(pusatCluster).setIsEverMedoid(true)
;
230.
this.listClusterAwal.getListCluster().get(i).getAnggotaCluster().ge
t(j).setIsEverMedoid(true);
231.                        for (int k = 0; k < this.listCluster.getK();
k++) {
232.
this.listCluster.getListCluster().get(k).setIsChange(false);
233.                        }
234.
this.listCluster.getListCluster().get(anggotaCluster[0]).getAnggota
Cluster().get(anggotaCluster[1]).setIsEverMedoid(true);
```

```
this.listCluster.getListCluster().get(anggotaCluster[0]).setPusatCluster2(pusatCluster);
this.listCluster.getListCluster().get(anggotaCluster[0]).setIsChange(true);
                    Sistem.out.println("Next Medoid: " + pusatCluster);
                    return i;
                }
            }
        }
        return -1;
    }
    public ListCluster process() {
        int itr = 0;
        do {
            if (itr == 0) {
                ArrayList<Integer> listPusatCluster = new ArrayList<Integer>();
                for (int i = 0; i < this.listData.getJumlahData(); i++) {
                    listPusatCluster.add(i);
                }
                Collections.shuffle(listPusatCluster);
                ArrayList<Cluster> listCluster = new ArrayList<Cluster>();
                for (int i = 0; i < this.listCluster.getK(); i++) {
                    ArrayList<Data> anggotaCluster = new ArrayList<Data>();
this.listData.getListData().get(listPusatCluster.get(i)).setIsEverMedoid(true);
anggotaCluster.add(this.listData.getListData().get(listPusatCluster.get(i)));
                    Cluster cluster = new Cluster();
                    cluster.setIdCluster(i);
                    cluster.setAnggotaCluster(anggotaCluster);
cluster.setPusatCluster1(listPusatCluster.get(i));
                    listCluster.add(cluster);
                }
                this.listCluster.setListCluster(listCluster);
                this.listClusterAwal.setListCluster(listCluster);
```

```
266.                 double clusterData[][] = this.hitungCost(itr),
totalCost = 0;
267.                 for (int i = 0; i < this.listData.getJumlahData();
i++) {
268.                     totalCost += clusterData[i][0];
269.                 }
270.                 this.listCluster.setTotalCost(totalCost);
271.                 this.setAnggotaCluster(clusterData, itr);
272.             } else {
273.                 int val = setPusatCluster();
274.                 if (val == -1) {
275.                     this.isStop = true;
276.                 } else {
277.                     double clusterData[][] = this.hitungCost(itr),
totalCost = 0;
278.                     for (int i = 0; i <
this.listData.getJumlahData(); i++) {
279.                         totalCost += clusterData[i][0];
280.                     }
281.                     if (totalCost <
this.listCluster.getTotalCost()) {
282.                         this.listCluster.setTotalCost(totalCost);
283.                         this.setAnggotaCluster(clusterData, itr);
284.                     } else {
285.                         for (int i = 0; i <
this.listCluster.getK(); i++) {
286.                             if
(this.listCluster.getListCluster().get(i).isIsTambah()) {
287.                                 int idRemove =
this.listCluster.getListCluster().get(i).getIdTambah();
288.
this.listCluster.getListCluster().get(i).getAnggotaCluster().remove
(idRemove);
289.
this.listCluster.getListCluster().get(i).setIsTambah(false);
290.                                 break;
291.                             }
292.                         }
293.                         for (int i = 0; i <
this.listCluster.getK(); i++) {
294.                             if
(this.listCluster.getListCluster().get(i).isIsKurang()) {
295.                                 int idTambah =
this.listCluster.getListCluster().get(i).getIdKurang();
296.                                 int valKurang =
this.listCluster.getListCluster().get(i).getValKurang();
```

```
297.                                    Data data =
    this.listData.getListData().get(valKurang);
298.
    this.listCluster.getListCluster().get(i).getAnggotaCluster().add(id
    Tambah, data);
299.
    this.listCluster.getListCluster().get(i).setIsKurang(false);
300.                                    break;
301.                                }
302.                            }
303.                        }
304.                    }
305.                }
306.                itr++;
307.            } while (!isStop);
308.            return this.listCluster;
309.        }
310.    }
```

Kode Program 5.2 Algoritma K-Medoids

Penjelasan dari Kode Program 5.2 :

1. Baris ke-9 sampai baris ke-23 adalah sebuah fungsi untuk menentukan medoid awal pada algoritma K-Medoids sebanyak nilai *k* yang diinputkan.
2. Baris ke-24 sampai baris ke-38 adalah untuk menghitung jarak manhattan antara data dengan medoid awal yang telah ditentukan pada proses sebelumnya dan dihitung cost dari masing-masing data.
3. Baris ke-39 sampai baris ke-54 adalah untuk menentukan nilai cost minimal pada masing-masing data dan mengelompokkan data pada cluster yang sesuai dengan perhitungan.
4. Baris ke-72 sampai baris ke-121 adalah sebuah method untuk mengurutkan cluster dan data anggota cluster.
5. Baris ke-142 sampai baris ke-220 adalah untuk mencari seluruh anggota cluster sesuai dengan jumlah cluster yang diinginkan pengguna.
6. Baris ke-222 sampai baris ke-243 adalah untuk mengganti salah satu medoid dengan medoid yang baru.
7. Baris ke-244 sampai baris ke-299 adalah method yang berisi semua proses K-Medoid melakukan perhitungan. Mulai dari pengambilan data medoid awal secara acak, menghitung jarak manhattan, menghitung cost dan total cost pada data dan cluster, mengganti medoid dan akan dilakukan perhitungan yang sama secara berulang-ulang sampai semua data pernah menjadi medoid.

5.2.3 Implementasi Algoritma Silhouette Coefficient

Setelah data berhasil dilakukan proses clusterisasi maka dapat dilanjutkan dengan evaluasi hasil cluster yaitu untuk mengetahui kualitas cluster yang dihasilkan. Ada beberapa tahap yang harus dilakukan dalam perhitungan *Silhouette Coefficient* sampai mendapatkan hasil dari semua *silhouette coefficient* dari setiap cluster yang telah terbentuk. Untuk lebih jelasnya proses perhitungan *Silhouette Coefficient* akan ditampilkan pada Kode Program 5.3.

```
1.      private ListCluster listCluster;
2.      private ListData listData;
3.      private ArrayList<SCData> listSCData = new ArrayList<SCData>();
4.      public SilhouetteCoefficient(ListCluster listCluster, ListData
listData) {
5.          this.listCluster = listCluster;
6.          this.listData = listData;
7.      }
8.      private void setIsChange(int id) {
9.          for (int i = 0; i < this.listCluster.getK(); i++) {
10.             if (i == id) {
11.
this.listCluster.getListCluster().get(i).setIsChange(true);
12.             } else {
13.
this.listCluster.getListCluster().get(i).setIsChange(false);
14.             }
15.         }
16.     }
17.     private int get_i_Cluster(int idData) {
18.         for (int i = 0; i < this.listCluster.getK(); i++) {
19.             for (int j = 0; j <
this.listCluster.getListCluster().size(); j++) {
20.                 Cluster cluster =
this.listCluster.getListCluster().get(j);
21.                 for (int k = 0; k <
cluster.getAnggotaCluster().size(); k++) {
22.                     if
(cluster.getAnggotaCluster().get(k).getIdData() == idData) {
23.                         return j;
24.                     }
25.                 }
26.             }
27.         }
28.         return -1;
29.     }
```

```
30.     private SCData setSCData(int iCluster, int pusatCluster,
        double[] dataPusat, boolean isChange, SCData sCData) {
31.         double costByCluster[] = new
        double[this.listCluster.getK()], a = 0, b = 9999, s = 0;
32.         double costByData[][] = new
        double[this.listData.getJumlahData() - 1][2];
33.         int itr = 0;
34.         for (int j = 0; j < this.listData.getJumlahData(); j++) {
35.             if (pusatCluster != j) {
36.                 int i_cluster =
        this.get_i_Cluster(this.listData.getListData().get(j).getIdData());
37.                 double[] data =
        this.listData.getListData().get(j).getDataDOU();
38.                 for (int k = 0; k < this.listData.getPanjangGen();
        k++) {
39.                     costByData[itr][0] += Math.abs(data[k] -
        dataPusat[k]);
40.                     costByData[itr][1] = i_cluster;
41.                 }
42.                 itr++;
43.             }
44.         }
45.         for (int i = 0; i < this.listCluster.getK(); i++) {
46.             for (int j = 0; j < this.listData.getJumlahData() - 1;
        j++) {
47.                 if (i == (int) costByData[j][1]) {
48.                     costByCluster[i] += costByData[j][0];
49.                 }
50.             }
51.             if (i == iCluster) {
52.                 costByCluster[i] /=
        (this.listCluster.getListCluster().get(i).getAnggotaCluster().size(
        ) - 1);
53.                 if
        (this.listCluster.getListCluster().get(i).getAnggotaCluster().size(
        ) != 1) {
54.                     a = costByCluster[i];
55.                 } else {
56.                     a = 0;
57.                 }
58.             } else {
59.                 costByCluster[i] /=
        this.listCluster.getListCluster().get(i).getAnggotaCluster().size()
        ;
60.                 if (costByCluster[i] < b) {
```

```
61.                         b = costByCluster[i];
62.                     }
63.                 }
64.             }
65.             if (a < b) {
66.                 s = 1 - (a / b);
67.             } else if (a > b) {
68.                 s = (b / a) - 1;
69.             } else {
70.                 s = 0;
71.             }
72.             sCData.setA(a);
73.             sCData.setB(b);
74.             sCData.setS(s);
75.             return sCData;
76.         }
77.         public ListCluster process() {
78.             for (int i = 0; i < this.listCluster.getListCluster().size(); i++) {
79.                 for (int j = 0; j < this.listCluster.getListCluster().get(i).getAnggotaCluster().size(); j++) {
80.                     int pusatCluster = this.listCluster.getListCluster().get(i).getAnggotaCluster().get(j).getIdData();
81.                     double[] dataPusat = this.listData.getListData().get(pusatCluster).getDataDOU();
82.                     this.setIsChange(i);
83.                     SCData sCData = new SCData();
84.                     sCData.setIdCluster(i);
85.                     sCData.setPusatCluster(pusatCluster);
86.                     sCData = this.setSCData(i, pusatCluster, dataPusat, this.listCluster.getListCluster().get(i).isChange(), sCData);
87.                     this.listSCData.add(sCData);
88.                 }
89.             }
90.             double sCluster[] = new double[this.listCluster.getK()], s = 0;
91.             for (int i = 0; i < this.listCluster.getK(); i++) {
92.                 for (int j = 0; j < this.listData.getJumlahData(); j++) {
93.                     if (this.listSCData.get(j).getIdCluster() == i) {
94.                         sCluster[i] += this.listSCData.get(j).getS();
```

```
95.                 }
96.             }
97.             sCluster[i] = sCluster[i] /
this.listCluster.getListCluster().get(i).getAnggotaCluster().size()
;
98.
this.listCluster.getListCluster().get(i).setS(sCluster[i]);
99.         }
100.        for (int i = 0; i < this.listCluster.getK(); i++) {
101.            s += sCluster[i];
102.        }
103.        s = s / this.listCluster.getK();
104.        this.listCluster.setS(s);
105.        return this.listCluster;
106.    }
107. }
```

Kode Program 5.3 Perhitungan Silhouette Coefficient

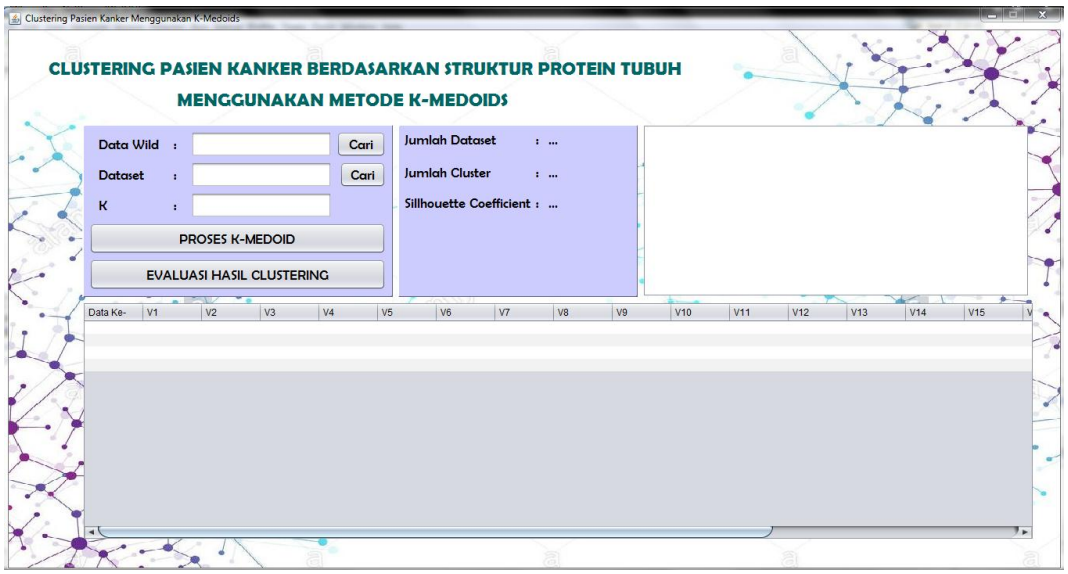
Penjelasan dari Kode Program 5.3 :

1. Baris ke-8 sampai baris ke-16 berfungsi untuk menentukan cluster yang akan dilakukan perhitungan lebih dahulu.
2. Baris ke-17 sampai baris ke-29 berfungsi untuk mencari anggota dari cluster yang telah dipilih pada proses sebelumnya.
3. Baris ke-30 sampai baris ke-76 berfungsi untuk berjalannya proses perhitungan *silhouette coefficient*. Mulai yang pertama dari mendapatkan cluster untuk dipilih datanya, kemudian memilih data untuk dilakukan perhitungan jarak dengan semua data pada clusternya sendiri juga dengan semua cluster yang lain. Lalu jarak semua data pada cluster yang sama dijumlahkan dan menjadi nilai “a”. kemudian rata-rata jarak pada cluster yang lain dipilih yang memiliki nilai paling kecil dan akan menjadi nilai “b”. dari nilai a dan b tersebut maka silhouette coefficient dapat dihitung yang kemudian disimbolkan dengan “s”.
4. Baris ke-77 sampai baris ke-105 berisi semua proses yang telah dideklarasikan sebelumnya sehingga akan dieksekusi pada tahap ini. Juga dilakukan perhitungan rata-rata nilai *silhouette coefficient* dari semua cluster yang telah dihitung pada proses sebelumnya.

5.3 Implementasi Antarmuka

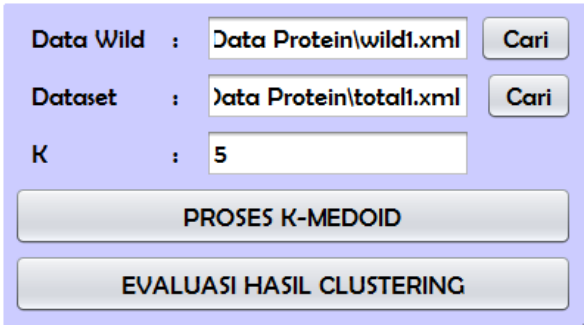
Antarmuka merupakan media penghubung antara sistem dengan pengguna. Dengan adanya antarmuka maka pengguna bisa menggunakan sistem yang telah dibangun dengan mudah, oleh karena itu antarmuka sebaiknya diciptakan sesuai dengan kebutuhan agar mudah dipahami oleh pengguna.

Sistem clustering pasien kanker ini merupakan aplikasi berbasis desktop yang hanya memiliki satu halaman saja. Antarmuka yang telah dibangun ditunjukkan pada Gambar 5.2.



Gambar 5.2 Implementasi Antarmuka Sistem Clustering Pasien Kanker

Dalam implementasi antarmuka sistem clustering pasien kanker ini terdapat empat bagian yang memiliki fungsi masing-masing. Yang pertama adalah bagian input data yang digunakan oleh pengguna untuk memasukkan data-data yang diperlukan dalam proses clustering. Bagian kedua adalah Tabel dataset protein, bagian ketiga adalah menunjukkan jumlah dataset dan cluster serta menampilkan nilai akhir dari *Silhouette Coefficient*, dan bagian keempat adalah sebuah textarea yang berisi hasil clustering yang dilakukan sistem. Bagian-bagian tersebut ditunjukkan pada Gambar 5.3, Gambar 5.4, Gambar 5.5, dan Gambar 5.6.

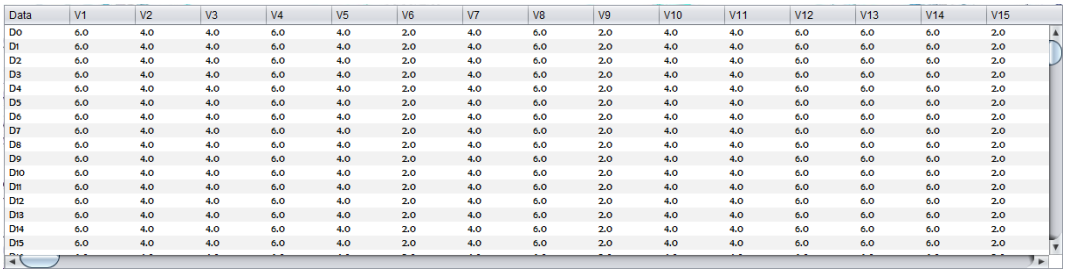


Gambar 5.3 Implementasi Input Data

Pada bagian implementasi input data maka pengguna wajib menginputkan 3 data. Yang pertama adalah data *wild*, data *wild* ini merupakan data acuan yang digunakan sebagai data perbandingan dalam melakukan proses konversi dari data String menjadi data integer dengan bantuan Tabel PAM. Yang kedua pengguna harus menginputkan dataset protein, dataset inilah yang akan dilakukan proses clustering menggunakan metode K-Medoids. Data ketiga yang dimasukkan adalah nilai *k*, nilai *k* ini diisi menggunakan bilangan integer bebas sesuai dengan

keinginan pengguna. Dari nilai *k* inilah banyak cluster yang akan diproses ditentukan. Kedua data *wild* dan dataset ini harus memiliki panjang data yang sama. Jika tidak sama maka data tidak akan bisa diproses.

Setelah pengguna menginputkan data secara benar maka pengguna bisa menekan tombol "Proses K-Medoid" maka sistem akan berjalan melakukan perhitungan clusterisasi dan menampilkan hasilnya. Butuh beberapa detik untuk menampilkan hasil tergantung dari banyak dataset yang diinputkan dan banyak cluster yang diinginkan. Setelah hasil dari proses K-Medoids muncul maka langkah selanjutnya bisa menekan tombol "Evaluasi Hasil Clustering", maka hasil perhitungan menggunakan Silhouette Coefficient akan dimunculkan pada text area.

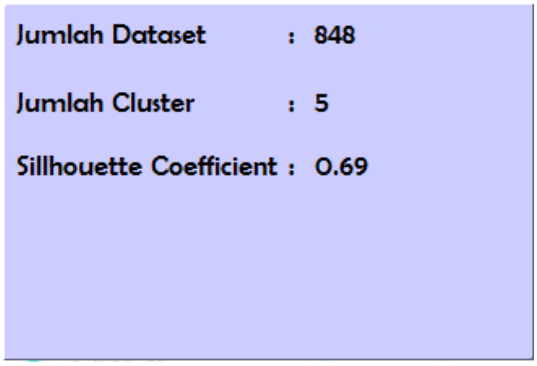


Data	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
D0	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D1	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D2	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D3	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D4	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D5	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D6	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D7	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D8	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D9	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D10	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D11	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D12	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D13	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D14	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0
D15	6.0	4.0	4.0	6.0	4.0	2.0	4.0	6.0	2.0	4.0	4.0	6.0	6.0	6.0	2.0

Gambar 5.4 Implementasi Dataset Protein

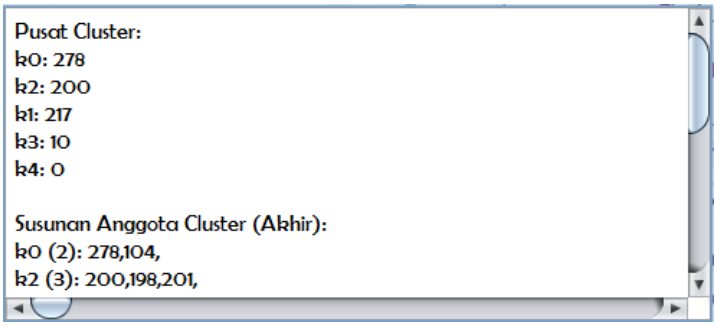
Dataset protein yang digunakan akan ditampilkan dalam bentuk Tabel yang dapat dilihat pada Gambar 5.4. Dataset yang digunakan disini sebanyak 588 data dengan masing-masing data per kelas memiliki 147 data.

Data yang ditampilkan berikutnya adalah informasi jumlah dataset, jumlah cluster, dan hasil akhir evaluasi cluster yang dapat dilihat pada Gambar 5.5.



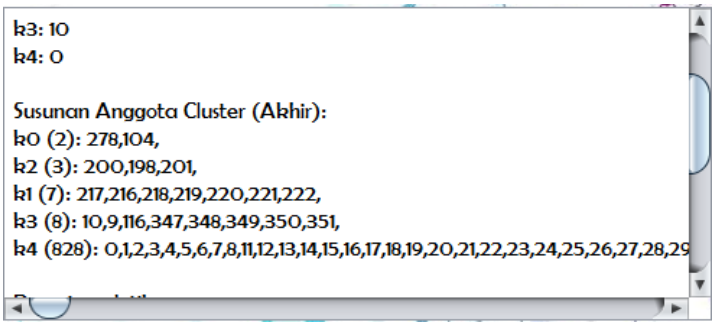
Gambar 5.5 Implementasi Informasi Data

Dan bagian akhir yang ditampilkan oleh sistem adalah sebuah textarea yang berisi informasi detail tentang masing-masing cluster beserta anggota cluster, lama waktu eksekusi metode K-Medoids dan hasil perhitungan Silhouette Coefficient tiap cluster serta rata-ratanya dari seluruh cluster.



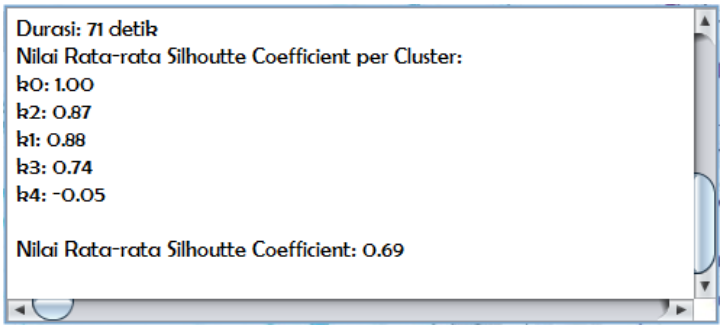
Gambar 5.6 Implementasi Hasil Clustering

Disini ditampilkan bahwa nilai k yang diinputkan adalah 5 sehingga data yang diproses menjadi 5 cluster, dengan medoid tertera pada Gambar 5.6.



Gambar 5.7 Implementasi Anggota Cluster

Disini ditampilkan dari 5 cluster yang telah diproses sebelumnya memiliki anggota data dari dataset yang diinputkan. Terlihat bahwa cluster 0 memiliki anggota 2 data, cluster 1 memiliki anggota 7 data, cluster 2 memiliki anggota 3 data, cluster 3 memiliki anggota 8 data, dan cluster 4 memiliki anggota 828 data.



Gambar 5.8 Implementasi Perhitungan Silhouette Coefficient

Pada Gambar 5.8 dapat dilihat hasil yang diperoleh dari perhitungan silhouette coefficient yaitu evaluasi untuk mengetahui kualitas hasil clustering yang telah dilakukan. Terlihat pada Gambar untuk cluster 0 hasil SC adalah 1.00 yang artinya semua data sudah berada pada cluster yang tepat, cluster 1 hasil SC adalah 0.88 yang artinya 80% data sudah berada cluster yang tepat, cluster 2 hasil SC adalah 0.87 yang artinya 80% data telah berada pada cluster yang tepat, cluster 3 hasil SC adalah 0.74 yang artinya sebagian besar data atau sekitar 70% data telah berada pada cluster yang tepat dan cluster 4 nilai SC adalah -0.05 yang artinya sebagian data belum berada pada cluster yang tepat atau dengan kata lain data seharusnya

berada pada cluster yang lain. Setelah nilai SC pada masing-masing cluster ditemukan maka dilakukan proses perhitungan silhouette coefficient untuk metode K-Medoids pada sistem ini dengan data yang telah diinputkan sebelumnya, dan mendapatkan hasil 0.69 yang artinya bahwa seluruh data cluster sekitar 69% berada pada cluster yang sama dan sisanya memiliki kemungkinan berada pada cluster yang benar atau lebih tepat berada pada cluster yang lain.