BAB 5 IMPLEMENTASI

Pada bab ini dijelaskan bagaimana mengimplementasikan algoritma *Naïve Bayes- Certainty Factor* untuk mendiagnosis penyakit kanker pada sistem reproduksi wanita. Selain itu, diimplementasikan perancangan antarmuka dari sistem yang dibuat.

5.1 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pengembangan aplikasi pada penelitian ini dapat dilihat pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Lunak

Perangkat Lunak	Keterangan
Sistem Operasi	Microsoft Windows 10 Pro
IDE	Android Studio
Bahasa Pemrograman	Java

5.2 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan dalam pengembangan aplikasi pada penelitian ini dapat dilihat pada Tabel 5.2.

Tabel 5.2 Spesifikasi Perangkat Keras

Perangkat Keras	Keterangan
Processor	*AMD A8-6410 APU with AMD Radeon R5*
Memory (RAM)	8.00 GB
Laptop	*Lenovo G40 64-bit*
Smartphone	*Samsung J3 Pro*

5.3 Implementasi Algoritma

Implementasi algoritma merupakan subbab penerapan dari algoritma *Naïve Bayes-Certainty Factor* untuk diagnose penyakit kanker pada sistem reproduksi wanita berbasis android. Implementasi algoritma ini dibangun berdasarkan perancangan yang sudah dirancang pada bab sebelumnya menggunakan bahasa pemrograman java.

5.3.1 Implementasi Perhitungan Jumlah Kemunculan Penyakit

Proses ini merupakan perhitungan berapa banyak muncul masing-masing penyakit pada data latih. Perhitungan ini dilakukan secara berurutan mulai dari penyakit pertama yaitu kanker ovarium sampai dengan penyakit ke lima yaitu miom. Jumlah masing-masing kemunculan penyakit akan disimpan kedalam variabel yang bernama counterPenyakit[] yang nantinya akan digunakan dalam proses selanjutnya. Untuk lebih jelasnya dapat dilihat pada *Source Code* 5.1.

```
01  checkbox[0] = (CheckBox) this.findViewById(R.id.gejala1);
02  checkbox[1] = (CheckBox) this.findViewById(R.id.gejala2);
03  checkbox[2] = (CheckBox) this.findViewById(R.id.gejala3);
04  checkbox[3] = (CheckBox) this.findViewById(R.id.gejala4);
05  checkbox[4] = (CheckBox) this.findViewById(R.id.gejala5);
06  checkbox[5] = (CheckBox) this.findViewById(R.id.gejala6);
07  checkbox[6] = (CheckBox) this.findViewById(R.id.gejala7);
08  checkbox[7] = (CheckBox) this.findViewById(R.id.gejala8);
09  checkbox[8] = (CheckBox) this.findViewById(R.id.gejala9);
10  checkbox[9] = (CheckBox) this.findViewById(R.id.gejala10);
11  checkbox[10] = (CheckBox) this.findViewById(R.id.gejala11);
12  checkbox[11] = (CheckBox) this.findViewById(R.id.gejala12);
13  checkbox[12] = (CheckBox) this.findViewById(R.id.gejala13);
14  checkbox[13] = (CheckBox) this.findViewById(R.id.gejala14);
15  checkbox[14] = (CheckBox) this.findViewById(R.id.gejala15);
16  checkbox[15] = (CheckBox) this.findViewById(R.id.gejala16);
17  checkbox[16] = (CheckBox) this.findViewById(R.id.gejala17);
18  checkbox[17] = (CheckBox) this.findViewById(R.id.gejala18);
19
20  btnProses = (Button) this.findViewById(R.id.buttonProses);
21  btnProses.setOnClickListener(this);
22  double counterP1 = 0;
23  double counterP2 = 0;
24  double counterP3 = 0;
25  double counterP4 = 0;
26  double counterP5 = 0;
27  for (int j = 0; j < 25; j++) {
28      if (data_latih[j][18].equalsIgnoreCase("Kanker
29  Ovarium")) {
30          counterP1++;
31          counterPenyakit[0] = counterP1;
32      } else if (data_latih[j][18].equalsIgnoreCase("Kanker
33  Endometrium")) {
34          counterP2++;
35          counterPenyakit[1] = counterP2;
36      } else if (data_latih[j][18].equalsIgnoreCase("Kanker
37  Serviks")) {
38          counterP3++;
39          counterPenyakit[2] = counterP3;
40      } else if (data_latih[j][18].equalsIgnoreCase("Kista"))
41  {
42          counterP4++;
43          counterPenyakit[3] = counterP4;
44      } else if (data_latih[j][18].equalsIgnoreCase("Miom"))
45  {
46          counterP5++;
47          counterPenyakit[4] = counterP5;
48      }
49  }
```

***Source Code* 5.1 Implementasi Perhitungan Jumlah Kemunculan Penyakit**

Penjelasan kode program proses perhitungan kemunculan penyakit sebagai berikut:

Baris 1-16 : Merupakan inisialisasi awal untuk *CheckBox* , yang berfungsi agar pengguna dapat memilih gejala yang diderita.

Baris 18-19 : Merupakan inisialisasi *Button* , yang berfungsi untuk ketika pengguna sudah memasukkan gejala yang diderita maka akan masuk ke proses selanjutnya.

Baris 20-23 : Merupakan inisialisasi varaiabel *Counter* untuk menyimpan hasil berapa jumlah kemunculan penyakit pada data latih.

Baris 24-45 : Merupakan proses menghitung kemunculan masing-masing penyakit pada data latih.

5.3.2 Implementasi Perhitungan Jumlah Kemunculan Gejala

Proses ini merupakan proses perhitungan berapa banyak jumlah gejala masukan pengguna pada masing masing penyakit yang tersimpan dalam data latih. Proses perhitungan pertama sistem akan mengecek gejala apa saja yang dimasukkan oleh pengguna, setelah itu akan masuk ke proses perhitungan gejala masukan pada masing-masing penyakit. Jumlah kemunculan gejala pada masing-masing penyakit akan disimpan pada array **counterGejalaMasuk[][],** selanjutnya akan masuk ke proses selanjutnya. Untuk lebih jelasnya dapat dilihat pada *Source Code* 5.2.

```
01  ArrayList<String> inputanGejala = new ArrayList<>();
02          if (button == R.id.buttonProses) {
03              double countergejalaP1;
04              double countergejalaP2;
05              double countergejalaP3;
06              double countergejalaP4;
07              double countergejalaP5;
08              double ceklist;
09              checkedCounter = 0;
10              for (int i = 0; i < checkbox.length; i++) {
11                  if (checkbox[i].isChecked()) {
12                      checkedCounter++;
13                  }
14              }
15              Log.i("Jumlah centang", " " +
    String.valueOf(checkedCounter));
16              counterGejalaMasuk = new
    double[counterPenyakit.length][checkedCounter];
17              int k = 0;
18
19              if (checkedCounter > 0) {
20                  for (int i = 0; i < checkbox.length; i++) {
21
22                      if (checkbox[i].isChecked()) {
23                          countergejalaP1 = 0;
24                          countergejalaP2 = 0;
25                          countergejalaP3 = 0;
26                          countergejalaP4 = 0;
27                          countergejalaP5 = 0;
28                          ceklist = 0;

                            for (int j = 0; j < 25; j++) {
```

```
                                if
(data_latih[j][18].equalsIgnoreCase("Kanker Ovarium")) {
                                    if
(data_latih[j][i].equalsIgnoreCase("1")) {
                                        countergejalaP1++;
                                    }
                                } else if
(data_latih[j][18].equalsIgnoreCase("Kanker Endometrium"))
{
                                    if
(data_latih[j][i].equalsIgnoreCase("1")) {
                                        countergejalaP2++;
                                    }
                                } else if
(data_latih[j][18].equalsIgnoreCase("Kanker Serviks")) {
                                    if
(data_latih[j][i].equalsIgnoreCase("1")) {
                                        countergejalaP3++;
                                    }
                                } else if
(data_latih[j][18].equalsIgnoreCase("Kista")) {
                                    if
(data_latih[j][i].equalsIgnoreCase("1")) {
                                        countergejalaP4++;
                                    }
                                } else if
(data_latih[j][18].equalsIgnoreCase("Miom")) {
                                    if
(data_latih[j][i].equalsIgnoreCase("1")) {
                                        countergejalaP5++;
                                    }
                                }
                            }

                            for (int c = 0; c <
checkbox.length; c++) {
                                if (checkbox[c].isChecked()) {
//                                  for (int a = 0; a < 2; a++) {
                                    Log.i("Gejala ", " " +
String.valueOf(dataGejala[c][1]));

inputanGejala.add(dataGejala[c][1]);
//                                  }
                                    ckList[c] = 1;
                                    Log.i("Ceklist ", "" +
String.valueOf(c));
                                } else {
                                    ckList[c] = 0;
                                }
                            }

                            counterGejalaMasuk[0][k] =
countergejalaP1;
                            counterGejalaMasuk[1][k] =
countergejalaP2;
                            counterGejalaMasuk[2][k] =
countergejalaP3;
                            counterGejalaMasuk[3][k] =
countergejalaP4;
                            counterGejalaMasuk[4][k] =
countergejalaP5;
                            k++;

```

***Source Code* 5.2 Implementasi Perhitungan Jumlah Kemunculan Gejala**

Penjelasan kode program proses perhitungan kemunculan gejala sebagai berikut:

Baris 1-2 : Merupakan inisialisasi awal array *inputanGejala* dan inisialisasi *button,* ketika menekan tombol tersebut maka akan di proses ke langkah selanjutnya.

Baris 3-7 : Merupakan inisialisasi variabel *counterGejala.*

Baris 9-13 : Merupakan proses perhitungan banyaknya gejala yang dimasukkan oleh pengguna.

Baris 20-86 : Merupakan proses menghitung berapa banyak masing-masing gejala masukan pengguna yang muncul pada setiap penyakit.

5.3.3 Implementasi Perhitungan Nilai *Prior*

Proses ini merupakan proses perhitungan nilai *prior.* Pada proses ini merupakan perhitungan jumlah kemunculan gejala masukan pada masing-masing penyakit yang teradpat pada data latih. Proses perhitungan nilai *prior* secara manual dapat dilihat pada bab sebelumnya. Nilai *prior* didapat dari hasil pembagian jumlah kemunculan gejala pada masing-masing penyakit yang terdapat pada data latih dengan keseluruhan data. Untuk lebih jelasnya dapat dilihat pada *Source Code* 5.3.

```
01  public void hitungPrior() {
02      probabilitasPenyakit = new
03  double[counterPenyakit.length];
04      for (int i = 0; i < probabilitasPenyakit.length; i++) {
05          probabilitasPenyakit[i] = counterPenyakit[i] /
06  data_latih.length;
07      }
    }
```

Source Code* 5.3 Implementasi Perhitungan Nilai *Prior

Penjelasan kode program proses perhitungan nilai *prior* sebagai berikut:

Baris 3-7 : Merupakan perhitungan nilai *prior* masing-masing penyakit yang muncul pada data latih.

5.3.4 Implementasi Perhitungan Nilai *Likelihood*

Proses ini merupakan proses perhitungan nilai probabilitas *likelihood.* Proses ini menghitung berapa banyak jumlah setiap gejala yang dimasukkan pada masing-masing data latih. Kemudian, jumlah masing-masing gejala masukan dilakukan perhitungan dengan membagi jumlah masing-masing gejala masukan dengan jumlah kemunculan masing-masing penyakit pada data latih. Proses perhitungan manual dapat dilihat pada bab sebelumnya. Untuk lebih jelasnya dapat dilihat pada *Source Code* 5.4.

```
01  public void hitungLikelihood() {
02          nilaiLikelihood = new
03  double[counterPenyakit.length][checkedCounter];
04          nilaiTotalLikelihood = new
05  double[counterPenyakit.length];
06          double likelihoodtemp = 1;
07          for (int i = 0; i < counterPenyakit.length; i++) {
08              for (int k = 0; k < checkedCounter; k++) {
09                  nilaiLikelihood[i][k] =
10  counterGejalaMasuk[i][k] / counterPenyakit[i];
11                  Log.i("Nilai Likelihood " + (i + 1) + "." +
12  (k + 1), " " + String.valueOf(nilaiLikelihood[i][k]));
13  //                nilaiTotalLikelihood[k] *=
14  nilaiLikelihood[k][i];
15              }
16          }
17          for (int j = 0; j < counterPenyakit.length; j++) {
18              for (int i = 0; i < checkedCounter; i++) {
19                  likelihoodtemp *= nilaiLikelihood[j][i];
20              }
21              nilaiTotalLikelihood[j] = likelihoodtemp;
22              likelihoodtemp = 1;
23              Log.i("Total Likelihood pada P" + (j + 1), " "
24  + String.valueOf(nilaiTotalLikelihood[j]));
          }
      }
```

Source Code* 5.4 Implementasi Perhitungan Nilai *Likelihood

Penjelasan kode program proses perhitungan nilai *likelihood* sebagai berikut:

Baris 6-15 : Merupakan perhitungan nilai *likelihood* yang berdasarkan masing-masing gejala masukan pengguna pada setiap penyakit.

Baris 16-24 : Merupakan perhitungan nilai total dari *likelihood* gejala masukan pengguna pada setiap penyakit.

5.3.5 Implementasi Perhitungan Nilai *Posterior*

Proses ini merupakan proses perhitungan nilai *posterior.* Proses ini didapatkan dari mengkalikan nilai *prior* dengan nilai *likelihood.* Setelah niali *posterior* selesai dihitung maka salah satu penyakit akan memiliki nilai probabilitas tertinggi, sehingga penyakit tersebut dipilih sebagai proses diagnosis. Proses perhitungan manual dapat dilihat pada bab sebelumnya. Untuk lebih jelasnya dapat dilihat pada *Source Code* 5.5.

```
01  public void hitungPosterior() {
02          nilaiPosterior = new double[5];
03          for (int i = 0; i < nilaiPosterior.length; i++) {
04              nilaiPosterior[i] = probabilitasPenyakit[i] *
05  nilaiTotalLikelihood[i];
06              Log.i("Nilai Posterior P" + (i + 1), " " +
07  String.valueOf(nilaiPosterior[i]));
08  //              Log.i("Total Likelihood ", " " +
09  String.valueOf(nilaiTotalLikelihood[i]));
10
11          }
      }
```

```
12      // memilih nilai max dari nilai posterior
13      public int max(double[] x) {
14          int panjang = 5;
15          int index[] = {0, 1, 2, 3, 4};
16          double temp[] = new double[5];
17          temp = x;
18          double nilaiTertinggi = 0;
19          int indexPenyakit = 0;
20          int swap;
21          System.out.println("input masuk Posterior:");
22
23          for (int i = 0; i < panjang; i++) {
24              System.out.println(temp[i]);
25          }
26          for (int i = 0; i < panjang; i++) {
27              System.out.println(index[i]);
28          }
29          for (int a = 0; a < panjang; a++) {
30              for (int b = 0; b < panjang - 1; b++) {
31                  if (temp[b] < temp[b + 1]) {
32                      double tampung = temp[b];
33                      swap = index[b];
34                      index[b] = index[b + 1];
35                      index[b + 1] = swap;
36                      temp[b] = temp[b + 1];
37                      temp[b + 1] = tampung;
38                  }
39              }
40          }
41
42          System.out.println("Output masuk MAX Posterior:");
43
44          for (int i = 0; i < panjang; i++) {
45              System.out.println(temp[i]);
46          }
47          for (int i = 0; i < panjang; i++) {
48              System.out.println(index[i]);
49          }
50          nilaiTertinggi = temp[0];
51          indexPenyakit = index[0];
52          return indexPenyakit;
53      }
```

Source Code* 5.5 Implementasi Perhitungan Nilai *Posterior

Penjelasan kode program proses perhitungan nilai *posterior* sebagai berikut:

Baris 3-11 : Merupakan perhitungan nilai *posterior* masing-masing penyakit.

Baris 14-50 : Merupakan perhitungan maksimum nilai *posterior* masing-masing penyakit dan nilai *posterior* penyakit tertinggi akan digunakan sebagai hasil diagnosis.

5.3.6 Implementasi Perhitungan Nilai *Certainty Factor*

Proses ini merupakan proses perhitungan nilai *Certainty Factor* hasil diagnosis metode *Naïve Bayes.* Setelah proses *Naïve Bayes* selesai dilakukan makan akan dihitung nilai keyakinannya. Gejala akan memiliki nilai bobot pakar. Untuk mendapatkan nilai *Certainty Factor* akan dilakukan pengkalian nilai bobot

pakar dengan nilai bobot gejala pengguna. Jika pengguna memilih gejala maka gejala tersebut akan bernilai bobot 1. Untuk perhitungan manual dapat dilihat pada bab sebelumnya. Untuk lebih jelasnya dapat dilihat pada *Source Code* 5.6.

```
01  public double CF(int x) {
02
03          double cfCombine = 0;
04          double temp = 0;
05          System.out.println("x : " + x);
06          for (int i = 0; i < 18; i++) {
07              cf[i] = bobotGejala[x][i] * ckList[i];
08              System.out.println("Bobot Pakar " + (i + 1) +
    ":" + bobotGejala[x][i]);
09
10          }
11          temp = cf[0] + (cf[1] * (1 - cf[0]));
12          for (int i = 2; i < 18; i++) {
13              temp = cf[i] + (temp * (1 - cf[i]));
14          }
15          Log.i("Nilai CF ", " " + temp);
16          return temp;
17      }
    }
```

Source Code* 5.5 Implementasi Perhitungan Nilai *Certainty Factor

Penjelasan kode program proses perhitungan nilai *Certainty Factor* sebagai berikut:

Baris 6-10 : Merupakan perhitungan nilai *certainty factor* hasil diagnosis.

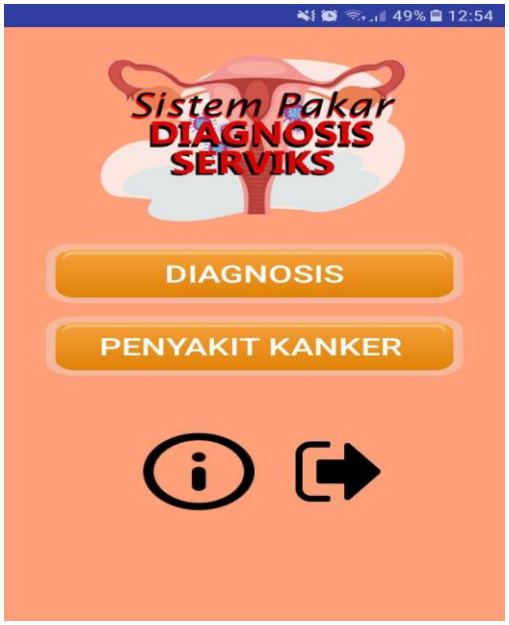
Baris 11-17 : Merupakan perhitungan nilai *certainty factor combine* hasil diagnosis.

5.4 Implementasi Antarmuka Pengguna

Pada sub-bab ini akan menjelaskan tentang implementasi antarmuka pengguna sistem pakar diagnose penyakit kanker pada sistem reproduksi wanita.

5.4.1 Implementasi Antarmuka Pengguna Halaman Awal

Gambar 5.1 menunjukkan hasil implementasi dari perancangan antarmuka halaman awal yang ditunjukkan pada Gambar 4.7.



Gambar 5.1 Implementasi Antarmuka Pengguna Halaman Awal

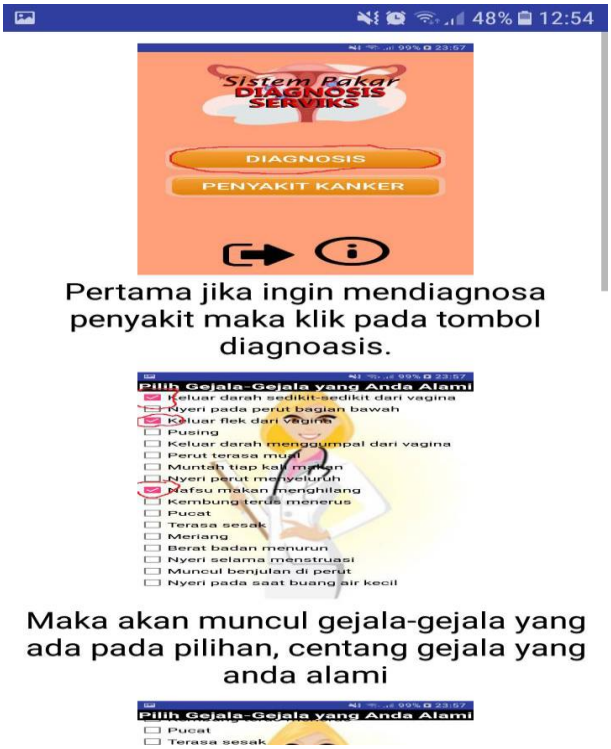
5.4.2 Implementasi Antarmuka Pengguna Informasi Penyakit Kanker

Gambar 5.2 menunjukkan hasil implementasi dari perancangan antarmuka Informasi penyakit kanker yang ditunjukkan pada Gambar 4.8.

Gambar 5.2 Implementasi Antarmuka Pengguna Infomasi Penyakit Kanker

5.4.3 Implementasi Antarmuka Pengguna Halaman Petunjuk

Gambar 5.3 menunjukkan hasil implementasi dari perancangan antarmuka Halaman petunjuk yang ditunjukkan pada Gambar 4.9.



Gambar 5.3 Implementasi Antarmuka Pengguna Halaman Petunjuk

5.4.4 Implementasi Antarmuka Pengguna Halaman Diagnosis

Gambar 5.4 menunjukkan hasil implementasi dari perancangan antarmuka Halaman Diagnosis yang ditunjukkan pada Gambar 4.10.

Gambar 5.4 Implementasi Antarmuka Pengguna Halaman Diagnosis

5.4.5 Implementasi Antarmuka Pengguna Halaman Hasil Diagnosis

Gambar 5.5 menunjukkan hasil implementasi dari perancangan antarmuka Halaman Hasil Diagnosis yang ditunjukkan pada Gambar 4.11.

Gambar 5.5 Implementasi Antarmuka Pengguna Halaman Hasil Diagnosis