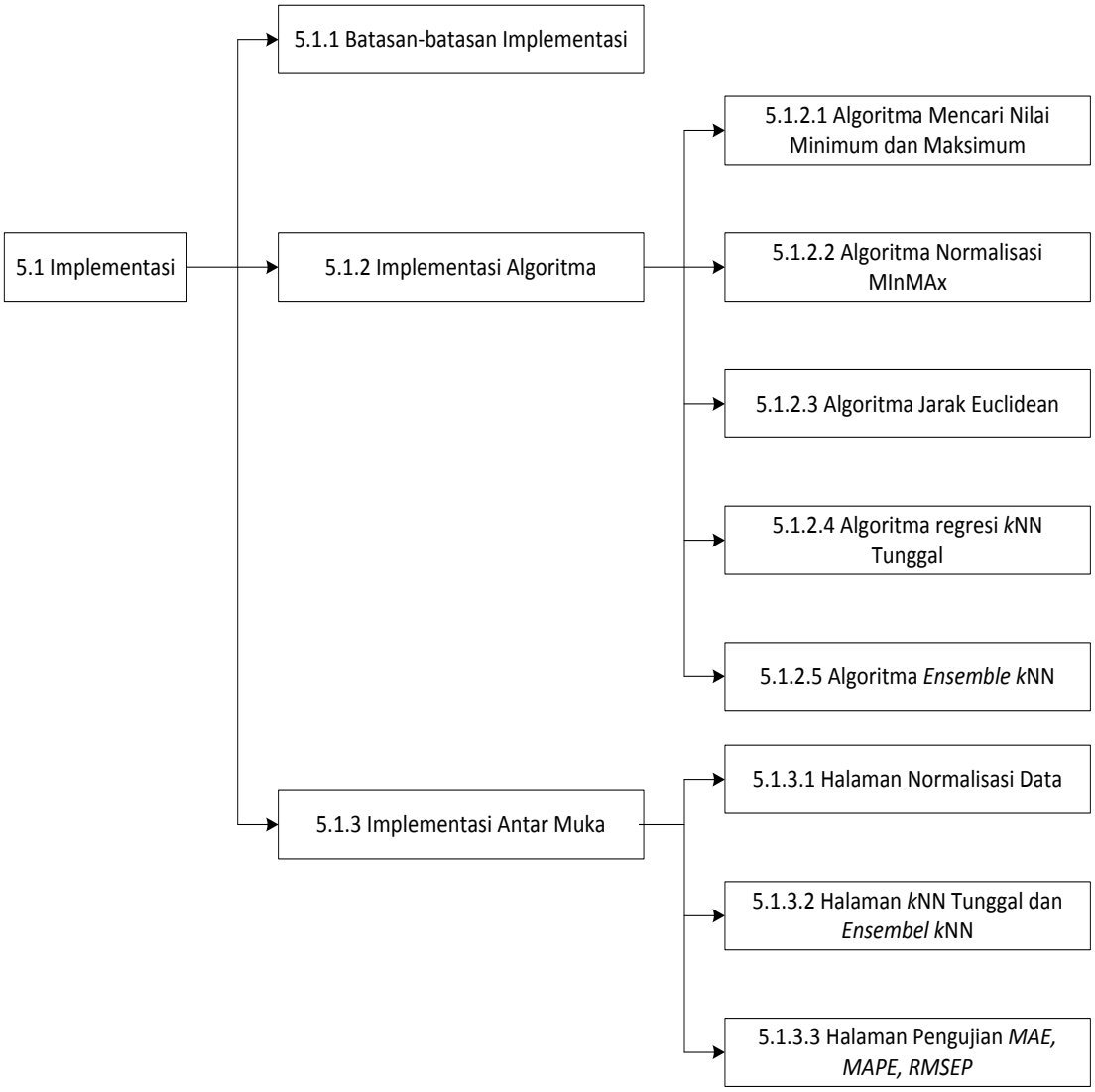
BAB 5 IMPLEMENTASI

1.1. IMPLEMENTASI

Bab ini membahas mengenai implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari proses perancangan perangkat lunak yang dibuat. Pembahasan terdiri dari penjelasan tentang batasan-batasan dalam implementasi, implementasi algoritma, dan implementasi antar muka. Tahapan pembahasan implementasi yang dikerjakan digambarkan pada Gambar 5.1



Gambar 5.1 Digram Blok Implementasi

1.1.1. Batasan-Batasan Implementasi

Batasan-batasan dalam mengimplementasikan sistem adalah sebagai berikut:

1. Sistem yang dibuat menerima masukan berupa data excel dengan ekstensi .xls .
2. Pembahasan ditekankan pada bagaimana mengimplementasikan metode *Ensemble* *k*NN serta mengetahui tingkat kinerja metode tersebut apabila diterapkan pada data *time series*.
3. Maksimal jumlah K yang diinputkan yaitu 8.

1.1.2. Implementasi Algoritma

Aplikasi Implementasi Metode *Ensemble* K-Nearest Neighbor untuk Prediksi Nilai Tukar Rupiah terhadap Dollar Amerika mempunyai beberapa proses utama yang terbagi dalam beberapa fungsi. Pada penulisan skripsi ini hanya dicantumkan algoritma dari beberapa proses saja sehingga tidak semua algoritma akan dicantumkan. Algoritma proses yang akan dicantumkan antara lain adalah proses mencari nilai minimum dan maksimum dari parameter, Normalisasi *MinMax*, mencari jarak *Euclidean*, proses *k*NN, dan proses *Ensemble* *k*NN. Algoritma ini akan direpresentasikan dalam bentuk *code* dengan bahasa pemrograman C#.

1.1.2.1. Implementasi Algoritma Mencari Nilai Min dan Maks

Operasi pada algoritma mencari nilai minimum dan maksimum pada tiap parameter ini bertujuan untuk mengetahui batas bawah dan atas dari data training yang nantinya akan dikonversi menjadi nilai range.

```
1  public void findMaxMin(List<data_mentah> alldataMentah, out double
2  min_ekspor, out double max_ekspor, out double min_import, out double
3  max_import, out double min_biRate, out double max_biRate, out double
4  min_inflasi, out double max_inflasi, out double min_hutang, out double
5  max_hutang)
6      {
7          min_ekspor = alldataMentah[0].ekspor;
8          max_ekspor = alldataMentah[0].ekspor;
9          min_import = alldataMentah[0].import;
10         max_import = alldataMentah[0].import;
11         min_biRate = alldataMentah[0].bi_rate;
12         max_biRate = alldataMentah[0].bi_rate;
13         min_inflasi = alldataMentah[0].inflasi;
14         max_inflasi = alldataMentah[0].inflasi;
15         min_hutang = alldataMentah[0].hutang;
16         max_hutang = alldataMentah[0].hutang;
17         foreach (var item in alldataMentah)
18         {
19             if (item.ekspor < min_ekspor)
20             {
```

```
21                  min_ekspor = item.ekspor;
22              }
23              if (item.ekspor> max_ekspor)
24              {
25                  max_ekspor = item.ekspor;
26              }
27              if (item.import < min_import)
28              {
29                  min_import = item.import;
30              }
31              if (item.import > max_import)
32              {
33                  max_import = item.import;
34              }
35              if (item.bi_rate < min_biRate)
36              {
37                  min_biRate = item.bi_rate;
38              }
39              if (item.bi_rate > max_biRate)
40              {
41                  max_biRate = item.bi_rate;
42              }
43              if (item.inflasi < min_inflasi)
44              {
45                  min_inflasi = item.inflasi;
46              }
47              if (item.inflasi > max_inflasi)
48              {
49                  max_inflasi = item.inflasi;
50              }
51              if (item.hutang < min_hutang)
52              {
53                  min_hutang = item.hutang;
54              }
55              if (item.hutang > max_hutang)
56              {
57                  max_hutang = item.hutang;
58              }
59          }
60      }
```

Gambar 5.2 Implementasi Algoritma Mencari Nilai Min dan Maks

Secara umum, proses pencarian nilai minimum dan maksimum dari tiap parameter yang akan dinormalisasi menggunakan fungsi seleksi IF. Baris 1-5 merupakan deklarasi fungsi `findMaxMin` dengan parameter input adalah semua data *training*. Pada penelitian ini, parameter yang akan dinormalisasi

adalah *Bi-rate*, inflasi, ekspor, *import* dan hutang. Sehingga fungsi `findMaxMin` memiliki parameter ouput berupa nilai maksimum dan minimum dari kelima parameter tersebut. Baris 7-16 merupakan proses inisialisasi awal dari nilai masing-masing parameter output. Baris 17 merupakan iterasi untuk menampilkan satu per satu nilai data *training*. Pada baris 19-57, proses seleksi untuk mencari nilai maksimum yaitu dengan seleksi jika nilai dari variabel maksimum sekarang lebih kecil dibandingkan dengan data perbandingan, maka data perbandingan tersebut akan menjadi data maksimum yang baru. Begitupun pada proses seleksi untuk mencari nilai minimum yaitu jika nilai dari variable minimum sekarang lebih besar dari data yang perbandingan, maka data perbandingan tersebut akan mnejadi data minimum yang baru.

1.1.2.2. Implementasi Algoritma Normalisasi MinMax

Nilai minimum dan maksimum dari kelima parameter yaitu *Bi-rate*, inflasi, ekspor, *import* dan hutang telah didapatkan pada proses sebelumnya. Nilai minimum dan maksimum digunakan sebagai batasan atas dan bawah dari range nilai hasil normalisasi MinMax. Tujuan dari normalisasi ini adalah menskalakan nilai awal dari data ke dalam range tertentu, dimana nilai atas dan bawah dapat ditentukan sesuai dengan kebutuhan range data. Pada penelitian ini, nilai batasan atas yang digunakan adalah 1.0 sedangkan nilai batasan bawah adalah 0.1 untuk mencegah menghilangkan informasi dari nilai maksimum jika diubah menjadi 0.

```
1   class minMaxNormalisasi
2     {
3       public List<data_mentah> normalisasi(List<data_mentah>
4   allDataMentah, List<data_mentah> dataMentah, data_mentah dataTesting,
5   out data_mentah dataTestingNormalisasi)
6       {
7           List<data_mentah> dataNormalisasi = new List<data_mentah>();
8           dataTestingNormalisasi = new data_mentah();
9           dataTestingNormalisasi = dataTesting;
10          dataNormalisasi = dataMentah;
11          double min_ekspor, max_ekspor, min_import, max_import,
12  min_biRate, max_biRate, min_inflasi, max_inflasi;
13          double min_hutang, max_hutang;
14          findMaxMin(allDataMentah, out min_ekspor, out max_ekspor, out
15  min_import, out max_import, out min_biRate,
16              out max_biRate, out min_inflasi, out max_inflasi, out min_hutang,
17  out max_hutang);
18          int x=0;
19          foreach (var item in dataMentah)
20          {
21              double itemNormalisasiEkspor = ((item.ekspor - min_ekspor) * (1 -
22  0.1) / (max_ekspor - min_ekspor)) + 0.1;
23              itemNormalisasiEkspor = Math.Round(itemNormalisasiEkspor, 2);
```

```
24              dataNormalisasi[x].ekspor = itemNormalisasiEkspor;
25
26              double itemNormalisasiImport = ((item.import - min_import) * (1 -
27  0.1) / (max_import - min_import)) + 0.1;
28              itemNormalisasiImport = Math.Round(itemNormalisasiImport,2);
29              dataNormalisasi[x].import = itemNormalisasiImport;
30
31              double itemNormalisasiBiRate = ((item.bi_rate - min_biRate) * (1 -
32  0.1) / (max_biRate - min_biRate)) + 0.1;
33              itemNormalisasiBiRate = Math.Round(itemNormalisasiBiRate, 3);
34              dataNormalisasi[x].bi_rate = itemNormalisasiBiRate;
35
36              double itemNormalisasiInflasi = ((item.inflasi - min_inflasi) * (1 -
37  0.1) / (max_inflasi - min_inflasi)) + 0.1;
38              itemNormalisasiInflasi = Math.Round(itemNormalisasiInflasi, 3);
39              dataNormalisasi[x].inflasi = itemNormalisasiInflasi;
40
41              double itemNormalisasiHutang = ((item.hutang - min_hutang) * (1 -
42  0.1) / (max_hutang - min_hutang)) + 0.1;
43              itemNormalisasiHutang = Math.Round(itemNormalisasiHutang, 3);
44              dataNormalisasi[x].hutang = itemNormalisasiHutang;
45              x++;
46          }
47          double itemDataTestingEkspor = ((dataTesting.ekspor - min_ekspor) *
48  (1 - 0.1) / (max_ekspor - min_ekspor)) + 0.1;
49          dataTestingNormalisasi.ekspor = itemDataTestingEkspor;
50          double itemDataTestingImport = ((dataTesting.import - min_import)
51  * (1 - 0.1) / (max_import - min_import)) + 0.1;
52          dataTestingNormalisasi.import = itemDataTestingImport;
53          double itemDataTestingBiRate = ((dataTesting.bi_rate - min_biRate) *
54  (1 - 0.1) / (max_biRate - min_biRate)) + 0.1;
55          dataTestingNormalisasi.bi_rate = itemDataTestingBiRate;
56          double itemDataTestingInflasi = ((dataTesting.inflasi - min_inflasi) * (1
57  - 0.1) / (max_inflasi - min_inflasi)) + 0.1;
58          dataTestingNormalisasi.inflasi = itemDataTestingInflasi;
59          double itemDataTestingHutang = ((dataTesting.hutang - min_hutang)
60  * (1 - 0.1) / (max_hutang - min_hutang)) + 0.1;
61          dataTestingNormalisasi.hutang = itemDataTestingHutang;
62          return dataNormalisasi;
63      }
64  }
```

Gambar 5.3 Implementasi Algoritma Normalisasi MinMax

Gambar 5.3 merupakan class untuk mengimplementasikan algoritma normalisasi MinMax. Baris 3-5 merupakan deklarasi fungsi normalisasi dengan tipe List<data_mentah>. Object data_mentah terdiri dari dua variable yaitu

kurs_jual dan kurs_beli. Parameter input dari fungsi ini adalah semua data *training*, *default* data *training* yang akan digunakan dalam proses perhitungan prediksi nanti dan data uji. Sedangkan parameter output dari fungsi ini adalah data *training* dan uji hasil normalisasi. Baris 7-13 merupakan deklarasi dan inisialisasi variabel dan objek yang dibutuhkan. Baris 14-17 merupakan baris untuk memanggil fungsi menghitung nilai maksimum dan minimum dari tiap parameter. Baris 19 merupakan iterasi tiap nilai data *training*. Baris 21-62 adalah proses perhitungan normalisasi min max dengan menggunakan persamaan (2-1) yang telah dijelaskan pada Bab 2. Fungsi normalisasi ini mengembalikan nilai berupa data hasil normalisasi.

1.1.2.3. Implementasi Algoritma Mencari Jarak Euclidean

Setelah data training dinormalisasi, maka proses selanjutnya adalah menghitung jarak antara data training dengan data uji. Perhitungan jarak yang digunakan pada penelitian ini adalah Jarak *Euclidien*. Dengan menghitung jarak antara data training dengan data uji, maka akan diketahui mana data training yang memiliki nilai kedekatan dengan data uji. Nilai kedekatan ini linier dengan besar jarak. Semakin kecil jarak antar data, maka kedekatannya semakin dekat. Sebaliknya semakin besar jarak antar data, maka kedekatannya semakin jauh. Implementasi algoritma mencari jarak Euclidien dapat dilihat pada Gambar 5.4.

```
1  class euclidienDistance
2    {
3      public List<data_distance> calculateDistance(data_mentah dataTesting,
4      List<data_mentah> dataTraining)
5      {
6        double distance;
7        int x = 1;
8        List<data_distance> dataJarak = new List<data_distance>();
9        foreach (var item in dataTraining)
10       {
11         data_distance jarak = new data_distance();
12         jarak.index = x;
13         jarak.distance = Math.Sqrt(Math.Pow((item.bi_rate -
14          dataTesting.bi_rate), 2) + Math.Pow((item.inflasi -
15          dataTesting.inflasi), 2) + Math.Pow((item.ekspor -
16          dataTesting.ekspor), 2) + Math.Pow((item.import -
17          dataTesting.import), 2));
18         jarak.distance = Math.Round(jarak.distance, 2);
19         dataJarak.Add(jarak);
20         x++;
21       }
22       List<data_distance> dataJarakSorting =
23        dataJarak.OrderBy(dataDistance => dataDistance.distance).ToList();
24       return dataJarakSorting;
25     }
```

```
26      }
```

Gambar 5.4 Implementasi Algoritma Mencari Jarak *Euclidean*

Baris 3-4 merupakan deklarasi fungsi untuk melakukan perhitungan jarak euclidien yang memiliki parameter input berupa data uji dan data *training*. Baris 6-8 merupakan deklarasi dan inisialisasi nilai-nilai variabel yang dibutuhkan. Baris 9 adalah proses iterasi dari nilai data *training*. Baris 11-20 merupakan proses perhitungan jarak euclidien dengan menggunakan persamaan (2-4) yang telah dijelaskan pada Bab 2. Pada Baris 22-23 dilakukan proses *sorting* terhadap data training sesuai dengan besarnya jarak dengan data uji. Proses *sorting* ini dilakukan secara *ascending* yakni dari jarak terkecil ke jarak terbesar. Data jarak setelah diurutkan inilah yang menjadi data keluaran dari fungsi ini.

1.1.2.4. Implementasi Algoritma Menghitung KNN

Metode KNN melakukan proses klasifikasi yang berbasis dari k-tetangga terdekatnya. Proses perhitungan jarak dan pengurutan dari data jarak terkecil ke terbesar sudah dilakukan pada proses sebelumnya. Sehingga proses penentuan k terdekat tergantung dari besarnya nilai k yang ditentukan. Pada penelitian ini, jumlah banyaknya k yang digunakan adalah 8 buah dimana masing-masing bernilai 3,4,6,9,10,12,15 dan 24. Gambar 5.5 menjelaskan mengenai implementasi algoritma menghitung KNN.

```
1   private void button2_Click(object sender, EventArgs e)
2       {
3           List<int> kCollection = new List<int>();
4           List<data_distance> sortedDistance = new List<data_distance>();
5           List<data_KNN_tunggal> prediksiKNNTunggal = new
6            List<data_KNN_tunggal>();
7           double w=0, sumW, sumWYBeli, sumWYJual, prediksiJual,
8            prediksiBeli, D=0, Dpart;
9           for (int x = 0; x < n_KNN; x++)
10          {
11              string nama = "k" + x.ToString();
12
13               kCollection.Add(int.Parse(((TextBox)tabPage2.Controls[nama]).Te
14               xt));
15              euclidienDistance calculateDistance = new euclidienDistance();
16              sortedDistance =
17               calculateDistance.calculateDistance(dataTestingNormalisasi,
18               dataNormalisasi);
19              Dpart = 0;
20              sumWYBeli = 0;
21              sumWYJual = 0;
22              sumW = 0;
23              for (int y = 0; y < kCollection[x]; y++)
24              {
```

```
25              //calculate weight
26              w = (25 - (double)sortedDistance[y].index)/24;
27              Dpart += (25 - (double)sortedDistance[y].index);
28              sumW += w;
29              sumWYBeli += (w * dataNormalisasi[sortedDistance[y].index-
30               1].kurs_beli);
31              sumWYJual += (w * dataNormalisasi[sortedDistance[y].index -
32               1].kurs_jual);
33            }
34            D = Dpart / kCollection[x];
35            //MessageBox.Show(D.ToString());
36            prediksiBeli = (sumWYBeli / sumW) + (bBeli * D);
37            prediksiJual = (sumWYJual / sumW) + (bJual * D);
38
39            Object[] row = new Object[] { (x+1), prediksiJual, prediksiBeli };
40            dataGridView3.Rows.Add(row);
41            data_KNN_tunggal dataKnn = new data_KNN_tunggal();
42            dataKnn.kursBeli = prediksiBeli;
43            dataKnn.kursJual = prediksiJual;
44            prediksiKNNTunggal.Add(dataKnn);
45          }
46          ensembleKNN ensemble = new ensembleKNN();
47          data_KNN_tunggal result = new data_KNN_tunggal();
48          result = ensemble.calculateEnsembleKNN(prediksiKNNTunggal);
49          Object[] rownya = new Object[] { ("Ensemble"), result.kursJual,
50           result.kursBeli };
            dataGridView3.Rows.Add(rownya);
        }
```

Gambar 5.5 Implementasi Algoritma Menghitung KNN

Baris 1 merupakan deklarasi dari fungsi dimana fungsi ini akan dijalankan saat Tombol2 ditekan. Baris 3-8 merupakan proses deklarasi dan inisialisasi dari variable dan objek yang diperlukan dalam proses perhitungan. Baris 9 merupakan iterasi untuk melakukan proses per nilai k yang telah ditentukan. Baris 12-16 merupakan pemanggilan fungsi untuk menghitung jarak *Euclidien* sekaligus melakukan pengurutan data. Baris 24 merupakan baris untuk menghitung nilai bobot dari tiap data *training*. Baris 32 untuk melakukan perhitungan nilai D. Baris 34 merupakan perhitungan prediksi untuk kurs Beli dan baris 35 perhitungan prediksi untuk kurs Jual. Perhitungan prediksi kurs ini disesuaikan dengan persamaan (2-7)pada Bab 2. Baris 37-38 adalah proses menampilkan data hasil prediksi dari tiap KNN ke dataGridView. Baris 39-42 adalah proses menampung nilai hasil tiap KNN ke dalam suatu objek. Baris 44-46 adalah proses pemanggilan fungsi untuk menghitung prediksi dari *Ensemble KNN*. Dimana fungsi nya akan dijelaskan lebih rinci pada sub proses berikutnya. Baris 47-49 adalah proses menampilkan data prediksi hasil *Ensemble KNN* ke dataGridView.

1.1.2.5. Implementasi Algoritma Menghitung Ensemble KNN

Pada metode *Ensemble KNN*, hasil prediksi dari KNN akan menjadi data *training*. Berdasarkan banyaknya *K* pada KNN tersebut (pada penelitian terdapat 8), akan dihitung dan menghasilkan satu nilai akhir prediksi yang akan menjadi keluaran dari program ini. Yaitu prediksi nilai kurs penukaran nilai dolar ke rupiah. Penjelasan detail mengenai implementasi algoritma menghitung *Ensemble KNN* dijelaskan pada Gambar 5.6.

```
1   class ensembleKNN
2     {
3       public                                          data_KNN_tunggal
4   calculateEnsembleKNN(List<data_KNN_tunggal>
5       dataKnn)
6       {
7         double              sumPrediksiBeli=0,          sumPrediksiJual=0,
8          sumXYJual=0,sumXYBeli=0, sumWBeli=0, sumWJual=0;
9
10        foreach (var item in dataKnn)
11        {
12          sumPrediksiBeli += item.kursBeli;
13          sumPrediksiJual += item.kursJual;
14        }
15        foreach (var item in dataKnn)
16        {
17          double wBeli = item.kursBeli/sumPrediksiBeli;
18          sumXYBeli += item.kursBeli * wBeli;
19          sumWBeli += wBeli;
20
21          double wJual = item.kursJual / sumPrediksiJual;
22          sumXYJual += item.kursJual* wJual;
23          sumWJual += wJual;
24        }
25        data_KNN_tunggal hasil = new data_KNN_tunggal();
26        hasil.kursJual = sumXYJual / sumWJual;
27        hasil.kursBeli = sumXYBeli / sumWBeli;
28
29        return hasil;
30      }
    }
```

Gambar 5.6 Implementasi Algoritma Menghitung *Ensemble k*NN

Gambar 5.6 merupakan kelas dari implementasi algoritma menghitung *Ensemble KNN*. Bari 3-5 merupakan deklarasi fungsi dengan parameter input adalah nilai List objek dari hasil prediksi KNN yang telah didapatkan pada proses sebelumnya. Baris 6-7 merupakan proses deklarasi dan inisialisasi dari variabel yang dibutuhkan. Baris 9-13 merupakan proses perhitungan

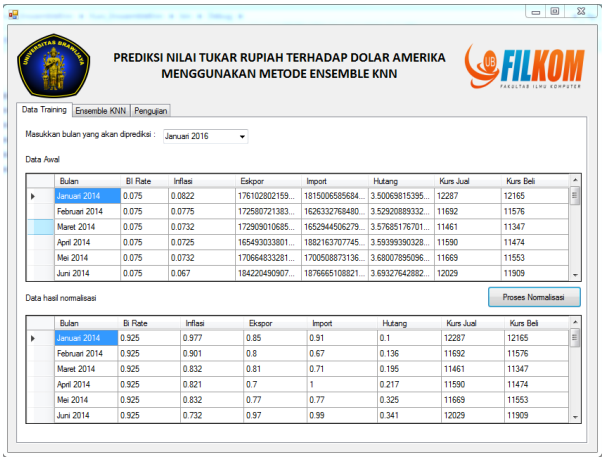
`sumPrediksiBeli` dan `sumPrediksiJual` dari data KNN. Baris 16 dan 20 untuk menghitung bobot masing-masing data prediksi KNN. Baris 24-26 merupakan proses perhitungan *Ensemble KNN* dengan menggunakan persamaan (2-9).... Pada Bab 2. Baris 28 merupakan pengembalian nilai prediksi hasil Ensemble KNN. Nilai kembalian inilah yang akan menjadi nilai keluaran dari sistem Implementasi Metode *Ensemble* K-Nearest Neighbor untuk Prediksi Nilai Tukar Rupiah terhadap Dollar Amerika ini.

1.1.3. Implementasi Antar Muka Aplikasi

Antarmuka Aplikasi Sistem Prediksi Nilai Tukar Rupiah terhadap Dollar Amerika menggunakan Metode *Ensemble* *k*NN digunkan oleh pengguna untuk dapat berinteraksi dengan sistem perangkat lunak. Antarmuka perangkat lunak ini dibagi menjadi 3 bagian yaitu antarmuka halaman normalisasi data, antarmuka *Ensemble kNN*, dan antarmuka pengujian.

1.1.3.1. Halaman Normalisasi Data

Halaman normalisasi data berfungsi untuk mengimpor data berupa *file .xls* yang nantinya akan dilakukan normalisasi data pada tiap parameter dan dimana hasil normalisasi data akan ditampilkan. Gambar 5.7 menunjukkan implementasi tampilan antarmuka dari halaman normalisasi data yang mengacu pada perancangan antar muka halaman normalisasi data.



Gambar 5.7 Implementasi Halaman Normalisasi Data

1.1.3.2. Halaman *Ensemble* *k*NN

Halaman *Ensemble* *k*NN berfungsi untuk memasukkan banyaknya variabel *K* yang diinginkan serta memasukkan nilai variabel *K* ke-n. Pada halaman ini juga menampilkan hasil dari metode *k*NN dan *Ensemble* *k*NN. Gambar 5.8 menunjukkan implementasi tampilan antarmuka dari halaman *Ensemble* *k*NN yang mengacu pada perancangan antar muka halaman *Ensemble* *k*NN.

Gambar 5.8 Implementasi Halaman *Ensemble k*NN

1.1.3.3. Halaman Pengujian

Halaman pengujian berfungsi untuk menampilkan nilai MAE, MAPE, RMSEP berdasarkan data *testing*, data *training*, dan variabel *K*. Gambar 5.9 menunjukkan implementasi tampilan antarmuka dari halaman pengujian yang mengacu pada perancangan antar muka halaman pengujian.

Gambar 5.9 Implementasi Halaman Pengujian