DIAGNOSA PENYAKIT PARKINSON MENGGUNAKAN METODE EDITED FUZZY K-NEAREST NEIGHBOR BASED ON THRESHOLD VALUE

SKRIPSI

LABORATORIUM KOMPUTASI CERDAS

Untuk Memenuhi Sebagaian Persyaratan Mencapai Gelar Sarjana Komputer

Disusun Oleh :

AGA SAPUTRA

105060800111037

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2015

LEMBAR PERSETUJUAN

DIAGNOSA PENYAKIT PARKINSON MENGGUNAKAN METODE EDITED FUZZY K-NEAREST NEIGHBOR BASED ON THRESHOLD VALUE

SKRIPSI

LABORATORIUM KOMPUTASI CERDAS

Untuk Memenuhi Sebagaian Persyaratan Mencapai Gelar Sarjana Komputer

Disusun Oleh :

AGA SAPUTRA

105060800111037

Skripsi ini telah disetujui oleh dosen pembimbing

Dosen Pembimbing I,	Dosen Pembimbing II,
M. Tanzil Furqon, S.Kom, M.Comp.Sc.	**Budi Darma S. , S.Kom, M.Cs**
NIP. 19820930 200801 1 004	**NIP.19841015 201404 1 002**

LEMBAR PENGESAHAN

DIAGNOSA PENYAKIT PARKINSON MENGGUNAKAN METODE EDITED FUZZY K-NEAREST NEIGHBOR BASED ON THRESHOLD VALUE

SKRIPSI

LABORATORIUM KOMPUTASI CERDAS

Diajukan untuk memenuhi persyaratan memperoleh gelar sarjana komputer

Disusun Oleh :

Aga Saputra

NIM : 105060800111037

Setelah dipertahankan di depan Majelis Penguji pada tanggal 1 Oktober 2015 dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana dalam bidang Ilmu Komputer

Penguji I	Penguji II
Nurul Hidayat, S.Pd., MSc	**Indriati, ST., M.Kom**
NIP. 19680430 200212 1 001	**NIK. 200909 831013 2 001**

Mengetahui

Ketua Program Studi Informatika/Ilmu Komputer

Drs. Marji, M.T.

NIP. 19670801 199203 1 001

LEMBAR PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan pasal 70).

Malang, 13 Oktober 2015

Mahasiswa,

Aga Saputra

NIM.105060800111037

KATA PENGANTAR

Puji syukur kehadirat Allah SWT, yang telah memberikan rahmat serta karunia-Nya sehingga penyusunan skripsi ini dapat diselesaikan dengan baik. Shalawat serta salam semoga selalu tercurahkan kepada suri tauladan kita Nabi Muhammad SAW.

Dalam penyusunan skripsi ini, telah banyak bimbingan dan bantuan yang didapatkan baik dari segi moral maupun segi material dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Bapak Ir. Sutrisno, M.T., Bapak Ir. Heru Nurwarsito, M. Kom, Bapak Himawat Aryadita, S.T., M. Sc., dan Bapak Eddy Santoso, S. Kom selaku Ketua, Wakil Ketua 1, Wakil Ketua 2, dan Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Malang.
2. Bapak Drs. Marji, MT. dan Issa Arwani, ST., MT., selaku Ketua dan Sekretaris Program Studi Informatika serta segenap Bapak/Ibu Dosen, Staff Administrasi dan Perpustakaan Program Studi Teknik Informatika Universitas Brawijaya Malang.
3. Muhammad Tanzil Furqon, S.Kom, M.Comp.Sc. selaku dosen pembimbing I yang telah bijaksana dan sabar dalam membimbing dan menyalurkan ilmu kepada penulis serta semua waktu dan nasehat yang telah diberikan selama proses pengerjaan skripsi ini.
4. Budi Darma S. , S.Kom, M.Cs selaku dosen pembimbing II yang telah bijaksana dan sabar dalam membimbing dan menyalurkan ilmu kepada penulis serta semua waktu dan nasehat yang telah diberikan selama proses pengerjaan skripsi ini.
5. Kedua Orang Tua Tercinta Ayah Imam Soepangkat Soerodjo dan Ibu Yenny Setyoningsih atas doa, motivasi, nasihat dan dukungannya yang amat sangat luar biasa dalam penyelesaian skripsi ini. Terima kasih semuanya terutama kasih sayang dan cinta yang terus menerus tercurahkan.
6. Teman-teman tersayang Syafrita Aprilianti, Hardika Wijaya, Aditya Shelly, Taufiqqilahi N.Y, Adi Cahya Hermawan, Daniel Alex Saroha Simamora, Roisul Mustain Fauzi, Yudha Eka Permana, Gian Rofi, Arbi Wicaksono,

Niko, Dedi, Erik, Jembos, Jokom, Lukman, Mamot, Ricky Bagus, Adi Bagus, Vendi Dwi Cahyono, Yusuf Nur Cahyo, Alfian, Ghozainul Alfa, Fatikatur, Tamy, Rosy yang selalu memberikan semangat, dorongan, dan bantuan pikiran.

7. Teman-teman angkatan 2010 yang terlalu banyak bila disebutkan, terima kasih atas semuanya.
8. Seluruh staf akademik dan karyawan Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya yang telah selalu bersama dalam perjalanan mencari ilmu.

Semoga segala pertolongan dan kebaikan semuanya mendapatkan berkah dan balasan dari Allah SWT. Penulis menyadari bahwa skripsi ini masih jauh dari sempurna, oleh karena itu kritik dan saran yang sifatnya membangun sangat diperlukan untuk memperbaiki mutu penulisan selanjutnya. Akhir kata, penulis berharap semoga skripsi ini dapat bermanfaat bagi semua pihak.

Malang, 13 Oktober 2015

Penulis

ABSTRAK

Aga Saputra. 2015. Diagnosa Penyakit Parkinson Menggunakan Metode Edited Fuzzy K-Nearest Neighbor Based On Threshold Value. Dosen Pembimbing: Muhammad Tanzil Furqon, S.Kom, M.Comp.Sc. Dan Budi Darma S. , S.Kom, M.Cs

Penelitian ini bertujuan untuk menerapkan metode Edited Fuzzy k-Nearest Neighbor (EFk-NN) yang dapat diterapkan pada dataset Parkinsons Dataset. Algoritma EFk-NN adalah algoritma yang nilai keanggotaannya digunakan untuk memberikan nilai pada setiap sampel di masing-masing kelas yang menunjukkan sejauh mana elemen milik himpunan fuzzy. Beberapa sampel mungkin dalam batas beberapa kelas , dan yang nilai keanggotaan maksimum mungkin menjadi kecil maka nilai keanggotaan yang kecil akan dihapus dari set pengujian. Hasil dari pengujian ini untuk mengetahui pengaruh data latih, data uji, nilai k, sebaran kelas, nilai threshold, serta mengetahui tingkat akurasi dari sistem ini. Tingkat akurasi pada penelitian ini adalah 90%. Pada penelitian Fk-NN sebelumnya, didapat hasil akurasi sebesar 76%. Dapat disimpulkan bahwa metode Edited Fuzzy k-Nearest Neighbor (EFk-NN) memiliki kinerja yang baik dalam diagnosa penderita parkinson.

Kata kunci : parkinson, EFk-NN, akurasi

ABSTRACT

Aga Saputra. 2015. Diagnosis of Parkinson's Disease Using Edited Fuzzy K - Nearest Neighbor Method Based On Threshold Value. Adviser: Muhammad Tanzil Furqon, S.Kom, M.Comp.Sc. and Budi Darma S. , S.Kom, M.Cs

This study aims to apply the Edited Fuzzy k - Nearest Neighbor (EFk - NN) method which can be applied to Parkinson dataset Dataset . EFk - NN algorithm is an algorithm whose membership values are used to assign values to each sample in each class that shows the extent to which the element belongs to fuzzy set . Some samples may be within a few classes , and the maximum membership value may be a little smaller then the value of membership will be removed from the test set . Results from this test to determine the effect of training data , test data , the value of k , grade distribution , the threshold value , and determine the accuracy of these systems . Accuracy rate in this study was 90 % . Fk - NN on previous research , the result accuracy by 76 % . It can be concluded that the method Edited Fuzzy k - Nearest Neighbor (EFk - NN) has a good performance in the diagnosis of Parkinson's patients

Keywords : parkinson , EFk - NN , accuracy

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR *SOURCE CODE*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Parkinson adalah salah satu jenis penyakit yang disebabkan oleh adanya gangguan pada organ otak, terutama pada bagian sistem syaraf pusat otak manusia yang mengalami kemunduran. Penyakit ini menyerang syaraf – syaraf seseorang di mana daya kontrol otot seseorang itu merosot dengan parah sehingga penderitanya tidak dapat mengurus diri sendiri.

Penyakit Parkinson pertama kali diuraikan dalam sebuah monograf oleh James Parkinson seorang dokter di London, Inggris, pada tahun 1817. Di dalam tulisannya, James Parkinson mengatakan bahwa penyakit (yang akhirnya dinamakan sesuai dengan namanya) tersebut memiliki karakteristik yang khas yakni *tremor* (gemetar), kekakuan, dan gangguan dalam cara berjalan. Parkinson menyerang sekitar 1 di antara 250 orang yang berusia di atas 40 tahun dan sekitar 1 dari 100 orang yang berusia di atas 65 tahun.

Dengan berkembangnya teknologi pada saat ini, dapat mendiagnosa seseorang yang menderita penyakit parkinson dengan kecerdasan komputasional. Kecerdasan Komputasional adalah sebuah pendekatan untuk membangun sistem komputasi cerdas yang bertujuan membantu menyelesaikan permasalahan manusia. Kecerdasan Komputasional terdiri dari tiga paradigma komputasi, yaitu: *Neural Network System*, *Fuzzy Logic*, *Probabilistic Reasoning*.

Salah satu metode komputasi cerdas yang sering dipakai saat ini adalah logika *fuzzy*. Logika *fuzzy* dapat dipakai untuk memodelkan proses berpikir manusia yang menyangkut unsur ketidakpastian, keraguan dan linguistik dimana metode klasik yang biasanya digunakan seperti metode matematis sulit untuk diterapkan karena kurang cukupnya pengetahuan [PRI-05].

Pada penelitian ini menggunakan metode *fuzzy* dengan metode data mining untuk memprediksi data. Pada data mining, terdapat beberapa metode untuk klasifikasi/prediksi yaitu metode k-Nearest Neighbor, ID3, C45, Bayesian dan beberapa metode lainnya. [KUS-09].

Metode *k-Nearest Neighbor* (k-NN) merupakan metode klasifikasi data yang cara kerjanya lebih sederhana bila dibandingkan dengan metode klasifikasi pada data mining lainnya. Dimana metode ini melakukan klasifikasi kepada data baru yang masih belum diketahui masuk kedalam kelas mana, dengan menggunakan beberapa data dengan sejumlah *k* yang letaknya terdekat dengan data baru tersebut [NIL-96].

Metode k-NN dapat digabungkan dengan beberapa metode lain, salah satu variasi dari penggunaan metode k-NN ini adalah *Fuzzy k-Nearest Neighbor* (Fk-NN) dimana metode ini menggabungkan teknik *fuzzy* dengan teknik data mining [ZHA-09]. Fk-NN sebelumnya telah digunakan untuk memprediksi *IRIS, Vertebratal Column, Liver, Diabetes* [KEL-85] dan *Web Classification Document* [ZHA-09]. Dimana, dalam implementasinya algoritma Fk-NN memberikan nilai keanggotaan kelas pada *vektor* sampel dan bukan menempatkan *vektor* pada kelas tertentu. Keuntungannya adalah nilai-nilai keanggotaan *vektor* memberikan tingkat jaminan pada hasil klasifikasi. Dasar dari algoritma Fk-NN adalah untuk menetapkan nilai keanggotaan sebagai fungsi jarak *vektor* dari k-NN dan keanggotaan tetangga di kelas-kelas yang memungkinkan.

Ada beberapa metode *modified* Fk-NN, salah satunya adalah *Edited* Fk-NN. Dalam EFKNN, nilai keanggotaan digunakan untuk memberikan nilai pada setiap sampel di masing-masing kelas, yang menunjukkan sejauh mana elemen milik himpunan *fuzzy*. Berikutnya, memperkirakan sampel kategori sesuai dengan nilai keanggotaan maksimum. Namun, dalam proses tersebut, beberapa sampel mungkin dalam batas beberapa kelas, dan yang nilai keanggotaan maksimum mungkin menjadi kecil. Sampel seperti ini biasanya tidak dapat diandalkan atau tidak berguna yang harus dihapus dari set pengujian.

Oleh karena itu, penelitian ini diberi judul *"**Diagnosa Penyakit Parkinson Menggunakan Metode Edited Fuzzy K-Nearest Neighbor Based On Threshold Value**"*. Data yang digunakan untuk klasifikasi pada *Parkinson Dataset* yang berupa MDVP:Fo(Hz), MDVP:Fhi(Hz), MDVP:Flo(Hz), MDVP:Jitter(%), status, PPE, spread1, spread2.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini ada 2 yaitu :

1. Bagaimana menerapkan metode EFk-NN untuk mendiagnosis penyakit parkinson.
2. Bagaimana tingkat akurasi hasil klasifikasi *Parkinson Dataset* menggunakan metode EFk-NN yang dipengaruhi sejumlah nilai *Threshold*, sebaran data dan perbandingan dengan Fk-NN.

1.3 Batasan Masalah

Dari permasalahan pada rumusan masalah, dapat diambil suatu batasan masalah sebagai berikut:

1. Data yang digunakan pada penelitian ini adalah *Parkinson Dataset* yang diperoleh dari url : *https://archive.ics.uci.edu/ml/datasets/Parkinsons*.
2. Parameter penyakit parkinson yang digunakan berjumlah 8 data yaitu MDVP:Fo(Hz), MDVP:Fhi(Hz), MDVP:Flo(Hz), MDVP:Jitter(%), status, PPE, spread1, spread2.

1.4 Tujuan

Tujuan yang ingin dicapai dalam pembuatan skripsi ini adalah:

1. Dapat mendiagnosa penyakit parkinson menggunakan metode *Edited Fuzzy K-Nearest Neighbor* (EFk-NN).
2. Dapat memberikan tingkat akurasi hasil diagnosa yang dipengaruhi sejumlah nilai *Threshold*, sebaran data dan perbandingan dengan Fk-NN dengan menggunakan metode *Edited Fuzzy K-Nearest Neighbor* (EFk-NN).

1.5 Manfaat

Manfaat yang dapat diambil dari skripsi ini yaitu dapat mendiagnosa penyakit parkinson dari pasien berdasarkan dataset, dengan menggunakan EFk-NN.

1.6 Sistematika Penulisan

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB 1 PENDAHULUAN

Bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, serta sistematika penulisan skripsi.

BAB 2 TINJAUAN PUSTAKA

Bab ini berisi tentang dasar teori dan referensi yang mendasari pembuatan klasifikasi data pada penyakit *parkinson* yaitu kajian pustaka, parkinson, klasifikasi, dan EFk-NN.

BAB 3 METODOLOGI DAN PERANCANGAN

Bab ini berisi tentang metode – metode yang digunakan untuk memprediksi tingkat resiko menggunakan klasifikasi EFk-NN, pada studi kasus diagnosis penyakit parkinson.

BAB 4 IMPLEMENTASI

Membahas implementasi dari klasifikasi EFk-NN penyakit parkinson yang digunakan dalam penelitian ini.

BAB 5 HASIL DAN PEMBAHASAN

Bab ini berisi pembahasan dari implementasi klasifikasi EFk-NN pada penyakit parkinson yang sesuai dengan sistem dan hasil pengujian yang dilakukan.

BAB 6 PENUTUP

Bab ini berisi kesimpulan yang diperoleh dari hasil pembuatan dan pengujian yang dikembangkan dalam skripsi ini serta saran untuk pengembangan lebih lanjut.

BAB II
KAJIAN PUSTAKA

Penelitian ini bertujuan untuk menerapkan metode *Edited Fuzzy k-Nearest Neighbor* (EFk-NN) yang dapat diterapkan pada dataset *Parkinsons Dataset*. Algoritma EFk-NN adalah algoritma yang nilai keanggotaannya digunakan untuk memberikan nilai pada setiap sampel di masing-masing kelas yang menunjukkan sejauh mana elemen milik himpunan *fuzzy*. Beberapa sampel mungkin dalam batas beberapa kelas, dan yang nilai keanggotaan maksimum mungkin menjadi kecil maka nilai keanggotaan yang kecil akan dihapus dari set pengujian. Hasil dari pengujian ini untuk mengetahui pengaruh data latih, data uji, nilai k, sebaran kelas, nilai threshold, serta mengetahui tingkat akurasi dari sistem ini dengan perbandingan metode Fk-NN. Tingkat akurasi pada penelitian ini adalah 90%. Pada penelitian Fk-NN sebelumnya, didapat hasil akurasi sebesar 76%. Dapat disimpulkan bahwa metode *Edited Fuzzy k-Nearest Neighbor* (EFk-NN) memiliki kinerja yang baik dalam diagnosa penderita parkinson[WIJ-14].

2.1 Parkinson

Penyakit parkinson merupakan penyakit neurodegeneratif. Penyakit neurodegeneratif adalah suatu penyakit yang dapat menurunkan fungsi jaringan atau organ tubuh seseorang secara berkelanjutan sejalan dengan berlalunya waktu. Pada penyakit ini terjadi kehilangan neuron dopamin di substansia nigra (sel yang terletak di otak tengah). Penyakit ini ditemukan pada semua etnis dengan distribusi kelamin yang sama dan terutama pada lansia. Pertambahan usia merupakan faktor resiko penyakit ini [DEW-07].

Penyakit parkinson dapat dideteksi dengan gejala – gejala yang timbul pada diri penderita. Gejala – gejala penyakit parkinson dibagi menjadi 3 bagian yaitu [DEW-07] :

1. Gejala Motorik.
    - Tremor (patognomonik, menonjol saat istirahat, asimetris, gerakan volunter berkurang).
    - Rigiditas.

- Bradikinesia (asimetris, kekuatan normal, gerakan tangkas melambar).
- Postur tubuh dan gaya jalan (menyeret kaki, langkah pendek, gerakan tangan menurun, postur tubuh membungkuk).

2. Gejala Nonmotorik.
    - Gangguan tidur (insomnia, parasomnia, *rapid eye movement* (REM), *sleep behavior disorder* (RBD), gerakkan ekstremitas secara periodik saat tidur, *sleep apmea*, dan *vivid dreaming*.
    - Halusinasi.
    - *Restless Legs Syndrome*.
    - Konstipasi
    - Inkontinensia urin.
    - *Drooling*.
    - Disfungsi seksual.
3. Gejala Psikiatrik.
    - Depresi
    - Demensia.
    - Psikosis.

Penyakit parkinson sangat menyiksa penderitanya karena menurunkan fungsi organ tubuh sehingga penderita penyakit tersebut kesulitan untuk bergerak dan melakukan aktifitas. Jika terserang penyakit parkinson, maka obat – obatan mulai diberikan saat pasien merasa terganggu dengan gejala yang ada atau pasien ingin diberikan terapi setelah mendapat penjelasan tentang penyakit parkinson. Terapi diberikan sesuai gejala yang timbul, dengan memperhatikan efektivitas optimal dan efek samping minimal sesuai dengan harapan pasien / keluarga pasien.

Penyakit parkinson bukan penyakit yang fatal, tetapi berkembang secara progresif sesuai dengan waktu serta tidak dapat diprediksi. Dengan terapi, pasien dapat cukup lama hidup produktif setelah didiagnosis. Angka harapan hidup penderita penyakit parkinson umumnya lebih rendah dibandingkan dengan orang sehat. Pada tahap akhir, penyakit parkinson menyebabkan komplikasi seperti tersedak, pneumonia, dan terjatuh yang dapat meyebabkan kematian.

2.2 *DataSet*

Dataset adalah objek yang merepresentasikan data dan relasi didalam memori. Objek *dataset* adalah representasi dari suatu kelompok data yang terdiri dari tabel – tabel, kolom, *relationship*, baris dan berbagai *constraints*, dimana proses interaksi antara objek *dataset* dan sumber data (*Database*) dilakukan melaui *Data Provider* [SAN-04].

Oleh karena bentuk representasi objek *dataset* menyerupai suatu *database* maka memungkinkan *programmer* untuk bekerja pada objek *dataset* tanpa harus dipusingkan oleh bentuk/representasi sumber data sebenarnya.

Dataset terdiri dari beberapa kelas yang berisi atribut yang dapat diolah menjadi hasil output. Atribut merupakan karakteristik dari *Entity* atau *relationship*, yang menyediakan penjelasan detail tentang *entity* atau *relationship* tersebut. Atau merupakan nama-nama *property* dari sebuah kelas yang menjelaskan batasan nilainya dari *property* yang dimiliki oleh sebuah kelas tersebut. Atribut dari sebuah kelas mempresentasikan property-property yang dimiliki oleh kelas tersebut.

Penelitian ini menggunakan *Parkinson DataSet* dan mengambil 8 atribut untuk mengolah data menjadi hasil output yaitu :

- *MDVP:Fo(Hz)* : Rata-rata frekuensi dasar vokal pasien.
- *MDVP:Fhi(Hz)* : Frekuensi dasar vokal maksimum pasien.
- *MDVP:Flo(Hz)* : Frekuensi dasar vokal minimum pasien.
- *MDVP:Jitter(%)* : Salah satu variasi ukuran pada frekuensi dasar.
- *Status* : Status pasien yang dibedakan menjadi Sehat atau Parkinson.
- *PPE, spread1, spread2,* : Tiga ukuran nonlinear dari dasar frekuensi.

Frekuensi dasar adalah tingkat getaran pita suara dan dinyatakan dalam hertz (Hz) atau siklus per detik. Frekuensi dasar merupakan ukuran audioperceptual dengan melihat korelasi nada. Frekuensi dasar dapat diukur dengan vokal yang berkelanjutan atau diekstrak dari kemampuan berbicara. Variasi didalam frekuensi dasar sebelum mengetahui cara berbicara dapat digunakan untuk menggambarkan

dari kemampuan berbicara menjadi sebuah intonasi. Frekuensi dasar juga dapat menghitung tinggi rendahnya nada pasien yang dihasilkan [JOHN-07].

2.3 Logika *Fuzzy*

Logika *fuzzy* adalah suatu cara yang tepat untuk memetakan ruang input kedalam suatu ruang output [KUS-03]. Konsep ini diperkenalkan dan dipublikasikan pertama kali oleh Lotfi A. Zadeh, seorang profesor dari University of California di Berkeley pada tahun 1965. Logika *fuzzy* menggunakan ungkapan bahasa untuk menggambarkan nilai variabel. Logika *fuzzy* bekerja dengan menggunakan derajat keanggotaan dari nilai yang kemudian digunakan untuk menentukan hasil yang ingin dihasilkan berdasarkan atas spesifikasi yang telah ditentukan. Sebelumnya telah disebutkan bahwa logika *fuzzy* memetakan ruang *input* ke ruang *output*. Antara *input* dan *output* terdapat kotak hitam yang harus memetakan *input* ke *output* yang sesuai. Alasan mengapa orang menggunakan logika *fuzzy*, yaitu [KUS-10]:

1. Konsep logika *fuzzy* mudah dimengerti karena menggunakan konsep matematis yang sederhana dan mudah dimengerti.
2. Logika *fuzzy* sangat fleksibel.
3. Logika *fuzzy* memiliki toleransi terhadap data yang tidak tepat.
4. Logika *fuzzy* mampu memodelkan fungsi-fungsi nonlinier yang sangat kompleks.
5. Logika *fuzzy* dapat membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan.
6. Logika *fuzzy* dapat bekerjasama dengan teknik-teknik kendali secara konvensional.
7. Logika *fuzzy* didasarkan pada bahasa alami.

2.4 Himpunan *Fuzzy*

Himpunan tegas (*crisp*) A didefinisikan oleh item-item yang ada pada himpunan itu. Pada himpunan tegas (*crisp*), nilai keanggotaan suatu item x dalam suatu himpunan A (μA(x)) memiliki dua kemungkinan [KUS-10], yaitu:

1. Satu (1), yang berarti bahwa suatu item menjadi anggota dalam suatu himpunan.
2. Nol (0), yang berarti bahwa suatu item tidak menjadi anggota dalam suatu himpunan.

Himpunan *Fuzzy* didasarkan pada gagasan untuk memperluas jangkauan fungsi karakteristik sedemikian hingga fungsi tersebut akan mencakup bilangan real pada interval [0,1]. Nilai keanggotaannya menunjukkan bahwa suatu item dalam semesta pembicaraan tidak hanya berada pada 0 atau 1, namun juga nilai yang terletak diantaranya. Dengan kata lain, nilai kebenaran suatu item tidak hanya benar atau salah. Nilai 0 menunjukkan salah, nilai 1 menunjukkan benar, dan masih ada nilai-nilai yang terletak antara benar dan salah. Himpunan *fuzzy* memiliki 2 atribut, yaitu [KUS-03]:

a. Linguistik, yaitu penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami.
b. Numeris, yaitu suatu nilai (angka) yang menunjukkan ukuran dari suatu variabel.

Ada beberapa hal yang perlu diketahui dalam memahami sistem *fuzzy*, yaitu:

a) Variabel *Fuzzy*.

   Variabel *fuzzy* merupakan variabel yang hendak dibahas dalam suatu sistem *fuzzy*.

b) Himpunan *Fuzzy*.

   Himpunan *fuzzy* merupakan suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu variabel.

c) Semesta Pembicaraan.

   Semesta pembicaraan adalah keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel *fuzzy*. Semesta pembicaraan merupakan himpunan bilangan real yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai semesta pembicaraan dapat berupa bilangan positif maupun negatif. Adakalanya nilai semesta pembicaraan ini tidak dibatasi batas atasnya.

d) Domain.

   Domain himpunan *fuzzy* adalah keseluruhan nilai yang diijinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan *fuzzy*. Seperti

halnya semesta pembicaraan, domain merupakan himpunan bilangan real yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai domain dapat berupa bilangan positif maupun negatif.

2.5 Data *Mining*

Data mining adalah suatu istilah yang digunakan untuk menguraikan penemuan pengetahuan di dalam database. Data Mining merupakan suatu proses yang menggunakan teknik statistika, matematika, kecerdasan buatan dan machine learning untuk mengidentifikasikan informasi yang bermanfaat dan pengetahuan yang terakit dari berbagai database besar [TUR-05]. Data Mining merupakan teknologi yang sangat berguna untuk membantu perusahaan-perusahaan menemukan informasi yang sangat penting dari gudang data mereka. Dengan data mining dapat meramalkan trend dan sifat-sifat perilaku bisnis yang sangat berguna untuk mendukung pengambilan keputusan penting. Analisis yang diotomatisasi yang dilakukan oleh data mining melebihi yang dilakukan oleh sistem pendukung keputusan tradisional yang sudah banyak digunakan. Data Mining mengeksplorasi basis data untuk menemukan pola-pola yang tersembunyi, mencari informasi untuk memprediksi yang mungkin saja terlupakan oleh para pelaku bisnis karena terletak di luar ekspektasi mereka [KHU-07].

Metode data *mining* ini dapat dibagi menjadi beberapa kelompok berdasarkan tugas yang dapat dilakukan, diantaranya yaitu metode deskripsi, estimasi, prediksi, klasifikasi, pengklusteran dan asosiasi [LAR-05].

Prosedur yang umum digunakan untuk permasalahan data *mining* meliputi tahap-tahap sebagai berikut [KAN-03]:

1. Menentukan permasalahan dan merumuskan hipotesis
   Pada tahap ini, ditentukan variabel-variabel dan hipotesis awal.
2. Mengumpulkan data
   Tahap ini berkaitan dengan bagaimana data dihasilkan dan dikumpulkan.
3. *Preprocessing* data
   Dilakukan pembersihan terhadap *outlier*, penanganan *missing value* maupun transformasi data.
4. Memperkirakan model

Pemilihan dan penerapan teknik data *mining* yang sesuai adalah tugas utama dalam tahap ini.

5. Menafsirkan model dan menarik kesimpulan

   Pada tahap ini, dilakukan penafsiran model untuk membantu dalam pengambilan keputusan.

Atribut cenderung memiliki nilai dengan rentang yang sangat bervariasi. Misalnya, dalam menentukan jarak antara dua *record*, atribut dengan rentang nilai yang besar, memiliki lebih banyak pengaruh dalam menentukan jarak daripada atribut dengan rentang nilai yang kecil. Oleh karena itu, perlu dilakukan transformasi data berupa normalisasi terhadap nilai untuk membakukan skala pengaruh yang ada pada atribut, terhadap hasil. Ada beberapa teknik normalisasi data seperti normalisasi *min-max* dan Z-score [MOR-09].

Dalam skripsi digunakan normalisasi *min-max*. Normalisasi *min-max* dihitung dengan persamaan 2.1 [MOR-09]. Normalisasi *min-max* memiliki keunggulan yaitu menjaga relasi pada data. Serta mempunyai fungsi yaitu menyatukan satuan dari berbagai atribut [JAY-11].

$$V' = \frac{V - min_A}{max_A - min_A} \quad (2.1)$$

Dimana,

V' : hasil normalisasi yang nilainya berkisar antara 0 dan 1.

V : nilai atribut A yang akan dinormalisasi.

minA : nilai minimum dari suatu atribut, A

maxA : nilai maksimum dari suatu atribut, A.

2.6 Klasifikasi

Klasifikasi adalah suatu fungsionalitas data *mining* yang akan menghasilkan model untuk memprediksi kelas atau kategori dari objek-objek di dalam baris data. Klasifikasi merupakan proses yang terdiri dari dua tahap, yaitu tahap pembelajaran dan tahap pengklasifikasian.

Pada tahap pembelajaran, sebuah algoritma klasifikasi membangun sebuah model klasifikasi dengan cara menganalisis data *training*. Tahap pembelajaran dapat juga dipandang sebagai tahap pembentukan fungsi atau pemetaan *y = f(x)*

dimana *y* adalah kelas hasil prediksi dan *x* adalah *tuple* yang ingin diprediksi kelasnya.

Selanjutnya, pada tahap pengklasifikasian, model yang telah dihasilkan digunakan untuk melakukan klasifikasi terhadap *unknown* data. Akan tetapi, sebuah model hanya boleh digunakan untuk klasifikasi jika akurasi model tersebut cukup tinggi. Akurasi dapat diketahui dengan cara menguji model tersebut dengan data *test*. Data *test* terdiri dari label kelas yang sudah diketahui, namun data *test* tidak boleh sama dengan data *training* karena menyebabkan pengujian tersebut menunjukkan akurasi yang tinggi, padahal belum tentu demikian [KHU-07].

2.7 Algoritma *k-Nearest Neighbor*

Algoritma *k-Nearest Neighbor* (KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Data pembelajaran diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi data pembelajaran. Sebuah titik pada ruang ini ditandai kelas c jika kelas c merupakan klasifikasi yang paling banyak ditemui pada k buah tetangga terdekat titik tersebut. Dekat atau jauhnya tetangga biasanya dihitung berdasarkan jarak *euclidean*.

Beberapa keuntungan dari metode KNN adalah sebagai berikut:

a. Sederhana dalam penggunaannya.
b. Dapat menangani data training yang mengandung noise.
c. Efektif jika data training besar.

Meskipun memiliki beberapa keuntungan metode KNN juga memiliki beberapa kelemahan seperti berikut ini [HAM-08]:

a. *Computation cost* cukup tinggi karena perlu untuk menghitung jarak setiap data training.
b. Membutuhkan memori yang besar.
c. Rendahnya tingkat akurasi pada dataset multidimensi.
d. Perlu untuk menentukan nilai *k* parameter, jumlah tetangga terdekat.
e. Menggunakan perhitungan jarak, yang belum diketahui pasti jenis jarak yang digunakan.

f. Belum diketahui atribut yang lebih baik untuk menghasilkan hasil terbaik.

Kelebihan dari algoritma KNN itu sendiri adalah sederhana dan mudah diimplementasikan. Algoritma ini mencari *k training record* (tetangga) yang memiliki jarak terdekat dari *record* baru, untuk memprediksi kelas dari *record* baru tersebut [YAN-13]. KNN juga merupakan algoritma yang sering digunakan untuk klasifikasi pada teknik data *mining* [MOR-09].

Untuk menghitung jarak digunakan fungsi jarak *euclidean*. Pertama, dihitung jarak terdekat antara data uji dengan data latih terlebih dahulu yang ditunjukkan oleh persamaan 2.2

$$di = \sqrt{\sum_{i=1}^{p}(x_{2i} - x_{1i})^2} \quad (2.2)$$

Keterangan:

x_1 = data latih

x_2 = data uji

i = variabel data

d = jarak

p = dimensi data

Setelah diketahui jarak antar *record*, kemudian diambil sebanyak *k* tetangga terdekat untuk menentukan label kelas dari *record* baru menggunakan label kelas tetangga.

2.8 *Fuzzy k-Nearest Neighbor*

Fuzzy k-Nearest Neighbor (Fk-NN) merupakan metode klasifikasi yang menggabungkan teknik *fuzzy* dengan *k-Nearest Neighbor classifier*. Algoritma Fk-NN memberikan nilai keanggotaan kelas pada vektor sampel dan bukan menempatkan vektor pada kelas tertentu. Fk-NN merupakan metode klasifikasi yang digunakan untuk memprediksi data uji menggunakan nilai derajat keanggotaan data uji pada setiap kelas. Kemudian diambil kelas dengan nilai derajat keanggotaan terbesar dari data uji sebagai kelas hasil prediksi. Keuntungannya adalah nilai-nilai keanggotaan vektor seharusnya memberikan tingkat jaminan pada hasil klasifikasi. Sebagai contoh, jika vektor diberikan nilai keanggotaan 0.9 pada kelas pertama dan 0.05 pada dua kelas lainnya peneliti dapat cukup yakin bahwa

kelas dengan nilai keanggotaan 0.9 adalah kelas milik vektor tersebut. Di sisi lain, jika sebuah vektor diberi nilai keanggotaan 0.55 pada kelas pertama, 0.44 pada kelas kedua, dan 0.01 pada kelas ketiga, maka peneliti harus ragu-ragu untuk menetapkan vektor berdasarkan hasil ini. Namun, dapat diyakinkan bahwa vektor tersebut bukan milik kelas ketiga. Dalam kasus seperti ini, perlu diperiksa lebih lanjut untuk menentukan klasifikasinya, karena memiliki derajat keanggotaan yang tinggi pada dua kelas yaitu satu dan dua. Pemberian nilai keanggotaan oleh algoritma ini jelas berguna dalam proses klasifikasi.

Sebuah data memiliki nilai keanggotaan pada setiap kelas yang berbeda dengan nilai derajat keanggotaan dalam interval [0, 1]. Teori himpunan *fuzzy* men-generalisasi teori k-NN klasik dengan mendefinisikan nilai keanggotaan sebuah data pada masing-masing kelas. Formula yang digunakan ditunjukkan oleh persamaan 2.3 [KEL-85].

$$u_i(x) = \frac{\sum_{j=1}^{k} u_{ij}(|| x-xj ||^{2/(m-1)})}{\sum_{j=1}^{k}(|| x-xj ||^{2/(m-1)})} \quad (2.3)$$

Keterangan :

k = tetangga terdekat (ditentukan sendiri).

j = kelas status pada tetangga terdekat.

i = jumlah kelas pada data latih (0 dan 1).

m = penentuan bobot jarak antara tetangga dengan nilai keanggotaan.

Dimana i = 1, 2, … , c, dan j = 1, 2, … , k, dengan c adalah jumlah kelas dan k adalah jumlah tetangga terdekat. $|| x - xj ||$ adalah nilai jarak euclidean. Variabel (m) merupakan penentuan seberapa banyak pemberian bobot pada jarak saat menghitung kontribusi jarak kedekatan pada masing-masing tetangga dengan nilai keanggotaan. Jika nilai m adalah dua, maka jarak kontribusi dari setiap titik tetangga(data latih) dibobotkan oleh nilai kebalikan dari jarak titik tetangga tersebut dengan titik yang sedang diklasifikasikan (data uji). Ketika nilai m naik, titik-titik tetangga tersebut dibobotkan lebih merata dan efek dari jarak relatif dari titik yang sedang diklasifikasikan akan berkurang. Ketika nilai m mendekati satu,semakin dekat tetangga maka akan dibobotkan lebih besar

daripada tetangga yang lebih jauh (semakin besar nilai jarak maka semakin besar bobotnya), yang mana hal ini akan mempengaruhi pengurangan jumlah titik (tetangga) yang berkontribusi terhadap nilai keanggotaan dari titik yang sedang diklasifikasikan. Hasil yang ditampilkan pada jurnal ini, menggunakan nilai m=2 tetapi perhatikan bahwa hampir tingkat kesalahan yang diperoleh pada data ini hampir sama dengan menggunakan nilai m yang beragam[KEL85].

Nilai u_{ij} pada $u_i(x)$ terlebih dahulu diproses dengan menggunakan persamaan 2.4 [KEL-85].

$$uij = \begin{cases} 0.51 + (nj/K) * 0.49, \text{ jika } j=i \\ (nj/K) * 0.49, \text{ jika } j \neq i \end{cases} \quad (2.4)$$

Keterangan:

n_j= jumlah anggota kelas j pada suatu dataset K

K= total data latih yang digunakan

j= kelas status tetangga terdekat

i = jumlah kelas (0 dan 1)

2.9 *Edited Fuzzy k-Nearest Neighbor*

Edited Fuzzy k-Nearest Neighbor (EFk-NN) adalah algoritma yang dibuat pada tahun 1972, yang ide utamanya adalah untuk menghapus data yang digunakan jika kelasnya tidak setuju dengan mayoritas kelas tetangganya.

Edited Fuzzy k-Nearest Neighbor adalah sebuah pengembangan metode dari metode Fk-NN. Dalam EFk-NN, nilai keanggotaan digunakan untuk memberikan nilai pada setiap sampel di masing-masing kelas, yang menunjukkan sejauh mana elemen milik himpunan fuzzy . Berikutnya memperkirakan kategori sampel sesuai dengan nilai keanggotaan maksimum . Namun, dalam proses tersebut , beberapa sampel mungkin dalam batas beberapa kelas, dan yang nilai keanggotaan maksimum mungkin menjadi kecil. Dari klasifikasi titik ini, sampel biasanya tidak dapat diandalkan atau tidak berguna yang harus dihapus dari set pengujian . Jika kita masih mengikuti teknik EFk-NN yaitu mengikuti aturan yang mengambil nilai

keanggotaan maksimal, beberapa sampel yang mempunyai nilai keanggotaan kecil maka dihilangkan dari pengujian [ZHA-13].

Langkah - langkah dari pengerjaan metode EFk-NN adalah :

- Langkah 1. Input nilai dari tetangga terdekat k, dan data sampel berlabel $D = \{(X_1, \theta_1), (X_2, \theta_2), \dots, (X_n, \theta_n)\} = \{C_1, C_2, \dots, C_c\}$. Kemudian *D* dibagi menjadi 2 bagian $D = D_1 \cup D_2$. $D_1 = \{Y_1, Y_2, \dots, Y_t\}$ sebagai data latih dan $D_2 = \{Z_1, Z_2, \dots, Z_m\}$ sebagai data uji. D2 ini akan di- *edit* menjadi data latih baru.
- Langkah 2. Menginisasi nilai keanggotaan matrix $U = (u_{ij})_{tc}$ pada langkah pertama, dimana $u_{ij} = u_j(Y_i)$ menunjukkan tingkat dari sampel vektor Y_i di kelas j.
- Langkah 3. Untuk sampel pengujian yang sudah diberikan $Z \in D_2$, hitung jarak antara n-dimensi sampel pengujian Z dan semua sampel berlabel Y_i, untuk semua $i \in \{1,2, \dots, t\}, d(Z, Y_i) = | Z - Y_i |$.
- Langkah 4. Hitung nilai derajat keanggotaan pada setiap D_2 terhadap D_1 menggunakan Fk-NN
- Langkah 5. Hapus *Z* dari data uji D_2 jika kategori prediksi j_{max} berbeda dari kelas label θ of Z , jika tidak maka sajikan Z di data uji D_2.
- Langkah 6. Lakukan langkah 3, langkah 4, langkah 5 sampai semua sampel di D_2 diprediksi. Kemudian data uji yang terbaru adalah data *edited* yang kita cari [ZHA-13].

2.10 Evaluasi Sistem

Penggunaan Threshold

Threshold digunakan untuk mendapatkan hasil prediksi yang lebih akurat. Threshold mempunyai *range* dari 0 – 1. Jika sudah didapat *maximum membership* pada data uji pertama (D_2), maka fungsi threshold dijalankan untuk mengeluarkan data uji yang mempunyai nilai keanggotaan rendah agar akurat dalam mewakili setiap kelasnya. Data uji terbaru yang telah melalui fungsi threshold dijadikan data latih baru untuk dihitung lagi derajat keanggotaannya dan mendapat hasil akurasi yang maksimal [ZHA-13].

Perhitungan Akurasi

Akurasi merupakan seberapa dekat suatu angka hasil pengukuran terhadap angka sebenarnya (*true value atau reference value*). Dalam penelitian ini akurasi diagnosis dihitung dari jumlah diagnosis yang tepat dibagi dengan jumlah data. Tingkat akurasi diperoleh dengan perhitungan sesuai dengan persamaan 2.5 [NUG-06].

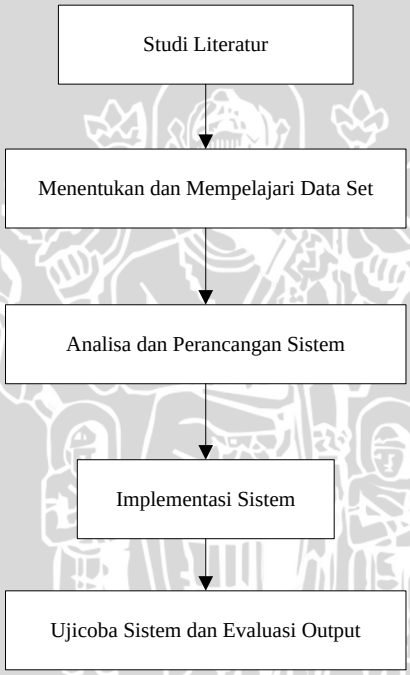
$$\text{Akurasi } (\%) = \frac{\sum data\ uji\ benar}{\sum\ total\ data\ uji}\ x\ 100\% \quad (2.5)$$

Jumlah prediksi benar adalah jumlah *record* data uji yang diprediksi kelasnya menggunakan metode klasifikasi dan hasilnya sama dengan kelas sebenarnya. Sedangkan jumlah total prediksi adalah jumlah keseluruhan *record* yang diprediksi kelasnya (seluruh data uji).

BAB III
METODOLOGI PENELITIAN DAN PERANCANGAN

Dalam bab metodologi penelitian dan perancangan ini akan dibahas metode perancangan dan langkah-langkah yang dilakukan dalam penelitian menggunakan metode *Edited Fuzzy k-Nearest Neighbor* untuk diagnosa penyakit Parkinson.

Alur diagram pada metodologi penelitian dan perancangan Penerapan *Edited Fuzzy k–Nearest Neighbor* (EFk-NN) untuk Diagnosa Penyakit Parkinson dapat dilihat pada gambar 3.1.



Gambar 3.1 Langkah-langkah penelitian

Penjelasan tahapan – tahapan diatas sebagai berikut :

1. Mempelajari Studi literature yang terkait dengan masalah penyakit *Parkinson* dan metode *Edited Fuzzy k-Nearest Neighbor*.
2. Melakukan pengumpulan data – data dari *Parkinson Dataset* yang diperoleh dari *https://archive.ics.uci.edu/ml/datasets/Parkinsons*
3. Menganalisis sistem dari data *Parkinson Dataset* dan melakukan perancangan sistem yang meliputi pelatihan dan pengujian.

4. Mengimplementasikan sistem untuk membuat perangkat lunak berdasarkan analisis dan perancangan dengan menggunakan bahasa pemrograman Java.
5. Melakukan uji coba terhadap perangkat lunak.
6. Mengevaluasi uji coba terhadap perangkat lunak.

3.1 Studi Literatur

Dalam penelitian ini dibutuhkan studi literatur yang dibutuhkan untuk merealisasikan tujuan dan penyelesaian masalah. Teori – teori mengenai penyakit Parkinson, algoritma *edited fuzzy k-nearest neighbor* digunakan sebagai dasar penelitian yang berasal dari buku – buku, browsing dari internet dan jurnal. Kemudian data yang diperoleh diubah sehingga dapat digunakan untuk analisis masalah. Setelah dianalisis maka dapat diimplementasikan ke dalam program.

3.2 Data Penelitian

Data yang digunakan dalam penerapan *edited fuzzy k-nearest neighbor* diambil dari situs *https://archive.ics.uci.edu/ml/datasets/Parkinsons* dan menggunakan data penyakit Parkinson pada tahun 2008. Pada *dataset* Parkinson, atribut yang digunakan adalah MDVP:Fo(Hz), MDVP:Fhi(Hz), MDVP:Flo(Hz), MDVP:Jitter(%), status, spread1, spread2, PPE.

3.3 Analisa dan Perancangan Sistem

3.3.1. Deskripsi Umum Sistem

Secara umum sistem yang dibuat merupakan perangkat lunak yang mengimplementasikan algoritma *Edited Fuzzy k-Nearest Neighbor* (EFk-NN) untuk mengklasifikasikan pasien menjadi 2 kategori yaitu Sehat dan Parkinson.

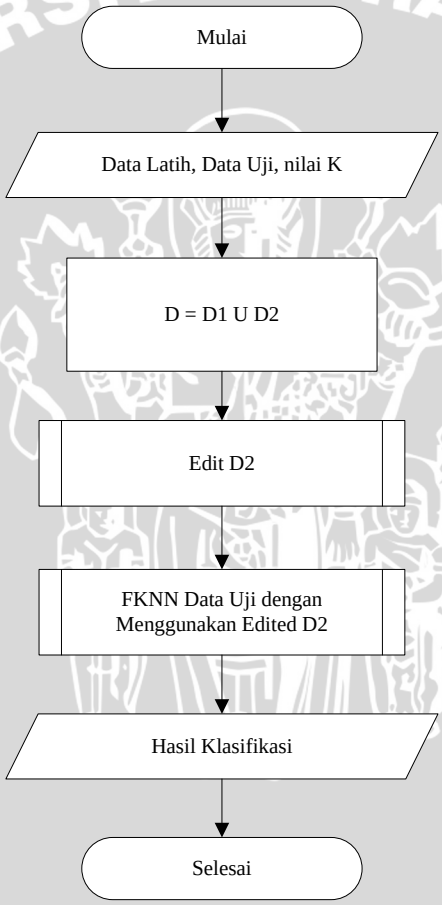
Perangkat lunak ini akan menguji keakuratan hasil klasifikasi *dataset* penderita Parkinson. Parameter uji yang berkaitan dengan nilai *threshold* dan data latih yang berpengaruh terhadap tingkat akurasi.

3.3.2. Perancangan Sistem

Tahap prediksi sistem EFk-NN untuk prediksi penderita Parkinson, langkah – langkahnya adalah sebagai berikut:

1. Menginputkan data latih dan data uji pasien sehat, parkinson dan *k*.
2. Melakukan proses Edit D2.
3. Melakukan proses FKNN menggunakan *edited* D2.
4. Didapatkanlah hasil klasifikasi.

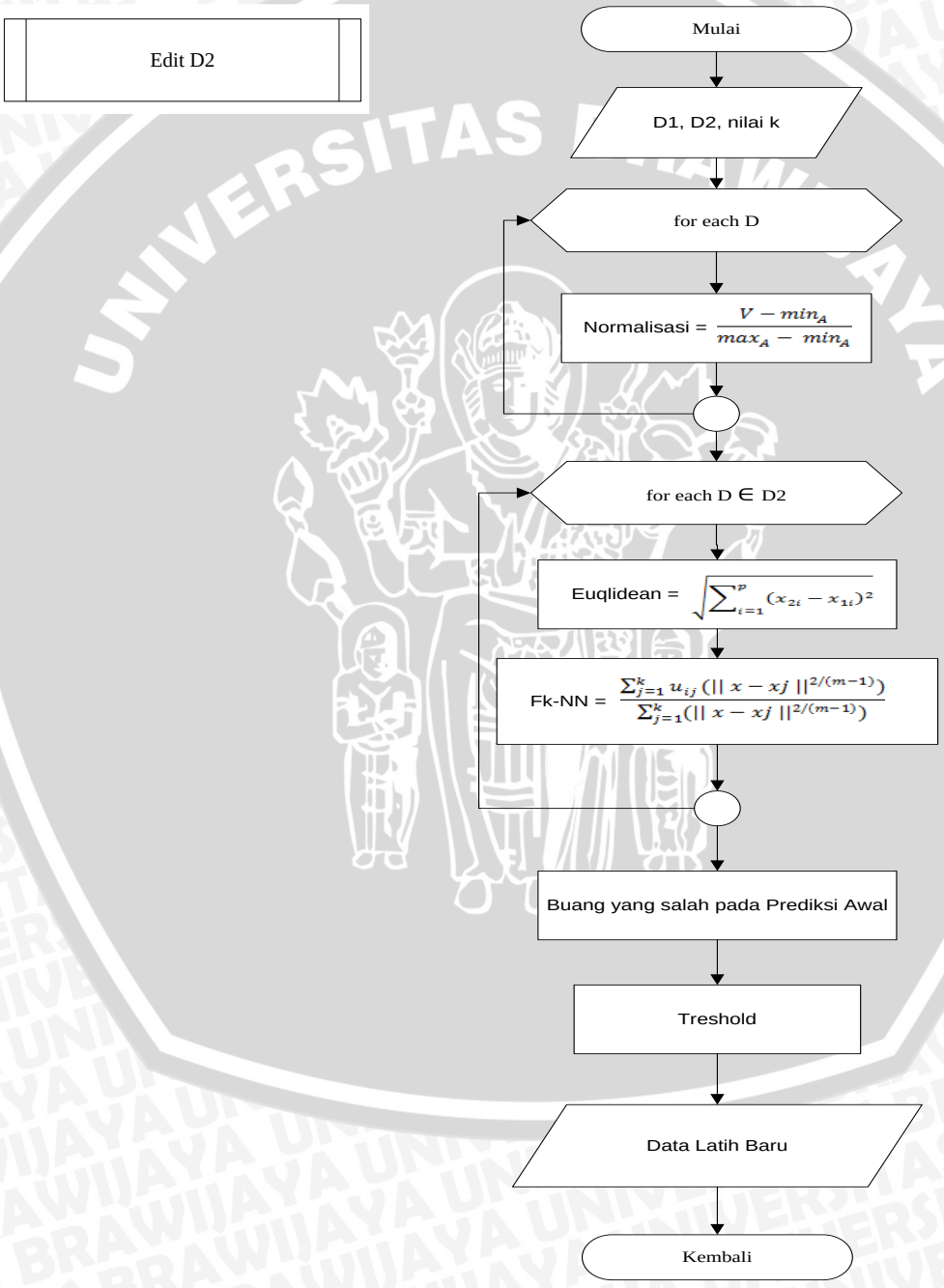
Tahapan proses klasifikasi dapat dlihat pada gambar 3.2 di bawah ini :



Gambar 3.2 Alur proses klasifikasi

3.3.2.1. Edit D2

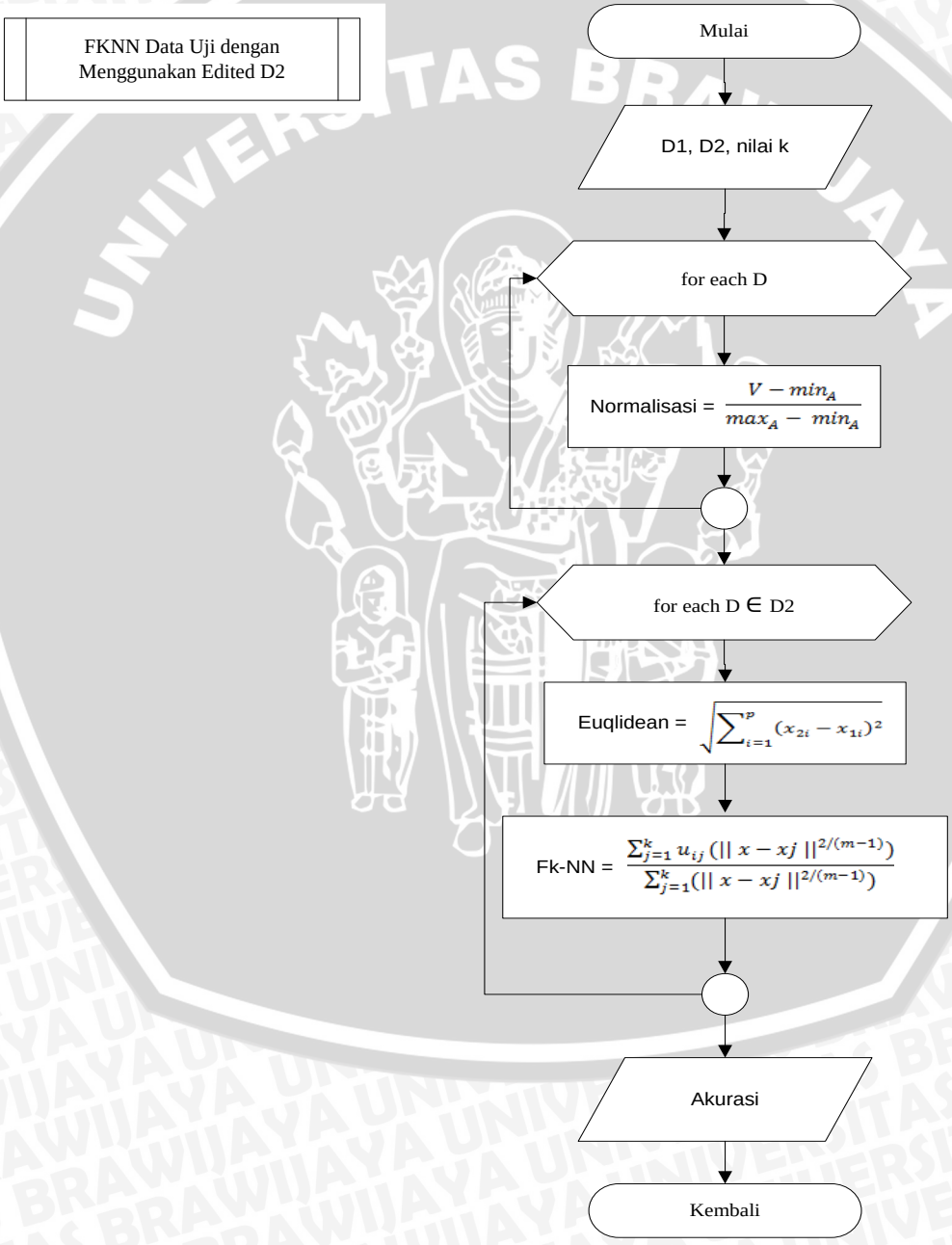
Pada proses Edit D2, didalamnya terdapat beberapa sub proses. Proses Edit D2 adalah proses awal untuk mencari data latih baru yang akan digunakan pada proses selanjutnya. Didalam proses edit D2, terdapat proses normalisasi, *Euclidean*, Fk-NN, buang data yang salah dan threshold. Diagram alur proses Edit D2 ditunjukkan oleh gambar (3.3) :



Gambar 3.3 Edit D2

3.3.2.2. Fk-NN data uji dengan menggunakan Edit D2

Pada proses Fk-NN data uji dengan menggunakan Edit D2, didalamnya terdapat beberapa sub proses. Proses Fk-NN data uji dengan menggunakan Edit D2 adalah proses yang akan menghasilkan hasil akurasi. Pada proses Fk-NN data uji dengan menggunakan Edit D2 terdapat proses normalisasi, *Euclidean*, Fk-NN. Diagram alur proses Fk-NN data uji dengan menggunakan Edit D2 ditunjukkan oleh gambar (3.4) :



Gambar 3.4 Fk-NN data uji dengan menggunakan Edit D2.

3.4. Contoh Perhitungan Manual

Pada subbab ini menampilkan perhitungan manual untuk proses diagnosa penyakit parkinson dan digunakan hanya beberapa atribut dataset tersebut yang berjumlah 8 atribut. Data yang diambil sebanyak 12 record dengan rincian record ke-1 sampai 6 merupakan D1 dan record ke-7 sampai 12 merupakan D2. D1 merupakan data latih dan D2 merupakan data uji.

3.4.1 Data Latih dan Data Uji pada Data Pasien Parkinson dan Pasien Sehat

Pada contoh perhitungan kali ini digunakan 7 data latih (D1) dan 7 data uji (D2). Data latih dan data uji yang dipakai seperti yang ditunjukkan pada tabel (3.1): Tabel (3.1) D1 dan D2.

KELAS STATUS	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	PPE	SPREAD1	SPREAD2	
0	202.266	211.604	197.079	0.0018	0.056141	-7.695734	0.17854	
0	203.184	211.526	196.16	0.00178	0.044539	-7.964984	0.163519	
1	119.992	157.302	74.997	0.00784	0.284654	-4.813031	0.266482	D1 = Data Latih
1	208.083	253.792	91.802	0.00757	0.231723	-5.410336	0.288917	
1	136.926	159.866	131.276	0.00293	0.138512	-6.547148	0.152813	
0	197.076	206.896	192.055	0.00289	0.085569	-7.3483	0.177551	
1	116.682	131.111	111.555	0.0105	0.332634	-4.443179	0.311173	
0	241.404	248.834	232.483	0.00281	0.117399	-6.793547	0.158266	
0	199.228	209.512	192.091	0.00241	0.068501	-7.682587	0.173319	
1	139.173	179.139	76.556	0.0039	0.199889	-5.660217	0.254989	D2 = Data Uji
1	177.876	192.921	168.013	0.00411	0.165827	-6.149653	0.218037	
0	243.439	250.912	232.435	0.0021	0.09147	-7.057869	0.091608	

Keterangan :

Kelas status pasien dibedakan menjadi 2 yaitu angka 0 dan 1. 0 adalah sehat dan 1 adalah penderita Parkinson. Keterangan setiap parameter ada pada bab 2.2.

Dari 12 data tersebut, memiliki 2 buah kelas status sesuai dengan nilai output, dimana ke-2 kelas tersebut adalah kelas pasien sehat dan kelas pasien Parkinson.

3.4.2 Melakukan proses normalisasi terhadap nilai atribut

Proses perhitungan normalisasi terhadap nilai atribut ialah dengan menggunakan normalisasi *min - max*. Sebelum melakukan normalisasi terhadap

nilai – nilai atribut, terlebih dahulu dicari nilai minimum dan nilai maksimum dari setiap atribut. Nilai minimum dan maksimum ditunjukkan pada tabel 3.2 :

Tabel (3.2) Nilai minimum dan maksimum.

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	PPE	SPREAD1	SPREAD2
minimum	116.682	131.111	74.997	0.00178	0.044539	-7.964984	0.091608
maksimum	243.439	253.792	232.483	0.0105	0.332634	-4.443179	0.311173

Setelah didapat nilai minimum dan maksimum, dilakukan proses normalisasi min – max. Berikut ini adalah contoh perhitungan normalisasi min - max nilai record pertama untuk atribut ke-1 (*MDVP:Fo(Hz)*).

V = 202,266 (nilai record pertama atribut *MDVP:Fo(Hz)*)

$\min_A$ = 116,682

$\max_A$ = 243,439

setelah diketahui nilai minimum dan maksimum, kemudian dilakukan perhitungan menggunakan persamaan 2.1.

$$v' = \frac{v - \min_a}{\max_a - min_a}$$

$$= \frac{202{,}266 - 116{,}682}{243{,}439 - 116{,}682}$$

= 0.675181647

Dari perhitungan ini diperoleh nilai normalisasi untuk record pertama pada atribut ke -1, yaitu 0.675181647. Proses yang sama juga dilakukan untuk nilai record pada seluruh atribut. Hasil perhitungan normalisasi *min - max* ditampilkan pada tabel 3.3.

Tabel (3.3) Data latih dan data uji yang telah dinormalisasi.

no data	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	PPE	SPREAD1	SPREAD2	Status	
1	0.675181647	0.656116269	0.775192716	0.002293578	0.040271438	0.076452274	0.395928313	0	
2	0.68242385	0.655480474	0.769357276	0	0	0	0.32751577	0	
3	0.026112956	0.213488641	0	0.694954128	0.833457714	0.894982261	0.79645663	1	D1 = Data Latih
4	0.721072603	1	0.106707898	0.663990826	0.649730124	0.725380309	0.898635939	1	
5	0.159707156	0.234388373	0.357358749	0.131880734	0.326187542	0.402587878	0.278755722	1	
6	0.634237162	0.617740318	0.743291467	0.127293578	0.1424183	0.175104527	0.391423952	0	
7	0	0	0.23213492	1	1	1	1	1	
8	0.98394566	0.959586244	1	0.118119266	0.252902688	0.332624038	0.303591192	0	
9	0.651214529	0.639063914	0.743520059	0.072247706	0.083173953	0.080185303	0.372149477	0	D2 = Data Uji
10	0.177433988	0.391486864	0.009899293	0.243119266	0.539231851	0.654427772	0.744112222	1	
11	0.482766238	0.503826998	0.590630278	0.267201835	0.421000017	0.515454717	0.575815818	1	
12	1	0.976524482	0.999695211	0.036697248	0.162901126	0.257571047	0	0	

3.4.3 Menghitung jarak record baru pada data uji dengan tiap record pada data latih

Dari data uji dengan data pelatihan, dicari terlebih dahulu jarak terdekat "Euclidean distance", dimana rumus untuk Euclidean distance dapat dilihat pada persamaan (2.2). Berikut ini adalah contoh perhitungan nilai jarak terdekat dari *record* pertama data latih dengan record data uji menggunakan data uji ke-1.

$$dist\ (x1, x2) = \sqrt{\sum_{i=1}^{p} (x_{2i} - x_{1i})^2}$$

Selanjutnya dihitung lagi jarak antara record uji dengan record data latih yang lainnya. Hasil perhitungan jarak ditampilkan pada tabel 3.4.

$$Jarak\ (data\ latih\ ke1\ , data\ uji\ ke1$$

$$= \sqrt{\begin{array}{c}((0-0.67)^2) + ((0-0.66)^2) + ((0.23-0.77)^2) + \\ ((1-0.002)^2) + ((1-0.04)^2) + ((1-0.08)^2) + \\ ((1-0.4)^2)\end{array}}$$

$$= 2.077404777$$

Tabel (3.4) Jarak *Euclidean* pada D2 ke-1 terhadap semua parameter D1 ke-1.

Kelas	DATA UJI 1
0	2.077404777
0	2.153183658
1	0.522870781
1	1.362702989
1	1.476778296
0	1.895526606

Setelah itu diurutkan dari nilai terkecil sampai terbesar untuk mengetahui jarak tetangga terdekat. Hasil perhitungan jarak setelah diurutkan dari yang terkecil ditunjukkan pada tabel 3.5.

Tabel (3.5) Jarak euclidean pada D2 ke-1 setelah diurutkan dari yang terkecil.

No data	Jarak euclidean	kelas
3	0.522870781	1
4	1.362702989	1
5	1.476778296	1
6	1.895526606	0
1	2.077404777	0
2	2.153183658	0

3.4.4 Menentukan k record terdekat

Berdasarkan hasil perhitungan jarak terkecil pada tabel 3.4, kemudian dilakukan pengurutan dari jarak terkecil yang ditunjukkan pada tabel 3.5. kemudian diambil k record dengan jarak terkecil. Apabila ditentukan k = 3, maka record yang terpilih adalah record ke-3, 4, dan 5.

Tabel (3.6) seleksi 3 data dengan jarak terdekat.

Data ke-	Jarak euclidean	kelas
3	0.522870781	1
4	1.362702989	1
5	1.476778296	1

3.4.5 Menentukan maximum membership dan prediksi awal untuk menentukan data latih baru

Proses menentukan maximum membership dan kelas target dengan mencari nilai *membership* untuk tiap kelas j menggunakan persamaan 2.4. K=6 (total jumlah data latih), u_0=pasien Sehat, u_1=pasien Parkinson, n_0=3, n_1=3.

$U_{1(1)}$= 0.51+(3/6)*0.49

= 0.51+0.5*0.49

= 0.51+0.245

= 0.755

$U_{1(0)}$= (3/6)*0.49

= 0.5*0.49

= 0.245

$U_{0(0)}$= 0.51+(3/6)*0.49

= 0.51+0.5*0.49

= 0.51+0.245

= 0.755

$U_{0(1)}$= (3/6)*0.49

= 0.5*0.49

= 0.245

Tabel (3.7) Perhitungan nilai *membership* pada D2 ke-1.

			Kelas i	
Data ke-	Jarak euclidean	kelas	0	1
3	0.522870781	1	0.245	0.755
4	1.362702989	1	0.245	0.755
5	1.476778296	1	0.245	0.755

Setelah didapatkan membership untuk tiap kelas j dilanjutkan dengan mencari nilai keanggotaan sebuah data pada masing – masing kelas dengan menggunakan persamaan 2.3

$$u_0 = \frac{\sum_{j=1}^{k} u_{ij}(|| x-xj ||^{2/(m-1)})}{\sum_{j=1}^{k}(|| x-xj ||^{2/(m-1)})}$$

$$= \frac{\left(0.245 * (0.523(\frac{2}{2-1})\right) + \left(0.245 * (1.363^{\frac{2}{2-1)}})\right) + \left(0.245 * (1.477^{\frac{2}{2-1}})\right)}{(0.523^{\frac{2}{2-1}}) + (1.363^{\frac{2}{2-1)}} + (1.477^{\frac{2}{2-1}})}$$

$$= \frac{1.14}{4.65}$$

$$= 0.245$$

$$u_1 = \frac{\sum_{j=1}^{k} u_{ij}(|| x-xj ||^{-2/(m-1)})}{\sum_{j=1}^{k}(|| x-xj ||^{-2/(m-1)})}$$

$$= \frac{\left(0.755 * (0.523^{\frac{2}{2-1}})\right) + \left(0.755 * (1.363^{\frac{2}{2-1)}})\right) + \left(0.755 * (1.477^{\frac{2}{2-1}})\right)}{(0.523^{\frac{2}{2-1}}) + (1.363^{\frac{2}{2-1)}} + (1.477^{\frac{2}{2-1}})}$$

$$= \frac{3.51}{4.65}$$

$$= 0.755$$

Berdasarkan hasil perhitungan nilai keanggotaan didapat dua nilai keanggotaan, untuk menentukan kelas target maka dipilih nilai keanggotaan terbesar yaitu 0.755 sehingga kelas targetnya yaitu 1 (pasien parkinson). Jadi, status diagnosa pasien tersebut adalah parkinson.

Setelah melakukan pengujian data uji langkah yang sama dilakukan terhadap data uji yang lain sehingga didapat prediksi awal sistem yang dihitung berdasarkan persamaan (2.5).

Tabel (3.8) Perhitungan prediksi awal.

DATA UJI KE-	KELAS DATA	KELAS HASIL PERHITUNGAN	HASIL PREDIKSI
1	1	1	BENAR
2	0	0	BENAR
3	0	0	BENAR
4	1	1	BENAR
5	1	0	SALAH
6	0	0	BENAR

Jika dilihat dari tabel 3.7, Data ke-5 tidak dilanjutkan untuk proses selanjutnya karena pada prediksinya salah maka data uji yang yang akan dilanjutkan untuk fungsi threshold adalah data uji ke 1, ke 2, ke 3, ke 4 dan ke 6.

Setelah didapat dua nilai keanggotaan pada setiap data uji dan hasil prediksi maka dilakukan threshold pada setiap u_0 dan u_1 pada setiap data prediksi yang benar.

Tabel (3.9) nilai keanggotaan data uji yang mempunyai prediksi benar.

Data uji 1	u(0)	0.245
	u(1)	0.755
Data uji 2	u(0)	0.755
	u(1)	0.245
Data uji 3	u(0)	0.755
	u(1)	0.245
Data uji 4	u(0)	0.245
	u(1)	0.755
Data uji 6	u(0)	0.755
	u(1)	0.245

Threshold yang digunakan adalah 0,7 jadi nilai keanggotaan pada data uji yang dibawah 0,7 akan dihapus dari data uji yang nantinya akan dijadikan data latih baru pada *Edited Fuzzy K-Nearest Neighbor*.

Jadi data uji yang diatas *threshold* dan yang akan dijadikan data latih baru pada proses *Edited Fuzzy K-Nearest Neighbor* adalah data uji ke 1,ke 2, ke 3, ke 4 ke 6.

3.4.6 Data latih baru yang didapat dari proses threshold dan data uji baru yang diambil data set

Data uji yang sebelumnya sudah melalui proses Fk-NN dan *threshold* digunakan lagi menjadi data latih baru (data *edited*) pada proses *Edited Fuzzy K-Nearest Neighbor*. Data latih tersebut diuji lagi dengan data uji baru untuk melalui proses *Edited Fuzzy K-Nearest*. Data edited dan data uji baru yang dipakai ditunjukkan pada tabel 3.10.

Tabel (3.10) Data *edited* dan data uji baru

KELAS STATUS	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	PPE	SPREAD1	SPREAD2	
1	116.682	131.111	111.555	0.0105	0.332634	-4.443179	0.311173	
0	241.404	248.834	232.483	0.00281	0.117399	-6.793547	0.158266	
0	199.228	209.512	192.091	0.00241	0.068501	-7.682587	0.173319	Data latih baru
1	139.173	179.139	76.556	0.0039	0.199889	-5.660217	0.254989	
0	243.439	250.912	232.435	0.0021	0.09147	-7.057869	0.091608	
0	243.439	250.912	232.435	0.0021	0.09147	-7.057869	0.091608	
1	202.544	241.35	164.168	0.00254	0.159777	-6.132663	0.220617	Data uji baru
0	117.004	144.466	99.923	0.00353	0.163118	-6.012559	0.229298	

Terdapat data uji baru yang berstatus 0 berjumlah 2 pasien dan 1 berjumlah 1 pasien untuk menguji data *edited*.

3.4.7 Melakukan proses normalisasi terhadapat setiap nilai atribut

Mengulang lagi proses normalisasi pada setiap data *edited*. Sebelum melakukan normalisasi terhadap nilai – nilai atribut, terlebih dahulu dicari nilai minimum dan nilai maksimum dari setiap atribut. Nilai minimum dan maksimum ditunjukkan pada tabel 3.11.

Tabel (3.11) Nilai minimum dan maksimum baru.

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	PPE	SPREAD1	SPREAD2
minimum	116.682	131.111	76.556	0.0021	0.068501	-7.682587	0.091608
maksimum	243.439	250.912	232.483	0.0105	0.332634	-4.443179	0.311173

Berikut ini adalah contoh perhitungan normalisasi nilai record pertama untuk atribut ke-1 (*MDVP:Fo(Hz)*).

V = 116,682 (nilai record pertama atribut *MDVP:Fo(Hz)*)

$\min_A$ = 116,682

$\max_A$ = 243,439

setelah diketahui nilai minimum dan maksimum, kemudian dilakukan perhitungan menggunakan persamaan 2.1.

$$v' = \frac{v - \min_a}{\max_a - min_a}$$

$$= \frac{116{,}682 - 116{,}682}{243{,}439 - 116{,}682}$$

$$= 0$$

Dari perhitungan ini diperoleh nilai normalisasi untuk record pertama pada atribut ke -1, yaitu 0. Proses yang sama juga dilakukan untuk nilai record pada seluruh atribut. Hasil perhitungan normalisasi *min - max* ditampilkan pada tabel 3.12.

Tabel (3.12) Data *edited* dan data uji baru yang sudah dinormalisasi.

no data	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	PPE	SPREAD1	SPREAD2	Status	
1	0	0	0.224457599	1	1	1	1	1	
2	0.98394566	0.982654569	1	0.08452381	0.185126433	0.274445207	0.303591192	0	
3	0.651214529	0.654426925	0.740955704	0.036904762	0	0	0.372149477	0	Data latih baru
4	0.177433988	0.400898156	0	0.214285714	0.497431218	0.624302342	0.744112222	1	
5	1	1	0.999692164	0	0.086959978	0.192849434	0	0	
6	1	1	0.999692164	0	0.086959978	0.192849434	0	0	
7	0.67737482	0.920184306	0.561878315	0.052380952	0.345568331	0.478459027	0.587566324	1	Data uji baru
8	0.002540294	0.111476532	0.149858588	0.170238095	0.358217262	0.515534937	0.627103591	0	

3.4.8 Menentukan jarak record baru pada data uji dengan tiap record pada data *edited*

Dari data uji dengan data pelatihan, dicari terlebih dahulu jarak terdekat "Euclidean distance", dimana rumus untuk Euclidean distance dapat dilihat pada persamaan (2.2). Berikut ini adalah contoh perhitungan nilai jarak terdekat dari *record* pertama data *edited* dengan record data uji beru menggunakan data uji baru ke-1.

$$dist\ (x1, x2) = \sqrt{\sum_{i=1}^{p} (x_{2i} - x_{1i})^2}$$

Selanjutnya dihitung lagi jarak antara record uji dengan record data latih yang lainnya. Hasil perhitungan jarak ditampilkan pada tabel 3.12.

$$Jarak\ (data\ latih\ ke1\ , data\ uji\ ke1$$

$$= \sqrt{\begin{array}{c}((1-0)^2) + ((1-0)^2) + ((0.99-0.22)^2) + \\ ((0-1)^2) + ((0.09-1)^2) + ((0.2-1)^2) + \\ ((0-1)^2)\end{array}}$$

= 2.46700684

Tabel (3.13) Hasil perhitungan jarak euclidean menggunakan data uji baru ke 1 terhadap semua parameter data *edited*.

Kelas	DATA UJI 1
1	2.46700684
0	0.298980298
0	0.590837639
1	1.709065811
0	0.236570326

Setelah itu diurutkan dari nilai terkecil sampai terbesar untuk mengetahui jarak tetangga terdekat. Hasil perhitungan jarak setelah diurutkan dari yang terkecil ditunjukkan pada tabel 3.14.

Tabel (3.14) Nilai data uji baru setelah diurutkan dari yang terkecil.

No data	Jarak euclidean	kelas
5	0.236570326	0
2	0.298980298	0
3	0.590837639	0
4	1.709065811	1
1	2.46700684	1

3.4.9 Menentukan k record baru

Berdasarkan hasil perhitungan jarak terkecil pada tabel 3.15, kemudian dilakukan pengurutan dari jarak terkecil yang ditunjukkan pada tabel 3.13. kemudian diambil k record dengan jarak terkecil. Apabila ditentukan k = 3, maka record yang terpilih adalah record ke-5, 2, dan 3.

Tabel (3.15) seleksi 3 data baru dengan jarak terdekat.

Data ke-	Jarak euclidean	kelas
5	0.236570326	0
2	0.298980298	0
3	0.590837639	0

3.4.10 Menentukan maximum membership, kelas target dan akurasi

Proses menentukan maximum membership dan kelas target dengan mencari nilai membership untuk tiap kelas j menggunakan persamaan 2.4. K=5 (total jumlah data latih), u_0=pasien Sehat, u_1=pasien Parkinson, n_0=3, n_1=2

$U_{1(1)}$= 0.51+(2/5)*0.49

= 0.51+0.4*0.49

= 0.51+0.196

= 0.706

$U_{1(0)}$= (2/5)*0.49

= 0.4*0.49

= 0.196

$U_{0(0)}$= 0.51+(3/5)*0.49

= 0.51+0.6*0.49

= 0.51+0.294

= 0.804

$U_{0(1)}$= (3/5)*0.49

= 0.6*0.49

= 0.294

Tabel (3.16) Perhitungan nilai *membership* pada data uji baru 1.

Data ke-	Jarak euclidean	kelas	Kelas i	
			0	1
5	0.236570326	0	0.804	0.196
2	0.298980298	0	0.804	0.196
3	0.590837639	0	0.804	0.196

Setelah didapatkan membership untuk tiap kelas j dilanjutkan dengan mencari nilai keanggotaan sebuah data pada masing – masing kelas dengan menggunakan persamaan 2.3.

$$u_0 = \frac{\sum_{j=1}^{k} u_{ij}(|| x - xj ||^{2/(m-1)})}{\sum_{j=1}^{k}(|| x - xj ||^{2/(m-1)})}$$

$$= \frac{\left(0.804 * (0.236^{\left(\frac{2}{2-1}\right)})\right) + \left(0.804 * (0.299^{\frac{2}{2-1)}})\right) + \left(0.804 * (0.591^{\frac{2}{2-1}})\right)}{(0.236^{\frac{2}{2-1}}) + (0.299^{\frac{2}{2-1)}} + (0.591^{\frac{2}{2-1}})}$$

$$= \frac{25.6}{31.92}$$

$$= 0.804$$

$$u_1 = \frac{\sum_{j=1}^{k} u_{ij}(|| x - xj ||^{2/(m-1)})}{\sum_{j=1}^{k}(|| x - xj ||^{2/(m-1)})}$$

$$= \frac{\left(0.196 * (0.236^{\frac{2}{2-1}})\right) + \left(0.196 * (0.299^{\frac{2}{2-1}})\right) + \left(0.196 * (0.591^{\frac{2}{2-1}})\right)}{(0.236^{\frac{2}{2-1}}) + (0.299^{\frac{2}{2-1}}) + (0.591^{\frac{2}{2-1}})}$$

$$= \frac{6.26}{31.92}$$

$$= 0.196$$

Berdasarkan hasil perhitungan nilai keanggotaan didapat dua nilai keanggotaan, untuk menentukan kelas target maka dipilih nilai keanggotaan terbesar yaitu 0.804 sehingga kelas targetnya yaitu 0 (pasien sehat). Jadi, status diagnosa pasien tersebut adalah sehat.

Setelah melakukan pengujian data uji baru, langkah yang sama dilakukan terhadap data uji baru yang lain sehingga didapat akurasi sistem yang dihitung berdasarkan persamaan (2.7).

Tabel (3.17) Perhitungan Akurasi Sistem.

DATA UJI KE-	KELAS DATA	KELAS HASIL PERHITUNGAN	
1	0	0	BENAR
2	1	0	SALAH
3	0	1	SALAH

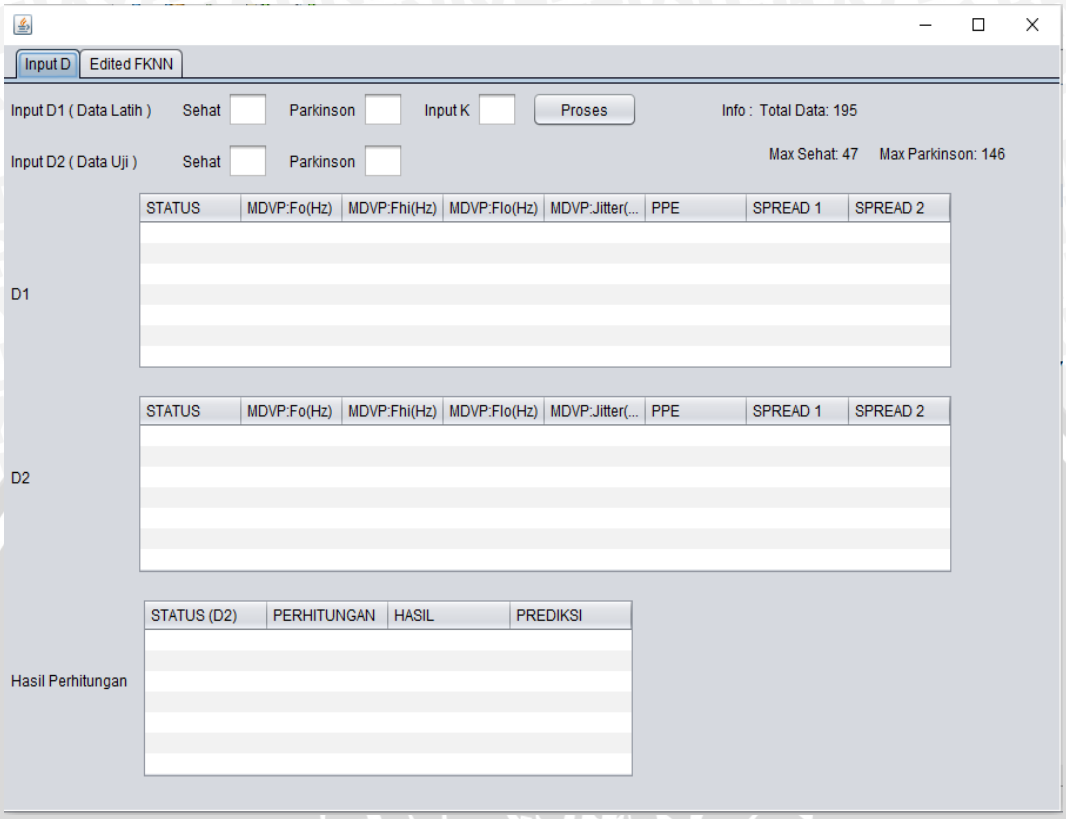
$$Akurasi\ (\%) = \frac{1}{3}\ X\ 100\%$$

$$= 33{,}33\ \%$$

3.5 Perancangan Antarmuka

Antarmuka (*interface*) untuk menampilkan data yang diolah oleh sistem, terdiri dari 2 bagian, yaitu Input D untuk memanggil, melatih dan menguji D1 dan D2 sehat atau parkinson dan menampilkan nilai k yang dimasukkan. *Edited* Fk-NN untuk memanggil dan menguji data uji sehat dan data uji parkinson yang telah diproses sebelumnya dan menampilkan hasil akurasi dengan memasukkan nilai k dan *threshold* terlebih dahulu. Perancangan antarmuka sistem akan ditunjukkan pada gambar 3.5 dan 3.6.

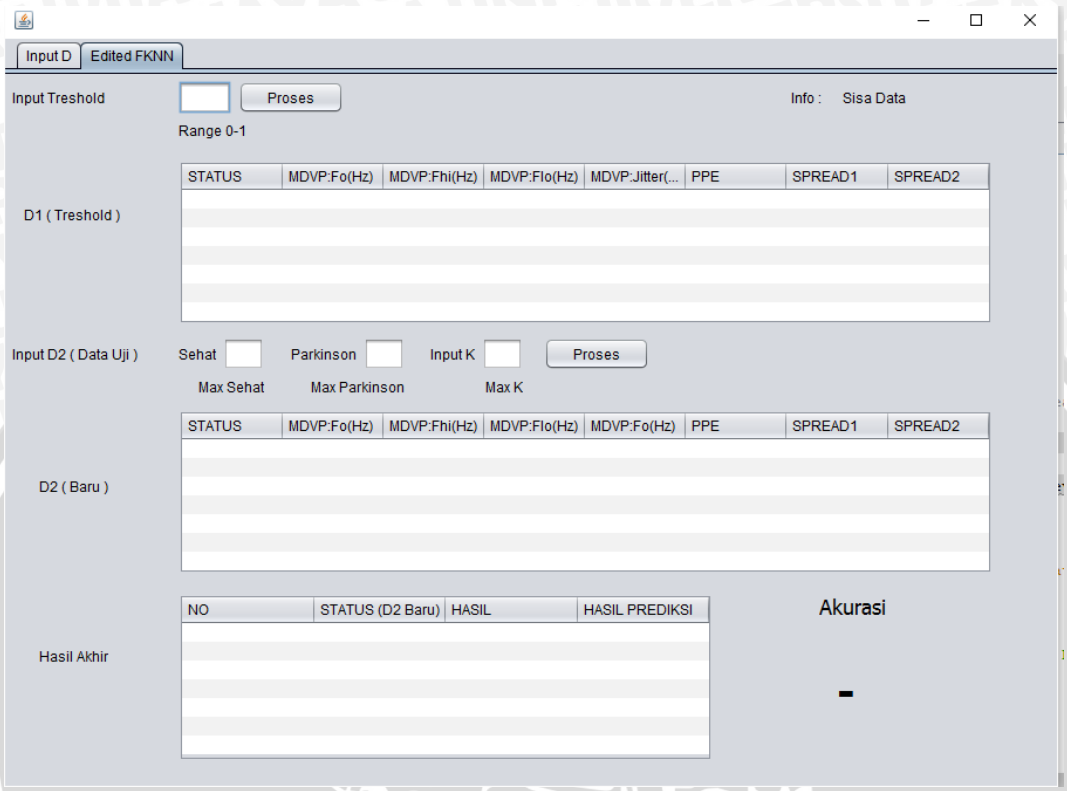
Antarmuka input D adalah antarmuka awal ketika menjalankan program. Terdapat 5 input dan 3 tabel pada antarmuka input D.



Gambar 3.5 Input D

1. Tombol Proses digunakan untuk mengambil data latih dan data uji sehat dan parkinson secara acak, juga menginputkan nilai k.
2. Tabel D1 dan D2 digunakan untuk menampilkan data latih dan data uji yang diinputkan.
3. Tabel Hasil Perhitungan digunakan untuk menampilkan prediksi awal pada perhitungan Fk-NN.

Antarmuka Edited FKNN adalah antarmuka dimana *user* akan mengetahui hasil diagnosa dan tingkat akurasi. Terdapat 4 input dan 3 tabel pada antarmuka Edited FKNN.



Gambar 3.6 Data Uji

1. Tombol Proses pertama digunakan untuk menginputkan nilai *threshold* yang akan dipakai.
2. Tabel D1 (*Threshold*) digunakan untuk menampilkan data uji yang benar dan diatas nilai *threshold* pada perhitungan sebelumnya yang dijadikan data latih baru.
3. Tombol Proses kedua digunakan untuk mengambil data uji sehat dan Parkinson baru.
4. Tabel D2 (Baru) digunakan untuk menampilkan data uji sehat dan Parkinson baru yang diinputkan.
5. Tabel Hasil Akhir digunakan untuk menampilkan hasil prediksi akhir.
6. *Label* "-" untuk menampilkan hasil akurasi dari proses Edited Fuzzy KNN.

3.6 Perancangan Uji Coba

Setelah sistem selesai dibuat, langkah selanjutnya adalah melakukan pengujian terhadap sistem tersebut. Pengujian dilakukan untuk mengetahui tingkat akurasi dari hasil klasifikasi *Parkinson Dataset* dengan menggunakan metode *Edited Fuzzy k-Nearest Neighbor*.

Terdapat 3 macam pengujian yang dilakukan dalam penelitian ini, yaitu: pengujian untuk mengetahui pengaruh nilai *threshold*, sebaran data dan perbandingan hasil akurasi yang didapat dengan *Fuzzy k-Nearest Neighbor* dan *Edited Fuzzy k-Nearest Neighbor*.

3.7 Uji Pengaruh Nilai *Threshold* Terhadap Tingkat Akurasi

Uji pengaruh nilai *Threshold* terhadap tingkat akurasi dilakukan pada beberapa data latih (D1) dan data uji (D2) sehat dan Parkinson. Dengan beberapa nilai *threshold* berbeda-beda, tentunya akan mempengaruhi keakuratan hasil akurasi. Dalam pengujian ini ditentukan nilai *threshold* = 0.5, 0.6, … n. Dari pengujian ini, diperoleh tingkat akurasi terhadap nilai *threshold*. Tabel 3.18 menampilkan rancangan tabel yang akan digunakan untuk mencatat hasil dari pengujian ini.

Tabel 3.18 Uji pengaruh nilai Threshold terhadap tingkat akurasi.

Data latih	Nilai Threshold	Akurasi (%)
n data latih	0.5	
	0.55	
	0.6	
	0.65	
	0.7	

Keterangan :

n :Sejumlah data latih

Akurasi (%) :Tingkat akurasi yang dihitung dalam persen.

3.8 Uji Pengaruh Sebaran Data Terhadap Tingkat Akurasi

Uji pengaruh sebaran data terhadap tingkat akurasi dilakukan dengan melakukan pengujian dengan menggunakan jumlah data yang berbeda atau sama pada masing – masing kelas untuk dijadikan data latih. Tabel 3.19 menampilkan rancangan tabel yang akan digunakan untuk mencatat hasil dari pengujian ini.

Tabel 3.19 Uji pengaruh sebaran data terhadap tingkat akurasi.

Sebaran data	Akurasi EFk-NN (%)
n kelas 0 m kelas 1	
n kelas 0 m kelas 1	
n kelas 0 m kelas 1	
n kelas 0 m kelas 1	

Keterangan :

n :Data latih pada kelas 0.

m :Data latih pada kelas 1.

Akurasi (%) :Tingkat akurasi yang dihitung dalam persen, yang dihitung dengan persamaan 2.7.

3.9 Uji Perbandingan Hasil Akurasi Fk-NN dan EFk-NN

Uji perbandingan hasil akurasi Fk-NN dan EFk-NN dilakukan dengan banyak data latih dan data uji yang sama dan membandingkan hasil akurasi yang didapat dari perhitungan Fk-NN dengan EFk-NN. Tabel 3.20 menampilkan rancangan tabel yang akan digunakan untuk mencatat hasil dari pengujian ini.

Tabel 3.20 Uji perbandingan hasil akurasi EFk-NN dan Fk-NN.

Data latih	Akurasi (%)	
	Fk-NN	EFk-NN
n data latih		
n data latih		
n data latih		

Keterangan :

n :Sejumlah data latih

Akurasi (%) :Tingkat akurasi yang dihitung dalam persen, yang dihitung dengan persamaan 2.7.

BAB IV
IMPLEMENTASI

4.1 Lingkungan Implementasi

Dalam implementasi metode *Edited Fuzzy K-nearest Neighbor* (EFK-NN) untuk diagnosa penyakit Parkinson dibutuhkan beberapa aspek yang perlu diperhatikan yaitu segi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Dalam mengembangkan sistem dan penerapan metode penelitian ini digunakan beberapa komponen perangkat keras sebagai berikut :

1. Processor : Intel ® Core™ 2Duo CPU @2.10GHz.
2. Memory : 2.00 GB.
3. Hard disk : 300 GB.

4.1.2 Lingkungan Implementasi Perangkat Lunak

Pengembangan sistem dan penelitian ini dibutuhkan beberapa perangkat lunak yang digunakan sebagai berikut ini :

1. Sistem Operasi yang digunakan Windows 7 Ultimate 32bit.
2. Aplikasi pembangunan GUI dan code menggunakan NetBeans IDE 8.0.2.
3. Bahasa pemrograman yang dipakai yaitu bahasa pemrograman java.
4. Komponen java yang digunakan yaitu JDK 1.8.

4.2 Implementasi Program

Berdasarkan metode penelitian dan perancangan proses yang terdapat dalam bab 3, maka pada sub bab ini akan dijelaskan implementasi proses-proses tersebut.

4.2.1 Proses Baca File

Proses baca file digunakan untuk membaca file data yang akan digunakan dalam pelatihan dan pengujian. Tahapan proses baca file ditunjukan pada source code 4.1.

Proses Baca File

```
package edited_fknn;

import java.io.File;
import java.io.IOException;
import java.util.Arrays;
import java.util.Collections;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;
import org.apache.commons.lang3.ArrayUtils;

public class FormGui extends javax.swing.JFrame {

Workbook w;
Sheet sheet1, sheet2;
String lib = "E:/Skripsi/skripsiku/edited
FKNN/edited_fknn/Dataset/Dataset.xls";
Integer maxData, maxSehat, maxParkinson;
Object[][] isitabelDL, isitabelDU, isitabelHasil,
isiTabelEditedD1, isiTabelEditedD2;
String[] header = new String[8];
String[] headerHasil = new String[4];
Integer totalSehat, totalParkinson;
Integer[] randomS1, randomS2;
Integer totalDLT;
//variabel
Integer[] status, dlstatus, dustatus;
Double[] fo, dlfo, dufo, fhi, dlfhi, dufhi, flo, dlflo,
duflo, jitter, dljitter, dujitter, ppe, dlppe, duppe,
spread1, dlspread1, duspread1, spread2, dlspread2,
duspread2;
//nilai minimum & maksimum
Double maxFo, maxFhi, maxFlo, maxJitter, maxppe,
maxspread1, maxspread2;
Double minFo, minFhi, minFlo, minJitter, minppe,
minspread1, minspread2;
//variabel normalisasi
Double[] nfo, nfhi, nflo, njitter, nppe, nspread1,
nspread2;
//variabel euqclidean
Double[][] euq, euqTemp;
Integer[][] statusSort;
//variabel membership
Double[][] u;
//variabel hasil FKNN
Double[][] fknn, pembilang, penyebut;
//variabel hasil
Double[] nfknn;
Integer[] hasil;
```

```
//variable prediksi awal
String[] prediksiAwal;
//variabel edited Data Latih
Integer[] edlstatus, dlstatusTemp;
Double[] edlfo, edufo, dlfoTemp, edlfhi, edufhi,
dlfhiTemp, edlflo, eduflo, dlfloTemp, edljitter,
edujitter, dljitterTemp, edlppe, eduppe, dlppeTemp,
edlspread1, eduspread1, dlspread1Temp, edlspread2,
eduspread2, dlspread2Temp, edlfknn, dlfknnTemp;
//variabel akurasi
Double akurasi, acc, specifity, sensitivity;

public FormGui() {
initComponents();
kapasitasData();
}

public void kapasitasData() {
try {
w = jxl.Workbook.getWorkbook(new File(lib));
sheet1 = w.getSheet(0);
sheet2 = w.getSheet(1);
maxSehat = sheet1.getRows();
maxParkinson = sheet2.getRows();
maxData = maxSehat + maxParkinson;
} catch (Exception ex) {
System.out.println("Error " + ex.getMessage());
}
kapasitasSehat.setText("Max Sehat: " + (maxSehat - 1));
kapasitasParkinson.setText("Max Parkinson: " +
(maxParkinson - 1));
kapasitasData.setText("Total Data: " + maxData);
}

public static Integer[] acakIndex(Integer total) {
Integer[] arr = new Integer[total];
for (int i = 0; i < arr.length; i++) {
arr[i] = i;
}
Collections.shuffle(Arrays.asList(arr));
return arr;
}

Integer inputdlSehat, inputdlParkinson, inputduSehat,
inputduParkinson, inputK, totalDL, totalDU, totalData;

inputdlSehat = Integer.parseInt(dlSehat.getText());
inputdlParkinson =
Integer.parseInt(dlParkinson.getText());
totalDL = inputdlSehat + inputdlParkinson;

inputduSehat = Integer.parseInt(duSehat.getText());
```

```
inputduParkinson =
Integer.parseInt(duParkinson.getText());
totalDU = inputduSehat + inputduParkinson;

totalSehat = inputdlSehat + inputduSehat;
totalParkinson = inputdlParkinson + inputduParkinson;
totalData = totalSehat + totalParkinson;

inputK = Integer.parseInt(nilaiK.getText());

if (totalSehat > (maxSehat - 1) || totalParkinson >
(maxParkinson - 1)) {
    JOptionPane.showMessageDialog(null, "Nilai Input
melebihi batas!!");
} else if (inputK > totalDL) {
    JOptionPane.showMessageDialog(null, "Nilai K melebihi
batas!!");
} else {
    kapasitasSehatBaru.setText("Max : " + (maxSehat -
totalSehat));
    kapasitasParkinsonBaru.setText("Max : " +
(maxParkinson - totalParkinson));
    kapasitasDataBaru.setText("Sisa Data: " + (maxData -
totalData));

    //inisialisasi variabel data latih
    dlstatus = new Integer[totalDL];
    dlfo = new Double[totalDL];
    dlfhi = new Double[totalDL];
    dlflo = new Double[totalDL];
    dljitter = new Double[totalDL];
    dlppe = new Double[totalDL];
    dlspread1 = new Double[totalDL];
    dlspread2 = new Double[totalDL];

    //inisialisasi variabel data uji
    dustatus = new Integer[totalDU];
    dufo = new Double[totalDU];
    dufhi = new Double[totalDU];
    duflo = new Double[totalDU];
    dujitter = new Double[totalDU];
    duppe = new Double[totalDU];
    duspread1 = new Double[totalDU];
    duspread2 = new Double[totalDU];

    //index baris
    int iDL = 0, iDU = 0;
    randomS1 = acakIndex(maxSehat);
    //proses mengambil data latih sehat dari excel
    for (int baris = 0; baris < inputdlSehat; baris++) {
//status
```

```
dlstatus[iDL] = Integer.parseInt(sheet1.getCell(0,
randomS1[baris]).getContents());
//MDVP:Fo(Hz)
dlfo[iDL] = Double.parseDouble(sheet1.getCell(1,
randomS1[baris]).getContents().replace(',', '.'));
//MDVP:Fhi(Hz)
dlfhi[iDL] = Double.parseDouble(sheet1.getCell(2,
randomS1[baris]).getContents().replace(',', '.'));
//MDVP:Flo(Hz)
dlflo[iDL] = Double.parseDouble(sheet1.getCell(3,
randomS1[baris]).getContents().replace(',', '.'));
//MDVP:Jitter(%)
dljitter[iDL] = Double.parseDouble(sheet1.getCell(4,
randomS1[baris]).getContents().replace(',', '.'));
//PPE
dlppe[iDL] = Double.parseDouble(sheet1.getCell(5,
randomS1[baris]).getContents().replace(',', '.'));
//SPREAD1
dlspread1[iDL] = Double.parseDouble(sheet1.getCell(6,
randomS1[baris]).getContents().replace(',', '.'));
//SPREAD2
dlspread2[iDL] = Double.parseDouble(sheet1.getCell(7,
randomS1[baris]).getContents().replace(',', '.'));

iDL++;
    }
    //proses mengambil data uji sehat dari excel
    for (int baris = inputdlSehat; baris < totalSehat;
baris++) {
//status
dustatus[iDU] = Integer.parseInt(sheet1.getCell(0,
randomS1[baris]).getContents());
//MDVP:Fo(Hz)
dufo[iDU] = Double.parseDouble(sheet1.getCell(1,
randomS1[baris]).getContents().replace(',', '.'));
//MDVP:Fhi(Hz)
dufhi[iDU] = Double.parseDouble(sheet1.getCell(2,
randomS1[baris]).getContents().replace(',', '.'));
//MDVP:Flo(Hz)
duflo[iDU] = Double.parseDouble(sheet1.getCell(3,
randomS1[baris]).getContents().replace(',', '.'));
//MDVP:Jitter(%)
dujitter[iDU] = Double.parseDouble(sheet1.getCell(4,
randomS1[baris]).getContents().replace(',', '.'));
//PPE
duppe[iDU] = Double.parseDouble(sheet1.getCell(5,
randomS1[baris]).getContents().replace(',', '.'));
//SPREAD1
duspread1[iDU] = Double.parseDouble(sheet1.getCell(6,
randomS1[baris]).getContents().replace(',', '.'));
//SPREAD2
```

```
duspread2[iDU] = Double.parseDouble(sheet1.getCell(7,
randomS1[baris]).getContents().replace(',', '.'));

iDU++;
    }

    randomS2 = acakIndex(maxParkinson);
    //proses mengambil data uji dari excel
    for (int baris = 0; baris < inputdlParkinson;
baris++) {
//status
dlstatus[iDL] = Integer.parseInt(sheet2.getCell(0,
randomS2[baris]).getContents());
//MDVP:Fo(Hz)
dlfo[iDL] = Double.parseDouble(sheet2.getCell(1,
randomS2[baris]).getContents().replace(',', '.'));
//MDVP:Fhi(Hz)
dlfhi[iDL] = Double.parseDouble(sheet2.getCell(2,
randomS2[baris]).getContents().replace(',', '.'));
//MDVP:Flo(Hz)
dlflo[iDL] = Double.parseDouble(sheet2.getCell(3,
randomS2[baris]).getContents().replace(',', '.'));
//MDVP:Jitter(%)
dljitter[iDL] = Double.parseDouble(sheet2.getCell(4,
randomS2[baris]).getContents().replace(',', '.'));
//PPE
dlppe[iDL] = Double.parseDouble(sheet2.getCell(5,
randomS2[baris]).getContents().replace(',', '.'));
//SPREAD1
dlspread1[iDL] = Double.parseDouble(sheet2.getCell(6,
randomS2[baris]).getContents().replace(',', '.'));
//SPREAD2
dlspread2[iDL] = Double.parseDouble(sheet2.getCell(7,
randomS2[baris]).getContents().replace(',', '.'));
iDL++;
    }

    for (int baris = inputdlParkinson; baris <
totalParkinson; baris++) {
//status
dustatus[iDU] = Integer.parseInt(sheet2.getCell(0,
randomS2[baris]).getContents());
//MDVP:Fo(Hz)
dufo[iDU] = Double.parseDouble(sheet2.getCell(1,
randomS2[baris]).getContents().replace(',', '.'));
//MDVP:Fhi(Hz)
dufhi[iDU] = Double.parseDouble(sheet2.getCell(2,
randomS2[baris]).getContents().replace(',', '.'));
//MDVP:Flo(Hz)
duflo[iDU] = Double.parseDouble(sheet2.getCell(3,
randomS2[baris]).getContents().replace(',', '.'));
//MDVP:Jitter(%)
```

```
dujitter[iDU] = Double.parseDouble(sheet2.getCell(4,
randomS2[baris]).getContents().replace(',', '.'));
//PPE
duppe[iDU] = Double.parseDouble(sheet2.getCell(5,
randomS2[baris]).getContents().replace(',', '.'));
//SPREAD1
duspread1[iDU] = Double.parseDouble(sheet2.getCell(6,
randomS2[baris]).getContents().replace(',', '.'));
//SPREAD2
duspread2[iDU] = Double.parseDouble(sheet2.getCell(7,
randomS2[baris]).getContents().replace(',', '.'));
iDU++;
    }

    //set header
    header[0] = "STATUS";
    header[1] = "MDVP:Fo(Hz)";
    header[2] = "MDVP:Fhi(Hz)";
    header[3] = "MDVP:Flo(Hz)";
    header[4] = "MDVP:Jitter(%)";
    header[5] = "PPE";
    header[6] = "SPREAD1";
    header[7] = "SPREAD2";

    //proses menampilkan data latih
    isitabelDL = new Object[totalDL][8];
    for (int i = 0; i < iDL; i++) {
isitabelDL[i][0] = dlstatus[i];
isitabelDL[i][1] = dlfo[i];
isitabelDL[i][2] = dlfhi[i];
isitabelDL[i][3] = dlflo[i];
isitabelDL[i][4] = dljitter[i];
isitabelDL[i][5] = dlppe[i];
isitabelDL[i][6] = dlspread1[i];
isitabelDL[i][7] = dlspread2[i];
    }
    tabelD1.setModel(new DefaultTableModel(isitabelDL,
header));

    //proses menampilkan data uji
    isitabelDU = new Object[totalDU][8];
    for (int i = 0; i < iDU; i++) {
isitabelDU[i][0] = dustatus[i];
isitabelDU[i][1] = dufo[i];
isitabelDU[i][2] = dufhi[i];
isitabelDU[i][3] = duflo[i];
isitabelDU[i][4] = dujitter[i];
isitabelDU[i][5] = duppe[i];
isitabelDU[i][6] = duspread1[i];
isitabelDU[i][7] = duspread2[i];
    }
```

```
    tabelD2.setModel(new DefaultTableModel(isitabelDU,
header));

    //inisialisasi menggabungkan semua data variabel
    status = ArrayUtils.addAll(dlstatus, dustatus);
    fo = ArrayUtils.addAll(dlfo, dufo);
    fhi = ArrayUtils.addAll(dlfhi, dufhi);
    flo = ArrayUtils.addAll(dlflo, duflo);
    jitter = ArrayUtils.addAll(dljitter, dujitter);
    ppe = ArrayUtils.addAll(dlppe, duppe);
    spread1 = ArrayUtils.addAll(dlspread1, duspread1);
    spread2 = ArrayUtils.addAll(dlspread2, duspread2);

    //mendefinisikan nilai min max
    setMinMax();
    //mendefinisikan normalisasi
    setNormalization(totalData);
    //menghitung jarak euqclidean tiap data uji terhadap
data latih
    setEuqclidean(totalDL, totalDU);
    //mengurutkan data berdasarkan euclidean dari
terkecil
    sortEuq(totalDL, totalDU);
    //mendefinisikan nilai membership
    setMembership(inputdlSehat, inputdlParkinson);
    //mendefinisikan nilai pembilang pada fknn
    setPembilang(totalDL, totalDU, inputK);
    //mendefinisikan nilai penyebut pada fknn
    setPenyebut(totalDL, totalDU, inputK);
    //mendefinisikan nilai FKNN, hasil serta hasil
prediksi sementara sebelum input treshold
    setFKNN(totalDL, totalDU, inputK);

    headerHasil[0] = "STATUS (D2)";
    headerHasil[1] = "PERHITUNGAN";
    headerHasil[2] = "HASIL";
    headerHasil[3] = "PREDIKSI";

    isitabelHasil = new Object[totalDU][4];
    for (int i = 0; i < totalDU; i++) {
isitabelHasil[i][0] = status[i + totalDL];
isitabelHasil[i][1] = nfknn[i];
isitabelHasil[i][2] = hasil[i];
isitabelHasil[i][3] = prediksiAwal[i];
    }
    tabelHasil.setModel(new
DefaultTableModel(isitabelHasil, headerHasil));

    //reset tabel tab Edited FKNN
    isiTabelEditedD1 = null;
    tabelEditedD1.setModel(new
DefaultTableModel(isiTabelEditedD1, header));
```

```
    isiTabelEditedD2 = null;
    tabelEditedD2.setModel(new
DefaultTableModel(isiTabelEditedD2, header));

////proses menampilkan data latih baru
//int length = dlstatusTemp.length;
//isiTabelEditedD1 = new Object[length][8];
//for (int i = 0; i < length; i++) {
//    isiTabelEditedD1[i][0] = dlstatusTemp[i];
//    isiTabelEditedD1[i][1] = dlfoTemp[i];
//    isiTabelEditedD1[i][2] = dlfhiTemp[i];
//    isiTabelEditedD1[i][3] = dlfloTemp[i];
//    isiTabelEditedD1[i][4] = dljitterTemp[i];
//    isiTabelEditedD1[i][5] = dlppeTemp[i];
//    isiTabelEditedD1[i][6] = dlspread1Temp[i];
//    isiTabelEditedD1[i][7] = dlspread2Temp[i];
//}
//tabelEditedD1.setModel(new
DefaultTableModel(isiTabelEditedD1, header));
}
    }

    private void
prosesInputDataUjiBaruActionPerformed(java.awt.event.Acti
onEvent evt) {
Integer inputdlSehat, inputdlParkinson, inputduSehat,
inputduParkinson, inputK, totalDL, totalDU, totalData;
inputduSehat = Integer.parseInt(newduSehat.getText());
inputduParkinson =
Integer.parseInt(newduParkinson.getText());
inputK = Integer.parseInt(newnilaiK.getText());

if (inputduSehat > (maxSehat - totalSehat) ||
inputduParkinson > (maxParkinson - totalParkinson)) {
    JOptionPane.showMessageDialog(null, "Nilai input
melebihi batas!!");
} else if (inputK > totalDLT) {
    JOptionPane.showMessageDialog(null, "Nilai K melebihi
batas!!");
} else {
    totalDL = edlstatus.length;
    inputdlSehat = 0;
    inputdlParkinson = 0;
    for (int i = 0; i < totalDL; i++) {
if (edlstatus[i] == 0) {
    inputdlSehat++;
} else {
    inputdlParkinson++;
}
    }

    totalDU = inputduSehat + inputduParkinson;
```

```
    totalData = totalDL + totalDU;


    //inisialisasi variabel data uji
    dustatus = new Integer[totalDU];
    dufo = new Double[totalDU];
    dufhi = new Double[totalDU];
    duflo = new Double[totalDU];
    dujitter = new Double[totalDU];
    duppe = new Double[totalDU];
    duspread1 = new Double[totalDU];
    duspread2 = new Double[totalDU];

    //proses mengambil data uji sehat dari excel
    int iDU = 0;
    for (int baris = totalSehat; baris < inputduSehat +
totalSehat; baris++) {
//status
dustatus[iDU] = Integer.parseInt(sheet1.getCell(0,
randomS1[baris]).getContents());
//MDVP:Fo(Hz)
dufo[iDU] = Double.parseDouble(sheet1.getCell(1,
randomS1[baris]).getContents().replace(',', '.'));
//MDVP:Fhi(Hz)
dufhi[iDU] = Double.parseDouble(sheet1.getCell(2,
randomS1[baris]).getContents().replace(',', '.'));
//MDVP:Flo(Hz)
duflo[iDU] = Double.parseDouble(sheet1.getCell(3,
randomS1[baris]).getContents().replace(',', '.'));
//MDVP:Jitter(%)
dujitter[iDU] = Double.parseDouble(sheet1.getCell(4,
randomS1[baris]).getContents().replace(',', '.'));
//PPE
duppe[iDU] = Double.parseDouble(sheet1.getCell(5,
randomS1[baris]).getContents().replace(',', '.'));
//SPREAD1
duspread1[iDU] = Double.parseDouble(sheet1.getCell(6,
randomS1[baris]).getContents().replace(',', '.'));
//SPREAD2
duspread2[iDU] = Double.parseDouble(sheet1.getCell(7,
randomS1[baris]).getContents().replace(',', '.'));
iDU++;
    }

    for (int baris = totalParkinson; baris <
inputduParkinson + totalParkinson; baris++) {
//status
dustatus[iDU] = Integer.parseInt(sheet2.getCell(0,
randomS2[baris]).getContents());
//MDVP:Fo(Hz)
dufo[iDU] = Double.parseDouble(sheet2.getCell(1,
randomS2[baris]).getContents().replace(',', '.'));
```

```
//MDVP:Fhi(Hz)
dufhi[iDU] = Double.parseDouble(sheet2.getCell(2,
randomS2[baris]).getContents().replace(',', '.'));
//MDVP:Flo(Hz)
duflo[iDU] = Double.parseDouble(sheet2.getCell(3,
randomS2[baris]).getContents().replace(',', '.'));
//MDVP:Jitter(%)
dujitter[iDU] = Double.parseDouble(sheet2.getCell(4,
randomS2[baris]).getContents().replace(',', '.'));
//PPE
duppe[iDU] = Double.parseDouble(sheet2.getCell(5,
randomS2[baris]).getContents().replace(',', '.'));
//SPREAD1
duspread1[iDU] = Double.parseDouble(sheet2.getCell(6,
randomS2[baris]).getContents().replace(',', '.'));
//SPREAD2
duspread2[iDU] = Double.parseDouble(sheet2.getCell(7,
randomS2[baris]).getContents().replace(',', '.'));
iDU++;
    }

    isiTabelEditedD2 = new Object[totalDU][8];
    for (int i = 0; i < totalDU; i++) {
isiTabelEditedD2[i][0] = dustatus[i];
isiTabelEditedD2[i][1] = dufo[i];
isiTabelEditedD2[i][2] = dufhi[i];
isiTabelEditedD2[i][3] = duflo[i];
isiTabelEditedD2[i][4] = dujitter[i];
isiTabelEditedD2[i][5] = duppe[i];
isiTabelEditedD2[i][6] = duspread1[i];
isiTabelEditedD2[i][7] = duspread2[i];
    }
    tabelEditedD2.setModel(new
DefaultTableModel(isiTabelEditedD2, header));

    //inisialisasi menggabungkan semua data variabel
    status = ArrayUtils.addAll(edlstatus, dustatus);
    fo = ArrayUtils.addAll(edlfo, dufo);
    fhi = ArrayUtils.addAll(edlfhi, dufhi);
    flo = ArrayUtils.addAll(edlflo, duflo);
    jitter = ArrayUtils.addAll(edljitter, dujitter);
    ppe = ArrayUtils.addAll(edlppe, duppe);
    spread1 = ArrayUtils.addAll(edlspread1, duspread1);
    spread2 = ArrayUtils.addAll(edlspread2, duspread2);

    //mendefinisikan nilai min max
    setMinMax();
    //mendefinisikan normalisasi
    setNormalization(totalData);
    //menghitung jarak euqclidean tiap data uji terhadap
data latih
    setEuqclidean(totalDL, totalDU);
```

```
    //mengurutkan data berdasarkan euclidean dari
terkecil
    sortEuq(totalDL, totalDU);
    //mendefinisikan nilai membership
    setMembership(inputdlSehat, inpetdlParkinson);
    //mendefinisikan nilai pembilang pada fknn
    setPembilang(totalDL, totalDU, inputK);
    //mendefinisikan nilai penyebut pada fknn
    setPenyebut(totalDL, totalDU, inputK);
    //mendefinisikan nilai FKNN
    setEditedFKNN(totalDL, totalDU, inputK);
int totalBenar = 0;
prediksiAkhir = new String[totalDU];
for (int i = 0; i < totalDU; i++) {
    if (hasil[i] == status[i + totalDL]) {
        prediksiAkhir[i] = "BENAR";
    } else {
        prediksiAkhir[i] = "SALAH";
    }
}

headerAkhir[0] = "NO";
headerAkhir[1] = "STATUS (D2 Baru)";
headerAkhir[2] = "HASIL";
headerAkhir[3] = "PREDIKSI";

isiTabelHasilAkhir = new Object[totalDU][4];
for (int i = 0; i < totalDU; i++) {
    isiTabelHasilAkhir[i][0] = i + 1;
    isiTabelHasilAkhir[i][1] = status[i + totalDL];
    isiTabelHasilAkhir[i][2] = hasil[i];
    isiTabelHasilAkhir[i][3] = prediksiAkhir[i];
}
TabelHasilAkhir.setModel(new
DefaultTableModel(isiTabelHasilAkhir, headerAkhir));
    //mendefinisikan akurasi
    setAkurasi(totalDL, totalDU);
    //acc, sensitivity, specify
    setACC(totalDL, totalDU);

    tampilAkurasi.setText(String.format("%.2f", akurasi)
+ " %");
}
    }
```

Source Code 4.1 Baca File.

4.2.2 Proses Normalisasi Data

Proses normalisasi data digunakan untuk normalisasi data latih atau data uji untuk bisa digunakan pada sistem ketika melakukan perhitungan. Source code normalisasi data ditunjukan pada source code 4.2.

Proses Normalisasi data

```
public void setMinMax() {

minFo = Collections.min(Arrays.asList(fo));
minFhi = Collections.min(Arrays.asList(fhi));
minFlo = Collections.min(Arrays.asList(flo));
minJitter Collections.min(Arrays.asList(jitter));
minppe = Collections.min(Arrays.asList(ppe));
minspread1 =Collections.min(Arrays.asList(spread1));
minspread2 =Collections.min(Arrays.asList(spread2));
maxFo = Collections.max(Arrays.asList(fo));
maxFhi = Collections.max(Arrays.asList(fhi));
maxFlo = Collections.max(Arrays.asList(flo));
maxJitter = Collections.max(Arrays.asList(jitter));
maxppe = Collections.max(Arrays.asList(ppe));
maxspread1 =Collections.max(Arrays.asList(spread1));
maxspread2 =Collections.max(Arrays.asList(spread2));
//System.out.println("min FO" + minFo);
      }

public void setNormalization() {
//mengambil salah satu panjang array, karena panjang tiap
variabel pasti sama
int totalArray = fo.length;
//mendefinisikan panjang array normalisasi
nfo = new Double[totalArray];
nfhi = new Double[totalArray];
nflo = new Double[totalArray];
njitter = new Double[totalArray];
nppe = new Double[totalArray];
nspread1 = new Double[totalArray];
nspread2 = new Double[totalArray];

//proses menghitung normalisasi
for (int i = 0; i < totalArray; i++) {
nfo[i] = (fo[i] - minFo) / (maxFo - minFo);
nfhi[i] = (fhi[i] - minFhi) / (maxFhi - minFhi);
nflo[i] = (flo[i] - minFlo) / (maxFlo - minFlo);
njitter[i] = (jitter[i] - minJitter) / (maxJitter -
minJitter);
nppe[i] = (ppe[i] - minppe) / (maxppe - minppe);
nspread1[i] = (spread1[i] - minspread1) / (maxspread1 -
minspread1);
nspread2[i] = (spread2[i] - minspread2) / (maxspread2 -
minspread2);
//  System.out.println("normalisasi fo "+nfo[i]);
```

```
}

}
```

Source Code 4.2 Normalisasi Data.

4.2.3 Proses KNN

Proses KNN digunakan untuk menghitung nilai jarak kedekatan tetangga data uji terhadap data latih menggunakan *Euclidean Distance*. Source code KNN ditunjukan pada source code 4.3 berikut ini.

Proses KNN

```
public void setEuqclidean(int totalDL, int totalDU) {
euq = new Double[totalDU][totalDL];
for (int i = 0; i < totalDU; i++) {
    for (int j = 0; j < totalDL; j++) {
euq[i][j] = Math.sqrt((Math.pow((nfo[i + totalDL] -
nfo[j]), 2))
+ (Math.pow((nfhi[i + totalDL] - nfhi[j]), 2))
+ (Math.pow((nflo[i + totalDL] - nflo[j]), 2))
+ (Math.pow((njitter[i + totalDL] - njitter[j]), 2))
+ (Math.pow((nppe[i + totalDL] - nppe[j]), 2))
+ (Math.pow((nspread1[i + totalDL] - nspread1[j]), 2))
+ (Math.pow((nspread2[i + totalDL] - nspread2[j]), 2))
);
    }
//    System.out.println("Euclidean " +
Arrays.toString(euq[i]));
}
    }

    public void sortEuq(int totalDL, int totalDU) {
euqTemp = new Double[totalDU][totalDL];
//meng-copy ke variabel sementara
for (int i = 0; i < totalDU; i++) {
    euqTemp[i] = euq[i].clone();
}

//pengurutan
for (int i = 0; i < totalDU; i++) {
    Arrays.sort(euq[i]);
}

statusSort = new Integer[totalDU][totalDL];

//nilai euclidean setelah di order akan dicocokkan, untuk
mencari index
for (int i = 0; i < totalDU; i++) {
    for (int j = 0; j < totalDL; j++) {
```

```
for (int k = 0; k < totalDL; k++) {
    if (euq[i][j] == euqTemp[i][k]) {
statusSort[i][j] = status[k];
    }
}
    }
```

Source Code 4.3 KNN.

4.2.4 Proses Fk-NN

Proses Fk-NN merupakan proses perhitungan nilai membership dengan 2 kelas (pasien sehat dan pasien parkinson). Source code FKNN dapat dilihat pada source code 4.4.

Proses FKNN

```
public void setMembership(Integer inputdlSehat, Integer
inputdlParkinson) {
//nilai membership
u = new Double[2][2];
u[0][0] = 0.51 + ((inputdlSehat.doubleValue() /
(inputdlSehat.doubleValue() +
inputdlParkinson.doubleValue())) * 0.49);
u[0][1] = (inputdlSehat.doubleValue() /
(inputdlSehat.doubleValue() +
inputdlParkinson.doubleValue())) * 0.49;
u[1][0] = (inputdlParkinson.doubleValue() /
(inputdlSehat.doubleValue() +
inputdlParkinson.doubleValue())) * 0.49;
u[1][1] = 0.51 + ((inputdlParkinson.doubleValue() /
(inputdlSehat.doubleValue() +
inputdlParkinson.doubleValue())) * 0.49);
//System.out.println("U0 " + Arrays.toString(u[0]));
//System.out.println("U1 " + Arrays.toString(u[1]));
    }

    public void setPembilang(int totalDL, int totalDU, int
inputK) {
pembilang = new Double[totalDU][2];
for (int i = 0; i < totalDU; i++) {
    for (int j = 0; j < inputK; j++) {
if (pembilang[i][0] == null && pembilang[i][1] == null) {
    pembilang[i][0] = 0.00;
    pembilang[i][1] = 0.00;
}
if (statusSort[i][j] == 0) {
    pembilang[i][0] += u[0][0] * (1 / (Math.pow(euq[i][j],
2)));
    pembilang[i][1] += u[0][1] * (1 / (Math.pow(euq[i][j],
2)));
} else {
    pembilang[i][0] += u[1][0] * (1 / (Math.pow(euq[i][j],
2)));
```

```
    pembilang[i][1] += u[1][1] * (1 / (Math.pow(euq[i][j],
2)));
}
    }
//System.out.println("pembilang " + i + " " +
Arrays.toString(pembilang[i]));
}
    }

    public void setPenyebut(int totalDL, int totalDU, int
inputK) {
penyebut = new Double[totalDU][1];
for (int i = 0; i < totalDU; i++) {
    for (int j = 0; j < inputK; j++) {
if (penyebut[i][0] == null) {
    penyebut[i][0] = 0.00;
}
penyebut[i][0] += 1 / (Math.pow(euq[i][j], 2));
    }
//    System.out.println("penyebut "+i+"
"+Arrays.toString(penyebut[i]));
}
    }

    public void setFKNN(int totalDL, int totalDU, int
inputK) {
fknn = new Double[totalDU][2];
nfknn = new Double[totalDU];
hasil = new Integer[totalDU];
for (int i = 0; i < totalDU; i++) {
    for (int j = 0; j < inputK; j++) {
if (fknn[i][0] == null && fknn[i][1] == null) {
    fknn[i][0] = 0.00;
    fknn[i][1] = 0.00;
}
fknn[i][0] = pembilang[i][0] / penyebut[i][0];
fknn[i][1] = pembilang[i][1] / penyebut[i][0];
if (fknn[i][0] > fknn[i][1]) {
    hasil[i] = 0;
    nfknn[i] = Math.round(fknn[i][0] * 1000.0) / 1000.0;
} else {
    hasil[i] = 1;
    nfknn[i] = Math.round(fknn[i][1] * 1000.0) / 1000.0;
}
    }
//    System.out.println("fknn " + i + " " +
Arrays.toString(fknn[i]));
}
//System.out.println("Hasil " + Arrays.toString(hasil));

//variabel sementara untuk menyimpan total benar
int totalBenar = 0;
```

```
prediksiAwal = new String[totalDU];
for (int i = 0; i < totalDU; i++) {
    if (hasil[i] == status[i + totalDL]) {
prediksiAwal[i] = "BENAR";
totalBenar++;
    } else {
prediksiAwal[i] = "SALAH";
    }
}
//System.out.println("Prediksi Awal " +
Arrays.toString(prediksiAwal));

dlstatusTemp = new Integer[totalBenar];
dlfoTemp = new Double[totalBenar];
dlfhiTemp = new Double[totalBenar];
dlfloTemp = new Double[totalBenar];
dljitterTemp = new Double[totalBenar];
dlppeTemp = new Double[totalBenar];
dlspread1Temp = new Double[totalBenar];
dlspread2Temp = new Double[totalBenar];
dlfknnTemp = new Double[totalBenar];

//variabel untuk menyimpan index
int iE = 0;
for (int i = 0; i < totalDU; i++) {
    if (prediksiAwal[i] == "BENAR") {
dlstatusTemp[iE] = dustatus[i];
dlfoTemp[iE] = dufo[i];
dlfhiTemp[iE] = dufhi[i];
dlfloTemp[iE] = duflo[i];
dljitterTemp[iE] = dujitter[i];
dlppeTemp[iE] = duppe[i];
dlspread1Temp[iE] = duspread1[i];
dlspread2Temp[iE] = duspread2[i];
dlfknnTemp[iE] = nfknn[i];
iE++;
    }
}
//System.out.println("Status " +
Arrays.toString(dlstatusTemp));
        }
```

Source code 4.4 FKNN.

4.2.5 Proses *Threshold*

Proses Threshold merupakan proses perhitungan nilai derajat keanggotaan Fk-NN yang berada diatas *threshold* yang akan dilanjutkan sebagai data latih baru pada proses setelahnya. Source code *threshold* dapat dilihat pada source code 4.5.

Proses *Threshold*

```
private void
tresholdProsesActionPerformed(java.awt.event.ActionEvent
evt) {
Double inputTreshold;
inputTreshold =
Double.parseDouble(nilaiTreshold.getText());
if (inputTreshold < 0 || inputTreshold > 1) {
    JOptionPane.showMessageDialog(null, "Nilai Treshold
melebihi batas!!");
} else {
    //reset tabel D2
    isiTabelEditedD2 = null;
    tabelEditedD2.setModel(new
DefaultTableModel(isiTabelEditedD2, header));

    int length = dlstatusTemp.length;
    //variabel definisi data latih setelah treshold
    totalDLT = 0;
    for (int i = 0; i < length; i++) {
if (dlfknnTemp[i] > inputTreshold) {
    totalDLT++;
}
    }

    kapasitasK.setText("Max : " + (totalDLT));

    edlstatus = new Integer[totalDLT];
    edlfo = new Double[totalDLT];
    edlfhi = new Double[totalDLT];
    edlflo = new Double[totalDLT];
    edljitter = new Double[totalDLT];
    edlppe = new Double[totalDLT];
    edlspread1 = new Double[totalDLT];
    edlspread2 = new Double[totalDLT];

    int iT = 0;
    for (int i = 0; i < length; i++) {
if (dlfknnTemp[i] > inputTreshold) {
    edlstatus[iT] = dlstatusTemp[i];
    edlfo[iT] = dlfoTemp[i];
    edlfhi[iT] = dlfhiTemp[i];
    edlflo[iT] = dlfloTemp[i];
    edljitter[iT] = dljitterTemp[i];
    edlppe[iT] = dlppeTemp[i];
    edlspread1[iT] = dlspread1Temp[i];
    edlspread2[iT] = dlspread2Temp[i];
    iT++;
}
    }

    isiTabelEditedD1 = new Object[iT][8];
```

```
    for (int i = 0; i < iT; i++) {
isiTabelEditedD1[i][0] = edlstatus[i];
isiTabelEditedD1[i][1] = edlfo[i];
isiTabelEditedD1[i][2] = edlfhi[i];
isiTabelEditedD1[i][3] = edlflo[i];
isiTabelEditedD1[i][4] = edljitter[i];
isiTabelEditedD1[i][5] = edlppe[i];
isiTabelEditedD1[i][6] = edlspread1[i];
isiTabelEditedD1[i][7] = edlspread2[i];
    }
    tabelEditedD1.setModel(new
DefaultTableModel(isiTabelEditedD1, header));
}
    }
```

Source code 4.5 *Threshold.*

4.2.6 Proses EFk-NN

Proses EFk-NN merupakan proses perhitungan nilai membership dengan 2 kelas (pasien sehat dan pasien parkinson) yang telah melalui proses *threshold.* Source code EFk-NN dapat dilihat pada source code 4.6.

Proses EFk-NN

```
public void setEditedFKNN(int totalDL, int totalDU, int
inputK) {
fknn = new Double[totalDU][2];
hasil = new Integer[totalDU];
for (int i = 0; i < totalDU; i++) {
    for (int j = 0; j < inputK; j++) {
if (fknn[i][0] == null && fknn[i][1] == null) {
    fknn[i][0] = 0.00;
    fknn[i][1] = 0.00;
}
fknn[i][0] = pembilang[i][0] / penyebut[i][0];
fknn[i][1] = pembilang[i][1] / penyebut[i][0];
if (fknn[i][0] > fknn[i][1]) {
    hasil[i] = 0;
} else {
    hasil[i] = 1;
}
    }
//    System.out.println("fknn " + i + " " +
Arrays.toString(fknn[i]));
}
//System.out.println("Hasil " + Arrays.toString(hasil));
    }

    public void setAkurasi(int totalDL, int totalDU) {
Integer total = totalDU;
```

```
Integer tempAkurasi = 0;
for (int i = 0; i < totalDU; i++) {
    if (hasil[i] == status[i + totalDL]) {
tempAkurasi++;
    }
}
akurasi = (tempAkurasi.doubleValue() /
total.doubleValue()) * 100;
System.out.println("Akurasi = " + String.format("%.2f",
akurasi) + " %");
    }
```

Source code 4.5 EFk-NN.

BAB V
PENGUJIAN DAN ANALISIS

Pada bab ini akan dilakukan pengujian dan analisis dari penerapan *Edited Fuzzy k-Nearest Neighbor* (EFk-NN) untuk diagnosa penyakit parkinson berdasarkan *Parkinson Dataset* yang diperoleh dari url : *https://archive.ics.uci.edu/ml/datasets/Parkinsons*.

5.1 Pengujian

Pengujian dilakukan untuk mengetahui hasil akurasi dari implementasi yang dilakukan. Pengujian dilakukan dengan 3 macam jenis pengujian yaitu pengujian tingkat akurasi terhadap nilai *threshold*, pengujian tingkat akurasi terhadap sebaran data dan pengujian perbandingan tingkat akurasi dengan metode *Fuzzy k-Nearest Neighbor*.

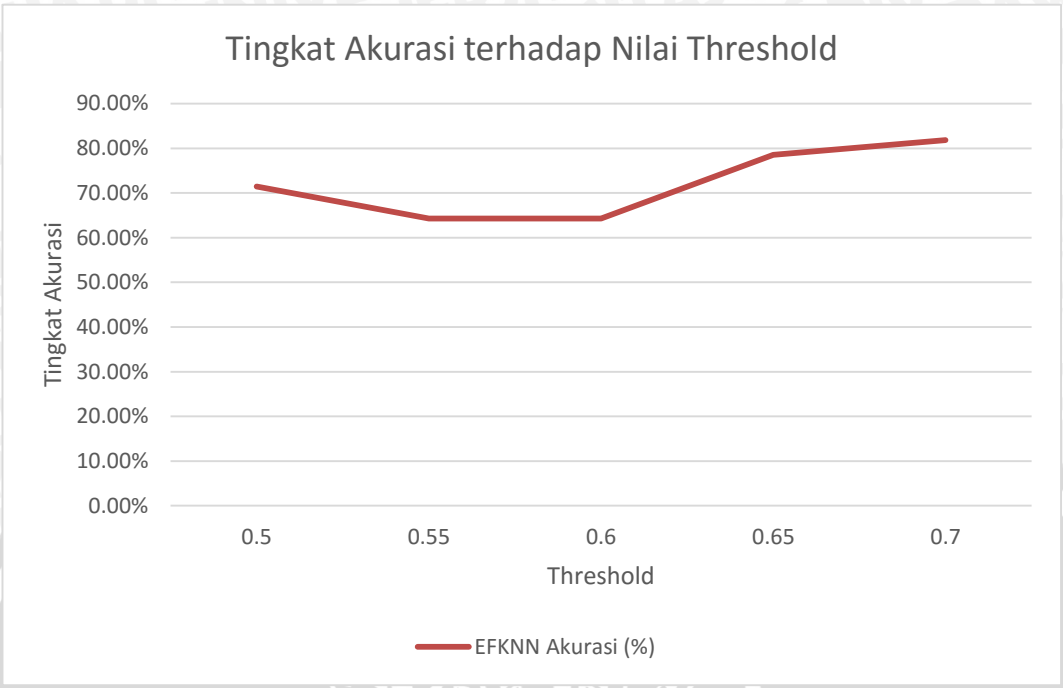
Pada pengujian ini, D1 merupakan data latih awal pada proses EFk-NN, D2 merupakan data uji awal pada proses EFk-NN yang akan dijadikan data latih baru pada proses selanjutnya, k adalah jarak tetangga terdekat dan "data uji" merupakan data uji akhir yang digunakan.

5.1.1 Pengujian tingkat akurasi terhadap nilai *threshold*

Pada pengujian tingkat akurasi terhadap pengaruh nilai *threshold*, D1 yang digunakan adalah 60 data, D2 yang digunakan adalah 20 data, k = 5 dan data uji yang digunakan adalah 14 data dan k = 5. Untuk nilai *threshold* yang digunakan adalah 0,5 sampai dengan 0,7. Hasil dari pengujian tingkat akurasi terhadap nilai *threshold* dapat dilihat pada tabel 5.1.

Tabel (5.1) Hasil uji terhadap nilai *threshold*.

Threshold	EFKNN Akurasi (%)
0.5	71.43 %
0.55	64.29 %
0.6	64.29 %
0.65	78.57 %
0.7	81.82 %



Gambar 5.1 Grafik Tingkat Akurasi terhadap Nilai *Threshold*.

Pada gambar grafik diatas menunjukan analisa dari pengujian dan analisis dari penerapan *Edited Fuzzy k-Nearest Neighbor* (Fk-NN) untuk diagnosa penderita Parkinson berdasarkan *Parkinson Dataset*. Berdasarkan grafik diatas maka dapat disumpulkan bahwa semakin tinggi nilai *threshold* maka menghasilkan tingkat akurasi yang lebih tinggi karena objek yang dipilih adalah objek yang memiliki derajat keanggotaan yang tinggi (diatas *threshold*).

5.1.2 Pengujian tingkat akurasi terhadap sebaran data

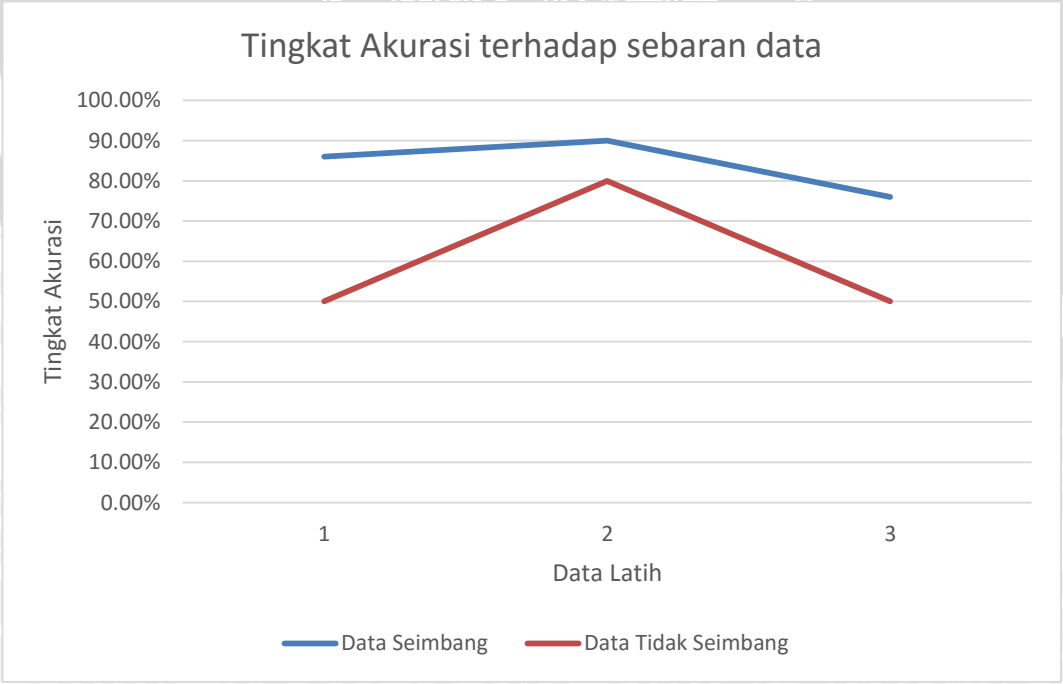
Pada pengujian tingkat akurasi terhadap sebaran data, data latih yang digunakan meliputi beberapa pasien sehat dan parkinson. Sebaran 1 terdiri dari D1 = 40 (kelas 0 = 30 dan kelas 1 = 10) . Sebaran data 2 terdiri dari D1 = 50 (kelas 0 = 30 dan kelas 1 = 20. Sebaran data 3 terdiri dari D1 =60 (kelas 0 = 30 dan kelas 1 = 30). Sebaran data 4 terdiri dari D1 = 50 (kelas 0 = 20 dan kelas 1 = 30). Sebaran data 5 terdiri dari D1 = 40 (kelas 0 =10 dan kelas 1 = 30). Sebaran data 6 terdiri dari D1 = 40 (kelas 0 = 20 dan kelas 1 = 20). Sebaran data 7 terdiri dari D1 = 20 (kelas 0 = 10 dan kelas 1 = 10). Pada semua sebaran data tersebut menggunakan D2 = 10

(kelas 0 = 5 dan kelas 1 = 5), k =5 dan *threshold* = 0.7. Hasil pada setiap sebaran data dilakukan dengan pengujian sebanyak 5 kali dan diambil rata – rata nya. Tabel 5.2 berikut ini menunjukan pengaruh sebaran data terhadap tingkat akurasi.

Tabel (5.2) Hasil akurasi terhadap sebaran data.

Percobaan	Sebaran data tidak imbang	Akurasi EFk-NN (%)
1	30 kelas 0 10 kelas 1	50 %
2	30 kelas 0 20 kelas 1	80 %
3	10 kelas 0 30 kelas 1	50 %

Percobaan	Sebaran data imbang	Akurasi EFk-NN(%)
1	30 kelas 0 30 kelas 1	86 %
2	20 kelas 0 20 kelas 1	90 %
3	10 kelas 0 10 kelas 1	76 %



Gambar 5.2 Grafik Tingkat Akurasi terhadap Sebaran data.

Berdasarkan gambar grafik diatas, terdapat 3 data seimbang dan 3 data tidak seimbang. Pada data seimbang didapat akurasi terbaik sebesar 90% dan pada data

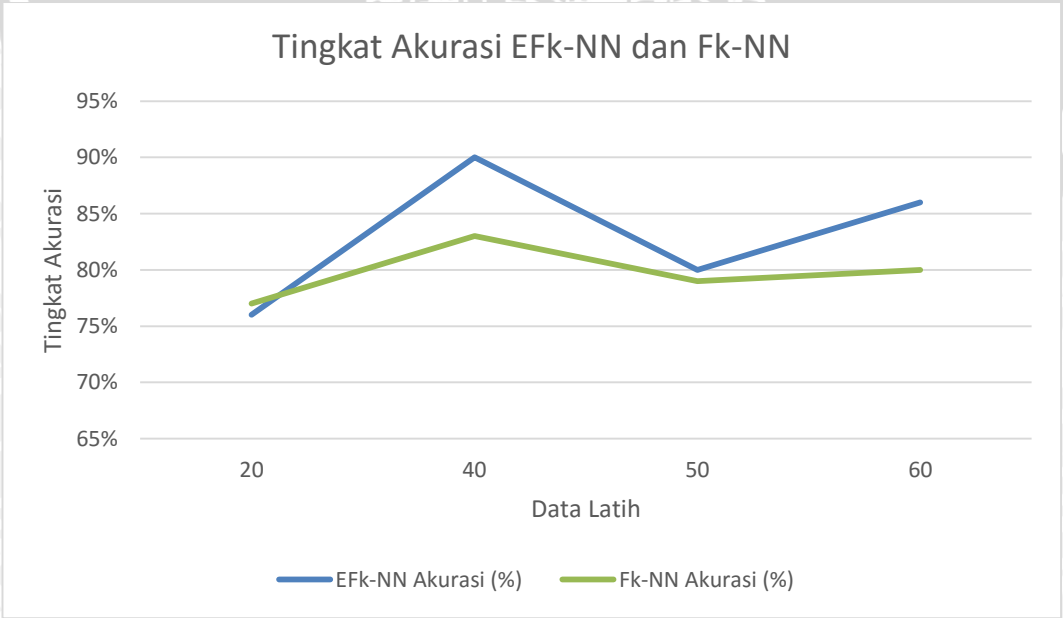
tida seimbang didapat akurasi terbaik sebesar 80%. Dapat disimpulkan bahwa pada data yang seimbang, didapat akurasi yang baik dengan menggunakan data uji, nilai k dan *threshold* yang sama daripada data yang dominan pada kelas tertentu. Karena, data yang lebih dominan pada suatu kelas tertentu akan memberikan pengaruh lebih dalam menentukan label kelas itu sendiri.

5.1.3 Pengujian perbandingan tingkat akurasi dengan metode *Fuzzy k-Nearest Neighbor*

Pada pengujian perbandingan tingkat akurasi dengan metode *fuzzy k-nearest neighbour*, data latih yang digunakan adalah 20, 40, 50 dan 60 data. Tabel 5.3 berikut ini menunjukan perbandingan akurasi metode EFk-NN dengan Fk-NN menggunakan D2 yang sama yaitu 20 data (kelas 0 = 10 dan kelas 1 = 10), nilai k = 5 dan threshold = 0.7.

Tabel (5.3) Hasil perbandingan metode EFk-NN dengan Fk-NN.

Data Latih	EFk-NN Akurasi (%)	Fk-NN Akurasi (%)
20	76 %	77 %
40	90 %	83 %
50	80 %	79 %
60	86 %	80 %



Gambar 5.3 Grafik Tingkat Akurasi EFk-NN dan Fk-NN.

Berdasarkan grafik diatas, terdapat 3 akurasi EFk-NN yang lebih bagus daripada Fk-NN dan 1 akurasi Fk-NN yang lebih bagus dari EFk-NN. Akurasi terbaik yang didapat Fk-NN adalah 83% dan EFk-NN adalah 90%. Dapat disimpulkan bahwa metode EFk-NN mendapatkan akurasi yang lebih bagus daripada Fk-NN. Karena pada proses EFk-NN, untuk mendapatkan hasil akurasi harus melewati 2 kali proses Fk-NN dan pada proses Fk-NN ke-2, data latih yang digunakan adalah data uji yang nilai derajat keanggotaannya tinggi (diatas threshold) pada proses Fk-NN pertama.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil penelitian untuk mengetahui tingkat akurasi dari hasil klasifikasi pada *Parkinson Dataset* dengan menggunakan metode *Edited Fuzzy k-Nearest Neighbor* (EFk-NN), dapat disimpulkan bahwa:

1. Metode *Edited Fuzzy k-Nearest Neighbor* (EFk-NN) bisa diterapkan untuk mendiagnosis penderita *Parkinson* dengan menggunakan 8 parameter yang terdapat pada data *Parkinson Dataset*. Langkah pertama adalah melakukan normalisasi pada data latih dan data uji dengan menggunakan *min-max normalization*. Langkah selanjutnya data latih dan data uji yang telah dinormalisasi dihitung menggunakan persamaan *euclidean distance*. Setelah didapat *euclidean distance*, diteruskan dengan perhitungan Fk-NN untuk mendapatkan nilai derajat keanggotaannya. Kemudian nilai tersebut di *Threshold* untuk menjadikan data uji awal menjadi data latih baru. Kemudian proses Fk-NN lagi untuk mendapatkan hasil akhir. Pada akhirnya didapatkan hasil diagnosa penderita *parkinson* yang dihitung menggunakan persamaan *Edited Fuzzy k-Nearest Neighbor* (EFk-NN).
2. Berdasarkan pengujian pada penelitian ini, didapatkan akurasi sebesar 81,82% jika menggunakan threshold 0,7 dan data latih sebanyak 60. Akurasi sebesar 90% didapat pada sebaran data 20 *record* kelas 0 dan 20 *record* kelas 1.
3. Dari hasil pengujian perbandingan Fk-NN dan EFk-NN diperoleh hasil bahwa EFk-NN lebih baik dengan akurasi sebesar 90%, sedangkan Fk-NN hanya 86%.

6.2 Saran

Beberapa saran yang dapat dijadikan bahan pertimbangan untuk penelitian selanjutnya adalah sebagai berikut:

1. Diharapkan pada penelitian selanjutnya menggunakan data yang lebih banyak untuk membantu dalam perhitungan akurasi.
2. Diharapkan pada penelitian berikutnya menggunakan metode lain agar mendapat hasil yang lebih maksimal.

DAFTAR PUSTAKA

[WIJ-14] Wijaya, Hardika Teguh. 2014. "Penerapan Fuzzy K-Nearest Neighbor (Fknn) Untuk Diagnosa Penderita Liver Berdasarkan Indian Liver Patient Dataset (ILPD)". Universitas Brawijaya. Malang

[KUS-09] Kusrini dan Luthfi, Emha Tufiq. 2009. "*Algoritma Data Mining*". Yogyakarta: Penerbit ANDI.

[SAN-04] Santoso, Harip. 2004. "VB.NET untuk .NET Programmer". Jakarta. PT Gramedia

[JOHN-07] F.Johnson. 2007. "Medical Speech – Language Pathology". New Yotk. Thieme Medical Publisher Inc.

[NIL-96] J.Nilsson, Nils."Introduction to Machine Learning". 1996. Standford University: Stanford, CA 94305

[PRI-05] Priyono, Agus Priyono, Muhammad Ridwan, Ahmad Jais Alias, Riza Atiq, O. K. Rahmat, Azmi Hassan & Mohd. Alauddin Mohd. Ali. 2005. "*Generation Of Fuzzy Rules With Subtractive Clustering*", Jurnal Teknologi, 43(D) Dis. Universitas Teknologi Malaysia.

[ZHA-09] Zhang, Juan, Yi Niu, Huabei Nie. 2009. "*Web Document Classification Based on Fuzzy KNN Algorithm*". International Conference on Computational Intelligence and Security.

[KEL-85] Keller, M. James, Michael R Gray, James A. Givens. 1985. "*A Fuzzy K-Nearest Neighbor*". IEEE Transactions on System, Man and Cybernetics, Vol. SMC-15 No. 4

[REV-13] Reviangga Dika Satyatama. 2013. "Klasifikasi Incomplete Data Penyakit Liver Pada Manusia Dengan Menggunakan Algoritma Voting Feature Interval-5 (Vfi5)". Universitas Brawijaya. Malang.

[DEW-07] Dewanto, George, Suwono J. Wita, Riyanto, Budi, Turana, Yuda. 2007. "Panduan Praktis Diagnosis & Tata Laksana Pneyakit Saraf". Jakarta. Penerbit Buku Kedokteran EGC.

[KUS-03] Kusumadewi, S dan Purnomo, H. 2003. "Artificial Intelligence (Teknik & Aplikasinya)". Graha Ilmu. Jogjakarta.

[KUS-10] Kusumadewi, S dan Purnomo, H. 2010. "Aplikasi Logika Fuzzy untuk pendukung Keputusan: Jilid 2". Graha Ilmu. Yogyakarta.

[TUR-05] Turban, 2005. "Decision Support Sistems and Intelligent Sistems (Sistem Pendukung Keputusan dan Sistem Cerdas) jilid 1". Yogyakarta. Andi Offset

[KHU-07] Khusnawi. 2007. "Pengantar Solusi Data Mining (Online)". STMIK AMIKOM.Yogyakarta.*http://p3m.amikom.ac.id/p3m/56%20%20PENGANTAR%20SOLUSI%20DATA%20MINING.pdf* diakses tanggal 27 November 2013.

[LAR-05] Larose, Daniel T. 2005. "*Discovering Knowledge in Data. An Introduction to Data Mining*". John Willey dan Sons. New Jersey.

[KAN-03] Kantardzic, Mehmed. 2003. "*Data Mining : Concepts, Models, Methods and Algorithm. John Wiley & Sons"*. New York.

[MOR-09] Moradian, Mehdi dan Baarani, Ahmad. 2009. "*KNNBA: k-Nearest-Neighbor-Based_Associaton Algorithm* (Online)", *http://www.jatit.org/volume/researchpapers/Vol6No1/14Vol6No1.pdf*, diakses tanggal 9 Juni 2015.

[JAY-11] Jayalakshmi, T. dan Santhakumaran, A. 2011. *Statistical Normalization and Backpropagation for Classification*, *http://www.ijcte.org/papers/288-L052.pdf*, diakses tanggal 9 Juni 2015).

[HAM-08] Hamid parvin, Hosein Alizadeh and Behrouz Minaei-Bidgoli. 2008. "*MKNN: Modified K-Nearest Neighbor. Proceedings of the World Congress on Engineering and Computer Science"*. San Fransisco.

[YAN-13] Yanita C. 2013. "Penerapan Fuzzy K-Nearest Neighbor untuk Menentukan Status Evaluasi Kinerja Karyawan". Universitas Brawijaya. Malang

[NUG-06] Nugraha, Dany, dkk. 2006. "Diagnosis Gangguan Sistem Urinari pada Anjing dan Kucing Menggunakan VFI 5". Institute Pertanian Bogor.

[ZHA-13] Zhang, Cao, dkk. 2013. "*A Method Based on the Edited FKNN by the Threshold Value*". China.