KLASIFIKASI PENYAKIT GINJAL KRONIS BERDASARKAN *CHRONIC KIDNEY DISEASE DATASET* MENGGUNAKAN METODE *FUZZY K-NEAREST NEIGHBOR (FK-NN)*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Septyan Teguh Mahendra
NIM: 0910960017

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

KLASIFIKASI PENYAKIT GINJAL KRONIS BERDASARKAN *CHRONIC KIDNEY DISEASE DATASET* MENGGUNAKAN METODE *FUZZY K-NEAREST NEIGHBOR (FK-NN)*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Septyan Teguh Mahendra
NIM: 0910960017

Skripsi ini telah diuji dan dinyatakan lulus pada
23 Agustus 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
M. Tanzil Furqon, S.Kom, M.CompSc	Randy Cahya W, S.ST.,M.kom
NIP: 19820930 200801 1 004	NIK: 201405 880206 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

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 disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam 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, 11 Agustus 2016

Septyan Teguh Mahandra

NIM: 0910960017

KATA PENGANTAR

Puji Syukur ke hadirat Allah SWT atas rahmat dan hidayah-Nya penulis mampu menyelesaikan skripsi yang merupakan salah satu persyaratan akademis untuk menyelesaikan studi program sarjana di Fakultas Ilmu Komputer Universitas Brawijaya dengan judul "Klasifikasi Penyakit Ginjal Kronis Berdasarkan *Chronic Kidney Disease Dataset Menggunakan Metode Fuzzy k-Nearest Neighbor (Fk-NN)*". Tidak lupa penulis mengucapkan terima kasih pada semua pihak yang telah membantu dalam penyelesaian skripsi, terutama kepada:

1. Bapak M. Tanzil Furqon, S.Kom, M.CompSc selaku dosen pembimbing I dan Bapak Randy Cahya W, S.ST.,M.kom selaku dosen pembimbing II yang telah membimbing penulis dalam menyelesaikan skripsi ini dari awal hingga akhir pengerjaan, serta segenap staff pengajar dan karyawan Fakultas Ilmu Komputer yang telah mendukung kelancaran proses perkuliahan secara keseluruhan.
2. Segenap anggota keluarga penulis, yang selalu memberikan dukungan untuk penulis.
3. Sekar Sayekti Luktriutami sebagai kekasih yang selalu setia mendampingi dan memberikan motivasi pada penulis untuk menyelesaikan skripsi ini.
4. Seluruh teman-teman penulis yang telah memberikan saran-saran dan berbagi pengalaman dalam menyelesaikan skripsi.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Oleh karena itu penulis sangat mengharapkan kritik dan saran yang membangun agar pengembangan dan penulisan selanjutnya menjadi lebih baik.

Malang, 11 Agustus 2016

Penulis

hendrahuget@ymail.com

ABSTRAK

Septyan Teguh Mahendra. 2016. Klasifikasi Penyakit Ginjal Kronis Berdasarkan *Chronic Kidney Disease Dataset* Menggunakan Metode *Fuzzy k-Nearest Neighbor (Fk-NN).* Program Studi Informatika/Ilmu Komputer. Jurusan Teknik Informatika. Fakultas Ilmu Komputer. Universitas Brawijaya. Malang. Pembimbing: M. Tanzil Furqon, S.Kom, M.CompSc dan Randy Cahya W, S.ST., M.kom.

Penelitian ini membahas tentang pengimplementasian algoritma *Fuzzy k-Nearest Neighbor* untuk melakukan pengklasifikasian penyakit ginjal kronis. Data yang digunakan dalam pembuatan sistem ini adalah dataset *chronic kidney disease* yang didapatkan dari *UCI Machine Learning Repository*. *Fuzzy k-Nearest Neighbor* (*Fk-NN*) merupakan metode klasifikasi yang menggabungkan teknik *Fuzzy* dengan *k-Nearest Neighbor classifier*. Dalam penelitian ini dilakukan beberapa pengujian untuk mengetahui tingkat akurasi sistem, pengujian yang pertama adalah pengaruh jumlah data latih terhadap hasil akurasi sistem, dan pengujian yang kedua adalah pengaruh nilai *k* terhadap hasil akurasi sistem. Pada pengujian pertama hasil akurasi tertinggi yang didapatkan sebesar 96.67%, dan pada pengujian kedua didapatkan hasil akurasi tertinggi sebesar 96.67%.

Kata kunci: klasifikasi, penyakit ginjal kronis, *Fk-NN*

ABSTRACT

Septyan Teguh Mahendra. 2016. ***Chronic Kidney Disease Classification Based on Chronic Kidney Disease Dataset Using Fuzzy k-Nearest Neighbor (Fk-NN) Method. Study Program of Informatics/Computer Science. Department of Informatics Engineering. Faculty of Computer Science. University of Brawijaya. Malang. Supervisor*: M. Tanzil Furqon, S.Kom, M.CompSc *and* Randy Cahya W, S.ST., M.kom.**

This study discusses about implementation of Fuzzy k-Nearest Neighbor algorithm to perform chronic kidney disease classification. The data used in this system development is the chronic kidney disease dataset, which is obtained from the UCI Machine Learning Repository. Fuzzy k-Nearest Neighbor (Fk-NN) is a classification method that combines fuzzy techniques with k-Nearest Neighbor classifier. In this study conducts several tests to determine the system accuracy level, the first test is the effect of training data amount to the system accuracy results, and the second testing is the effect of the k value to system accuracy results. In the first test, the highest accuracy results obtained is in the amount of 96.67%, and in the second test obtained highest accuracy in the amount of 96.67%.

Key word: classification, chronic kidney disease, Fk-NN

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SOURCE CODE

BAB 1 PENDAHULUAN

1.1 Latar belakang

Salah satu organ dalam tubuh manusia yang memegang peranan yang sangat vital adalah ginjal. Ginjal memiliki fungsi utama dalam pengeluaran bahan-bahan buangan yang oleh tubuh sudah tidak diperlukan lagi, yang dimaksud dengan bahan buangan adalah bahan-bahan yang tidak diperlukan yang merupakan hasil dari proses metabolisme tubuh seperti pemrosesan makanan, degradasi jaringan tubuh, dan sebagainya. Air dan garam dalam darah yang berlebihan juga akan disekresikan oleh ginjal. Setiap harinya, hampir 200 liter darah diproses dan kurang lebih 2 liter urin dihasilkan oleh ginjal. Selain itu ginjal juga memiliki fungsi untuk mengatur konsentrasi garam dan berbagai macam mineral dalam darah serta mengatur keseimbangan asam-basa darah (Pranay, 2010).

Satu dari sembilan orang atau setara 20 juta lebih penduduk Amerika adalah penderita *chronic kidney disease (CKD)* atau penyakit ginjal kronis (Lange, 2008). Penyakit ginjal kronis adalah suatu keadaan terjadinya kerusakan ginjal atau laju filtrasi glomerulus (LFG) < 60 mL/menit dalam waktu 3 bulan atau lebih (Nahas, 2003; Vijayakumar, et al, 2007; Parmar, 2002). Dalam kondisi tersebut, secara berangsur dan *irreversible*, ginjal mengalami penurunan fungsi dan akan terus berkembang hingga terjadi gagal ginjal terminal. Terjadinya kerusakan ginjal tersebut bisa dilihat dari kelainan yang terjadi pada darah atau urin, pencitraan, dan biopsi ginjal. Tidak hanya gagal ginjal, dalam perkembangannya, penyakit ginjal kronis juga akan mengakibatkan terjadinya komplikasi lain karena fungsi ginjal yang mengalami penurunan dan juga penyakit kardiovaskular (Sharon, 2006; Levey, et al, 2003).

Penderita penyakit ginjal kronis memiliki angka morbiditas dan mortalitas yang tinggi. Prevalensi penyakit ginjal kronis stadium 2-5 terus meningkat sejak tahun 1988 dimana prevalensi diabetes (40%) dan hipertensi (20%) yang merupakan salah satu faktor etiologi dalam kasus penyakit ginjal kronis. Berdasarkan data NHANES tahun 2003-2006, prevalensi penyakit ginjal kronis berdasarkan usia adalah 39,4% pada usia diatas 60 tahun, 12,6% pada usia 40-59 tahun dan 8,5% pada usia 20-39 tahun , sedangkan prevalensi penyakit ginjal kronis pada penderita diabetes sebanyak 40,2% dibandingkan dengan bukan penderita diabetes yaitu 15,4%, pada penderita penyakit kardiovaskular sebnyak 28,2% dan bukan penderita penyakit kardiovaskular sebanyak 15,4% (Yee, et al, 2011).

Penelitian terdahulu mengenai klasifikasi penyakit ginjal kronis pernah dilakukan oleh Abhinandan Dubey pada tahun 2015 dari Birla Institute of Applied Sciences dengan judul penelitian "*A Classification of CKD Cases Using MultiVariate K-Means Clustering*" (Dubey, 2015).

Untuk penelitian kali ini, metode yang digunakan adalah *Fuzzy* dengan gabungan metode untuk pengklasifikasian data pada *data mining*. Ada beberapa metode dalam *data mining* yang dapat digunakan dalam pengklasifikasian data,

antara lain metode *k-Nearest Neighbor*, *Bayesian,* ID3 dan C45, serta masih ada pula beberapa metode lainnya, dimana terdapat target variable kategori dalam klasifikasi (Kusrini, 2009).

Sesuai uraian diatas, salah satu metode dalam *data mining* yang dapat digunakan untuk mengklasifikasikan data adalah metode *k-Nearest Neighbor*. Metode ini dapat digabungkan dengan metode lain, salah satunya adalah digabungkan dengan metode *Fuzzy*, sehingga menjadi metode *Fuzzy k-Nearest Neighbor (Fk-NN)*. Dalam metode ini, teknik *Fuzzy* digabungkan dengan teknik *data mining* (Zhang et al*.,* 2009). Pada dasarnya algoritma *Fk-NN* menetapkan nilai keanggotaan sebagai fungsi jarak vektor dari *k-NN* dan keanggotaan tetangganya pada kelas-kelas yang memungkinkan. Algoritma *Fk-NN* tidak menempatkan vektor pada kelas tertentu dalam implementasinya, algoritma ini memberikan nilai keanggotaan kelas pada vektor sampel. Sehingga, nilai-nilai keanggotaan vektor akan memberikan tingkat jaminan terhadap hasil pengklasifikasian.

Metode *Fk-NN* sudah banyak diterapkan pada berbagai penelitian antara lain pada tahun 2013 penelitian yang dilakukan Rahmi Amiratus untuk menentukan kualitas rendemen tanaman tebu, tingkat akurasi tertinggi yang diperoleh adalah 98%. Adi Yuistiyanto pada tahun 2016 menggunakan metode *Fk-NN* untuk pengklasifikasian penyakit kelainan tulang, akurasi yang dihasilkan sebesar 83,30%. Dan masih banyak lagi penelitian lain yang menerapkan metode *Fk-NN,* namun dari penelitian yang dilakukan Rahmi Amiratus dan Adi Yuistiyanto tersebut, dapat dilihat bahwa hasil akurasi yang didapatkan cukup tinggi. Selain karena tingkat akurasi yang cukup tinggi, metode ini juga sederhana dan mudah digunakan.

Berdasarkan uraian di atas, maka penelitian ini diberi judul *"***KLASIFIKASI PENYAKIT GINJAL KRONIS BERDASARKAN *CHRONIC KIDNEY DISEASE DATASET* MENGGUNAKAN METODE *FUZZY K-NEAREST NEIGHBOR (FK-NN)***"**. Data yang digunakan untuk klasifikasi pada *Chronic Kidney Disease Dataset* yaitu *age, blood pressure, specific gravity, albumin, sugar, red blood cells, pus cell, pus cell clumps, bacteria, blood glucose random, blood urea, serum creatinine, sodium, potassium, hemoglobin, packed cell volume, white blood cell count, red blood cell count, hypertension, diabetes mellitus, coronary artery disease, appetite, pedal edema,* dan *anemia.*

1.2 Rumusan masalah

Berdasarkan apa yang telah disampaikan dalam latar belakang penelitian tentang klasifikasi penyakit ginjal kronis berdasarkan *chronic kidney disease dataset* menggunakan metode *Fuzzy k-Nearest Neighbor (Fk-NN)*, maka berikut ini adalah rumusan masalah dari penelitian yang akan dilakukan.

1. Bagaimana melakukan klasifikasi penyakit ginjal kronis berdasarkan *chronic kidney disease dataset* dengan menerapkan metode *Fuzzy k–Nearest Neighbor* (*Fk-NN*).

2. Bagaimana tingkat akurasi hasil klasifikasi penyakit ginjal kronis berdasarkan pada *chronic kidney disease dataset* menggunakan metode *Fuzzy k-Nearest Neighbor* (*Fk-NN*).

1.3 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini ialah sebagai berikut:

1. Menerapkan *Fuzzy k–Nearest Neighbor* (*Fk-NN*) untuk klasifikasi penyakit ginjal kronis berdasarkan pada *chronic kidney disease dataset*.
2. Menguji tingkat akurasi hasil klasifikasi penyakit ginjal kronis berdasarkan pada *chronic kidney disease dataset* menggunakan metode *Fuzzy k-Nearest Neighbor* (*Fk-NN*).

1.4 Manfaat

Setelah penelitian ini diselesaikan diharapkan dapat membantu masyarakat umum atau bahkan para tenaga medis dalam mengklasifikasikan penyakit ginjal kronis, sehingga untuk kedepannya bisa segera dilakukan tindakan medis untuk pasien yang membutuhkan.

1.5 Batasan masalah

Berikut adalah batasan masalah yang telah ditentukan dalam penelitian ini.

1. Batasan subjek penelitian berupa *dataset* dari *UCI Machine Learning Repository* yang dapat diakses pada alamat website berikut ini [https://archive.ics.uci.edu/ml/datasets/Chronic_Kidney_Disease], *data*set yang digunakan adalah *Chronic Kidney Disease Dataset* yang memiliki dua kelas yaitu *ckd* dan *notckd*.
2. Parameter yang digunakan untuk menentukan kelas yaitu *age, blood pressure, specific gravity, albumin, sugar, red blood cells, pus cell, pus cell clumps, bacteria, blood glucose random, blood urea, serum creatinine, sodium, potassium, hemoglobin, packed cell volume, white blood cell count, red blood cell count, hypertension, diabetes mellitus, coronary artery disease, appetite, pedal edema,* dan *anemia.* Dengan jumlah total data yang digunakan sebanyak 158 data.

1.6 Sistematika pembahasan

Susunan penulisan berikut adalah sistematika pembahasan yang diterapkan dalam penelitian ini.

BAB 1 PENDAHULUAN

Bagian pendahuluan dalam penelitian ini memaparkan tentang latar belakang penelitian, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika pembahasan.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan berisi tentang pembahasan dasar teori yang diambil dari berbagai sumber mengenai metode yang diterapkan dan objek penelitian yang digunakan dalam penelitian ini.

BAB 3 METODOLOGI

Pada bab ini dijelaskan tentang tahapan-tahapan dalam menerapkan metode yang digunakan untuk perancangan proses sistem yang dibangun dalam penelitian ini.

BAB 4 HASIL

Bab ini berisi mengenai hasil implementasi dari metode dan perancangan yang telah dilakukan dalam penelitian ini.

BAB 5 PEMBAHASAN

Pembahasan dalam penelitian ini mencakup pengujian terhadap hasil penerapan metode serta analisa dari hasil pengujian.

BAB 6 PENUTUP

Bagian penutup berisi kesimpulan yang didapat dari hasil penerapan metode yang digunakan dalam pembangunan sistem, serta pengujian dan analisa hasil yang dilakukan dalam penelitian ini, dan juga saran-saran untuk penelitian selanjutnya terkait penelitian yang telah dilakukan ini.

BAB 2 LANDASAN KEPUSTAKAAN

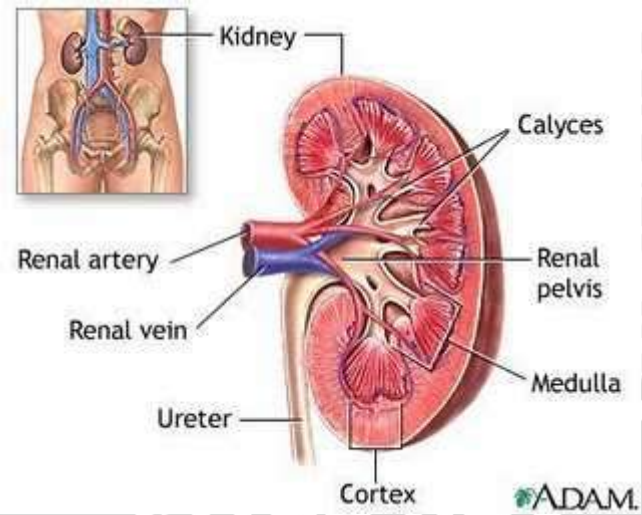
2.1 Ginjal

Salah satu organ dalam tubuh manusia yang memegang peranan yang sangat vital adalah ginjal. Ginjal memiliki fungsi utama dalam pengeluaran bahan-bahan buangan yang oleh tubuh sudah tidak diperlukan lagi, yang dimaksud dengan bahan buangan adalah bahan-bahan yang tidak diperlukan yang merupakan hasil dari proses metabolisme tubuh seperti pemrosesan makanan, degradasi jaringan tubuh, dan sebagainya. Air dan garam dalam darah yang berlebihan juga akan disekresikan oleh ginjal. Setiap harinya, hampir 200 liter darah diproses dan kurang lebih 2 liter urin dihasilkan oleh ginjal. Selain itu ginjal juga memiliki fungsi untuk mengatur konsentrasi garam dan berbagai macam mineral dalam darah serta mengatur keseimbangan asam-basa darah (Pranay, 2010).

Berikut ini adalah struktur dan anatomi ginjal menurut Pearce dan Wilson (2006). Ginjal terletak pada dinding posterior abdomen terutama didaerah lumbal, disebelah kanan dan kiri tulang belakang, dibungkus lapisan lemak yang tebal dibelakang pritonium. Kedudukan gijal dapat diperkirakan dari belakang, mulai dari ketinggian vertebra torakalis terakhir sampai vertebra lumbalis ketiga. Dan ginjal kanan sedikit lebih rendah dari ginjal kiri karena tertekan oleh hati.

Setiap ginjal panjangnya antara 12 cm sampai 13 cm, lebarnya 6 cm dan tebalnya antara 1,5 sampai 2,5 cm, pada orang dewasa berat ginjal antara 140 sampai 150 gram. Bentuk ginjal seperti kacang dan sisi dalamnya atau hilus menghadap ketulang belakang, serta sisi luarnya berbentuk cembung. Pembuluh darah ginjal semuanya masuk dan keluar melalui hilus. Di atas setiap ginjal menjulang kelenjar suprarenal.

Setiap ginjal dilingkupi kapsul tipis dan jaringan fibrus yang membungkusnya, dan membentuk pembungkus yang halus serta di dalamnya terdapat struktur-struktur ginjal. Setruktur ginjal warnanya ungu tua dan terdiri dari bagian kapiler disebelah luar, dan medulla disebelah dalam. Bagian medulla tersusun atas 15 sampai 16 bagian yang berbentuk piramid, yang disebut sebagai piramid ginjal. Puncaknya mengarah ke hilus dan berakhir di kalies, kalies akan menghubungkan dengan pelvis ginjal.



Gambar 2.1 Potongan vertikal ginjal

2.2 Penyakit ginjal kronis

Penyakit ginjal kronis adalah suatu keadaan terjadinya kerusakan ginjal atau laju filtrasi glomerulus (LFG) < 60 mL/menit dalam waktu 3 bulan atau lebih (Nahas, 2003; Vijayakumar, et al, 2007; Parmar, 2002). Dalam kondisi tersebut, secara berangsur dan *irreversible*, ginjal mengalami penurunan fungsi dan akan terus berkembang hingga terjadi gagal ginjal terminal. Terjadinya kerusakan ginjal tersebut bisa dilihat dari kelainan yang terjadi pada darah atau urin, pencitraan, dan biopsi ginjal. Tidak hanya gagal ginjal, dalam perkembangannya, penyakit ginjal kronis juga akan mengakibatkan terjadinya komplikasi lain karena fungsi ginjal yang mengalami penurunan dan juga penyakit kardiovaskular (Sharon, 2006; Levey, et al, 2003).

Penyakit ginjal kronis diklasifikasikan menjadi beberapa stadium untuk tujuan pencegahan, identifikasi awal kerusakan ginjal dan penatalaksanaan, serta untuk pencegahan komplikasi penyakit ginjal kronis (Nahas, 2003; Vijayakumar, et al, 2007; Levey, et al, 2003).

1. Kerusakan ginjal dengan GFR normal/meningkat (>90 mL/menit/1,73m^2). Penatalaksanaan pada stadium ini yaitu diagnosa dan perawatan, memperlambat perkembangan, perawatan kondisi komorbiditas, deduksi resiko penyakit kardiovaskular.
2. Kerusakan ginjal dengan penurunan GFR ringan (60-89 mL/menit/1,73m^2). Penatalaksanaan pada stadium ini yaitu estimasi perkembangan penyakit.
3. Kerusakan ginjal dengan penurunan GFR sedang (30-59 mL/menit/1,73m^2)). Penatalaksanaan pada stadium ini yaitu evaluasi dan perawatan komplikasi.
4. Kerusakan ginjal dengan penurunan GFR berat (15-29 mL/menit/1,73m^2)). Penatalaksanaan pada stadium ini yaitu persiapan terapi penggantian ginjal.
5. Gagal ginjal (<15 mL/menit/1,73m^2 atau dialisis). Penatalaksanaan pada stadium ini yaitu penggantian (apabila terdapat uremia).

Smeltzer dan Bare (2002) menyatakan bahwa terjadinya penyakit ginjal kronis sebagai akibat dari menurunnya fungsi renal sehingga terjadi uremia yang menimbulkan pengaruh terhadap sistem tubuh, hal ini dikarenakan produk akhir dari hasil proses metabolisme protein yang seharusnya disekresikan kedalam urin tertimbun dalam darah. Gejala-gejala yang timbul akan semakin meningkat seiring semakin banyaknya timbunan dalam darah yang akan menimbulkan gangguan kliren renal. Menurunnya jumlah glomerulus yang berfungsi akan menyebabkan banyak masalah terjadi pada ginjal, hal ini berakibat pada kliren substansi darah yang harusnya dibersihkan oleh ginjal mengalami penurunan.

Deteksi terhadap penurunan yang terjadi pada laju filtrasi glomerulus (LFG) dapat dilakukan dengan melakukan pemeriksaan kliren kreatinin pada urin 24 jam. Tidak berfungsinya glomeluri kliren kreatinin atau menurunnya filtrasi glomelurus mengakibatkan kadar kreatinin serum akan meningkat, serta biasanya kadar nitrogen urea darah (NUD) juga mengalami peningkatan. Tubuh memproduksi kreatinin serum secara konstan, sehingga substansi ini menjadi indikator fungsi renal yang paling sensitif. Selain penyakit renal, masukan protein dalam diet, katabolisme dan medikasi seperti steroid adalah hal-hal yang mempengaruhi nitrogen urea darah.

Retensi cairan dan natrium juga akan timbul saat terjadi penurunan laju filtrasi glomerulus, keadaan tersebut timbul karena pada penyakit ginjal tahap akhir ginjal tidak lagi mampu melakukan pengkonsentrasian atau pengenceran urin secara normal, serta ginjal tidak lagi mampu merespon dengan normal terhadap perubahan elektrolit dan cairan yang masuk sehari-hari. Sering tertahannya cairan dan natrium dalam tubuh akan meningkatkan resiko terjadinya gagal jantung kongesti, hipertensi, dan oedema. Aktivasi aksis renin angiostenin dan kerjasama keduanya yang menyebabkan meningkatnya sekresi aldosterone juga merupakan penyebab terjadinya hipertensi. Pada kasus lain pasien mempunyai kecenderungan kehilangan garam yang menimbulkan resiko terjadinya hipotensi dan hipovolemia. Status uremik akan semakin memburuk pada kejadian muntah dan diare yang dikarenakan penipisan air dan natrium.

Asidosis metabolik terjadi akibat ketidakmampuan ginjal mensekresikan muatan asam (H^+) yang berlebihan. Sekresi asam terutama akibat ketidakmampuan tubulus ginjal untuk mensekresi amonia (NH^3) dan mengabsorpsi natrium bikarbonat (HCO3). Penurunan sekresi fosfat dan asam organik lain juga terjadi.

Terjadinya anemia yang disertai sesak nafas, angina, keletihan, dan penurunan eritoprotein disebabkan oleh kerusakan ginjal pada penyakit ginjal kronis. Kecenderungan terjadinya pendarahan terutama pada saluran gastrointestinal sehingga menyebabkan terjadinya anemia berat atau sedang, defisiensi nutrisi, dan pendeknya usia sel darah merah merupakan keadaan-keadaan yang diakibatkan oleh eritroprotein yang tidak adekuat. Eritroprotein merupakan substansi yang dihasilkan oleh ginjal guna menstimulasi sum-sum tulang dalam menghasilkan sel darah merah.

Menurut Smeltzer dan Bare (2002) kelainan utama yang lain, yang terjadi pada penyakit ginjal kronis antara lain adalah gangguan pada metabolisme kalsium dan fosfat dalam tubuh, saat salah satunya mengalami peningkatan maka yang lain mengalami penurunan, hal ini dikarenakan keduanya memiliki hubungan yang saling timbal balik. Saat laju filtrasi glomelurus mengalami penurunan akan menyebabkan terjadinya peningkatan kadar fosfat serum, dan sebaliknya saat kadar serum mengalami penurunan akan menyebabkan sekresi parathormon dari kelenjar paratiroid mengalami penurunan. Sedangkan kondisi yang terjadi pada penyakit ginjal kronis adalah ketidakmampuan tubuh dalam merespon secara normal atas peningkatan sekresi *parathormon* yang terjadi sehingga mengakibatkan penurunan kalsium dalam tulang. Selain itu, metabolik aktif vitamin D (1,25 dihidrokolekalsiferol) yang normalnya diproduksi didalam ginjal juga akan mengalami penurunan, dan akan menyebabkan terjadinya penyakit tulang uremik yang sering disebut *Osteodistrofienal* seiring makin berkembangnya penyakit ginjal kronis. Perubahan komplek kalsium, fosfat dan keseimbangan *parathormon* adalah penyebab terjadinya *Osteodistrofienal.* Gangguan yang mendasari ekskresi protein dan urin serta terjadinya hipertensi berkaitan dengan laju penurunan fungsi ginjal. Pasien yang mengekresikan sejumlah protein secara signifikan atau mengalami peningkatan tekanan darah memiliki kecenderungan terhadap kondisi yang cepat memburuk dibandingkan dengan mereka yang tidak mengalami kondisi ini.

Pada *Chronic Kidney Disease Dataset* yang digunakan dalam penelitian ini, terdapat sejumlah parameter yang digunakan untuk menentukan apakah seorang pasien menderita penyakit ginjal kronis atau tidak. Parameter-parameter tersebut adalah *age, blood pressure, specific gravity, albumin, sugar, red blood cells, pus cell, pus cell clumps, bacteria, blood glucose random, blood urea, serum creatinine, sodium, potassium, hemoglobin, packed cell volume, white blood cell count, red blood cell count, hypertension, diabetes mellitus, coronary artery disease, appetite, pedal edema,* dan *anemia.* Berikut adalah penjelasan singkat tentang parameter-parameter yang digunakan dalam *dataset* tersebut.

1. *Age* (Dorland, 2002)
   - Durasi, atau pengukuran waktu, dari keberadaan seseorang atau objek.
   - Pengukuran suatu benda relatif terhadap usia kronologis rata-rata individu normal.
2. *Blood pressure* (Dorland, 2002)

   Tekanan darah pada dinding arteri, yang bergantung pada kekuatan gerak jantung, kelenturan dinding arteri, serta volume dan viskositas darah.
3. *Specific gravity* (Dorland, 2002)

   Berat substansi dibanding dengan berat zat lain yang diambil sebagai standar.

4. *Albumin* (Dorland, 2002)
    - Protein yang terlarut dalam air dan juga dalam konsentrasi larutan garam yang sedang.
    - Serum albumin.
5. *Sugar* (Dorland, 2002)

    Setiap golongan karbohidrat yang larut dalam air gula, monosakarida dan oligosakarida terkecil, kadang-kadang sukrosa tertentu.
6. *Red blood cells* (Dorland, 2002)

    Eritrosit, sel darah merah atau korpuskel; salah satu unsur yang dibentuk dalam darah tepi.
7. *Pus cell* (Mosby, 2009)

    Leukosite polimorfonuklear yang nekrosis yang merupakan komponen terbesar dari pus.
8. *Pus cell clumps* (Dorland, 2002)

    Kumpulan atau kelompok *pus cell*.
9. *Bacteria* (Dorland, 2002)

    Setiap mikroorganisme prokariotik uniseluler yang umumnya memperbanyak diri dengan pembelahan sel, tidak ada nukleus atau organel yang terikat membrane dan mempunyai dinding sel; bakteri ini dapat aerobik maupun anaerobik, motil maupun non motil, hidup bebas, saprofitik, parasitic, atau patogenik.
10. *Blood glucose random* (Mooney, 2009)

    Gula darah, jumlah glukosa dalam aliran darah, dilihat tanpa mempedulikan kapan terakhir kali mengkonsumsi makanan.
11. *Blood urea* (Youngson, 2004,2005)

    Level urea dalam darah. Ginjal yang berfungsi normal menjaga level urea dalam darah rendah dengan cara mengekskresikannya lewat urin. Level urea yang tinggi mengindikasikan adanya kegagalan fungsi ginjal.
12. *Serum creatinine* (Dorland, 2002)

    Bentuk anhidrida keratin; hasil akhir metabolism fosfokreatin; pengukuran laju akskres urin digunakan sebagai indikator diagnostik fungsi ginjal dan massa otot.
13. *Sodium* (Dorland, 2002)

    Natrium, unsur kimia dalam tabel periodik yang memiliki simbol Na dan nomor atom 11, logam reaktif yang lunak, keperakan, dan seperti lilin, yang termasuk ke logam alkali yang banyak terdapat dalam senyawa alam (terutama halite).

14. *Potassium* (Dorland, 2002)

    Kalium, unsur kimia dalam table periodik yang memiliki simbol K dan nomor atom 19, merupakan kation utama cairan intraselular.

15. *Hemoglobin* (Dorland, 2002)

    Pigmen pembawa oksigen eritrosit, dibentuk oleh eritrosit yang berkembang dalam sumsum tulang, merupakan empat rantai polipeptida globin yang berbeda, masing-masing terdiri dari beberapa ratus asam amino.

16. *Packed cell volume* (Farlex, 2012)

    Hematokrit atau volume eritrosit yang dimampatkan yang merupakan persentase volume eritrosit dalam darah yang dimampatkan dengan cara diputar pada kecepatan tertentu dan dalam waktu tertentu guna mengetahui konsentrasi eritrosit dalam darah.

17. *White blood cell count* (Mosby, 2008)

    Pemeriksaan klinis laboratorium untuk menentukan jumlah dan tipe leukosit yang ada dalam sampel pengambilan darah. Secara umum jumlah leukosit normal berkisar dari 5000 hingga 10,000/mm^3.

18. *Red blood cell count* (Farlex, 2012)

    Erythrocyte count, konsentasi eritrosit dalam spesimen darah. Jumlahnya bervariasi bergantung pada usia (lebih tinggi pada bayi), waktu perhari (rendah ketika tidur), aktivitas, temperatur lingkungan, *altitude*. Nilai rata-rata hitung eritrosit untuk laki-laki adalah 4.7 hingga 6.1 juta sel/mcL dan bagi perempuan adalah 4.2 hingga 5.4 juta sel/mcL.

19. *Hypertension* (Dorland, 2002)

    Tekanan darah arterial yang tetap tinggi, dapat tidak memiliki sebab yang diketahui (*essential, idiopathic, atau primary hypertension*) atau berkaitan dengan penyakit lain (*secondary hypertension*).

20. *Diabetes mellitus* (Dorland, 2002)

    Kelainan *metabolic* dimana ditemukan ketidakmampuan untuk mengoksidasi karbohidrat, akibat gangguan pada mekanisme insulin yang normal, menimbulkan hiperglikemia, glikosuria, poliuria, rasa haus, rasa lapar, badan kurus, kelemahan asidosis, sering menyebabkan *dyspnea*, *lipemia*, *ketonuria*, dan akhirnya koma.

21. *Coronary artery disease* (Farlex, 2012)

    Penyakit arteri koroner, penyempitan atau blokade arteri koroner (pembuluh arteri yang memasok darah ke jantung). Penyakit ini biasanya disebabkan oleh *aterosklerosis*, suatu kondisi yang disebabkan oleh penimbunan kolesterol, kalsium dan endapan lemak (plak) pada dinding bagian dalam arteri.

22. *Appetite* (Farlex, 2012)

    Keinginan atau motivasi dalam pemenuhan kebutuhan biologis maupun psikologis untuk makanan, minuman, hubungan seksual, maupun kasih sayang. Keinginan atau kebutuhan untuk memuaskan kebutuhan fisik maupun mental.

23. *Pedal edema* (McGraw-Hill, 2002)

    Pengumpulan cairan secara abnormal dalam ruang jaringan interseluler tubuh pada kaki, terutama pada bagian dorsum. *PE* merupakan karakteristik *CHF* (*congestive heart failure*)

24. *Anemia* (Dorland, 2002)

    Penurunan dibawah normal dalam jumlah eritrosit, banyaknya hemoglobin, atau volume sel darah merah (*packed red cells*) dalam darah; gejala berbagai penyakit dan kelainan.

2.3 *Data mining*

2.3.1 Pengertian *data mining*

Dalam sebuah *database* perlu dilakukan penguraian untuk dapat menemukan suatu pengetahuan yang terkandung didalamnya, istilah yang digunakan untuk melakukan hal tersebut adalah *data mining*. Dalam prosesnya untuk mengidentifikasikan informasi yang berguna dan pengetahuan yang terkait dari berbagai *database* besar, *data mining* menerapkan teknik matematika, statistika, kecerdasan buatan, dan juga *machine learning* (Turban, 2005). Data mining mampu meramalkan *trend* dan sifat-sifat perilaku bisnis yang sangat bermanfaat dalam mempertimbangkan pengambilan keputusan-keputusan penting. Dibandingkan dengan sistem pendukung keputusan tradisional yang sudah banyak digunakan, *data mining* dengan analisis yang telah diotomatisasi mampu melakukan hal yang lebih. Pola-pola tersembunyi dan informasi yang mungkin terlupakan karena terletak di luar ekspektasi pelaku bisnis dapat ditemukan dan diprediksikan dengan melakukan eksplorasi *database* menggunakan *data mining*. Oleh karena itu teknologi *data mining* sangat bermanfaat bagi perusahaan-perusahaan dalam rangka membantu menemukan informasi-informasi penting yang ada dalam *database* mereka (Khusnawi, 2007).

Berdasarkan tugas yang dilakukan, metode *data mining* terbagi dalam beberapa kelompok, diantaranya adalah metode deskripsi, estimasi, prediksi, klasifikasi, pengklasteran dan asosiasi (Larose, 2005).

Pada penelitian ini digunakan metode *data mining* dengan prediksi menggunakan metode gabungan klasifikasi *k-Nearest Neighbor* dengan metode *Fuzzy*.

Terdapat beberapa fungsionalitas *data mining*, sebagai berikut (Han et al., 2000):

1. Analisis asosiasi

   Menemukan aturan (*rule*) asosiasi yang menunjukkan kondisi nilai atribut yang sering ada bersamaan dalam satu kumpulan data.

2. Klasifikasi

   Fungsi pembelajaran yang memetakan (mengklasifikasi) sebuah item data ke dalam salah satu dari beberapa kelas yang sudah didefinisikan.

3. *Clustering*

   Melakukan pengelompokan data tanpa berdasarkan kelas data tertentu.

4. Pendeteksian perubahan dan deviasi

   Berfokus pada penemuan perubahan yang paling signifikan di dalam data dari nilai-nilai yang telah diukur sebelumnya.

2.3.2 Proses *data mining*

Prosedur yang umum digunakan untuk permasalahan data *mining* meliputi tahap-tahap sebagai berikut (Kantardzic, 2003):

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 (Moradian et al., 2009).

Dalam skripsi digunakan normalisasi *min-max*. Normalisasi *min-max* dihitung dengan persamaan 2.1 (Moradian et al., 2009). Normalisasi *min-max* memiliki keunggulan yaitu menjaga relasi pada data. Serta mempunyai fungsi yaitu menyatukan satuan dari berbagai atribut (Jayalakshmi et al., 2011).

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

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

2.4 Klasifikasi

Menurut Khusnawi (2007), *data mining* memiliki suatu fungsionalitas guna menghasilkan model untuk melakukan prediksi kategori atau kelas dari objek-objek dalam baris data yang disebut klasifikasi. Dalam prosesnya, klasifikasi memiliki dua tahapan. Tahap pertama adalah tahap pembelajaran, dan tahap selanjutnya adalah tahap pengklasifikasian.

Pada tahap pembelajaran, data *training* dianalisis untuk membangun sebuah model klasifikasi dengan menggunakan sebuah algoritma klasifikasi. Pada tahap ini lah dilakukan pembentukan fungsi atau pemetaan *y = f(x)* dimana *y* merupakan kelas hasil prediksi, dan *x* merupakan *tuple* yang akan diprediksi kelasnya.

Tahap selanjutnya adalah tahap pengklasifikasian, pada tahap ini dilakukan klasifikasi terhadap *unknown data* menggunakan model yang telah dihasilkan. Untuk melakukan klasifikasi, model yang boleh digunakan adalah adalah model yang cukup tinggi akurasinya. Data *test* dapat digunakan untuk melakukan pengujian terhadap model guna mengetahui akurasinya. Pada data *test*, label kelas sudah diketahui. Data *test* yang digunakan harus data yang berbeda dengan data yang digunakan untuk data *training*, hal ini dikarenakan jika data *test* menggunakan data yang sama dengan data *training* maka pengujian yang dilakukan akan menunjukkan hasil akurasi yang tinggi, padahal belum tentu demikian hasil sesungguhnya.

2.5 Algoritma *k-Nearest Neighbor*

2.5.1 Pengertian *k-Nearest Neighbor*

Algoritma *k-Nearest Neighbor* merupakan metode pengklasifikasian objek yang dilakukan berdasarkan data pembelajaran dengan jarak terdekat dari objek tersebut. Data pembelajaran tersebut kemudian diproyeksikan kedalam ruang berdimensi banyak yang tiap dimensinya merepresentasikan fitur-fitur yang dimiliki data tersebut. Ruang tersebut terbagi menjadi bagian-bagian

berdasarkan hasil pengklasifikasian pada data pembelajaran. Jika kelas c merupakan hasil klasifikasi yang paling banyak ditemui pada *k* buah tetangga terdekat sebuah titik, maka sebuah titik tersebut ditandai dengan kelas c. Jauh dekatnya tetangga dapat dihitung dengan menggunakan *Euclidean Distance*.

Berikut adalah beberapa keuntungan dari metode *k-NN*:

1. Sederhana dalam penggunaannya.
2. Mampu menangani data *training* yang didalamnya terdapat *noise*.
3. Efektif untuk data *training* yang besar.

Namun metode ini tetap memiliki beberapa kelemahan disamping keuntungan-keuntungan yang dimiliki, kelemahan metode *k-NN* antara lain adalah sebagai berikut (Hamid et al., 2008):

1. Algoritma ini perlu menghitung jarak setiap data *training* sehingga *computation cost* cukup tinggi.
2. Memori yang dibutuhkan cukup besar.
3. Tingkat akurasi rendah pada dataset multidimensi.
4. Perlu menentukan nilai parameter *k*, jumlah tetangga terdekat.
5. Menggunakan perhitungan jarak namun belum diketahui pasti jenis jarak yang digunakan.
6. Belum diketahui atribut yang lebih baik untuk menghasilkan hasil terbaik.

Namun secara keseluruhan keuntungan dari algoritma *k-NN* 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 (Yanita, 2013). Algoritma ini juga sering diterapkan untuk klasifikasi dalam teknik *data mining* (Moradian et al., 2009).

2.5.2 Proses *k-Nearest Neighbor*

Agusta (2007) menyatakan bahwa prinsip kerja dari algoritma ini yaitu mencari jarak terdekat antara data yang dievaluasi dengan *k* tetangga terdekatnya dalam data pelatihan.

Lalu, melakukan perhitungan dengan menggunakan fungsi *Euclidean Distance* seperti yang ditunjukkan pada persamaan 2.2 yang digunakan untuk menghitung jarak terdekat antara data uji dengan data latih.

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

Keterangan:

x_1 = Data latih

x_2 = Data uji

i = Variabel data

d = Jarak

p = Dimensi data

Selanjutnya diambil sebanyak *k* tetangga terdekat untuk menentukan label kelas dari *record* baru menggunakan label kelas tetangga setelah diketahui jarak antar *record*.

2.6 Logika *Fuzzy*

Pada tahun 1965 Prof. Lotfi A. Zadeh memperkenalkan untuk pertama kalinya tentang logika *Fuzzy* yang merupakan salah satu komponen pembentuk *soft computing.* Penalaran dengan logika *Fuzzy* memiliki ciri utama yaitu pada *membership function* atau derajat keanggotaan. Dalam teori himpunan *Fuzzy,* derajat keanggotaan memiliki peranan yang sangat penting sebagai penentu keberadaan elemen dalam suatu himpunan. Teori himpunan *Fuzzy* merupakan dasar logika fuzzy (Kusumadewi et al*.,* 2010).

Logika *Fuzzy* sering digunakan karena beberapa alasan, antara lain karena konsep logika *Fuzzy* mudah dimengerti, sangat fleksibel, memiliki toleransi terhadap data-data yang tidak tepat, mampu memodelkan fungsi-fungsi nonlinear yang sangat kompleks, kemampuan untuk membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan, selain itu logika *Fuzzy* dapat bekerjasama dengan teknik-teknik kendali secara konvensional, serta didasarkan pada bahasa alami. Dalam logika *Fuzzy* dikenal keadaan berhingga dari nilai "0" sampai ke nilai "1". Logika *Fuzzy* mengenal sejumlah keadaan yang berkisar dari keadaan salah sampai keadaan benar, bukan hanya mengenal dua keadaan (Yulianto, 2008).

Dalam memahami sistem *Fuzzy* ada beberapa hal yang perlu diketahui, yaitu:

1. Variable *Fuzzy*

   Yang dimaksud dengan variabel *Fuzzy* adalah variabel yang akan dibahas dalam suatu sistem *Fuzzy*. Contoh: kecepatan *processor*, kapasitas *memory*, kapasitas *harddisk.*

2. Himpunan *Fuzzy*

   Himpunan yang tiap elemennya mempunyai derajat keanggotaan tertentu terhadap himpunannya disebut himpunan *Fuzzy*. Himpunan *Fuzzy* memiliki dua macam atribut, yaitu atribut linguistik dan atribut numeris. Atribut linguistik merupakan penamaan suatu grup yang menandakan suatu keadaan atau kondisi tertentu menggunakan bahasa alami. Sedangkan atribut numeris merupakan suatu nilai (angka) yang diberikan untuk menunjukkan ukuran dari suatu variabel. Contoh himpunan *Fuzzy* yang digunakan adalah himpunan kecepatan lambat, sedang dan cepat untuk variabel kecepatan *processor*, himpunan kapasitas kecil, sedang dan besar untuk variabel kapasitas memori.

3. Semesta pembicaraan

   Semesta pembicaraan adalah suatu keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel *Fuzzy*. Nilai semesta pembicaraan dapat berupa bilangan positif atau bilangan negatif. Contoh: semesta pembicaraan variabel kecepatan *processor* adalah [0 +∞].

4. Domain

   Domain himpunan *Fuzzy* adalah keseluruhan nilai yang diijinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan *Fuzzy*. Nilai domain dapat berupa bilangan positif maupun bilangan negatif. Contoh: domain untuk variabel kecepatan *processor* adalah lambat [0, 2.6], sedang [1.8, 3] dan cepat [2.6, +∞], variabel kapasitas *memory* kecil [0, 256], sedang [64, 512] dan besar [256, +∞], variabel kapasitas harddisk kecil [0, 160], sedang [40, 400] dan besar [160, +∞].

Logika *Fuzzy* memiliki beberapa karakteristik yaitu himpunan *Fuzzy* dan fungsi keanggotan. Pada logika *Boolean*, sebuah individu dipastikan menjadi anggota salah satu himpunan saja. Sedangkan dalam himpunan *Fuzzy* adalah hal yang memungkinkan terdapat individu yang menjadi anggota dari dua himpunan yang berbeda, yang seberapa besar eksistensinya dapat dilihat dari nilai keanggotaan yang dimilikinya (Yulianto, 2008).

2.7 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 (Kusumadewi et al*.,* 2010), 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]. Suatu *item* dalam semesta pembicaraan tidak hanya berada pada 0 atau 1, namun juga memiliki nilai yang berada diantaranya, hal tersebut ditunjukkan dari nilai keanggotaannya. Sehingga dapat dikatakan bahwa nilai kebenaran suatu item bukan hanya benar atau salah. Nilai 0 menunjukkan salah, nilai 1 menunjukkan benar, serta terdapat nilai-nilai yang terletak antara benar dan salah. Himpunan *Fuzzy* memiliki 2 atribut, yaitu (Kusumadewi et al*.,* 2003):

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

2.8 *Fuzzy k-Nearest Neighbor*

2.8.1 Pengertian *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. Metode klasifikasi dalam *Fk-NN* menggunakan nilai derajat keanggotaan data uji pada setiap kelas untuk melakukan prediksi terhadap data uji. Selanjutnya kelas dengan nilai derajat keanggotaan terbesar dari data uji diambil sebagai kelas hasil prediksi (Keller et al., 1985).

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* konvensional dengan mendefinisikan nilai keanggotaan sebuah data pada masing-masing kelas. Persamaan yang digunakan ditunjukkan pada persamaan 2.3 berikut ini (Keller et al., 1985).

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

Keterangan:

u_i = Fungsi Fuzzy k-nearest neighbor

u_{ij} = Nilai keanggotaan Fuzzy pada contoh pengujian

K = Banyaknya nilai ketetanggaan terdekat

j = Variabel data untuk keanggotaan data latih

i = Variabel data untuk keanggotaan data uji

m = Berat kebalikan yang sebanding dengan jarak antara xj dan x

Variabel (m) merupakan penentuan seberapa banyak pemberian bobot pada jarak saat menghitung kontribusijarak 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) (Keller et al., 1985).

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 (Keller et al., 1985).

Nilai u_{ij} pada $u_i(x)$ terlebih dahulu diproses dengan menggunakan persamaan 2.4 (Keller et al., 1985).

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

Keterangan:

n_j= Jumlah anggota kelas j pada suatu *dataset* K

K= Total data latih yang digunakan

j= Kelas target (*training*/tidak *training*)

2.8.2 Proses *Fuzzy k-Nearest Neighbor*

Tahapan proses yang dilakukan pada algoritma *Fuzzy k-Nearest Neighbor* adalah:

1. Melakukan normalisasi terhadap nilai-nilai atribut menggunakan normalisasi *min-max* yang ditunjukkan oleh persamaan 2.1.
2. Menghitung jarak antara dua *record* menggunakan *Euclidean Distance* yang ditunjukkan oleh persamaan 2.2.
3. Mengambil nilai terbesar dari proses nomer 3 untuk semua $1 \leq i \leq C$, C adalah jumlah kelas.
4. Menghitung nilai keanggotaan u_i (x) menggunakan persamaan 2.3 untuk setiap i, dimana $1 \leq i \leq C$, C adalah jumlah kelas.
5. Memberikan label kelas baru pada proses nomer 4.

2.9 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 2.3 (Nugraha, 2006).

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

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).

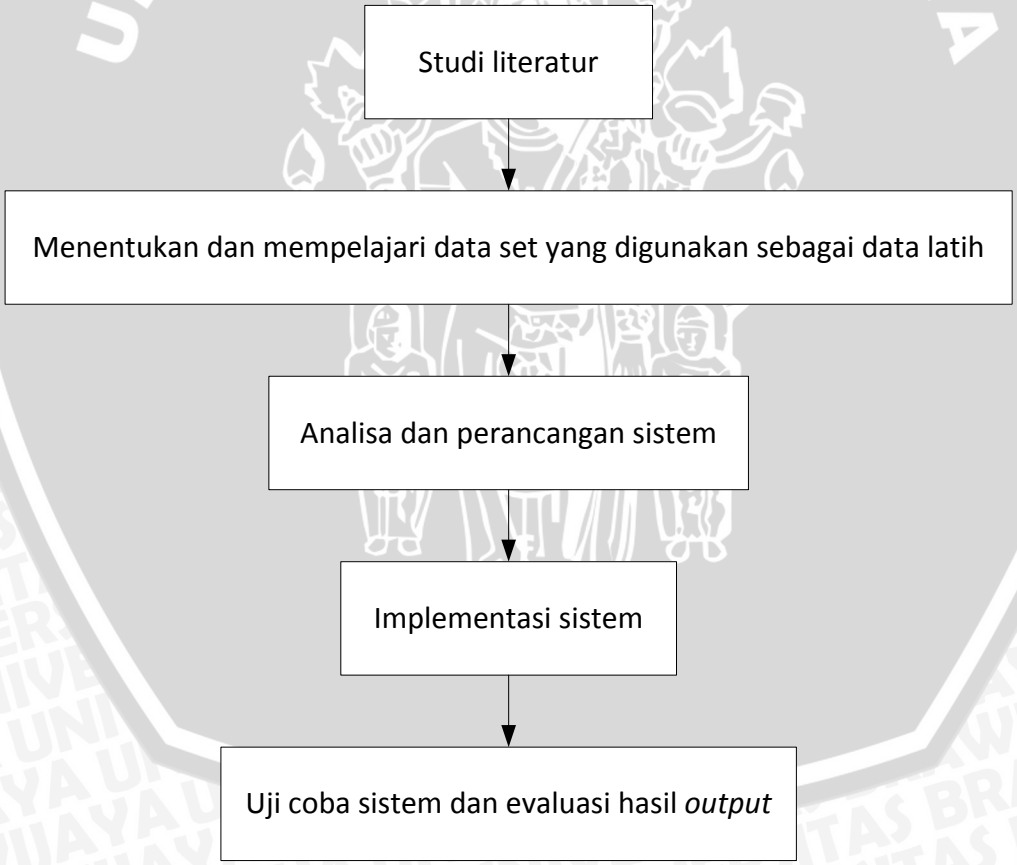
BAB 3 METODOLOGI

Pada bab ini dijelaskan tentang tahapan-tahapan perancangan sistem yang dibangun dalam penelitian ini dengan menerapkan metode *Fuzzy k-Nearest Neighbor* yang digunakan untuk pengklasifikasian penyakit ginjal kronis.

Langkah-langkah yang dilakukan dalam penelitian ini adalah sebagai berikut:

1. Mempelajari sumber-sumber yang mendukung untuk mendalami materi mengenai penyakit ginjal kronis dan metode *Fuzzy k-Nearest Neighbor.*
2. Mengumpuklan data-data dari *Chronic Kidney Disease Dataset.*
3. Menganalisis data-data yang terdapat pada *Chronic Kidney Disease Dataset.*
4. Mengimplementasikan sistem untuk membuat perangkat lunak berdasarkan analisis dan perancangan yang telah dilakukan dengan bahasa pemrograman *Java*.
5. Melakukan uji coba terhadap perangkat lunak.
6. Mengevaluasi uji coba terhadap perangkat lunak.

Langkah-langkah penelitian ini dapat dilihat pada gambar 3.1



Gambar 3.1 Langkah-langkah penelitian

3.1 Studi literatur

Studi literature dilakukan sebagai pendalaman materi guna mencapai tujuan dan penyelesaian masalah yang ada dalam penelitian ini. Konsep dan teori mengenai penyakit ginjal kronis dan metode *Fuzzy k-Nearest Neighbor* yang dijadikan sebagai dasar penelitian ini berasal dari berbagai sumber, mulai buku-buku, jurnal nasional maupun internasional, dan juga sumber dari internet.

3.2 Data penelitian

Dalam penelitian ini penulis menggunakan *Chronic Kidney Disease Dataset* tahun 2015 yang diperoleh dari website *UCI Machine Learning Repository*. Pada dataset tersebut, terdapat 24 atribut yang digunakan, yaitu *age, blood pressure, specific gravity, albumin, sugar, red blood cells, pus cell, pus cell clumps, bacteria, blood glucose random, blood urea, serum creatinine, sodium, potassium, hemoglobin, packed cell volume, white blood cell count, red blood cell count, hypertension, diabetes mellitus, coronary artery disease, appetite, pedal edema,* dan *anemia.* Dengan 2 kelas *output* yaitu 1 *(ckd)* dan 2 *(notckd).*

3.3 Analisa dan perancangan sistem

3.3.1 Deskripsi umum sistem

Pada dasarnya sistem yang dibangun adalah perangkat lunak yang menerapkan metode *Fuzzy k-Nearest Neighbor* (*Fk-NN*) untuk pengklasifikasian penyakit ginjal kronis atau *chronic kidney disease*. Selain itu perangkat lunak yang dibangun juga akan melakukan pengujian terhadap tingkat akurasi terhadap hasil pengklasifikasian penyakit ginjal kronis yang telah dilakukan oleh sistem.

3.3.2 Batasan sistem

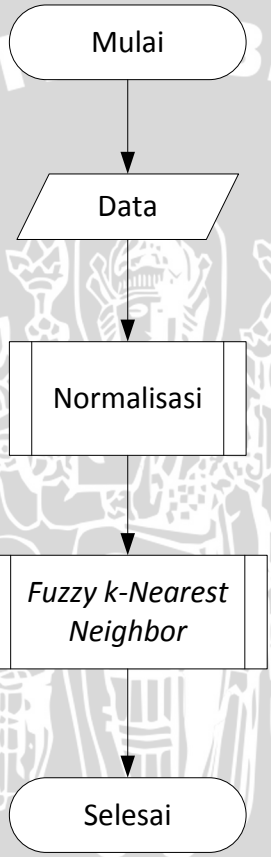
Yang menjadi batasan sistem dalam penelitian mengenai klasifikasi penyakit ginjal kronis ini adalah input sistem berupa *dataset* pasien penderita penyakit ginjal kronis atau *cronic kidney disease*.

3.3.3 Perancangan sistem

Berikut adalah langkah–langkah pada tahap klasifikasi yang diterapkan dalam sistem pengklasifikasian penyakit ginjal kronis dengan metode *Fk-NN*.

1. Menginputkan data latih dan data uji penderita penyakit ginjal kronis.
2. Menormalisasi setiap atribut pada data yang diinputkan.
3. Memproses data yang telah dinormalisasi dengan perhitungan *k-NN*.
4. Mentransformasikan hasil perhitungan *k-NN* kedalam data *Fuzzy.*
5. Menampilkan hasil klasifikasi kelas penyakit ginjal kronis.

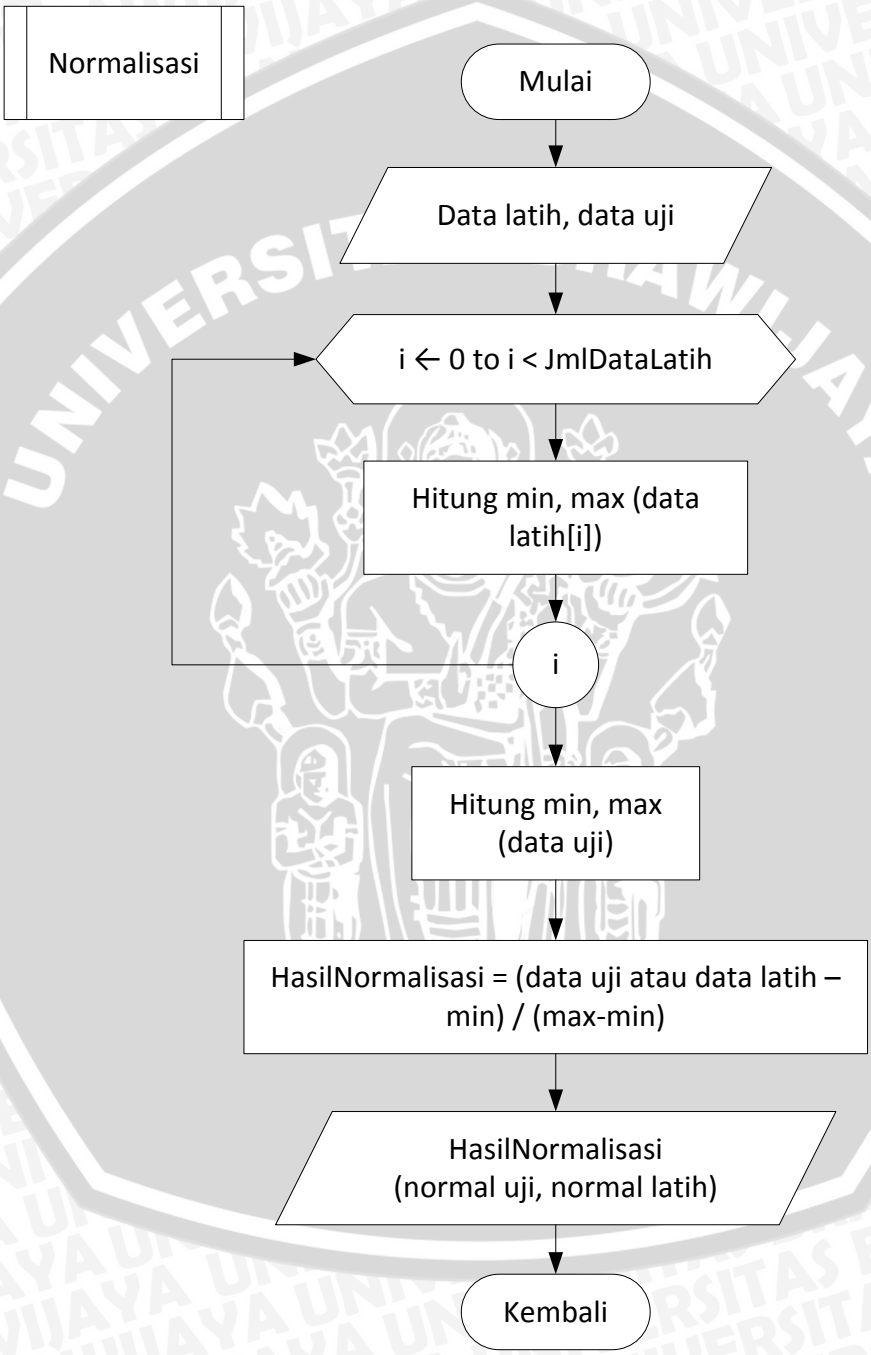
Gambar 3.2 dibawah ini menunjukkan alur dari tahap klasifikasi.



Gambar 3.2 Alur proses klasifikasi

3.3.3.1 Normalisasi *chronic kidney disease dataset*

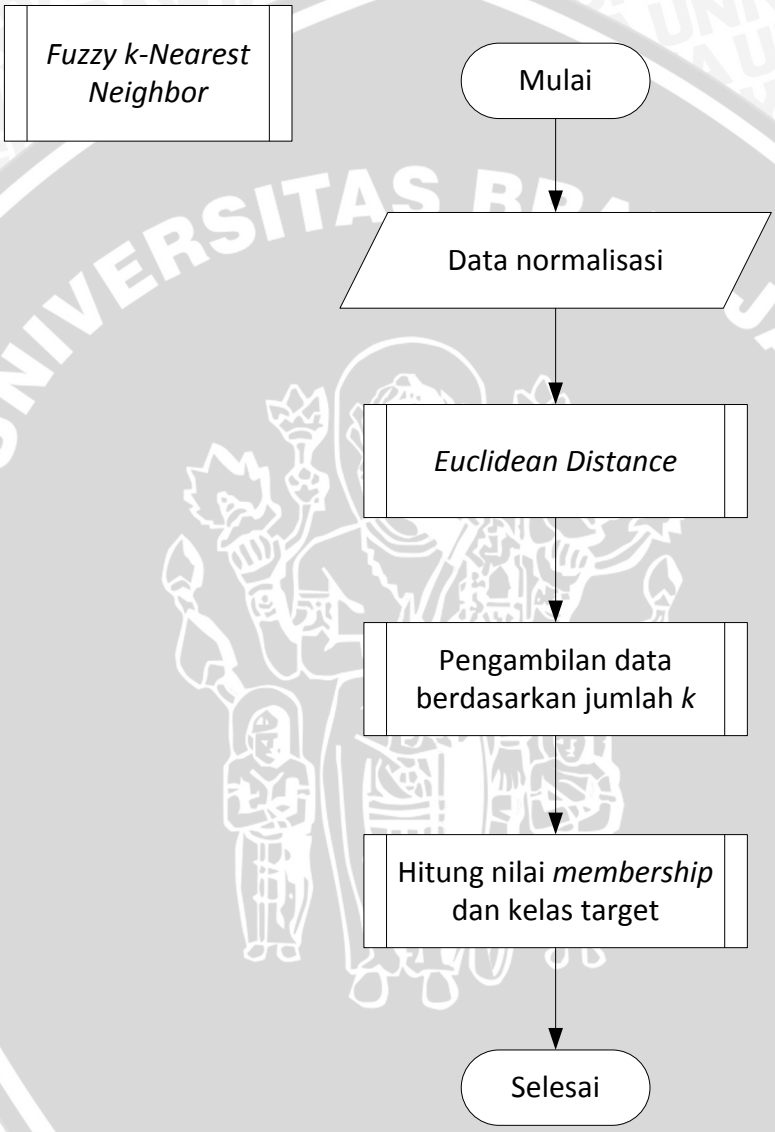
Pada penelitian ini digunakan *min-max normalization* untuk proses normalisasi data. Normalisasi bertujuan untuk memberikan skala pengaruh pada tiap atribut terhadap hasil dengan mentransformasikan data berupa normalisasi terhadap nilai dari setiap atribut. Gambar 3.3 menunjukkan alur proses normalisasi data.



Gambar 3.3 Alur proses normalisasi data

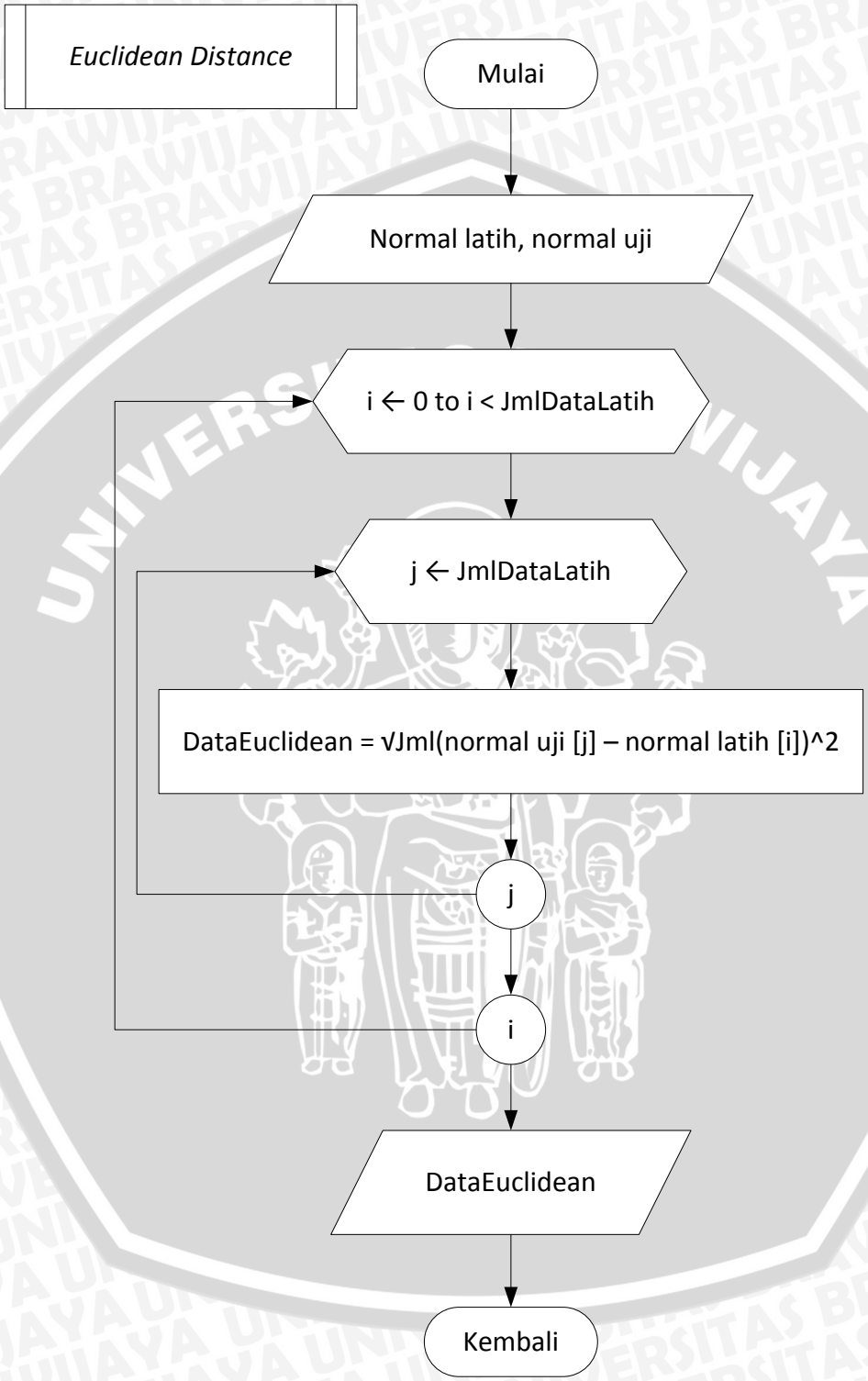
3.3.3.2 Proses *Fuzzy k-Nearest Neighbor*

Proses ini merupakan proses perhitungan nilai *membership* dengan 2 kelas, yaitu kelas *ckd* atau pasien penderita penyakit ginjal kronis dan kelas *notckd* atau pasien bukan penderita penyakit ginjal kronis. Pada tahap ini digunakan fungsi *Euclidean Distance* untuk menghitung jarak kedekatan tetangga data uji terhadap data latih. Gambar 3.4 di bawah ini menunjukkan alur dari proses *Fuzzy k-Nearest Neighbor*.



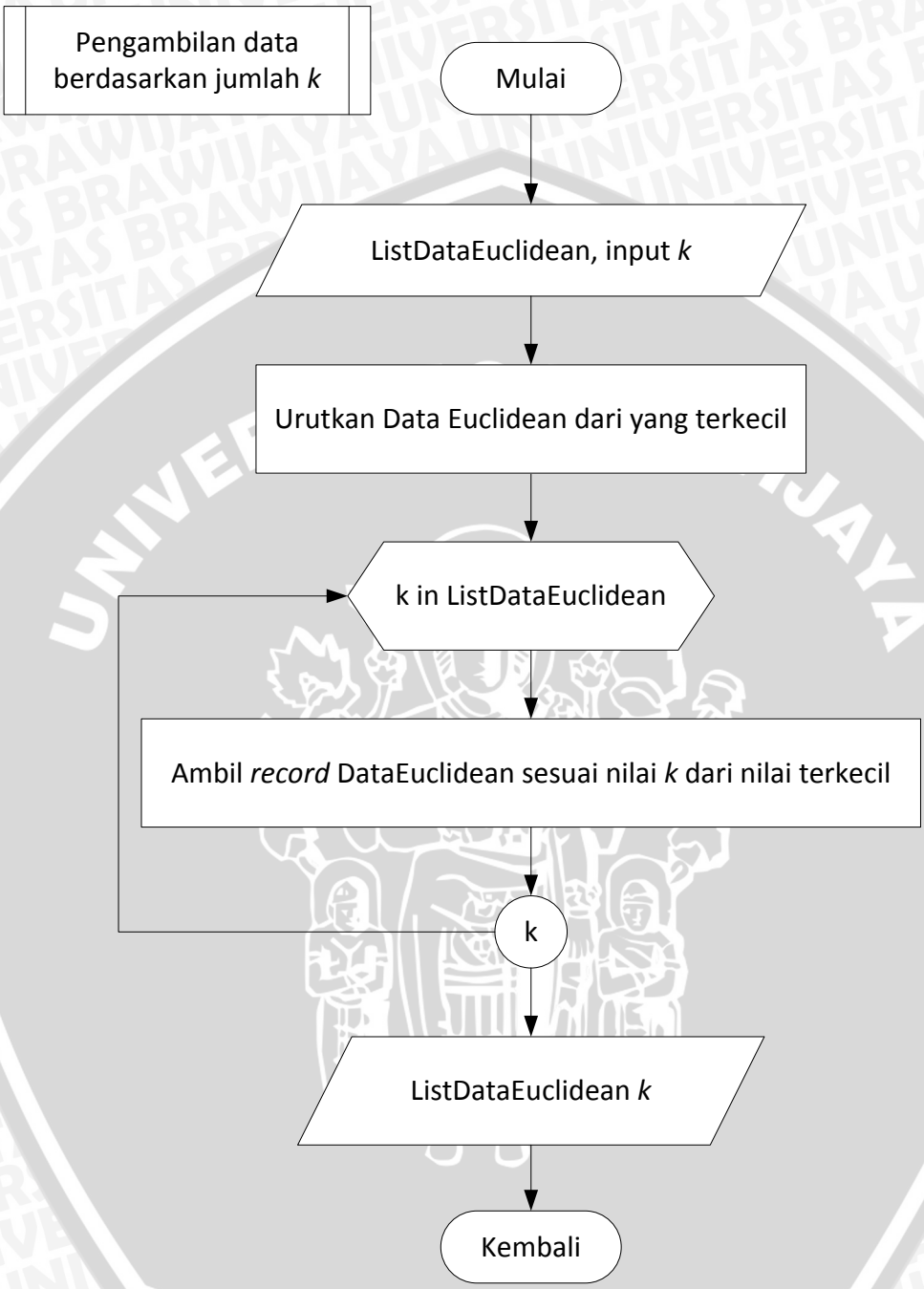
Gambar 3.4 Alur proses *Fuzzy k-Nearest Neighbor*

Untuk alur proses perhitungan *Euclidean Distance* sendiri dapat dilihat pada gambar 3.5 di bawah ini.



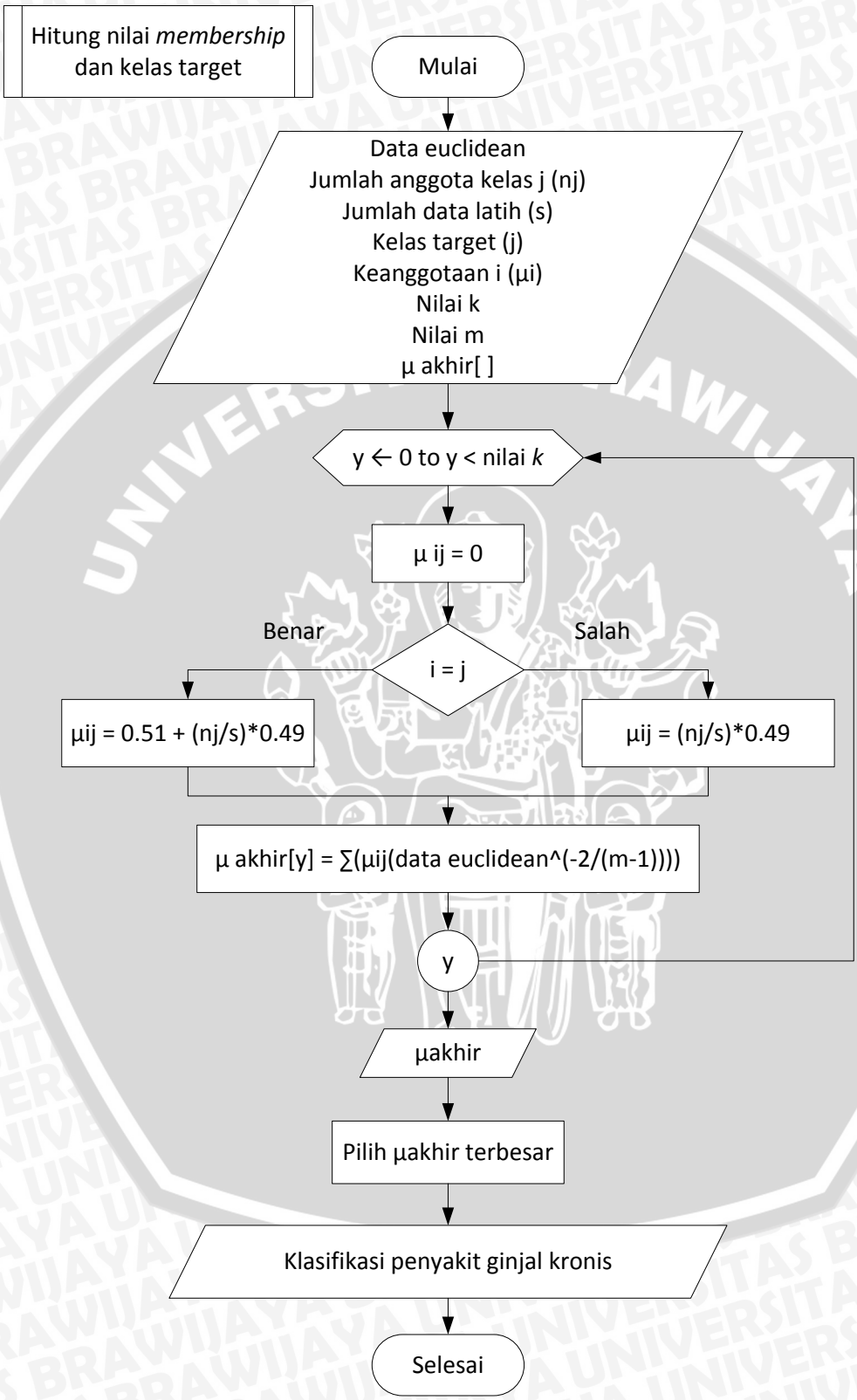
Gambar 3.5 Alur proses *Euclidean Distance*

Pada gambar 3.6 di bawah ini diperlihatkan bagaimana alur dari proses pengambilan data berdasarkan jumlah *k*.



Gambar 3.6 Alur proses pengambilan data berdasarkan jumlah *k*

Diagram alir untuk proses perhitungan *membership* dan kelas target ditunjukkan oleh gambar 3.7



Gambar 3.7 Alur proses perhitungan *membership* dan kelas target

3.4 Contoh perhitungan manual

Konsep perhitungan yang dilakukan dalam sistem pengklasifikasian penyakit ginjal kronis dengan metode *Fk-NN* ini dapat dilihat pada contoh perhitungan manual yang disajikan di bawah ini, contoh perhitungan manual di bawah ini menggunakan seluruh atribut *dataset* yang berjumlah 24 atribut, serta data yang digunakan sejumlah 90 *instances*, dengan 60 *instance* sebagai data latih dan 30 *instances* sebagai data uji.

3.4.1 Data latih dan data uji pada penderita penyakit ginjal kronis

Tabel 3.1 dan tabel 3.2 di bawah ini menunjukkan 60 data latih dan 30 data uji yang digunakan dalam contoh perhitungan manual yang dilakukan kali ini.

Tabel 3.1 Data latih sistem

no	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wbcc	rbcc	htn	dm	cad	appet	pe	ane	kelas
1	48	70	1.005	4	0	0.575	1	1	0.603	117	56	3.8	111	2.5	11.2	32	6700	3.9	1	0.43	0.59	1	1	1	1
2	53	90	1.02	2	0	1	1	1	0.603	70	107	7.2	114	3.7	9.5	29	12100	3.7	1	1	0.59	1	0.537	1	1
3	63	70	1.01	3	0	1	1	1	0.603	380	60	2.7	131	4.2	10.8	32	4500	3.8	1	1	0.59	1	1	0.559	1
4	68	80	1.01	3	2	0.575	1	1	1	157	90	4.1	130	6.4	5.6	16	11000	2.6	1	1	1	1	1	0.559	1
5	46	60	1.01	1	0	0.575	0.537	0.581	0.603	163	92	3.3	141	4	9.8	28	14600	3.2	1	1	0.59	0.528	0.537	0.559	1
6	56	90	1.015	2	0	1	1	0.581	0.603	129	107	6.7	131	4.8	9.1	29	6400	3.4	1	0.43	0.59	0.528	0.537	0.559	1
7	48	80	1.005	4	0	1	1	0.581	1	133	139	8.5	132	5.5	10.3	36	6200	4	0.407	1	0.59	0.528	1	0.559	1
8	59	70	1.01	3	0	0.575	1	0.581	0.603	76	186	15	135	7.6	7.1	22	3800	2.1	1	0.43	0.59	1	1	1	1
9	63	100	1.01	2	2	0.575	0.537	0.581	1	280	35	3.2	143	3.5	13	40	9800	4.2	1	0.43	1	0.528	0.537	0.559	1
10	56	70	1.015	4	1	1	0.537	0.581	0.603	210	26	1.7	136	3.8	16.1	52	12500	5.6	0.407	0.43	0.59	0.528	0.537	0.559	1

Tabel 3.2 Data uji sistem

no	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wbcc	rbcc	htn	dm	cad	appet	pe	ane	kelas
1	61	80	1.015	2	0	1	1	0.581	0.603	173	148	3.9	135	5.2	7.7	24	9200	3.2	1	1	1	1	1	1	1
2	48	80	1.025	4	0	0.575	1	0.581	0.603	95	163	7.7	136	3.8	9.8	32	6900	3.4	1	0.43	0.59	0.528	0.537	1	1
3	69	70	1.01	3	4	0.575	1	0.581	0.603	264	87	2.7	130	4	12.5	37	9600	4.1	1	1	1	0.528	1	0.559	1
4	73	70	1.005	0	0	0.575	0.537	0.581	0.603	70	32	0.9	125	4	10	29	18900	3.5	1	1	0.59	0.528	1	0.559	1
5	73	80	1.02	2	0	1	1	0.581	0.603	253	142	4.6	138	5.8	10.5	33	7200	4.3	1	1	1	0.528	0.537	0.559	1
6	59	80	1.01	1	0	1	0.537	0.581	0.603	303	35	1.3	122	3.5	10.4	35	10900	4.3	0.407	1	0.59	1	0.537	0.559	1
7	83	70	1.02	3	0	0.575	0.537	0.581	0.603	102	60	2.6	115	5.7	8.7	26	12800	3.1	1	0.43	0.59	1	0.537	1	1
8	21	90	1.01	4	0	0.575	1	1	1	107	40	1.7	125	3.5	8.3	23	12400	3.9	0.407	0.43	0.59	0.528	0.537	1	1
9	45	70	1.025	2	0	0.575	1	1	0.603	117	52	2.2	136	3.8	10	30	19100	3.7	0.407	0.43	0.59	0.528	0.537	0.559	1
10	64	60	1.01	4	1	1	1	0.581	1	239	58	4.3	137	5.4	9.5	29	7500	3.4	1	1	0.59	1	1	0.559	1

Data-data yang digunakan dalam perhitungan seperti yang ditampilkan pada tabel diatas memiliki dua kelas sesuai dengan nilai target output, kedua kelas itu yakni kelas *ckd* atau pasien penderita penyakit ginjal kronis (1) dan kelas *notckd* atau pasien bukan penderita penyakit ginjal kronis (2).

3.4.2 Normalisasi terhadap nilai atribut

Normalisasi yang dilakukan terhadap nilai tiap atribut data yang digunakan adalah normalisasi min-max. Sebelum melakukan normalisasi terhadap nilai-nilai atribut, terlebih dahulu dicari nilai minimum-maksimum data latih dan nilai minimum-maksimum data uji dari setiap atribut. Berikut ini adalah contoh perhitungan normalisasi nilai record pertama untuk atribut ke-1 (*age*).

V = 48 (nilai *record* pertama atribut *age*)

min_A = 20

max_A = 73

Setelah diketahui nilai minimum dan maksimum maka dilakukan perhitungan menggunakan persamaan 2.1.

$$v' = \frac{v - \min_a}{\max_a - min_a}$$
$$= \frac{48-20}{73-20}$$
$$= 0.528$$

Dari perhitungan tersebut diperoleh nilai normalisasi data pertama untuk atribut *age* yaitu 0.528. Selanjutnya proses yang sama juga dilakukan terhadap nilai – nilai dari atribut yang lainnya. Hasil perhitungan untuk seluruh atribut ditampilkan pada tabel 3.3 dan tabel 3.4.

Tabel 3.3 Data latih setelah normalisasi

no	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wbcc	rbcc	htn	dm	cad	appet	pe	ane	kelas
1	0.528	0.200	0.000	1.000	0.000	0.000	1.000	1.000	0.000	0.133	0.154	0.224	0.000	0.000	0.570	0.535	0.232	0.305	1.000	0.000	0.000	1.000	1.000	1.000	1
2	0.623	0.600	0.750	0.500	0.000	1.000	1.000	1.000	0.000	0.000	0.324	0.456	0.077	0.027	0.451	0.465	0.664	0.271	1.000	1.000	0.000	1.000	0.000	1.000	1
3	0.811	0.200	0.250	0.750	0.000	1.000	1.000	1.000	0.000	0.876	0.167	0.150	0.513	0.038	0.542	0.535	0.056	0.288	1.000	1.000	0.000	1.000	1.000	0.000	1
4	0.906	0.400	0.250	0.750	0.667	0.000	1.000	1.000	1.000	0.246	0.268	0.245	0.487	0.088	0.176	0.163	0.576	0.085	1.000	1.000	1.000	1.000	1.000	0.000	1
5	0.491	0.000	0.250	0.250	0.000	0.000	0.000	0.000	0.000	0.263	0.274	0.190	0.769	0.034	0.472	0.442	0.864	0.186	1.000	1.000	0.000	0.000	0.000	0.000	1
6	0.679	0.600	0.500	0.500	0.000	1.000	1.000	0.000	0.000	0.167	0.324	0.422	0.513	0.052	0.423	0.465	0.208	0.220	1.000	0.000	0.000	0.000	0.000	0.000	1
7	0.528	0.400	0.000	1.000	0.000	1.000	1.000	0.000	1.000	0.178	0.431	0.544	0.538	0.067	0.507	0.628	0.192	0.322	0.000	1.000	0.000	0.000	1.000	0.000	1
8	0.736	0.200	0.250	0.750	0.000	0.000	1.000	0.000	0.000	0.017	0.589	0.986	0.615	0.115	0.282	0.302	0.000	0.000	1.000	0.000	0.000	1.000	1.000	1.000	1
9	0.811	0.800	0.250	0.500	0.667	0.000	0.000	0.000	1.000	0.593	0.084	0.184	0.821	0.022	0.697	0.721	0.480	0.356	1.000	0.000	1.000	0.000	0.000	0.000	1
10	0.679	0.200	0.500	1.000	0.333	1.000	0.000	0.000	0.000	0.395	0.054	0.082	0.641	0.029	0.915	1.000	0.696	0.593	0.000	0.000	0.000	0.000	0.000	0.000	1

Tabel 3.4 Data uji setelah normalisasi

no	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	bu	sc	sod	pot	hemo	pcv	wbcc	rbcc	htn	dm	cad	appet	pe	ane	kelas
1	0.714	0.600	0.500	0.500	0.000	1.000	1.000	0.000	0.000	0.245	0.762	0.276	0.636	0.793	0.041	0.125	0.222	0.031	1.000	1.000	1.000	1.000	1.000	1.000	1
2	0.545	0.600	1.000	1.000	0.000	0.000	1.000	0.000	0.000	0.060	0.845	0.585	0.667	0.310	0.258	0.375	0.118	0.094	1.000	0.000	0.000	0.000	0.000	1.000	1
3	0.818	0.400	0.250	0.750	0.800	0.000	1.000	0.000	0.000	0.462	0.425	0.179	0.485	0.379	0.536	0.531	0.240	0.313	1.000	1.000	1.000	0.000	1.000	0.000	1
4	0.870	0.400	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.122	0.033	0.333	0.379	0.278	0.281	0.661	0.125	1.000	1.000	0.000	0.000	1.000	0.000	1
5	0.870	0.600	0.750	0.500	0.000	1.000	1.000	0.000	0.000	0.436	0.729	0.333	0.727	1.000	0.330	0.406	0.131	0.375	1.000	1.000	1.000	0.000	0.000	0.000	1
6	0.688	0.600	0.250	0.250	0.000	1.000	0.000	0.000	0.000	0.555	0.138	0.065	0.242	0.207	0.320	0.469	0.299	0.375	0.000	1.000	0.000	1.000	0.000	0.000	1
7	1.000	0.400	0.750	0.750	0.000	0.000	0.000	0.000	0.000	0.076	0.276	0.171	0.030	0.966	0.144	0.188	0.385	0.000	1.000	0.000	0.000	1.000	0.000	1.000	1
8	0.195	0.800	0.250	1.000	0.000	0.000	1.000	1.000	1.000	0.088	0.166	0.098	0.333	0.207	0.103	0.094	0.367	0.250	0.000	0.000	0.000	0.000	0.000	1.000	1
9	0.506	0.400	1.000	0.500	0.000	0.000	1.000	1.000	0.000	0.112	0.232	0.138	0.667	0.310	0.278	0.313	0.670	0.188	0.000	0.000	0.000	0.000	0.000	0.000	1
10	0.753	0.200	0.250	1.000	0.200	1.000	1.000	0.000	1.000	0.402	0.265	0.309	0.697	0.862	0.227	0.281	0.145	0.094	1.000	1.000	0.000	1.000	1.000	0.000	1

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

Dari data latih dan data uji yang telah dinormalisasi dilakukan perhitungan untuk mencari jarak terdekat menggunakan fungsi *Euclidean Distance* seperti yang telah dituliskan pada persamaan (2.2). Di bawah ini contoh perhitungan nilai jarak terdekat dari *record* pertama data latih dengan *record* data uji.

$$d_i = \sqrt{\sum_{i=1}^{p} x_{2i} - x_{1i}^{\ 2}}$$

$$d_i = \sqrt{\begin{array}{l} 0.528 - 0.714^{\ 2} + 0.200 - 0.600^{\ 2} + 0.000 - 0.500^{\ 2} + \\ 1.000 - 0.500^{\ 2} + 0.000 - 0.000^{\ 2} + 0.000 - 1.000^{\ 2} + \\ 1.000 - 0.000^{\ 2} + 1.000 - 1.000^{\ 2} + 0.000 - 0.000^{\ 2} + \\ 0.133 - 0.245^{\ 2} + 0.154 - 0.762^{\ 2} + 0.224 - 0.276^{\ 2} + \\ 0.000 - 0.636^{\ 2} + 0.000 - 0.793^{\ 2} + 0.570 - 0.041^{\ 2} + \\ 0.535 - 0.125^{\ 2} + 0.232 - 0.222^{\ 2} + 0.305 - 0.031^{\ 2} + \\ 1.000 - 1.000^{\ 2} + 0.000 - 1.000^{\ 2} + 0.000 - 1.000^{\ 2} + \\ 1.000 - 1.000^{\ 2} + 1.000 - 1.000^{\ 2} + 1.000 - 1.000^{\ 2} \end{array}}$$

$$d_i = 2.576$$

Kemudian dilakukan perhitungan yang sama untuk data yang lain. Hasil perhitungan jarak ditampilkan pada tabel (3.5).

Tabel 3.5 Hasil *Euclidean Distance*

data no	nilai target	kelas
1	2.576318437	1
2	2.188956897	1
3	2.271532493	1
4	2.357938996	1
5	2.845667079	1
6	2.466243861	1
7	2.617617568	1
8	2.114720777	1
9	3.119640789	1
10	3.277143863	1

Setelah hasil perhitungan *Euclidean Distance* didapatkan maka proses selanjutnya adalah data tersebut diurutkan mulai dari nilai paling kecil sampai yang paling besar seperti yang ditunjukkan pada tabel 3.6 berikut ini.

Tabel 3.6 Hasil *Euclidean Distance* setelah diurutkan dari yang terkecil

data no	nilai target	kelas
19	1.697487473	1
23	2.109706943	1
8	2.114720777	1
2	2.188956897	1
12	2.24226794	1
3	2.271532493	1
4	2.357938996	1
14	2.360664215	1
6	2.466243861	1
13	2.54824894	1

3.4.4 Menentukan k-*record* terdekat

Setelah didapatkan hasil perhitungan *Euclidean Distance* yang telah diurutkan seperti yang ditunjukkan pada tabel 3.6, kemudian ditentukan *k record* dengan nilai terkecil berdasarkan hasil pengurutan yang telah dilakukan. Karena sebelumnya telah ditentukan untuk nilai *k* yang digunakan adalah 3, maka *record* yang digunakan untuk proses perhitungan selanjutnya adalah *record* ke-19, ke-23, dan ke-8.

Tabel 3.7 Tiga data hasil *Euclidean Distance* yang digunakan berdasarkan pengurutan terkecil

data no.	nilai target	kelas
19	1.697487473	1
23	2.109706943	1
8	2.114720777	1

3.4.5 Menentukan maksimum *membership* dan kelas target

Proses selanjutnya adalah menentukan maksimum *membership* dan kelas target tiap kelas j dengan menerapkan persamaan 2.4 dengan jumlah data latih atau K=60, u_1 = pasien penderita penyakit ginjal kronis, dan u_2 = pasien bukan penderita penyakit ginjal kronis.

μ1(1)=0.51+(n1/K)*0.49
μ1(1)=0.51+(23/60)*0.49
μ1(1)=0.697833333

μ1(2)=(n1/K)*0.49
μ1(2)=(23/60)*0.49
μ1(2)=0.187833333

μ2(2)=0.51+(n2/K)*0.49
μ2(2)=0.51+(37/60)*0.49
μ2(2)=0.812166667

μ2(1)=(n2/K)*0.49
μ2(1)=(37/60)*0.49
μ2(1)=0.302166667

Proses perhitungan selanjutnya adalah mencari nilai keanggotaan sebuah data pada masing-masing kelas. Pada proses ini digunakan persamaan 2.3.

$$u_1 = \frac{0.697833333 * 1.697487473^{-\frac{2}{2-1}} + 0.697833333 * 2.109706943^{-\frac{2}{2-1}} + 0.697833333 * 2.114720777^{-\frac{2}{2-1}}}{1.697487473^{-\frac{2}{2-1}} + 2.109706943^{-\frac{2}{2-1}} + 2.114720777^{-\frac{2}{2-1}}}$$

$$u_1 = 0.697833333$$

$$u_2 = \frac{0.302166667 * 1.697487473^{-\frac{2}{2-1}} + 0.302166667 * 2.109706943^{-\frac{2}{2-1}} + 0.302166667 * 2.114720777^{-\frac{2}{2-1}}}{1.697487473^{-\frac{2}{2-1}} + 2.109706943^{-\frac{2}{2-1}} + 2.114720777^{-\frac{2}{2-1}}}$$

$$u_2 = 0.302166667$$

Langkah selanjutnya adalah menentukan kelas target berdasarkan hasil perhitungan nilai keanggotaan yang telah dilakukan. Dari dua nilai keanggotaan, yang didapatkan, dipilih nilai keanggotaan paling besar yakni 0.697833333. Maka kelas target yang didapat adalah 1 (pasien penderita penyakit ginjal kronis).

Setelah dilakukan perhitungan pada satu data uji seperti yang telah dicontohkan di atas, selanjutnya dilakukan perhitungan terhadap data uji yang lain dengan langkah-langkah yang sama seperti contoh perhitungan di atas. Sehingga setelah semua data uji telah diproses dan diperoleh hasilnya, akan dapat dilihat akurasi sistem yang dengan menerapkan persamaan 2.5.

Tebel 3.8 berikut menunjukkan proses perhitungan akurasi berdasarkan hasil yang telah diperoleh dari proses perhitungan seluruh data uji.

Tabel 3.8 Perhitungan akurasi sistem

Data Uji Ke-	Kelas Data	Kelas Hasil Perhitungan	Hasil Prediksi
1	1	1	benar
2	1	1	benar
3	1	1	benar
4	1	1	benar
5	1	1	benar
6	1	1	benar
7	1	1	benar
8	1	1	benar
9	1	2	salah
10	1	1	benar
11	1	1	benar
12	1	1	benar
13	1	1	benar
14	1	1	benar
15	1	1	benar
16	1	1	benar
17	1	1	benar
18	1	1	benar
19	1	1	benar
20	1	1	benar
21	2	2	benar
22	2	2	benar
23	2	2	benar
24	2	2	benar
25	2	2	benar
26	2	2	benar

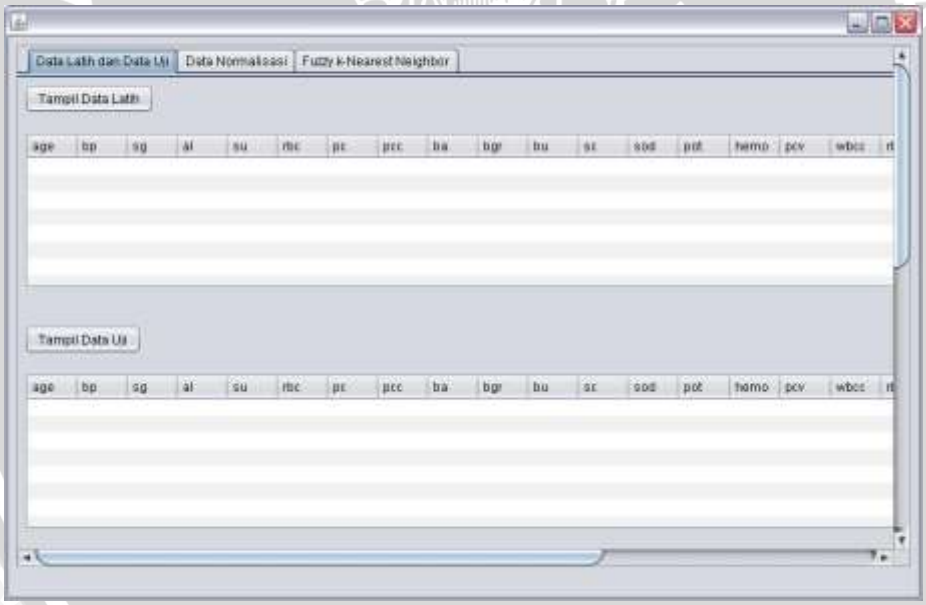
Tabel 3.9 Perhitungan akurasi sistem (lanjutan)

Data Uji Ke-	Kelas Data	Kelas Hasil Perhitungan	Hasil Prediksi
27	2	2	benar
28	2	2	benar
29	2	2	benar
30	2	2	benar

$$Akurasi = \frac{29}{30} \times 100\%$$
$$= 96{,}667\%$$

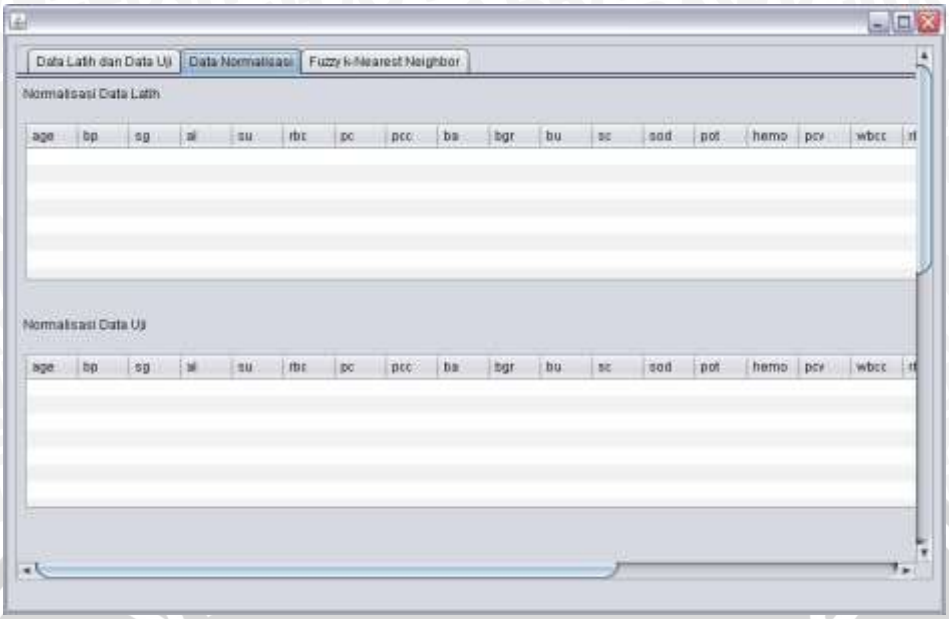
3.5 Perancangan antarmuka

Interface program yang digunakan untuk menjalankan sistem dirancang menjadi tiga bagian utama, yang pertama adalah halaman Data Latih dan Data Uji, yang kedua halaman Data normalisasi, dan yang ketiga adalah halaman *Fuzzy k-Nearest Neighbor*. Beberapa gambar berikut adalah tampilan dari perancangan *interface* program yang dibuat.



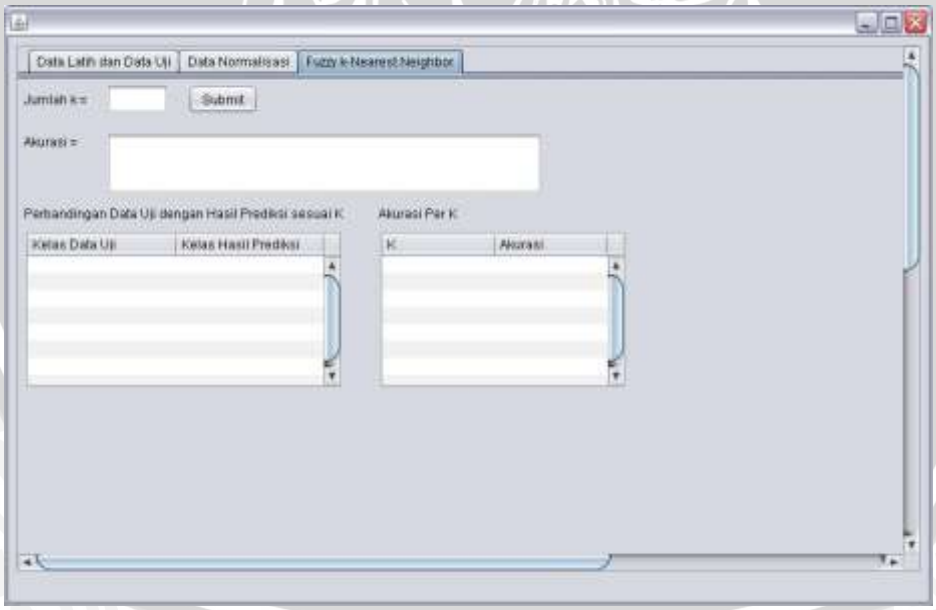
Gambar 3.8 Data latih dan data uji

1. Tombol Tampil Data Latih digunakan untuk mencari data latih yang ingin digunakan.
2. Tabel Data Latih digunakan untuk menampilkan data latih yang dipilih saat menekan tombol Tampil Data Latih.
3. Tombol Tampil Data Uji digunakan untuk mencari data uji yang ingin digunakan.
4. Tabel Data Uji digunakan untuk menampilkan data uji yang dipilih saat menekan tombol Tampil Data Uji.



Gambar 3.9 Data normalisasi

1. Tabel Data Latih yang telah dinormalisasi digunakan untuk menampilkan hasil normalisasi data latih.
2. Tabel Data Uji yang telah dinormalisasi digunakan untuk menampilkan hasil normalisasi dari data uji.



Gambar 3.10 *Fuzzy k-Nearest Neighbor*

1. *Textbox* untuk menampilkan nilai *k* yang dimasukkan.
2. Tombol *Submit* digunakan untuk melakukan proses *Fuzzy k-NN*.
3. *Textfield* untuk menampilkan hasil akurasi hasil pengujian dari proses *Fuzzy k-NN*.

3.6 Perancangan uji coba

Langkah selanjutnya dalam penelitian ini adalah melakukan pengujian terhadap sistem yang telah dibuat untuk mengetahui tingkat akurasi sistem dalam melakukan pengklasifikasian penyakit ginjal kronis berdasarkan *chronic kidney disease dataset* dengan penerapan metode *Fuzzy k-Nearest Neighbor*.

Pengujian yang dilakukan dalam penelitian ini adalah pengujian untuk mengetahui pengaruh nilai *k* (jumlah tetangga terdekat) dan data latih terhadap tingkat akurasi klasifikasi.

3.7 Uji pengaruh nilai *k* dan data latih terhadap tingkat akurasi klasifikasi

Pengujian tingkat akurasi sistem dalam hubungannya dengan pengaruh nilai *k* dan jumlah data latih yang digunakan dilakukan dengan menggunakan 30 data uji yang sama, sedangkan untuk data latihnya digunakan beberapa variasi jumlah yang digunakan. Nilai *k* yang akan digunakan dalam pengujian ini adalah *k* = 3 dan variasi jumlah data latih yang akan diujikan adalah 60, 80, 100, 120 dan 128 data. Akan diteliti apakah dengan jumlah data latih yang berbeda akan berpengaruh pada tingkat akurasi sistem.

Dari pengujian yang dilakukan akan didapatkan hasil tingkat akurasi terhadap pengaruhnya dengan nilai *k* dan jumlah data latih. Hasil pengujian akan dicatat kedalam tabel seperti tabel 3.9 di bawah ini.

Tabel 3.10 Uji Pengaruh nilai *k* dan data latih terhadap tingkat akurasi

k	Jumlah Data Latih	Akurasi (%)
3	60	
…	…	
n	n	

Keterangan:

k : nilai *k* yang diujikan.

Jumlah Data Latih : jumlah data yang diujikan.

Akurasi : hasil perhitungan tingkat akurasi

BAB 4 HASIL

4.1 Lingkungan implementasi

Beberapa aspek yang perlu diperhatikan dalam proses pembangunan sistem dengan menerapkan metode *Fuzzy K-nearest Neighbor (FK-NN)* untuk pengklasifikasian penyakit ginjal kronis ini anatara lain dari segi *hardware* dan *software* yang digunakan.

4.1.1 Lingkungan implementasi perangkat keras

Dalam pembuatan sistem yang dibangun dalam penelitian ini digunakan beberapa perangkat keras seperti yang disebutkan dibawah ini.

1. Processor : Intel(R) Atom(TM) CPU N475 @ 1.83GHz (2 CPUs)
2. Memory : 1 GB
3. Harddisk : 150 GB

4.1.2 Lingkungan implementasi perangkat lunak

Kebutuhan perangkat lunak yang digunakan untuk membangun sistem dalam penelitian ini disebutkan pada rincian di bawah ini.

1. Sistem Operasi yang digunakan Windows XP Professional (5.1, Build 2600) Service Pack 3
2. Aplikasi pembangunan GUI dan code menggunakan NetBeans IDE 6.8
3. Bahasa pemrograman yang dipakai yaitu bahasa pemrograman java
4. Komponen java yang digunakan yaitu Java SE Development Kit 7 update 17

4.2 Implementasi program

Berikut beberapa pemaparan tentang pengimplementasian metode dan perancangan yang digunakan sesuai yang telah dijelaskan pada bab sebelumnya kedalam program yang akan dibangun.

4.2.1 Proses baca *file*

Tahapan proses untuk membaca *file* yang akan digunakan dalam pelatihan dan pengujian data ditunjukkan pada *source code* 4.1.

Proses baca *file*

```
package ckd_fknn;


import java.io.File;
import java.util.ArrayList;
import java.util.List;
```

```
import jxl.Sheet;
import jxl.Workbook;

public class ReadData {

  private String path;
  private List<AtributData> lstAtributData = new ArrayList<AtributData>();
  private AtributData atributData;

  public ReadData (String path){

    this.path = path;

  }

  public List<AtributData> bacaData(){

    File file = new File(path);

   try{

    Workbook workbook = jxl.Workbook.getWorkbook(file);
    Sheet[] sheets = workbook.getSheets();
    int row = sheets[0].getRows();
    int column = sheets[0].getColumns();

    for (int i=1 ; i<row ; i++){

      atributData = new AtributData();
      atributData.setAge(Double.parseDouble(sheets[0].getCell(0,
i).getContents()));
      atributData.setBp(Double.parseDouble(sheets[0].getCell(1,
i).getContents()));
```

```
            atributData.setSg(Double.parseDouble(sheets[0].getCell(2,
i).getContents()));
            atributData.setAl(Double.parseDouble(sheets[0].getCell(3,
i).getContents()));
            atributData.setSu(Double.parseDouble(sheets[0].getCell(4,
i).getContents()));
            atributData.setRbc(Double.parseDouble(sheets[0].getCell(5,
i).getContents()));
            atributData.setPc(Double.parseDouble(sheets[0].getCell(6,
i).getContents()));
            atributData.setPcc(Double.parseDouble(sheets[0].getCell(7,
i).getContents()));
            atributData.setBa(Double.parseDouble(sheets[0].getCell(8,
i).getContents()));
            atributData.setBgr(Double.parseDouble(sheets[0].getCell(9,
i).getContents()));
            atributData.setBu(Double.parseDouble(sheets[0].getCell(10,
i).getContents()));
            atributData.setSc(Double.parseDouble(sheets[0].getCell(11,
i).getContents()));
            atributData.setSod(Double.parseDouble(sheets[0].getCell(12,
i).getContents()));
            atributData.setPot(Double.parseDouble(sheets[0].getCell(13,
i).getContents()));
            atributData.setHemo(Double.parseDouble(sheets[0].getCell(14,
i).getContents()));
            atributData.setPcv(Double.parseDouble(sheets[0].getCell(15,
i).getContents()));
            atributData.setWbcc(Double.parseDouble(sheets[0].getCell(16,
i).getContents()));
            atributData.setRbcc(Double.parseDouble(sheets[0].getCell(17,
i).getContents()));
            atributData.setHtn(Double.parseDouble(sheets[0].getCell(18,
i).getContents()));
            atributData.setDm(Double.parseDouble(sheets[0].getCell(19,
i).getContents()));
            atributData.setCad(Double.parseDouble(sheets[0].getCell(20,
i).getContents()));
```

```
            atributData.setAppet(Double.parseDouble(sheets[0].getCell(21,
    i).getContents()));
            atributData.setPe(Double.parseDouble(sheets[0].getCell(22,
    i).getContents()));
            atributData.setAne(Double.parseDouble(sheets[0].getCell(23,
    i).getContents()));
            atributData.setKelas(Integer.parseInt(sheets[0].getCell(24,
    i).getContents()));

            lstAtributData.add(atributData);
          }
          }
          catch (Exception ex)
          {
            System.out.println("Gagal"+ ex.getMessage());
          }

          return lstAtributData;
        }
    }
```

Source code* 4.1 Baca *file

4.2.2 Proses menampilkan data latih dan data uji

Untuk menampilkan data latih dan data uji yang digunakan dalam sistem digunakan proses yang ditunjukkan pada *source code* 4.2 dan *source code* 4.3 di bawah ini.

Proses menampilkan data latih

```
    private void
    btnTampilDataLatihActionPerformed(java.awt.event.ActionEvent evt) {

        BrowseData browseData = new BrowseData();

        browseData.choose();
        String pathDataLatih = browseData.getPath();
        ReadData readData = new ReadData(pathDataLatih);
```

```
lstAtributDataLatih = readData.bacaData();

namaKolomDataLatih = new String [25];

namaKolomDataLatih[0] = "age";
namaKolomDataLatih[1] = "bp";
namaKolomDataLatih[2] = "sg";
namaKolomDataLatih[3] = "al";
namaKolomDataLatih[4] = "su";
namaKolomDataLatih[5] = "rbc";
namaKolomDataLatih[6] = "pc";
namaKolomDataLatih[7] = "pcc";
namaKolomDataLatih[8] = "ba";
namaKolomDataLatih[9] = "bgr";
namaKolomDataLatih[10] = "bu";
namaKolomDataLatih[11] = "sc";
namaKolomDataLatih[12] = "sod";
namaKolomDataLatih[13] = "pot";
namaKolomDataLatih[14] = "hemo";
namaKolomDataLatih[15] = "pcv";
namaKolomDataLatih[16] = "wbcc";
namaKolomDataLatih[17] = "rbcc";
namaKolomDataLatih[18] = "htn";
namaKolomDataLatih[19] = "dm";
namaKolomDataLatih[20] = "cad";
namaKolomDataLatih[21] = "appet";
namaKolomDataLatih[22] = "pe";
namaKolomDataLatih[23] = "ane";
namaKolomDataLatih[24] = "kelas";

isiTabelDataLatih = new Object[lstAtributDataLatih.size()][25];
for (int i=0 ; i<lstAtributDataLatih.size();i++){
```

```
            isiTabelDataLatih [i][0]=lstAtributDataLatih.get(i).getAge();
            isiTabelDataLatih [i][1]=lstAtributDataLatih.get(i).getBp();
            isiTabelDataLatih [i][2]=lstAtributDataLatih.get(i).getSg();
            isiTabelDataLatih [i][3]=lstAtributDataLatih.get(i).getAl();
            isiTabelDataLatih [i][4]=lstAtributDataLatih.get(i).getSu();
            isiTabelDataLatih [i][5]=lstAtributDataLatih.get(i).getRbc();
            isiTabelDataLatih [i][6]=lstAtributDataLatih.get(i).getPc();
            isiTabelDataLatih [i][7]=lstAtributDataLatih.get(i).getPcc();
            isiTabelDataLatih [i][8]=lstAtributDataLatih.get(i).getBa();
            isiTabelDataLatih [i][9]=lstAtributDataLatih.get(i).getBgr();
            isiTabelDataLatih [i][10]=lstAtributDataLatih.get(i).getBu();
            isiTabelDataLatih [i][11]=lstAtributDataLatih.get(i).getSc();
            isiTabelDataLatih [i][12]=lstAtributDataLatih.get(i).getSod();
            isiTabelDataLatih [i][13]=lstAtributDataLatih.get(i).getPot();
            isiTabelDataLatih [i][14]=lstAtributDataLatih.get(i).getHemo();
            isiTabelDataLatih [i][15]=lstAtributDataLatih.get(i).getPcv();
            isiTabelDataLatih [i][16]=lstAtributDataLatih.get(i).getWbcc();
            isiTabelDataLatih [i][17]=lstAtributDataLatih.get(i).getRbcc();
            isiTabelDataLatih [i][18]=lstAtributDataLatih.get(i).getHtn();
            isiTabelDataLatih [i][19]=lstAtributDataLatih.get(i).getDm();
            isiTabelDataLatih [i][20]=lstAtributDataLatih.get(i).getCad();
            isiTabelDataLatih [i][21]=lstAtributDataLatih.get(i).getAppet();
            isiTabelDataLatih [i][22]=lstAtributDataLatih.get(i).getPe();
            isiTabelDataLatih [i][23]=lstAtributDataLatih.get(i).getAne();
            isiTabelDataLatih [i][24]=lstAtributDataLatih.get(i).getKelas();
        }
        tblDataLatih.setModel(new DefaultTableModel(isiTabelDataLatih,
namaKolomDataLatih));
```

***Source code* 4.2 Menampilkan data latih**

Proses menampilkan data uji

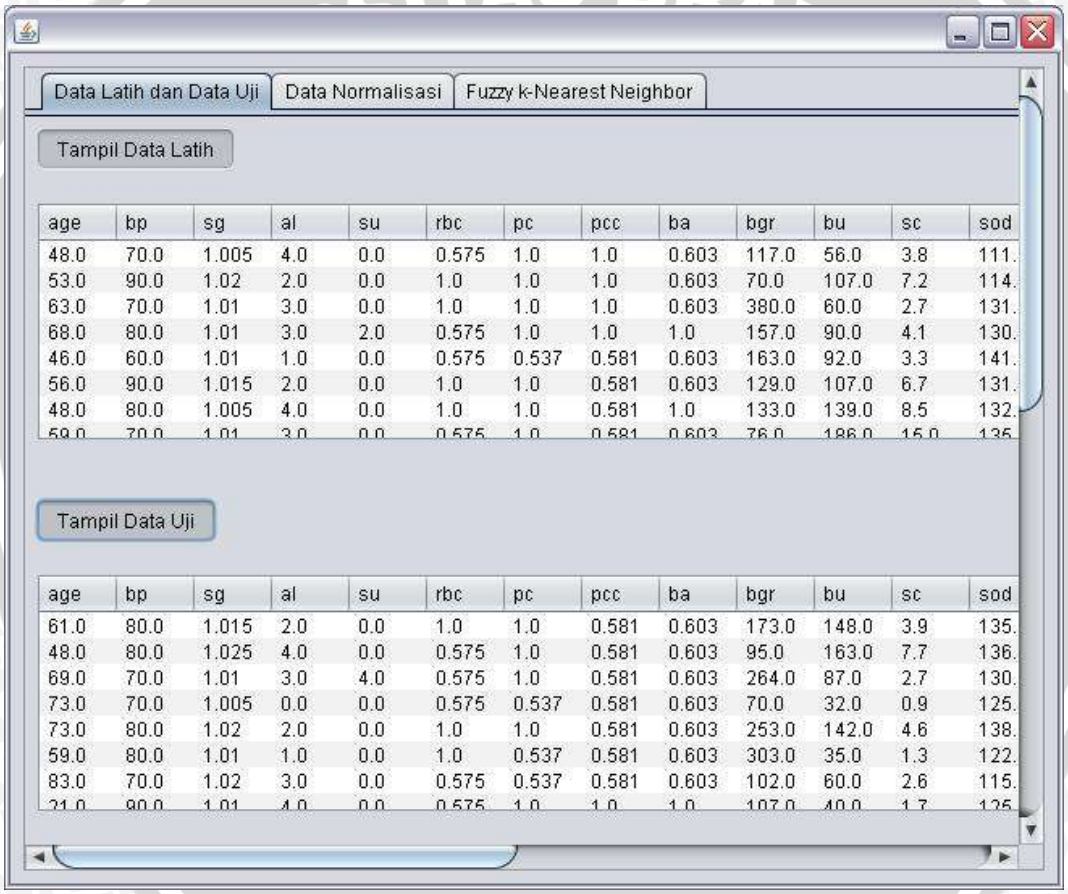
```
    private void btnTampilDataUjiActionPerformed(java.awt.event.ActionEvent
evt) {
        BrowseData browseData = new BrowseData();

        browseData.choose();
        String pathDataUji = browseData.getPath();
        ReadData readData = new ReadData(pathDataUji);
        lstAtributDataUji = readData.bacaData();

        namaKolomDataUji = new String [25];

        namaKolomDataUji[0] = "age";
        namaKolomDataUji[1] = "bp";
        namaKolomDataUji[2] = "sg";
        namaKolomDataUji[3] = "al";
        namaKolomDataUji[4] = "su";
        namaKolomDataUji[5] = "rbc";
        namaKolomDataUji[6] = "pc";
        namaKolomDataUji[7] = "pcc";
        namaKolomDataUji[8] = "ba";
        namaKolomDataUji[9] = "bgr";
        namaKolomDataUji[10] = "bu";
        namaKolomDataUji[11] = "sc";
        namaKolomDataUji[12] = "sod";
        namaKolomDataUji[13] = "pot";
        namaKolomDataUji[14] = "hemo";
        namaKolomDataUji[15] = "pcv";
        namaKolomDataUji[16] = "wbcc";
        namaKolomDataUji[17] = "rbcc";
        namaKolomDataUji[18] = "htn";
        namaKolomDataUji[19] = "dm";
```

```
namaKolomDataUji[20] = "cad";
namaKolomDataUji[21] = "appet";
namaKolomDataUji[22] = "pe";
namaKolomDataUji[23] = "ane";
namaKolomDataUji[24] = "kelas";

isiTabelDataUji = new Object[lstAtributDataUji.size()][25];

for (int i=0 ; i<lstAtributDataUji.size();i++){

  isiTabelDataUji [i][0]=lstAtributDataUji.get(i).getAge();
  isiTabelDataUji [i][1]=lstAtributDataUji.get(i).getBp();
  isiTabelDataUji [i][2]=lstAtributDataUji.get(i).getSg();
  isiTabelDataUji [i][3]=lstAtributDataUji.get(i).getAl();
  isiTabelDataUji [i][4]=lstAtributDataUji.get(i).getSu();
  isiTabelDataUji [i][5]=lstAtributDataUji.get(i).getRbc();
  isiTabelDataUji [i][6]=lstAtributDataUji.get(i).getPc();
  isiTabelDataUji [i][7]=lstAtributDataUji.get(i).getPcc();
  isiTabelDataUji [i][8]=lstAtributDataUji.get(i).getBa();
  isiTabelDataUji [i][9]=lstAtributDataUji.get(i).getBgr();
  isiTabelDataUji [i][10]=lstAtributDataUji.get(i).getBu();
  isiTabelDataUji [i][11]=lstAtributDataUji.get(i).getSc();
  isiTabelDataUji [i][12]=lstAtributDataUji.get(i).getSod();
  isiTabelDataUji [i][13]=lstAtributDataUji.get(i).getPot();
  isiTabelDataUji [i][14]=lstAtributDataUji.get(i).getHemo();
  isiTabelDataUji [i][15]=lstAtributDataUji.get(i).getPcv();
  isiTabelDataUji [i][16]=lstAtributDataUji.get(i).getWbcc();
  isiTabelDataUji [i][17]=lstAtributDataUji.get(i).getRbcc();
  isiTabelDataUji [i][18]=lstAtributDataUji.get(i).getHtn();
  isiTabelDataUji [i][19]=lstAtributDataUji.get(i).getDm();
  isiTabelDataUji [i][20]=lstAtributDataUji.get(i).getCad();
  isiTabelDataUji [i][21]=lstAtributDataUji.get(i).getAppet();
```

```
            isiTabelDataUji [i][22]=lstAtributDataUji.get(i).getPe();
            isiTabelDataUji [i][23]=lstAtributDataUji.get(i).getAne();
            isiTabelDataUji [i][24]=lstAtributDataUji.get(i).getKelas();
        }


        tblDataUji.setModel(new DefaultTableModel(isiTabelDataUji,
    namaKolomDataUji));
```

***Source code* 4.3 Menampilkan data uji**

Screenshot program saat menampilkan data latih dan data uji akan ditunjukkan pada gambar 4.1 di bawah ini.



Gambar 4.1 *Screenshot* program saat menampilkan data latih dan data uji

4.2.3 Proses normalisasi data

Untuk melakukan normalisasi terhadap data latih dan data uji pada data yang digunakan ditunjukkan pada *source code* 4.4 berikut.

Proses normalisasi data
package ckd_fknn;

```
import java.util.ArrayList;
import java.util.List;

public class Normalisasi {
   private List<AtributData> lstAtributDataLatih;
   private List<AtributData> lstAtributDataUji;
   private List<AtributData> lstAtributDataLatihNormal;
   private List<AtributData> lstAtributDataUjiNormal;
   private List<AtributData> mixLatihUji;

   private double minAge;
   private double minBp;
   private double minSg;
   private double minAl;
   private double minSu;
   private double minRbc;
   private double minPc;
   private double minPcc;
   private double minBa;
   private double minBgr;
   private double minBu;
   private double minSc;
   private double minSod;
   private double minPot;
   private double minHemo;
   private double minPcv;
   private double minWbcc;
   private double minRbcc;
   private double minHtn;
   private double minDm;
   private double minCad;
   private double minAppet;
   private double minPe;
   private double minAne;
   private double maxAge;
   private double maxBp;
   private double maxSg;
   private double maxAl;
   private double maxSu;
   private double maxRbc;
   private double maxPc;
   private double maxPcc;
   private double maxBa;
   private double maxBgr;
   private double maxBu;
```

```
private double maxSc;
private double maxSod;
private double maxPot;
private double maxHemo;
private double maxPcv;
private double maxWbcc;
private double maxRbcc;
private double maxHtn;
private double maxDm;
private double maxCad;
private double maxAppet;
private double maxPe;
private double maxAne;
private double maxAgeUji;
private double maxBpUji;
private double maxSgUji;
private double maxAlUji;
private double maxSuUji;
private double maxRbcUji;
private double maxPcUji;
private double maxPccUji;
private double maxBaUji;
private double maxBgrUji;
private double maxBuUji;
private double maxScUji;
private double maxSodUji;
private double maxPotUji;
private double maxHemoUji;
private double maxPcvUji;
private double maxWbccUji;
private double maxRbccUji;
private double maxHtnUji;
private double maxDmUji;
private double maxCadUji;
private double maxAppetUji;
private double maxPeUji;
private double maxAneUji;
private double minAgeUji;
private double minBpUji;
private double minSgUji;
private double minAlUji;
private double minSuUji;
private double minRbcUji;
private double minPcUji;
private double minPccUji;
```

```
    private double minBaUji;
    private double minBgrUji;
    private double minBuUji;
    private double minScUji;
    private double minSodUji;
    private double minPotUji;
    private double minHemoUji;
    private double minPcvUji;
    private double minWbccUji;
    private double minRbccUji;
    private double minHtnUji;
    private double minDmUji;
    private double minCadUji;
    private double minAppetUji;
    private double minPeUji;
    private double minAneUji;

    public Normalisasi (List<AtributData> atributDataLatih, List<AtributData>
atributDataUji){
        this.lstAtributDataLatih = atributDataLatih;
        this.lstAtributDataUji = atributDataUji;
        mixLatihUji = new ArrayList<AtributData>();

        for(AtributData object: lstAtributDataLatih)
        {
           mixLatihUji.add(object);
        }
        for(AtributData objek : lstAtributDataUji)
        {
           mixLatihUji.add(objek);
        }
    }
    public void minMax (){
            maxAge = 0;
            maxBp = 0;
            maxSg = 0;
            maxAl = 0;
            maxSu = 0;
            maxRbc = 0;
            maxPc = 0;
            maxPcc = 0;
            maxBa = 0;
            maxBgr = 0;
            maxBu = 0;
            maxSc = 0;
```

```
maxSod = 0;
maxPot = 0;
maxHemo = 0;
maxPcv = 0;
maxWbcc = 0;
maxRbcc = 0;
maxHtn = 0;
maxDm = 0;
maxCad = 0;
maxAppet = 0;
maxPe = 0;
maxAne = 0;
maxAgeUji = 0;
maxBpUji = 0;
maxSgUji = 0;
maxAlUji = 0;
maxSuUji = 0;
maxRbcUji = 0;
maxPcUji = 0;
maxPccUji = 0;
maxBaUji = 0;
maxBgrUji = 0;
maxBuUji = 0;
maxScUji = 0;
maxSodUji = 0;
maxPotUji = 0;
maxHemoUji = 0;
maxPcvUji = 0;
maxWbccUji = 0;
maxRbccUji = 0;
maxHtnUji = 0;
maxDmUji = 0;
maxCadUji = 0;
maxAppetUji = 0;
maxPeUji = 0;
maxAneUji = 0;
minAge = lstAtributDataLatih.get(0).getAge();
minBp = lstAtributDataLatih.get(0).getBp();
minSg = lstAtributDataLatih.get(0).getSg();
minAl = lstAtributDataLatih.get(0).getAl();
minSu = lstAtributDataLatih.get(0).getSu();
minRbc = lstAtributDataLatih.get(0).getRbc();
minPc = lstAtributDataLatih.get(0).getPc();
minPcc = lstAtributDataLatih.get(0).getPcc();
minBa = lstAtributDataLatih.get(0).getBa();
```

```
        minBgr = lstAtributDataLatih.get(0).getBgr();
        minBu = lstAtributDataLatih.get(0).getBu();
        minSc = lstAtributDataLatih.get(0).getSc();
        minSod = lstAtributDataLatih.get(0).getSod();
        minPot = lstAtributDataLatih.get(0).getPot();
        minHemo = lstAtributDataLatih.get(0).getHemo();
        minPcv = lstAtributDataLatih.get(0).getPcv();
        minWbcc = lstAtributDataLatih.get(0).getWbcc();
        minRbcc = lstAtributDataLatih.get(0).getRbcc();
        minHtn = lstAtributDataLatih.get(0).getHtn();
        minDm = lstAtributDataLatih.get(0).getDm();
        minCad = lstAtributDataLatih.get(0).getCad();
        minAppet = lstAtributDataLatih.get(0).getAppet();
        minPe = lstAtributDataLatih.get(0).getPe();
        minAne = lstAtributDataLatih.get(0).getAne();
        minAgeUji = lstAtributDataLatih.get(0).getAge();
        minBpUji = lstAtributDataLatih.get(0).getBp();
        minSgUji = lstAtributDataLatih.get(0).getSg();
        minAlUji = lstAtributDataLatih.get(0).getAl();
        minSuUji = lstAtributDataLatih.get(0).getSu();
        minRbcUji = lstAtributDataLatih.get(0).getRbc();
        minPcUji = lstAtributDataLatih.get(0).getPc();
        minPccUji = lstAtributDataLatih.get(0).getPcc();
        minBaUji = lstAtributDataLatih.get(0).getBa();
        minBgrUji = lstAtributDataLatih.get(0).getBgr();
        minBuUji = lstAtributDataLatih.get(0).getBu();
        minScUji = lstAtributDataLatih.get(0).getSc();
        minSodUji = lstAtributDataLatih.get(0).getSod();
        minPotUji = lstAtributDataLatih.get(0).getPot();
        minHemoUji = lstAtributDataLatih.get(0).getHemo();
        minPcvUji = lstAtributDataLatih.get(0).getPcv();
        minWbccUji = lstAtributDataLatih.get(0).getWbcc();
        minRbccUji = lstAtributDataLatih.get(0).getRbcc();
        minHtnUji = lstAtributDataLatih.get(0).getHtn();
        minDmUji = lstAtributDataLatih.get(0).getDm();
        minCadUji = lstAtributDataLatih.get(0).getCad();
        minAppetUji = lstAtributDataLatih.get(0).getAppet();
        minPeUji = lstAtributDataLatih.get(0).getPe();
        minAneUji = lstAtributDataLatih.get(0).getAne();
for(int i=0; i<lstAtributDataLatih.size(); i++)
 {
    if(lstAtributDataLatih.get(i).getAge() > maxAge)
    {
       maxAge= lstAtributDataLatih.get(i).getAge();
    }
```

```
if(lstAtributDataLatih.get(i).getBp() > maxBp)
{
   maxBp= lstAtributDataLatih.get(i).getBp();
}
if(lstAtributDataLatih.get(i).getSg() > maxSg)
{
   maxSg= lstAtributDataLatih.get(i).getSg();
}
if(lstAtributDataLatih.get(i).getAl() > maxAl)
{
   maxAl= lstAtributDataLatih.get(i).getAl();
}
if(lstAtributDataLatih.get(i).getSu() > maxSu)
{
   maxSu= lstAtributDataLatih.get(i).getSu();
}
if(lstAtributDataLatih.get(i).getRbc() > maxRbc)
{
   maxRbc= lstAtributDataLatih.get(i).getRbc();
}
if(lstAtributDataLatih.get(i).getPc() > maxPc)
{
   maxPc= lstAtributDataLatih.get(i).getPc();
}
if(lstAtributDataLatih.get(i).getPcc() > maxPcc)
{
   maxPcc= lstAtributDataLatih.get(i).getPcc();
}
if(lstAtributDataLatih.get(i).getBa() > maxBa)
{
   maxBa= lstAtributDataLatih.get(i).getBa();
}
if(lstAtributDataLatih.get(i).getBgr() > maxBgr)
{
   maxBgr= lstAtributDataLatih.get(i).getBgr();
}
if(lstAtributDataLatih.get(i).getBu() > maxBu)
{
   maxBu= lstAtributDataLatih.get(i).getBu();
}
if(lstAtributDataLatih.get(i).getSc() > maxSc)
{
   maxSc= lstAtributDataLatih.get(i).getSc();
}
if(lstAtributDataLatih.get(i).getSod() > maxSod)
```

```
{
   maxSod= lstAtributDataLatih.get(i).getSod();
}
if(lstAtributDataLatih.get(i).getPot() > maxPot)
{
   maxPot= lstAtributDataLatih.get(i).getPot();
}
if(lstAtributDataLatih.get(i).getHemo() > maxHemo)
{
   maxHemo= lstAtributDataLatih.get(i).getHemo();
}
if(lstAtributDataLatih.get(i).getPcv() > maxPcv)
{
   maxPcv= lstAtributDataLatih.get(i).getPcv();
}
if(lstAtributDataLatih.get(i).getWbcc() > maxWbcc)
{
   maxWbcc= lstAtributDataLatih.get(i).getWbcc();
}
if(lstAtributDataLatih.get(i).getRbcc() > maxRbcc)
{
   maxRbcc= lstAtributDataLatih.get(i).getRbcc();
}
if(lstAtributDataLatih.get(i).getHtn() > maxHtn)
{
   maxHtn= lstAtributDataLatih.get(i).getHtn();
}
if(lstAtributDataLatih.get(i).getDm() > maxDm)
{
   maxDm= lstAtributDataLatih.get(i).getDm();
}
if(lstAtributDataLatih.get(i).getCad() > maxCad)
{
   maxCad= lstAtributDataLatih.get(i).getCad();
}
if(lstAtributDataLatih.get(i).getAppet() > maxAppet)
{
   maxAppet= lstAtributDataLatih.get(i).getAppet();
}
if(lstAtributDataLatih.get(i).getPe() > maxPe)
{
   maxPe= lstAtributDataLatih.get(i).getPe();
}
if(lstAtributDataLatih.get(i).getAne() > maxAne)
{
```

```
            maxAne= lstAtributDataLatih.get(i).getAne();
        }

        if(minAge > lstAtributDataLatih.get(i).getAge())
        {
            minAge = lstAtributDataLatih.get(i).getAge();
        }
        if(minBp > lstAtributDataLatih.get(i).getBp())
        {
            minBp = lstAtributDataLatih.get(i).getBp();
        }
        if(minSg > lstAtributDataLatih.get(i).getSg())
        {
            minSg = lstAtributDataLatih.get(i).getSg();
        }
        if(minAl > lstAtributDataLatih.get(i).getAl())
        {
            minAl = lstAtributDataLatih.get(i).getAl();
        }
        if(minSu > lstAtributDataLatih.get(i).getSu())
        {
            minSu = lstAtributDataLatih.get(i).getSu();
        }
        if(minRbc > lstAtributDataLatih.get(i).getRbc())
        {
            minRbc = lstAtributDataLatih.get(i).getRbc();
        }
        if(minPc > lstAtributDataLatih.get(i).getPc())
        {
            minPc = lstAtributDataLatih.get(i).getPc();
        }
        if(minPcc > lstAtributDataLatih.get(i).getPcc())
        {
            minPcc = lstAtributDataLatih.get(i).getPcc();
        }
        if(minBa > lstAtributDataLatih.get(i).getBa())
        {
            minBa = lstAtributDataLatih.get(i).getBa();
        }
        if(minBgr > lstAtributDataLatih.get(i).getBgr())
        {
            minBgr = lstAtributDataLatih.get(i).getBgr();
        }
        if(minBu > lstAtributDataLatih.get(i).getBu())
        {
```

```
        minBu = lstAtributDataLatih.get(i).getBu();
    }
    if(minSc > lstAtributDataLatih.get(i).getSc())
    {
        minSc = lstAtributDataLatih.get(i).getSc();
    }
    if(minSod > lstAtributDataLatih.get(i).getSod())
    {
        minSod = lstAtributDataLatih.get(i).getSod();
    }
    if(minPot > lstAtributDataLatih.get(i).getPot())
    {
        minPot = lstAtributDataLatih.get(i).getPot();
    }
    if(minHemo > lstAtributDataLatih.get(i).getHemo())
    {
        minHemo = lstAtributDataLatih.get(i).getHemo();
    }
    if(minPcv > lstAtributDataLatih.get(i).getPcv())
    {
        minPcv = lstAtributDataLatih.get(i).getPcv();
    }
    if(minWbcc > lstAtributDataLatih.get(i).getWbcc())
    {
        minWbcc = lstAtributDataLatih.get(i).getWbcc();
    }
    if(minRbcc > lstAtributDataLatih.get(i).getRbcc())
    {
        minRbcc = lstAtributDataLatih.get(i).getRbcc();
    }
    if(minHtn > lstAtributDataLatih.get(i).getHtn())
    {
        minHtn = lstAtributDataLatih.get(i).getHtn();
    }
    if(minDm > lstAtributDataLatih.get(i).getDm())
    {
        minDm = lstAtributDataLatih.get(i).getDm();
    }
    if(minCad > lstAtributDataLatih.get(i).getCad())
    {
        minCad = lstAtributDataLatih.get(i).getCad();
    }
    if(minAppet > lstAtributDataLatih.get(i).getAppet())
    {
        minAppet = lstAtributDataLatih.get(i).getAppet();
```

```
        }
        if(minPe > lstAtributDataLatih.get(i).getPe())
        {
            minPe = lstAtributDataLatih.get(i).getPe();
        }
        if(minAne > lstAtributDataLatih.get(i).getAne())
        {
            minAne = lstAtributDataLatih.get(i).getAne();
        }
    }
    for(int i=0; i<lstAtributDataUji.size(); i++)
    {
        if(lstAtributDataUji.get(i).getAge() > maxAgeUji)
        {
            maxAgeUji= lstAtributDataUji.get(i).getAge();
        }
        if(lstAtributDataUji.get(i).getBp() > maxBpUji)
        {
            maxBpUji= lstAtributDataUji.get(i).getBp();
        }
        if(lstAtributDataUji.get(i).getSg() > maxSgUji)
        {
            maxSgUji= lstAtributDataUji.get(i).getSg();
        }
        if(lstAtributDataUji.get(i).getAl() > maxAlUji)
        {
            maxAlUji= lstAtributDataUji.get(i).getAl();
        }
        if(lstAtributDataUji.get(i).getSu() > maxSuUji)
        {
            maxSuUji= lstAtributDataUji.get(i).getSu();
        }
        if(lstAtributDataUji.get(i).getRbc() > maxRbcUji)
        {
            maxRbcUji= lstAtributDataUji.get(i).getRbc();
        }
        if(lstAtributDataUji.get(i).getPc() > maxPcUji)
        {
            maxPcUji= lstAtributDataUji.get(i).getPc();
        }
        if(lstAtributDataUji.get(i).getPcc() > maxPccUji)
        {
            maxPccUji= lstAtributDataUji.get(i).getPcc();
        }
        if(lstAtributDataUji.get(i).getBa() > maxBaUji)
```

```
{
   maxBaUji= lstAtributDataUji.get(i).getBa();
}
if(lstAtributDataUji.get(i).getBgr() > maxBgrUji)
{
   maxBgrUji= lstAtributDataUji.get(i).getBgr();
}
if(lstAtributDataUji.get(i).getBu() > maxBuUji)
{
   maxBuUji= lstAtributDataUji.get(i).getBu();
}
if(lstAtributDataUji.get(i).getSc() > maxScUji)
{
   maxScUji= lstAtributDataUji.get(i).getSc();
}
if(lstAtributDataUji.get(i).getSod() > maxSodUji)
{
   maxSodUji= lstAtributDataUji.get(i).getSod();
}
if(lstAtributDataUji.get(i).getPot() > maxPotUji)
{
   maxPotUji= lstAtributDataUji.get(i).getPot();
}
if(lstAtributDataUji.get(i).getHemo() > maxHemoUji)
{
   maxHemoUji= lstAtributDataUji.get(i).getHemo();
}
if(lstAtributDataUji.get(i).getPcv() > maxPcvUji)
{
   maxPcvUji= lstAtributDataUji.get(i).getPcv();
}
if(lstAtributDataUji.get(i).getWbcc() > maxWbccUji)
{
   maxWbccUji= lstAtributDataUji.get(i).getWbcc();
}
if(lstAtributDataUji.get(i).getRbcc() > maxRbccUji)
{
   maxRbccUji= lstAtributDataUji.get(i).getRbcc();
}
if(lstAtributDataUji.get(i).getHtn() > maxHtnUji)
{
   maxHtnUji= lstAtributDataUji.get(i).getHtn();
}
if(lstAtributDataUji.get(i).getDm() > maxDmUji)
{
```

```
        maxDmUji= lstAtributDataUji.get(i).getDm();
    }
    if(lstAtributDataUji.get(i).getCad() > maxCadUji)
    {
        maxCadUji= lstAtributDataUji.get(i).getCad();
    }
    if(lstAtributDataUji.get(i).getAppet() > maxAppetUji)
    {
        maxAppetUji= lstAtributDataUji.get(i).getAppet();
    }
    if(lstAtributDataUji.get(i).getPe() > maxPeUji)
    {
        maxPeUji= lstAtributDataUji.get(i).getPe();
    }
    if(lstAtributDataUji.get(i).getAne() > maxAneUji)
    {
        maxAneUji= lstAtributDataUji.get(i).getAne();
    }

    if(minAgeUji > lstAtributDataUji.get(i).getAge())
    {
        minAgeUji = lstAtributDataUji.get(i).getAge();
    }
    if(minBpUji > lstAtributDataUji.get(i).getBp())
    {
        minBpUji = lstAtributDataUji.get(i).getBp();
    }
    if(minSgUji > lstAtributDataUji.get(i).getSg())
    {
        minSgUji = lstAtributDataUji.get(i).getSg();
    }
    if(minAlUji > lstAtributDataUji.get(i).getAl())
    {
        minAlUji = lstAtributDataUji.get(i).getAl();
    }
    if(minSuUji > lstAtributDataUji.get(i).getSu())
    {
        minSuUji = lstAtributDataUji.get(i).getSu();
    }
    if(minRbcUji > lstAtributDataUji.get(i).getRbc())
    {
        minRbcUji = lstAtributDataUji.get(i).getRbc();
    }
    if(minPcUji > lstAtributDataUji.get(i).getPc())
    {
```

```
        minPcUji = lstAtributDataUji.get(i).getPc();
    }
    if(minPccUji > lstAtributDataUji.get(i).getPcc())
    {
        minPccUji = lstAtributDataUji.get(i).getPcc();
    }
    if(minBaUji > lstAtributDataUji.get(i).getBa())
    {
        minBaUji = lstAtributDataUji.get(i).getBa();
    }
    if(minBgrUji > lstAtributDataUji.get(i).getBgr())
    {
        minBgrUji = lstAtributDataUji.get(i).getBgr();
    }
    if(minBuUji > lstAtributDataUji.get(i).getBu())
    {
        minBuUji = lstAtributDataUji.get(i).getBu();
    }
    if(minScUji > lstAtributDataUji.get(i).getSc())
    {
        minScUji = lstAtributDataUji.get(i).getSc();
    }
    if(minSodUji > lstAtributDataUji.get(i).getSod())
    {
        minSodUji = lstAtributDataUji.get(i).getSod();
    }
    if(minPotUji > lstAtributDataUji.get(i).getPot())
    {
        minPotUji = lstAtributDataUji.get(i).getPot();
    }
    if(minHemoUji > lstAtributDataUji.get(i).getHemo())
    {
        minHemoUji = lstAtributDataUji.get(i).getHemo();
    }
    if(minPcvUji > lstAtributDataUji.get(i).getPcv())
    {
        minPcvUji = lstAtributDataUji.get(i).getPcv();
    }
    if(minWbccUji > lstAtributDataUji.get(i).getWbcc())
    {
        minWbccUji = lstAtributDataUji.get(i).getWbcc();
    }
    if(minRbccUji > lstAtributDataUji.get(i).getRbcc())
    {
        minRbccUji = lstAtributDataUji.get(i).getRbcc();
```

```
            }
            if(minHtnUji > lstAtributDataUji.get(i).getHtn())
            {
               minHtnUji = lstAtributDataUji.get(i).getHtn();
            }
            if(minDmUji > lstAtributDataUji.get(i).getDm())
            {
               minDmUji = lstAtributDataUji.get(i).getDm();
            }
            if(minCadUji > lstAtributDataUji.get(i).getCad())
            {
               minCadUji = lstAtributDataUji.get(i).getCad();
            }
            if(minAppetUji > lstAtributDataUji.get(i).getAppet())
            {
               minAppetUji = lstAtributDataUji.get(i).getAppet();
            }
            if(minPeUji > lstAtributDataUji.get(i).getPe())
            {
               minPeUji = lstAtributDataUji.get(i).getPe();
            }
            if(minAneUji > lstAtributDataUji.get(i).getAne())
            {
               minAneUji = lstAtributDataUji.get(i).getAne();
            }
         }
      }
      public void proses(){
         lstAtributDataLatihNormal = new ArrayList<AtributData>();
         lstAtributDataUjiNormal = new ArrayList<AtributData>();
         for(int i=0 ; i<lstAtributDataLatih.size() ; i++)
         {
            AtributData atributData = new AtributData();
            atributData.setAge(((lstAtributDataLatih.get(i).getAge()) - minAge) /
(maxAge - minAge));
            atributData.setBp(((lstAtributDataLatih.get(i).getBp()) - minBp) /
(maxBp - minBp));
            atributData.setSg(((lstAtributDataLatih.get(i).getSg()) - minSg) / (maxSg
- minSg));
            atributData.setAl(((lstAtributDataLatih.get(i).getAl()) - minAl) / (maxAl
- minAl));
            atributData.setSu(((lstAtributDataLatih.get(i).getSu()) - minSu) /
(maxSu - minSu));
            atributData.setRbc(((lstAtributDataLatih.get(i).getRbc()) - minRbc) /
(maxRbc - minRbc));
```

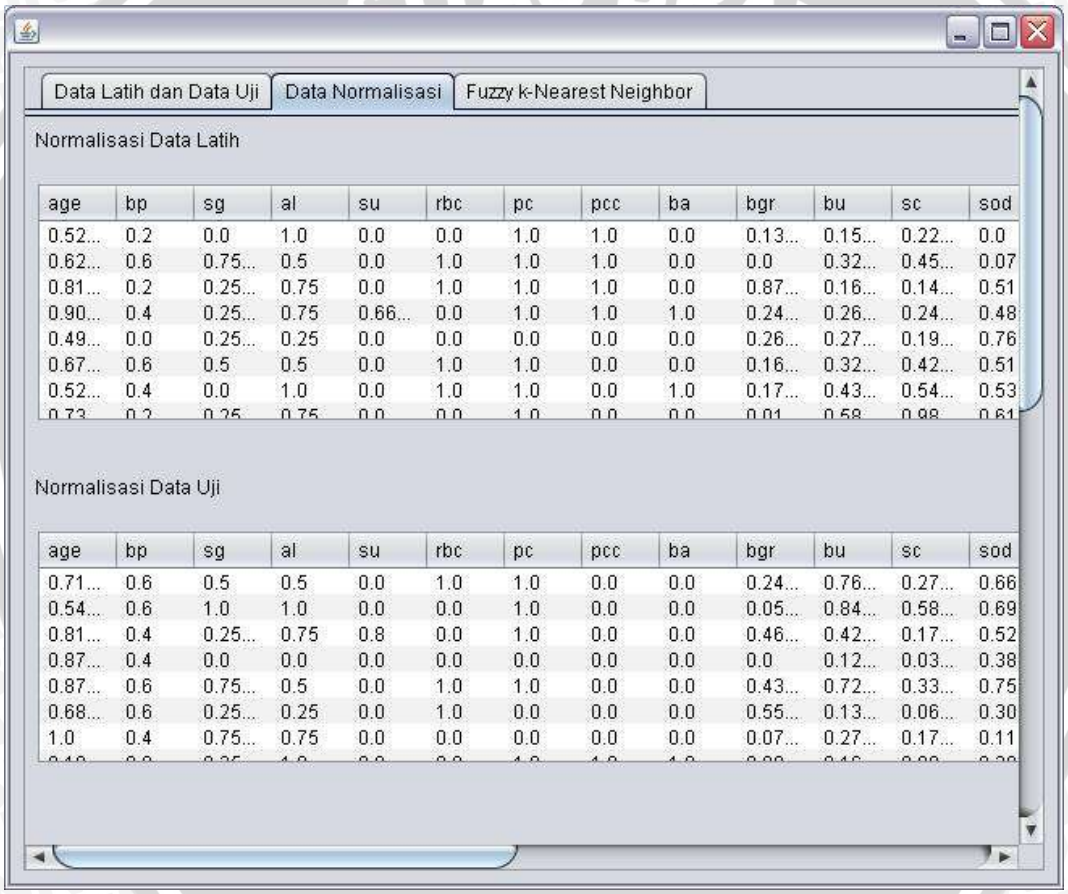
```
            atributData.setPc(((lstAtributDataLatih.get(i).getPc()) - minPc) /
(maxPc - minPc));
            atributData.setPcc(((lstAtributDataLatih.get(i).getPcc()) - minPcc) /
(maxPcc - minPcc));
            atributData.setBa(((lstAtributDataLatih.get(i).getBa()) - minBa) /
(maxBa - minBa));
            atributData.setBgr(((lstAtributDataLatih.get(i).getBgr()) - minBgr) /
(maxBgr - minBgr));
            atributData.setBu(((lstAtributDataLatih.get(i).getBu()) - minBu) /
(maxBu - minBu));
            atributData.setSc(((lstAtributDataLatih.get(i).getSc()) - minSc) / (maxSc
- minSc));
            atributData.setSod(((lstAtributDataLatih.get(i).getSod()) - minSod) /
(maxSod - minSod));
            atributData.setPot(((lstAtributDataLatih.get(i).getPot()) - minPot) /
(maxPot - minPot));
            atributData.setHemo(((lstAtributDataLatih.get(i).getHemo()) -
minHemo) / (maxHemo - minHemo));
            atributData.setPcv(((lstAtributDataLatih.get(i).getPcv()) - minPcv) /
(maxPcv - minPcv));
            atributData.setWbcc(((lstAtributDataLatih.get(i).getWbcc()) -
minWbcc) / (maxWbcc - minWbcc));
            atributData.setRbcc(((lstAtributDataLatih.get(i).getRbcc()) - minRbcc) /
(maxRbcc - minRbcc));
            atributData.setHtn(((lstAtributDataLatih.get(i).getHtn()) - minHtn) /
(maxHtn - minHtn));
            atributData.setDm(((lstAtributDataLatih.get(i).getDm()) - minDm) /
(maxDm - minDm));
            atributData.setCad(((lstAtributDataLatih.get(i).getCad()) - minCad) /
(maxCad - minCad));
            atributData.setAppet(((lstAtributDataLatih.get(i).getAppet()) -
minAppet) / (maxAppet - minAppet));
            atributData.setPe(((lstAtributDataLatih.get(i).getPe()) - minPe) /
(maxPe - minPe));
            atributData.setAne(((lstAtributDataLatih.get(i).getAne()) - minAne) /
(maxAne - minAne));
            lstAtributDataLatihNormal.add(atributData);
        }
        for(int i=0 ; i<lstAtributDataUji.size() ; i++)
        {
            AtributData atributData = new AtributData();
            atributData.setAge(((lstAtributDataUji.get(i).getAge()) - minAgeUji) /
(maxAgeUji - minAgeUji));
            atributData.setBp(((lstAtributDataUji.get(i).getBp()) - minBpUji) /
(maxBpUji - minBpUji));
```

```
            atributData.setSg(((lstAtributDataUji.get(i).getSg()) - minSgUji) /
(maxSgUji - minSgUji));
            atributData.setAl(((lstAtributDataUji.get(i).getAl()) - minAlUji) /
(maxAlUji - minAlUji));
            atributData.setSu(((lstAtributDataUji.get(i).getSu()) - minSuUji) /
(maxSuUji - minSuUji));
            atributData.setRbc(((lstAtributDataUji.get(i).getRbc()) - minRbcUji) /
(maxRbcUji - minRbcUji));
            atributData.setPc(((lstAtributDataUji.get(i).getPc()) - minPcUji) /
(maxPcUji - minPcUji));
            atributData.setPcc(((lstAtributDataUji.get(i).getPcc()) - minPccUji) /
(maxPccUji - minPccUji));
            atributData.setBa(((lstAtributDataUji.get(i).getBa()) - minBaUji) /
(maxBaUji - minBaUji));
            atributData.setBgr(((lstAtributDataUji.get(i).getBgr()) - minBgrUji) /
(maxBgrUji - minBgrUji));
            atributData.setBu(((lstAtributDataUji.get(i).getBu()) - minBuUji) /
(maxBuUji - minBuUji));
            atributData.setSc(((lstAtributDataUji.get(i).getSc()) - minScUji) /
(maxScUji - minScUji));
            atributData.setSod(((lstAtributDataUji.get(i).getSod()) - minSodUji) /
(maxSodUji - minSodUji));
            atributData.setPot(((lstAtributDataUji.get(i).getPot()) - minPotUji) /
(maxPotUji - minPotUji));
            atributData.setHemo(((lstAtributDataUji.get(i).getHemo()) -
minHemoUji) / (maxHemoUji - minHemoUji));
            atributData.setPcv(((lstAtributDataUji.get(i).getPcv()) - minPcvUji) /
(maxPcvUji - minPcvUji));
            atributData.setWbcc(((lstAtributDataUji.get(i).getWbcc()) -
minWbccUji) / (maxWbccUji - minWbccUji));
            atributData.setRbcc(((lstAtributDataUji.get(i).getRbcc()) - minRbccUji)
/ (maxRbccUji - minRbccUji));
            atributData.setHtn(((lstAtributDataUji.get(i).getHtn()) - minHtnUji) /
(maxHtnUji - minHtnUji));
            atributData.setDm(((lstAtributDataUji.get(i).getDm()) - minDmUji) /
(maxDmUji - minDmUji));
            atributData.setCad(((lstAtributDataUji.get(i).getCad()) - minCadUji) /
(maxCadUji - minCadUji));
            atributData.setAppet(((lstAtributDataUji.get(i).getAppet()) -
minAppetUji) / (maxAppetUji - minAppetUji));
            atributData.setPe(((lstAtributDataUji.get(i).getPe()) - minPeUji) /
(maxPeUji - minPeUji));
            atributData.setAne(((lstAtributDataUji.get(i).getAne()) - minAneUji) /
(maxAneUji - minAneUji));
            lstAtributDataUjiNormal.add(atributData);
```

```
        }
    }
    public List<AtributData> getLstAtributDataLatihNormal() {
        return lstAtributDataLatihNormal;
    }
    public List<AtributData> getLstAtributDataUjiNormal() {
        return lstAtributDataUjiNormal;
    }
}
```

***Source code* 4.4 Normalisasi data**

Screenshot program saat menampilkan hasil dari normalisasi data latih dan data uji ditunjukkan pada gambar 4.2 berikut ini.



Gambar 4.2 *Screenshot* program saat menampilkan hasil normalisasi

4.2.4 Proses *Fuzzy k-Nearest Neighbor*

Proses perhitungan data menggunakan metode *Fuzzy k-Nearest Neighbor* ditunjukkan pada *source code* 4.5.

Proses *Fuzzy k-Nearest Neighbor*
package ckd_fknn;

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class Knn {
   private List<AtributData> lstAtributDataLatih;
   private List<AtributData> lstAtributDataLatihNormal;
   private List<AtributData> lstAtributDataUjiNormal;
   private List<EuclideanData> lstEuclideanDatas;
   private List<Integer> hasilPrediksi;
   int K;
   double n1;
   double n2;
   double u11;
   double u12;
   double u21;
   double u22;
   double [] pembilang;
   double [] penyebut;
   double [] hasil;

   public Knn (List<AtributData> lstAtributDataLatihNormal, List<AtributData>
lstAtributDataLatihUjiNormal){
      this.lstAtributDataLatihNormal = lstAtributDataLatihNormal;
      this.lstAtributDataUjiNormal = lstAtributDataLatihUjiNormal;
   }
   public void process (){
      //cari euclidean
      hasilPrediksi = new ArrayList<Integer>();
      for (int j = 0; j<lstAtributDataUjiNormal.size(); j++){
         lstEuclideanDatas = new ArrayList<EuclideanData>();
        for (int i = 0; i<lstAtributDataLatihNormal.size(); i++){
           EuclideanData euclideanData = new EuclideanData();

euclideanData.setNilaiTarget((Math.pow((lstAtributDataUjiNormal.get(j).get
Age()-lstAtributDataLatihNormal.get(i).getAge()), 2))
           + (Math.pow((lstAtributDataUjiNormal.get(j).getBp() -
lstAtributDataLatihNormal.get(i).getBp()), 2))
           + (Math.pow((lstAtributDataUjiNormal.get(j).getSg() -
lstAtributDataLatihNormal.get(i).getSg()), 2))
           + (Math.pow((lstAtributDataUjiNormal.get(j).getAl() -
lstAtributDataLatihNormal.get(i).getAl()), 2))
           + (Math.pow((lstAtributDataUjiNormal.get(j).getSu() -
lstAtributDataLatihNormal.get(i).getSu()), 2))
```

```
            + (Math.pow((lstAtributDataUjiNormal.get(j).getRbc() -
lstAtributDataLatihNormal.get(i).getRbc()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getPc() -
lstAtributDataLatihNormal.get(i).getPc()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getPcc() -
lstAtributDataLatihNormal.get(i).getPcc()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getBa() -
lstAtributDataLatihNormal.get(i).getBa()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getBgr() -
lstAtributDataLatihNormal.get(i).getBgr()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getBu() -
lstAtributDataLatihNormal.get(i).getBu()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getSc() -
lstAtributDataLatihNormal.get(i).getSc()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getSod() -
lstAtributDataLatihNormal.get(i).getSod()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getPot() -
lstAtributDataLatihNormal.get(i).getPot()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getHemo() -
lstAtributDataLatihNormal.get(i).getHemo()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getPcv() -
lstAtributDataLatihNormal.get(i).getPcv()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getWbcc() -
lstAtributDataLatihNormal.get(i).getWbcc()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getRbcc() -
lstAtributDataLatihNormal.get(i).getRbcc()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getHtn() -
lstAtributDataLatihNormal.get(i).getHtn()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getDm() -
lstAtributDataLatihNormal.get(i).getDm()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getCad() -
lstAtributDataLatihNormal.get(i).getCad()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getAppet() -
lstAtributDataLatihNormal.get(i).getAppet()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getPe() -
lstAtributDataLatihNormal.get(i).getPe()), 2))
            + (Math.pow((lstAtributDataUjiNormal.get(j).getAne() -
lstAtributDataLatihNormal.get(i).getAne()), 2)));

euclideanData.setNilaiTarget(Math.sqrt(euclideanData.getNilaiTarget()));
            euclideanData.setKelas(lstAtributDataLatih.get(i).getKelas());

            lstEuclideanDatas.add(euclideanData);

        }
        for(EuclideanData a : lstEuclideanDatas)
```

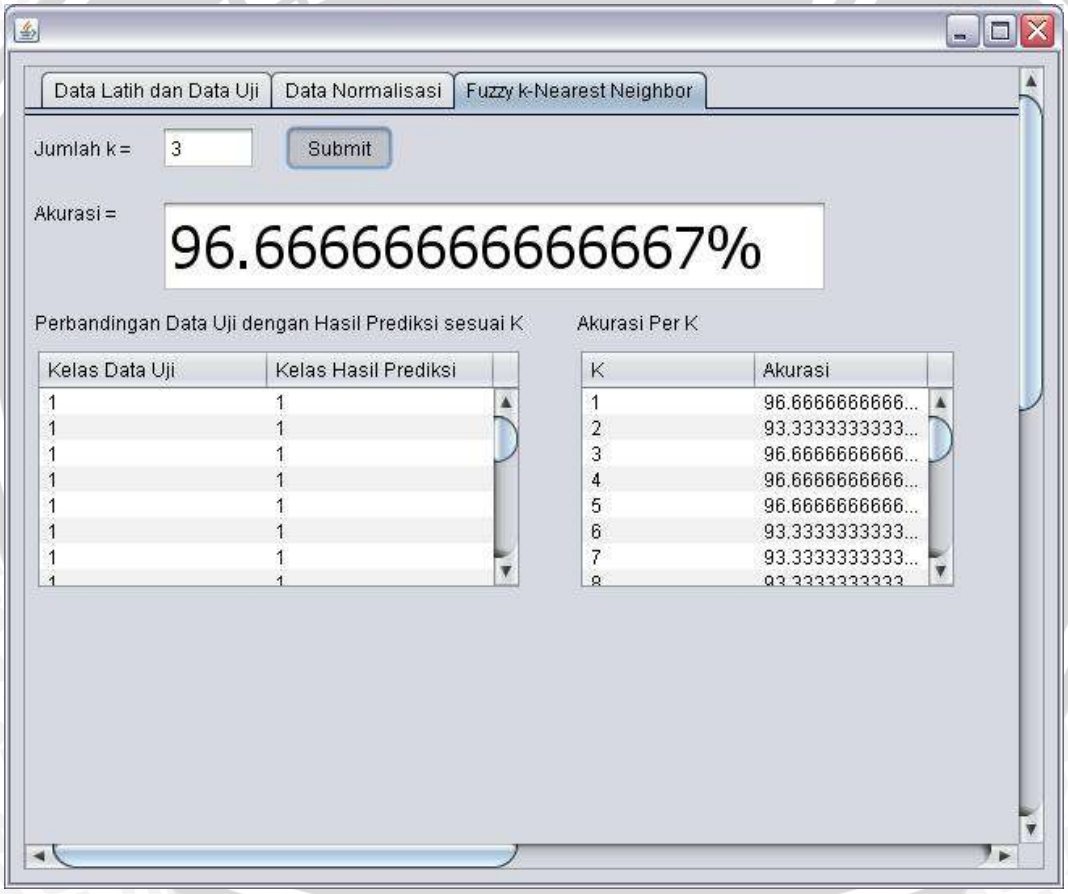
```
{
}
Collections.sort(lstEuclideanDatas, new TargetComparator());
for(EuclideanData a : lstEuclideanDatas)
{
}
//membership
n1 = 0;
n2 = 0;

for(AtributData object : lstAtributDataLatih)
{
  if(object.getKelas() == 1)
  {
     n1++;
  }
  else
  {
     n2++;
  }
}
//mencari u11-u22
u11= (0.51+((n1/lstEuclideanDatas.size())*0.49));
u12= ((n1/lstEuclideanDatas.size())*0.49);
u22= (0.51+((n2/lstEuclideanDatas.size())*0.49));
u21= ((n2/lstEuclideanDatas.size())*0.49);

for(int k=0; k<lstEuclideanDatas.size(); k++)
{
  if(lstEuclideanDatas.get(k).getKelas() == 1)
  {
     lstEuclideanDatas.get(k).setN1(u11);
     lstEuclideanDatas.get(k).setN2(u21);
  }
  else
  {
     lstEuclideanDatas.get(k).setN1(u12);
     lstEuclideanDatas.get(k).setN2(u22);
  }
}
{
}
// cari pembilang dan penyebut
pembilang = new double[2];
pembilang[0] = 0;
```

```
            pembilang[1] = 0;
            penyebut = new double[2];
            penyebut[0] = 0;
            penyebut[1] = 0;
            hasil = new double[2];

            for(int p=0; p<K; p++)
            {
               pembilang[0] = pembilang[0] + (lstEuclideanDatas.get(p).getN1() *
(Math.pow(1/lstEuclideanDatas.get(p).getNilaiTarget(),2)));
               pembilang[1] = pembilang[1] + (lstEuclideanDatas.get(p).getN2() *
(Math.pow(1/lstEuclideanDatas.get(p).getNilaiTarget(),2)));
               penyebut[0] = penyebut[0] +
(Math.pow(1/lstEuclideanDatas.get(p).getNilaiTarget(),2));
               penyebut[1] = penyebut[1] +
(Math.pow(1/lstEuclideanDatas.get(p).getNilaiTarget(),2));
            }

            //fknn
               hasil[0] = pembilang[0]/penyebut[0];
               hasil[1] = pembilang[1]/penyebut[1];
            if(hasil[0] > hasil[1])
            {
               hasilPrediksi.add(1);
            }
            else
            {
               hasilPrediksi.add(2);
            }
         }
      }
      public List<EuclideanData> getLstEuclideanDatas() {
         return lstEuclideanDatas;
      }
      public void setLstEuclideanDatas(List<EuclideanData> lstEuclideanDatas) {
         this.lstEuclideanDatas = lstEuclideanDatas;
      }
      public int getK() {
         return K;
      }
      public void setK(int K) {
         this.K = K;
      }
      public List<Integer> getHasilPrediksi() {
         return hasilPrediksi;
```

```
    }
    public void setHasilPrediksi(List<Integer> hasilPrediksi) {
        this.hasilPrediksi = hasilPrediksi;
    }
    public List<AtributData> getLstAtributDataLatih() {
        return lstAtributDataLatih;
    }
    public void setLstAtributDataLatih(List<AtributData> lstAtributDataLatih) {
        this.lstAtributDataLatih = lstAtributDataLatih;
    }
}
```

Source code* 4.5 *Fuzzy k-Nearest Neighbor

Screenshot program saat menampilkan hasil perhitungan *Fk-NN* ditunjukkan pada gambar 4.3 di bawah ini.



Gambar 4.3 *Screenshot* program saat menampilkan hasil perhitungan *Fk-NN*

BAB 5 PEMBAHASAN

Bab ini membahas tentang hasil yang didapat dari pengujian dan analisis yang dilakukan terhadap penerapan metode *Fuzzy k-Nearest Neighbor* untuk pengklasifikasian penyakit ginjal kronis.

5.1 Pengujian

Untuk mengetahui tingkat akurasi dari metode *Fuzzy k-Nearest Neighbor* yang diterapkan dalam pengklasifikasian penyakit ginjal kronis, maka perlu dilakukan pengujian terhadap sistem yang telah dibuat. Berikut adalah pengujian-pengujian yang dilakukan dalam penelitian ini.

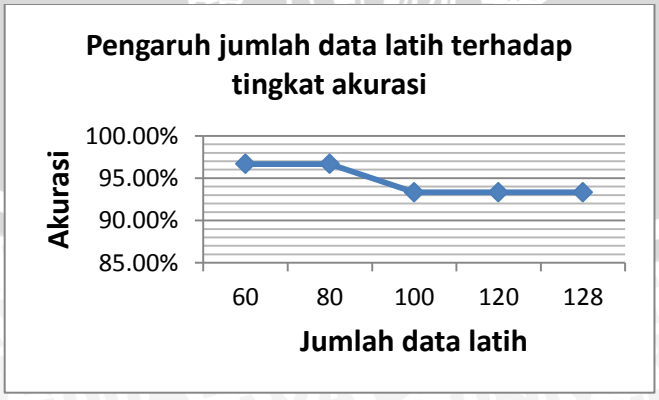
5.1.1 Pengujian tingkat akurasi terhadap data latih

Dalam pengujian ini, variasi jumlah data yang digunakan meliputi 60 data latih, 80 data latih, 100 data latih, 120 data latih, dan 128 data latih. Untuk data uji yang digunakan pada pengujian-pengujian tersebut adalah sebanyak 30 data uji dan merupakan data yang sama untuk setiap pengujian. Hasil dari pengujian ini ditunjukkan dalam tabel 5.1 di bawah ini, dengan menggunakan nilai k = 3.

Tabel 5.1 Hasil pengujian tingkat akurasi terhadap data latih

Jumlah data latih	Akurasi
60	96.67%
80	96.67%
100	93.33%
120	93.33%
128	93.33%

Grafik dari hasil pengujian tingkat akurasi terhadap data latih di atas dapat dilihat pada gambar 5.1 berikut.



Gambar 5.1 Grafik hasil pengujian tingkat akurasi terhadap data latih

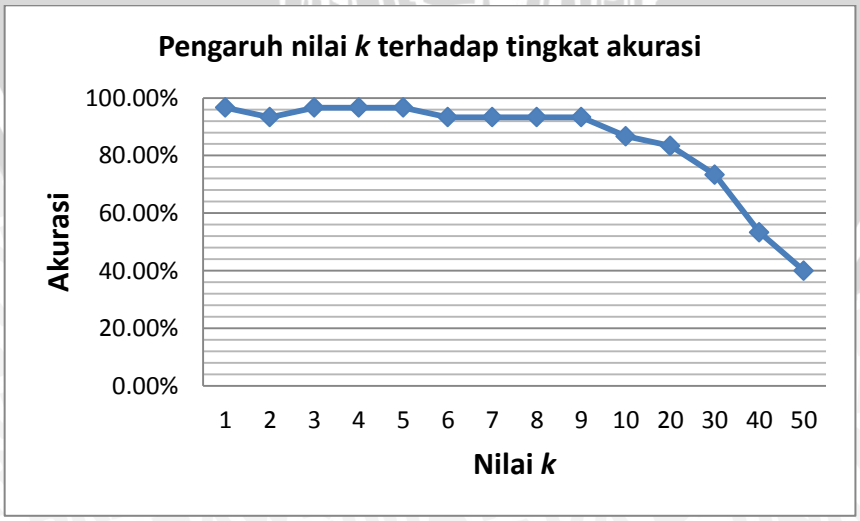
5.1.2 Pengujian tingkat akurasi terhadap nilai *k*

Dalam pengujian ini, data latih yang digunakan sebanyak 60 data. Dan yang digunakan sebagai data uji sebanyak 30 data. Hasil dari pengujian tingkat akurasi terhadap nilai *k* ini dapat dilihat pada tabel 5.2 di bawah ini.

Tabel 5.2 Hasil pengujian tingkat akurasi terhadap nilai *k*

Nilai *k*	Akurasi
1	96.67%
2	93.33%
3	96.67%
4	96.67%
5	96.67%
6	93.33%
7	93.33%
8	93.33%
9	93.33%
10	86.67%
20	83.33%
30	73.33%
40	53.33%
50	40.00%

Grafik dari hasil pengujian tingkat akurasi terhadap nilai *k* di atas dapat dilihat pada gambar 5.2 berikut.



Gambar 5.2 Grafik hasil pengujian tingkat akurasi terhadap nilai *k*

5.2 Analisis

Dari hasil pengujian-pengujian terhadap penerapan metode *Fuzzy k-Nearest Neighbor* untuk pengklasifikasian penyakit ginjal kronis yang telah dilakukan maka dilakukan analisis terhadap hasil pengujian yang didapat sebagai berikut ini.

5.2.1 Analisis hasil pengujian tingkat akurasi terhadap data latih

Dari hasil pengujian pengaruh jumlah data latih terhadap tingkat akurasi yang telah dilakukan dapat dilihat bahwa hasil akurasi tertinggi ada pada pengujian dengan 60 dan 80 data latih yaitu dengan hasil akurasi 96.67%, tingkat akurasi mengalami sedikit penurunan ketika dilakukan pengujian dengan data latih yang lebih banyak. Padahal seharusnya semakin banyak data latih yang digunakan akan menghasilkan akurasi yang semakin tinggi persentasenya, karena dengan semakin banyaknya data latih yang digunakan, maka sistem dapat mengenali pola yang ada pada dataset yang digunakan dalam pengujian. Namun dalam penelitian ini dataset yang digunakan memiliki sebaran kelas yang tidak seimbang atau *imbalance class*, tingkat *imbalance* data mempengaruhi hasil klasifikasi, semakin tinggi tingkat *imbalance* sebuah data, maka nilai *precision*, *recall*, dan *f-measure* pada kelas mayor semakin meningkat sedangkan untuk nilai *precision*, *recall*, dan *f-measure* pada kelas minor cenderung mengalami penurunan.

5.2.2 Analisis hasil pengujian tingkat akurasi terhadap nilai *k*

Pada pengujian pengaruh nilai *k* terhadap tingkat akurasi dapat dilihat hasil tingkat akurasi akan semakin menurun ketika pengujian dilakukan dengan nilai *k* yang semakin besar. Untuk kasus ini penurunan tingkat akurasi terjadi karena 30 data yang digunakan sebagai data uji memiliki 20 data yang termasuk dalam kelas 1 dan 10 data yang termasuk dalam kelas 2, maka semakin besar nilai *k*, akan semakin banyak data termasuk dalam kelas mayor yang sebenarnya bukan merupakan tetangga terdekat yang mempengaruhi data yang termasuk dalam kelas minor, sehingga kelas minor akan memiliki peluang yang besar untuk salah diklasifikasikan.

BAB 6 PENUTUP

6.1 Kesimpulan

Berikut ini adalah kesimpulan yang dapat ditarik dari hasil penerapan metode *Fuzzy k-Nearest Neighbor* (*Fk-NN*) untuk pengklasifikasian penyakit ginjal kronis, dan juga hasil pengujian dan analisa hasil yang dilakukan dalam penelitian ini

1. Metode *Fuzzy k-Nearest Neighbor (Fk-NN)* bisa diterapkan dalam pengklasifikasian penyakit ginjal kronis. Langkah awal yang dilakukan adalah menormalisasi data latih dan data uji, lalu mencari tetangga terdekat dengan persamaan *Euclidean Distance,* hingga pada akhirnya hasil dari data yang telah dilakukan perhitungan ditransformasikan kedalam bentuk *fuzzy,* maka akan diperoleh hasil klasifikasi penyakit ginjal kronis dengan menggunakan metode *Fuzzy k-Nearest Neighbor*
2. Berdasarkan pengujian tingkat akurasi yang dipengaruhi oleh banyaknya data latih yang digunakan, didapatkan tingkat akurasi sebesar 96.67% dengan 60 dan 80 data latih dan k = 3. Untuk pengujian tingkat akurasi terhadap nilai k dihasilkan akurasi sebesar 96.67% dengan k = 1 dan menggunakan 60 data latih.

6.2 Saran

Berikut ini adalah saran-saran yang diharapkan bisa diperhatikan sebagai pertimbangan guna pengembangan penelitian-penelitian selanjutnya.

1. Diharapkan pada penelitian-penelitian selanjutnya digunakan metode lain atau modifikasi dari *Fk-NN* untuk mendapatkan hasil yang lebih maksimal.
2. Diharapkan pada penelitian selanjutnya untuk bisa menggunakan data yang lebih baik lagi, misalnya yang lebih minim *missing value* dan sebagainya, agar pengujian bisa dilakukan dengan data yang lebih besar jumlahnya.

DAFTAR PUSTAKA

Dorland. 2002. Kamus Saku Kedokteran. Edisi ke-25 .Jakarta: Penerbit Buku Kedokteran EGC.

Dubey, Abhinandan. 2015. A Classification of CKD Cases Using MultiVariate K-Means Clustering. Dalam: International Journal of Scientific and Research Publications, Volume 5, Issue 8. Tersedia di: < http://www.ijsrp.org/research-paper-0815/ijsrp-p4437.pdf> [Diakses 26 Desember2015]

Farlex, 2012. Farlex Partner Medical Dictionary. Farlex, Inc.

Farlex, 2012. Medical Dictionary for the Health Professions and Nursing. Farlex, Inc.

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.

Han, Jiwei dan Kamber, Micheline. 2000. *Data Mining: Concepts and Technique*. Morgan Kaufmann Publishers.

Jayalakshmi, T. dan Santhakumaran, A. 2011. *Statistical Normalization and Backpropagation for Classification*. Tersedia di: <http://www.ijcte.org/papers/288-L052.pdf> [Diakses 27 November 2015]

Kantardzic, Mehmed. 2003. *Data Mining : Concepts, Models, Methods and Algorithm. John Wiley & Sons*. New York.

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.

Khusnawi. 2007. Pengantar Solusi Data Mining [online]. STMIK AMIKOM. Yogyakarta.

Kusrini dan Luthfi, Emha Tufiq. 2009. *Algoritma Data Mining*. Yogyakarta: Penerbit ANDI.

Kusumadewi, S dan Purnomo, H. 2003. Artificial Intelligence (Teknik & Aplikasinya). Graha Ilmu. Jogjakarta.

Kusumadewi, S dan Purnomo, H. 2010. Aplikasi Logika Fuzzy untuk Pendukung Keputusan: Jilid 2. Graha Ilmu. Yogyakarta.

Larose, Daniel T. 2005. *Discovering Knowledge in Data. An Introduction to Data Mining*. John Willey dan Sons. New Jersey.

Levey AS, Coresh J, Balk E, Kautz T, Levin A, Steves M et al. 2003. National Kidney Foundation Guidelines for Chronic Kidney Disease: Evaluation,

Classification, and Stratification. Ann Intern Med. USA : Annals of Internal Medicine.

Mc Phee SJ, Papadakis MA. 2008. Chronic Kidney Disease. In: Lange Current Medical Diagnosis & Treatment. California: Mc Graw Hill.

McGraw-Hill , 2002. McGraw-Hill Concise Dictionary of Modern Medicine. USA: The McGraw-Hill Companies, Inc.

Mooney, Jean. 2009. Illustrated Dictionary of Podiatry and Foot Science by Jean Mooney St. Louis, Mo: Elsevier.

Moradian, Mehdi dan Baarani, Ahmad. 2009. *KNNBA: k-Nearest-Neighbor-Based_Associaton Algorithm*. Tersedia di: <http://www.jatit.org/volume/researchpapers/Vol6No1/14Vol6No1.pdf> [Diakses 27 November 2015]

Mosby, 2008. Mosby's Dental Dictionary, 2nd edition. St. Louis, Mo: Elsevier, Inc.

Mosby, 2009. Mosby's Medical Dictionary, 9th edition. St. Louis, Mo: Elsevier.

Nahas, ME. 2003. The Patient with Failing Renal Failure. Dalam: Cameron JS, Davison AM. Oxford Textbook of Clinical Nephrology. Edisi ke-3. Oxford University Press.

Nugraha, Dany, dkk. 2006. Diagnosis Gangguan Sistem Urinari pada Anjing dan Kucing Menggunakan VFI 5. Institute Pertanian Bogor.

Parmar, MS. 2002. Chronic Renal Disease. United Kingdom: BMJ.

Pearce, E. 2006. Anatomi dan Fisiologi untuk Paramedis. Jakarta: PT Gramedia Pustaka Utama.

Pranay, K., Stoppler, M.C., 2010. Chronic Kidney Disease. Tersedia di: <www.emedicinehealth.com/chronic_kidney_disease/page18_em.htm#Authors%20and%20Editors> [Diakses pada 26 Desember 2015]

Price, S. A. dan Wilson, L. M. 2006. Patofisiologi : Konsep Klinis Proses-Proses Penyakit, Edisi 6, Volume 1. Jakarta: Penerbit Buku Kedokteran EGC.

Satyatama, Reviangga Dika. 2013. Klasifikasi Incomplete Data Penyakit Liver Pada Manusia Dengan Menggunakan Algoritma Voting Feature Interval-5 (VFI5). Universitas Brawijaya. Malang.

Sharon K. 2006. Chronic Kidney Disease. Critical Care Nurse.

Smeltzer C. Suzanne, Brunner & Suddarth. 2002. Buku Ajar Keperawatan Medikal Bedah. Jakarta: Penerbit Buku Kedokteran EGC.

Vijayakumar M, Namalwar R, Prahlad N. 2007. Prevention of Chronic Kidney Disease in Children. Ind J of Nephrol. New Delhi: Indian Journal of Nephrology.

Yanita C. 2013. Penerapan Fuzzy K-Nearest Neighbor untuk Menentukan Status Evaluasi Kinerja Karyawan. Universitas Brawijaya. Malang.

Yee, J., Krol, G. 2011. CKD-Chronic Kidney Disease: clinical Practice Recommendations for Primary Care Physicians and Healthcare Providers—A Collaborative Approach. 6th ed. Dynamic Marketing,Plymouth, MI.

Youngson, Robert M. 2004, 2005. Collins Dictionary of Medicine. Collins.

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