PENERAPAN *FUZZY K-NEAREST NEIGHBOR* (FKNN) UNTUK DIAGNOSA PENYAKIT LAYU PADA HUTAN BERDASARKAN *WILT DATASET*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Lintang Anginseto
NIM: 0910963087

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

PENGESAHAN

PENERAPAN ALGORITMA *FUZZY K-NEAREST NEIGHBOR* UNTUK DIAGNOSA PENYAKIT LAYU PADA HUTAN BERDASARKAN *WILT DATASET*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Lintang Anginseto
NIM: 0910963087

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

Dosen Pembimbing I	Dosen Pembimbing II
Edy Santoso, S.Si., M.Kom	Sutrisno, Ir., M.T
NIP: 19740414 200312 1 004	NIP: 195703251987011001

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, 23 Agustus 2016

Lintang Anginseto

NIM: 0910963087

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 "Penerapan Algoritma *Fuzzy k-Nearest Neighbor (FKNN)* Untuk Diagnosa Penyakit Layu Pada Hutan Berdasarkan *Wilt Dataset*". Tidak lupa penulis mengucapkan terima kasih pada semua pihak yang telah membantu dalam penyelesaian skripsi, terutama kepada:

1. Edy Santoso, S.Si., M.Kom., selaku dosen pembimbing I dan Ir.Sutrisno, M.T., selaku dosen pembimbing II yang telah membimbing penulis dalam pengerjaan skripsi ini dari awal hingga akhir.
2. Ibu Darmaslindri dan Bapak Sruwandono atas kesabaran, kasih sayang dan dukungan doa dan semangat yang tulus dan ikhlas kepada penulis semenjak kecil.
3. Bapak Issa Arwani yang telah membimbing dan mengarahkan mahasiswa Ilmu Komputer dan Teknik Informatika angkatan 2009 menuju kelulusan.
4. Mas Dana, Dinda, Arda dan Ovi.
5. Teman-teman NFS, Aries, Hendro, Muid dan Posank yang telah berjuang bersama menyelesaikan skripsi.
6. Hardika Teguh Wijaya dan Septyan Teguh Mahendra yang telah membantu dalam *progress* skripsi.
7. Teman-teman Teknik Informatika dan Ilmu Komputer yang telah memberikan dukungan, saran-saran dan berbagi informasi serta 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, 23 Agustus 2016

Penulis

anginseto@gmail.com

ABSTRAK

Hutan sangat penting bagi kehidupan di muka bumi dan kehidupan generasi mendatang. Kesalahan dalam pengelolaan hutan berarti membahayakan kehidupan generasi kita mendatang. Salah satu hal yang merusak kelestarian hutan adalah penyakit layu (*wilt disease*) pada pepohonan. *Wilt Disease* atau penyakit layu adalah gejala alami dari penyakit tanaman yang diakibatkan kekurangan air pada daun dan batang. Penelitian ini membahas tentang penerapan metode algoritma *Fuzzy K-Nearest Neighbor* (FKNN) untuk diagnosa penyakit layu (*Wilt Disease*) pada hutan. Sistem ini dirancang dan dibangun berdasarkan *Wilt Dataset* yang diambil dari website *UCI Machine Learning Repository* yang diambil pada tahun 2014. Dalam penelitian ini dilakukan beberapa pengujian untuk mengetahui tingkat akurasi sistem. Pengujian dilakukan terhadap 6 jumlah data latih yang berbeda yaitu 60, 90, 150, 220, 300 dan 350 data dengan data uji sebanyak 30 dan 60 data. Pengujian dilakukan untuk mengetahui pengaruh jumlah data latih dan nilai k terhadap tingkat akurasi sistem. Dari pengujian didapatkan hasil penelitian bahwa tingkat akurasi sistem sebesar 76.67% saat menggunakan 60 data latih dengan 30 data uji dan nilai k = 4. Tingkat akurasi sistem sebesar 71.67% saat nilai k bernilai 5 dengan menggunakan 200 data latih dan 60 data uji.

Kata kunci : *wilt disease*, FKNN, klasifikasi

ABSTRACT

Forests are essential for life on earth and the lives of future generations. Fault in forest management means endangering the lives of generations to come. One of the things that damage the preservation of the forest is a wilt disease (wilt disease) on trees. Wilt Disease or wilt disease is a natural symptom of plant diseases caused by lack of water on the leaves and stems. This study discusses the application of the algorithm method Fuzzy K-Nearest Neighbor (FKNN) for diagnosis wilt disease (wilt disease) in the forest. This system was designed and built by Wilt dataset taken from the website of the UCI Machine Learning Repository taken in 2014. In this research, conducted several tests to determine the level of accuracy of the system. Tests carried out on six different amount of training data that is 60, 90, 150, 220, 300 and 350 of data with test data as much as 30 and 60 data. Tests conducted to determine the effect of the amount of training data and k values of the level of accuracy of the system. From the test showed that the level of research amounted to 76.67% system accuracy when using training data 60 to 30, the test data and the value of k = 4. The accuracy of the system amounted to 71.67% when the value of k-value 5 using 200 training data and 60 test data.

Keywords: wilt disease, FKNN, classification

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Hutan sangat penting bagi kehidupan di muka bumi, terutama bagi kehidupan generasi mendatang. Kesalahan dalam pengelolaan hutan berarti menyiksa kehidupan generasi kita mendatang. Hutan merupakan kumpulan pepohonan yang tumbuh rapat beserta tumbuh-tumbuhan memanjat dengan bunga yang beraneka warna yang berperan sangat penting bagi kehidupan di bumi ini. Dari sudut pandang ekonomi, hutan merupakan tempat menanam modal jangka panjang yang sangat menguntungkan dalam bentuk Hak Pengusahaan Hutan (HPH) (Arifin, 2001).

Hutan sangat berperan sangat penting bagi kehidupan manusia karena memiliki banyak manfaat. Tumbuhan-tumbuhan di hutan dapat menyerap karbondioksida dalam jumlah besar, dan mengubahnya menjadi oksigen yang dibutuhkan manusia untuk bernafas. Hutan juga sebagai sumber penghasil kayu, buah-buahan, makanan, madu bahkan obat-obatan herbal bagi manusia. Selain itu hutan dapat menyimpan cadangan air yang tersimpan pada akar-akar tumbuhan hutan. Kemampuan tumbuhan-tumbuhan hutan untuk menyimpan air dalam jumlah besar ini membuat hutan dapat mencegah terjadi nya banjir. Akar-akar pada tumbuhan juga dapat menahan tanah agar tidak bergeser, hal ini dapat mencegah terjadi nya erosi. Lahan hutan yang luas dan tumbuhan serta tanaman yang tumbuh rapat membuat hutan sebagai habitat alami untuk berbagai macam jenis burung, serangga dan mamalia di dalam nya. Banyaknya jenis hewan dan tumbuhan ini dapat dijadikan sebagai riset dan studi lainnya.

Dengan banyaknya manfaat dan mengetahui pentingnya hutan berdasarkan penjelasan di atas, maka sangat penting untuk menjaga kelestarian hutan. Beberapa hal yang dapat merusak kelestarian hutan, seperti perkembangan populasi penduduk yang mengakibatkan perluasan area pemukiman dan usaha atau perusahaan di sekitar hutan yang mengakibatkan penebangan hutan yang kadang tidak terkontrol, kebakaran yang merusak hutan secara luas dan kerusakan karena penyebab alami seperti penyakit pada tanaman hutan. Penyebab alami ini dapat dikaitkan oleh perubahan iklim yang terjadi saat ini.

Salah satu penyakit tanaman yang mengganggu kelestarian hutan adalah penyakit layu (*wilt disease*). *Wilt disease* atau penyakit layu adalah gejala alami dari penyakit tanaman. Tanaman atau tumbuhan yang terkena penyakit layu dapat terlihat dari kondisi batang dan daun, warna batang dan daun, bahkan terlihat dari tanaman secara keseluruhan. Bagian tanaman yang terkena akan kehilangan ketebalannya dan menjadi layu, jika hal ini terjadi terus menerus akan menyebabkan tanaman menjadi mati. Gejala ini dapat disebabkan oleh kekurangan air, bakteri atau virus (Imas, 2012). Warna batang dan daun pada tumbuhan yang terkena penyakit layu juga berubah warna menjadi kuning atau kuning kecoklatan hingga berwarna coklat tua.

Jika tumbuhan menjadi layu dan mati, maka tumbuhan tidak dapat berfungsi sebagaimana yang dibutuhkan oleh manusia, yaitu sebagai penghasil oksigen dalam jumlah besar, buah-buahan, sumber makanan, kayu dan banyak manfaat lainnya tersebut. Selama ini penyakit layu dapat dideteksi berdasarkan penglihatan dekat secara mudah. Tetapi hal ini akan menjadi berbeda jika mendeteksi penyakit pada hutan yang memiliki lahan yang luas, diperlukan waktu dan tenaga untuk jika mendeteksi penyakit satu-persatu pada tumbuhan.

Diperlukan sebuah alat bantu atau metode untuk dapat mendeteksi penyakit layu ini pada lahan hutan yang luas. Dengan sebuah alat bantu berupa sistem dengan penerapan metode di dalamnya, diharapkan sistem ini dapat mendeteksi penyakit layu.

Salah satu metode yang dapat digunakan untuk klasifikasi adalah metode *K-Nearest Neighbor.* Metode bekerja relatif dengan cara yang lebih sederhana dibandingkan dengan metode pengklasifikasian data lainnya. Algoritma ini mengklasifikasikan data uji (data baru) berdasarkan K terdekat dengan data latih yang telah diklasifikasikan sebelumnya. Untuk menghitung dekat dan jauhnya tetangga dapat dilakukan dengan mengetahui jarak e*uclidian*. Hasil akhir dari algoritma ini adalah dengan melakukan voting terbanyak diantara klasifikasi K objek tetangga terdekat (Larose, 2005).

Salah satu varian dari algoritma K-nearest neighbor yaitu *Fuzzy K-Nearest Neighbor* yang merupakan metode klasifikasi yang menggabungkan teknik *Fuzzy* dan *K-Nearest Neighbor*. *Fuzzy K-Nearest Neighbor* memiliki beberapa keunggulan utama daripada algoritma K-NN. Pertama, algoritma ini mampu mempertimbangkan sifat ambigu dari tetangga jika ada. Dalam arti lain adalah algoritma ini melakukan proses lanjut terhadap hasil *voting* pada metode K-NN dalam penentuan kelas yang memiliki jumlah sama diantara kelas. Keunggulan kedua yaitu sebuah *instance* akan memiliki derajat nilai keanggotaan pada setiap kelas sehingga akan lebih memberikan kekuatan atau kepercayaan suatu *instance* berada pada suatu kelas. Hal tersebut diperlukan jika terdapat hasil *voting* yang memiliki jumlah kelas sama. Misalnya jika dalam proses akhir K-NN penentuan terkena penyakit layu = 2 dan lahan *cover* lain = 2, maka proses *fuzzy* lah yang akan mengambil keputusan, karena setiap *instance* memiliki e*uclidean distance* yang berbeda, sehingga setiap *instance* memiliki derajat keanggotaan yang berbeda-beda dan dapat menyelesaikan permasalahan dari hasil *voting* yang sama (Jowik, 2013)

Maka penelitian ini menggunakan metode *Fuzzy K-Nearest Neighbor* untuk klasifikasi penyakit. Harapannya dengan metode *Fuzzy K-Nearest Neighbor* akan menghasilkan suatu sistem yang dapat membantu dalam klasifikasi penyakit layu pada hutan dengan tingkat akurasi yang tinggi.

Berdasarkan uraian di atas maka penelitian ini diberi judul "Penerapan *Fuzzy K-Nearest Neighbor* (FKNN) Untuk Diagnosa Penyakit Pada Hutan Berdasarkan

Wilt Dataset" dengan klasifikasi berdasarkan *parameter GLCM mean texture, Mean Green value, Mean red value , Mean NIR value ,* dan *Standard Deviation*.

1.2 Rumusan Masalah

Dari latar belakang yang dibahas diatas maka dapat dirumuskan permasalah sebagai berikut :

1. Bagaimana penerapan metode *Fuzzy K-Nearest Neighbor* pada klasifikasi penyakit berdasarkan *Wilt Dataset* dengan parameter *GLCM mean texture, Mean green value, Mean red value , Mean NIR value , dan Standard Deviation.*
2. Berapa besar tingkat akurasi dari hasil klasifikasi *Wilt Dataset* menggunakan metode *Fuzzy k-Nearest Neighbor* yang dipengaruhi sejumlah nilai k (tetangga).

1.3 Batasan Masalah

Pada penelitian penerapan metode *Fuzzy k-Nearest Neighbor* pada klasifikasi penyakit layu ini diberikan batasan masalah yaitu :

1. Batasan subjek penelitian berupa *dataset* dari UCI *Machine Learning Repository*, yaitu *Wilt Dataset* yang memiliki dua klasifikasi (sakit dan lahan tutupan lainnya).
2. *Parameter* yang digunakan untuk menentukan klasifikasi, yaitu *GLCM mean texture, Mean green value, Mean red value , Mean NIR value ,* dan *Standard Deviation* dengan jumlah total data sebanyak 410 data berdasarkan Wilt Dataset yang diambil dari webiste UCI *Machine Learning Repository*.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah:

1. Menerapkan metode *Fuzzy k-Nearest Neighbor* pada *Wilt Dataset* untuk klasifikasi penyakit layu.
2. Memperoleh hasil tingkat akurasi klasifikasi yang dipengaruhi sejumlah nilai k pada *Wilt Dataset* menggunakan metode *Fuzzy K-Nearest Neighbor*.

1.5 Manfaat Penelitian

1. Dapat menerapkan metode *Fuzzy k-Nearest Neighbor* pada sebuah sistem untuk mendiagnosa penyakit layu.
2. Dapat dijadikan referensi untuk penelitian selanjut nya tentang diagnosa penyakit layu dan penerapan algoritma *Fuzzy k-Nearest Neighbor*.

1.6 Sistematika Penulisan

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB 1 PENDAHULUAN

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

BAB 2 TINJAUAN PUSTAKA

Bab ini berisi tentang dasar teori dan referensi yang mendasari dan mendukung penelitian dalam penerapan metode *Fuzzy K-Nearest Neighbor* untuk klasifikasi penyakit layu (*wilt disease*).

BAB 3 METODOLOGI

Bab ini berisi langkah-langkah yang dilakukan untuk mendiagnosis penyakit layu dengan menggunakan metode *Fuzzy K-Nearest Neighbor* dengan subyek *wilt dataset*

BAB 4 PERANCANGAN

Bab ini membahas tentang rancangan implementasi metode algoritma *Fuzzy K-Nearest Neighbor* untuk diagnosa penyakit layu pada sistem perangkat lunak.

BAB 5 IMPLEMENTASI DAN PEMBAHASAN

Bab ini berisi pembahasan hasil implementasi yang berupa perangkat lunak dengan menggunakan metode algoritma *Fuzzy K-Nearest Neighbor* untuk diagnosa penyakit penyakit layu.

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini berisi tentang metode untuk menguji tingkat akurasi klasifikasi dengan menggunakan metode algoritma *Fuzzy K-Nearest Neighbor* dan analisis hasil pengujian agar dapat ditarik kesimpulan.

BAB 7 PENUTUP

Bab ini berisi kesimpulan yang diperoleh berdasarkan analisis pengujian dan saran untuk penelitian selanjut nya.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan berisi uraian dan pembahasan tentang teori, konsep, model, metode, atau sistem dari literatur ilmiah, yang berkaitan dengan tema, masalah, atau pertanyaan penelitian. Dalam landasan kepustakaan terdapat landasan teori dari berbagai sumber pustaka yang terkait dengan teori dan metode yang digunakan dalam penelitian. Juga terdapat kajian pustaka yang diambil dari penelitian-penelitian sebelumnya untuk mendukung penelitian penerapan metode algoritma *Fuzzy K-Nearest Neighbor* untuk diagnosa penyakit layu.

2.1 Penyakit Layu (*Wilt Disease*)

Kelayuan pada tanaman terutama pada bagian daun, tunas atau tanaman secara keseluruhan, dapat disebabkan karena hilangnya turgor pada bagian-bagian tersebut. Hilangnya turgor tersebut dapat disebabkan karena adanya gangguan di dalam berkas pembuluh/pengangkutan atau adanya kerusakan pada susunan akar, yang menyebabkan tidak seimbangnya penguapan dengan pengangkutan air. Penyakit layu (*wilt disease*) pada tanaman dapat disebabkan oleh faktor biotik yaitu bakteri sehingga disebut layu bakteri (*Pseudomonas solanacearum*) atau oleh jamur/cendawan yang disebut penyakit layu *Fusarium* (*Fusarium oxysporum*). Selain karena penyakit *biotik*, kelayuan pada tanaman juga dapat disebabkan karena faktor *abiotik* (kekurangan air). Berikut adalah karakteristik kelayuan pada tanaman, penyebab dan cara penyebaran nya (Aisyah, 2012). Gambar 2.1 menunjukkan pohon Ek (*Oak*) yang terkena penyakit layu.

Gambar 2.1 Pohon Ek yang terkena penyakit layu

2.2 Hutan

Hutan adalah suatu wilayah yang memiliki banyak tumbuh-tumbuhan lebat yang berisi antara lain pohon, semak, paku-pakuan, rumput, jamur dan lain sebagainya serta menempati daerah yang cukup luas. Hutan berfungsi sebagai penampung karbondioksida (carbon dioxide sink), habitat hewan, modulator arus hidrologika, dan pelestari tanah serta merupakan salah satu aspek biosfer bumi yang paling penting. Hutan adalah bentuk kehidupan yang tersebar di seluruh dunia. Kita dapat menemukan hutan baik di daerah tropis maupun daerah beriklim dingin, di dataran rendah maupun di pegunungan, di pulau kecil maupun di benua besar (Arief, 2001).

Menurut Undang-undang Nomor 41 Tahun 1999 tentang Kehutanan, pengertian hutan adalah suatu kesatuan ekosistem berupa hamparan lahan berisi sumberdaya alam hayati yang didominasi pepohonan dalam persekutuan alam lingkungan, yang satu dengan yang lainnya tidak dapat dipisahkan. Definisi hutan yang disebutkan di atas, terdapat unsur-unsur yang meliputi:

1. Suatu kesatuan ekosistem
2. Berupa hamparan lahan
3. Berisi sumberdaya alam hayati beserta alam lingkungannya yang tidak dapat dipisahkan satu dengan yang lainnya.
4. Mampu memberi manfaat secara lestari

Keempat ciri pokok dimiliki suatu wilayah yang dinamakan hutan, merupakan rangkaian kesatuan komponen yang utuh dan saling ketergantungan terhadap fungsi ekosistem di bumi. Eksistensi hutan sebagai subekosistem global menenpatikan posisi penting sebagai paru-paru dunia (Zain 1996).

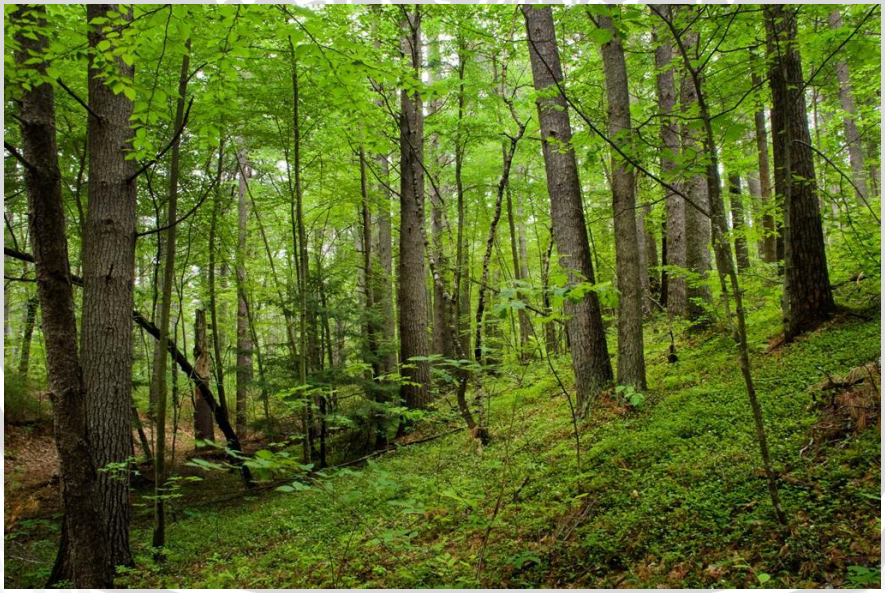
Di permukaan bumi ini, kurang lebih terdapat 90% biomassa yang terdapat di dalam hutan berbentuk kayu, dahan, daun, akar, dan sampah hutan (serasah), hewan, dan jasad renik. Biomassa ini merupakan hasil fotosintesis berupa selullosa, lignin, gula bersama dengan lemak, pati, protein, damar, fenol, dan berbagai unsur lain yg dibutuhkan tumbuhan melalui perakaran. Biomassa inilah yang merupakan kebutuhan makhluk di atas bumi melalui mata rantai antara binatang dan manusia dalam proses kebutuhan CO2 yang diikat dan O2 yang dilepas.

Secara sederhana, hutan ahli kehutanan mengartikan hutan sebagai suatu komunitas biologi yang didominasi oleh pohon-pohonan tanaman keras. Sedangkan menurut UU No. 5 tahun1967, hutan diartikan sebagai lapangan pertumbuhan pohon-pohon yang secara menyeluruh merupakan persekutuan hidup alam hayati beserta alam lingkungannya.

Hutan diartikan sebagai suatu asosiasi sehingga antara jenis pohon yang satu dan jenis pohon lain yang terdapat di dalamnya akan saling tergantung. Jenis-jenis

tanaman yang tidak menyukai sinar matahari penuh tentu memerlukan perlindungan dari tanaman yang lebih tinggi dan suka akan sinar matahari penuh. Tanaman yang suka sinar matahari penuh akan memperoleh keuntungan dari tanaman yang hidup di bawahnya karena mampu menjaga kelembaban dan suhu yang diperlukan oleh tanaman tinggi tersebut. Cahaya matahari yang sampai di lantai hutan tropika secara menyeluruh adalah sebesar 1,0%-1,7% yang dihitung berdasarkan waktu (jam). Pada pukul 12.00 (siang), saat matahari datang tegak lurus sebesar 100%, maka sinar akan sampai di lantai hutan sebesar 0%-1%. Pada pukul 15.00 saat sinar matahari condong 450, maka sebesar 67% sinar akan sampai di lantai hutan adalah 0%-0,5 %. Pada pukul 16.00 sinar matahari condong 300, kekuatan sebesar 44% sinar matahari yang akan sampai di lantai hutan adalah sebesar 0%-0,2% (Arief, 2001).

Selain terjadi ketergantungan, di dalam hutan akan terjadi pula persaingan antar anggota-anggota yang hidup saling berdekatan, misalnya persaingan di dalam penyerapan unsur hara, air, sinar matahari ataupun tempat tumbuh. Persaingan ini tidak hanya terjadi pada tumbuhan saja, tetapi juga pada binatang. Hutan merupakan suatu ekosistem natural yang telah mencapai keseimbangan klimaks dan merupakan komunitas tumbuhan yang paling besar yang mampu pulih kembali dari perubahan-perubahan yang dideritanya sejauh tidak melampaui batas-batas yang dapat ditoleransi. Berikut dibawah contoh gambar hutan pinus dan ek pada Gambar 2.7



Gambar 2.2 Hutan Pinus dan Ek (*Oak*)

2.3 Grey-Level Co-Occurrence Matrix (GLCM)

Grey Level Co-Occurrence Matrix juga disebut sebagai *Grey Tone Spatial Dependency Matrix*, adalah sebuah tabulasi seberapa sering kombinasi dari nilai-nilai kecerahan piksel (tingkat abu-abu) yang berbeda terjadi pada gambar (Hall-Beyer,2016).

Grey Level Co-Occurrence Matrix merupakan jarak dan ruang sudut hubungan pada gambar *sub-region* dengan *size* tertentu.

GLCM merupakan salah satu metode untuk memperoleh ciri statistik orde dua dengan cara menghitung probabilitas hubungan ketetanggaan antara dua piksel pada jarak dan orientasi sudut tertentu. Pendekatan proses GLCM bekerja dengan membentuk sebuah matriks kookurensi data citra, dilanjutkan dengan menentukan ciri sebagai fungsi dari matriks antara tersebut. *Co-occurrence* (kookurensi) berarti kejadian bersama, yaitu jumlah kejadian satu level nilai piksel bertetangga dengan satu level nilai piksel lain dalam jarak (d) dan orientasi sudut (θ) tertentu. Jarak dinyatakan dalam piksel dan orientasi dinyatakan dalam derajat. Orientasi dibentuk dalam empat arah sudut dengan interval sudut 45°, yaitu 0°, 45°, 90°, dan 135°. Sedangkan jarak antar piksel biasanya ditetapkan sebesar 1 piksel (Kasim et al., 2014).

2.4 *Near-infra Red* (*NIR*)

Near-infra Red merupakan gelombang elektromagnetik, juga bersifat dualism sebagai partikel cahaya dengan panjang gelombang 800-2500 nm atau bilangan gelombang 12,500-4000. Panjang gelombang merupakan invers dari bilangan gelombang. Jadi pengertian bilangan gelombang 12,5000 adalah dalam setiap meter, NIR bergetar sebanyak 12,500 kali sehingga setiap gelombang yang dihasilkan NIR berukuran 800 nm.

Infrared juga digunakan untuk remote control, satelit dan lain sebagainya. *Infrared* dibagi menjadi tiga region yaitu *far, mid* dan *near infrared* walau belum ada konsesus secara umum untuk pembagian panjang gelombang tersebut. NIR secara umum dianggap memiliki panjang gelombang 800-2500 nm. Ukuran panjang gelombang yang kecil tersebut dapat menembus molekul bahan organik (Adnan, 2013).

2.5 Data Mining

2.5.1 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.

2.5.2 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, 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

2.6 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.

2.7 Algoritma *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, 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, 2009).

2.7.1 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.

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

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*.

2.8 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).

2.8.1 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).

2.9 *Fuzzy k-Nearest Neigbor*

Fuzzy k-Nearest Neighbor (FKNN) merupakan metode klasifikasi yang menggabungkan teknik *Fuzzy* dengan *k-Nearest Neighbor classifier*. Algoritma *Fuzzy k-Nearest Neighbor* memberikan nilai keanggotaan kelas pada vektor sampel dan bukan menempatkan vektor pada kelas tertentu. Metode ini menggunakan nilai derajat keanggotaan data uji pada setiap kelas untuk melakukan prediksi terhadap data uji yang di inputkan ke dalam sistem. 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* mengeneralisasi teori *k-Nearest Neighbor* 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).

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

Keterangan:

u_i = Fungsi nilai keanggotaan

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 kontribusi jarak kedekatan pada masing-masing tetangga dengan nilai keanggotaan. Contoh, jika m = 2, 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).

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

Keterangan:

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

K = Total data latih yang digunakan

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

2.9.1 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.

2.10 Perhitungan Akurasi

Akurasi merupakan seberapa dekat suatu angka hasil pengukuran terhadap angka sebenarnya (*true value* atau *reference value*). Dalam penelitian ini akurasi diagnosis dihitung dari jumlah diagnosis yang tepat dibagi dengan jumlah data. Tingkat akurasi diperoleh dengan perhitungan sesuai dengan persamaan 2.5 (Nugraha, 2006).

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

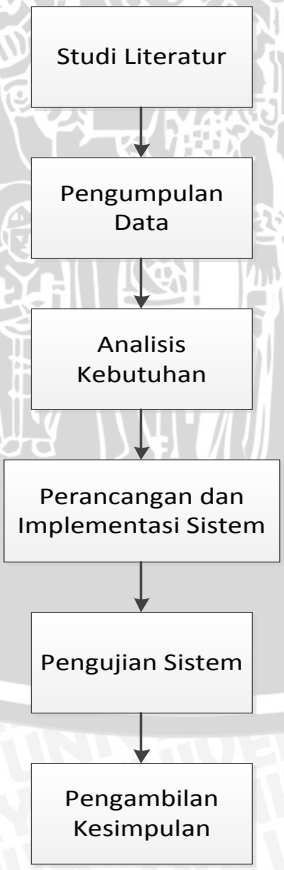
Jumlah prediksi benar adalah jumlah *record* data uji yang diprediksi kelasnya menggunakan metode klasifikasi dan hasilnya sama dengan kelas sebenarnya. Sedangkan jumlah total prediksi adalah jumlah keseluruhan *record* yang diprediksi kelasnya (seluruh data uji).

BAB 3 METODOLOGI

Dalam bab ini dibahas metode dan langkah-langkah yang dilakukan dalam penelitian menggunakan metode *Fuzzy k-Nearest Neighbor* untuk diagnosa penyakit layu (*Wilt Disease*).

Penelitian dilakukan dengan tahap-tahap sebagai berikut :

1. Mempelajari Studi literatur yang terkait dengan masalah penyakit layu dan metode *Fuzzy k-Nearest Neighbor*.
2. Mengumpulkan data dari *Wilt Dataset*.
3. Menganalisis sistem dari data yang terkumpul dan melakukan perancangan sistem yang meliputi pelatihan dan pengujian data.
4. Mengimplementasikan hasil analisis dengan membuat perangkat lunak berdasarkan perancangan yang telah dilakukan dengan menggunakan bahasa pemrograman *Java*.
5. Melakukan uji coba pada perangkat lunak dengan menggunakan data training dan data test.
6. Mengevaluasi hasil uji coba perangkat lunak berdasarkan tingkat akurasi terhadap diagnosis penyakit layu.



Gambar 3.1 Tahap-tahap penelitian

3.1 Studi Literatur

Dalam penelitian ini dibutuhkan studi literatur untuk merealisasikan tujuan dan menyelesaikan masalah. Teori-teori mengenai penyakit layu dan metode algoritma *Fuzzy k-Nearest Neighbor* digunakan sebagai dasar penelitian yang berasal dari buku-buku, *e-book* maupun artikel dari internet dan jurnal nasional maupun jurnal internasional. Kemudian data yang diperoleh, dipilah dan diubah sehingga dapat digunakan untuk analisis. Baru hasil analisis tersebut dapat diimplementasikan ke dalam program.

3.2 Data Penelitian

Data yang digunakan dalam penerapan metode *Fuzzy k-Nearest Neighbor* ini menggunakan *Wilt Dataset* tahun 2014 yang diambil dari situs http://archive.ics.uci.edu/ml/*dataset*s.html. Pada *Wilt dataset*, atribut yang digunakan adalah *GLCM mean texture, Mean green value, Mean red value , Mean NIR value ,* dan *Standard deviation*. Kelas output nya yaitu:

1 = Lahan pepohonan yang sakit.

2 = Lahan cover lain.

3.3 Analisa Sistem

3.3.1 Deskripsi Umum Sistem

Sistem yang dibuat di dalam penelitian ini merupakan perangkat lunak yang mengimplementasikan algoritma *Fuzzy k-Nearest Neighbor* (FKNN) untuk memprediksi *wilt disease* pada hutan.

Perangkat lunak ini akan menguji keakuratan hasil klasifikasi *dataset* penyakit layu. Parameter uji yang berkaitan dengan nilai k (tetangga) dan data latih yang berpengaruh terhadap tingkat akurasi.

3.3.2 Batasan Dan Perancangan Sistem

Batasan system yang dibuat yaitu input sistem adalah menggunakan *wilt dataset* dengan tahun 2014.

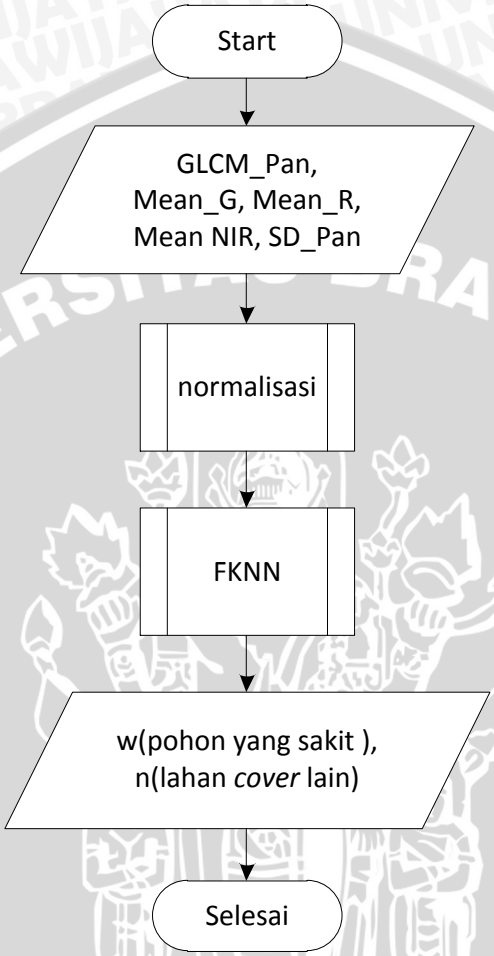
3.3.3 Perancangan Sistem

Tahap prediksi sistem FKNN untuk prediksi *wilt disease*, langkah - langkahnya adalah sebagai berikut:

1. Menginputkan data latih dan data uji *wilt dataset*.
2. Melakukan normalisasi data.
3. Melakukan perhitungan k-NN.
4. Transformasi output data ke dalam data *Fuzzy*.
5. Mendapatkan hasil prediksi kelas positif *wilt disease* dan yang bukan.
6. Menentukan nilai k.

7. Mendapatkan pengaruh nilai k terhadap tingkat akurasi klasifikasi dalam bentuk presentase.

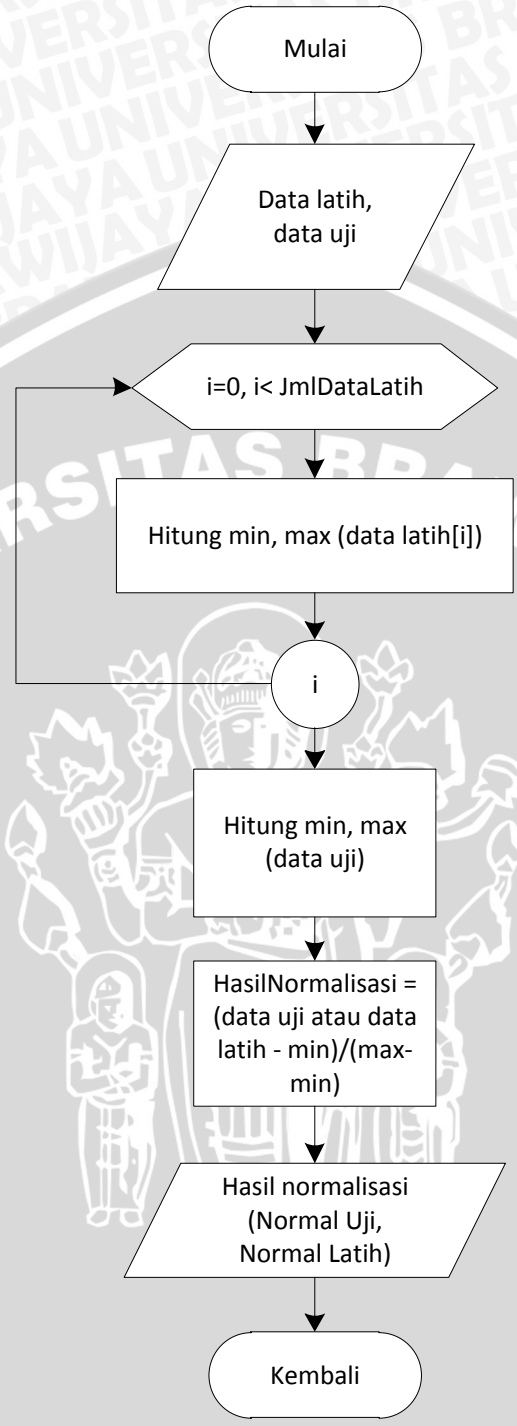
Tahapan proses klasifikasi dapat dlihat pada Gambar 3.2.



Gambar 3.2 Alur proses klasifikasi

3.3.3.1 Normalisasi Wilt Dataset

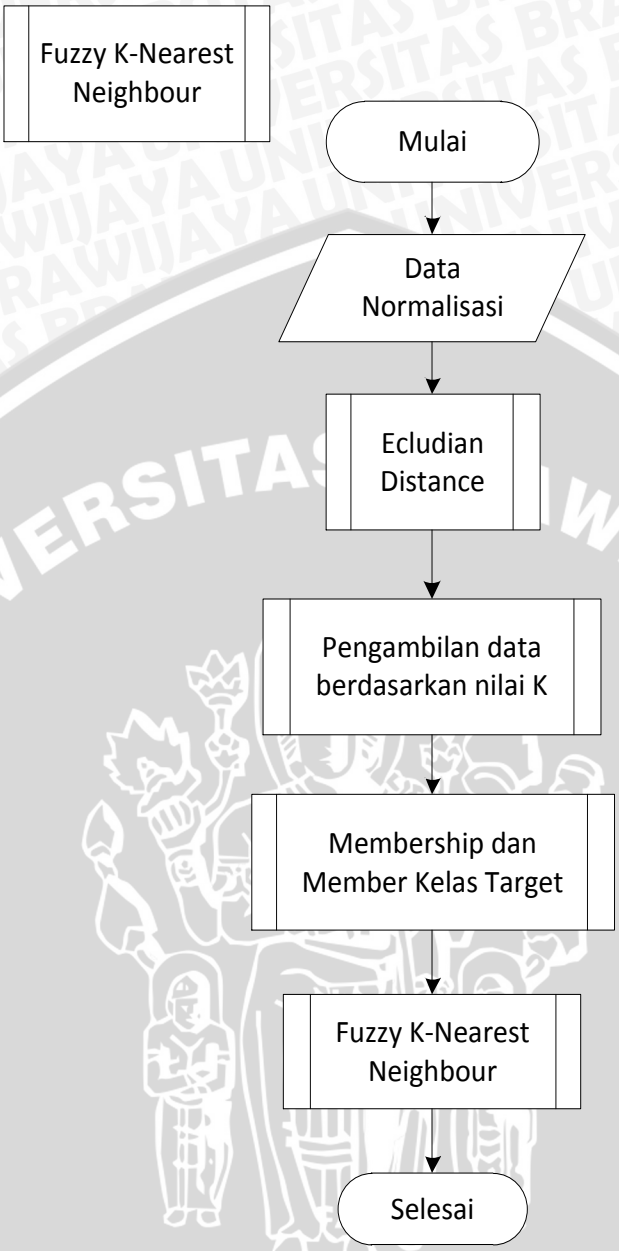
Proses normalisasi merupakan proses transformasi berupa normalisasi terhadap nilai dari setiap atribut untuk memberlakukan skala pengaruh yang ada pada atribut terhadap hasil. Proses normalisasi yang digunakan adalah min-max normalization. Diagram alur proses normalisasi data ditunjukkan pada Gambar 3.3.



Gambar 3.3 Proses normalisasi data

3.3.3.2 Proses Fuzzy K-Nearest Neighbor

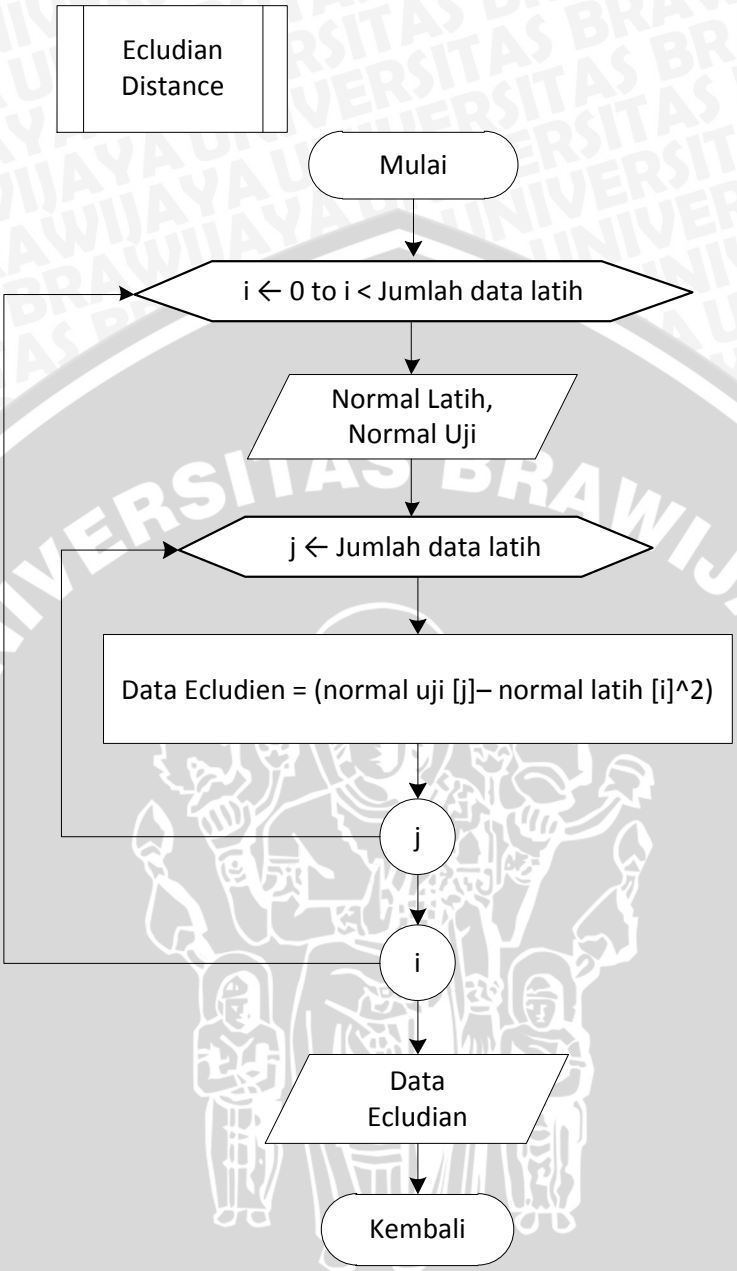
Tahapan ini merupakan proses perhitungan nilai jarak kedekatan tetangga data uji terhadap data latih menggunakan *Euclidean Distance*. Flowchart dari proses *Fuzzy k-Nearest Neighbor* ditunjukkan pada Gambar 3.4.



Gambar 3.4 Alur proses *Fuzzy k-Nearest Neighbor*

Input dalam proses algoritma *Fuzzy k-Nearest Neighbor* adalah data latih dan data uji yang sebelum nya telah di normalisasi menggunakan normalisasi *min-max.* Selanjutnya di hitung nilai Euclidean Distance terlebih dahulu. Lalu di tentukan nilai k untuk di ambil data sejumlah nilai k yang di masukkan untuk di hitung membership dan member kelas target nya. Setelah itu baru di hitung ke dalam algoritma *Fuzzy k-Nearest Neighbor.*

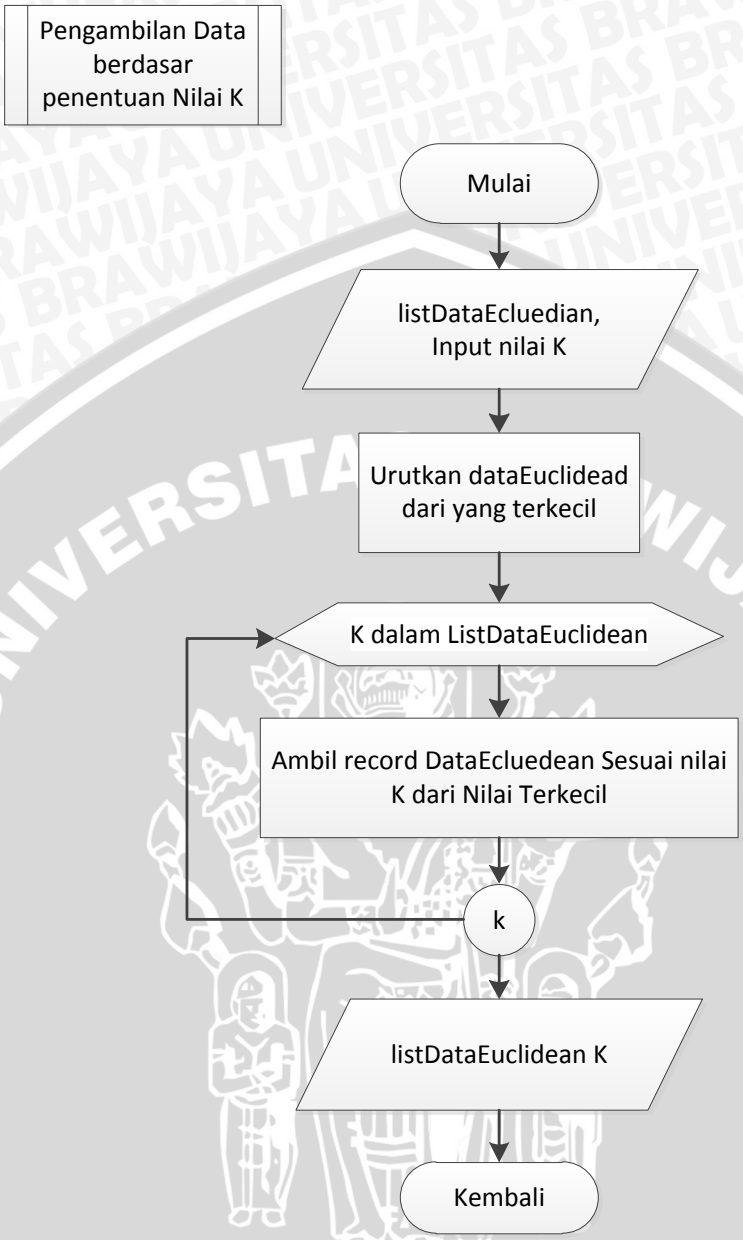
Diagram alir untuk proses pencarian *Euclidean Distance* ditunjukkan pada Gambar 3.5.



Gambar 3.5 Alur proses *Euclidean Distance*

Input pada proses adalah data latih dan data uji yang telah di normalisasi menggunakan normalisasi *min-max.* Lalu data latih dan data uji di hitung dengan rumus *Euclidean Distance.* Setelah itu nilai yang di dapatkan akan di urutkan dan di ambil berdasarkan jumlah k-nya.

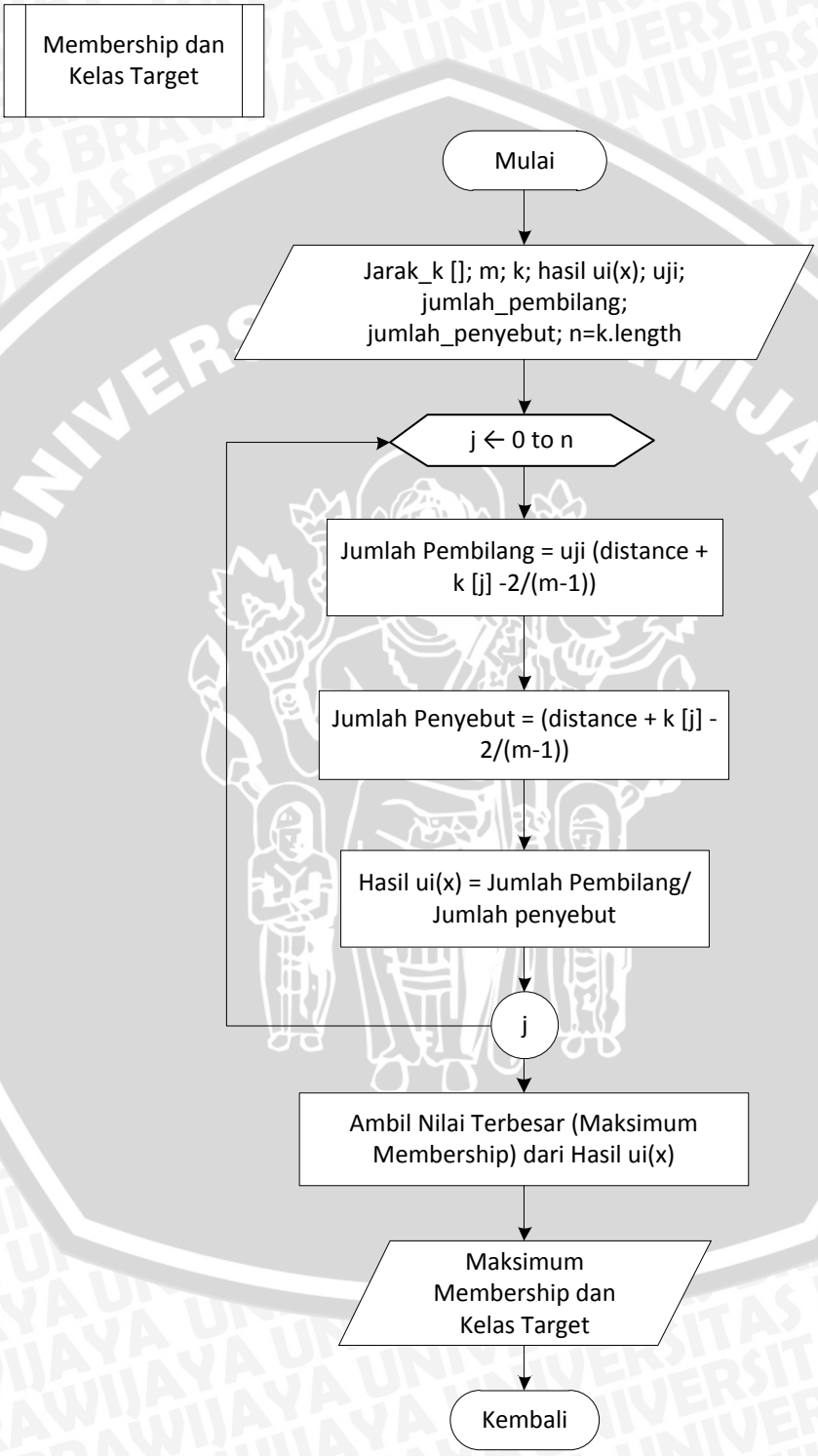
Diagram alir untuk proses pengambilan data berdasarkan jumlah k ditunjukkan pada Gambar 3.6.



Gambar 3.6 Alur proses pengambilan data berdasarkan jumlah k

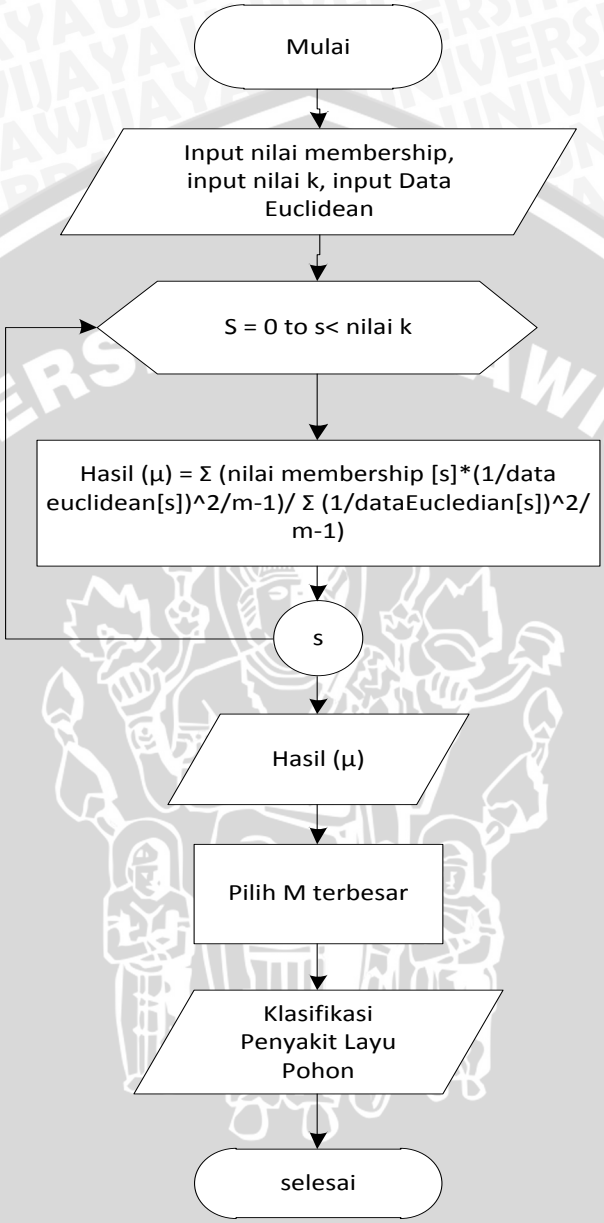
Input di dalam proses pengambilan data berdasarkan jumlah k adalah hasil perhitungan *euclidean distance* dan nilai k. Lalu data *euclidean distance* tadi di urutkan dari nilai yang terkecil. Lalu di ambil sejumlah data atau *record* sesuai dengan nilai k yang telah di inputkan sebelumnya.

Diagram alir untuk proses perhitungan membership ditunjukkan pada Gambar 3.7.



Gambar 3.7 Alur proses perhitungan membership

Fuzzy k-Nearest Neighbor (FKNN) merupakan proses perhitungan nilai membership dengan dua kelas, yaitu *wilt disease* dan lahan. Diagram alir untuk proses FKNN ditunjukkan pada Gambar 3.8.



Gambar 3.8 Alur proses perhitungan *Fuzzy k-Nearest Neighbor*

3.4 Contoh Perhitungan Manual

Pada subbab ini menampilkan contoh perhitungan manual yang telah dilakukan untuk proses diagnosa penyakit layu dengan menggunakan seluruh atribut *dataset* yang berjumlah 6 buah. Data yang digunakan sebanyak 21 buah dengan rincian record ke-1 sampai ke-18 adalah data latih dan record ke-19 adalah data uji, dengan k=3.

3.4.1 Data Latih dan Data Uji Sistem

Pada contoh perhitungan kali ini digunakan 60 data latih dengan 1 data uji. Data latih dan data uji yang dipakai pada Tabel 3.1 dan Tabel 3.2.

Tabel 3.1 Data latih sistem

No	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan	Kelas
1	120.363	205.500	119.395	416.581	20.676	1
2	124.740	202.800	115.333	354.333	16.707	1
3	134.692	199.286	116.857	477.857	22.497	1
4	127.946	178.368	92.368	278.474	14.977	1
5	135.432	197.000	112.690	532.952	17.604	1
6	118.348	226.150	138.850	608.900	29.073	1
7	135.436	184.500	95.143	309.190	13.055	1
8	121.170	226.000	146.214	595.571	22.809	1
9	131.127	232.784	144.588	563.843	11.949	1
10	134.498	210.212	116.909	594.848	27.938	1
11	131.165	206.682	114.727	483.045	22.904	1
12	136.096	201.400	111.400	444.600	16.554	1
13	122.632	230.561	142.195	473.707	24.522	1
14	113.656	213.107	129.786	480.857	29.238	1
15	133.894	191.622	115.514	419.514	17.489	1
16	142.868	228.679	104.393	674.964	32.400	2
17	143.497	211.609	95.739	612.348	29.530	2
18	135.725	238.917	107.583	741.271	20.841	2
19	112.825	236.920	108.040	705.840	22.671	2
20	111.208	233.100	105.575	729.775	23.674	2
21	124.955	226.696	100.565	686.435	27.871	2
22	115.850	235.894	107.106	757.213	19.602	2
23	124.970	218.095	97.429	595.857	40.781	2
24	130.712	169.527	69.401	187.684	8.876	2
25	113.465	244.259	107.556	735.519	34.575	2
26	130.118	230.231	104.192	689.346	16.023	2
27	120.414	216.316	104.895	617.316	29.404	2
28	127.223	219.429	100.571	588.857	17.274	2

Tabel 3.1 Data latih sistem (lanjutan)

No	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan	Kelas
29	128.945	196.000	93.000	437.000	27.336	2
30	133.359	166.000	84.000	247.000	14.102	2
31	131.111	213.667	125.444	537.889	19.435	1
32	127.836	201.358	116.509	372.132	23.929	1
33	121.611	199.745	121.064	338.128	15.019	1
34	106.242	221.750	140.000	381.167	18.903	1
35	147.771	220.111	123.722	479.167	23.337	1
36	105.100	206.800	97.000	564.200	21.304	2
37	132.387	203.733	95.800	531.867	53.597	2
38	107.577	213.308	90.769	514.154	45.348	2
39	127.766	187.000	82.357	310.143	15.444	2
40	124.018	170.927	69.073	217.836	13.337	2
41	132.299	203.407	90.000	495.407	36.944	2
42	153.542	204.000	87.000	452.000	1.700	2
43	125.739	169.203	69.416	206.411	7.412	2
44	140.857	175.750	79.917	293.000	6.016	2
45	132.592	164.683	68.683	177.898	8.124	2
46	138.240	198.250	87.056	445.389	27.264	2
47	136.251	191.686	87.176	441.765	20.391	2
48	130.011	199.182	95.273	520.636	12.187	2
49	120.872	226.387	101.710	746.806	16.683	2
50	144.340	174.400	77.650	302.100	10.067	2
51	106.344	231.800	111.800	726.400	28.667	2
52	141.351	224.444	107.111	668.667	30.064	2
53	128.177	206.500	94.438	456.875	23.224	2
54	127.479	206.818	93.591	525.432	18.795	2
55	130.767	217.000	97.400	648.400	24.677	2
56	123.854	213.273	100.568	645.273	34.318	2
57	122.674	183.333	80.667	390.778	20.084	2
58	119.149	200.000	93.524	488.619	29.795	2

Tabel 3.1 Data latih sistem (lanjutan)

No	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan	Kelas
59	111.151	193.059	88.353	493.824	33.466	2
60	109.639	206.556	97.667	463.556	20.259	2

Keterangan :

Kelas : 1 = *wilt disease*; 2 = lahan *cover* lain

Data-data tersebut memiliki klasifikasi 2 buah kelas sesuai dengan nilai target output, kedua kelas tersebut adalah kelas pepohonan yang sakit dan kelas lahan cover lain.

Tabel 3.2 Data uji sistem

No	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan	Kelas
1	109.829	183.700	82.950	251.750	16.079	2

3.4.2 Normalisasi Terhadap Nilai Atribut

Normalisasi yang dilakukan terhadap nilai tiap atribut dalam penelitian ini dengan menggunakan normalisasi min-max. Hal yang pertama dilakukan dalam normalisasi ini adalah mencari *range* dari setiap atribut dari *dataset*, yang ditunjukkan pada Tabel 3.3.

Tabel 3.3 Nilai minimum, maksimum dan range

Nilai	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan
min.	105.100	164.683	68.683	177.898	1.700
max.	153.542	244.259	146.214	757.213	53.597
range.	48.442	79.576	77.531	579.315	51.897

Berikut ini merupakan contoh normalisasi nilai terhadap atribut *Grey Level Co-Occurency Matrix Pan* untuk data latih :

V = 120.362 (nilai record pertama atribut GP)

minA = 105.100

maxA = 153.542

setelah diketahui nilai minimum dan maksimum maka dilakukan perhitungan menggunakan persamaan 2.5.

$$v' = \frac{v - \min_a}{\max_a - min_a} \quad (2.5)$$

$$= \frac{120.363 - 105.100}{153.542 - 105.100}$$

= 0.315

Dari hasil normalisasi didapatkan nilai untuk atribut *Gray Level Co-occurency Matrix Pan* yaitu 0.315. Selanjutnya proses yang sama juga dilakukan terhadap nilai-nilai dari atribut yang lainnya. Hasil perhitungan untuk seluruh atribut pada data latih ditampilkan pada Tabel 3.4 dan data uji pada Tabel 3.5.

Tabel 3.4 Data latih setelah normalisasi

no.	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan
1	0.315	0.513	0.654	0.412	0.366
2	0.405	0.479	0.602	0.305	0.289
3	0.611	0.435	0.621	0.518	0.401
4	0.472	0.172	0.305	0.174	0.256
5	0.626	0.406	0.568	0.613	0.306
6	0.273	0.772	0.905	0.744	0.527
7	0.626	0.249	0.341	0.227	0.219
8	0.332	0.771	1.000	0.721	0.407
9	0.537	0.856	0.979	0.666	0.197
10	0.607	0.572	0.622	0.720	0.506
11	0.538	0.528	0.594	0.527	0.409
12	0.640	0.461	0.551	0.460	0.286
13	0.362	0.828	0.948	0.511	0.440
14	0.177	0.609	0.788	0.523	0.531
15	0.594	0.339	0.604	0.417	0.304
16	0.780	0.804	0.461	0.858	0.592
17	0.793	0.590	0.349	0.750	0.536
18	0.632	0.933	0.502	0.972	0.369
19	0.159	0.908	0.508	0.911	0.404
20	0.126	0.860	0.476	0.953	0.423
21	0.410	0.779	0.411	0.878	0.504
22	0.222	0.895	0.496	1.000	0.345
23	0.410	0.671	0.371	0.721	0.753
24	0.529	0.061	0.009	0.017	0.138
25	0.173	1.000	0.501	0.963	0.633
26	0.516	0.824	0.458	0.883	0.276
27	0.316	0.649	0.467	0.759	0.534
28	0.457	0.688	0.411	0.709	0.300
29	0.492	0.394	0.314	0.447	0.494
30	0.583	0.017	0.198	0.119	0.239
31	0.537	0.616	0.732	0.621	0.342

Tabel 3.4 Data latih setelah normalisasi (lanjutan)

no.	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan
32	0.469	0.461	0.617	0.335	0.428
33	0.341	0.441	0.676	0.277	0.257
34	0.024	0.717	0.920	0.351	0.331
35	0.881	0.697	0.710	0.520	0.417
36	0.000	0.529	0.365	0.667	0.378
37	0.563	0.491	0.350	0.611	1.000
38	0.051	0.611	0.285	0.580	0.841
39	0.468	0.280	0.176	0.228	0.265
40	0.391	0.078	0.005	0.069	0.224
41	0.561	0.487	0.275	0.548	0.679
42	1.000	0.494	0.236	0.473	0.000
43	0.426	0.057	0.009	0.049	0.110
44	0.738	0.139	0.145	0.199	0.083
45	0.568	0.000	0.000	0.000	0.124
46	0.684	0.422	0.237	0.462	0.493
47	0.643	0.339	0.239	0.455	0.360
48	0.514	0.434	0.343	0.592	0.202
49	0.326	0.775	0.426	0.982	0.289
50	0.810	0.122	0.116	0.214	0.161
51	0.026	0.843	0.556	0.947	0.520
52	0.748	0.751	0.496	0.847	0.547
53	0.476	0.525	0.332	0.482	0.415
54	0.462	0.529	0.321	0.600	0.329
55	0.530	0.657	0.370	0.812	0.443
56	0.387	0.611	0.411	0.807	0.629
57	0.363	0.234	0.155	0.367	0.354
58	0.290	0.444	0.320	0.536	0.541
59	0.125	0.357	0.254	0.545	0.612
60	0.094	0.526	0.374	0.493	0.358

Tabel 3.5 Data uji setelah normalisasi

No	GLCM_pan	Mean_Green	Mean_Red	Mean_NIR	SD_pan
1	0.098	0.239	0.184	0.127	0.277

3.4.3 Menghitung Jarak Antara Record Baru Pada Data Uji Dengan Tiap Record Pada Data Latih

Dari 60 data latih dengan 1 data uji yang telah di normalisasi, dicari terlebih dahulu jarak terdekat "*Euclidean Distance*", dimana rumus untuk *Euclidean Distance* dapat dilihat pada persamaan 2.6 beserta contoh, perhitungannya.

$$d_i = \sqrt{\sum_{i=1}^{p}(x_{2i} - x_{1i})^2} \quad (2.6)$$

Keterangan:

x_1 = Data latih

x_2 = Data uji

i = Variabel data

d = Jarak

p = Dimensi data

$$d_i = \sqrt{\begin{pmatrix}((0.315 - 0.098)^2) + ((0.513 - 0.239)^2) + \\ ((0.654 - 0.184)^2) + \\ ((0.412 - 0.127)^2) + ((0.366 - 0.277)^2)\end{pmatrix}}$$

$$d_i = \sqrt{\begin{pmatrix}(0.217^2) + (0.274^2) + \\ (0.470^2) + (0.285^2) + (0.089^2)\end{pmatrix}}$$

$$d_i = \sqrt{(0.047089 + 0.075076 + 0.2209 + 0.081225 + 0.007921)}$$

$$d_i = \sqrt{(0.432211)} = 0.657$$

Kemudian dilakukan perhitungan yang sama untuk data yang lain. Hasil perhitungan jarak ditampilkan pada Tabel 3.6.

Tabel 3.6 Hasil *Euclidean Distance* antara 60 data latih dengan 1 data uji

no.	nilai target	kelas
1	0.657	1
2	0.599	1
3	0.813	1
4	0.402	1
5	0.831	1
6	1.131	1
7	0.563	1
8	1.171	1

Tabel 3.7 Hasil *Euclidean Distance* antara 60 data latih dengan 1 data uji (lanjutan)

no.	nilai target	kelas
9	1.226	1
10	0.982	1
11	0.789	1
12	0.768	1
13	1.083	1
14	0.853	1
15	0.720	1
16	1.222	2
17	1.043	2
18	1.261	2
19	1.089	2
20	1.083	2
21	1.027	2
22	1.144	2
23	0.948	2
24	0.529	2
25	1.229	2
26	1.078	2
27	0.872	2
28	0.849	2
29	0.588	2
30	0.536	2
31	0.940	1
32	0.664	1
33	0.603	1
34	0.910	1
35	1.128	1
36	0.654	2
37	1.032	2
38	0.821	2
39	0.386	2
40	0.387	2
41	0.789	2
42	1.038	2
43	0.453	2
44	0.682	2
45	0.593	2
46	0.734	2
47	0.652	2

Tabel 3.8 Hasil *Euclidean Distance* antara 60 data latih dengan 1 data uji (lanjutan)

no.	nilai target	kelas
48	0.677	2
49	1.062	2
50	0.739	2
51	1.113	2
52	1.172	2
53	0.626	2
54	0.680	2
55	0.945	2
56	0.927	2
57	0.367	2
58	0.578	2
59	0.553	2
60	0.509	2

Setelah mendapatkan nilai *Euclidean Distance* maka hal selanjutnya yang dilakukan adalah mengurutkan data tersebut mulai dari nilai target terkecil hingga nilai yang terbesar. Hasil perhitungan setelah diurutkan ditunjukkan pada Tabel 3.7.

Tabel 3.9 Data hasil *Euclidean Distance* diurutkan dari yang terkecil

no.	nilai target	kelas
57	0.367	2
39	0.386	2
40	0.387	2
4	0.402	1
43	0.453	2
60	0.509	2
24	0.529	2
30	0.536	2
59	0.553	2
7	0.563	1
58	0.578	2
29	0.588	2
45	0.593	2
2	0.599	1
33	0.603	1
53	0.626	2

Tabel 3.7 Data hasil *Euclidean Distance* diurutkan dari yang terkecil (lanjutan)

no.	nilai target	kelas
47	0.652	2
36	0.654	2
1	0.657	1
32	0.664	1
48	0.677	2
54	0.680	2
44	0.682	2
15	0.720	1
46	0.734	2
50	0.739	2
12	0.768	1
11	0.789	1
41	0.789	2
3	0.813	1
38	0.821	2
5	0.831	1
28	0.849	2
14	0.853	1
27	0.872	2
34	0.910	1
56	0.927	2
31	0.940	1
55	0.945	2
23	0.948	2
10	0.982	1
21	1.027	2
37	1.032	2
42	1.038	2
17	1.043	2
49	1.062	2
26	1.078	2
20	1.083	2
13	1.083	1
19	1.089	2
51	1.113	2
35	1.128	1
6	1.131	1
22	1.144	2
8	1.171	1
52	1.172	2

Tabel 3.7 Data hasil *Euclidean Distance* diurutkan dari yang terkecil (lanjutan)

no.	nilai target	kelas
16	1.222	2
9	1.226	1
25	1.229	2
18	1.261	2

3.4.4 Menentukan k-Record Terdekat

Setelah melakukan perhitungan *Euclidean Distance* dan dilakukan pengurutan mulai nilai dari yang terkecil sampai terbesar yang sesuai hasil yang ditunjukkan pada Tabel 3.6 dan Tabel 3.7 kemudian diambil k record dengan nilai terkecil. Karena k = 3, maka record yang terpilih adalah record ke-40, 43 dan ke-24 yang di tunjukkan pada Tabel 3.8.

Tabel 3.10 Data hasil *Euclidean Distance* di ambil sesuai jumlah nilai k

no.	nilai target	kelas
57	0.367	2
39	0.386	2
40	0.387	2

3.4.5 Menentukan Maksimum Membership dan Kelas Target

Untuk menentukan maksimum membership dan kelas target untuk tiap kelas j adalah dengan menggunakan persamaan 2.7. Dengan K=60 (jumlah data latih), u1 = *wilt disease*, dan u2 = lahan cover lain.

$u_1(1)$ = 0.51+(20/60)*0.49

= 0.673

$u_1(2)$ = (20/60)*0.49

= 0.163

$u_2(2)$ = 0.51+(40/60)*0.49

= 0.837

$u_2(1)$ = (40/60)*0.49

= 0.327

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

$$u_1 = \frac{\left(0.782 * (0.662^{-\frac{2}{2-1}})\right) + \left(0.782 * (0.757^{-\frac{2}{2-1}})\right) + \left(0.782 * (0.858^{-\frac{2}{2-1}})\right)}{(0.662^{-\frac{2}{2-1}}) + (0.757^{-\frac{2}{2-1}}) + (0.858^{-\frac{2}{2-1}})}$$

$$= \frac{4.210}{5.382}$$

$$= 0.782$$

$$u_2 = \frac{\left(0.218 * (0.662^{-\frac{2}{2-1}})\right) + \left(0.218 * (0.757^{-\frac{2}{2-1}})\right) + \left(0.218 * (0.858^{-\frac{2}{2-1}})\right)}{(0.662^{-\frac{2}{2-1}}) + (0.757^{-\frac{2}{2-1}}) + (0.858^{-\frac{2}{2-1}})}$$

$$= \frac{1.172}{5.382}$$

$$= 0.218$$

Berdasarkan hasil perhitungan nilai keanggotaan didapat dua nilai keanggotaan, untuk menentukan kelas target maka dipilih nilai keanggotaan terbesar yaitu 0.782 sehingga kelas targetnya yaitu w (pepohonan yang sakit).

Setelah melakukan pengujian data uji langkah yang sama dilakukan terhadap data uji yang lain sehingga didapat akurasi sistem yang dihitung berdasarkan persamaan (2.5) yang ditunjukkan pada Tabel 3.9.

Tabel 3.11 Perhitungan akurasi sistem

Data Uji Ke-	Kelas Data	Kelas Hasil Perhitungan	Hasil Prediksi
1	1	1	Benar
2	2	1	Salah
3	1	1	Benar

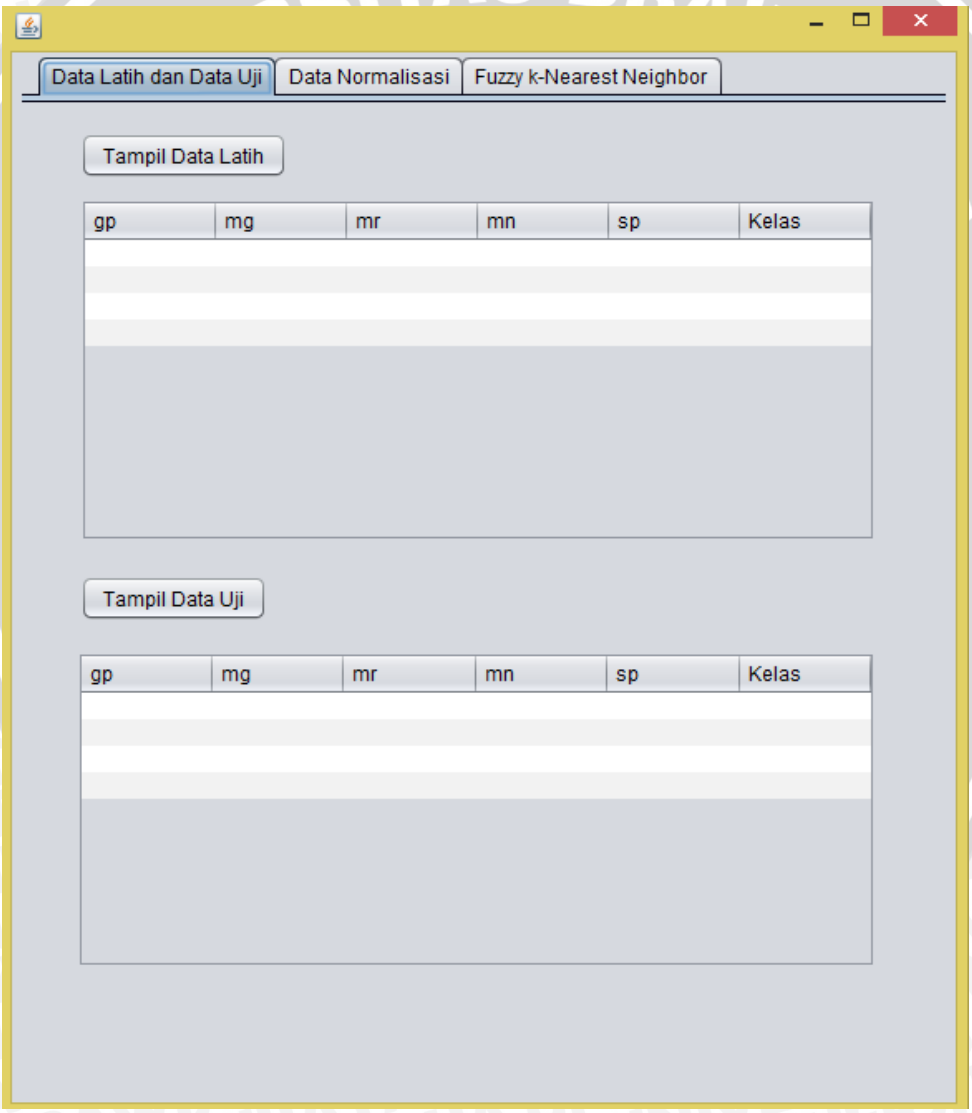
$$Akurasi\ (\%) = \frac{2}{3}\ X\ 100\%$$

$$= 66{,}66667\ \%$$

BAB 4 PERANCANGAN

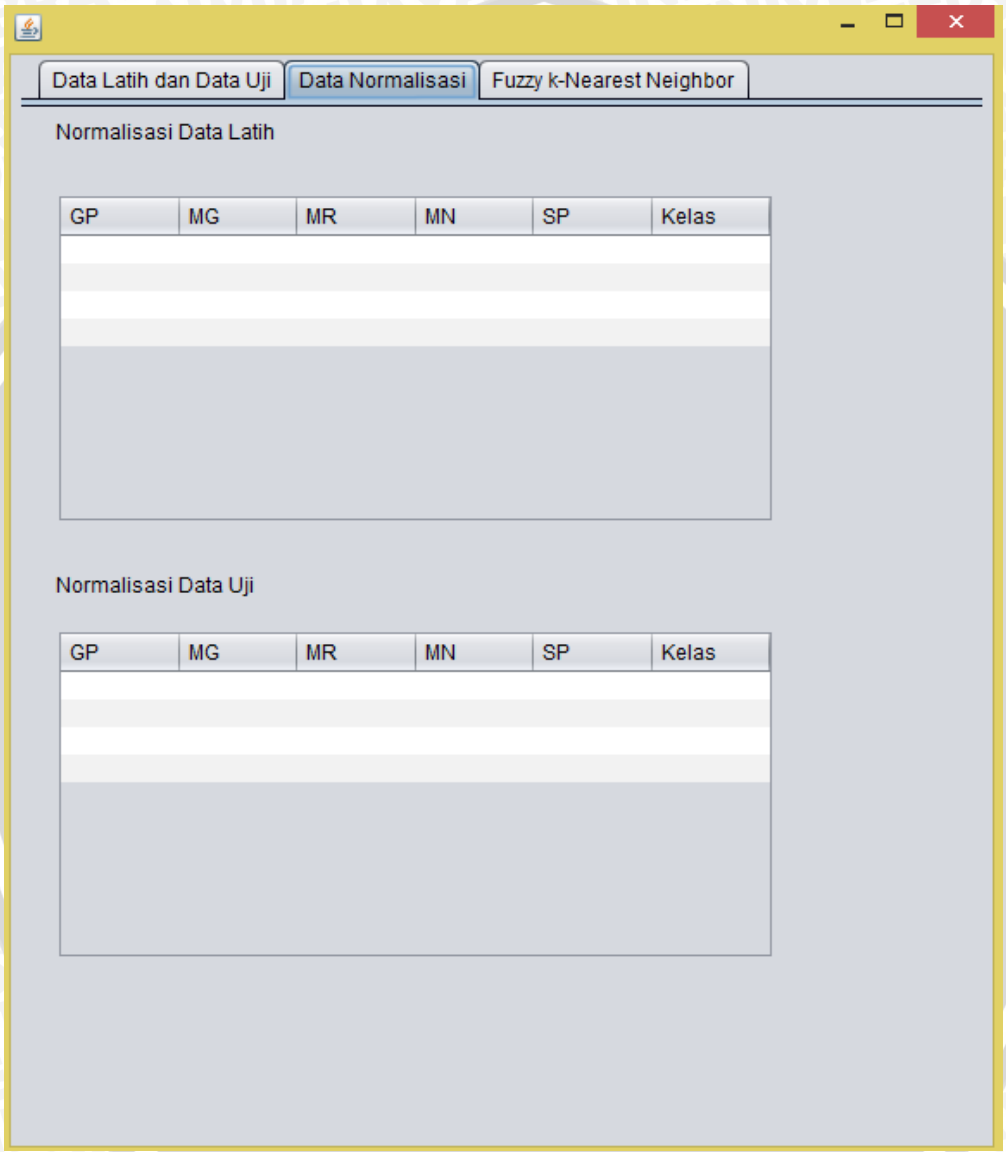
4.1 Perancangan Antarmuka

Antarmuka (interface) untuk menampilkan data yang digunakan sistem, terdiri dari 3 bagian, yaitu Tampil Data Latih untuk memanggil dan menampilkan data latih, Tampil Data Uji untuk memanggil dan menampilkan, dan *Tab Fuzzy k-Nearest Neighbor* untuk menampilkan nilai k yang dimasukkan beserta hasil akurasi. Berikut penjelasan label pada tabel antara lain : gp adalah *Grey Level Co-Occurrence Matrix Pan*, mg adalah *mean green*, mr adalah *mean red*, mn adalah *mean NIR* dan sp adalah *standard deviation Pan.* Perancangan antarmuka sistem akan ditunjukkan pada Gambar 4.1.



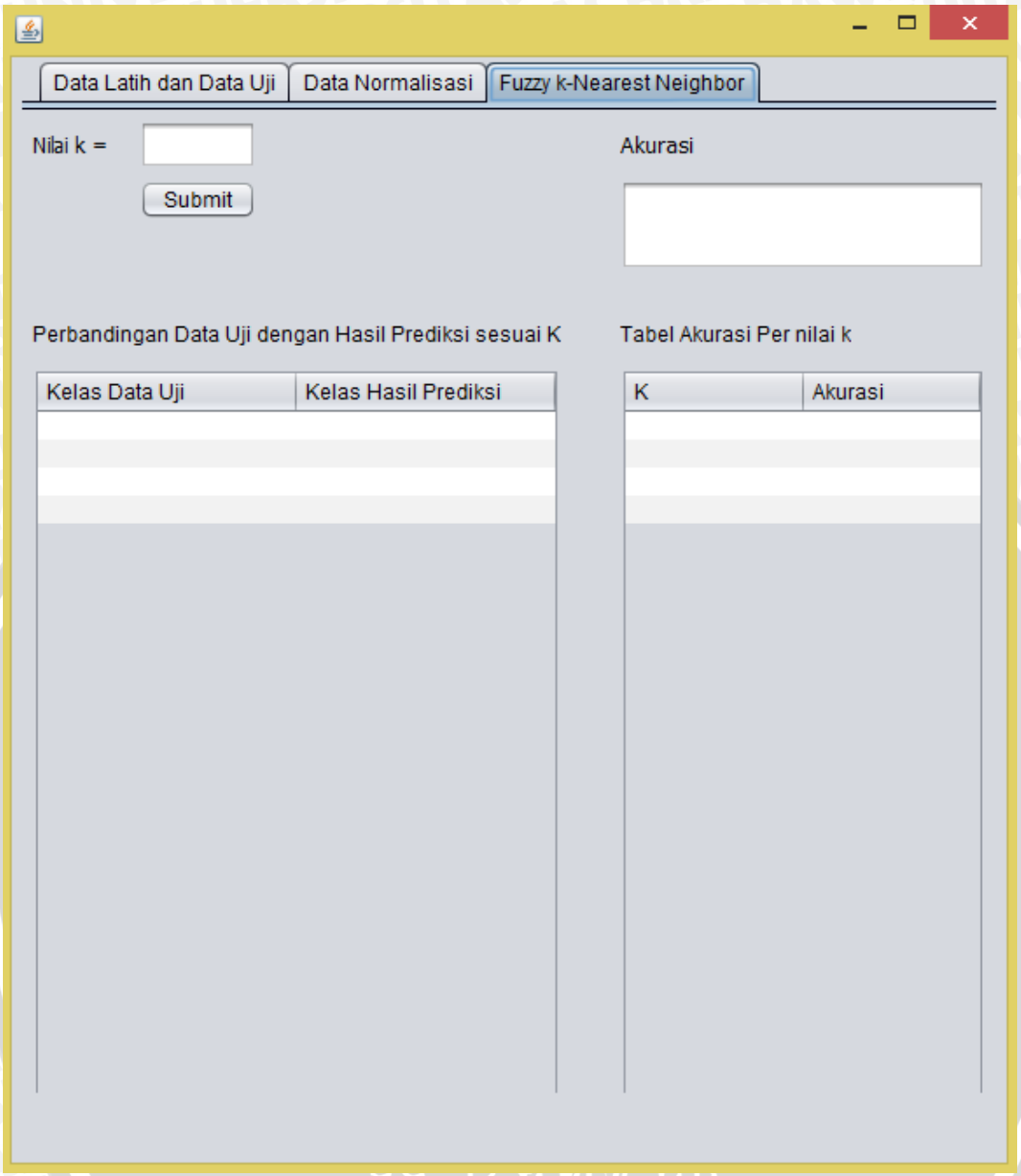
Gambar 4.1 Tampilan antarmuka

1. Tombol Tampil Data Latih digunakan untuk memanggil data latih dan tabel dibawah nya berfungsi menampilkan data latih dari *wilt dataset* yang akan digunakan sistem.
2. Tombol Tampil Data Uji digunakan untuk memanggil data uji dan tabel dibawah nya berfungsi menampilkan data uji dari *wilt dataset* yang akan digunakan untuk klasifikasi.



Gambar 4.2 Tampilan tabel data yang telah dinormalisasi

1. Tab data normalisasi berisi tabel data latih dan data uji yang telah dinormalisasi agar dapat digunakan dalam pengujian.
2. Tabel pertama menunjukkan data latih yang telah melalui perhitungan normalisasi.
3. Tabel dibawah nya menunjukkan data uji yang telah melalui perhitungan normalisasi.



Gambar 4.3 Tampilan input untuk nilai K dan tingkat akurasi sistem

1. Textfield sebagai input nilai untuk menentukan nilai k yang digunakan dalam pengujian.
2. Tombol submit berfungsi menjalankan proses perhitungan *Fuzzy k-Nearest Neighbor*.

4.2 Perancangan Uji Coba

Setelah metode *Fuzzy k-Nearest Neighbor* diimplementasikan ke dalam sistem, langkah selanjutnya adalah melakukan pengujian terhadap sistem. Pengujian didalam penelitian ini dilakukan untuk mengetahui tingkat akurasi dari hasil klasifikasi pada *Wilt Dataset* dengan menggunakan metode *Fuzzy k-Nearest Neighbor*.

Pengujian yang dilakukan dalam penelitian ini adalah untuk mengetahui pengaruh nilai k (jumlah tetangga terdekat) dan jumlah data latih terhadap tingkat akurasi klasifikasi dengan menggunakan metode *Fuzzy k-Nearest Neighbor*.

4.3 Pengujian Pengaruh Nilai k dan Jumlah Data Latih Terhadap Tingkat Akurasi Klasifikasi

Uji pengaruh nilai k dan jumlah data latih terhadap tingkat akurasi klasifikasi dilakukan dengan menggunakan beberapa jumlah data latih yang berbeda pada sejumlah data uji yaitu sebanyak 30 data uji. Dengan beberapa jumlah data latih yang berbeda-beda, diharapkan akan mempengaruhi keakuratan hasil keputusan yang berpengaruh terhadap klasifikasi. Dalam pengujian ini ditentukan nilai k = 2…n. Dari data latih diambil sejumlah data, yaitu sebanyak 60, 80, 120 200, dan 350 data.

Dari pengujian ini, diperoleh tingkat akurasi terhadap nilai k dan jumlah data latih. Tabel 4.1 menampilkan rancangan tabel yang akan digunakan untuk mencatat hasil dari pengujian ini.

Tabel 4.1 Uji pengaruh nilai k dan data latih terhadap tingkat akurasi

K	Jumlah Data Latih	Akurasi (%)
2	60	
…	…	
N	n	

Keterangan:

k : nilai k yang akan diuji.

Jumlah Data Latih : jumlah data latih yang digunakan dalam pengujian.

Akurasi : tingkat akurasi klasifikasi dengan menggunakan metode *Fuzzy k-Nearest Neighbor* dalam bentuk persen.

BAB 5 IMPLEMENTASI DAN PEMBAHASAN

5.1 Lingkungan Implementasi

Dalam implementasi metode *Fuzzy K-Nearest Neighbor* (FKNN) untuk diagnosa penyakit *wilt* pada hutan dibutuhkan beberapa aspek yang perlu diperhatikan yaitu segi perangkat keras dan perangkat lunak.

5.1.1 Lingkungan Implementasi Perangkat Keras

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

Processor	: Intel ® Core i3-4005u 1,7ghz
Memory	: 6 GB
GPU	: NVIDIA GEFORCE 930m 2 GB
Hard disk	: 500 GB

5.1.2 Lingkungan Implementasi Perangkat Lunak

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

1. Sistem Operasi yang digunakan Windows 8.1 64bit.
2. Aplikasi pembangunan GUI dan code menggunakan Netbeans.
3. Bahasa pemrograman yang dipakai yaitu bahasa pemrograman java.

5.2 Batasan Implementasi

Berikut batasan implementasi pada sistem diagnosa penyakit layu (*wilt disease)* pada hutan :

1. Sistem dibangun berbasis desktop dengan menggunakan bahasa pemrograman *Java*.
2. Metode yang digunakan untuk mendiagnosis *wilt disease* menggunakan metode *Fuzzy K-Nearest Neighbor* (FKNN).
3. Data input sistem menggunakan format .xls.
4. Input dalam sistem ini berdasarkan *Wilt Dataset* yang diambil dari website UCI.
5. Sistem ini mendiagnosa adanya *wilt disease* berdasarkan *wilt dataset.*

5.3 Implementasi Program

Pada subbab berikut dijelaskan proses-proses klasifikasi pada aplikasi.

5.3.1 Proses Baca File

Proses ini berfungsi untuk untuk membaca file yang berisi data latih atau uji yang akan digunakan dalam perhitungan.

```
1   package fknn_adi;
2
3   import java.io.File;
4   import java.util.ArrayList;
5   import java.util.List;
6   import jxl.Sheet;
7   import jxl.Workbook;
8
9   /**
10   *
11   * @author ANGINSETO
12   */
13  public class ReadData {
14
15      private String path;
        private List<AtributData> lstAtributData = new
16  ArrayList<AtributData>();
17      private AtributData atributData;
18
19      public ReadData (String path){ //inisialisasi location
20
21          this.path = path;
22
23      }
24
25      public List<AtributData> bacaData(){
26
27          File file = new File(path); //browse path file
28
29        try{
30
            Workbook workbook = jxl.Workbook.getWorkbook(file);
31  //get excel 97
32          Sheet[] sheets = workbook.getSheets();
33          int row = sheets[0].getRows();
34          int column = sheets[0].getColumns();
35
36
37
38          for (int i=1 ; i<row ; i++){
39
40              atributData = new AtributData();

    atributData.setGp(Double.parseDouble(sheets[0].getCell(0,
41  i).getContents()));
```

```
    atributData.setMg(Double.parseDouble(sheets[0].getCell(1,
42  i).getContents()));

    atributData.setMr(Double.parseDouble(sheets[0].getCell(2,
43  i).getContents()));

    atributData.setMn(Double.parseDouble(sheets[0].getCell(3,
44  i).getContents()));

    atributData.setSp(Double.parseDouble(sheets[0].getCell(4,
45  i).getContents()));
46

    atributData.setKelas(Integer.parseInt(sheets[0].getCell(5,
47  i).getContents()));
48
49              lstAtributData.add(atributData);
50          }
51      }
52      catch (Exception ex)
53      {
54          System.out.println("Gagal"+ ex.getMessage());
55      }
56
57      return lstAtributData;
58
59    }
60  }
```

Source Code 5.1 Baca file

5.3.2 Proses Tampil Data Latih Dan Data Uji

Proses ini menampilkan data latih dan data uji untuk sistem yang telah di input pada tabel antarmuka aplikasi.

```
     private void
    btnTampilDataLatihActionPerformed(java.awt.event.ActionEvent
1   evt) {
2           // TODO add your handling code here:
3           BrowseData browseData = new BrowseData();
4
5           browseData.choose();
6           String pathDataLatih = browseData.getPath();
7
8           ReadData readData = new ReadData(pathDataLatih);
9           lstAtributDataLatih = readData.bacaData();
```

```
10
11          /*
12          for (int i=0 ; i<lstAtributDataLatih.size() ; i++){
13
                System.out.println
    (lstAtributDataLatih.get(i).getPi() + " " +
    lstAtributDataLatih.get(i).getPt() + " " +
    lstAtributDataLatih.get(i).getLla() + " " +
    lstAtributDataLatih.get(i).getSs() + " " +
    lstAtributDataLatih.get(i).getPr() + " " +
    lstAtributDataLatih.get(i).getDs() + " " +
14  lstAtributDataLatih.get(i).getKelas());
15
16          } */
17
18          namaKolomDataLatih = new String [6];
19
20          namaKolomDataLatih[0] = "gp";
21          namaKolomDataLatih[1] = "mg";
22          namaKolomDataLatih[2] = "mr";
23          namaKolomDataLatih[3] = "mn";
24          namaKolomDataLatih[4] = "sp";
25
26          namaKolomDataLatih[5] = "Kelas";
27
            isiTabelDataLatih = new
28  Object[lstAtributDataLatih.size()][6];
29
30          for (int i=0 ; i<lstAtributDataLatih.size();i++){
31
                isiTabelDataLatih
32  [i][0]=lstAtributDataLatih.get(i).getGp();
                isiTabelDataLatih
33  [i][1]=lstAtributDataLatih.get(i).getMg();
                isiTabelDataLatih
34  [i][2]=lstAtributDataLatih.get(i).getMr();
                isiTabelDataLatih
35  [i][3]=lstAtributDataLatih.get(i).getMn();
                isiTabelDataLatih
36  [i][4]=lstAtributDataLatih.get(i).getSp();
37
                isiTabelDataLatih
38  [i][5]=lstAtributDataLatih.get(i).getKelas();
39          }
40
            tblDataLatih.setModel(new
41  DefaultTableModel(isiTabelDataLatih, namaKolomDataLatih));
42
43      }
```

```
44
      private void
    btnTampilDataUjiActionPerformed(java.awt.event.ActionEvent evt)
45  {
46          // TODO add your handling code here:
47          BrowseData browseData = new BrowseData();
48
49          browseData.choose();
50          String pathDataUji = browseData.getPath();
51          ReadData readData = new ReadData(pathDataUji);
52          lstAtributDataUji = readData.bacaData();
53
54          namaKolomDataUji = new String [6];
55
56          namaKolomDataUji[0] = "gp";
57          namaKolomDataUji[1] = "mg";
58          namaKolomDataUji[2] = "mr";
59          namaKolomDataUji[3] = "mn";
60          namaKolomDataUji[4] = "sp";
61
62          namaKolomDataUji[5] = "Kelas";
63
            isiTabelDataUji = new
64  Object[lstAtributDataUji.size()][6];
65
66          for (int i=0 ; i<lstAtributDataUji.size();i++){
67
                isiTabelDataUji
68  [i][0]=lstAtributDataUji.get(i).getGp();
                isiTabelDataUji
69  [i][1]=lstAtributDataUji.get(i).getMg();
                isiTabelDataUji
70  [i][2]=lstAtributDataUji.get(i).getMr();
                isiTabelDataUji
71  [i][3]=lstAtributDataUji.get(i).getMn();
                isiTabelDataUji
72  [i][4]=lstAtributDataUji.get(i).getSp();
73
                isiTabelDataUji
74  [i][5]=lstAtributDataUji.get(i).getKelas();
75          }
76
            tblDataUji.setModel(new
77  DefaultTableModel(isiTabelDataUji, namaKolomDataUji));
78
79          //untuk menampilkan normalisasi
            Normalisasi normalisasi = new
80  Normalisasi(lstAtributDataLatih, lstAtributDataUji);
```

```
81          normalisasi.minMax();
82          normalisasi.proses();
83
            lstAtributDataLatihNormal =
84  normalisasi.getLstAtributDataLatihNormal();
            lstAtributDataUjiNormal =
85  normalisasi.getLstAtributDataUjiNormal();
86
87          namaKolomDataLatihNormal = new String [6];
88
89          namaKolomDataLatihNormal[0] = "gp";
90          namaKolomDataLatihNormal[1] = "mg";
91          namaKolomDataLatihNormal[2] = "mr";
92          namaKolomDataLatihNormal[3] = "mn";
93          namaKolomDataLatihNormal[4] = "sp";
94
95          namaKolomDataLatihNormal[5] = "Kelas";
96
            isiTabelDataLatihNormal = new
97  Object[lstAtributDataLatihNormal.size()][6];
98
99          for (int i=0 ; i<lstAtributDataLatihNormal.size();i++){
100
                isiTabelDataLatihNormal
101 [i][0]=lstAtributDataLatihNormal.get(i).getGp();
                isiTabelDataLatihNormal
102 [i][1]=lstAtributDataLatihNormal.get(i).getMg();
                isiTabelDataLatihNormal
103 [i][2]=lstAtributDataLatihNormal.get(i).getMr();
                isiTabelDataLatihNormal
104 [i][3]=lstAtributDataLatihNormal.get(i).getMn();
                isiTabelDataLatihNormal
105 [i][4]=lstAtributDataLatihNormal.get(i).getSp();
106
                isiTabelDataLatihNormal
107 [i][5]=lstAtributDataLatih.get(i).getKelas();
108         }
             tblNormalLatih.setModel(new
    DefaultTableModel(isiTabelDataLatihNormal,
109 namaKolomDataLatihNormal));
110
111         namaKolomDataUjiNormal = new String [6];
112
113         namaKolomDataUjiNormal[0] = "gp";
114         namaKolomDataUjiNormal[1] = "mg";
115         namaKolomDataUjiNormal[2] = "mr";
116         namaKolomDataUjiNormal[3] = "mn";
117         namaKolomDataUjiNormal[4] = "sp";
```

```
118
119         namaKolomDataUjiNormal[5] = "Kelas";
120
            isiTabelDataUjiNormal = new
121 Object[lstAtributDataUjiNormal.size()][6];
122
123         for (int i=0 ; i<lstAtributDataUjiNormal.size();i++){
124
                isiTabelDataUjiNormal
125 [i][0]=lstAtributDataUjiNormal.get(i).getGp();
                isiTabelDataUjiNormal
126 [i][1]=lstAtributDataUjiNormal.get(i).getMg();
                isiTabelDataUjiNormal
127 [i][2]=lstAtributDataUjiNormal.get(i).getMr();
                isiTabelDataUjiNormal
128 [i][3]=lstAtributDataUjiNormal.get(i).getMn();
                isiTabelDataUjiNormal
129 [i][4]=lstAtributDataUjiNormal.get(i).getSp();
130
                isiTabelDataUjiNormal
131 [i][5]=lstAtributDataUji.get(i).getKelas();
132         }
133
            tblNormalUji.setModel(new
    DefaultTableModel(isiTabelDataUjiNormal,
134 namaKolomDataUjiNormal));
135
136         //for ecludian distance
137
138
139     }
```

Source Code 5.2 Tampil data latih dan data uji

5.3.3 Proses Normalisasi Data Latih Dan Data Uji

Proses ini menjalankan proses normalisasi *min-max* pada data latih dan data uji yang diinputkan sebelumnya.

```
1   package fknn_adi;
2
3   import java.util.ArrayList;
4   import java.util.List;
5
6   /**
7    *
8    * @author ANGINSETO
9    */
```

```
10  public class Normalisasi {
11
12      private List<AtributData> lstAtributDataLatih;
13      private List<AtributData> lstAtributDataUji;
14      private List<AtributData> lstAtributDataLatihNormal;
15      private List<AtributData> lstAtributDataUjiNormal;
16      private List<AtributData> mixLatihUji;
17      private double minGp;
18      private double minMg;
19      private double minMr;
20      private double minMn;
21      private double minSp;
22
23      private double maxGp;
24      private double maxMg;
25      private double maxMr;
26      private double maxMn;
27      private double maxSp;
28
29
30
31
32
33
34      private double maxGpUji;
35      private double maxMgUji;
36      private double maxMrUji;
37      private double maxMnUji;
38      private double maxSpUji;
39
40      private double minGpUji;
41      private double minMgUji;
42      private double minMrUji;
43      private double minMnUji;
44      private double minSpUji;
45
46
        public Normalisasi (List<AtributData> atributDataLatih,
47  List<AtributData> atributDataUji){
48          this.lstAtributDataLatih = atributDataLatih;
49          this.lstAtributDataUji = atributDataUji;
50
51          mixLatihUji = new ArrayList<AtributData>();
52
```

```
53          for(AtributData object: lstAtributDataLatih)
54          {
55              mixLatihUji.add(object);
56          }
57
58          for(AtributData objek : lstAtributDataUji)
59          {
60              mixLatihUji.add(objek);
61          }
62      }
63
64      public void minMax (){
65
66          maxGp = 0;
67          maxMg = 0;
68          maxMr = 0;
69          maxMn = 0;
70          maxSp = 0;
71
72          maxGpUji = 0;
73          maxMgUji = 0;
74          maxMrUji = 0;
75          maxMnUji = 0;
76          maxSpUji = 0;
77
78
79          minGp = lstAtributDataLatih.get(0).getGp();
80          minMg = lstAtributDataLatih.get(0).getMg();
81          minMr = lstAtributDataLatih.get(0).getMr();
82          minMn = lstAtributDataLatih.get(0).getMn();
83          minSp = lstAtributDataLatih.get(0).getSp();
84
85          minGpUji = lstAtributDataLatih.get(0).getGp();
86          minMgUji = lstAtributDataLatih.get(0).getMg();
87          minMrUji = lstAtributDataLatih.get(0).getMr();
88          minMnUji = lstAtributDataLatih.get(0).getMn();
89          minSpUji = lstAtributDataLatih.get(0).getSp();
90
91
92          for(int i=0; i<lstAtributDataLatih.size(); i++)
93          {
94              //untuk maximum
95              if(lstAtributDataLatih.get(i).getGp() > maxGp)
```

```
96              {
97                  maxGp = lstAtributDataLatih.get(i).getGp();
98              }
99
100              if(lstAtributDataLatih.get(i).getMg() > maxMg)
101             {
102                 maxMg = lstAtributDataLatih.get(i).getMg();
103             }
104
105              if(lstAtributDataLatih.get(i).getMr() > maxMr)
106             {
107                 maxMr = lstAtributDataLatih.get(i).getMr();
108             }
109
110              if(lstAtributDataLatih.get(i).getMn() > maxMn)
111             {
112                 maxMn = lstAtributDataLatih.get(i).getMn();
113             }
114
115              if(lstAtributDataLatih.get(i).getSp() > maxSp)
116             {
117                 maxSp = lstAtributDataLatih.get(i).getSp();
118             }
119
120
121
122
123
124
125
126
127             //untuk minimum
128             if(minGp > lstAtributDataLatih.get(i).getGp())
129             {
130                 minGp = lstAtributDataLatih.get(i).getGp();
131             }
132
133             if(minMg > lstAtributDataLatih.get(i).getMg())
134             {
135                 minMg = lstAtributDataLatih.get(i).getMg();
136             }
137
138             if(minMr > lstAtributDataLatih.get(i).getMr())
```

```
139             {
140                 minMr = lstAtributDataLatih.get(i).getMr();
141             }
142 
143             if(minMn > lstAtributDataLatih.get(i).getMn())
144             {
145                 minMn = lstAtributDataLatih.get(i).getMn();
146             }
147 
148             if(minSp > lstAtributDataLatih.get(i).getSp())
149             {
150                 minSp= lstAtributDataLatih.get(i).getSp();
151             }
152 
153 
154 
155 
156 
157         }
158 
159         for(int i=0; i<lstAtributDataUji.size(); i++)
160         {
161             //max Uji
162              if(lstAtributDataUji.get(i).getGp() > maxGpUji)
163             {
164                 maxGpUji = lstAtributDataUji.get(i).getGp();
165             }
166 
167              if(lstAtributDataUji.get(i).getMg() > maxMgUji)
168             {
169                 maxMgUji = lstAtributDataUji.get(i).getMg();
170             }
171 
172              if(lstAtributDataUji.get(i).getMr() > maxMrUji)
173             {
174                 maxMrUji = lstAtributDataUji.get(i).getMr();
175             }
176 
177              if(lstAtributDataUji.get(i).getMn() > maxMnUji)
178             {
179                 maxMnUji = lstAtributDataUji.get(i).getMn();
180             }
181 
```

```
182             if(lstAtributDataUji.get(i).getSp() > maxSpUji)
183             {
184                 maxSpUji = lstAtributDataUji.get(i).getSp();
185             }
186 
187 
188              //min uji
189             if(minGpUji > lstAtributDataUji.get(i).getGp())
190             {
191                 minGpUji = lstAtributDataUji.get(i).getGp();
192             }
193 
194             if(minMgUji > lstAtributDataUji.get(i).getMg())
195             {
196                 minMgUji = lstAtributDataUji.get(i).getMg();
197             }
198 
199             if(minMrUji > lstAtributDataUji.get(i).getMr())
200             {
201                 minMrUji = lstAtributDataUji.get(i).getMr();
202             }
203 
204             if(minMnUji > lstAtributDataUji.get(i).getMn())
205             {
206                 minMnUji = lstAtributDataUji.get(i).getMn();
207             }
208 
209             if(minSpUji > lstAtributDataUji.get(i).getSp())
210             {
211                 minSpUji = lstAtributDataUji.get(i).getSp();
212             }
213 
214         }
215 
216     }
217 
218     public void proses(){
        lstAtributDataLatihNormal = new
219 ArrayList<AtributData>();
220         lstAtributDataUjiNormal = new ArrayList<AtributData>();
221 
222         //normalisasi data latih
223         for(int i=0 ; i<lstAtributDataLatih.size() ; i++)
224         {
```

```
225             AtributData atributData = new AtributData();
226 
    atributData.setGp(((lstAtributDataLatih.get(i).getGp()) -
227 minGp) / (maxGp - minGp));
    atributData.setMg(((lstAtributDataLatih.get(i).getMg()) -
228 minMg) / (maxMg - minMg));
    atributData.setMr(((lstAtributDataLatih.get(i).getMr()) -
229 minMr) / (maxMr - minMr));
    atributData.setMn(((lstAtributDataLatih.get(i).getMn()) -
230 minMn) / (maxMn - minMn));
    atributData.setSp(((lstAtributDataLatih.get(i).getSp()) -
231 minSp) / (maxSp - minSp));
232 
233 
234             lstAtributDataLatihNormal.add(atributData);
235         }
236 
237         // normalisasi data uji
238 
239         for(int i=0 ; i<lstAtributDataUji.size() ; i++)
240         {
241             AtributData atributData = new AtributData();
242 
    atributData.setGp(((lstAtributDataUji.get(i).getGp()) -
243 minGpUji) / (maxGpUji - minGpUji));
    atributData.setMg(((lstAtributDataUji.get(i).getMg()) -
244 minMgUji) / (maxMgUji - minMgUji));
    atributData.setMr(((lstAtributDataUji.get(i).getMr()) -
245 minMrUji) / (maxMrUji - minMrUji));
    atributData.setMn(((lstAtributDataUji.get(i).getMn()) -
246 minMnUji) / (maxMnUji - minMnUji));
    atributData.setSp(((lstAtributDataUji.get(i).getSp()) -
247 minSpUji) / (maxSpUji - minSpUji));
248 
249 
250             lstAtributDataUjiNormal.add(atributData);
251         }
252 
253     }
254 
```

```
255     public List<AtributData> getLstAtributDataLatihNormal() {
256         return lstAtributDataLatihNormal;
257     }
258 
259     public List<AtributData> getLstAtributDataUjiNormal() {
260         return lstAtributDataUjiNormal;
261     }
262 
263 }
```

Source Code 5.3 Proses normalisasi data latih dan data uji

5.3.4 Proses metode algoritma *Fuzzy k-Nearest Neighbor*

Proses ini adalah untuk melakukan perhitungan implementasi algoritma *Fuzzy k-Nearest Neighbor* data pada data yang telah di normalisasi.

```
1   package fknn_adi;
2   
3   import java.util.ArrayList;
4   import java.util.Collections;
5   import java.util.List;
6   
7   /**
8    *
9    * @author ANGINSETO
10   */
11  public class Knn {
12  
13      private List<AtributData> lstAtributDataLatih;
14      private List<AtributData> lstAtributDataLatihNormal;
15      private List<AtributData> lstAtributDataUjiNormal;
16      private List<EuclideanData> lstEuclideanDatas;
17      private List<Integer> hasilPrediksi;
18      int K;
19      double n1;
20      double n2;
21      double u11;
22      double u12;
23      double u21;
24      double u22;
25      double [] pembilang;
26      double [] penyebut;
27      double [] hasil;
28  
```

```
        public Knn (List<AtributData> lstAtributDataLatihNormal,
29  List<AtributData> lstAtributDataLatihUjiNormal){
            this.lstAtributDataLatihNormal =
30  lstAtributDataLatihNormal;
            this.lstAtributDataUjiNormal =
31  lstAtributDataLatihUjiNormal;
32      }
33
34      public void process (){
35
36          //cari euclidean
37
38          hasilPrediksi = new ArrayList<Integer>();
            for (int j = 0; j<lstAtributDataUjiNormal.size(); j++){
39  //ulang data uji
40              //System.out.println("data ke " + j );
41              lstEuclideanDatas = new ArrayList<EuclideanData>();
                for (int i = 0; i<lstAtributDataLatihNormal.size();
42  i++){ //ulang data latih
                    EuclideanData euclideanData = new
43  EuclideanData();
44

    euclideanData.setNilaiTarget((Math.pow((lstAtributDataUjiNormal
45  .get(j).getGp()-lstAtributDataLatihNormal.get(i).getGp()), 2))
                    +
    (Math.pow((lstAtributDataUjiNormal.get(j).getMg() -
46  lstAtributDataLatihNormal.get(i).getMg()), 2))
                    +
    (Math.pow((lstAtributDataUjiNormal.get(j).getMr() -
47  lstAtributDataLatihNormal.get(i).getMr()), 2))
                    +
    (Math.pow((lstAtributDataUjiNormal.get(j).getMn() -
48  lstAtributDataLatihNormal.get(i).getMn()), 2))
                    +
    (Math.pow((lstAtributDataUjiNormal.get(j).getSp() -
49  lstAtributDataLatihNormal.get(i).getSp()), 2)));
50
51

    euclideanData.setNilaiTarget(Math.sqrt(euclideanData.getNilaiTa
52  rget()));

53  euclideanData.setKelas(lstAtributDataLatih.get(i).getKelas());
54
                    lstEuclideanDatas.add(euclideanData); //
55  ecludian sebelum di sort
56
57              }
    //              System.out.println("  ----------------- data ke
58  "+j);
```

```
59
                //System.out.println(" hasil eucledian data ke " +
60  j + "---------------------- sebelum disorting");
61              for(EuclideanData a : lstEuclideanDatas)
62              {
                    //System.out.println(a.getNilaiTarget() +" kelas
63  " +a.getKelas());
64              }
                Collections.sort(lstEuclideanDatas, new
    TargetComparator()); // sorting berdasarkan nilai target
65  terkecil
66
                //System.out.println(" hasil eucledian data ke " + j
67  + "---------------------- setelah disorting");
68              for(EuclideanData a : lstEuclideanDatas)
69              {
                    // System.out.println(a.getNilaiTarget() +"
70  kelas " +a.getKelas());
71              }
72
73              //membership
74              //cari total kelas 1 dan kelas 2
75              n1 = 0;
76              n2 = 0;
77
78              for(AtributData object : lstAtributDataLatih)
79              {
80
81                  if(object.getKelas() == 1)
82                  {
83                      n1++;
84                  }
85                  else
86                  {
87                      n2++;
88                  }
89              }
90
91              //mencari u1,1-u2,2
92              // System.out.println(lstEuclideanDatas.size());
93
94              u11= (0.51+((n1/lstEuclideanDatas.size())*0.49));
95              u12= ((n1/lstEuclideanDatas.size())*0.49);
96              u22= (0.51+((n2/lstEuclideanDatas.size())*0.49));
97              u21= ((n2/lstEuclideanDatas.size())*0.49);
98
```

```
99
100             for(int k=0; k<lstEuclideanDatas.size(); k++)
101             {
102                 if(lstEuclideanDatas.get(k).getKelas() == 1)
103                 {
104                     lstEuclideanDatas.get(k).setN1(u11);
105                     lstEuclideanDatas.get(k).setN2(u21);
106                 }
107                 else
108                 {
109                     lstEuclideanDatas.get(k).setN1(u12);
110                     lstEuclideanDatas.get(k).setN2(u22);
111                 }
112             }
113
114             //System.out.println(" hasil eucledian data ke " + j
    + "----------------------");
115             for(EuclideanData a : lstEuclideanDatas)
116             {
117                // System.out.println(a.getNilaiTarget() + " dan
    N1 " + a.getN1() + " dan N2 "+ a.getN2());
118             }
119
120             // cari pembilang dan penyebut
121             pembilang = new double[2];
122             pembilang[0] = 0;
123             pembilang[1] = 0;
124             penyebut = new double[2];
125             penyebut[0] = 0;
126             penyebut[1] = 0;
127             hasil = new double[2];
128
129
130             for(int p=0; p<K; p++)
131             {
132                 pembilang[0] = pembilang[0] +
    (lstEuclideanDatas.get(p).getN1() *
    (Math.pow(1/lstEuclideanDatas.get(p).getNilaiTarget(),2)));
133                 pembilang[1] = pembilang[1] +
    (lstEuclideanDatas.get(p).getN2() *
    (Math.pow(1/lstEuclideanDatas.get(p).getNilaiTarget(),2)));
134                 penyebut[0] = penyebut[0] +
    (Math.pow(1/lstEuclideanDatas.get(p).getNilaiTarget(),2));
135                 penyebut[1] = penyebut[1] +
    (Math.pow(1/lstEuclideanDatas.get(p).getNilaiTarget(),2));
136             }
```

```
137
138             //fknn
139                 hasil[0] = pembilang[0]/penyebut[0];
140                 hasil[1] = pembilang[1]/penyebut[1];
141
                //System.out.println("pembilang1  " + pembilang[0] +
    " penyebut1  " + penyebut[0] + " pembilang 2  " + pembilang[1]
    + " penyebut 2  " + penyebut[1] + " n1
    "+lstEuclideanDatas.get(j).getN1()+" N2
142 "+lstEuclideanDatas.get(j).getN2());
143
144             if(hasil[0] > hasil[1])
145             {
146                 hasilPrediksi.add(1);
147             }
148             else
149             {
150                 hasilPrediksi.add(2);
151             }
152
                // System.out.println(hasil[0] + " dan " + hasil[1]
153 + " dan " + hasilPrediksi.get(j) );
154         }
155
156     }
157
158     public List<EuclideanData> getLstEuclideanDatas() {
159         return lstEuclideanDatas;
160     }
161
        public void setLstEuclideanDatas(List<EuclideanData>
162 lstEuclideanDatas) {
163         this.lstEuclideanDatas = lstEuclideanDatas;
164     }
165
166     public int getK() {
167         return K;
168     }
169
170     public void setK(int K) {
171         this.K = K;
172     }
173
174     public List<Integer> getHasilPrediksi() {
175         return hasilPrediksi;
```

```
176     }
177
178     public void setHasilPrediksi(List<Integer> hasilPrediksi) {
179         this.hasilPrediksi = hasilPrediksi;
180     }
181
182     public List<AtributData> getLstAtributDataLatih() {
183         return lstAtributDataLatih;
184     }
185
186     public void setLstAtributDataLatih(List<AtributData>
    lstAtributDataLatih) {
187         this.lstAtributDataLatih = lstAtributDataLatih;
188     }
189
190 }
```

Source Code 5.4 Metode algoritma *Fuzzy k-Nearest Neighbor*

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini berisi pengujian dan analisis dari hasil implementasi *Fuzzy k-Nearest Neighbor* untuk diagnosa penyakit layu berdasarkan *wilt dataset*.

6.1 Pengujian

Pengujian bertujuan untuk mengetahui tingkat akurasi klasifikasi oleh system. Pengujian dilakukan dengan 2 macam pengujian, yaitu : pengujian tingkat akurasi terhadap data latih dan pengujian tingkat akurasi terhadap nilai k.

6.1.1 Pengujian Pengaruh Jumlah Data Latih Terhadap Tingkat Akurasi

Pada pengujian ini digunakan beberapa jumlah data latih yaitu : 60, 80, 120, 200, dan 350 data. Pada Tabel 6.1 ditunjukkan hasil tingkat akurasi yang dilakukan terhadap 30 data uji dengan nilai k = 4.

Tabel 6.1 Pengaruh jumlah data latih terhadap tingkat akurasi

Jumlah data latih	Akurasi
60	76.67%
80	73.33%
120	70%
200	66.67%
350	66.67%

6.1.2 Pengujian Pengaruh Nilai k Terhadap Tingkat Akurasi

Pada pengujian tingkat akurasi terhadap pengaruh nilai k, digunakan sebanyak 200 data latih dan data uji yang digunakan adalah sebanyak 60 data. Untuk nilai k yang digunakan adalah k = 2 sampai dengan k = n. Hasil dari pengujian tingkat akurasi terhadap nilai k dapat dilihat pada Tabel 6.2

Tabel 6.2 Pengujian nilai k terhadap tingkat akurasi

k	Akurasi
1	71.67%
2	71.67%
3	73.33%
4	71.67%
5	71.67%

Tabel 6.3 Pengujian tingkat akurasi terhadap nilai k (lanjutan)

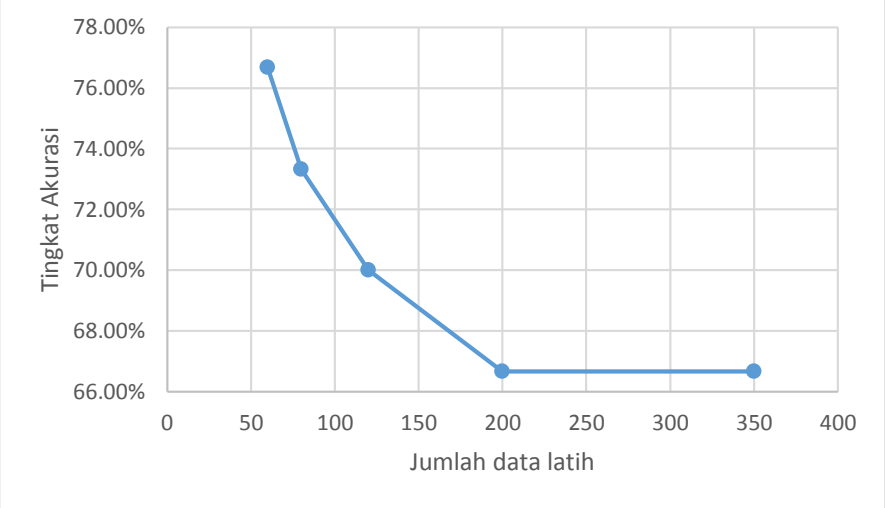
k	Akurasi
6	71.67%
7	71.67%
8	71.67%
9	73.33%
10	73.33%
15	71.67%
21	71.67%
22	70%
26	70%
27	68.33%
50	68.33%
100	71.67%

6.2 Grafik Hasil Pengujian Dan Analisis

Berdasarkan hasil pengujian tingkat akurasi terhadap jumlah data latih dan nilai k maka dibuatlah grafik hubungan tingkat akurasi terhadap jumlah data latih dan nilai k. Juga dilakukan analisis dari hasil penerapan dan pengujian *Fuzzy k-Nearest Neighbor* (FKNN) untuk diagnosa penyakit layu berdasarkan *wilt dataset* berdasarkan grafik. Analisis ini dilakukan untuk mengetahui karakteristik tingkat akurasi berdasarkan pengujian.

6.2.1 Analisis Pengaruh Jumlah Data latih Terhadap Tingkat Akurasi

Grafik pengaruh jumlah data latih terhadap tingkat akurasi ditunjukkan pada Gambar 6.1

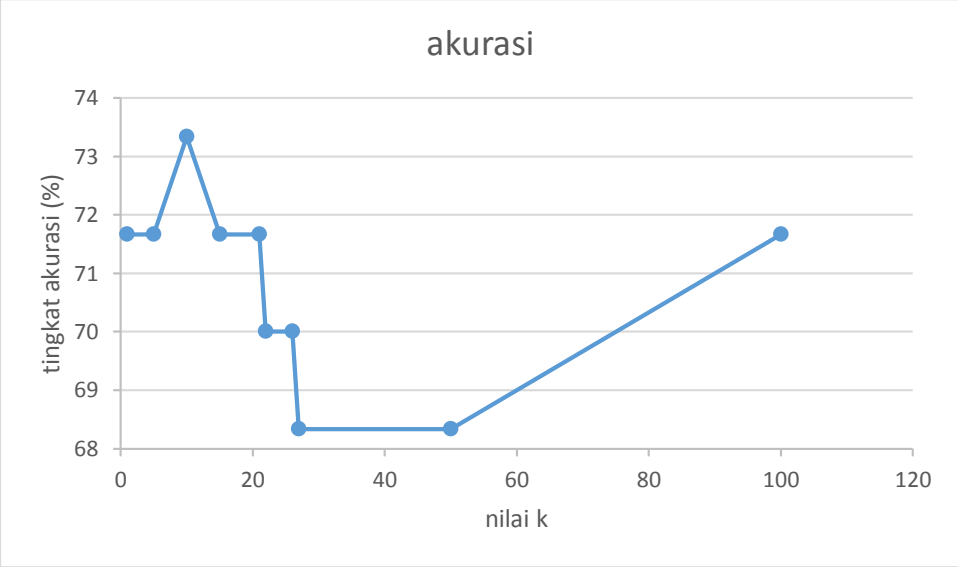


Gambar 6.1 Grafik perbandingan jumlah data latih terhadap tingkat akurasi

Pada grafik pengujian tingkat akurasi terhadap jumlah data latih terlihat bahwa jumlah data latih berpengaruh terhadap tingkat akurasi. Pada jumlah data latih sebanyak 60 tingkat akurasi sistem sebesar 76.67%, pada jumlah data latih sebanyak 80 tingkat akurasi nya turun menjadi 70.33%. Penurunan tingkat akurasi juga semakin menurun saat menggunakan 120 dan 200 data latih yaitu sebesar 70% dan 66.67%. Namun tingkat akurasi terlihat tetap sebesar 66.67% saat data latih berjumlah 200 hingga 350 data.

6.2.2 Analisis Pengaruh Nilai k Terhadap Tingkat Akurasi

Grafik pengaruh jumlah nilai k terhadap tingkat akurasi ditunjukkan pada Gambar 6.2.



Gambar 6.2 Grafik perbandingan nilai k terhadap tingkat akurasi

Berdasarkan grafik perbandingan tingkat akurasi diatas, terlihat nilai k memberikan pengaruh yang cukup kecil terhadap tingkat akurasi klasifikasi.

Tingkat akurasi sistem sebesar 71.67% saat nilai k bernilai 1 dan 2. Lalu tingkat akurasi terlihat naik menjadi 73.33% saat nilai k = 3. Namun tingkat akurasi turun kembali menjadi 71.67% saat nilai k = 4. Selanjutnya tingkat akurasi tetap sebesar 71.67% saat k bernilai 5 hingga 8. Tingkat akurasi naik kembali menjadi 73.33% saat nilai k bernilai 9 hingga 14. Saat menggunakan nilai k yang semakin besar yaitu 15 hingga 21 nilai turun kembali menjadi 71.67%, semakin turun menjadi 70% saat nilai bernilai 22 hingga 26 dan tingkat akurasi sebesar 68.33% saat k bernilai 27 hingga 50. Tingkat akurasi sistem terlihat naik dari sebelum dan stabil sebesar 71.67% saat k bernilai 51 hingga 100.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil penelitian dengan metode *Fuzzy k-Nearest Neighbor* untuk mengetahui tingkat akurasi untuk klasifikasi *Wilt Dataset* pada hutan, dapat diambil kesimpulan sebagai berikut :

1. Metode *Fuzzy k-Nearest Neighbor* (FKNN) bisa diterapkan untuk penyakit layu. Langkah pertama adalah menyiapkan data latih dan data uji yang akan digunakan lalu diproses menggunakan min-max normalization. Kemudian data latih dan data uji yang telah dinormalisasi dihitung menggunakan persamaan *Euclidean Distance*. Setelah itu output data penyakit layu ditransformasikan kedalam bentuk *Fuzzy*. Pada akhirnya didapatkan hasil diagnosa penyakit layu yang dihitung menggunakan persamaan *Fuzzy k-Nearest Neighbor* (FKNN).
2. Jumlah data latih yang semakin banyak cukup berpengaruh pada penurunan tingkat akurasi klasifikasi penyakit layu dengan menggunakan metode *Fuzzy k-Nearest Neighbor*. Namun tingkat akurasi terlihat stabil saat menggunakan data latih berjumlah 200-350 data.
3. Penentuan nilai k sedikit berpengaruh terhadap tingkat akurasi sistem, tingkat akurasi mulai menurun saat nilai k bernilai 1 hingga 27. Namun mulai mengalami peningkatan akurasi saat nilai k ≥ 50.

7.2 Saran

Ada beberapa saran yang dapat dijadikan pertimbangan agar penelitian selanjutnya menjadi lebih baik, yaitu :

1. Diharapkan penelitian berikutnya menggunakan data dengan tiga atau lebih kelas klasifikasi yang berbeda agar tingkat akurasi klasifikasi menjadi lebih akurat.
2. Dengan menggunakan metode lain atau menggunakan metode *Fuzzy k-Nearest Neighbor* yang telah dimodikasi diharapkan dapat memperoleh hasil akurasi klasifikasi yang lebih baik.

DAFTAR PUSTAKA

Adnan. 2013. Penentuan Kualitas Kopi Menggunakan Teknologi Near-Infra Red. Germany. Tersedia di : <https://postharvestnotes.wordpress.com/2013/06/11/penentuan-kualitas-kopi-menggunakan-teknologi-near-infra-red/> [diakses 1 Maret 2016].

Agusta, Yudi. 2007. "*k-Means* – Penerapan permasalahan dan metode terkait". STMIK STIKOM BALI.

Aisyah, Imas. 2012. Mengenal Gejala Penyakit Layu pada Tanaman dan Cara Menanganinya. Cianjur. Widyaiswara PPPPTK Pertanian Cianjur.

Arief A. 2001. Hutan dan Kehutanan. Yogyakarta: Penerbit Kanisius.

Hall-Beyer, Mryka. 2008. *Tutorial : GLCM Texture*. Tersedia di: <http://www.fp.ucalgary.ca/mhallbey/tutorial.htm> [diakses 13 Agustus 2016].

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.

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]

Johnson, Brian Alan, Tateishi, Ryutaro dan Hoan, Nguyen Thanh. 2013. *A hybrid pansharpening approach and multiscale object-based image analysis for mapping diseased pine and oak trees*. Japan: Chiba University.

Jowik, A. 2013. *A Learning scheme for A Fuzzy K-NN Rule*. Pattern Recognition Letters, vol 1, pp. 287-289.

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

Kasim, Anita Ahmad dan Harjoko, Agus. 2014. Klasifikasi Citra Batik Menggunakan Jaringan Syaraf Tiruan Berdasarkan *Gray Level CoOccurrence Matrices (GLCM).* Yogyakarta.

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

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

Matworks, 2016. Gray Level Co-Occurence Matrix (GLCM) Tersedia di : <http://www.mathworks.com/help/images/gray-level-co-occurrence-

matrix-glcm.html?requestedDomain=www.mathworks.com> [diakses 1 Maret 2016].

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].

Risnandar, Cecep. Hutan Hujan Tropis. Tersedia di : <https://jurnalbumi.com/hutan-hujan-tropis/> [di akses 1 Maret 2016] .

UCI *Machine Learning Repository*, 2013. *Wilt Data Set*. Tersedia di: <https://archive.ics.uci.edu/ml/datasets/Wilt> [diakses 10 Mei 2016].

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

Yuistianto, Adi. 2016. Penerapan Algoritma *Fuzzy K-Nearest Neighbor* (FKNN) pada Klasifikasi Penyakit Kelainan Tulang Belakang Berdasarkan Vertebral Column *Dataset*. S1. Universitas Brawijaya.

Zain, AS. 1996. Hukum lingkungan Konservasi Hutan. Jakarta: Penerbit Rineka Cipta.