IMPLEMENTASI *MODIFIED K-NEAREST NEIGHBOR* DENGAN OTOMATISASI NILAI K PADA PENGKLASIFIKASIAN PENYAKIT TANAMAN KEDELAI

SKRIPSI

Sebagai salah satu syarat untuk memperoleh

Gelar Sarjana dalam bidang Ilmu Komputer

Disusun oleh :

TRI HALOMOAN SIMANJUNTAK

NIM. 105060807111058

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2014

LEMBAR PERSETUJUAN

IMPLEMENTASI *MODIFIED K-NEAREST NEIGHBOR* DENGAN OTOMATISASI NILAI K PADA PENGKLASIFIKASIAN PENYAKIT PADA TANAMAN KEDELAI

SKRIPSI

Sebagai salah satu syarat untuk memperoleh
Gelar Sarjana dalam bidang Ilmu Komputer

Disusun Oleh :

TRI HALOMOAN SIMANJUNTAK

NIM. 105060807111058

Telah diperiksa dan disetujui oleh :

Pembimbing I	Pembimbing II
Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D	**Ir. Sutrisno, M.T.**
NIP. 19720919 199702 1 001	**NIP. 19570325 198701 1 001**

LEMBAR PENGESAHAN

IMPLEMENTASI *MODIFIED K-NEAREST NEIGHBOR* DENGAN OTOMATISASI NILAI K PADA PENGKLASIFIKASIAN PENYAKIT PADA TANAMAN KEDELAI

SKRIPSI

Sebagai salah satu syarat untuk memperoleh
Gelar Sarjana dalam bidang Ilmu Komputer

Disusun Oleh:

TRI HALOMOAN SIMANJUNTAK
NIM. 105060807111058

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 4 Juli 2014

Penguji I	Penguji II
Candra Dewi, S,Kom., M.Sc	**Ahmad Afif Supianto, S.Si., M.Kom.**
NIP. 19771114 2003 12 2 001	**NIK. 820623 16 1 1 0425**

Penguji III

M.Ali Fauzi S.Kom., M.Kom

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS SKRIPSI

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

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

Malang, 11 Juli 2014
Mahasiswa,

Tri Halomoan Simanjuntak
NIM. 105060807111058

KATA PENGANTAR

Segala puji syukur dan kemulian bagi Tuhan Yang Maha Esa atas hikmat dan berkat yang berikan, sehingga penulis dapat menyelesaikan laporan skripsi dengan judul **"Implementasi *Modified K-Nearest Neighbor* Dengan Otomatisasi Nilai K Pada Pengklasifikasian Penyakit Pada Tanaman Kedelai"**. Skripsi ini disusun guna memenuhi syarat memperoleh gelar Sarjana Komputer pada program Studi Informatika / Ilmu Komputer, Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.

Dalam proses penyelesaian skripsi ini penulis mendapatkan banyak bantuan dan tidak lepas dari adanya kerjasama dari berbagai pihak. Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan dan dukungan selama penulisan skripsi ini. Penulis mengucapkan banyak terima kasih kepada :

1. Bapak Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D selaku pembimbing pertama yang telah dengan sabar, tekun, tulus, dan ikhlas meluangkan waktu, tenaga, serta pikiran dalam memberikan arahan, bimbingan, motivasi dan saran-saran yang sangat berharga kepada penulis selama menyusun skripsi.
2. Bapak Ir. Sutrisno, M.T. selaku pembimbing kedua yang telah dengan sabar, tekun, tulus, dan ikhlas meluangkan waktu, tenaga, serta pikiran dalam memberikan arahan, bimbingan, motivasi dan saran-saran yang sangat berharga kepada penulis selama menyusun skripsi.
3. Drs. Marji, M.T. selaku Ketua Program Studi Ilmu Komputer / Informatika.
4. Ayahanda D. Simanjuntak dan ibunda E.Sitompul selaku orang tua penulis, Juniko Simanjuntak, Dedi Simanjuntak, Dessy Christiana Simanjuntak selaku saudara kandung beserta segenap keluarga besar yang selalu memberikan dukungan, motivasi, saran, pengharapan baru dan doa yang tiada henti.
5. Bapak Achmad Basuki, ST., MMG., Ph.D, Bapak Eko Sakti P., S.Kom., M.Kom, Bapak Kasyful Amron, ST., M.Sc serta segenap dosen beserta

staff administrasi Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya yang telah membantu selama kegiatan belajar.

6. Ko Alex, Ko Yo, Ko Lucky, Ce fitri, Bobby, Anes, Munthe, Christi, Ardo, Ce Melly dan seluruh teman-teman Kingdom Generation Community (KGC) yang selalu mendoakan, memberikan dorongan dan semangat yang luar biasa untuk segera menyelesaikan penyusunan skripsi ini.
7. Oskar, Magdalena, Novi, Yuan, Harry, Naldo, Antha, Ano dan seluruh teman PMK Daniel 2010 beserta seluruh keluarga besar PMK Daniel yang selalu memberikan penghiburan, motivasi, doa, dukungan dan semangat yang luar biasa.
8. Rekan-rekan Mahasiswa Informatika 2010 yang telah banyak memberikan bantuan, masukan, dan pengalaman selama mengikuti perkuliahan maupun dalam penulisan skripsi ini.
9. Semua pihak yang tidak dapat penulis sebut satu persatu yang telah membantu dalam penyelesaian penulisan skripsi ini.

Akhirnya, dengan segala kerendahan hati penulis menyadari masih banyak terdapat kekurangan-kekurangan, sehingga penulis mengharapkan adanya saran dan kritik yang bersifat membangun demi kesempurnaan skripsi ini. Semoga skripsi ini dapat bermanfaat bagi semua pihak, baik penulis maupun pembaca.

Malang, Juni 2014

Penulis

ABSTRAK

Tri Halomoan Simanjuntak. 2014. Implementasi *Modified K-Nearest Neighbor* Dengan Otomatisasi Nilai K Pada Pengklasifikasian Penyakit Tanaman Kedelai. Skripsi Program Studi Informatika / Ilmu Komputer, Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya. Pembimbing: Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D dan Ir. Sutrisno, M.T.

Berbagai serangan penyakit dan hama dapat menimbulkan masalah yang serius terhadap tanaman kedelai. Salah satu ancaman pengembangan tanaman kedelai bagi balai-balai penelitian dan pihak pengembang tanaman tersebut adalah gangguan hama. Serangan hama dapat menurunkan hasil kedelai hingga 80% bahkan lebih jika tidak ada pengendalian yang serius. Diperlukan aplikasi untuk menentukan jenis penyakit yang menyerang tanaman kedelai.

Penelitian ini mengimplementasikan algoritma *Modified K-Nearest Neighbor* menggunakan *Soybean Disease Data Set* yang terdiri dari 266 data dan akan dibangun aplikasi berbasis *desktop* dimana parameter nilai *K* pada algoritma tersebut ditentukan oleh program dengan menggunakan metode *Brute Force* sehingga menemukan nilai *K* terbaik. Setiap nilai *K* dengan akurasi hasil terbaik akan disimpan dan digunakan sebagai parameter nilai *K* pada proses pengujian data baru. Nilai *K* pada metode ini mendefinisikan jumlah tetangga terdekat yang digunakan untuk proses klasifikasi.

Hasil pengujian menunjukkan bahwa parameter nilai *K* sangat berpengaruh terhadap hasil klasifikasi dan akurasi yang dihasilkan. Rata-rata akurasi cenderung menurun seiring dengan penambahan nilai *k* sedangkan peningkatan jumlah data latih turut disertai dengan peningkatan hasil akurasi, untuk data latih dengan kelas tidak seimbang mengalami penurunan nilai akurasi seiring dengan bertambahnya jumlah data. Hasil akurasi tertinggi pada pengujian ini sebesar 100% dengan nilai *k*=1 dan rata-rata akurasi dari 5 percobaan sebesar 98,83%.

Kata Kunci: Klasifikasi, *Modified K-Nearest Neighbor*, Penyakit Tanaman Kedelai.

ABSTRACT

Tri Halomoan Simanjuntak. 2014. Implementation of Modified K-Nearest Neighbor With Automation Value K On Classifying Disease Soybean Plants. Minor Thesis Program of Study Information Technology / Computer Science, Program of Technology Information and Computer Science University of Brawijaya. Advisor: Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D dan Ir. Sutrisno, M.T.

There are many deseases and pest attacks that can cause serious problem to the soybean plant. Pest attacks is a threat to the soybean development research centers and the developers. Pests can reduce soybean yields up to 80% or more if no serious control. Application is needed to determine the types of diseases that attack soybean plants. This study implements the algorithm Modified K-Nearest Neighbor using Soybean Disease Data Set which consists of 266 data and will build a desktop based application where the algorithm parameters on the value of K is determined by the program by using the brute force method to find the best K value. Each value of K with accuracy the best results will be recorded and used as the parameter value of K in the process of testing new data. K values in this method to define the number of nearest neighbors used for the classification process.

The test results showed that the value of the parameter K affects the classification and the accuracy. The accuracy average tends to decrease with the addition of the value of K, while increasing the number of data training also accompanied by an increase in the accuracy of the results. The data training with imbalanced class impaired accuracy values decreased along with increasing amount of data. The results of the highest accuracy on the testing is 100% with value of k = 1 and the average accuracy for 5 experimenst is 98.83%.

Keywords: *Classification, Modified K-Nearest Neighbor, Soybean Plant Diseases.*

DAFTAR ISI

DAFTAR GAMBAR

Gambar BAB II

Gambar BAB III

Gambar BAB V

DAFTAR TABEL

TABEL BAB III

TABEL BAB V

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kedelai merupakan salah satu komoditas yang mempunyai fungsi multiguna dan sumber utama minyak nabati dunia. Di Indonesia sendiri tanaman kedelai banyak digunakan sebagai bahan dasar pembuatan kecap, tahu, dan tempe. Kedelai memiliki arti penting sebagai sumber protein nabati untuk peningkatan gizi, selain itu kedelai dapat digunakan sebagai bahan pangan yang dapat menurunkan kolesterol darah, antioksidan, dan mencegah penyakit kanker. Hingga saat ini banyak olahan makanan dan juga minuman yang berasal dari kedelai. Oleh karena itu, kebutuhan kedelai akan terus meningkat seiring dengan kesadaran masyarakat tentang makanan sehat.

Berbagai serangan penyakit dan hama dapat menimbulkan masalah yang serius terhadap tanaman kedelai. Salah satu ancaman pengembangan tanaman kedelai bagi balai-balai penelitian dan pihak pengembang tanaman tersebut adalah gangguan hama. Serangan hama dapat menurunkan hasil kedelai hingga 80% bahkan lebih jika tidak ada pengendalian yang serius, hal ini disebabkan karena lemahnnya dalam identifikasi hama dan gejala serangan, dan tindakan pengendalian yang terlambat [MWT-07]. Mengidentifikasi jenis penyakit dengan tepat menjadi masalah penting yang harus diselesaikan saat ini.

Dalam mengenali penyakit tanaman kedelai, metode klasifikasi dapat digunakan peneliti untuk membantu menentukan jenis penyakit yang menyerang dan bagaimana cara mengatasinya. Data dari faktor-faktor penyakit tanaman kedelai dapat diperlakukan sebagai *training set* yang memuat atribut yang beragam untuk membentuk sebuah model klasifikasi.

Seiring dengan perkembangan teknologi yang semakin pesat, maka dibuatlah suatu *software* yang dapat mengklasifikasikan penyakit pada tanaman kedelai, sehingga dapat membantu peneliti untuk mengetahui jenis dari suatu penyakit tanaman kedelai. Klasifikasi merupakan proses mengidentifikasi obyek ke dalam sebuah kelas, grup, atau ketegori berdasarkan prosedur, karakteristik & definisi yang telah ditentukan sebelumnya [HAN-06]. Teknik klasifikasi digunakan dengan

cara membuat model klasifikasi dengan menguji data yang telah diklasifikasi dan menemukan pola yang dapat diprediksi. Untuk membentuk sebuah model klasifikasi, terdapat data latih yang memuat atribut maupun fitur yang beragam. Tujuan dari klasifikasi ini adalah untuk menganalisa data latih dan membentuk sebuah model yang mewakili sifat dari kelas data yang mendeskripsikan sifat-sifat dari kelas tersebut berdasarkan atribut ataupun fitur termuat didalamnya, sehingga dapat digunakan untuk melakukan pengujian terhadap data test baru.

Algoritma *Modified K-Nearest Neighbor (MKNN)* merupakan pengembangan performansi dari metode *K-Nearest Neighbor (KNN).* Pemikiran utama dari metode ini adalah pengklasifikasian *sample* uji sesuai *tag* tetangganya. *MKNN* terdiri dari dua pemrosesan, pertama validasi data *training* dan yang kedua adalah menerapkan pembobotan *KNN* [HAM-08]. Berdasarkan hasil pengujian yang dilakukan oleh Hamid Parvin, didapatkan bahwa nilai K sangat berpengaruh terhadap hasil keakurasian data dan harus melalui serangakaian percobaan pendahuluuan [HAM-10]. Tingkat akurasi *MKNN* terbukti lebih baik jika dibandingkan dengan metode sebelumnya yaitu *KNN*. Terbukti dari penelitian sebelumnya pada dataset *Balance-sc* metode *KNN* mempunyai tingkat akurasi sebesar 80.69% sedangkan metode *MKNN* 85.49%, begitu juga pada *data set Monk 1* metode *KNN* memiliki tingkat akurasi 84.49% sedangkan metode *MKNN* 87.81% [HAM-10].

Berdasarkan latar belakang tersebut, penulis akan melakukan penelitian dengan judul ”**Implementasi *Modified K-Nearest Neighbor* Pada Pengklasifikasian Penyakit Tanaman Kedelai Dengan Otomatisasi Nilai K**”.

1.2 Rumusan Masalah

Mengacu permasalahn yang diuraikan dalam latar belakang, maka perumusan masalah sebagai berikut :

1. Bagaimana mengimplementasikan *Modified K-Nearest Neighbor (MKNN)* dengan otomatisasi nilai K pada pengklasifikasian jenis penyakit pada tanaman kedelai.

2. Bagaimana tingkat akurasi metode *Modified K-Nearest Neighbor (MKNN)* dengan otomatisasi nilai K dalam klasifikasi penyakit tanaman kedelai.

1.3 Batasan Masalah

Agar permasalahan yang dirumuskan lebih terfokus dan tidak terjadi pelebaran topik, maka penelitian tugas akhir ini dibatasi dalam hal :

1. Data karakteristik penyakit tanaman kedelai diambil dari https://archive.ics.uci.edu/ml/machine-learning-databases/soybean/ yang terdiri jenis penyakit dan attribute penyakitnya.

1.4 Tujuan

Tujuan penulisan tugas akhir ini adalah :

1. Membuat *software* yang dapat mengklasifikasikan jenis penyakit tanaman kedelai berdasarkan data yang ada menggunakan metode *Modified K-Nearest Neighbor (MKNN)* dengan otomatisasi pada nilai K.
2. Mengukur tingkat akurasi algoritma *Modified K-Nearest Neighbor (MKNN)* dalam pengklasifikasian penyakit pada tanaman kedelai.

1.5 Manfaat

Manfaat yang dapat diperoleh dari penelitian ini adalah :

1. Diharapkan dapat membantu memudahkan pihak-pihak yang terkait dalam mengidentifikasikan penyakit pada tanaman kedelai dengan *software* yang tersedia sehingga dapat melakukan penanganan dengan tepat.
2. Dapat mengetahui tingkat akurasi terbaik pada algoritma *Modified K-Nearest Neighbor (MKNN)* untuk pengklasifikasian penyakit pada tanaman kedelai.

1.6 Sistematika Penulisan

Sistematika penulisan yang digunakan pada tugas akhir ini sebagai berikut :

BAB I PENDAHULUAN

Memuat latar belakang penelitian, rumusan masalah, batasan masalah, tujuan dan manfaat penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini membahas mengenai teori-teori yang berkaitan dan menunjang dalam penyelesaian tugas akhir ini.

BAB III METODOLOGI PENELITIAN DAN PERANCANGAN

Bab ini membahas metode-metode yang digunakan perancangan, pengujian dan analisis dalam sistem klasifikasi penyakit pada tanaman kedelai.

BAB IV IMPLEMENTASI

Bab ini membahas implementasi dari algoritma *Modified K-Nearest Neighbor* untuk pengklasifikasian penyakit pada tanaman kedelai.

BAB V PENGUJIAN DAN ANALISA

Bab ini membahas tentang penujian terhadap program yang telah dibuat dan dilakukan analisa terhadap program tersebut.

BAB VI KESIMPULAN DAN SARAN

Dalam bab ini diberikan kesimpulan dan juga saran dari hasil penelitian berdasarkan uraian-uraian dari bab-bab sebelumnya, untuk pengembangan penelitian ini selanjutnya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan membahas tentang teori-teori yang menunjang penelitian, yaitu mengenai Algoritma *K-Nearest Neighbor* (KNN), *Modified K-Nearest Neighbor* (MKNN), Perhitungan Akurasi, dan Tanaman kedelai.

2.1 Klasifikasi

Klasifikasi adalah sebuah metode untuk menyusun data secara sistematis atau menurut beberapa aturan atau kaidah yang telah ditetapkan. Klasifikasi juga dapat memiliki arti pengelompokkan obyek ke dalam satu atau beberapa kelompok berdasarkan variable yang diamati. Klasifikasi bertujuan untuk memprediksi kelas dari suatu objek yang label atau kategorinyanya masih belum diketahui.

Klasifikasi merupakan suatu teknik *data mining* yang melihat sifat dari atribut dari kelompok data yang telah didefinisikan. Teknik ini dapat digunakan untuk memberi pengetahuan pada data baru dengan memanipulasi data yang ada yang telah diklasifikasikan dan dengan menggunakan hasilnya untuk memberikan pengetahuan atau sejumlah aturan. Aturan tersebut digunakan data baru untuk dapat diklasifikasikan terhadap suatu kategori atau kelas tertentu [SUL-12].

Klasifikasi merupakan metode untuk menyatakan objek dalam suatu kategori atau kelas tertentu berdasarkan data latih sebagai pengetahuan bagi data-data baru. Tahapan-tahapan klasifikasi terdiri dari :

1. Pembangunan model

   Model ini dibuat untuk menyelesaikan masalah klasifikasi data, yang dibuat berdasarkan data latih yang ada.

2. Penerapan model

   Model yang sudah dibangun sebelumnya digunakan untuk menentukan *attribut* atau *class* bagi data baru yang *attribut*nya belum diketahui.

3. Evaluasi

   Hasil dari tahap sebelumnya dievaluasi menggunakan parameter yang terukur sehingga dapat menyatakan apakah model tersebut dapat diterima.

2.2 *K-Nearest* Neighbor (KNN)

2.2.1 Definisi K-Nearest Neighbor (KNN)

K-Nearest Neighbor (KNN) merupakan metode yang biasa digunakan pada klasifikasi data. Algoritma digunakan untuk mengklasifikasikan terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut.

KNN merupakan suatu metode yang menggunakan algoritma *supervised* dengan hasil dari *query instance* yang baru diklasifikasikan berdasarkan mayoritas dari kategori pada KNN. Tujuan dari algoritma ini ialah mengklasifikasikan objek baru berdasarkan atribut dan *training sample* [LAR-05]. Data uji dikategorikan berdasarkan jumlah kelas dengan hasil klasifikasi yang sama, misal terdapat 3 data hasil klasifikasi dengan kelas "YA", dan 2 data hasil klasifikasi dengan kelas "TIDAK", maka data uji tersebut akan dikategorikan sebagai kelas "YA" karena data dengan klasifikasi "YA" lebih banyak dari pada "TIDAK". Algoritma KNN menggunakan klasifikasi ketetanggaan sebagai prediksi terhadap data baru.

Beberapa keuntungan dari metode ini adalah sebagai berikut :

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

Meski begitu KNN juga memiliki kelemahan seperti berikut [HAM-08]:

a. *Computation cost* yang tinggi karena perlu menghitung jarak pada setiap *data training.*
b. Membutuhkan memori yang cukup besar.
c. Perlu untuk menentukan nilai parameter K, jumlah tetangga terdekat.
d. Menggunakan perhitungan jarak, yang belum diketahui pasti fungsi jarak yang digunakan.

Prinsip umum dari algoritma ini adalah menemukan *k* data training untuk menentukan *k-nearest neighbor* berdasarkan ukuran jarak. Selanjutnya mayoritas dari *k* tetangga terdekat akan menjadi dasar untuk memutuskan kategori dari sample berikutnya [YCH-10]. Selain itu algoritma ini sendiri sering digunakan untuk klasifikasi pada teknik data mining meskipun dapat digunakan untuk estimasi dan prediksi data. Metode ini adalah contoh dari *instance-based learning* dimana data training di simpan, sehingga proses klasifikasi dari data yang belum

diklasifikasi dapat dengan mudah ditemukan dengan membandingkan data tersebut dengan data yang paling mirip di data training yang ada [LAR-05].

2.2.2 Proses K-Nearest Neighbor (KNN)

A. Langkah-langkah KNN [YCH-10]**:**

1. Tentukan parameter *K*
2. Hitung jarak antara data yang akan dievaluasi dengan semua data latih.
3. Urutkan jarak yang terbentuk dan tentukan *k* tetangga terdekat berdasarkan nilai *K* terdekat.
4. Pasang kelas yang bersesuaian.
5. Tentukan kategori berdasarkan dari kelas yang mayoritas dari kelas tetangga sebagai nilai prediksi data baru.

B. Rumus KNN :

Persamaan perhitungan untuk mencari euclidean dengan d adalah jarak dan p adalah dimensi data dengan rumus sebagai berikut:

$$\boldsymbol{d_{(x,y)} = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}} \qquad (\mathbf{2.1})$$

Dimana :

$\boldsymbol{x_i}$: Sampel data latih

$\boldsymbol{y_i}$: Data uji

$\boldsymbol{d_{(x,y)}}$: Jarak antara titik pada data latih x dan titik data uji y

$\boldsymbol{p}$: Dimensi data

2.3 *Modified K-Nearest Neighbor* (MKNN)

Ide utama dari metode ini adalah memasukan label kelas dari data berdasarkan data poin pada data latih yang sudah divalidasi dengan nilai k. Dengan kata lain, hal pertama yang dilakukan adalah perhitungan validitas untuk semua data yang terdapat pada data latih. Selanjutnya, dilakukan perhitungan *Weight Voting* pada semua data uji menggunakan validitas data [HAM-10].

2.3.1 Proses Modified K-Nearest Neighbor (MKNN)

A. Validitas Data Training

Validitas digunakan untuk menghitung jumlah titik dengan label yang sama untuk semua data pada data latih. Validitas setiap data tergantung pada setiap tetangga terdekatnya. Setelah dilakukan validasi data, selanjutnya data tersebut digunakan sebagai informasi lebih mengenai data tersebut. Persamaan yang digunakan untuk menghitung validitas setiap data latih adalah [HAM-10] :

$$\boldsymbol{Validitas}(\boldsymbol{x}) = \frac{\mathbf{1}}{\boldsymbol{H}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{H}} \boldsymbol{S}\left(\boldsymbol{lbl}(\boldsymbol{x}), \boldsymbol{lbl}\left(\boldsymbol{N_i}(\boldsymbol{x})\right)\right) \qquad (\mathbf{2.2})$$

Dimana :

H : Jumlah titik terdekat

Lbl(x) : Kelas x

Ni(x) : Label kelas titik terdekat x

Fungsi *S* digunakan untuk menghitung kesamaan antara titik $\boldsymbol{a}$ dan data ke-$\boldsymbol{b}$ tetangga terdekat. Persamaan untuk mendefinisikan fungsi *S* terdapat dalam persamaan dibawah ini :

$$\boldsymbol{S}(\boldsymbol{a}, \boldsymbol{b}) = \begin{cases} \mathbf{1}\ \boldsymbol{a} = \boldsymbol{b} \\ \mathbf{0}\ \boldsymbol{a} \neq \boldsymbol{b} \end{cases} \qquad (\mathbf{2.3})$$

Dimana :

$\boldsymbol{a}$: kelas $\boldsymbol{a}$ pada data training

$\boldsymbol{b}$ = kelas lain selain $\boldsymbol{a}$ pada data training

B. Weight Voting KNN

Weight voting KNN adalah salah satu variasi metode KNN yang menggunakan *K* tetangga terdekat, terlepas dari kelas data, tetapi menggunakan *weight voting* dari masing-masing data pada data latih [HAM-10].

Dalam metode *MKNN*, *weight* masing-masing tetangga dihitung dengan menggunakan $\mathbf{1} / (\boldsymbol{d_e} + \mathbf{0.5})$. Kemudian, validitas dari setiap data pada data latih dikalikan dengan *weight* berdasarkan pada jarak Euclidean. Sehingga metode *MKNN*, didapatkan persamaan *weight voting* tiap tetangga sebagai berikut :

$$\boldsymbol{W}(\boldsymbol{x}) = \boldsymbol{Validasi}\ (\boldsymbol{x}) \times \frac{\mathbf{1}}{\boldsymbol{d_e} + \mathbf{0,5}} \qquad (\mathbf{2.4})$$

Dimana :

W(i) : Perhitungan Weight Voting

Validasi (x) : Nilai Validasi

d_e : Jarak Euclidean

Teknik *weight voting* ini mempunyai pengaruh yang penting terhadap data yang memiliki validitas yang lebih besar dan kedekatan dengan data.

2.4 Normalisasi Data

Normalisasi pada penelitian ini digunakan untuk memperkecil *range* data morfologi tanaman kedelai. Normalisasi yang digunakan pada penelitian ini adalah *min-max normalization* yang merupakan proses transformasi nilai dari data yang dikumpulkan pada range *value* antara 0.0 dan 1.0, dimana nilai terkecil (*min*) adalah 0.0 dan nilai tertinggi (*max)* adalah 1.0 [TKE-11]. Dibawah ini adalah rumus yang digunakan untuk melakukan normalisasi *min-max* :

$$v' = \frac{v - min_A}{max_A - min_A} (new\_max_A - new\_min_A) + new\_min_A \qquad (2.5)$$

Dimana :

v' : Data Baru Setelah Normalisasi

v : Data Sebelum Normalisasi

new_max_A : Batas Nilai Max Baru adalah 1.0

new_min_A : Batas Nilai Min Baru adalah 0.0

max_A : Nilai Maximun Pada Kolom

min_A : Nilai Minimum Pada Kolom

2.5 Perhitungan Akurasi

Perhitungan ini dilakukan untuk mengetahui tingkat akurasi metode *MKNN* dalam hasil klasifikasi. Akurasi dapat diperoleh dari presentase kebenaran, yaitu perbandingan antara jumlah data uji dengan jumlah data keseluruhan dikalikan 100%. Akurasi bisa didapat melalui persamaan berikut :

$$Akurasi = \frac{Jumlah\ Data\ Uji\ Benar}{Jumlah\ Seluruh\ Data\ Uji} \times 100\% \qquad (2.6)$$

2.6 Tanaman Kedelai

Kedelai , *(Glycine max (L) Merril),* merupakan komoditi tanaman pangan nomor tiga setelah padi dan jagung. Sampai saat ini diduga berasal dari kedelai liar China, Manchuria dan korea.

Klasifikasi tanaman kedelai sebagai berikut [SUP-91]:

Divisio : Spermatophyta
Classis : Dicotyledoseae
Ordo : Rosales
Familia : Papilionaceae
Genus : Glycine
Species : *Glycine mas* (L.) Merill

Gambar 2.1 Tanaman Kedelai
Sumber: [WAW-06]

2.6.1 Morfologi Tanaman Kedelai

Tanaman kedelai umumnya tumbuh tegak, berbentuk semak, dan merupakan tanaman semusim. Morfologi tanaman kedelai didukung oleh komponen utamanya, yaitu akar, daun, batang, polong, dan biji sehingga pertumbuhannya bisa optimal [WAW-06].

A. Akar

Akar kedelai muncul dari belahan kulit biji yang muncul di sekitar misofil. Calon akar tersebut kemudian tumbuh dengan cepat ke dalam tanah, sedangkan kotiledon yang terdiri dari dua keping akan terangkat ke permukaan tanah akibat pertumbuhan cepat dari hipokotil.

Perkembangan akar kedelai sangat dipengaruhi oleh kondisi fisik dan kimia tanah jenis tanah, cara pengolahan lahan, kecukupan unsur hara serta ketersediaan air di dalam tanah.

B. Batang dan Cabang

Hipokotil pada proses perkecambahan merupakan bagian batang, mulai dari pangkal akar sampai kotiledon.. Hipokotil dan dua keping kotiledon yang masih melekat pada hipokotil akan menerobos ke permukaan tanah. Cabang akan muncul di batang tanaman. Jumlah cabang bergantung dari varietas dan kondisi tanah, tetapi ada juga varietas kedelai yang tidak bercabang. Jumlah batang tidak mempunyai hubungan yang signifikan dengan jumlah biji yang di produksi.

C. Daun

Tanaman kedelai mempunyai dua bentuk daun yang dominan, yaitu stadia kotiledon yang tumbuh saat tanaman masih berbentuk kecambah dengan dua helai daun tunggal dan daun bertangkai tiga. Umumnya, daun mempunyai bulu dengan warna cerah dan jumlahnya bervariasi. Panjang bulu bisa mencapai 1 mm dan lebar 0,0025 mm. Kepadatan bulu bervariasi, tergantung varietas, tetapi biasanya antara 3-20 buah/mm2. Jumlah bulu pada varietas berbulu lebat, dapat mencapai 3-4 kali lipat dari varietas yang berbulu normal. Contoh varietas yang berbulu lebat yaitu IAC 100, sedangkan varietas yang berbulu jarang yaitu Wilis, Dieng, Anjasmoro, dan Mahameru.

D. Bunga

Tanaman kacang-kacangan, termasuk tanaman kedelai, mempunyai dua stadia tumbuh, yaitu stadia vegetatif dan stadia reproduktif. Stadia vegetatif mulai dari tanaman berkecambah sampai saat berbunga, sedangkan stadia reproduktif mulai dari pembentukan bunga sampai pemasakan biji. Tanaman kedelai di Indonesia yang mempunyai panjang hari rata-rata sekitar 12 jam dan suhu udara yang tinggi (>30° C), sebagian besar mulai berbunga pada umur antara 5-7 minggu.

Tanaman kedelai termasuk peka terhadap perbedaan panjang hari, khususnya saat pembentukan bunga. Bunga kedelai menyerupai kupu-kupu.

E. Polong dan Biji

Polong kedelai pertama kali terbentuk sekitar 7-10 hari setelah munculnya bunga pertama. Panjang polong muda sekitar 1 cm. Jumlah polong yang terbentuk pada setiap ketiak tangkai daun sangat beragam, antara 1-10 buah dalam setiap kelompok. Pada setiap tanaman, jumlah polong dapat mencapai lebih dari 50, bahkan ratusan. Kecepatan pembentukan polong dan pembesaran biji akan semakin cepat setelah proses pembentukan bunga berhenti. Ukuran dan bentuk polong menjadi maksimal pada saat awal periode pemasakan biji. Hal ini kemudian diikuti oleh perubahan warna polong, dari hijau menjadi kuning kecoklatan pada saat masak.

F. Bintil Akar dan Fiksasi Nitrogen

Tanaman kedelai dapat mengikat nitrogen (N2) di atmosfer melalui aktivitas bekteri pengikat nitrogen, yaitu Rhizobium japonicum. Bakteri ini terbentuk di dalam akar tanaman yang diberi nama nodul atau bintil akar. Keberadaan Rhizobium japonicum di dalam tanah memang sudah ada karena tanah tersebut ditanami kedelai atau memang sengaja ditambahkan ke dalam tanah. Nodul atau bintil akar tanaman kedelai umumnya dapat mengikat nitrogen dari udara pada umur 10 – 12 hari setelah tanam, tergantung kondisi lingkungan tanah dan suhu.

2.6.2 Penyakit Tanaman Kedelai

Penyakit yang menyerang tanaman kedelai kurang begitu populer bagi petani, berbeda dengan hama yang mudah untuk mengenalnya. Hal ini disebabkan karena ukuran tubuh hama jauh lebih besar daripada mikroorganisme penyebab penyakit. Akan tetapi, penyakit dapat dikenal bila memperhatikan gejala yang ditunjukkan oleh tanaman yang sakit. Gejala yang ditimbulkan akibat serangan penyakit bermacam-macam, misalnya [DJW-87]:

A. Daun berkarat, seperti pada penyakit karat.
B. Tanaman layu, seperti pada penyakit busuk akar dan batang.
C. Gejala busuk, seperti yang terjadi pada benih, akar, dan batang.

D. Tanaman kerdil dan menguning.
E. Gejala kanker pada pangkal batang dan akar.
F. Daun berbelang-belang mosaik pada penyakit virus.
G. Beberapa gejala lain yang tidak tampak pada tanaman yang sehat.

Penyakit pada tanaman kedelai, bila ditinjau dari penyebabnya, dibagi ke dalam dua golongan, yaitu penyakit biotik dan abiotik. Penyakit biotik ialah penyakit yang disebabkan oleh kegiatan mikroorganisme yang hidup seperti jamur, bakteri, dan mikroplasma, dan virus serta sejenisnya. Sedangkan penyakit abiotik ialah penyakit yang disebabkan bukan oleh makhluk hidup, melainkan misalnya karena kekurangan unsur hara, suhu udara yang terlalu panas atau dingin, kekurangan sinar matahari, dan terendam air.

2.6.3 Jenis Penyakit Lanjut Pada Tanaman Kedelai

Berikut ini adalah beberapa penyakit yang dapat menyerang tanaman kedelai, beserta gejala dan cara penanganannya :

a. Penyakit Antraknose (*anthracnose)*

   Penyakit ini menyerang daun dan polong yang telah tua. Penularan dengan perantara biji-biji yang telah kena penyakit, lebih parah jika cuaca yang cukup lembab. Gejala dari penyakit ini adalah daun dan polong terdapat bintik-bintik kecil berwarna hitam, daun yang paling rendah rontok, polong muda yang terserang hama menjadi kosong dan isi polong tua menjadi kerdil. Pengendalian pada penyakit ini adalah dengan memperhatikan pola pergiliran tanam yang tepat, dengan melakukan penyemprotan Antracol 70 WP, Dithane M 45, Copper Sandoz [WAW-06].

Gambar 2.2 Penyakit Anthracnose
Sumber: [MAR-11]

b. Penyakit Busuk Rhizoctonia (*rhizoctonia-root-rot)*

Penyakit busuk yang disebabkan oleh *Rhizoctonia Solani (R. Solani)* ini merupakan salah satu penyakit cendawan tular yang banyak menimbulkan kerugian pada tanaman budidaya, salah satunya kedelai. Gejala serangan pada tanaman kedelai dapat dilihat dari luar tanaman dengan mengamati gejala eksternal yang tampak. Tanaman yang terserang akan menunjukkan gejala layu yang diawali dengan menguningnya daun tanaman. Gejala pada batang ditandai timbulnya bercak-bercak cokelat, yang cepat melebar sehingga batang tanaman menjadi busuk mengering dan berwarna cokelat. Menguningnya daun dan timbulnya bercak cokelat pada tanaman disebabkan oleh adanya serangan *R. Solani* pada jaringan pembuluh tanaman. Hal tersebut menyebabkan tanaman tidak mendapatkan suplai air, mineral, dan unsur hara dari dalam tanah. Pengendalian dapat dilakukan dengan menggunakan bakteri antagonis, penggunaan fungisida namun hal tersebut sangat berpengaruh buruk pada lingkungan [AKA-12].

Gambar 2.3 Penyakit Rhizoctonia root rot
Sumber: [MAR-11]

c. Penyakit *Diapothe-stem-canker*

Ekspresi awal gejala dapat terjadi selama tahap-tahap awal reproduksi, pengkembangan yang kecil, terdapat luka kecil seperti lecet bewarna coklat kemerahan didasar cabang atau tangkai. Lecet pertama yang dapat diamati pada bekas luka daun setelah tangkai daun telah jatuh. Luka menjadi panjang dan menjadi cokelat gelap dan hitam, kemudian daun menjadi cekung kedalam dan terlihat mengering dan mengeriput. Penyakit ini dapat menyebabkan kematian tanaman. Suhu sangat mempengaruhi infeksi terhadap tanaman, dengan tingkat infeksi tertinggi terjadi ketika suhu udatra antara 82 dan 93^{o} F. Saat cuaca kering lebih besar tingkat kematian berikut infeksi pada tanaman dibandingkan saat cuaca basah. *Stem Canker* dapat dengan efektif dikelola dengan mengombinasi dengan menanam variasi tanaman yang tahan dan mengurangi penanaman kedelai yang sangat penuh atau berdekatan. Membajak yang dalam dapat mengurangi sisa tanaman kedelai sebelumnya dan sebaiknya benih yang digunakan untuk penanaman selanjutnya bukan berasal dari tanaman kedelai yang pernah terserang penyakit ini. Pengendalian penyakit ini dapat dilakukan dengan penggunaan fungisida, sebaiknya melakukan penundaan penanaman

pada lahan yang sama karena kemungkinan penyakit akan timbul pada tanaman kedelai yang baru [CRG-06].

Gambar 2.4 Penyakit Diapothe-stem-canker
Sumber: [CRG-06]

d. Penyakit *Downy Mildew*

Penyakit ini disebabkan oleh cendawan *Peronospora manshurica*. Pada permukaan bawah daun timbul bercak warna putih kekuningan, umumnya bulat dengan batas yang jelas, berukuran 1-2 mm. Kadang-kadang bercak menyatu membentuk bercak lebih lebar yang selanjutnya dapat menyebabkan bentuk daun abnormal, kaku dan mirip penyakit yang disebabkan oleh virus. mampu bertahan sampai beberapa musim dalam bentuk oospora pada daun atau biji, menginfeksi tanaman dalam kondisi dingin dengan gejala klorotik pada daun. Apabila terjadi embun maka sporangium akan terbentuk dan selanjutnya tersebar pada daun baru dengan perantara udara. Perkembangan penyakit didukung oleh kelembaban tinggi dan suhu 20-22 °C. Sporulasi terjadi pada suhu 10-25 °C. Pada suhu di atas 30 °C atau di bawah 10 °C sporulasi tidak terjadi. Daun-daun lebih tahan terhadap infeksi dengan bertambahnya umur tanaman dan pada suhu tinggi. Apabila jumlah bercak kuning bertambah maka ukuran daun makin menyusut. Pengendalian dari penyakit ini adalah Perawatan benih dengan fungisida, membenamkan sisa tanaman terinfeksi, rotasi tanam selama 1 tahun atau lebih [MAR-11].

Gambar 2.5 Penyakit Downy Mildew
Sumber: [MAR-11]

e. Penyakit *Purple-seed-stain*

Gejala pada daun, batang dan polong sulit dikenali sehingga pada polong yang normal mungkin bijinya sudah terinfeksi. Gejala awal pada daun timbul saat pengisian biji dengan kenampakan warna ungu muda yang selanjutnya menjadi kasar, kaku dan berwarna ungu kemerahan. Bercak berbentuk menyudut sampai tidak beraturan dengan ukuran yang beragam dari sebuah titik sebesar jarum sampai 10 mm dan menyatu menjadi bercak yang lebih besar. Gejala mudah diamati pada biji yang terserang yaitu timbul bercak berwarna ungu. Biji mengalami diskolorasi dengan warna yang bervariasi dari merah muda atau ungu pucat sampai ungu tua dan berbentuk titik sampai tidak beraturan dan membesar. Pengendalian pada penyakit ini adalah dengan menanam benih yang sehat/bersih, perawatan benih dengan fungisida [MAR-11].

Gambar 2.6 Penyakit Purple-seed-stain
Sumber: [MAR-11]

f. Penyakit *Charcoal-rot*

Penyakit ini menyebabkan batang kedelai berubah warna menjadi abu-abu, ungu, hingga hitam seperti arang. Penyakit ini disebabkan oleh jamur Macrophomina phaseolina, jamur tersebut dapar bertahan hidup ditanah yang kering. Namun jamur tersebut tidak dapat bertahan lebih dari 7 sampai 8 minggu ditanah basah. Infeksi terjadi diawal musim pada bibit dan jamur tumbuh perlahan sampai pada cuaca kering panas terjadi setelah tanaman kedelai berbunga. Pengendalian dapat dilakukan dengan menggunakan fungisida secara tepat, melakukan rotasi tanaman, memperhatikan tingkat kesuburan tanah yang memadai untuk mengurangi kekurangan gizi dan mendorong pertumbuhan yang kuat [ADD-09].

Gambar 2.7 Penyakit Charcoal-rot
Sumber: [ADD-09]

BAB III
METODOLOGI PENELITIAN DAN PERANCANGAN

Pada bab ini akan dibahas metode perancangan yang akan digunakan dan langkah-langkah dalam penelitian. Penelitian dilakukan dengan tahapan-tahapan sebagai berikut :

1. Studi Literatur

   Mempelajari literatur-literatur yang berkaitan dengan masalah klasifikasi data menggunakan algoritma *Modified K-Nearest Neighbor* (MKNN) dan penyakit-penyakit pada tanaman kedelai. Studi literatur dilakukan untuk mendukung penelitian dan meningkatkan pemahaman terhadap permasalahan yang diangkat, serta penyelesaian permasalahan tersebut. Pengetahuan diambil dari berbagai sumber, seperti buku, jurnal, dan sumber lain yang dinilai dapat memberi tambahan wawasan untuk penelitian ini.

2. Pengumpulan Data

   Dalam tahap pengumpulan data dilakukan dengan menentukan dan mempelajari *data set* yang digunakan sebagai data training.

3. Analisis dan Perancangan Sistem

   Merancang dan membangun perangkat lunak yang mengimplementasikan algoritma Modified K-Nearest Neighbor (MKNN) dimana nilai k tidak diinputkan secara manual.

4. Implementasi sistem

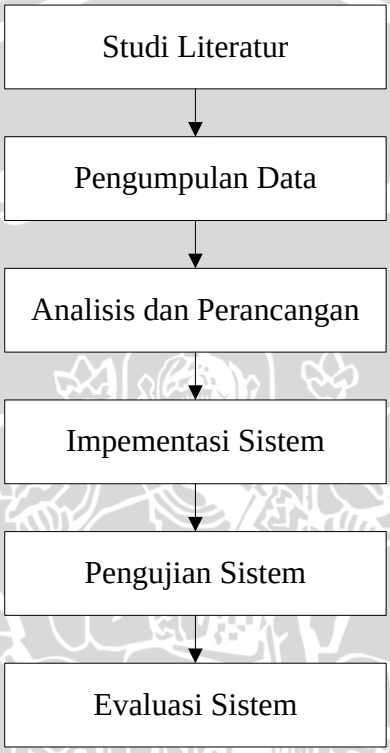
   Mengimplementasikan metode klasifikasi penyakit pada tanaman kedelai dengan algortima *Modified K-Nearest Neighbor* (MKNN) menggunakan data training yang telah disiapkan sebagai proses pembelajaran kedalam sebuah aplikasi.

5. Pengujian Sistem

   Pengujian dilakukan dengan memasukkan data uji ke sistem untuk menguji ketepatan sistem dalam mengklasifikasikan penyakit pada tanaman kedelai.

6. Evaluasi Sistem

Hasil dari pengujian tersebut dianalisis dan dievaluasi untuk mengetahui keakurasian sistem dan performansi algoritma MKNN dengan otomatisasi nilai k dalam mengklasifikan penyakit pada tanaman kedelai. Bagan alur penelitian ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Alur Penelitian
Sumber: Perancangan

3.1 Penyususan Tinjauan Pustaka

Penyusunan tinjauan pustaka sebagai dasar teori dilakukan setelah mendapatkan beberapa referensi yang tepat dari berbagai sumber yang dapat mendukung penulisan penelitian ini. Dasar teori tersebut meliputi :

1. Pengetahuan dasar tentang tanaman kedelai

   Teori-teori yang berhubungan tanaman kedelai, morfologi tanaman kedelai, gejala-gejala, macam-macam penyakit yang menyerang, dan cara penanggulangannya.

2. Algoritma *Modified K-Nearest Neighbor*

   Dasar teori meliputi tahap-tahapan perhitungan normalisasi, menghitung jarak euclidean, *Modified K-Nearest Neighbor,* dan perhitungan akurasi hasil evaluasi.

3.2 Analisis Sistem

Dalam penelitian ini analisa dan perancangan dilakukan untuk memudahkan proses penelitian ini, analisa dan perancangan ini meliputi deskripsi umum sistem, arsitektur program yang akan dibuat, dan diagram alir dari sistem.

3.2.1 Deskripsi Umum Sistem

Secara umum sistem ini akan melakukan diagnosa penyakit pada tanaman kedelai menggunakan algoritma *Modified K-Nearest Neighbor.* Nilai k yang dibutuhkan pada sistem ini tidak diinput secara manual oleh *user,* melainkan sistem akan melakukan perhitungan dengan menggunakan algoritma *brute force*. Sistem akan mengolah masukan yang diberikan oleh *user*, kemudian sistem memberikan keluaran berupa jenis klasifikasi penyakit tanaman kedelai.

Sistem ini akan menguji keakurasian hasil klasifikasi *dataset* evaluasi pada penyakit tanaman kedelai. Parameter uji yang berkaitan adalah nilai *k* (tetangga) dan data latih yang berpengaruh terhadap tingkat akurasi.

3.2.2 Deskripsi Data

Dataset yang digunakan dalam penelitian adalah dataset penyakit tanaman kedelai. Kumpulan data tersebut diperoleh dari situs *Center for Machine Learning and Intelligent Systems* (http://archive.ics.uci.edu/ml/machine-learning-databases/soybean/) yang terdiri dari 35 atribut yaitu *date, plant-stand, precip, temp, hail, crop-hist, area-damaged, severity, seed-tmt, germination, plant-growth, leaves, leafspots-halo, leafspots-marg, leafspots-size, leaf-shread, leaf-malf, leaf-mild, stem, lodging, stem-cankers, canker-lesion, fruiting-bodies, external decay, mycelium, int-discolor, sclerotia, fruit-pods, fruit spots, seed, mold-growth, seed-discolor, seed-size, shriveling,* dan *roots.*

Dataset ini dikelompokkan menjadi 15 kategori yaitu *diaporthe-stem-canker* sebanyak 10 data, *charcoal-rot* sebanyak 10 data, *rhizoctonia-root-rot* sebanyak 10 data, *phytophthora-rot* sebanyak 16 data, *brown-stem-rot* sebanyak 20 data, *powdery-mildews* sebanyak 10 data, *downy-mildew* sabanyak 10 data, *brown-spot* sebanyak 40 data, *bacterial-blight* sebanyak 10 data, *bacterial-pustule* sebanyak 10 data, *purple-seed-stain* sebanyak 10 data, *anthracnose* 20 data, *phyllosticta-leaf-spot* sebanyak 10 data, *alternarialeaf-spot* sebanyak 40 data, *frog-eye-leaf-spot* sebanyak 40 data. *Dataset* ini terdiri dari 266 data dan tidak terdapat *missing value.*

3.3 Perancangan Sistem

Pada perancangan sistem ini akan dijelaskan mengenai proses-proses dalam membangun sebuah sistem. Sistem ini akan memberikan informasi kepada *user* tentang klasifikasi penyakit pada tanaman kedelai dan sistem akan melakukan klasifikasi sesuai dengan kriteria data yang dimasukkan, klasifikasi dilakukan dengan menggunakan metode *Modified K-Nearest Neighbor* (MKKN).

3.3.1. Proses Klasifikasi Algoritma *Modified K-Nearest Neighbor*

Tahapan proses klasifikasi dengan algoritma ini adalah sebagai berikut :

1. Proses request data adalah sebuah proses yang melakukan pemanggilan data yang disimpan dalam sebuah data base.
2. Proses klasifikasi adalah sebuah proses yang menunjukkan alur perhitungan yang menjadi proses utama pada sistem ini.
3. Proses normalisasai data.
4. Proses menghitung *euclidian*.
5. Proses menghitung nilai k.
6. Proses menghitung validitas.
7. Proses *weight voting*.

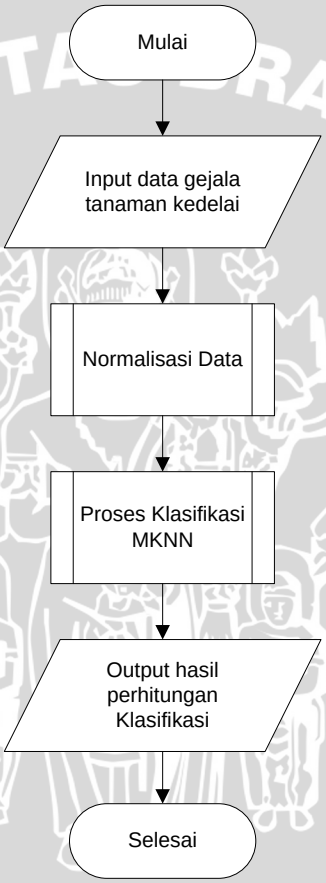
3.3.1.1. Flowchart Sistem

Langkah-langkah dalam proses ini antara lain :

1. Menginput data karakteristik tanaman kedelai
2. Melakukan proses normalisasi data.

3. Melakukan klasifikasi menggunakan MKNN untuk menentukan penyakit tanaman kedelai.
4. Output data setelah dilakukan proses klasifikasi.

Untuk lebih jelasnya proses ini dapat dilihat pada Gambar 3.2 Alur Proses Sistem. Alur proses sistem terdiri dari 2 proses yaitu proses normalisasi pada setiap data dan proses klasifikasi dengan algoritma *modified k-nearest neigbor*.

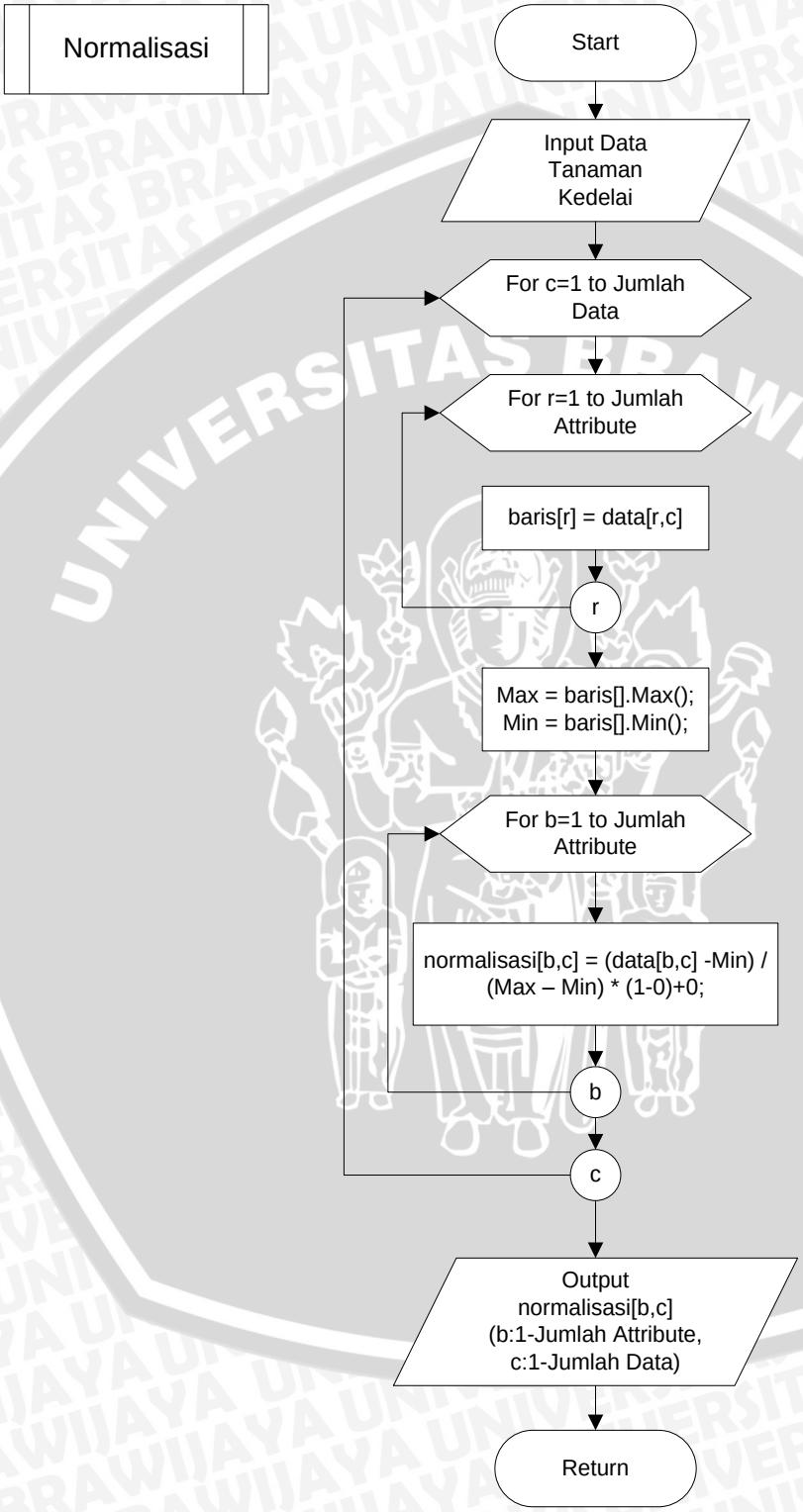


Gambar 3.2 Diagram Alir Proses Sistem
Sumber: Peracangan

3.3.1.2. Proses Normalisasi Data

Data masukan merupakan data gejala pada tanaman kedelai yang dimasukkan oleh *user*, sebelumnya data tersebut harus dinormalisasi sehingga data berada pada range [0,1], sehingga sebaran data tersebut tidak terlalu jauh. Pada perancangan ini metode normalisasi yang digunakan adalah normalisasi

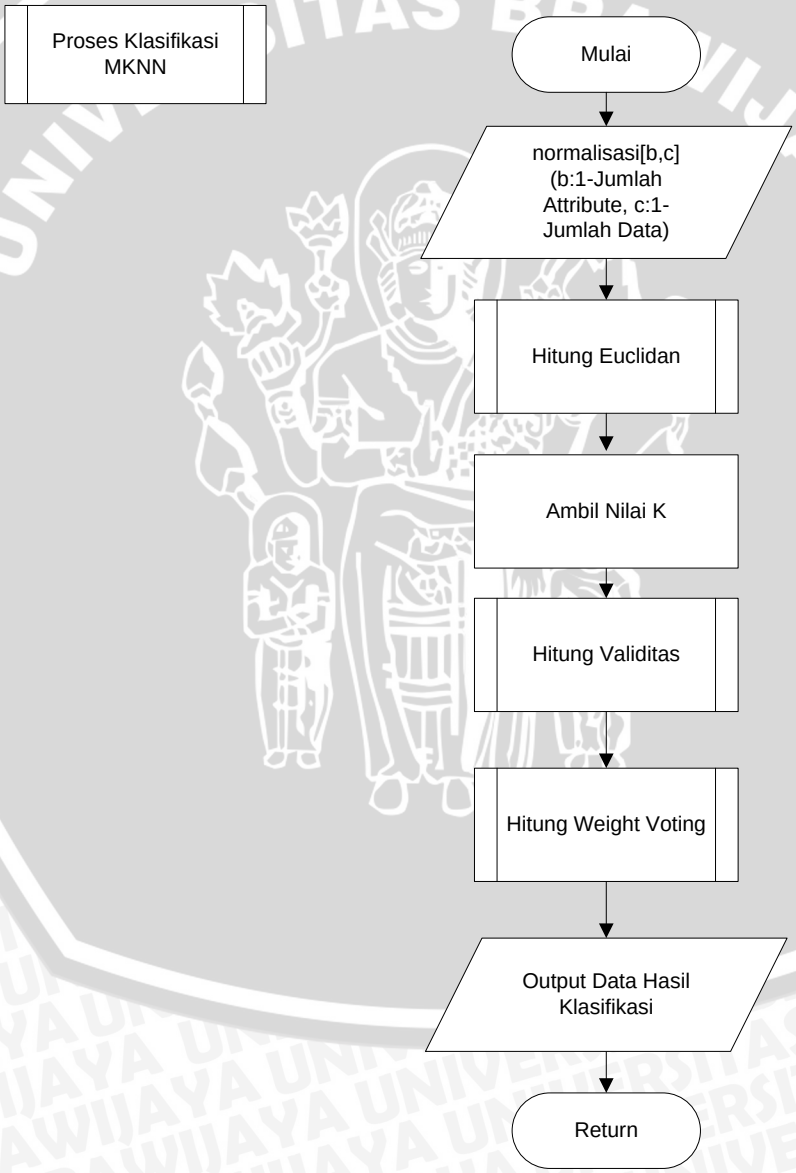
Min-Max. Flowchart untuk proses normalisasi nilai atribut ditampilkan pada Gambar 3.3.



Gambar 3.3 Diagram Alir Proses Normalisasi Data
Sumber: Perancangan

3.3.1.3. Perancangan Proses Klasifikasi

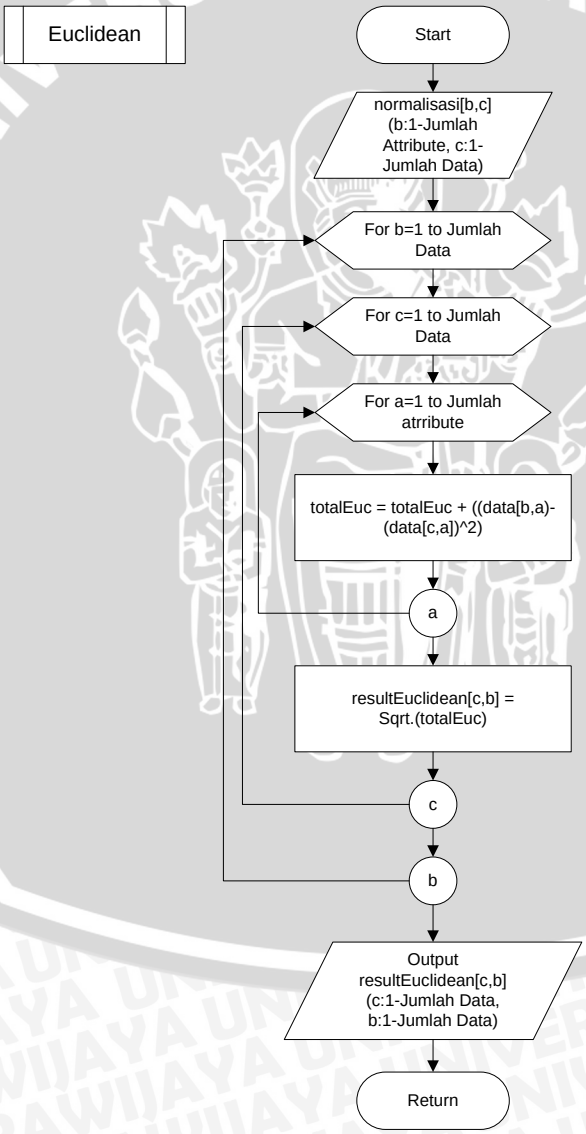
Perancangan sistem klasifikasi ini dilakukan untuk menemukan klasifikasi penyakit tanaman kedelai berdasarkan morfologi karakteristik tanaman yang dimasukkan dan *dataset* yang ada, metode ini bekerja berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tetangganya. Diagram alir proses klasifikasi sistem dengan menggunakan metode MKNN dimana nilai k sebagai parameter jumlah tetangga dalam proses klasifikasi didapat dapat secara otomatis oleh sistem. Flowchart dapat dilihat pada Gambar 3.4 dibawah ini :



Gambar 3.4 Diagram Alir Proses Klasifikasi MKNN
Sumber: Perancangan

3.3.1.4. Proses Menghitung Jarak Euclidean

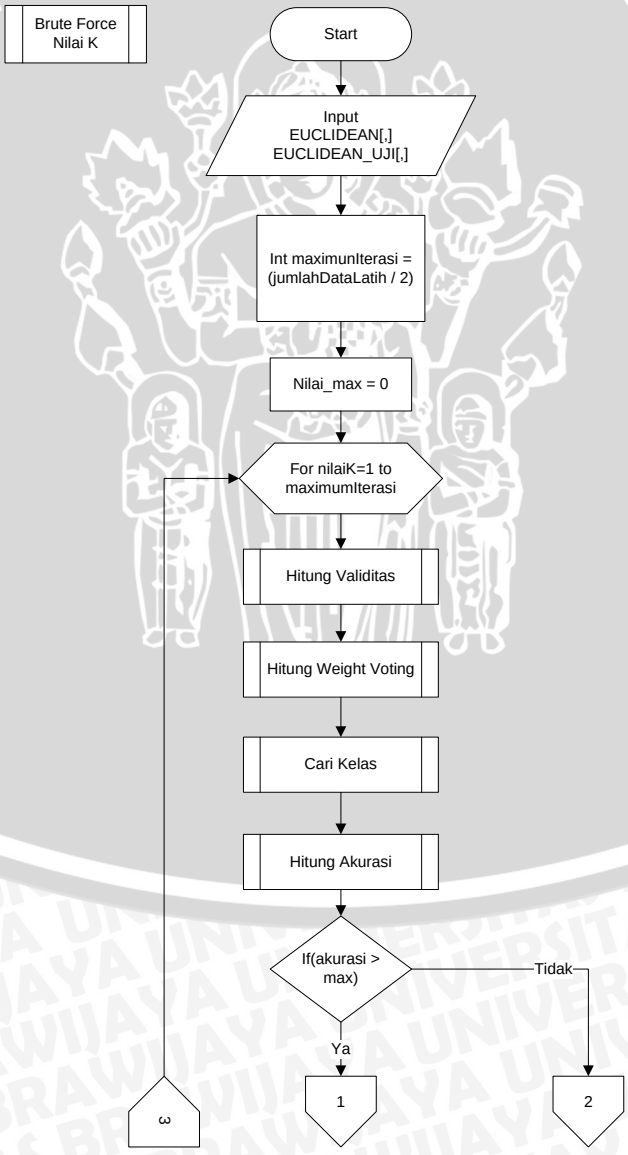
Untuk mendapatkan perhitungan Euclidean sistem diberikan masukkan berupa data kriteria morfologi tanaman kedelai yang sudah dilakukan proses normalisasi data. Perhitungan ini bertujuan untuk mendapatkan jarak data pada *dataset* sehingga dapat mengetahui tetangga mana yang terdekat ataupun yang terjauh dari suatu kumpulan data. Hasil perhitungan ini akan sangat berpengaruh terhadap penggunaan nilai *k*, dimana *k* adalah jumlah data terdekat dari suatu data yang akan diuji dan didapat berdasarkan jarak euclidean tersebut. Proses perhitungan jarak Euclidean ditunjukkan melalui Gambar 3.5 dibawah ini :

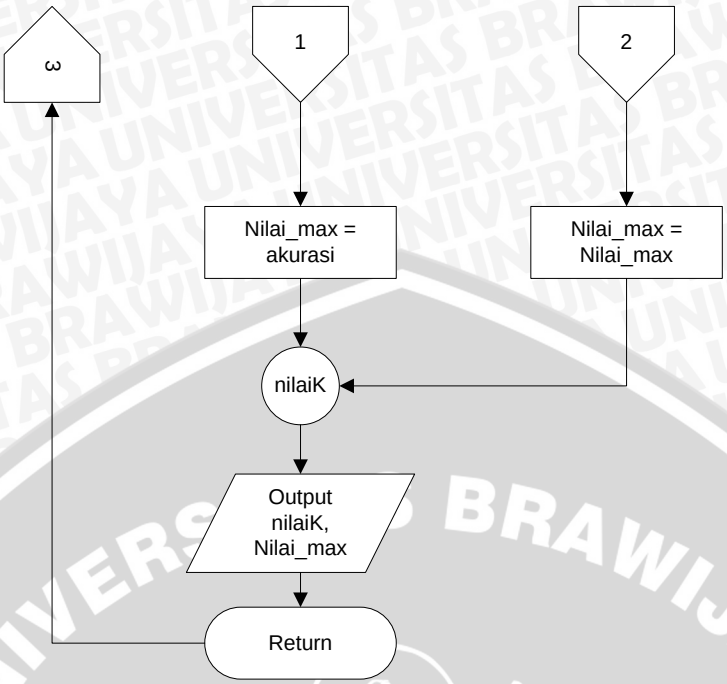


Gambar 3.5 Diagram Alir Perhitungan Jarak *Euclidean*
Sumber: Perancangan

3.3.1.5. Proses Menghitung Nilai K

Jika pada penelitian lainnya sistem mendapatkan inputan nilai k secara manual oleh *user*. Pada penelitian ini sistem mendapatkan nilai k secara otomatis dengan melakukan perhitungan menggunakan algoritma *brute force*. Proses pencarian nilai K terjadi pada proses pelatihan atau pembelajaran pada saat program menjalankan proses seperti pada subbab 4.3.1. Nilai k akan digunakan sebagai parameter jumlah tetangga yang akan digunakan untuk mendapatkan klasifikasi terhadap data baru. Langkah-langkah yang dilakukan sistem pertama-pertama sistem mendapatkan inputan jumlah data yang berasal dari perhitungan euclidean. Gambar 3.6 merupakan diagram alir proses algoritma *brute force* untuk mendapatkan nilai K.

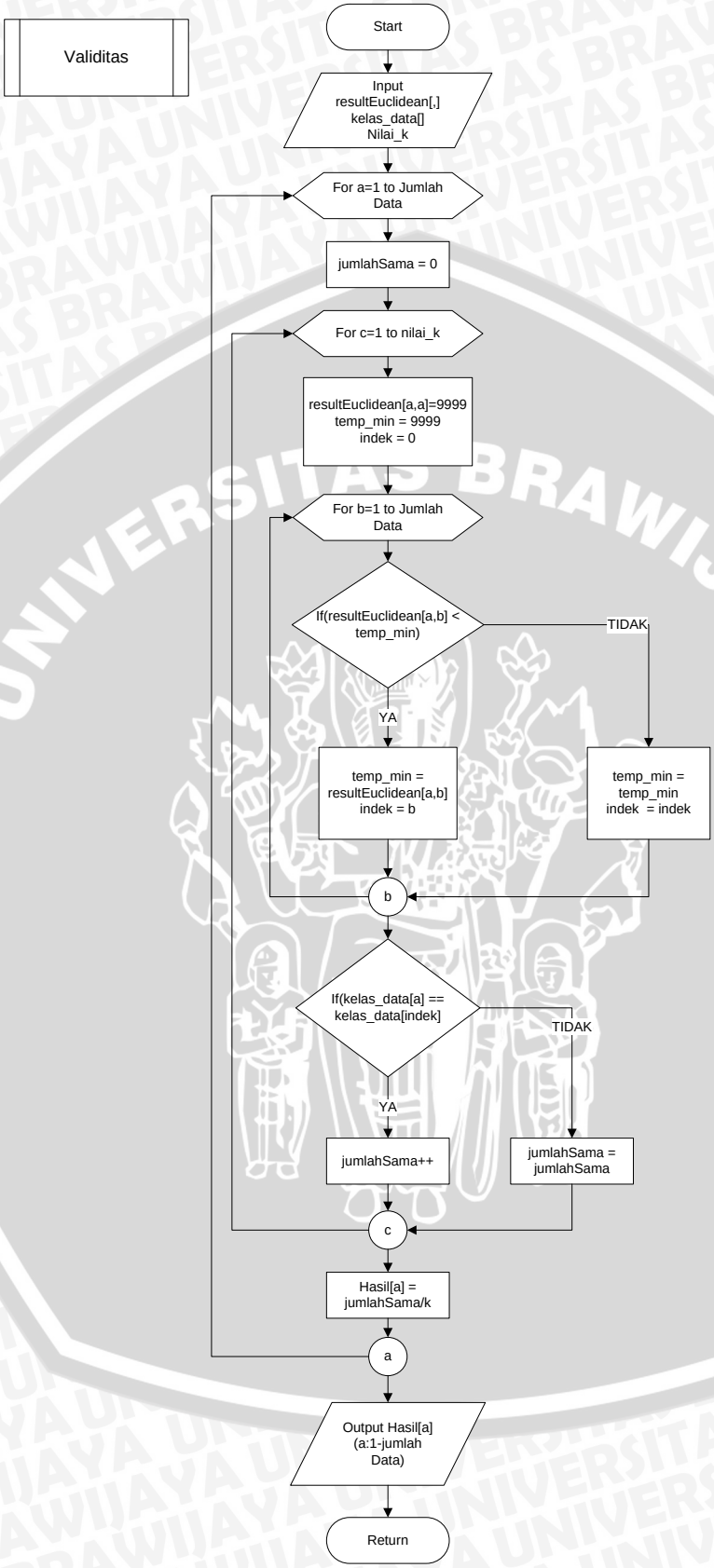




Gambar 3.6 Diagram Alir Proses Algoritma Brute Force
Sumber: Perancangan

3.3.1.6. Proses Menghitung Validitas

Langkah-langkah dalam proses ini adalah memberikan inputan data berupa hasil perhitungan euclidean data, data kelas atau kategori penyakit tanaman, dan nilai k yang ditentukan oleh sistem, sehingga dapat melakukan perhitungan validitas sesuai persamaan 2.2 yang sudah dijelaskan sebelumnya sehingga sistem dapat memberikan keluaran berupa hasil validitas. Diagram alir proses perhitungan validitas dapat dilihat pada Gambar 3.7 dibawah ini :

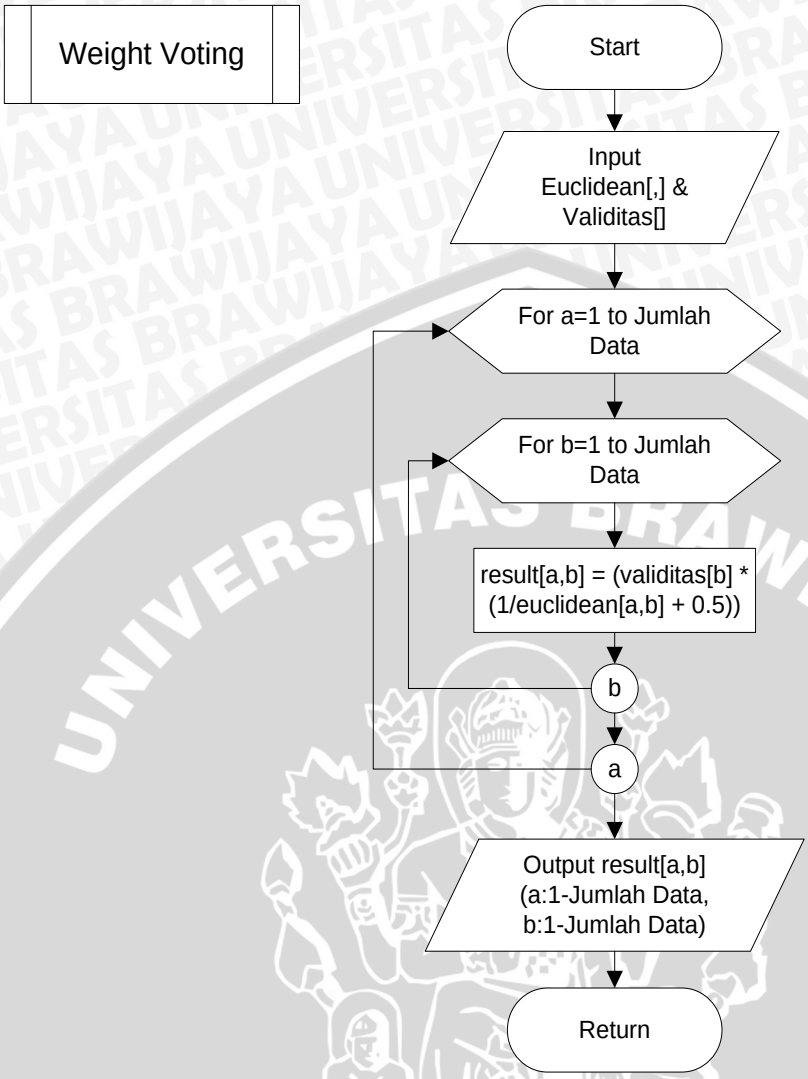


Gambar 3.7 Diagram Alir Perhitungan Validitas
Sumber: Perancangan

Perhitungan validitas dapat dilakukan setelah mendapatkan jarak euclidean yang menjadi jarak antar data, ketika *user* memasukkan data morfologi tanaman kedelai ke sistem, data tersebut diolah sesuai dengan data latih yang sudah dikumpulkan, kemudian dicari jarak antar data berdasarkan jarak euclidean, dan parameter *k* yang didapat dari sistem menentukan jumlah data terdekat yang akan dibandingkan dengan ketentuan jika label objek data 1 sama dengan objek data tangga terdekatnya maka akan diberi nilai 1, sedangkan jika label objek data tersebut tidak sama maka akan diberi nilai 0, berikut selanjutnya berdasarkan label objek dari data terdekatnya saja yang dibandingkan. Misal nilai *k*=3 maka hanya ada 3 tetangga terdekat dari data tersebut saja yang label objek datanya dibandingkan. Setelah itu nilai-nilai tersebut dikalikan dengan 1/*k*.

3.3.1.7. Proses Menghitung *Weight Voting*

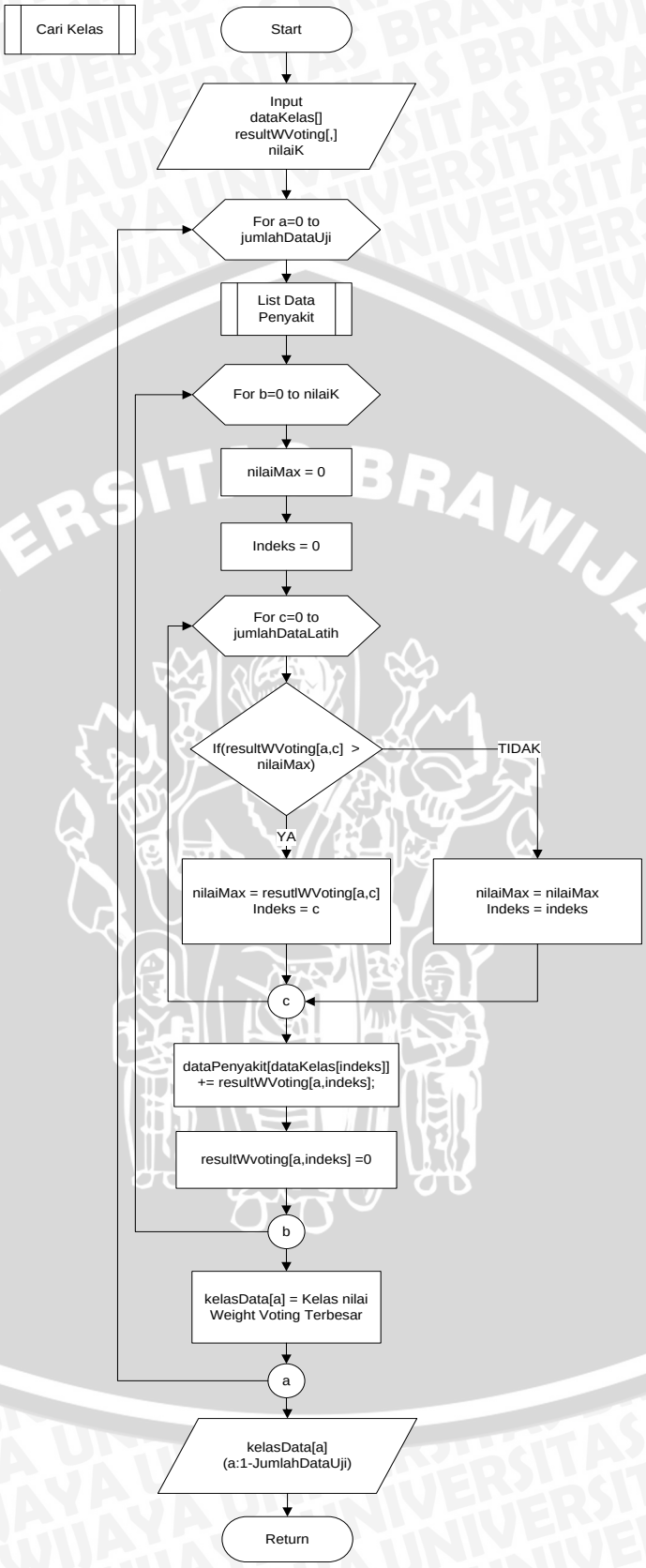
Perhitungan ini dilakukan setelah mendapatkan nilai validitas[i] dan jarak data atau Euclidean [i] untuk mendapatkan perhitungan *weight voting*. Langkah-langkah yang dilakukan dalam proses ini adalah memasukkan nilai eulidean dan nilai validitas data, melakukan perhitungan weight voting berdasarkan persamaan 2.4, sehingga sistem dapat memberikan keluaran berupa nilai *weight voting*. Diagram alir perhitungan *weight voting* ditunjukkan pada Gambar 3.8 dibawah ini:



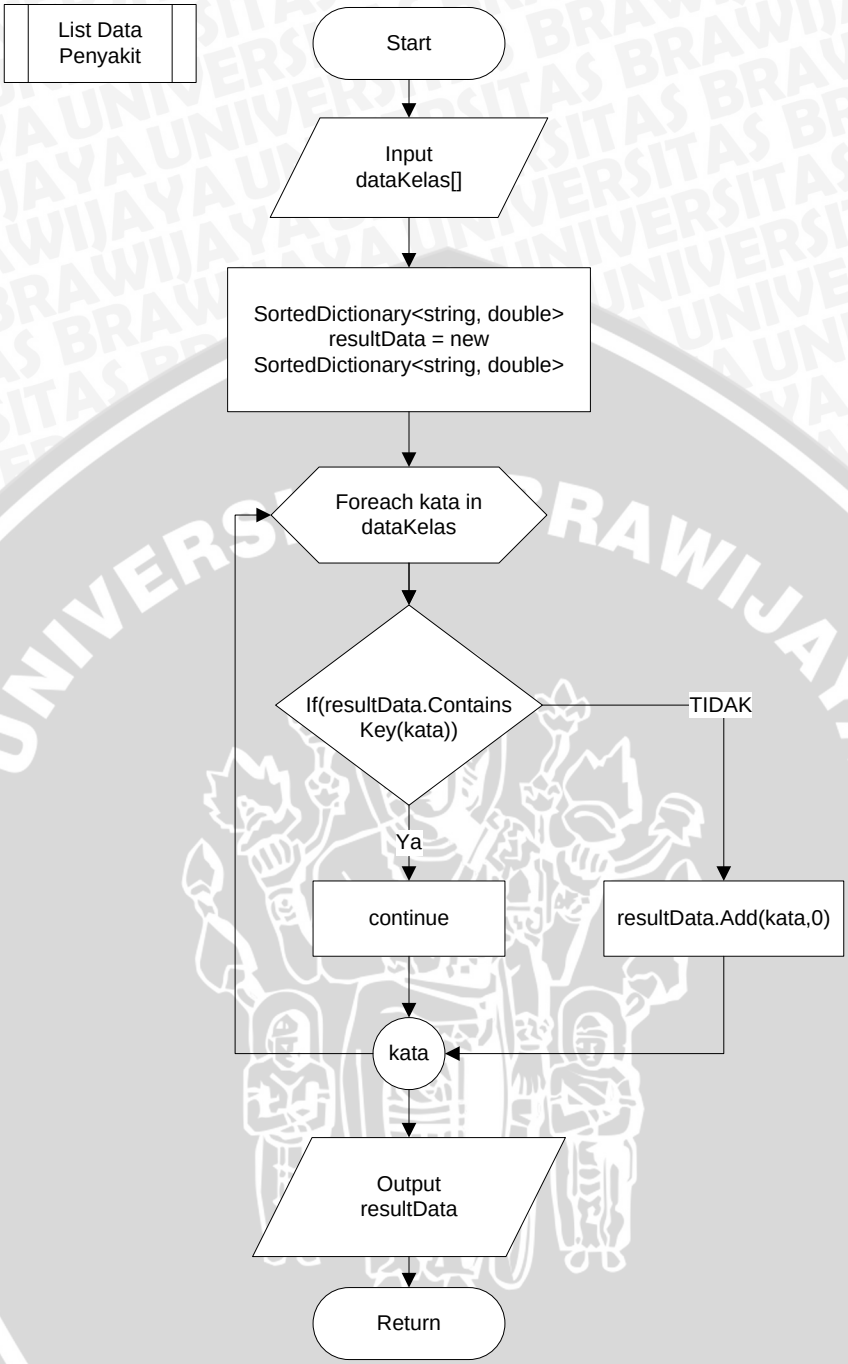
Gambar 3.8 Diagram Alir Proses *Weight Voting*
Sumber: Perancangan

3.3.1.8. Proses Mencari Kelas Data Uji

Pada proses ini bertujuan untuk mencari kelas prediksi dari data uji yang akan diklasifikasikan. Langkah-langkah dalam proses ini adalah memberikan inputan berupa data kelas yang ada, hasil dari perhitungan *weight voting*, dan nilai K yang digunakan, sehingga dapat menentukan kelas dari data uji berdasarkan kelas dengan nilai *weight voting* terbesar. Diagram alir proses perhitungan dapat dilihat pada Gambar 3.9 dibawah ini:



Gambar 3.9 Diagram Alir Pencarian Kelas
Sumber: Perancangan

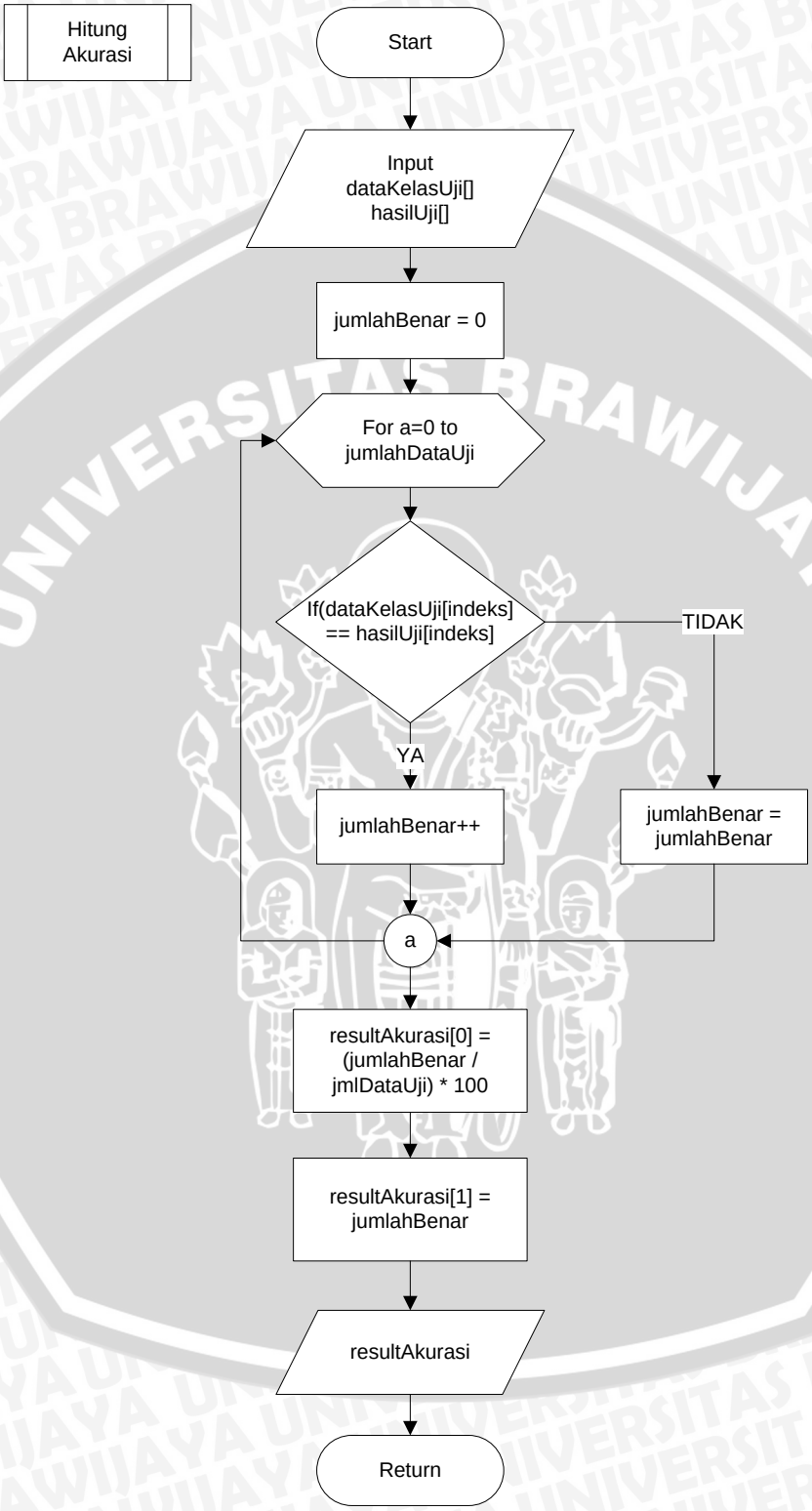


Gambar 3.10 Diagram Alir Pencarian Data Kelas Penyakit
Sumber: Perancangan

3.3.1.9. Proses Perhitungan Akurasi

Proses ini bertujuan untuk mengetahui keakuratan sistem dalam melakukan klasifikasi data dengan benar. Langkah-langkah yang dilakukan dalam proses ini adalah memasukkan data kelas uji dan data kelas hasil

klasifikasi sistem, perhitungan menggunakan persamaan 2.6. Diagram alir proses perhitungan akurasi ditunjukkan pada Gambar 3.11 dibawah ini :



Gambar 3.11 Diagram Alir Perhitungan Akurasi
Sumber: Perancangan

3.4 Contoh Perhitungan Manual

Perhitungan manual dilakukan untuk mengetahui tahapan-tahapan serta proses yang dilakukan oleh sistem secara detail, hingga sistem ini mendapatkan hasil klasifikasi penyakit tanaman kedelai berdasarkan data masukan yang diberikan. Perhitungan yang ada pada sistem dimulai dari proses normalisasi pada *dataset* dan data uji, kemudian sistem menghitung jarak euclidean, mencari nilai validitas, dan *weight voting.* Pada contoh perhitungan manual ini menggunakan 10 *dataset*.

Tabel 3.1 Data Training sebelum dinormalisasi

No	Kelas	T	AD	S	G	CH	L	ST	ED	SE	R
1	DSC	2	2	2	1	2	2	2	2	1	1
2	DSC	2	1	3	2	3	2	2	2	1	1
3	DSC	2	1	3	3	2	2	2	2	1	1
4	PR	2	2	2	1	2	2	2	2	1	1
5	PR	2	2	0	0	4	2	2	1	2	2
6	PSS	1	3	1	1	3	1	2	1	2	1
7	PSS	1	3	1	2	3	2	2	1	2	1
8	PSS	2	2	1	3	2	1	1	1	2	1
9	DSC	2	1	3	2	2	2	2	2	1	1
10	PR	3	2	0	0	3	2	2	1	0	2
Max		3	3	3	3	4	2	2	2	2	2
Min		1	1	0	0	2	1	1	1	0	1

Keterangan :

Fitur :

T : Temp

AD : area-damaged

S : severity

G : germination

CH : crop-hist

L : leaves

Kelas Penyakit :

DSC : Diaporthe-stem-canker

PR : Phytophthora-rot

PSS : Purple-seed-stain

ST : stem

ED : external-decay

SE : seed

R : roots

3.4.1 Menghitung Normalisasi Data

Langkah selanjutnya yang harus dilakukan adalah melakukan normalisasi pada setiap data latih, sehingga sebaran skala data berada dalam skala kecil tertentu seperti 0,0 – 1,0. Fungsi *min-max normalization* digunakan untuk normalisasi data seperti ditunjukkan pada persamaan 2.5, pada fungsi normalisasi ini dibutuhkan nilai *min* dan *max* pada setiap attribut data. Data yang akan dinormalisasi diinisialisasikan sebagai y_{ij}. Berikut ini adalah contoh perhitungan normalisasi :

$$y_{1,1} = \frac{2-1}{3-1}(1-0)+0 = 0{,}500 \qquad y_{1,6} = \frac{2-1}{2-1}(1-0)+0 = 1{,}000$$

$$y_{1,2} = \frac{2-1}{3-1}(1-0)+0 = 0{,}500 \qquad y_{1,7} = \frac{2-1}{2-1}(1-0)+0 = 1{,}000$$

$$y_{1,3} = \frac{2-0}{3-0}(1-0)+0 = 0{,}667 \qquad y_{1,8} = \frac{2-1}{2-1}(1-0)+0 = 1{,}000$$

$$y_{1,4} = \frac{1-0}{3-0}(1-0)+0 = 0{,}333 \qquad y_{1,9} = \frac{1-1}{2-0}(1-0)+0 = 0{,}500$$

$$y_{1,5} = \frac{2-2}{4-2}(1-0)+0 = 0{,}000 \qquad y_{1,10} = \frac{1-1}{2-1}(1-0)+0 = 0{,}000$$

Hasil perhitungan lengkap untuk setiap baris data ditunjukkan pada Tabel 3.2 dibawah ini :

Tabel 3.2 Data Training setelah dinormalisasi

Kelas	T	AD	S	G	CH	L	ST	ED	SE	R
DSC	0,500	0,500	0,667	0,333	0,000	1,000	1,000	1,000	0,500	0,000
DSC	0,500	0,000	1,000	0,667	0,500	1,000	1,000	1,000	0,500	0,000
DSC	0,500	0,000	1,000	1,000	0,000	1,000	1,000	1,000	0,500	0,000
PR	0,500	0,500	0,667	0,333	0,000	1,000	1,000	1,000	0,500	0,000
PR	0,500	0,500	0,000	0,000	1,000	1,000	1,000	0,000	0,000	1,000
PSS	0,000	1,000	0,333	0,333	0,500	0,000	1,000	0,000	1,000	0,000
PSS	0,000	1,000	0,333	0,667	0,500	1,000	1,000	0,000	1,000	0,000

Kelas	T	AD	S	G	CH	L	ST	ED	SE	R
PSS	0,500	0,500	0,333	1,000	0,000	0,000	0,000	0,000	1,000	0,000
DSC	0,500	0,000	1,000	0,667	0,000	1,000	1,000	1,000	0,500	0,000
PR	1,000	0,500	0,000	0,000	0,500	1,000	1,000	0,000	0,000	1,000

3.4.2 Menghitung Jarak Euclidean Data

Jarak euclidean dapat dihitung menggunakan persamaan 2.1. Perhitungan ini dilakukan untuk mendapatkan jarak antar setiap data pada data latih, sehingga dapat mengetahui data mana yang lebih dekat terhadap data lainnya, hasil dari perhitungan ini akan menjadi dasar dalam pemilihan data terdekat sejumlah *k*. Berikut ada contoh perhitungan jarak euclidean data 1 dan data 2:

$$\mathbf{D} = \sqrt{\begin{matrix}(\mathbf{0,500}-\mathbf{0,500})^2+(\mathbf{0,500}-\mathbf{0,000})^2+(\mathbf{0,677}-\mathbf{1,000})^2 \\ +(\mathbf{0,333}-\mathbf{0,667})^2+(\mathbf{0,000}-\mathbf{0,500})^2+(\mathbf{1,000}-\mathbf{1,000})^2+(\mathbf{1,000}-\mathbf{1,000})^2 \\ +(\mathbf{1,000}-\mathbf{1,000})^2+(\mathbf{0,500}-\mathbf{0,500})^2+(\mathbf{0,000}-\mathbf{0,000})^2 = \mathbf{0,850}\end{matrix}}$$

Berikut adalah hasil lengkap dari perhitungan jarak euclidean pada setiap data training :

Tabel 3.3 Hasil Perhitungan Jarak Euclidean Antar Data *Training*

Data	1	2	3	4	5	6	7	8	9	10
1	0,000	0,850	0,898	0,000	1,951	1,764	1,491	1,951	0,687	1,818
2	0,850	0,000	0,601	0,850	2,048	2,014	1,716	2,075	0,500	2,048
3	0,898	0,601	0,000	0,898	2,345	2,154	1,818	1,986	0,333	2,236
4	0,000	0,850	0,898	0,000	1,951	1,764	1,491	1,951	0,687	1,818
5	1,951	2,048	2,345	1,951	0,000	1,993	1,818	2,472	2,224	0,707
6	1,764	2,014	2,154	1,764	1,993	0,000	1,054	1,481	2,075	2,115
7	1,491	1,716	1,818	1,491	1,818	1,054	0,000	1,691	1,787	1,951
8	1,951	2,075	1,986	1,951	2,472	1,481	1,691	0,000	2,014	2,369
9	0,687	0,500	0,333	0,687	2,224	2,075	1,787	2,014	0,000	2,108
10	1,818	2,048	2,236	1,818	0,707	2,115	1,951	2,369	2,108	0,000

3.4.3 Menghitung Nilai Validitas Data Training

Nilai validitas dapat dihitung setelah mendapatkan nilai *k* yang dihasilkan oleh sistem dan akan menjadi parameter jumlah tetangga yang akan diambil dan diolah. Misal dalam contoh perhitungan ini nilai k=3, sehingga akan diambil 3 data terdekat berdasarkan jarak euclidean.

Berikut adalah contoh perhitungan validitas pada data 1 akan diambil 3 tetangga terdekat yaitu data ke-4 yang jaraknya 0,000 kemudian data ke-9 yang jaraknya 0,687 dan data ke-2 yang jaraknya 0,850. Kemudian bandingkan objek data ke-1 dengan data ke-4, data ke-9 dan data ke-2, dengan ketentuan jika sama beri label 1 sebaliknya jika berbeda beri label 0. Karena data ke-1 label kelasnya berbeda dengan data ke-4 maka k1=0 sedangkan data ke-9 memiliki label kelas sama dengan data ke-1 maka k2=1 dan data ke-2 memiliki label kelas yang sama dengan data ke-1 maka k3=1. Untuk mendapatkan nilai validitas dapat dilakukan perhitungan menggunakan persaman 2.2. Berikut adalah contoh perhitungannya:

$$Validitas(1) = \frac{1}{3}\sum_{i=1}^{3} S\left(lbl(1), lbl\left(N_i(x=2)\right)\right)$$

$$= \frac{1}{3}\, x\, (0+1+1)$$

$$= \frac{2}{3} = 0{,}667$$

Perhitungan yang sama dilakukan pada semua data training. Hasil perhitungan validitas data training keseluruhan seperti ditunjukkan pada Tabel 3.4 dibawah ini :

Tabel 3.4 Hasil Perhitungan Validitas

Data	K=1	K=2	K=3	Sum S(a,b)	Validitas
1	0	1	1	2	0.667
2	1	1	0	2	0.667
3	1	1	0	2	0.667
4	0	0	0	0	0.000
5	1	0	1	2	0.667
6	1	1	0	2	0.667

Data	K=1	K=2	K=3	Sum S(a,b)	Validitas
7	1	0	1	2	0.667
8	1	1	0	2	0.667
9	1	1	0	2	0.667
10	1	1	0	2	0.667

3.4.4 Pengujian Data Uji

Data uji adalah salah satu *sample* yang diambil dari *dataset*, yang belum diketahui kelasnya. Berikut adalah contoh data uji dan hasil normalisasinya yang ditunjukkan pada Tabel 3.5 dibawah ini :

Tabel 3.5 Data Uji

T	AD	S	G	CH	L	ST	ED	SE	R
3	2	0	0	4	2	2	1	0	2

Tabel 3.6 Hasil Normalisasi Data Uji

T	AD	S	G	CH	L	ST	ED	SE	R
1,000	0,500	0,000	0,000	1,000	1,000	1,000	0,000	0,000	1,000

3.4.5 Menghitung Jarak Euclidean Data Uji

Berikut adalah hasil perhitungan jarak euclidean antara data uji dengan dengan setiap data yang ada pada data training. Hasil perhitungan dapat dilihat pada Tabel 3.7 dibawah ini :

Tabel 3.7 Hasil Perhitungan Jarak Eulidean Data Uji

Data	1	2	3	4	5	6	7	8	9	10
Jarak	2,014	2,108	2,398	2,014	0,500	2,173	2,014	2,522	2,279	0,500

3.4.6 Menghitung Weight Voting

Untuk mendapatkan nilai *weight voting* dibutuhkan hasil dari nilai validitas data training sebelumnya. Sehingga untuk mendapatkan nilai *weight voting*[1] maka data yang dibutuhkan adalah validitas[1] pada data latih dan euclidean[1] atau jarak data uji dengan data ke-1 pada data latih. *Weight voting* dapat dilakukan

menggunakan persamaan 2.4. Berikut adalah contoh perhitungan *weight voting* data uji dengan menggunakan nilai validitas data ke-1 :

$$\boldsymbol{W(1) = 0,667 \times \frac{1}{2,014 + 0,5} = 0,664}$$

Dengan melakukan perhitungan yang sama pada setiap data, berikut adalah hasil keseluruhan dari perhitungan *weight voting* :

Tabel 3.8 Hasil Perhitungan Weight Voting

Data Ke	Weight Voting
1	0.664375689
2	0.649561099
3	0.611352552
4	0
5	1.666666667
6	0.640119329
7	0.664375689
8	0.597660774
9	0.625842303
10	1.666666667

Setelah nilai *weight voting* didapat, kemudian urutkan data berdasarkan nilai terbesar hingga yang terkecil, lalu ambil data sejumlah *k* terbesar. Karena pada contoh perhitungan manual ini nilai *k*=3 maka ambil 3 data terbesar yang ditunjukkan pada Tabel 3.9 dibawah ini :

Tabel 3.9 Hasil Pengolahan

Data ke-	Weight Voting	Kelas
10	1.666666667	PR
5	1.666666667	PR
7	0.664375689	PSS

Pada Tabel 3.9 terdapat 3 nilai *weight voting* terbesar yaitu data ke-10 sebesar 1.666666667 dengan kelas PR, data ke-5 sebesar 1.666666667 dengan kelas PR, dan data ke-7 sebesar 0.664375689 dengan kelas DSC. Dari ketiga nilai *weight voting* tersebut, setiap nilai *weight voting* yang memiliki kelas yang sama dijumlahkan, lalu bandingkan hasil dari penjumlahan nilai *weight voting* pada setiap kelas yang berbeda. Jika salah satu kelas memiliki hasil yang lebih besar dari kelas yang lainnya maka dapat disimpulkan bahwa data uji tergolong dalam kelas tersebut.

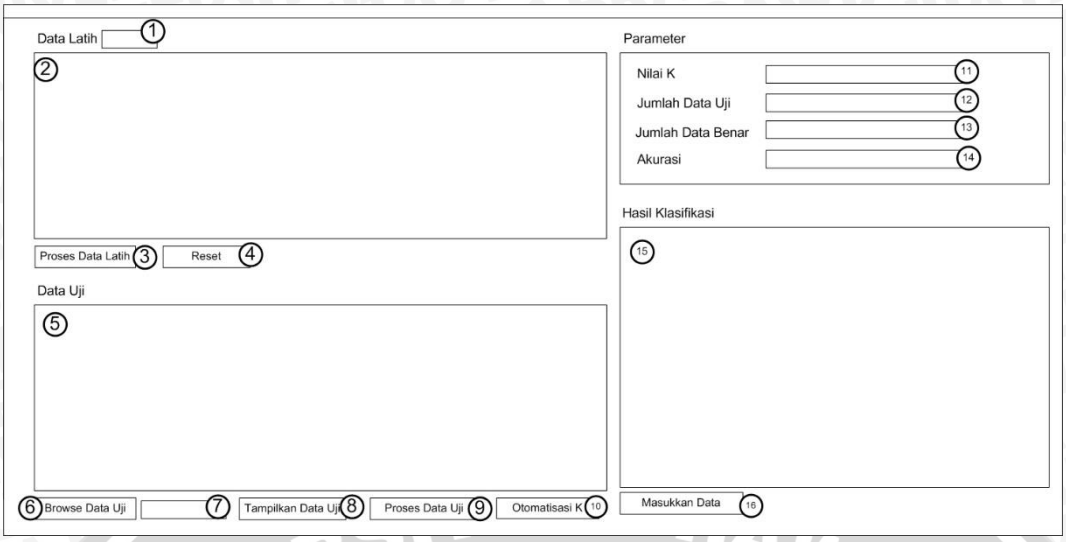
Pada perhitungan manual ini terdapat 2 kelas berbeda yaitu PR sebanyak 2 data dengan total jumlah nilai *weight voting*-nya adalah 3.333333334, dan PSS sebanyak 1 data dengan total jumlah nilai *weight voting*-nya adalah 0.664375689. Dari hasil tersebut dapat disimpulkan bahwa data uji yang diinputkan tergolong dalam kelas PR.

3.5 Perancangan Antarmuka

Perancangan ini meliputi desain *layout* aplikasi yang akan digunakan oleh *user*. Pada perancangan interface ini ada terdapat 2 bagian utama tampilan program yaitu bagian pengujian dan pelatihan.

1. Proses Pelatihan

   Pada bagian ini *user* dapat melakukan pelatihan terhadap data sebelum melakukan pengujian, data latih diproses untuk mendapatkan nilai validitas dari sehingga dapat digunakan pada pengujian data. Pada bagian ini *user* dapat melihat data latih dan dapat menginputkan sekumpulan data uji (dalam bentuk excel). Rancangan *layout* halaman ini dapat dilihat pada Gambar 3.12 dibawah ini :



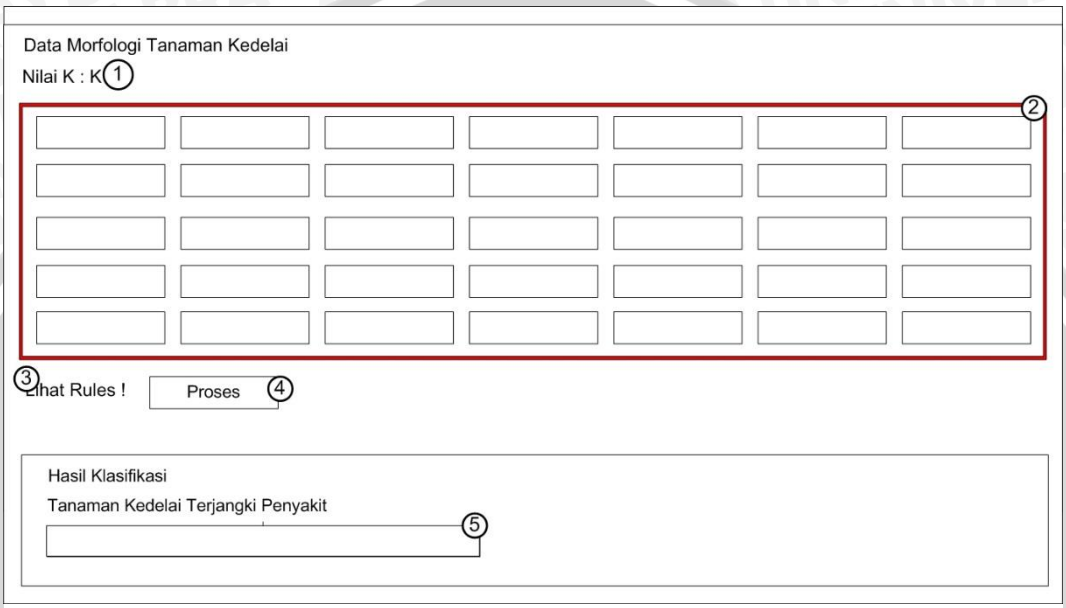
Gambar 3.12 Perancangan Antarmuka Pelatihan
Sumber : Perancangan

Pada perancangan antarmuka diatas ada beberapa tampilan antara lain :

1. *TextBox* yang akan menampilkan jumlah data latih.
2. *DataGridView* untuk menampilkan kumpulan data latih.
3. Tombol untuk memulai pemrosesan data latih.
4. Tombol untuk mereset semua proses yang sudah dilakukan untuk dapat memulai proses baru.
5. *DataGridView* yang akan menampilkan kumpulan data uji.
6. Tombol yang akan membuat dialog untuk memilih file berisi data uji dengan format .xls.
7. *TextBox* yang akan menampilkan *path* data uji yang dipilih.
8. Tombol yang akan menampilkan data uji pada *DataGridView*.
9. Tombol yang akan melakukan pemrosesan pada data uji.
10. Tombol yang akan melakukan pemrosesan klasifikasi *MKNN* dengan pencarian nilai K terbaik menggunakan *brute force.*
11. *TextBox* yang akan menampilkan parameter k yang digunakan.
12. *TextBox* yang akan menampilkan seluruh jumlah data uji.
13. *TextBox* yang akan menampilkan data uji yang benar.
14. *TextBox* yang akan menampilkan akurasi berdasarkan data uji.
15. *RichTextBox* yang akan menampilkan klasifikasi data pada setiap data uji.
16. Tombol yang akan menampilkan halaman pengujian data baru.

2. Proses pengujian

Pada bagian ini pengguna dapat memberikan inputan pada program sehingga program dapat mengolah masukan tersebut untuk dapat menentukan klasifikasi penyakit tanaman kedelai, inputan *user* ialah data tanaman kedelai, data tersebut berupa nilai *attribute*/fitur sesuai dengan data latih. Rancangan antarmuka untuk proses pengujian dapat dilihat pada Gambar 3.13 dibawah ini :



Gambar 3.13 Perancangan Antarmuka Pengujian
Sumber: Perancangan

Berdasarkan perancangan antarmuka proses pengujian pada gambar diatas terdapat bagian-bagian pada tampilan program yaitu :

1. *TextLabel* yang akan menampilkan nilai k yang digunakan untuk proses klasifikasi.
2. Kumpulan *ComboBox* digunakan untuk proses input data sesuai dengan atribut data yang ada.
3. *TextLabel* yang akan menampilkan deskripsi dari data.
4. Tombol yang digunakan untuk memulai proses klasifikasi MKNN.
5. *TextBox* yang akan jenis penyakit dari proses klasifikasi yang sudah dilakukan.

3.6 Perancangan Pengujian

Proses ini dilakukan untuk menguji dan mengetahui akurasi sistem. Hasil setiap uji coba akan dianalisis dan dihitung akurasinya. Pengujian ini menggunakan

data sampel yang akan diuji sebanyak n kali menggunakan data tes yang telah ditetapkan. Perancangan skenario uji coba sistem ditunjukkan pada Tabel 3.10 dibawah ini.

Tabel 3.10 Rancangan Tabel Pengujian Akurasi

No	Nilai K	Jumlah Data Tes	Jumlah Data Benar	Akurasi (%)
Rata-rata				

Keterangan :

Jumlah Data Tes : Jumlah data yang nantinya akan diuji.

Jumlah Data Benar : Jumlah hasil klasifikasi yang benar.

Akurasi : Tingkat akurasi perhitungan.

Rata-rata Akurasi : Jumlah akurasi data dibagi dengan jumlah data uji.

BAB IV
IMPLEMENTASI

Bab ini menjelaskan implementasi program dan analisa mengenai hasil uji coba terhadap program yang dibuat. Implementasi program merupakan penerapan dari rancangan program yang sudah dibahas pada bab sebelumnya.

4.1. Lingkungan Implementasi

Lingkungan implementasi dari rancangan aplikasi yang menerapkan *Modified K-Nearest Neighbor* dengan nilai k otomatis pada tanaman kedelai ini terdiri dari perangkat keras dan perangkat lunak.

4.1.1. Lingkungan Perangkat Keras

Lingkungan perangkat keras yang digunakan untuk mengimplementasikan dan mengembangkan aplikasi pada penelitian ini adalah sebagai berikut :

1. *Processor* Intel® Core™ 2 Duo T6500 2.10 GHz
2. *Memory* 4096MB RAM
3. *Harddisk* 500GB
4. Monitor 14”
5. *Keyboard*
6. *Mouse*

4.1.2. Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam mengimplementasikan dan pengembangan aplikasi klasifikasi penyakit tanaman kedelai menggunakan algoritma MKNN dengan otomatisasi nilai *k* adalah sebagai berikut :

1. Sistem Operasi Microsoft Windows 8 Pro
2. Microsoft Visual C# 2010 Express
3. Bahasa pemrograman yang digunakan adalah C#
4. Microsoft Excel 2013, Microsoft Word 2013, Microsoft Office Visio 2007

4.2. Implementasi Program

Implementasi program dibuat berdasarkan perancangan yang terdapat pada bab III. Program dibuat dengan menggunakan method-method, dimana method itu sendiri merupakan tahapan dan mewakili proses pelatihan atau pengujian yang terdapat pada algoritma MKNN. Proses implementasi program ini dimulai dari load data *training* dari database yang disimpan dalam dokumen excel, kemudian proses perhitungan MKNN yang terdiri dari perhitungan normalisasi data, perhitungan jarak *euclidean*, perhitungan untuk mencari nilai *k*, perhitungan validitas data *training*, dan perhitungan *weight voting*.

4.2.1. Proses Load data

Pada tahap ini dilakukan proses pengambilan data *training* dari *database*, dan menyimpannya kedalam variabel array sehingga dapat diolah untuk tahapan proses selanjutnya. Tahapan proses *load* data ditunjukkan dalam *source code* 4.1.

Proses *load* data

```
private void importExcelLatih()
        {
//inisialisasi xlWorkbook pada Library Microsoft.Office.Interop.Excel
agar dapat mengakses file excel pada "dataLatihPath"
//dataLatihpath adalah variabel string yang menyimpan path file excel
data latih dikomputer
            xlWorkBook = xlApp.Workbooks.Open(dataLatihPath, 0, true, 5,
"", "", true, Microsoft.Office.Interop.Excel.XlPlatform.xlWindows, "\t",
false, false, 0, true, 1, 0);
xlWorkSheet[0] = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
            range = xlWorkSheet[0].UsedRange;
            int panjangBaris = range.Rows.Count;
            int panjangKolom = range.Columns.Count;

//inisialisasi panjang array 2 dimensi
            dataLatih = new double[panjangBaris - 1, panjangKolom - 1];
//simpan dari index ke 1
            dataKelas = new string[panjangBaris - 1];

//memasukkan data dari tiap-tiap cell pada file excel kedalam variabel
dataLatih
            for (int r = 2; r <= panjangBaris; r++)
            {
                for (int c = 2; c <= panjangKolom; c++)
                {
dataLatih[r - 2, c - 2] = (range.Cells[r, c] as Excel.Range).Value2;
                }
dataKelas[r - 2] = (range.Cells[r, 1] as Excel.Range).Value2;
            }

//menyimpan data kelas kedalam variabel permanen pada program
            for (int r = 0; r < dataKelas.Length; r++)
```

```
            {
Properties.Settings.Default.datakelasLatih.Add(dataKelas[r]);
            }

            jmlDataLatih.Text = dataKelas.GetLength(0).ToString();
//inisialisasi file txt sebagai penyimpanan data latih dan data kelas
            string dataLatihFile = "dataLatih.txt";
            string dataKelasFile = "kelasLatih.txt";
            StreamWriter dl = new StreamWriter(dataLatihFile);
            StreamWriter dkelas = new StreamWriter(dataKelasFile);

//menulis datakelas dan datalatih kedalam data base berupa file txt
            for (int a = 0; a < dataLatih.GetLength(0); a++)
            {
                dkelas.WriteLine(dataKelas[a]);
                for (int b = 0; b < dataLatih.GetLength(1); b++)
                {
                    dl.WriteLine(dataLatih[a, b]);
                }
            }
//simpan jumlah data dan jumlah fitur data latih kedalam variabel
permanen program
Properties.Settings.Default.jumlahData = dataLatih.GetLength(0);
Properties.Settings.Default.jumlahFitur = dataLatih.GetLength(1);
            Properties.Settings.Default.Save();
            dl.Close();
            dkelas.Close();
        }

        private void importExcelUji()
        {
//inisialisasi xlWorkbook pada Library Microsoft.Office.Interop.Excel
agar dapat mengakses file excel pada "dataLatihPath"
//dataUjiPath adalah variabel string yang menyimpan path file excel data
latih dikomputer
xlWorkBook_datauji = xlApp_datauji.Workbooks.Open(dataUjiPath, 0, true,
5, "", "", true, Microsoft.Office.Interop.Excel.XlPlatform.xlWindows,
"\t", false, false, 0, true, 1, 0);
xlWorkSheet_datauji[0] =
(Excel.Worksheet)xlWorkBook_datauji.Worksheets.get_Item(1);
            range = xlWorkSheet_datauji[0].UsedRange;
            int panjangBaris = (range.Rows.Count) - 1;
            int panjangKolom = range.Columns.Count;

//inisialisasi panjang array 2 dimensi
            dataUji = new double[panjangBaris, panjangKolom - 1];
            dataKelasUji = new string[panjangBaris];

//deklarasi file txt sebagai penyimpanan data uji dan data kelas uji
            string dataUjiFile = "dataUji.txt";
            string dataKelasUjiFile = "kelasDataUji.txt";
            StreamWriter duji = new StreamWriter(dataUjiFile);
            StreamWriter dkelasUji = new StreamWriter(dataKelasUjiFile);

//simpan data tiap-tiap cell file excel kedalam variabel dataUji dan
dataKelasUji
            for (int r = 0; r < panjangBaris; r++)
            {
                for (int c = 0; c < panjangKolom - 1; c++)
```

```
                {
dataUji[r, c] = (range.Cells[r + 2, c + 2] as Excel.Range).Value2;
//cetak data pada file txt
duji.WriteLine("{0},{1} = {2}", r, c, dataUji[r, c]);
                }
dataKelasUji[r] = (range.Cells[r + 2, 1] as Excel.Range).Value2;
dkelasUji.WriteLine("{0} = {1}", r, dataKelasUji[r]);

//simpan data kelas uji kedalam variabel permanen di program

Properties.Settings.Default.datakelasUji.Add(dataKelasUji[r]);
            }
            //simpan panjang baris dari
Properties.Settings.Default.jumlahDataUji = dataUji.GetLength(0);
Properties.Settings.Default.Save();
textBox4.Text = dataUji.GetLength(0).ToString();
duji.Close();
dkelasUji.Close();
        }
```

***Source Code* 4.1** *Load Data*

4.2.2. Proses Menampilkan Data

Pada tahapan ini data akan ditampilkan pada tabel *dataGridView*, fungsi ini dapat menampilkan data latih maupun data uji dari database pada dokumen excel. Tahapan proses menampilkan data ditunjukkan pada kode program 4.2.

Proses menampilkan data

```
private void TampilData(string sourceFile, int idGrid)
        {
            try
            {
                System.Data.OleDb.OleDbConnection MyConnection;
                System.Data.DataSet DtSet;
                System.Data.DataTable dt = new DataTable();
                System.Data.OleDb.OleDbDataAdapter MyCommand;
//Menghubungkan program agar dapat mengakses file excel
MyConnection = new
System.Data.OleDb.OleDbConnection("provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" + sourceFile + ";Extended Properties=Excel 8.0;");
MyConnection.Open();
dt =
MyConnection.GetOleDbSchemaTable(System.Data.OleDb.OleDbSchemaGuid.Tables
, null);
string ExcelSheetName = dt.Rows[0]["Table_Name"].ToString();

//select data yang akan ditampilkan
MyCommand = new System.Data.OleDb.OleDbDataAdapter("select * from [" +
ExcelSheetName + "]", MyConnection);
DtSet = new System.Data.DataSet();
MyCommand.Fill(DtSet);

//jika idGrid 1 tampilkan ke dataGridview pada kolom data latih
//jika idGrid 2 tampilkan ke dataGridview pada kolom data uji
                if (idGrid == 1)
```

```
            {
                dataGridView1.DataSource = DtSet.Tables[0];
            }
            else if (idGrid == 2)
            {
                dataGridView2.DataSource = DtSet.Tables[0];
            }
            MyConnection.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
    }
```

***Source Code* 4.2** Menampilkan Data

4.2.3. Proses Klasifikasi MKNN

Pada proses klasifikasi MKNN terdiri dari beberapa tahapan, setelah melakukan *load* data, dilakukan perhitungan normalisasi, perhitungan jarak *euclidean*, perhitungan untuk mencari nilai *k*, kemudian perhitungan validitas data *training* berdasarkan nilai *k* yang didapat pada proses sebelumnya. Perhitungan data uji dapat dilakukan setelah mendapatkan validitas pada setiap data *training*, selanjutnya terdapat tahapan perhitungan jarak *euclidean* data uji terhadap setiap data training, hingga akhirnya dilakukan proses perhitungan *weight voting* untuk mendapatkan bobot pada data uji.

4.2.3.1. Normalisasi

Pada tahap ini akan dilakukan proses normalisasi yang berguna untuk menyetarakan sebaran data sehingga akan memudahkan proses perhitungan pada tahap selanjutnya. Proses normalisasi dilakukan pada data latih ataupun pada data uji. *Source code* proses normalisasi sesuai dengan persamaan 2.5 ditunjukkan pada kode program 4.3.

Normalisasi

```
    class Normalisasi
    {
//Digunakan untuk normalisasi data latih
        public static double[,] prosesNorLatih(double[,] dataLatih)
        {
            double max;
            double min;
            double[] baris = new double[dataLatih.GetLength(0)];
            for (int c = 0; c < dataLatih.GetLength(1); c++)
            {
```

```
                for (int r = 0; r < dataLatih.GetLength(0); r++)
                {
                    baris[r] = dataLatih[r, c];
                }
                max = baris.Max();
                min = baris.Min();

//melakukan perhitungan normalisasi dan menyimpannya pada index array
yang sama
                for (int r = 0; r < dataLatih.GetLength(0); r++)
                {
dataLatih[r, c] = (((dataLatih[r, c]) - min)/(max - min))*(1 - 0) + 0;
                }
            }
            return dataLatih;
        }

//Dipanggil untuk proses normalisasi data uji
        public static double[,] prosesNorUji(double[,] dataUji)
        {
            double max;
            double min;
            double[] baris = new double[dataUji.GetLength(0)];
            for (int c = 0; c < dataUji.GetLength(1); c++)
            {
                for (int r = 0; r < dataUji.GetLength(0); r++)
                {
                    baris[r] = dataUji[r, c];
                }
                max = baris.Max();
                min = baris.Min();

//melakukan perhitungan normalisasi dan menyimpannya pada index array
yang sama
                for (int r = 0; r < dataUji.GetLength(0); r++)
                {
dataUji[r, c] = (((dataUji[r, c]) - min) / (max - min)) * (1 - 0) + 0;
                }
            }
            return dataUji;
        }

    }
```

***Source Code* 4.3** Normalisasi Data

4.2.3.2. Hitung Jarak *Euclidean*

Pada tahap ini akan dilakukan proses perhitungan untuk mendapatkan jarak *euclidean* pada setiap data, baik data latih ataupun data uji. *Source code* proses perhitungan *euclidean* ditunjukkan pada kode program 4.4.

Euclidean

```
class Euclidean
    {
        //Dipanggil untuk menghitung euclidean antar dataLatih
```

```
        public static double[,] prosesEucLatih(double[,] dataLatih)
        {
//deklarasi variabel yang menyimpan hasil perhitungan eulidean
double[,] resultEuclidean = new double[dataLatih.GetLength(0),
dataLatih.GetLength(0)];
            for (int b = 0; b < dataLatih.GetLength(0); b++)
            {
                for (int c = 0; c < dataLatih.GetLength(0); c++)
                {
                    double totalEuc = 0;
                    for (int a = 0; a < dataLatih.GetLength(1); a++)
                    {
//Math.pow berarti : (dataLatih[b, a]) - (dataLatih[c, a]) kemudian
hasilnya dipangkatkan 2
//Konsep perhitungan dengan jarak Euclidean
totalEuc = totalEuc + (Math.Pow((dataLatih[b, a]) - (dataLatih[c, a]),
2));
                    }
//mencari akar dari variabel totalEuc
resultEuclidean[c, b] = Math.Sqrt(totalEuc);
                }
            }
return resultEuclidean;
        }

//Dipanggil untuk melakukan proses perhitungan euclide antar data
latih dan data uji
public static double[,] prosesEucUji(double[,] dataLatih, double[,]
dataUji)
        {
//deklarasi variabel yang menyimpan hasil perhitungan eulidean
double[,] resultEuclideanUji = new double[dataUji.GetLength(0),
dataLatih.GetLength(0)];
            for (int b = 0; b < dataUji.GetLength(0); b++)
            {
                for (int c = 0; c < dataLatih.GetLength(0); c++)
                {
                    double totalEuc = 0;
                    for (int a = 0; a < dataLatih.GetLength(1); a++)
                    {
//Math.pow berarti : (dataUji[b, a]) - (dataLatih[c, a]) kemudian
hasilnya dipangkatkan 2
//Konsep perhitungan dengan jarak Euclidean
totalEuc = totalEuc + (Math.Pow((dataUji[b, a]) - (dataLatih[c,
a]),2));
                    }
//mencari akar dari variabel totalEuc
resultEuclideanUji[b, c] = Math.Sqrt(totalEuc);
                }
            }
            return resultEuclideanUji;
        }
    }
```

***Source Code* 4.4** Hitung Jarak *Euclidean*

4.2.3.3. Hitung Validitas Data

Pada perhitungan validitas terdapat proses untuk membandingkan kelas pada data latih dengan data tetangga yang terdekat sejumlah *k*, oleh sebab itu sebelum melakukan perhitungan validitas sebelumnya nilai *k* sudah ditentukan. Sesuai dengan persamaan 2.2, kode program 4.5 menunjukkan proses untuk mendapatkan nilai validitas pada data latih.

Validitas

```
class validity
    {
public static double[] prosesValidity(double[,] resultEuc, string[]
kelasData, double k)
        {
            double[] hasil = new double[resultEuc.GetLength(0)];

//perhitungan validitias dilakukan sejumlah data latih
            for (int a = 0; a < resultEuc.GetLength(0); a++)
            {
                double jmlSama = 0;
//proses pencarian tetangga terdekat sebanyak k berdasarkan jarak
euclidean terkecil
                for (int c = 0; c < k; c++)
                {
                    double tempMin = 9999;
                    resultEuc[a, a] = 9999;
                    int indek = 0;

                    for (int b = 0; b < resultEuc.GetLength(1); b++)
                    {
//percabangan jika nilai euclidean < tempMin maka nilai tempMin =
nilai euclidean
//proses ini dilakukan sebanyak nilai k sehingga akan didapat data
terdekat sebanyak k
                        if (resultEuc[a, b] < tempMin)
                        {
                            tempMin = resultEuc[a, b];
                            indek = b;
                        }
                    }
//setelah mendapatkan data-data terdekat sebanyak k
//proses perbandingan kelas data terdekat dengan index saat ini (data
ke 1-terakhir)
//rumus dibawah berdasarkan persamaan 2.3
                    if (kelasData[a] == kelasData[indek])
                    {
//variabel ini akan menyimpan jumlah kelas yang sama
                        jmlSama++;
                    }
                    resultEuc[a, indek] = 9999;
                }
//bagi jumlah kelas yang sama dengan nilai k, simpan dalam array hasil
2 dimensi
//rumus dibawah berdasarkan persamaan 2.2
```

```
                hasil[a] = jmlSama / k;
            }
            //mengembalikan nilai saat method ini dipanggil
            return hasil;
        }
    }
```

***Source Code* 4.5** Hitung Validitas Data Latih

4.2.3.4. Hitung Weight Voting

Perhitungan *weight voting* pada sistem dilakukan untuk mendapatkan bobot nilai data uji terhadap data *training* yang diperoleh dari perbandingan data validitas dengan *euclidean.* Berdasarkan persamaan 2.4 untuk mendapatkan nilai *weight voting* ditunjukkan pada kode program 4.6.

Weight Voting

```
class weightVoting
    {
public static double[,] prosesWeightVotingDataMultiple(double[]
resultValiditas, double[,] resultEuclidean)
        {
//deklarasi variable yang menyimpan hasil perhitungan weight voting
double[,] resultWeightVoting = new
double[resultEuclidean.GetLength(0), resultEuclidean.GetLength(1)];

//dilakukan proses pembobotan setiap data uji terhadap setiap data
latih
for (int jmlWv = 0; jmlWv < resultEuclidean.GetLength(0); jmlWv++)
            {
                for (int b = 0; b < resultEuclidean.GetLength(1); b++)
                {
//proses perhitungan berdasarkan persamaan 2.4
resultWeightVoting[jmlWv,b] = (resultValiditas[b] * (1 /
(resultEuclidean[jmlWv,b] + 0.5)));
                }
            }
            return resultWeightVoting;
        }
    }
```

***Source Code* 4.6** Hitung *Weight Voting*

4.2.3.5. Proses Klasifikasi

Proses klasifikasi dilakukan untuk mendapatkan kelas data uji berdasarkan nilai *weight voting.* Proses klasifikasi dilakukan dengan melakukan perulangan sejumlah *k,* dan mencari data terbesar sejumlah *k* lalu menambahkan nilai *weight voting* pada kelas yang sama. Hasil klasifikasi kelas pada data uji

ditentukan pada kelas dengan jumlah nilai *weight voting* terbesar, seperti bisa dilihat pada kode program 4.7.

Proses Klasifikasi

```
class cariKelas
    {
public static string[] getKelasData(string[] dataKelas, double[,]
resultWVoting, double nilaiK)
        {
string[] kelasData = new string[resultWVoting.GetLength(0)];
//proses klasifikasi kelas setiap data uji
            for (int a = 0; a < resultWVoting.GetLength(0); a++)
            {
//dilakukan untuk mendapatkan seluruh kategori kelas apa saja pada
data latih
SortedDictionary<string, double> dataPenyakit =
dataPenyakitList(dataKelas);

//mencari nilai bobot terbesar sejumlah k
                for (int b = 0; b < Convert.ToInt32(nilaiK); b++)
                {
                    double nilaiMax = 0;
                    int indeks = 0;
                  for (int c = 0; c < resultWVoting.GetLength(1); c++)
                    {
                        if (resultWVoting[a, c] > nilaiMax)
                        {
                            nilaiMax = resultWVoting[a, c];
                            indeks = c;
                        }
                    }

//masukkan nilai weight voting berdasarkan label kelas data diindeks c
dataPenyakit[dataKelas[indeks]] += resultWVoting[a, indeks];
                    resultWVoting[a, indeks] = 0;
                }
//pilih kelas dengan jumlah nilai bobot terbesar
kelasData[a] = dataPenyakit.OrderByDescending(key =>
key.Value).First().Key;

            }
            return kelasData;
        }

public static SortedDictionary<string, double>
dataPenyakitList(string[] dataKelas)
        {
SortedDictionary<string, double> resultData = new
SortedDictionary<string, double>();
            int i = 0;
            foreach (string kata in dataKelas)
            {
//MessageBox.Show("kata" + i.ToString() + "" + kata.ToString());
                if (resultData.ContainsKey(kata))
                {
//MessageBox.Show("tampung" + i.ToString() + "" +
tampung.ContainsKey(kata).ToString());
                    continue;
```

```
                }
                else
                {
//MessageBox.Show("ini adalah else ke  " +i.ToString()+"ditambahkan");
                    resultData.Add(kata, 0);
                }
                i++;
            }
            return resultData;
        }
    }
```

***Source Code* 4.7** Proses Klasifikasi Data

4.2.3.6. Otomatisasi Nilai K

Proses otomatisasi nilai k dilakukan dengan melakukan perulangan dari 1 hingga n/2, dimana n adalah jumlah data latih. Dari proses tersebut setiap hasil pada nilai k tertentu akan disimpan hingga perulangan terakhir dan bandingkan setiap hasil akurasinya, nilai k yang dipilih adalah nilai k dengan akurasi tertinggi, nilai k tersebut akan disimpan dan akan digunakan sebagai parameter nilai k pada pengujian data baru. Proses otomatisasi nilai k ditunjukkan pada kode program 4.8

***Brute Force* Nilai K**

```
private void otomatisasiNilaiK()
        {
//deklarasi variabel array
string[] dataKelasUji = new
string[Properties.Settings.Default.datakelasUji.Count];
string[] dataKelasLatih = new
string[Properties.Settings.Default.datakelasLatih.Count];

//Copy array permanen pada program kedalam array yang dideklarasikan
sebelumnya
Properties.Settings.Default.datakelasUji.CopyTo(dataKelasUji, 0);
Properties.Settings.Default.datakelasLatih.CopyTo(dataKelasLatih, 0);

//deklarasi maksimum nilai k
int maximumIterasi = (Properties.Settings.Default.jumlahData) / 2;
int jumlahDataLatih = Properties.Settings.Default.jumlahData;
int jumlahDataUji = Properties.Settings.Default.jumlahDataUji;

//deklarasi file penyimpanan txt
string hasilOto = "OtomatisasiResult.txt";
string hasilKlasifikasi = "hasilKlasifikasi.txt";
StreamWriter otomatis = new StreamWriter(hasilOto);
StreamWriter hasilKlasi = new StreamWriter(hasilKlasifikasi);

double nilaiMax = 0;

ArrayList goodAkurasi = new ArrayList();
```

```
            for (int loop = 1; loop <= maximumIterasi; loop++)
            {
//deklarasi nilai k sama dengan index loop saat ini
double nilaiK = Convert.ToDouble(loop);

//load data euclidean data latih dari file txt dan disimpan dalam
variabel
euclideanLatih = getEuclFromFile.getEuclideanFromFile("euclideanPure",
jumlahDataLatih, jumlahDataLatih);

//menjalankan proses perhitungan valitidy pada method di class
validity
validitasDataLatih = validity.prosesValidity(euclideanLatih,
dataKelasLatih, nilaiK);

//load data euclidean data uji dari file rxt dan disimpan dalam
variabel
double[,] euclideanUji =
getEuclFromFile.getEuclideanFromFile("euclideanDataUJi",
jumlahDataUji, jumlahDataLatih);

//menjalankan proses perhitungan weight voting pada method di class
weightVoting
wVoting =
weightVoting.prosesWeightVotingDataMultiple(validitasDataLatih,
euclideanUji);

//menjalankan proses pencarian kelas bagi data uji dari class
cariKelas dan disimpan dalam variabel
string[] klasifikasiResult = cariKelas.getKelasData(dataKelasLatih,
wVoting, nilaiK);

//melakukan proses perhitungan akurasi dan menyimpan kedalam array
//accur[0] = prosentase akurasi, accur[1] = jumlah data benar
double[] accur = akurasiSistem.getAkurasi(dataKelasUji,
klasifikasiResult);

if (accur[0] > nilaiMax)
{
     nilaiMax = accur[0];
     goodAkurasi.Clear();
     goodAkurasi.Add(loop); //nilai k
     goodAkurasi.Add(accur[1]);//jumlah data benar
     goodAkurasi.Add(accur[0]);

//simpan hasil klasifikasi kedalam file txt
for (int a = 0; a < jumlahDataUji; a++)
                    {
hasilKlasi.WriteLine("Data uji ke {0} = {1}",a + 1,
klasifikasiResult[a]);
                    }
                }
otomatis.WriteLine(loop.ToString() + "," + accur[0].ToString() + "%");
            }
goodAkurasi.ToArray();

Properties.Settings.Default.valueNilaiK.Add(goodAkurasi[0].ToString())
;
```

```
//simpan akurasi terbesar beserta nilai k-nya kedalam variable
permanan program
if (Convert.ToDouble(goodAkurasi[2]) >
Properties.Settings.Default.savedAkurasi)
            {
Properties.Settings.Default.savedAkurasi =
Convert.ToDouble(goodAkurasi[2]);

Properties.Settings.Default.savedNilaiK =
Convert.ToDouble(goodAkurasi[0]);
            }

//tampilkan hasil kedalam textField pada program
textBox2.Text = goodAkurasi[0].ToString();
textBox4.Text = jumlahDataUji.ToString();
textBox5.Text = goodAkurasi[1].ToString();
textBox6.Text = goodAkurasi[2].ToString() + " %";
hasilKlasi.Close();
richTextBox1.LoadFile(@"...\Debug\hasilKlasifikasi.txt",
RichTextBoxStreamType.PlainText);
otomatis.Close();

MessageBox.Show("Otomatisasi Selesai!");
        }
```

***Source Code* 4.8** Proses Otomatisasi Nilai K

4.2.4. Akurasi Sistem

Proses perhitungan akurasi dilakukan dengan membandingkan setiap kelas data uji hasil klasifikasi dengan kelas data uji sebenarnya. Untuk mendapatkan presentase jumlah data uji hasil klasifikasi dilakukan perhitungan sesuai dengan persamaan 2.6. Proses perhitungan akurasi sistem ditunjukkan pada kode program 4.9.

Akurasi Sistem

```
class akurasiSistem
    {
public static double[] getAkurasi(string[] dataKelasUji, string[]
hasilUji)
        {
double jumlahBenar=0;
double[] resultAkurasi = new double[2];
double JmlDataUji = dataKelasUji.Length;

//bandingkan kelas hasil klasifikasi data dengan kelas data sebenarnya
for (int indeks = 0; indeks < JmlDataUji; indeks++)
            {
if (dataKelasUji[indeks] == hasilUji[indeks])
 {
//jika sama variabel jumlahBenar akan bertambah 1
   jumlahBenar++;
                }
            }
```

```
//perhitungan akurasi
resultAkurasi[0] = (jumlahBenar / JmlDataUji) * 100;

//hasil jumlah data yang benar
resultAkurasi[1] = jumlahBenar;
return resultAkurasi;
        }
    }
```

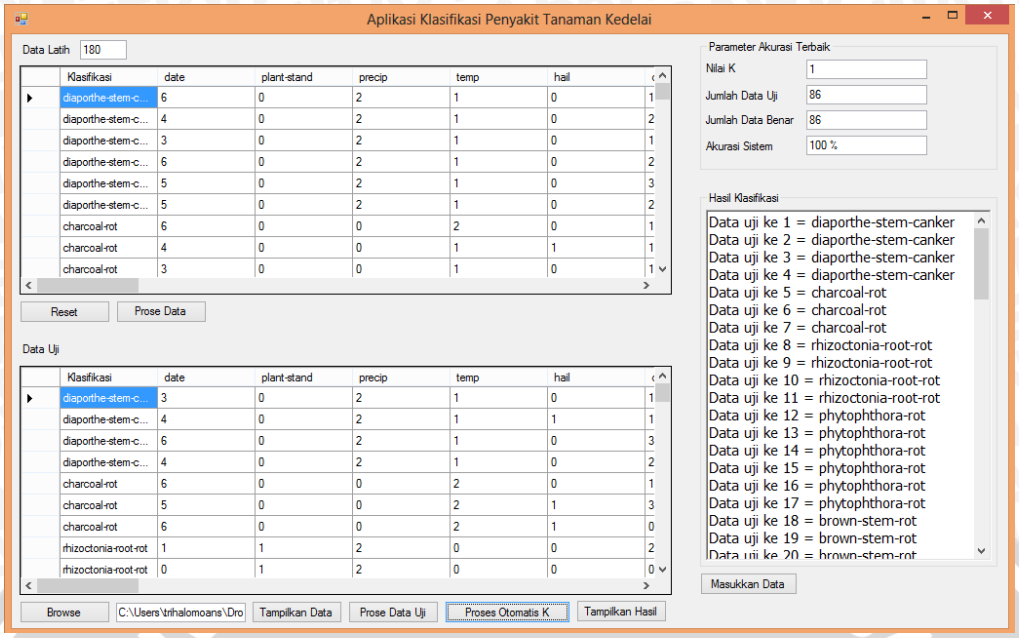
***Source Code* 4.9** Hitung Akurasi Sistem

4.3. Penerapan Aplikasi

Aplikasi yang dibangun untuk melakukan proses klasifikasi pada penyakit tanaman kedelai ini diterapkan berdasarkan pada bab perancangan.

4.3.1. Implementasi Antarmuka

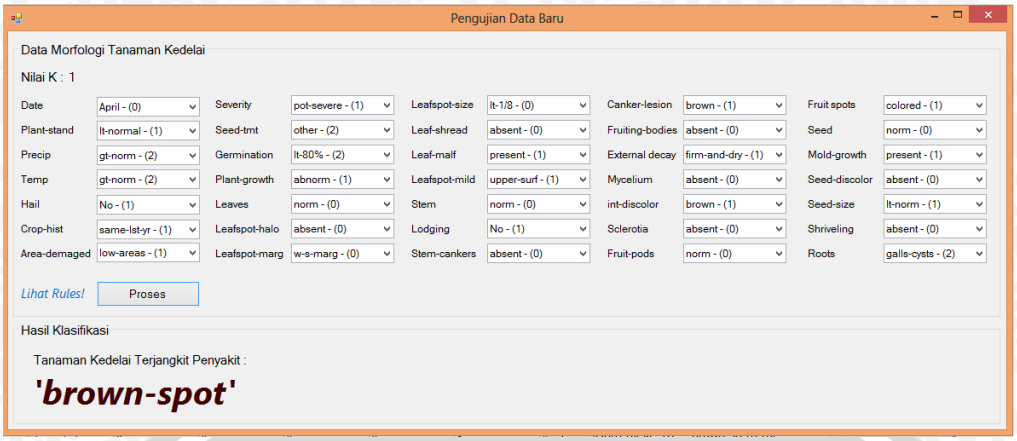
Implementasi antarmuka terdiri dari form yang didalamnya terdapat *field* yang akan menampilkan jumlah data latih, kolom dataGridView untuk menampilkan data latih dan juga data uji, terdapat tombol Proses Data latih akan melakukan normalisasi pada data latih, dan menghitung jarak *euclidean* antar data latih. Terdapat GroupBox berisi *field* yang akan menampilkan nilai k, jumlah data uji yang diproses, jumlah data uji yang dikategorikan benar, dan nilai akurasi. Berikut adalah tampilan dari *form* utama yang ditunjukkan pada Gambar 4.1. Pada *form* tersebut terdapat tombol Browse yang akan menampilkan dialog untuk memilih data uji dalam format .xls. Lalu terdapat tombol Proses Data Uji yang akan melakukan perhitungan jarak *euclidean* untuk setiap data uji terhadap data latih. Tombol Proses Otomarisasi K akan melakukan perhitungan validitas, *weight voting*, klasifikasi kelas data uji, dan perhitungan akurasi perhitungan, dimana nilai k didapatkan secara *brute force.* Untuk setiap prosesnya nilai k dengan akurasi terbaik akan simpan sehingga dapat digunakan sebagai parameter nilai k untuk pengujian data baru.



Gambar 4.1 Implementasi Form Utama

4.3.2. Implementasi Pengujian Data Uji Baru

Pada *form* pengujian data baru terdapat 35 *ComboBox* yang akan mendeskripsikan attribut dari data uji tersebut. Terdapat tombol Proses yang akan melakukan proses input data atribut kedalam sebuah *array*, selanjutnya melakukan proses perhitungan jarak *euclidean* data latih, validitas, perhitungan jarak *euclidean* data uji, perhitungan *weight voting*, dan klasifikasi kelas data uji berdasarkan perhitungan sebelumnya. Hasil klasifikasi akan ditampilkan setelah rangkaian proses berhasil dilakukan. Pada proses pengujian ini, nilai k didapatkan dari perhitungan dengan akurasi terbaik pada proses pelatihan sebelumnya. Berikut adalah tampilan *form* pengujian data baru yang ditunjukkan pada Gambar 4.2.



Gambar 4.2 Implementasi Form Pengujian Data Baru

BAB V
PENGUJIAN DAN ANALISIS

Bab ini akan membahas tentang pengujian dan analisis kinerja algoritma serta perangkat lunak yang dibuat.

5.1. Hasil Uji Coba

Dalam pengujian ini terdapat 266 *dataset* yang nantinya dapat diolah sebagai data latih maupun data uji. Pengujian pertama yang akan dilakukan adalah pengujian variasi nilai K pada Algoritma *Modified K-Nearest Neigbor* selanjutnya pengujian pengaruh parameter k terhadap akurasi sesuai dengan algoritma *Modified K-Nearest Neigbor* dimana nilai k merupakan parameter jumlah tetangga untuk mengklasifikasikan data baru kedalam kelas tertentu, selanjutnya akan dilakukan pengujian data uji tetap dengan jumlah data latih yang berbeda-beda terhadap hasil akurasi, dan yang terakhir adalah pengujian pengaruh jumlah data latih dengan kelas seimbang *(Balanced Class)* dan tidak seimbang *(Imbalanced Class).*

5.1.1. Pengujian Pengaruh Parameter K terhadap Akurasi

Pada pengujian ini diambil 86 data yang akan digunakan sebagai data uji dan 180 data yang akan digunakan sebagai data latih. Data uji diambil secara *random* pada setiap kategori penyakit tanaman kedelai sehingga dapat mewakili setiap kategori penyakit yang ada. Pengujian ini dilakukan untuk mengetahui pengaruh nilai k terhadap hasil akurasi sistem. Dalam pengujian ini data latih terdiri dari beberapa data pada setiap kategori kelas penyakit, dan memiliki jumlah sebaran data yang berbeda-beda pada setiap kelasnya. Dari setiap proses pembelajaran ini sistem akan mendapatkan parameter k terbaik dan akan disimpan sehingga dapat digunakan sebagai parameter k pada pengujian data baru. Hasil pengujian ditunjukkan pada Tabel 5.1 dibawah ini :

Tabel 5.1 Hasil Pengujian Pengaruh Parameter K terhadap Akurasi

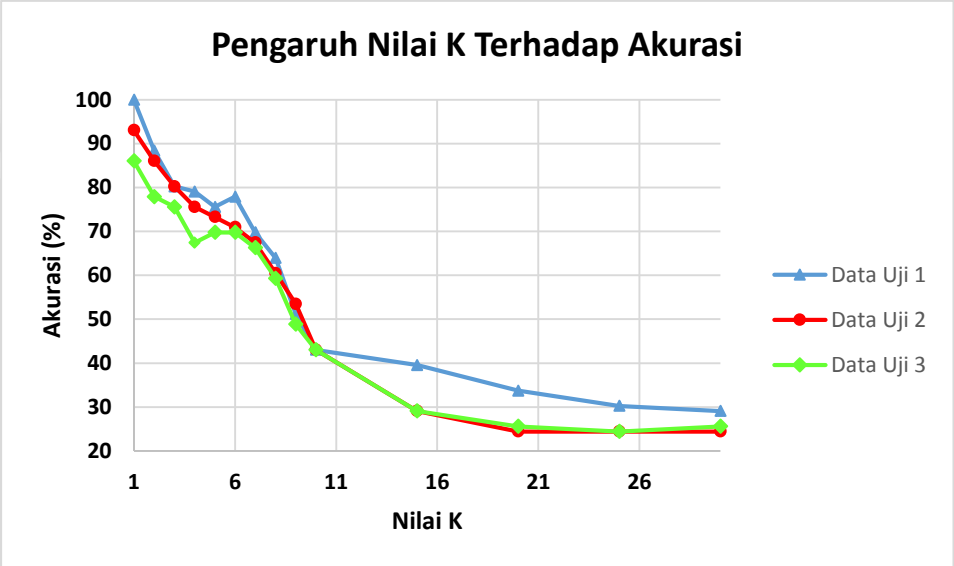
K	Hasil Akurasi (%)		
	Data Uji 1	Data Uji 2	Data Uji 3
1	100,00	93,02	86,05
2	88,37	86,05	77,91

K	Hasil Akurasi (%)		
	Data Uji 1	Data Uji 2	Data Uji 3
3	80,23	80,23	75,58
4	79,07	75,58	67,44
5	75,58	73,26	69,77
6	77,91	70,93	69,77
7	69,77	67,44	66,28
8	63,95	60,47	59,30
9	51,16	53,49	48,84
10	43,02	43,02	43,02
15	39,53	29,07	29,07
20	33,72	24,42	25,58
25	30,23	24,42	24,42
30	29,07	24,42	25,58
35	27,91	25,58	25,58
40	20,93	24,42	24,42
45	15,12	23,26	23,26
50	15,12	22,09	22,09
55	15,12	22,09	22,09
60	15,12	22,09	22,09
65	16,28	22,09	22,09
70	16,28	23,26	23,26
75	24,42	25,58	24,42
80	30,23	27,91	27,91
85	33,72	32,56	29,07
90	36,05	34,88	31,40

Pada pengujian ini didapatkan hasil pengujian dengan akurasi maksimum sebesar 100% pada nilai k=1 dan akurasi minimun sebesar 15,12% pada nilai k=45. Dapat dilihat pada Tabel 5.7 bahwa untuk setiap kenaikan nilai k maka hasil akurasi akan semakin menurun hingga nilai k=45, kemudian hasil akurasi stabil pada saat nilai k=45 hingga k=60 lalu hasil akurasi meningkat pada saat nilai k=65, namun hasil akurasi tidak sebesar pada saat nilai k bernilai kurang dari 10.

Berdasarkan hasil pengujian yang dilakukan terlihat bahwa nilai k sangat berpengaruh terhadap akurasi yang dihasilkan. Namun rata-rata akurasi cenderung menurun seiring dengan penambahan nilai k. Semakin kecil nilai k berarti semakin sedikit jumlah tetangga yang digunakan untuk proses klasifikasi data baru. Metode *euclidean distance* digunakan untuk mencari kedekatan antar data dimana semakin

kecil nilainya maka jarak antar 2 data semakin dekat. Dengan demikian ketika nilai K kecil maka hanya tetangga dengan kedekatan terbaik atau yang memiliki kesamaan karakteristik data terbaik saja yang digunakan untuk proses klasifikasi. Hal ini sebenarnya tergantung dengan bentuk data latih yang digunakan dan bukan tidak mungkin jika akurasi maksimum dapat terjadi saat nilai k > 20 pada suatu pengujian data tertentu. Grafik hasil pengujian ditunjukkan pada Gambar 5.1 dibawah ini:



Gambar 5.1 Grafik Pengaruh Nilai K Terhadap Akurasi

Pada pengujian yang dilakukan nilai akurasi maksimum cenderung terjadi saat nilai K=1. Hal ini disebabkan karena pada data latih rata-rata data antar kelas memiliki selisih yang sangat kecil, dengan demikian dapat diartikan jarak antar kelas pada *dataset* yang digunakan sangat berdekatan. Tabel 5.3 dan 5.4 berisi hasil rata-rata data pada setiap kelas untuk semua fitur yang ada, terlihat bahwa selisih rata-rata data antar kelasnya sangat kecil sekali, semakin kecil rata-rata berarti semakin banyak kesamaan karakteristik antar 2 data tersebut.

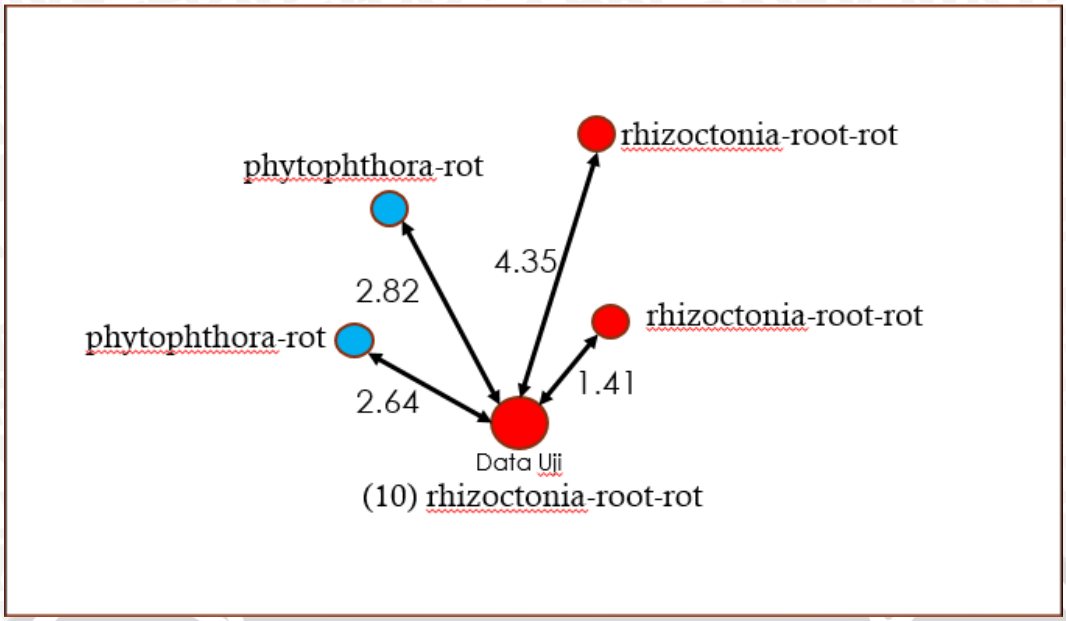
Semakin dekat jarak suatu data dapat diartikan data tersebut hampir memiliki karakteristik yang sama. Sehingga ketika nilai k=1 berarti hanya ada 1 data tetangga terdekat dengan karakteristik data yang hampir sama dengan data uji bahkan tergolong dalam kelas yang sama yang digunakan untuk proses klasifikasi, hal ini memungkinkan untuk melakukan klasifikasi dengan benar. Sebaliknya, jika nilai K semakin besar berarti semakin banyak tetangga yang digunakan untuk proses

klasifikasi termasuk data yang berjauhan dengan data uji dimana hampir tidak memiliki karateristik yang sama hal ini menyebabkan terjadinya *noise* yang semakin besar ditambah lagi dengan adanya dominasi atau frekuensi kelas data latih yang tidak seimbang dari suatu kelas tertentu sehingga hasilnya data cenderung diklasifikasikan pada data kelas yang mendominasi.

Pada *dataset* yang digunakan juga didapati sebaran data yang tidak normal yang menyebabkan kesalahan dalam proses klasifikasi. Pada umumnya jarak antara data dengan kelas yang sama seharusnya lebih dekat dibandingkan dengan jarak antar data dengan kelas yang berbeda, hal ini karena data dengan kelas yang sama umumnya memiliki karakteristik yang hampir sama, sehingga jarak antar 2 data tersebut seharusnya berdekatan. Tabel 5.2 menunjukkan contoh dengan bentuk sebaran data yang tidak normal pada *data set* yang digunakan.

Tabel 5.2 Contoh Sebaran Data Tidak Normal Pada *Data Set*

No	Data	Jarak *Euclidean* Data Ke-10 (rhizoctonia-root-rot)
1	rhizoctonia-root-rot	**1,732050808**
2	rhizoctonia-root-rot	**4,358898944**
3	rhizoctonia-root-rot	**2,236067977**
4	rhizoctonia-root-rot	**2,828427125**
5	rhizoctonia-root-rot	**1,414213562**
6	rhizoctonia-root-rot	**2,236067977**
7	phytophthora-rot	3,16227766
8	phytophthora-rot	**2,645751311**
9	phytophthora-rot	3,31662479
10	phytophthora-rot	3,872983346
11	phytophthora-rot	3,16227766
12	phytophthora-rot	**2,828427125**
13	phytophthora-rot	3,464101615
14	phytophthora-rot	3,464101615
15	phytophthora-rot	3,16227766
16	phytophthora-rot	3,872983346



Gambar 5.2 Sebaran Data Tidak Normal Pada *Data Set*

Gambar 5.2 merupakan contoh beberapa sebaran data pada Tabel 5.2. Dapat dilihat pada Gambar 5.2 terdapat data uji ke-10 dengan kelas *rhizoctonia-root-rot* memiliki jarak sekitar 2,82 dan 2,64 dengan kelas yang berbeda yaitu *phytophthora-rot,* jauh lebih dekat dibandingkan dengan jarak antar kelas yang sama yaitu *rhizoctonia-root-rot* dengan jarak sekitar 4,35. Pada metode *Modified K-Nearest Neighbor* terdapat tahapan untuk mendapatkan tetangga terdekat sejumlah K, hal ini diasumsikan bahwa data terdekat memiliki karakteristik data yang hampir sama bahkan mungkin tergolong dalam kelas sama dengan data uji prediksi dan diharapkan dapat membantu sebagai referensi dalam proses klasifikasi. Namun pada *data set* ini banyak ditemukan jarak antara data dengan kelas yang sama justru lebih jauh dibandingkan dengan kelas yang berbeda, hal inilah yang menyebabkan pada pengujian ini seiring dengan penambahan nilai K maka diikuti penurunan akurasi.

Tabel 5.3 Rata-Rata Sebaran Data Setiap Kelas Pada Fitur 1 Hingga 18

Kelas/Fitur	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
diaporthe-stem-canker	4.83	0.00	2.00	1.00	0.00	1.83	0.17	1.33	0.33	0.83	1.00	1.00	0.00	2.00	2.00	0.00	0.00	0.00
charcoal-rot	4.43	0.00	0.00	1.43	0.43	1.71	2.43	1.00	0.43	0.86	1.00	1.00	0.00	2.00	2.00	0.00	0.00	0.00
rhizoctonia-root-rot	1.67	0.83	2.00	0.00	0.17	1.33	1.00	1.67	0.00	1.33	1.00	0.00	0.00	2.00	2.00	0.00	0.00	0.00
phytophthora-rot	1.20	1.00	1.80	0.50	0.20	1.50	1.00	1.70	0.50	0.80	1.00	1.00	0.00	2.00	2.00	0.00	0.00	0.00
brown-stem-rot	4.17	0.33	0.17	0.67	0.17	2.17	1.67	1.08	0.67	1.08	0.50	0.75	0.33	1.67	1.75	0.00	0.00	0.00
powdery-mildew	4.43	0.57	0.29	0.71	0.43	1.29	1.29	0.71	0.57	1.29	0.00	1.00	0.00	2.00	2.00	0.00	0.00	1.00
downy-mildew	3.57	0.71	2.00	0.71	0.29	2.00	1.00	0.71	0.43	1.43	0.00	1.00	1.57	0.00	1.00	0.00	0.29	2.00
brown-spot	2.20	0.50	1.93	1.10	0.17	2.30	2.27	1.13	0.70	1.07	0.10	1.00	2.00	0.00	1.00	0.43	0.00	0.00
bacterial-blight	3.43	0.14	1.57	1.29	0.57	1.57	1.86	0.57	0.57	0.57	0.00	1.00	1.57	0.00	0.00	0.86	0.00	0.00
bacterial-pustule	2.86	0.43	1.43	0.71	0.43	1.71	1.29	0.43	0.14	1.29	0.29	1.00	1.29	0.86	0.00	0.57	0.14	0.00
purple-seed-stain	4.71	0.00	2.00	0.57	0.43	0.86	0.86	0.00	0.43	1.00	0.00	0.57	1.14	0.86	0.86	0.00	0.00	0.00
anthracnose	4.33	0.50	2.00	1.33	0.33	1.75	1.75	0.83	0.67	1.08	0.42	0.58	0.00	2.00	2.00	0.00	0.00	0.00
phyllosticta-leaf-spot	2.43	0.57	0.57	1.57	0.43	1.71	1.57	0.29	0.71	1.00	0.14	1.00	2.00	0.00	1.00	0.57	0.43	0.00
alternarialeaf-spot	4.81	0.37	1.93	1.41	0.00	1.70	1.74	0.44	0.41	1.11	0.00	1.00	2.00	0.00	1.00	0.19	0.00	0.00
frog-eye-leaf-spot	4.57	0.29	1.82	1.36	0.00	1.86	1.61	0.54	0.64	0.89	0.04	1.00	2.00	0.00	1.00	0.00	0.00	0.00

Tabel 5.4 Rata-Rata Sebaran Data Setiap Kelas Pada Fitur 19 Hingga 35

Kelas/Fitur	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
diaporthe-stem-canker	1.00	0.33	3.00	0.67	1.00	1.00	0.00	0.00	0.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00
charcoal-rot	1.00	0.14	0.00	3.00	0.00	0.00	0.00	2.00	1.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00
rhizoctonia-root-rot	1.00	0.17	1.00	1.00	0.00	1.00	0.17	0.00	0.00	3.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00
phytophthora-rot	1.00	0.10	1.50	2.00	0.00	0.20	0.00	0.00	0.00	3.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00
brown-stem-rot	1.00	0.33	0.00	1.50	0.00	0.00	0.00	1.00	0.00	0.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00
powdery-mildew	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
downy-mildew	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	0.00	0.00	0.00
brown-spot	0.37	0.00	0.70	0.63	0.37	0.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bacterial-blight	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bacterial-pustule	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.43	0.43	0.43	0.43	0.00	0.71
purple-seed-stain	0.57	0.29	0.00	3.00	0.00	0.00	0.00	0.00	0.00	0.43	0.43	1.00	0.00	1.00	0.00	0.00	0.00
anthracnose	1.00	0.08	2.83	1.75	0.67	0.58	0.00	0.00	0.00	0.83	1.67	0.58	0.58	0.25	0.33	0.33	0.00
phyllosticta-leaf-spot	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
alternarialeaf-spot	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
frog-eye-leaf-spot	0.54	0.00	1.61	0.86	0.07	0.50	0.00	0.00	0.00	0.54	0.57	0.04	0.00	0.00	0.00	0.00	0.00

Pada subbab 5.1.1 akurasi terbaik selalu terjadi pada saat nilai K=1 hal ini sebenarnya tergantung dari bentuk data latih yang digunakan. Oleh sebab itu akan dilakukan pengujian dimana *data set* yang digunakan tidak berasal dari *data set* penyakit tanaman kedelai seperti pada pengujian sebelumnya, hal ini dilakukan unttuk membuktikan pengaruh dan perbedaan bentuk data terhadap hasil klasifikasi dan pola akurasi yang dihasilkan. Jika pada subbab 5.1.1 didapati bahwa bentuk *data set* sangat rapat artinya jarak antar data dengan data lainnya sangat berdekatan, maka pada pengujian ini akan dikelola *data set* dengan sebaran data yang normal dimana jarak antara data tidak terlalu berdekatan. Data latih dan data uji ditunjukkan pada Tabel 5.5 dan 5.6.

Tabel 5.5 Sample Data Latih

Penyakit	f1	f2
A	20	10
A	25	12
B	24	9
B	29	8
C	15	9
C	16	11

Tabel 5.6 Sample Data Uji

Penyakit	f1	f2
A	22	11
A	26	14
B	25	8
B	28	9
C	14	10
C	17	9

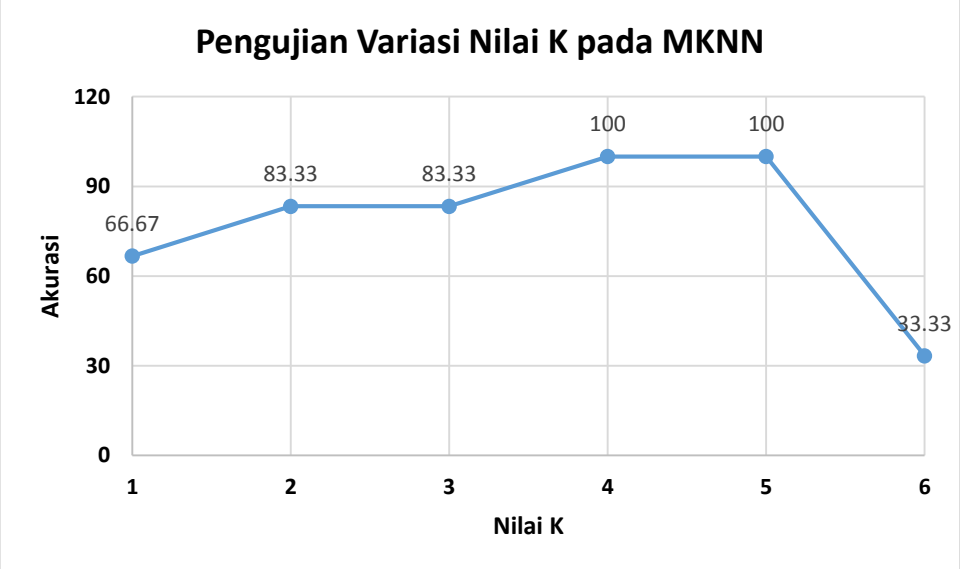
Pada pengujian ini terdapat *dataset* dengan 3 kategori kelas data yaitu kelas A, kelas B, dan kelas C. Hasil pengujian ditunjukkan pada Tabel 5.7.

Tabel 5.7 Hasil Pengujian

K	Akurasi (%)
1	66.67
2	83.33
3	83.33

K	Akurasi (%)
4	100.00
5	100.00
6	33.33

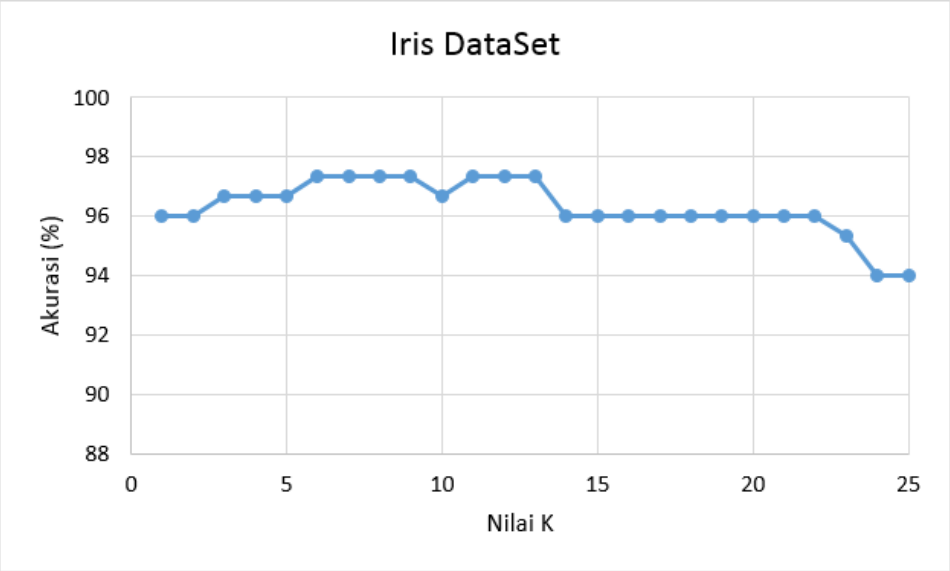
Dapat dilihat pada Tabel 5.7 pada saat K=1 akurasi sebesar 66.67% seiring dengan penambahan nilai K maka akurasi meningkat, kemudian pada puncak akurasi tertentu untuk selanjutnya akurasi akan menurun seiring dengan penambahan nilai K. Pada metode *Modified K-Nearest Neighbor* hasil akurasi memang tergantung dari bentuk data dan nilai K yang digunakan. Pada sebaran data yang normal umumnya ketika K=1 akurasi berada pada tingkat tertentu, seiring dengan penambahan nilai k maka tingkat akurasi meningkat hingga pada nilai akurasi tertentu kemudian akurasi akan menurun (seperti yang ditunjukkan Gambar 5.3) dan bukan tidak memungkinkan jika terjadi peningkatan akurasi lagi.



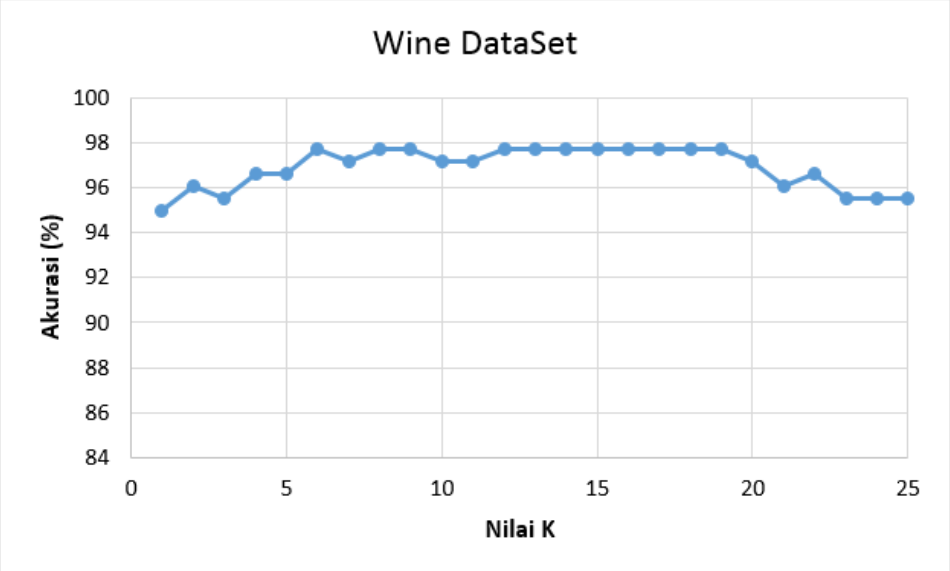
Gambar 5.3 Grafik Variasi Nilai K pada MKNN

Berdasarkan pengujian yang dilakukan pada umumnya akurasi terbaik pada metode *Modified K-Nearest Neighbor* tidak terjadi saat nilai K=1 (seperti yang ditunjukkan Gambar 5.3). Pola yang sama juga ditunjukkan pada pengujian

menggunakan *Iris Data Set* dan *Wine Data Set*. Grafik hasil pengujian ditunjukkan pada gambar dibawah ini :



Gambar 5.4 Hasil Pengujian Pada *Iris Data Set*



Gambar 5.5 Hasil Pengujian Pada *Wine Data Set*

Pada Gambar 5.4 dapat dilihat pada pengujian menggunakan *Iris Data Set*, akurasi terbaik tidak terjadi saat nilai K=1 melainkan K=6, dan pada pengujian menggunakan *Wine Data Set* akurasi terbaik terjadi saat nilai K=9 yang ditunjukkan pada Gambar 5.5. Hal ini terjadi karena pada *data set* memiliki sebaran data normal dimana rata-rata data antar kelas memiliki selisih yang tidak terlalu kecil, karena

jika memiliki selisih yang kecil berarti jarak data antar 2 kelas sangat dekat, bahkan hampir memiliki karateristik yang sama. Rata-rata data pada tiap-tiap kelas dapat dilihat pada Tabel 5.8.

Tabel 5.8 Rata-Rata Sebaran Data Setiap Kelas

Kelas	Rata-rata data	
	f1	f2
A	22.5	11
B	26.5	8.5
C	15.2	10

Berdasarkan Tabel 5.9 dan Tabel 5.10 dapat dilihat selisih sebaran data antar kelas pada setiap fitur cukup besar dibandingkan selisih rata-rata pada subbab 5.1.1 yang ditunjukkan pada Tabel 5.3. Semakin besar selisih rata-rata maka semakin jauh kesamaan karakteristik antar data tersebut. Karena jarak antar data pada pengujian ini tidak terlalu dekat sehingga ketika nilai K=1, maka belum mendapatkan referensi data yang cukup untuk melakukan klasifikasi dengan benar, terbukti ketika nilai K=5 terdapat banyak data yang digunakan untuk referensi dalam klasifikasi, hal ini yang memungkinkan untuk melakukan klasifikasi dengan benar.

Tabel 5.9 Perbandingan Selisih Rata-Rata Setiap Kelas Pada Fitur 1

Kelas	Fitur 1 (f1)		
	A	B	C
A	0	4	7,3
B	4	0	11,3
C	7,3	11,3	0

Tabel 5.10 Perbandingan Selisih Rata-Rata Setiap Kelas Pada Fitur 2

Kelas	Fitur 2 (f2)		
	A	B	C
A	0	2,5	1
B	2,5	0	1,5
C	1	1,5	0

5.1.2. Pengujian Pengaruh Jumlah Data Uji Tetap dengan Jumlah Data Latih Berbeda

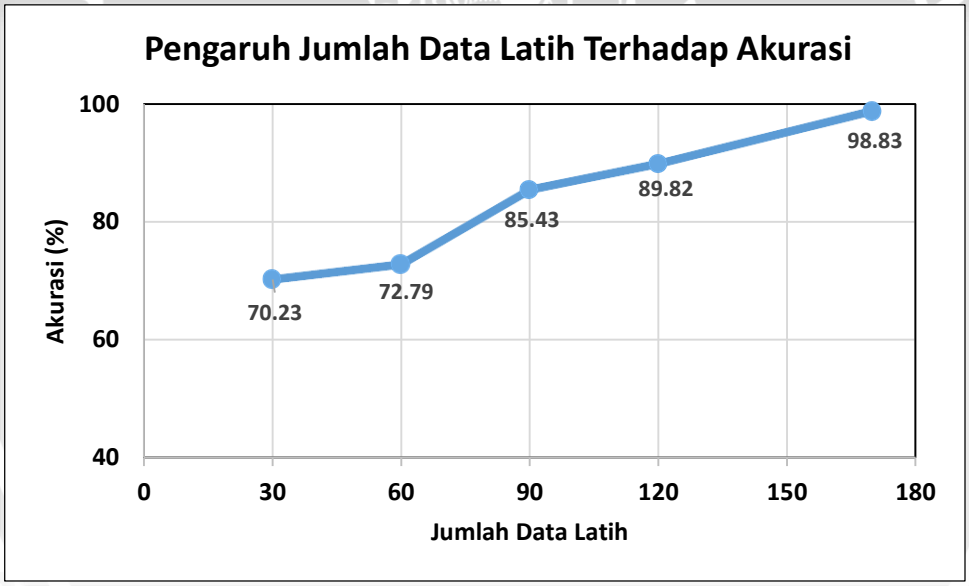
Pada pengujian ini data uji yang digunakan pada setiap percobaan sama, terdapat data uji tetap sebanyak 86 data yang tidak berubah sedangkan data latih bertambah besar untuk setiap percobaan yang dilakukan. Dalam pengujian ini terdapat 5 kali percobaan untuk setiap jumlah data latih yang sama, data latih diambil secara *random* untuk setiap percobaan dimasing-masing kelas sehingga data latih tidak akan sama pada setiap percobaan yang dilakukan dan dapat mewakili setiap kelas data yang ada.

Tabel 5.11 Hasil Pengujian Jumlah Data uji Tetap dengan Jumlah Data Latih Berbeda

Jumlah Data	Akurasi (%)					Rata-rata Akurasi (%)
	Percobaan 1	Percobaan 2	Percobaan 3	Percobaan 4	Percobaan 5	
30	68,60	62,79	72,09	75,58	72,09	70,23
60	75,58	63,95	75,58	77,91	70,93	72,79
90	86,05	81,40	89,53	83,72	86,05	85,43
120	91,86	87,21	89,53	87,21	93,33	89,82
170	97.67	96.51	100	100	100	98,83

Pada pengujian ini didapatkan hasil dengan akurasi maksimum sebesar 100% pada pengujian dengan jumlah data latih sebanyak 170 pada percobaan 3, 4, dan 5, dan akurasi minimum sebesar 62,79% pada pengujian dengan jumlah data latih sebanyak 30 pada percobaan 2. Berdasarkan pengujian yang dilakukan, terjadi peningkatan akurasi seiring dengan bertambahnya jumlah data latih yang digunakan. Pengujian dengan jumlah data latih sebanyak 30 menghasilkan rata-rata akurasi sebesar 70,23%, pengujian dengan jumlah data latih 60 menghasilkan rata-rata akurasi sebesar 72,79%, terjadi peningkatan 2,56% dari pengujian dengan jumlah data latih sebanyak 30. Rata-rata akurasi terbaik terjadi pada pengujian dengan jumlah data latih sebanyak 170 dengan hasil 98,83%. Untuk hasil pengujian jumlah data uji tetap dengan jumlah data latih berbeda dapat dilihat pada Tabel 5.11.

Berdasarkan hasil uji coba yang telah dilakukan, terlihat bahwa jumlah data latih sangat berpengaruh pada besar nilai akurasi yang dihasilkan terlebih jika data latih dengan kelas data seimbang. Pada pengujian ini akurasi yang diambil merupakan akurasi terbaik sesuai dengan parameter k yang sudah ditentukan oleh aplikasi. Berdasarkan hasil uji coba peningkatan jumlah data latih turut disertai dengan peningkatan hasil akurasi. Hal ini dipengaruhi dikarenakan semakin banyak data latih yang digunakan maka semakin banyak data yang dibandingkan dan akan berpengaruh pada hasil akurasi. Dalam pengujian ini data latih dalam keadaan kelas data seimbang *(Balanced Class)* artinya tidak ada kelas yang lebih mendominasi. Karena data latih yang tidak seimbang menimbulkan *noise* dalam penentuan keputusan. Grafik Pengaruh Data Uji Tetap dengan Jumlah Data Latih Berbeda ditunjukkan pada Gambar 5.6.



Gambar 5.6 Grafik Pengaruh Data Latih Terhadap Akurasi

5.1.3. Pengujian Data Latih dengan Kelas Seimbang dan Tidak Seimbang

Pengujian ini dilakukan untuk mengetahui pengaruh jumlah data latih dengan kelas seimbang dan tidak seimbang terhadap akurasi yang dihasilkan. Data uji disiapkan dengan mengambil 3 data dari masing-masing kelas sehingga terkumpul sebanyak 45 data uji dengan data latih yang semakin bertambah besar yaitu 60, 90, dan 120. Pada pengujian data latih dengan kelas tidak seimbang, terdapat data yang

mendominasi jumlah yang lebih banyak dibandingkan dengan kelas lain. Sebagai contoh pada pengujian ini untuk data latih dengan kelas tidak seimbang berjumlah 120 data dengan kelas *diaporthe-stem-canker* sebanyak 2 data, *charcoal-rot* sebanyak 2 data, *rhizoctonia-root-rot* sebanyak 2 data, *phytophthora-rot* sebanyak 2 data, *brown-stem-rot* sebanyak 3 data, *powdery-mildews* sebanyak 3 data, *downy-mildew* sabanyak 3 data, *brown-spot* sebanyak 4 data, *bacterial-blight* sebanyak 3 data, *bacterial-pustule* sebanyak 2 data, *purple-seed-stain* sebanyak 2 data, *anthracnose* 1 data, *phyllosticta-leaf-spot* sebanyak 7 data, *alternarialeaf-spot* sebanyak 37 data, dan *frog-eye-leaf-spot* sebanyak 37 data.

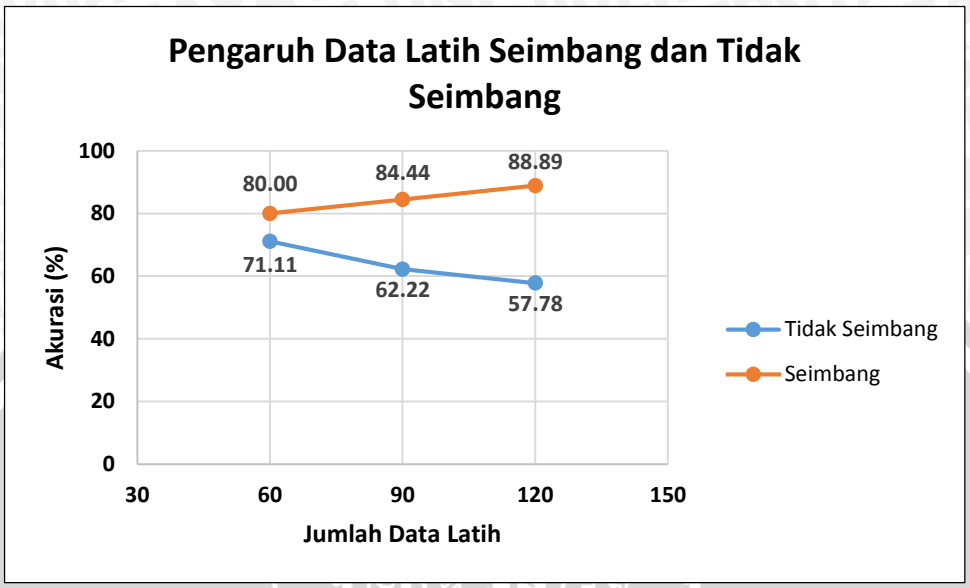
Tabel 5.12 Hasil Pengujian Data Latih dengan Kelas Seimbang dan Tidak Seimbang

Jenis Data	Jumlah Data Latih	Rata-rata Akurasi (%)
Tidak Seimbang	60	71,11
	90	62,22
	120	57,78
Seimbang	60	80
	90	84,44
	120	88,89

Pada pengujian ini didapatkan akurasi maksimum sebesar 88,89% pada data latih dengan kelas seimbang menggunakan 120 data, dan akurasi menurun seiring dengan berkurangnya jumlah data latih tersebut, sedangkan akurasi minimum terjadi pada pengujian data latih dengan kelas tidak seimbang yaitu sebesar 57,78% menggunakan 120 data, dan akurasi semakin meningkat seiring dengan berkurangnya data latih. Untuk hasil pengujian data latih dengan kelas seimbang dan tidak seimbang ditunjukkan pada Tabel 5.12.

Berdasarkan hasil pengujian, untuk data latih dengan kelas tidak seimbang mengalami penurunan nilai akurasi seiring dengan bertambahnya jumlah data. Hal sebaliknya terjadi pada data latih dengan kelas seimbang, peningkatan akurasi terjadi seiring dengan bertambahnya jumlah data latih. Pada saat data latih tidak seimbang, terjadi dominasi data pada kelas tertentu sehingga menimbulkan *noise* yang menyebabkan terjadinya kesalahan dalam mengklasifikasikan data dan

cenderung mengacu pada kelas yang mendominasi tersebut. Grafik Pengaruh Data Latih dengan Kelas Seimbang dan Tidak Seimbang ditunjukkan pada Gambar 5.7.



Gambar 5.7 Grafik Pengaruh Data Latih Seimbang dan Tidak Seimbang

BAB VI
PENUTUP

6.1. Kesimpulan

Berdasarkan penelitian yang telah dilakukan, maka didapatkan kesimpulan sebagai berikut :

1. Metode *Modified K-Nearest Neighbor* dapat diimplementasikan untuk klasifikasi *dataset* penyakit pada tanaman kedelai. Serangkain percobaan menunjukkan tingkat akurasi yang tinggi.
2. Rata-rata akurasi maksimum yang dihasilkan pada penelitian ini sebesar 98,83% pada saat jumlah data latih 170 data dan akurasi minimum sebesar 70,23% pada saat jumlah data latih 30 data. Tingkat akurasi yang dihasilkan pada metode MKNN dipengaruhi oleh beberapa parameter sebagai berikut:
    a. Penambahan ataupun pengurangan nilai k berpengaruh pada tingkat akurasi. Nilai k yang terbaik pada metode ini tergantung pada data yang ada. Secara umum penambahan nilai k akan menyebabkan penurunan akurasi, hal ini disebabkan semakin banyaknya data yang digunakan untuk proses klasifikasi, terlebih jika adanya kelas data tertentu yang mendominasi pada pemilihan data tetangga terdekat sejumlah k.
    b. Peningkatan jumlah data latih turut disertai dengan peningkatan akurasi terutama jika data latih memiliki kelas data yang seimbang. Hal ini disebabkan semakin banyaknya data latih maka kemungkinan semakin banyak data referensi yang mendekati kelas data prediksi.
    c. Untuk data latih dengan kelas tidak seimbang mengalami penurunan nilai akurasi seiring dengan bertambahnya jumlah data. Hal sebaliknya terjadi pada data latih dengan kelas seimbang, peningkatan akurasi terjadi seiring dengan bertambahnya jumlah data latih. Hal ini terjadi karena data latih dengan kelas tidak seimbang menimbulkan *noise* terutamaa pada k yang bernilai besar,

sehingga klasifikasi cenderung mengacu pada kelas yang mendominasi dalam data tersebut.

3. Metode *Brute Force* dapat diimplementasikan untuk mendapatkan nilai k terbaik berdasarkan akurasi terbesar dari suatu proses percobaan pendahulu, sehingga tidak perlu memasukkan parameter nilai k secara manual untuk suatu pengujian data.

6.2. Saran

Berkaitan dengan penelitian ini, penulis menemukan beberapa hal yang mungkin perlu untuk pengembangan selanjutnya, yaitu :

1. Diharapkan pada data latih menggunakan variasi data yang lebih beragam dan memiliki sebaran data yang seimbang secara keseluruhan.
2. Secara fungsional sistem ini dikembangkan agar dapat digunakan secara langsung oleh petani dengan mudah, melalui parameter-parameter yang mudah diketahui oleh petani.

DAFTAR PUSTAKA

[MWT-07] Marwoto. 2007. *Dukungan Pengendalian Hama Terpadu dalam Program Bangkit Kedelai.* Balai Penelitian Tanaman Kacang-kacangan dan Umbi-umbian. Malang.

[YCH-10] Yu, Chen. 2010. *How KNN works ?*. Indiana University. USA.

[HAM-10] Hamid Parvin, Hoseinali and Behrouz Minati. 2010. *Modification on K-Nearest Neighbor Classification.* Global Journal of Computer Science and Technology Vol.10 Issue 14 (Ver.1.0).

[HAM-08] Hamid Parvin, Hosein Alizadeh and Behrouz Minae-Bidgoli. 2008. *MKNN: Modification on K-Nearest Neighbor Classification.* San Francisco. USA.

[SUL-12] Sulistyandari. 2012. *Penerapan Algoritma Modified K-Neares Neighbor (MKNN) untuk Mengklasifikasikan Letak Protein pada Bacteri E-Coli.* Skripsi. Universitas Brawijaya Malang.

[HAN-06] Han, Jiawei, Micheline Kamber. 2006. *Data Mining Concepts and Techniques Second Edition.* Morgan Kaufmann Publishers.

[DJW-87] Djoko Widodo. 1987. *Hama dan Penyakit Kedelai.* Penerbit Pustaka Buana. Bandung.

[WAW-06] Wawan Irwan, Aep. 2006. *Budidaya Tanaman Kedelai*. Universitas Padjadjaran Bandung.

[LAR-05] Larose, D.T. 2005. Discovering Knowledge in Data: An Introduction to Data Minig. Wiley, Chichester.

[BAM-99] Bambang, Budi DP. dkk. 1999. Teknik Jaringan Syaraf Tiruan Feedforward Untuk Prediksi Harga Saham Pada Pasar Modal Indonesia. Jurnal Indonesia Vol. 1, No. 1, Mei 1999:11-22

[AKA-12] Andi Khaeruni, Abdul Rahman. 2012. *Penggunaan Bakteri Kitinolitik sebagai Agens Biokontrol Penyakit Busuk Batang Rhizoctonia solani pada Tanaman Kedelai.* Universitas Haluoleo. Kendari.

[CRG-06] Craig Grau. 2006. *Stem Canker of Soybean*. Dept. Of Plant Pathology. University of WI-Madison.

[MAR-11] Marwoto. 2011. *Masalah Hama, Penyakit, dan Hara pada Tanaman Kedelai (Identifikasi dan Pengendaliannya).* Badan Penelitian dan Pengembangan Pertanian. Dipa Balitkabi.

[ADD-09] Anne E. Dorrance and Dennis R. Mills. 2009. *Charcoal Rot of Soybeans.* Department of Plant Pathology. The Ohio State University.

[TKE-11] T.Chandrasekhar, K.Thangavel and E.Elayaraja. 2011. *Effective Clustering Algorithms for Gene Expression Data. Department of Computer Science.* Periyar University. Tamil Nadu, India.

LAMPIRAN

Lampiran 1. *Data Set* Penyakit Tanaman Kedelai

Kelas Peyakit	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	F31	F32	F33	F34	F35
diaporthe-stem-canker	6	0	2	1	0	1	1	1	0	0	1	1	0	2	2	0	0	0	1	1	3	1	1	1	0	0	0	0	4	0	0	0	0	0	0
diaporthe-stem-canker	4	0	2	1	0	2	0	2	1	1	1	1	0	2	2	0	0	0	1	0	3	1	1	1	0	0	0	0	4	0	0	0	0	0	0
diaporthe-stem-canker	3	0	2	1	0	1	0	2	1	2	1	1	0	2	2	0	0	0	1	0	3	0	1	1	0	0	0	0	4	0	0	0	0	0	0
diaporthe-stem-canker	3	0	2	1	0	1	0	2	0	1	1	1	0	2	2	0	0	0	1	0	3	0	1	1	0	0	0	0	4	0	0	0	0	0	0
diaporthe-stem-canker	6	0	2	1	0	2	0	1	0	2	1	1	0	2	2	0	0	0	1	0	3	1	1	1	0	0	0	0	4	0	0	0	0	0	0
diaporthe-stem-canker	5	0	2	1	0	3	0	1	0	1	1	1	0	2	2	0	0	0	1	0	3	0	1	1	0	0	0	0	4	0	0	0	0	0	0
diaporthe-stem-canker	5	0	2	1	0	2	0	1	1	0	1	1	0	2	2	0	0	0	1	1	3	1	1	1	0	0	0	0	4	0	0	0	0	0	0
diaporthe-stem-canker	4	0	2	1	1	1	0	1	0	2	1	1	0	2	2	0	0	0	1	0	3	1	1	1	0	0	0	0	4	0	0	0	0	0	0
diaporthe-stem-canker	6	0	2	1	0	3	0	1	1	1	1	1	0	2	2	0	0	0	1	0	3	1	1	1	0	0	0	0	4	0	0	0	0	0	0
diaporthe-stem-canker	4	0	2	1	0	2	0	2	0	2	1	1	0	2	2	0	0	0	1	0	3	1	1	1	0	0	0	0	4	0	0	0	0	0	0
charcoal-rot	6	0	0	2	0	1	3	1	1	0	1	1	0	2	2	0	0	0	1	0	0	3	0	0	0	2	1	0	4	0	0	0	0	0	0
charcoal-rot	4	0	0	1	1	1	3	1	1	1	1	1	0	2	2	0	0	0	1	1	0	3	0	0	0	2	1	0	4	0	0	0	0	0	0
charcoal-rot	3	0	0	1	0	1	2	1	0	0	1	1	0	2	2	0	0	0	1	0	0	3	0	0	0	2	1	0	4	0	0	0	0	0	0
charcoal-rot	6	0	0	1	1	3	3	1	1	0	1	1	0	2	2	0	0	0	1	0	0	3	0	0	0	2	1	0	4	0	0	0	0	0	0
charcoal-rot	6	0	0	2	0	1	3	1	1	1	1	1	0	2	2	0	0	0	1	0	0	3	0	0	0	2	1	0	4	0	0	0	0	0	0
charcoal-rot	5	0	0	2	1	3	3	1	1	2	1	1	0	2	2	0	0	0	1	0	0	3	0	0	0	2	1	0	4	0	0	0	0	0	0
charcoal-rot	6	0	0	2	1	0	2	1	0	0	1	1	0	2	2	0	0	0	1	1	0	3	0	0	0	2	1	0	4	0	0	0	0	0	0
charcoal-rot	4	0	0	1	0	2	2	1	0	1	1	1	0	2	2	0	0	0	1	0	0	3	0	0	0	2	1	0	4	0	0	0	0	0	0
charcoal-rot	3	0	0	2	0	2	2	1	0	2	1	1	0	2	2	0	0	0	1	0	0	3	0	0	0	2	1	0	4	0	0	0	0	0	0
charcoal-rot	5	0	0	2	1	2	2	1	0	2	1	1	0	2	2	0	0	0	1	0	0	3	0	0	0	2	1	0	4	0	0	0	0	0	0
rhizoctonia-root-rot	1	1	2	0	0	2	1	2	0	2	1	0	0	2	2	0	0	0	1	0	1	1	0	1	1	0	0	3	4	0	0	0	0	0	0
rhizoctonia-root-rot	1	1	2	0	0	1	1	2	0	1	1	0	0	2	2	0	0	0	1	0	1	1	0	1	0	0	0	3	4	0	0	0	0	0	0
rhizoctonia-root-rot	3	0	2	0	1	3	1	2	0	1	1	0	0	2	2	0	0	0	1	1	1	1	0	1	1	0	0	3	4	0	0	0	0	0	0
rhizoctonia-root-rot	0	1	2	0	0	0	1	1	1	2	1	0	0	2	2	0	0	0	1	0	1	1	0	1	0	0	0	3	4	0	0	0	0	0	0
rhizoctonia-root-rot	0	1	2	0	0	1	1	2	1	2	1	0	0	2	2	0	0	0	1	0	1	1	0	1	0	0	0	3	4	0	0	0	0	0	0
rhizoctonia-root-rot	1	1	2	0	0	3	1	2	0	2	1	0	0	2	2	0	0	0	1	0	1	1	0	1	0	0	0	3	4	0	0	0	0	0	0
rhizoctonia-root-rot	1	1	2	0	0	0	1	1	0	1	1	0	0	2	2	0	0	0	1	0	1	1	0	1	0	0	0	3	4	0	0	0	0	0	0
rhizoctonia-root-rot	2	1	2	0	0	2	1	1	0	1	1	0	0	2	2	0	0	0	1	0	1	1	0	1	0	0	0	3	4	0	0	0	0	0	0
rhizoctonia-root-rot	1	1	2	0	0	1	1	2	0	2	1	0	0	2	2	0	0	0	1	0	1	1	0	1	0	0	0	3	4	0	0	0	0	0	0
rhizoctonia-root-rot	2	1	2	0	0	1	1	2	0	2	1	0	0	2	2	0	0	0	1	0	1	1	0	1	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	0	1	2	1	1	1	1	1	0	0	1	1	0	2	2	0	0	0	1	0	1	2	0	1	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	1	1	2	0	0	2	1	2	1	1	1	1	0	2	2	0	0	0	1	0	2	2	0	0	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	0	1	1	1	0	1	1	1	0	0	1	1	0	2	2	0	0	0	1	0	1	2	0	0	0	0	0	3	4	0	0	0	0	0	0

Kelas Peyakit	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	F31	F32	F33	F34	F35
phytophthora-rot	3	1	2	0	0	2	1	2	1	1	1	1	0	2	2	0	0	0	1	0	2	2	0	0	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	2	1	2	0	0	1	1	2	0	1	1	1	0	2	2	0	0	0	1	0	1	2	0	0	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	0	1	2	1	0	3	1	1	0	0	1	1	0	2	2	0	0	0	1	0	1	2	0	0	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	0	1	1	1	1	2	1	2	1	0	1	1	0	2	2	0	0	0	1	1	2	2	0	1	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	3	1	2	0	0	1	1	2	1	0	1	1	0	2	2	0	0	0	1	0	2	2	0	0	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	0	1	2	1	0	2	1	1	0	1	1	1	0	2	2	0	0	0	1	0	1	2	0	0	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	2	1	2	1	1	1	1	2	0	2	1	1	0	2	2	0	0	0	1	0	1	2	0	1	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	0	1	2	1	0	3	1	1	0	2	1	1	0	2	2	0	0	0	1	0	1	2	0	0	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	1	1	2	1	0	0	1	2	1	1	1	1	0	2	2	0	0	0	1	0	2	2	0	0	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	1	1	2	1	1	3	1	2	0	1	1	1	0	2	2	0	0	0	1	1	1	2	0	1	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	2	1	2	0	0	1	1	2	0	0	1	1	0	2	2	0	0	0	1	0	1	2	0	0	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	1	1	2	0	0	0	1	2	1	0	1	1	0	2	2	0	0	0	1	0	2	2	0	0	0	0	0	3	4	0	0	0	0	0	0
phytophthora-rot	3	1	1	0	0	2	1	2	1	2	1	1	0	2	2	0	0	0	1	0	2	2	0	0	0	0	0	3	4	0	0	0	0	0	0
brown-stem-rot	4	0	0	1	0	1	3	1	1	2	0	1	0	2	2	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
brown-stem-rot	4	0	0	1	0	1	3	1	1	2	0	1	0	2	2	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
brown-stem-rot	3	1	0	0	0	3	0	1	1	2	1	0	0	2	2	0	0	0	1	0	0	3	0	0	0	1	0	0	4	0	0	0	0	0	0
brown-stem-rot	5	0	0	2	0	1	3	1	1	2	0	1	0	2	2	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
brown-stem-rot	5	0	0	2	0	2	3	1	1	1	0	1	0	2	2	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
brown-stem-rot	4	0	0	1	0	3	2	1	0	1	0	1	0	2	2	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
brown-stem-rot	5	0	0	1	1	3	3	1	0	2	1	1	2	0	1	0	0	0	1	1	0	3	0	0	0	1	0	0	4	0	0	0	0	0	0
brown-stem-rot	6	0	1	1	1	2	0	1	1	0	1	0	0	2	1	0	0	0	1	1	0	3	0	0	0	1	0	0	4	0	0	0	0	0	0
brown-stem-rot	6	0	1	1	1	2	0	1	1	0	1	0	0	2	1	0	0	0	1	1	0	3	0	0	0	1	0	0	4	0	0	0	0	0	0
brown-stem-rot	5	1	0	0	0	3	2	1	0	0	1	1	0	2	2	0	0	0	1	1	0	3	0	0	0	1	0	0	4	0	0	0	0	0	0
brown-stem-rot	5	1	0	1	0	1	3	1	1	0	1	1	2	0	1	0	0	0	1	1	0	3	0	0	0	1	0	0	4	0	0	0	0	0	0
brown-stem-rot	4	0	1	0	1	2	3	1	1	2	1	0	0	2	2	0	0	0	1	1	0	3	0	0	0	1	0	0	4	0	0	0	0	0	0
brown-stem-rot	4	1	0	0	0	3	2	1	1	1	1	1	2	0	1	0	0	0	1	1	0	3	0	0	0	1	0	0	4	0	0	0	0	0	0
brown-stem-rot	4	0	0	1	0	2	0	1	1	0	0	1	0	2	2	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
brown-stem-rot	3	1	0	0	0	2	0	2	0	1	1	1	2	0	1	0	0	0	1	0	0	3	0	0	0	1	0	0	4	0	0	0	0	0	0
brown-stem-rot	5	0	0	1	0	3	2	1	0	1	0	1	0	2	2	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
brown-stem-rot	4	0	0	1	0	3	3	1	1	0	0	1	0	2	2	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
brown-stem-rot	4	0	0	1	0	3	2	1	0	2	0	1	0	2	2	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
brown-stem-rot	4	0	0	1	0	1	2	1	1	2	0	1	0	2	2	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
brown-stem-rot	4	0	0	1	0	1	2	1	0	0	0	1	0	2	2	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
brown-stem-rot	3	0	0	1	0	3	2	1	0	0	0	1	0	2	2	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
powdery-mildew	5	0	0	1	1	3	3	1	0	1	0	1	0	2	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
powdery-mildew	6	0	1	0	1	0	0	0	1	2	0	1	0	2	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
powdery-mildew	1	1	0	1	0	3	3	1	2	0	0	1	0	2	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
powdery-mildew	6	1	1	0	0	2	2	0	1	2	0	1	0	2	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
powdery-mildew	4	1	1	0	0	2	2	0	2	0	0	1	0	2	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
powdery-mildew	6	0	0	1	1	1	1	1	0	2	0	1	0	2	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
powdery-mildew	2	1	1	0	0	2	2	0	0	1	0	1	0	2	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
powdery-mildew	6	1	0	1	0	1	1	1	1	2	0	1	0	2	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Kelas Peyakit	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	F31	F32	F33	F34	F35
powdery-mildew	5	1	0	1	0	1	1	1	0	1	0	1	0	2	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
powdery-mildew	1	1	0	1	0	1	1	1	2	0	0	1	0	2	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
downy-mildew	6	0	2	0	1	2	1	0	1	2	0	1	2	0	1	0	1	2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
downy-mildew	2	0	2	1	1	1	1	1	1	2	0	1	2	0	1	0	1	2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
downy-mildew	1	0	2	1	1	3	2	1	0	1	0	1	1	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
downy-mildew	4	1	2	2	0	2	2	1	0	1	0	1	1	0	1	0	1	2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
downy-mildew	1	0	2	0	1	0	0	1	0	1	0	1	1	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
downy-mildew	2	1	2	0	0	3	0	1	0	1	0	1	1	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
downy-mildew	2	1	2	1	0	2	0	1	0	1	0	1	2	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
downy-mildew	4	1	2	2	0	2	1	0	1	2	0	1	1	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
downy-mildew	4	1	2	0	0	1	2	1	0	1	0	1	2	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
downy-mildew	5	1	2	1	0	3	2	1	0	1	0	1	1	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
brown-spot	1	1	2	2	1	3	3	1	0	2	1	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	2	0	2	1	0	2	3	1	1	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	2	0	2	1	0	2	3	1	1	0	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	2	0	2	1	0	1	0	1	2	0	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	1	1	2	2	1	3	3	1	1	1	1	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	1	1	2	1	0	2	3	1	0	2	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	0	1	2	2	1	3	3	1	2	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	2	0	2	1	0	2	3	1	0	0	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	1	0	2	1	0	2	3	1	1	1	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	2	1	2	1	0	3	3	1	0	2	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	5	0	2	1	0	2	2	1	0	1	0	1	2	0	1	0	0	0	1	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	1	1	2	1	0	3	3	1	1	2	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	1	0	2	1	0	3	3	1	2	0	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	4	0	2	1	0	1	3	1	0	0	0	1	2	0	1	1	0	0	1	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	1	0	2	1	0	2	3	1	0	1	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	4	1	2	1	0	3	3	1	0	2	0	1	2	0	1	0	0	0	1	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	2	0	2	1	0	3	3	1	0	0	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	0	1	1	1	1	2	2	0	2	1	1	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	1	1	1	1	1	2	0	0	0	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	1	1	2	1	0	1	0	1	2	2	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	1	0	2	1	0	1	3	1	0	0	0	1	2	0	1	0	0	0	1	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	2	0	2	1	0	3	3	1	1	1	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	3	0	2	1	0	2	3	2	2	1	0	1	2	0	1	0	0	0	1	0	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0
brown-spot	2	1	2	2	1	3	1	1	1	0	1	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	1	0	2	1	0	2	3	1	2	0	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	1	1	2	1	0	2	3	1	0	2	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	5	0	2	1	0	1	3	1	0	0	0	1	2	0	1	1	0	0	1	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	4	1	1	1	1	2	2	0	0	2	0	1	2	0	1	0	0	0	1	0	3	1	1	0	0	0	0	0	1	0	0	0	0	0	0
brown-spot	3	1	2	1	0	1	3	1	0	2	0	1	2	0	1	1	0	0	1	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Kelas Peyakit	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	F31	F32	F33	F34	F35
brown-spot	1	0	2	1	0	3	3	1	0	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	4	0	2	1	0	2	3	2	1	1	0	1	2	0	1	0	0	0	1	0	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0
brown-spot	2	1	2	1	0	2	3	1	0	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	2	1	1	1	1	0	0	1	1	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	3	1	2	1	0	3	1	1	0	2	0	1	2	0	1	0	0	0	1	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	3	0	2	1	0	3	3	2	2	0	0	1	2	0	1	0	0	0	1	0	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0
brown-spot	2	0	2	1	0	2	2	1	0	1	0	1	2	0	1	0	0	0	1	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	3	0	2	1	0	3	1	1	0	0	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	3	1	2	1	0	3	1	1	0	2	0	1	2	0	1	1	0	0	1	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	2	1	2	1	0	3	3	2	2	2	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
brown-spot	5	1	2	1	0	3	3	2	0	2	0	1	2	0	1	0	0	0	1	0	0	3	1	1	0	0	0	0	0	0	0	0	0	0	0
bacterial-blight	5	0	2	1	1	3	3	1	1	0	0	1	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bacterial-blight	4	0	2	2	1	2	3	1	1	1	0	1	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bacterial-blight	2	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bacterial-blight	3	0	1	1	0	1	2	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bacterial-blight	3	0	1	1	0	3	2	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bacterial-blight	3	0	2	1	1	2	1	1	1	0	0	1	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bacterial-blight	3	0	1	1	0	1	0	0	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bacterial-blight	4	0	2	1	1	0	3	1	1	1	0	1	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bacterial-blight	2	0	1	1	0	3	1	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bacterial-blight	4	1	2	2	1	2	1	1	1	2	0	1	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bacterial-pustule	2	1	1	2	0	2	2	0	0	2	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
bacterial-pustule	3	0	2	0	1	2	3	1	1	1	1	1	2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
bacterial-pustule	2	0	1	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bacterial-pustule	4	1	2	1	0	3	0	1	0	2	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
bacterial-pustule	3	0	2	1	1	1	1	1	0	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1
bacterial-pustule	3	1	1	0	0	2	0	0	0	2	0	1	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
bacterial-pustule	3	0	1	1	1	2	3	0	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
bacterial-pustule	3	1	2	1	0	0	2	1	0	2	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
bacterial-pustule	4	0	1	1	1	1	3	0	0	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
bacterial-pustule	5	1	1	1	0	2	0	0	1	2	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
purple-seed-stain	6	0	2	0	1	2	2	0	0	0	0	0	0	2	2	0	0	0	1	1	0	3	0	0	0	0	0	1	1	1	0	1	0	0	0
purple-seed-stain	6	0	2	0	0	2	2	0	1	1	0	1	2	0	0	0	0	0	1	0	0	3	0	0	0	0	0	1	1	1	0	1	0	0	0
purple-seed-stain	4	0	2	1	1	1	1	0	1	2	0	0	0	2	2	0	0	0	0	0	0	3	0	0	0	0	0	0	0	1	0	1	0	0	0
purple-seed-stain	4	0	2	1	1	0	0	0	0	1	0	1	2	0	0	0	0	0	0	1	0	3	0	0	0	0	0	0	0	1	0	1	0	0	0
purple-seed-stain	4	0	2	0	0	0	0	0	0	2	0	1	2	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	1	0	1	0	0	0
purple-seed-stain	6	0	2	2	0	2	2	0	0	1	0	1	2	0	0	0	0	0	1	0	0	3	0	0	0	0	0	1	1	1	0	1	0	0	0
purple-seed-stain	3	0	2	0	1	0	0	0	0	1	0	0	0	2	2	0	0	0	0	1	0	3	0	0	0	0	0	0	0	1	0	1	0	0	0
purple-seed-stain	3	0	2	1	1	3	3	0	1	1	0	0	0	2	2	0	0	0	0	0	0	3	0	0	0	0	0	0	0	1	0	1	0	0	0
purple-seed-stain	5	0	2	1	0	1	1	0	0	0	0	1	2	0	0	0	0	0	1	0	0	3	0	0	0	0	0	1	1	1	0	1	0	0	0
purple-seed-stain	4	0	2	1	0	0	0	0	1	1	0	0	0	2	2	0	0	0	1	0	0	3	0	0	0	0	0	0	0	1	0	1	0	0	0

Kelas Peyakit	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	F31	F32	F33	F34	F35
anthracnose	5	1	2	1	0	3	3	1	1	0	0	0	0	2	2	0	0	0	1	0	3	2	0	0	0	0	0	1	2	0	0	0	0	0	0
anthracnose	5	1	2	2	1	2	2	0	1	2	0	1	0	2	2	0	0	0	1	0	3	1	1	1	0	0	0	1	2	1	1	1	0	1	0
anthracnose	6	0	2	1	0	1	1	1	1	1	0	0	0	2	2	0	0	0	1	0	3	2	1	0	0	0	0	1	2	1	1	0	1	1	0
anthracnose	2	1	2	2	1	0	0	1	0	0	1	1	0	2	2	0	0	0	1	0	2	1	0	1	0	0	0	0	0	0	0	0	0	0	0
anthracnose	3	0	2	1	0	3	3	1	0	0	1	1	0	2	2	0	0	0	1	0	3	2	1	1	0	0	0	1	2	1	1	1	0	0	0
anthracnose	4	1	2	2	1	2	2	1	0	1	1	1	0	2	2	0	0	0	1	0	3	1	1	1	0	0	0	1	2	0	0	0	0	0	0
anthracnose	6	0	2	1	0	2	2	1	0	1	0	0	0	2	2	0	0	0	1	0	3	2	1	0	0	0	0	1	2	1	1	0	1	1	0
anthracnose	1	0	2	1	0	1	1	1	1	1	1	1	0	2	2	0	0	0	1	0	2	2	0	1	0	0	0	0	0	1	0	1	0	0	0
anthracnose	6	1	2	1	0	2	2	1	1	1	0	0	0	2	2	0	0	0	1	0	3	2	1	0	0	0	0	1	2	1	1	0	1	1	0
anthracnose	5	0	2	1	0	1	1	1	2	2	1	1	0	2	2	0	0	0	1	0	3	2	1	1	0	0	0	1	2	1	1	1	0	0	0
anthracnose	5	1	2	2	1	3	3	0	1	2	1	1	0	2	2	0	0	0	1	1	3	2	1	1	0	0	0	1	2	0	0	0	0	0	0
anthracnose	0	0	2	1	0	3	3	1	1	2	1	1	0	2	2	0	0	0	1	0	2	2	0	1	0	0	0	0	0	1	1	0	1	0	0
anthracnose	6	0	2	1	0	2	2	0	0	0	1	1	0	2	2	0	0	0	1	0	3	1	1	1	0	0	0	1	2	1	0	1	1	1	0
anthracnose	5	1	2	1	0	1	1	1	0	1	0	0	0	2	2	0	0	0	1	0	3	2	0	0	0	0	0	1	2	1	1	0	1	1	0
anthracnose	5	0	2	1	0	2	2	1	0	2	0	0	0	2	2	0	0	0	1	0	3	2	1	0	0	0	0	1	2	0	0	0	0	0	0
anthracnose	6	1	2	2	1	0	0	1	0	1	0	1	0	2	2	0	0	0	1	0	3	1	1	1	0	0	0	1	2	0	0	0	0	0	0
anthracnose	5	0	2	1	0	1	1	1	0	0	0	0	0	2	2	0	0	0	1	0	3	2	1	0	0	0	0	1	2	1	1	0	1	1	0
anthracnose	6	1	2	2	1	3	3	0	2	1	0	1	0	2	2	0	0	0	1	1	3	2	1	1	0	0	0	1	2	0	0	0	0	0	0
anthracnose	5	1	2	1	0	3	3	1	0	1	0	0	0	2	2	0	0	0	1	0	3	2	0	0	0	0	0	1	2	1	1	0	1	1	0
anthracnose	5	1	2	1	0	2	2	1	1	1	0	0	0	2	2	0	0	0	1	0	3	2	0	0	0	0	0	1	2	1	1	0	1	1	0
phyllosticta-leaf-spot	3	1	1	1	0	0	2	0	0	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phyllosticta-leaf-spot	3	0	0	1	1	0	2	0	0	1	0	1	2	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phyllosticta-leaf-spot	3	1	1	1	0	0	0	0	0	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phyllosticta-leaf-spot	3	0	0	1	1	2	0	0	0	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phyllosticta-leaf-spot	3	1	1	2	0	3	2	0	1	1	0	1	2	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phyllosticta-leaf-spot	2	0	0	1	1	0	3	0	2	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phyllosticta-leaf-spot	1	0	0	2	1	3	2	1	1	1	0	1	2	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phyllosticta-leaf-spot	2	1	1	1	0	2	2	1	1	1	0	1	2	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phyllosticta-leaf-spot	2	0	0	2	1	3	0	1	1	0	1	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
phyllosticta-leaf-spot	2	1	1	2	0	3	3	0	2	2	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	4	1	2	1	0	1	1	1	1	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	4	0	1	1	0	3	3	1	0	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	3	0	2	1	0	0	0	1	0	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	6	0	2	2	0	3	3	0	1	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	6	0	1	1	1	2	2	0	2	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
alternarialeaf-spot	5	0	2	2	0	3	3	1	0	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	6	0	1	1	0	3	3	0	1	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	5	1	2	2	0	3	1	0	0	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	6	0	2	2	0	3	3	0	1	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	6	0	2	2	0	3	2	1	1	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	5	0	2	2	0	2	3	0	0	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Kelas Peyakit	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	F31	F32	F33	F34	F35
alternarialeaf-spot	4	1	2	1	0	3	0	1	1	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	6	0	2	1	0	1	1	0	1	0	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	5	0	2	2	0	2	2	1	0	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	5	1	2	1	0	0	0	0	0	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	4	0	2	1	0	2	2	0	0	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	4	0	2	1	0	1	1	1	1	1	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	5	0	2	1	0	2	1	0	0	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	6	0	2	2	0	3	2	0	1	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	6	1	2	2	0	1	1	0	1	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	5	1	2	2	0	3	1	1	0	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	5	1	2	2	0	3	3	1	1	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	4	1	2	1	0	2	1	0	0	2	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	6	1	1	2	0	2	2	0	2	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
alternarialeaf-spot	4	1	2	1	0	1	2	1	1	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	6	1	2	2	0	2	1	0	1	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	4	1	2	1	0	0	3	0	0	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	4	0	2	2	0	3	3	1	0	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	5	0	2	2	0	2	3	1	0	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	3	0	2	1	0	0	0	1	0	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	5	0	2	1	0	1	2	0	1	0	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	5	0	2	2	0	1	1	0	1	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	4	0	2	2	0	1	1	1	0	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	5	1	2	1	0	3	2	1	0	2	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	6	0	2	1	0	2	1	0	0	1	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	5	0	2	2	0	2	1	1	0	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	5	0	2	1	0	3	0	0	1	0	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alternarialeaf-spot	6	0	2	1	0	1	2	0	1	1	0	1	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
frog-eye-leaf-spot	6	0	1	2	0	3	3	0	0	0	0	1	2	0	1	0	0	0	1	0	3	2	1	1	0	0	0	1	2	1	0	0	0	0	0
frog-eye-leaf-spot	4	0	1	2	0	1	1	0	1	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
frog-eye-leaf-spot	5	0	1	1	0	2	1	0	0	0	0	1	2	0	1	0	0	0	1	0	3	1	0	1	0	0	0	0	0	0	0	0	0	0	0
frog-eye-leaf-spot	5	1	2	1	0	3	2	0	0	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
frog-eye-leaf-spot	6	1	2	2	0	3	3	0	1	2	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	4	0	1	1	0	3	3	1	1	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
frog-eye-leaf-spot	3	0	2	1	0	2	3	0	1	1	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	5	0	2	2	0	2	2	0	0	1	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	5	0	2	1	0	1	1	1	1	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
frog-eye-leaf-spot	5	0	2	2	0	2	3	0	1	1	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	5	0	2	1	0	0	1	0	1	1	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	4	0	2	1	0	2	3	0	1	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Kelas Peyakit	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	F31	F32	F33	F34	F35
frog-eye-leaf-spot	4	0	2	2	0	1	1	1	1	0	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	4	0	2	1	0	2	1	1	1	1	0	1	2	0	1	0	0	0	1	0	3	1	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	3	1	2	1	0	3	2	1	0	2	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	5	0	2	1	0	3	0	1	0	1	0	1	2	0	1	0	0	0	1	0	3	1	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	5	0	2	2	0	1	1	0	1	0	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	4	0	2	2	0	1	2	1	0	0	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	5	0	2	2	0	2	1	0	1	1	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	5	0	2	1	0	3	0	1	0	0	0	1	2	0	1	0	0	0	1	0	3	1	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	3	0	2	1	0	1	2	1	0	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
frog-eye-leaf-spot	6	0	1	2	0	3	3	0	1	0	0	1	2	0	1	0	0	0	1	0	3	2	1	0	0	0	0	1	2	1	0	1	1	1	0
frog-eye-leaf-spot	5	0	1	1	0	1	3	1	2	0	0	1	2	0	1	0	0	0	1	0	3	0	1	0	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	5	0	2	1	0	3	2	1	0	0	0	1	2	0	1	0	0	0	1	0	3	1	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	5	1	2	1	0	3	3	0	1	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
frog-eye-leaf-spot	3	1	2	1	0	3	0	1	0	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
frog-eye-leaf-spot	6	1	2	2	0	3	1	0	1	2	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	4	0	2	1	0	1	2	1	0	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
frog-eye-leaf-spot	4	0	2	2	0	1	0	1	0	0	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	6	1	2	2	0	3	0	0	0	2	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	5	1	2	2	0	3	3	0	1	2	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	4	0	2	1	0	0	1	1	1	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
frog-eye-leaf-spot	4	0	2	1	0	2	3	1	1	1	0	1	2	0	1	0	0	0	1	0	3	1	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	4	1	1	2	0	1	1	0	2	2	1	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
frog-eye-leaf-spot	4	0	2	1	0	2	0	0	0	1	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
frog-eye-leaf-spot	5	1	2	1	0	1	2	1	0	2	0	1	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
frog-eye-leaf-spot	4	0	2	2	0	1	3	1	1	0	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	5	0	2	1	0	1	2	0	0	0	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	5	0	2	2	0	2	0	0	0	1	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0
frog-eye-leaf-spot	5	1	2	1	0	2	3	0	1	2	0	1	2	0	1	0	0	0	1	0	3	2	0	1	0	0	0	1	1	0	0	0	0	0	0

Keterangan *Data Set* :

Nilai untuk attribut dienkodekan secara numerikal, dimana nilai pertama dienkodekan dengan nilai ‘0’, nilai yang kedua ‘1’ dan begitu seterusnya. Nilai ‘dna’ menunjukkan fitur tersebut tidak digunakan. Contoh : penamanan kedelai pada bulan april maka F1 bernilai 0.

Simbol	Nama	Nilai Attribut
F1	date	april,may,june,july,august,september,october.
F2	plant-stand	normal,lt-normal.
F3	precip	lt-norm,norm,gt-norm.
F4	temp	lt-norm,norm,gt-norm.
F5	hail	yes,no.
F6	crop-hist	diff-lst-year,same-lst-yr,same-lst-two-yrs, same-lst-sev-yrs.
F7	area-damaged	scattered,low-areas,upper-areas,whole-field.
F8	severity	minor,pot-severe,severe,?.
F9	seed-tmt	none,fungicide,other.
F10	germination	90-100%,80-89%,lt-80%.
F11	plant-growth	norm,abnorm.
F12	leaves	norm,abnorm.
F13	leafspots-halo	absent,yellow-halos,no-yellow-halos.
F14	leafspots-marg	w-s-marg,no-w-s-marg,dna.
F15	leafspots-size	lt-1/8,gt-1/8,dna.
F16	leaf-shread	absent,present.
F17	leaf-malf	absent,present.
F18	leaf-mild	absent,upper-surf,lower-surf.
F19	stem	norm,abnorm.
F20	lodging	yes,no.
F21	stem-cankers	absent,below-soil,above-soil,above-sec-nde.
F22	canker-lesion	dna,brown,dk-brown-blk,tan.
F23	fruiting-bodies	absent,present.
F24	external decay	absent,firm-and-dry,watery.
F25	mycelium	absent,present.
F26	int-discolor	none,brown,black.
F27	sclerotia	absent,present.
F28	fruit-pods	norm,diseased,few-present,dna.
F29	fruit spots	absent,colored,brown-w/blk-specks,distort,dna.
F30	seed	norm,abnorm.
F31	mold-growth	absent,present.
F32	seed-discolor	absent,present.
F33	seed-size	norm,lt-norm.
F34	shriveling	absent,present.
F35	roots	norm,rotted,galls-cysts.