IDENTIFIKASI JENIS TANAMAN OBAT INDONESIA BERDASARKAN BENTUK CITRA DAUN MENGGUNAKAN METODE DETEKSI TEPI DAN *GAUSSIAN NAÏVE BAYES*

SKRIPSI

Oleh :
AYU PUTRI RIZKIA
NIM. 19650023

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2023

IDENTIFIKASI JENIS TANAMAN OBAT INDONESIA BERDASARKAN BENTUK PADA CITRA DAUN MENGGUNAKAN METODE DETEKSI TEPI DAN *GAUSSIAN NAÏVE BAYES*

SKRIPSI

Oleh :
AYU PUTRI RIZKIA
NIM. 19650023

Diajukan Kepada :
Fakultas Sains dan Teknologi
Universitas Islam Negeri (UIN) Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S. Kom)

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2023

HALAMAN PERSETUJUAN

IDENTIFIKASI JENIS TANAMAN OBAT INDONESIA BERDASARKAN BENTUK CITRA DAUN MENGGUNAKAN METODE DETEKSI TEPI DAN *GAUSSIAN NAÏVE BAYES*

SKRIPSI

Oleh :
AYU PUTRI RIZKIA
NIM. 19650023

Telah Diperiksa dan Disetujui untuk Diuji
Tanggal 23 Mei 2023

Pembimbing I,	Pembimbing II,
Dr. Cahyo Crysdian	Fajar Rohman Hariri, M.Kom
NIP. 19740424 200901 1 008	NIP. 19890515 201801 1 001

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Fachrul Kurniawan, M. MT, IPM
NIP. 19771020 200912 1 001

HALAMAN PENGESAHAN

IDENTIFIKASI JENIS TANAMAN OBAT INDONESIA BERDASARKAN BENTUK CITRA DAUN MENGGUNAKAN METODE DETEKSI TEPI DAN *GAUSSIAN NAÏVE BAYES*

SKRIPSI

Oleh:
AYU PUTRI RIZKIA
NIM. 19650023

Telah Dipertahankan di Depan Dewan Penguji
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Pada Tanggal: 12 Juni 2023

Susunan Dewan Penguji

Ketua Penguji	: Dr.Irwan Budi Santoso NIP. 19770103 201101 1 004	( )
Anggota Penguji I	: Okta Qomaruddin Aziz, M.Kom NIP. 19911019 201903 1 013	( )
Anggota Penguji II	: Dr.Cahyo Crysdian NIP. 19740424 200901 1 008	( )
Anggota Penguji III	: Fajar Rohman Hariri, M.Kom NIP. 19890515 201801 1 001	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr.Fachrul Kurniawan, M. MT., IPM
NIP. 19771020 200912 1 001

PERNYATAAN KEASLIAN PENULISAN

Saya yang bertanda tangan dibawah ini:

Nama	: Ayu Putri Rizkia
NIM	: 19650023
Fakultas / Program Studi	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	:Identifikasi Jenis Tanaman Obat Indonesia Berdasarkan Bentuk Citra Daun Menggunakan Metode Deteksi Tepi dan *Gaussian Naïve Bayes*

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila di kemudian hari terbukti atau dapat dibuktikan Skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 12 Juni 2023
Yang membuat pernyataan,

Ayu Putri Rizkia
NIM. 19650023

HALAMAN MOTTO

"Keep it trial and error till you know what the problem is"

"Never be ashamed of trying, Effortlessness is a myth"

-Taylor Swift

HALAMAN PERSEMBAHAN

Alhamdulillah puji syukur kehadirat Allah *subhanahu wa ta'ala* serta shalawat serta salam kepada Rasulullah *SAW*. Dengan segenap hati dan rasa syukur yang turut menggenapi kebahagiaan akan terselesainya Skripsi ini, penulis mempersembahkan karya ini kepada :

1. Kepada keluarga, cinta dan kasih yang mengiringi serta doa yang selalu terurai : Papa, Mama, dan Mas Jefri tersayang.
2. Kepada pihak-pihak yang tidak dapat disebutkan satu persatu yang secara langsung dan tidak langsung memotivasi penulis untuk segera menyelesaikan Skripsi ini dengan bentuk pertanyaan "kapan lulus?", "kapan sidang?", "kapan wisuda?". Akhirnya pertanyaan ini tidak hanya bisa dijawab dengan senyuman manis dengan hati yang dongkol sekaligus panik.
3. Kepada diri sendiri, yang sudah menyelesaikan tanggung jawab yang pernah dimulai di bawah bentuk rasa kepanikan, kegagalan, dan segala ketidakpastian yang senantiasa mengiringi proses pengerjaan Skripsi ini.

Terimakasih sudah berkontribusi dalam hari-hari penulis selama proses penyelesaian Skripsi ini. Kepada yang sedang berjuang, apapun yang terjadi, tetaplah bernafas.

KATA PENGANTAR

Assalamualaikum Wr. Wb.

Alhamdulillah segala puji bagi Allah *subhanahu wa ta'ala* atas rahmat karunia-Nya sehingga penulis mampu menyelesaikan Skripsi ini dengan judul "Identifikasi Jenis Tanaman Obat Indonesia Berdasarkan Bentuk Citra Daun Menggunakan Metode Deteksi Tepi dan *Gaussian Naïve Bayes*". Shalawat serta salam tetap senantiasa tercurahkan kepada Nabi Muhammad SAW. Semoga kita semua mendapatkan syafaatnya kelak. *Aamiin*.

Dalam proses pengerjaan Skripsi yang penuh kesan ini, banyak pihak-pihak yang senantiasa menemani penulis untuk selalu berproses dan berprogres. Atas bantuan dan dukungan yang telah diberikan, penulis menyampaikan doa dan terimakasih yang setulus-tulusnya kepada :

1. Prof. Dr. H. M. Zainuddin, M.A., selaku rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Sri Hariani, M.Si., selaku dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Fachrul Kurniawan, M.MT., IPM, selaku Ketua Program Studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Dr. Cahyo Crysdian dan Fajar Rohman Hariri, M.Kom selaku Dosen Pembimbing I dan Dosen Pembimbing II dengan segala arahan dan wejangan yang sangat berarti dan senantiasa membimbing setiap langkah yang saya lalui dalam proses pengerjaan skripsi ini

5. Segenap Dosen Teknik Informatika yang telah memberikan bimbingan dan pengalaman yang berharga selama masa studi.
6. Papa, Mama, dan Mas Jefri tersayang yang selalu mencurahkan cinta dan kasihnya yang membangun asa melalui doa-doa penuh makna, sehingga begitu lancar dan selesainya Skripsi ini dengan baik dan tepat waktu.
7. Saudara-saudara Angkatan 2019 ALIEN *"Alliance of Informatics Engineering"* yang selalu peduli, mendukung serta memotivasi dan bersama-sama membuat pengalaman-pengalaman berharga yang tidak pernah penulis dapatkan dimanapun.
8. Saudara-saudara HIMATIF ENCODER yang telah menemani penulis berproses dan memberikan kesempatan untuk bersama-sama menciptakan ide, membangun inovasi, serta mendiskusikan pelajaran dari sebuah peristiwa.
9. Sobat-sobat "Ahlussunnah Wal Jama'ah" yaitu Widia, Bila, Bisyri, Deri, Thoriq, Dicky, Pejon, Zulfan, Fikri, Cipmank, Riduan, Sadad, Anam, Dayat, Andi, Alphin, Faiz, sekumpulan manusia terbaik yang senantiasa penulis andalkan apapun yang terjadi. Terimakasih untuk segala bentuk teori, opini, analogi, dan obrolan-obrolan seru yang pernah diperdebatkan serta didiskusikan bersama

Penulis menyadari dalam penyusunan skripsi ini masih banyak terdapat kekurangan. Penulis berharap skripsi ini dapat memberikan manfaat kepada para pembaca.

Wassalamu'alaikum Wr. Wb.

Malang, 15 Juni 2023

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

Rizkia, Ayu Putri.2023. **Identifikasi Jenis Tanaman Obat Indonesia Berdasarkan Bentuk Citra Daun Menggunakan Metode Deteksi Tepi dan *Gaussian Naïve Bayes***. Skripsi. Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Dr. Cahyo Crysdian (II) Fajar Rohman Hariri, M.Kom.

Kata Kunci : *Computer Vision*, *Gaussian Naïve Bayes* , *Image Recognition*

Indonesia menjadi salah satu negara dengan tingkat keanekaragaman hayati terbesar di dunia yang mencakup flora dan fauna. Terdapat sekitar 30.000 jenis tanaman yang tumbuh di hutan tropis Indonesia. Selain menjadi sektor utama dalam produksi bahan pangan, banyak tanaman juga memiliki khasiat dalam pengobatan tradisional dan mempunyai efek positif dalam kesehatan tubuh manusia. Perkembangan teknologi yang berkembang pesat, popularitas tanaman obat semakin menurun terkikis oleh industri kimia obat modern. Masyarakat saat ini kurang mengenali jenis tanaman yang memiliki khasiat obat. Penelitian ini bertujuan untuk membuat sistem identifikasi jenis tanaman obat berdasarkan bentuk citra daun menggunakan metode deteksi tepi dan Gaussian Naïve Bayes (GNB). Input yang digunakan dalam penelitian ini berupa data citra daun yang berjumlah 16 jenis daun dengan masing-masing jenis daun terdapat 50 data citra. Dataset tersebut dilakukan ekstraksi fitur dengan pemilihan fitur bentuk yaitu panjang, lebar, luas, keliling, slimness (kerampingan daun) kemudian diambil parameter rata-rata dan varian untuk setiap fitur sebagai input dalam identifikasi menggunakan GNB. Berdasarkan hasil pengujian dengan 3 skenario pembagian data dan 3 variasi jumlah kelas didapatkan performa terbaik terdapat pada rasio 90:10 untuk 6 kelas daun dengan nilai accuracy sebesar 90%, precision 92,46%, recall 90%, dan f-measure 89,69%. Selanjutnya, hasil uji coba pada 16 kelas daun mengalami penurunan akurasi menjadi 57,50%. Hal ini dikarenakan nilai fitur-fitur pada kategori jenis daun memiliki rentang nilai yang sangat dekat sehingga pemilihan jenis fitur bentuk dan banyaknya jumlah kelas sangat mempengaruhi hasil performa sistem.

ABSTRACT

Rizkia, Ayu Putri.2023. **Identification of Indonesian Medicinal Plants Based on Leaf Image Shape Using Edge Detection Method and *Gaussian Naïve Bayes***. Thesis. Department of Informatics Engineering, Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University Malang. Supervisor: (I) Dr. Cahyo Crysdian (II) Fajar Rohman Hariri, M.Kom

Indonesia is one of the countries with the highest level of biodiversity in the world, encompassing flora and fauna. There are approximately 30,000 species of plants that grow in the tropical forests of Indonesia. Besides being a primary sector in food production, many plants also possess medicinal properties and have positive effects on human health in traditional medicine. The rapid development of technology has caused a decline in the popularity of herbal plants, eroded by the modern pharmaceutical industry. Presently, society has limited knowledge of plant species with medicinal properties. This research aims to develop a system for identifying medicinal plant species based on leaf image morphology, utilizing edge detection and Gaussian Naïve Bayes (GNB) methods. The input used in this study consists of leaf image data, comprising 16 types of leaves with 50 image samples for each leaf type. Feature extraction is performed on the dataset, selecting shape features such as length, width, area, perimeter, and slimness (leaf elongation). The average and variance parameters for each feature are then utilized as inputs for identification using GNB. Based on the test results from three data division scenarios and three variations in the number of classes, the best performance is achieved with a 90:10 data ratio for 6 leaf classes, yielding an accuracy of 90%, precision of 92.46%, recall of 90%, and f-measure of 89.69%. Furthermore, when tested on 16 leaf classes, the accuracy decreased to 57.50%. This is due to the close range of feature values within leaf type categories, indicating that the selection of shape features and the number of classes significantly affect the system's performance.

Keywords: *Computer Vision, Gaussian Naïve Bayes , Image Recognition*

مستخلص البحث

رزقية، أيو بوتري .٢٠٢٣. "تحديد أنواع النباتات الطبية الإندونيسية بناء على شكل صورة الورقة باستخدام طريقة الكشف الحواف وغواثيان بايز ساذج ". البحث الجامعي. قسم الهندسة المعلوماتية، كلية العلوم والتكنولوجيا بجامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف الأول: د. جهيو كرسديان. المشرف الثاني: فجر الرحمن الحريري، الماجستير.

الكلمات الرئيسية: رؤية الكمبيوتر، الغواثيان بايز ساذج ، التعرف على الصور

إندونيسيا هي إحدى البلدان التي لديها أكبر مستوى من التنوع البيولوجي في العالم والذي يشمل النباتات والحيوانات. هناك حوالي ٣٠٠٠٠ نوع من النباتات التي تنمو في الغابات الاستوائية في إندونيسيا. بالإضافة إلى كونها قطاعا رئيسيا في إنتاج الغذاء، فإن عدة النباتات لها أيضا خصائص في الطب التقليدي ولها تأثير إيجابي على صحة جسم الإنسان. التطور السريع للتكنولوجيا، وانخفاض شعبية النباتات الطبية تتآكل من قبل الصناعة الكيميائية الطبية الحديثة. لا يتعرف الناس اليوم على أنواع النباتات التي لها خصائص طبية. يهدف هذا البحث إلى إنشاء نظام لتحديد أنواع النباتات الطبية بناء على شكل صورة الورقة باستخدام طرق الكشف عن الحواف وغواثيان بايز ساذج . كانت المدخلات المستخدمة في هذا البحث في شكل بيانات صورة ورقة بلغ مجموعها ١٦ نوعا من الأوراق مع كل نوع من الأوراق كان هناك ٥٠ بيانات صورة. تم تنفيذ مجموعة البيانات باستخراج المعالم عن طريق اختيار معالم الشكل، وهي الطول والعرض والمساحة والمحيط والنحافة (رقة الورقة) ثم أخذ متوسط المعلمات والمتغيرات لكل معلم كمدخل في التعريف باستخدام. استنادا إلى نتائج الاختبار مع ٣ سيناريوهات لمشاركة البيانات و ٣ اختلافات في عدد الفئات، تم العثور على أفضل أداء بنسبة ١٠:٩٠ لفئات ٦ أوراق بقيمة دقة ٩٠%، وضبط ٩٢.٤٦%، واستدعاء ٩٠%، و ف قياسي ٨٩.٦٩%. علاوة على ذلك، انخفضت دقة نتائج الاختبار على ١٦ فئة ورقة إلى ٥٧.٥٠%. وذلك لأن قيمة المعالم في فئة نوع الورقة لها نطاق قريب جدا من القيم بحيث يؤثر اختيار نوع الشكل والعدد الكبير من الفئات بشكل كبير على نتائج أداء النظام.

BAB I

PENDAHULUAN

1.1. Latar Belakang

Indonesia menjadi salah satu negara dengan tingkat keanekaragaman hayati terbesar di dunia. Terdapat banyak macam dan varietas tumbuhan di hutan tropis Indonesia. Keanekaragaman hayati yang mencakup flora di Indonesia terdapat sekitar 30.000 jenis tanaman dari total jenis tumbuhan dunia yang berjumlah sekitar 40.000 jenis di dunia. Tumbuhan di Indonesia juga menjadi salah satu komoditas bahan pangan nasional terbesar yang memiliki potensi tinggi dalam dunia industri. Selain menjadi sektor utama dalam produksi bahan pangan, banyak tanaman juga memiliki manfaat serta khasiat dalam pengobatan tradisional oleh masyarakat Indonesia dan mempunyai efek positif dalam kesehatan tubuh manusia. Tercatat sekitar 9600 jenis tumbuhan di Indonesia yang teridentifikasi memiliki khasiat obat, namun baru sekitar 2000 jenis tanaman obat yang sudah di produksi menjadi ramuan obat tradisional.

Potensi perkembangan produksi tanaman obat sebagai pengobatan tradisional di kalangan masyarakat masih sangat tinggi, hal ini karena nenek moyang dari generasi ke generasi telah menggunakan tanaman obat sebagai pengobatan dan pemeliharaan kesehatan dan kecantikan bagi tubuh manusia. Hal tersebut dibuktikan dengan dokumen serat primbon Jampi, Relief Candi Borobudur, dan Daun Lontar Husodo yang menceritakan bahwa ada seseorang yang meracik obat menggunakan bahan tanaman (Sumayyah dan Nada, 2017)

Perkembangan teknologi saat ini telah berkembang pesat di berbagai bidang. Popularitas pengobatan tradisional menggunakan tanaman obat semakin menurun karena terkikis oleh industri kimia obat modern. Masyarakat saat ini kurang mengenali jenis tanaman yang mempunyai khasiat obat. Hal ini karena keterbatasan memori manusia dalam mengingat serta mengidentifikasi jenis tanaman obat dari generasi ke generasi (Damayanti & Adi, 2019). Berdasarkan faktor tersebut ribuan jenis tanaman obat di Indonesia berpotensi menjadi sia-sia dan terbuang karena dapat dianggap sebagai tanaman liar, sehingga perkembangan produksi dan konsumsi dari tanaman obat dapat menurun seiring berjalannya waktu karena pengetahuan masyarakat mengenai tanaman obat yang semakin rendah. Maka untuk saat ini diperlukan klasifikasi jenis tanaman obat untuk meningkatkan pengetahuan masyarakat mengenai jenis dan khasiat dari tanaman obat untuk memaksimalkan potensinya.

Proses identifikasi tanaman obat dapat dilakukan dengan membangun sistem secara otomatis yang menerapkan teknologi kecerdasan buatan. Sistem tersebut disebut sebagai machine learning. Dalam membuat sebuah machine learning dibutuhkan algoritma untuk dapat melakukan eksplorasi data. Salah satu contoh algoritma yang digunakan dalam machine learning yaitu naïve bayes. Naïve bayes merukan salah satu algoritma machine learning yang digunakan untuk klasifikasi suatu data. Algoritma naïve bayes yang didasarkan pada teorema bayes, algoritma ini menerapakan konsep memprediksi probabilitas tertentu di masa depan untuk pengambilan keputusan berdasarkan pengalaman atau data yang ada

sebelumnya. Dalam Al-Qur'an surah An-Nahl ayat 11 Allah *Subhanahu wa ta'ala* berfirman:

يُنۢبِتُ لَكُم بِهِ ٱلزَّرۡعَ وَٱلزَّيۡتُونَ وَٱلنَّخِيلَ وَٱلۡأَعۡنَٰبَ وَمِن كُلِّ ٱلثَّمَرَٰتِۗ إِنَّ فِي ذَٰلِكَ لَأٓيَةٗ لِّقَوۡمٖ يَتَفَكَّرُونَ

"Dengan (air hujan) itu Dia menumbuhkan untuk kamu tanam-tanaman, zaitun, kurma, anggur dan segala macam buah-buahan. Sungguh, pada yang demikian itu benar-benar terdapat tanda (kebesaran Allah) bagi orang yang berpikir." (QS.an-Nahl : 11)

Menurut tafsir Ibnu Katsir, ayat ini menjelaskan bahwa Allah *Subhanahu wa ta'ala* telah menurunkan air hujan untuk kita semua untuk menumbuhkan berbagai tanaman dan buah-buahan untuk memenuhi kebutuhan manusia sebagai tanda kekuasaan bagi kaum yang memikirkan yaitu petunjuk bahwa tiada Tuhan selain Allah.

1.2. Pernyataan Masalah

a. Berapa nilai akurasi, presisi, *recall,* dan *f-measure* dari metode Gaussian Naïve Bayes dalam melakukan klasifikasi tanaman obat Indonesia berdasarkan bentuk daun?
b. Bagaimana hasil analisis pengujian dalam melakukan klasifikasi tanaman obat berdasarkan bentuk daun menggunakan deteksi tepi dan Gaussian Naïve Bayes?

1.3. Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah :

a. Mengukur nilai akurasi, presisi, dan recall, dan f-measure dari metode Gaussian *Naïve Bayes* dalam melakukan klasifikasi tanaman obat Indonesia.

b. Menganalisis hasil pengujian dalam klasifikasi tanaman obat Indonesia dengan menggunakan metode Deteksi Tepi dan *Naïve Bayes.*

1.4. Batasan Penelitian

Mengingat luasnya cakupan masalah yang akan diteliti, maka penulis membatasi penelitian ini yaitu :

a. Citra daun yang digunakan berupa satu lembar daun dengan background putih.
b. Jumlah dataset ada 16 jenis tanaman dengan masing-masing jenis berjumlah 50 gambar.
c. Data gambar diasumsikan dalam jarak dan dimensi yang sama.
d. Fokus penelitian diarahkan pada analisis bentuk pada daun.

1.5. Manfaat Penelitian

Dengan dilakukannya penelitian ini diharapkan dapat memberi beberapa manfaat yaitu :

a. Bagi produsen obat herbal dapat mengidentifikasi khasiat dari tanaman obat dalam memaksimalkan potensi penggunaan obat herbal dari tanaman.
b. Bagi pelajar di bidang farmakologi dapat membantu mengidentifikasi jenis tanaman obat Indonesia yang dapat mengembangkan penelitian tentang pengobatan herbal.

BAB II

TINJAUAN PUSTAKA

2.1. Klasifikasi Tanaman

Klasifikasi merupakan kata serapan bahasa Belanda yaitu *classificatie* dan berasal dari bahasa Perancis *classification.* Pengertian dari klasifikasi ini sendiri mengacu pada proses penyusunan data secara sistematis atau berdasarkan beberapa kaidah yang telah ditetapkan. Menurut Wikipedia, tanaman merupakan salah satu jenis organisma yang umumnya ditanam atau dibudidayakan oleh manusia untuk meningkatkan nilai ekonominya. Tanaman juga biasanya dimanfaatkan sebagai sumber makanan dan pengobatan. Tanaman obat atau biasanya disebut herbal merupakan tanaman yang mempunyai khasiat obat sehingga sering digunakan sebagai pengobatan tradisional masyarakat (Harefa, 2020).

Damayanti dan Adi (2019) melakukan penelitian mengenai klasifikasi tanaman obat berdasarkan bentuk daun, tekstur, dan warna daun dengan menggunakan metode jaringan tiruan Backpropagation. Peneliti menggunakan 7 jenis tanaman obat dengan masing-masing jenis tanaman terdapat 27 sampel daun. Preprocessing data yang dilakukan yaitu grayscalling, binarisasi, pengubahan beackground berwarna putih, deteksi tepi dan cropping. Pada proses ekstraksi ciri bentuk menghasilkan 8 ciri, 6 tekstur dan 7 ciri warna sehingga total ciri yang digunakan berjumlah 21 ciri. Ekstrasi ciri bentuk mengambil citra hasil binarisasi dan deteksi tepi sebagai input. Dari proses tersebut dihasilkan 4 ciri yang dilakukan normalisasi yaitu ciri panjang, luas, lebar dan keliling. Identifikasi tanaman obat ini menggunakan metode jaringan syaraf tiruan backpropagation dengan hasil akurasi

yang didapatkan dalam penelitian ini jika seluruh ciri digunakan menghasilkan nilai optimal sebesar 91% dengan jumlah 21 ciri sebagai inputan, total 40 neuron terdapat pada *hidden layer* 1 dan 15 neuron pada *hidden layer 2.*

Dileep dan Pournami (2019) melakukan penelitian klasifikasi tanaman obat menggunakan deep learning dengan model *Convolutional Neural Network*. Penelitian ini mengklasifikasikan tanaman obat berdasarkan ciri daun yaitu, bentuk, warna, tekstur daun dengan total dataset yaitu 2400 citra daun dari 40 jenis tanaman obat, kemudian arsitektur yang digunakan yaitu Alexnet dengan pengklasifikasian Softmax dan SVM. Hasil penelitian ini mencapai nilai akurasi sebesar 96,76%.

Padao dan Maravillas (2016) melakukan penelitian tentang klasifikasi tumbuhan berdasarkan bentuk dan tekstur daun dengan metode naïve bayes. Dataset yang digunakan dalam penelitian ini berjumlah 30 jenis tumbuhan dengan total 340 citra daun. Hasil rata rata area ROC menggunakan naïve bayes sebesar 0,981 dengan standar deviasi area ROC sebesar 0,023.

Rahayuda (2016) melakukan penelitian mengenai identifikasi jenis obat berdasarkan citra logo menggunakan naïve bayes pada proses klasifikasinya. Penentuan kelas atau jenis logo berjmlah 7 kelas dengan 3 jenis dari obat tradisional dan 4 jenis dari obat modern. Total data sampel yang digunakan yaitu 70 data yang didapatkan menggunakan camera pocket atau kamera handphone. Proses klasifikasinya dilakukan menggunakan matlab programming yang kemudian data gambar logo tersebut di proses dengan melakukan proses RGB to Grey. Kemudian untuk mendapatkan bentuk dari logo dilakukan deteksi tepi dengan pendekatan menggunakan *Canny Edge Detector*. Dalam proses ekstrasi fitur menggunakan

metode GLCM dengan ciri tekstur yang digunakan yaitu *Contrast*, *Correlation*, *Homogeneity*, dan *Energy* dan untuk proses training menggunakan metode Naïve Bayes. Hasil dari penelitian ini mendapatkan nilai akurasi untuk 3 kategori jenis obat bebas sebesar 80%, obat bebas terbatas sebesar 100%, dan obat keras sebesar 100%.

2.2. Deteksi Tepi

Pada proses analisis pengolahan data citra digital sangat penting untuk menggunakan data citra yang cocok dalam proses pengolahannya. Data citra yang digunakan dalam penelitian sangat berpengaruh pada hasil akhir penelitian. Dalam memproses sebuah gambar terdapat fitur yang paling dasar yaitu tepi gambar, tepi gambar terletak paling menonjol dari intensitas sebuah gambar atau citra. Deteksi tepi merupakan cara yang tepat untuk dilakukan dalam pemrosesan gambar. Deteksi tepi berfungsi untuk menandai detail bagian dari sebuah gambar dan peran deteksi tepi terletak pada penentuan nilai ambang batas untuk menghasilkan nilai yang optimal.

Tanaman mangga merupakan salah satu tanaman potensial yang sering dijumpai di sekitar lingkungan. Tanaman yang menghasilkan buah ini memiliki banyak varietas dalam berbagai ukuran, bentuk, dan warna. Hal ini dikarenakan keberagaman genetik dalam varietas tanaman mangga. Salah satu fitur unik yang membedakan varietas mangga adalah struktur tulang daun, namun saat ini untuk dapat membedakan varietas mangga masih menggunakan cara manual, maka dibutuhkan sebuah sistem untuk dapat mengenali jenis mangga secara otomatis berdasarkan struktur daun mangga. Liantoni dan Hermanto (2017) telah melakukan

penelitian mengenai klasifikasi jenis mangga menggunakan metode K-Nearest Neighbor. Dalam penelitian ini juga melakukan deteksi tepi dengan metode *ant colony optimization* untuk mendapatkan hasil deteksi tepi struktur tulang mangga secara optimal sehingga dapat meningkatkan hasil akurasi klasifikasi jenis mangga. Hasil akurasi yang didapatkan dalam penelitian ini didapatkan nilai sebesar 66,67%.

Kurniawan Budhi *et al* (2019) pernah melakukan penelitian deteksi penyakit tanaman jagung berdasarkan pola daun dengan metode deteksi tepi sobel. Sampel data yang digunakan dalam penelitian ini berasal dari ladang jagung di Desa Tiripan, Nganjuk dengan total data daun yang terkena penyakit sebanyak 30 data daun, dan daun yang sehat sebanyak 50 data daun. Hasil dari penelitian ini disimpulkan bahwa berdasarkan analisis histogram, daun yang terindikasi daun sehat mempunyai komposisi warna yang relatif stabil antara warna merah, hijau, biru daripada daun yang terindikasi penyakit. Apabila dibandingkan nilai intensitas warananya, semakin parah penyakit yang terdapat pada daun makan nilai intensitas warna akan semakin rendah dan warna daun akan lebih kusam.

Aulia Annisa Br Bangun *et al* (2022) telah melakukan penelitian untuk mendeteksi *patch* luar kayu kelapa yang berkualitas berdasarkan kepadatan serat kayu kelapa melalui citra gambar kayu kelapa. Dalam penelitian ini dilakukan perbandingan metode deteksi tepi untuk mendapatkan hasil optimal dalam pengolahan data. Adapun metode deteksi tepi yang digunakan dalam penelitian ini yaitu metode canny dan sobel. Hasil dari penelitian ini disimpulkan bahwa hasil deteksi tepi menggunakan canny lebih sempurna dibandingkan dengan metode

sobel, dari hasil konversinya terlihat bahwa tepi output gambar terlihat lebih jelas menggunakan canny daripada sobel.

2.3. Naïve Bayes Classifier

Bemando *et al* (2021) telah memaparkan bahwa naïve bayes classifier merupakan supervised classifier yang penerapannya bergantung pada teorema bayes dengan menggunakan asumsi independensi diantara kondisi kelas (*predictors*). Naïve bayes classifier dapat digunakan menggunakan beberapa pendekatan misalnya, Gaussian naïve bayes. Gaussian naïve bayes digunakan untuk atribut dengan nilai nyata, sehingga nilai rata-rata dan standar deviasi pada setiap input dapat dihitung.

Muhathir *et al* (2022) telah melakukan penelitian tentang klasifikasi Autism Spectrum Disorder (ASD) berdasarkan citra wajah dengan beberapa variasi naïve bayes dan pada proses ekstraksi fiturnya menggunakan fitur HoG. Dataset yang digunakan dalam penelitian ini didapatkan langsung dari beberapa SLB (Sekolah Luar Biasa) di sekitar wilayah kota Medan. Jumlah dataset yaitu 100 sampel wajah anak autis dan 100 sampel wajah anak normal dari rentang usia 7-12 tahun. Hasil dari penelitian disimpulkan bahwa dari 3 variasi naïve bayes yang digunakan, yaitu gaussian, Bernoulli, dan multinomial didapatkan nilai tertinggi untuk Bernoulli naïve bayes dengan accuracy bernilai 89,725%, Precision 90,54%, Recall 89,725%, dan F1-Score 89,9%.

Penelitian serupa juga pernah dilakukan oleh Pratama *et al* (2022) dengan objek yang berbeda, yaitu klasifikasi kanker payudara dan kanker paru-paru. Fokus penelitian ini yaitu membandingkan nilai akurasi diantara variasi naïve bayes.

Dataset yang digunakan berjumlah 699 untuk breast cancer dan 309 data untuk lung cancer. Hasil dari penelitian ini didapatkan bahwa untuk klasifikasi kanker payudara, hasil akurasi dan recall paling optimal dalam metode Gaussian NB sebesar 96,2% dan 97,85%. Untuk hasil precision terletak pada Multinomial NB dengan nilai 96,78%. Hasil klasifikasi kanker paru-paru hasil akurasi dan precision tertinggi terletak pada Gaussian NB sebesar 90,95% dan 93,81%. Hasil recall tertinggi pada Multinomial NB yaitu sebesar 100%.

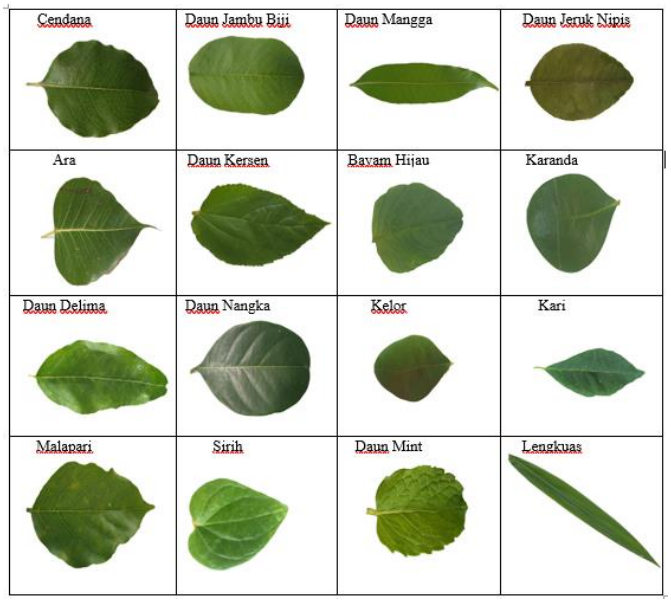
Penelitian lain juga pernah dilakukan oleh Bemando *et al* (2021) dengan memprediksi diagnosis dini penyakit jantung coroner dengan menggunakan metode gaussian naïve bayes dan random forest. Data yang digunakan dalam penelitian ini berasal dari Cleveland database UCI Riwayat penyakit jantung coroner pasien. Hasil dari penelitian menghasilkan nilai akurasi sebesar 85% dan 75%. Hasil precision, F-measure, dan recall dari gaussian naïve bayes lebih tinggi dibandingkan dengan random forest.

BAB III

DESAIN DAN IMPLEMENTASI

3.1. Pengumpulan Data

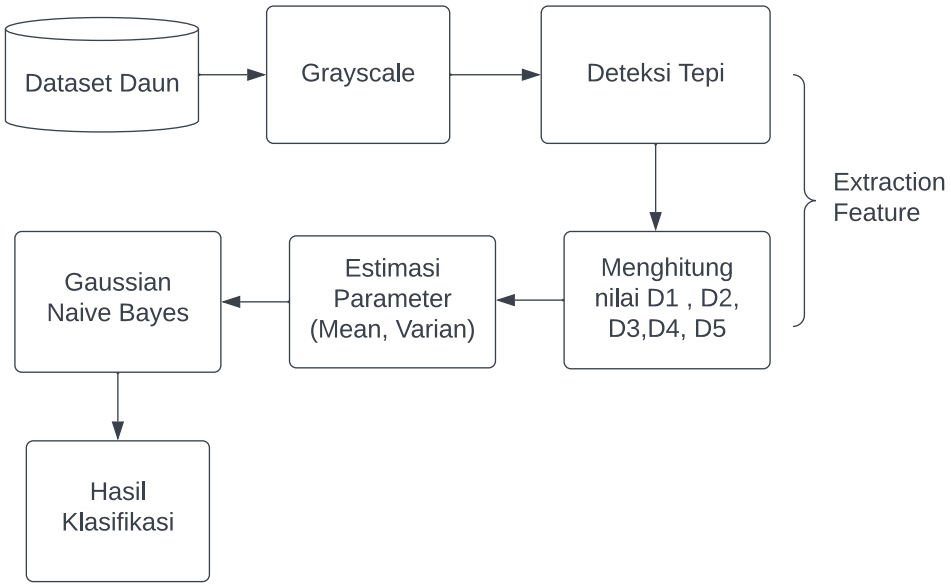
Pada penelitian ini dalam melakukan identifikasi jenis tanaman obat, penulis mengumpulkan beberapa dataset daun tanaman obat yang berasal dari mendeley. Dalam melakukan klasifikasi jenis tanaman obat, daun yang digunakan dalam penelitian ini dibagi menjadi 16 jenis tanaman dengan masing-masing jenis tanaman berjumlah 50 citra gambar. Adapun jenis tanaman yang dikumpulkan yaitu daun sirih, kelor, nangka, jeruk nipis, jambu biji, laos, kunyit, bayam hijau, cendana, bunga sepatu, delima, mondodaki, mangga, kersen, dan malapari. Total seluruh citra daun yang telah dikumpulkan berjumlah 800 citra daun. Data penelitian ini dapat diakses pada link https://bit.ly/dataset-medicinal-plants. Adapun merupakan contoh sampel gambar daun yang digunakan dalam penelitian ditunjukkan pada Gambar 3.1.



Gambar 3. 1 Sampel Data

3.2. Desain Sistem

Adapun desain sistem yang digunakan dalam penelitian ini sebagai berikut pada Gambar 3.2



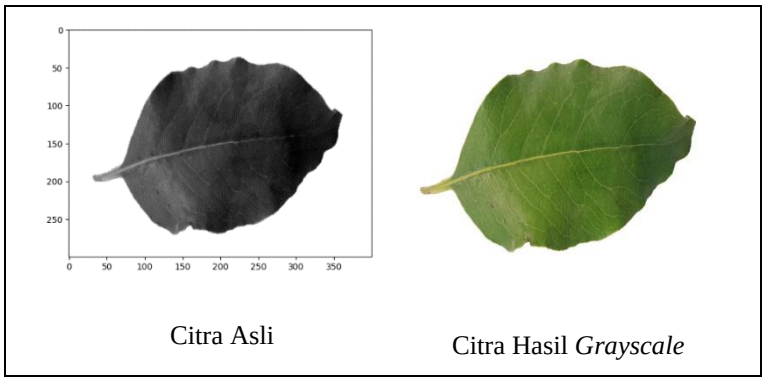
Gambar 3. 2 Desain Sistem

3.2.1. Grayscale

Grayscale merupakan proses mengubah citra dari RGB (Red, Green, Blue menjadi skala keabuan. Pada tahap pertama, data citra diubah menjadi citra *grayscale* agar proses komputasi menjadi lebih sederhana dan lebih mudah diproses karena citra *grayscale* memiliki satu kanal warna yang lebih sedikit jika dibandingkan dengan citra RGB yang memiliki tiga kanal warna. Cara yang paling umum digunakan untuk mengubah citra RGB menjadi *grayscale* adalah dengan rumus :

$$G = 0.2989\,R + 0.5870\,G + 0.1140\,B \quad (3.1)$$

Rumus diatas mengambil nilai dari setiap kanal warna dan menggabungkannya menjadi satu nilai untuk menyatakan tingkat keabuan dari suatu pixel. Implementasi *grayscale* pada *google collab* cukup menggunakan library *opencv* dengan perintah *cv2.IMREAD_GRAYSCALE*. Hasil proses grayscale ini nantinya akan digunakan dalam proses deteksi tepi. Gambar 3.3 menampilkan *image* salah satu jenis daun tanaman obat sebelum dan sesudah dilakukan proses *grayscale*.



Gambar 3. 3 Image daun cendana di *grayscale*

3.2.2. Deteksi Tepi

Pada tahap deteksi tepi ini dilakukan untuk mencari perubahan yang signifikan dalam intensitas suatu citra piksel, proses ini dapat dilakukan dengan menggunakan operator sobel. Tahapan yang dilakukan adalah dengan mengubah citra gambar menjadi citra *grayscale* untuk meningkatkan efisiensi proses, kemudian dilakukan penghalusan citra menggunakan filter *medianBlur()* untuk menghilangkan noise pada citra. Pada penelitian ini untuk mendapatkan hasil tepi yang jelas menerapkan proses dilasi dan erosi untuk mengurangi *noise* pada *image* dan memperjelas tepi-tepi yang tidak terdeteksi dengan mengisi celah-celah pada

garis tepi. Penelitian ini juga menggunakan operator kombinasi antara sobel dan prewitt untuk mendapatkan garis tepi yang lebih jelas. Operator sobel terdiri dari 2 matriks kernel dalam mengukur derajat perubahan intensitas piksel sepanjang sumbu *x* dan *y*. Kernel sumbu *x* dan sumbu *y* didefinisikan sebagai :

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{3.2}$$

Operator prewitt juga mempunyai 2 matriks kernel sumbu *x* dan kernel sumbu *y* yang didefinisikan sebagai berikut :

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad Gy = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.3}$$

Kedua jenis sumbu kernel diatas digunakan untuk menghitung *gradient* setiap piksel gambar. Rumus *gradient* yang digunakan adalah :

$$M = \sqrt{Gx^2 + Gy^2} \tag{3.4}$$

Keterangan :
M = Gradien
Gx = nilai gradien horizontal (sumbu *x*)
Gy = nilai gradien vertikal (sumbu *y*)

Berdasarkan pada Gambar 3.2 dan Gambar 3.3 dapat dilihat bahwa pada operator sobel memiliki kernel matriks dengan bobot lebih berat pada bagian tengah dibandingkan dengan kernel matriks pada operator prewitt. Operator sobel cenderung lebih baik dalam mendeteksi tepi pada arah diagonal sedangkan operator prewitt cenderung mendeteksi tepi dari arah vertikal dan horizontal. *Source code* implementasi *grayscale*, *resize*, dan deteksi tepi kombinasi sobel dan prewitt ditunjukkan pada Gambar 3.4.

```
# Fungsi mengubah gambar menjadi black n white dengan deteksi
tepi Prewitt dan Sobel
def bnwImg(img):
  width = 400
  height = 300

  img = cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
  img = cv2.resize(img, (width, height))

  # melakukan median filter dengan ukuran kernel 3x3
  img = cv2.medianBlur(img, 3)

  sobel_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])
  sobel_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])

  #buat array baru
  img_sobel_x = np.zeros(img.shape,np.float32)
  img_sobel_y = np.zeros(img.shape,np.float32)
  img_prewitt_x = np.zeros(img.shape,np.float32)
  img_prewitt_y = np.zeros(img.shape,np.float32)

  #looping
  for i in range (1, img.shape[0] - 1):
    for j in range(1, img.shape[1] - 1):

      img_sobel_x[i][j]= abs(np.sum(img[i-1:i+2, j-1:j+2] *
      sobel_x))
      img_prewitt_x[i][j]= abs(np.sum(img[i-1:i+2, j-1:j+2] *
      np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])))

      img_sobel_y[i][j]= abs(np.sum(img[i-1:i+2, j-1:j+2] *
      sobel_y))
      img_prewitt_y[i][j]= abs(np.sum(img[i-1:i+2, j-1:j+2] *
      np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])))

  #menghitung gradient
  img_sobel = np.sqrt(img_sobel_x ** 2 + img_sobel_y ** 2)
  img_prewitt = np.sqrt(img_prewitt_x ** 2 + img_prewitt_y ** 2)

  # menggabungkan hasil deteksi tepi sobel dan prewitt
  img_edge = cv2.addWeighted(img_sobel, 0.5, img_prewitt, 0.5,0)

  #dilasi
  kernel = np.ones((3, 3), np.uint8)
  dilation = cv2.dilate(img_edge, kernel, iterations =1)

  #erosi
  erosion = cv2.erode(dilation, kernel, iterations = 1)

  #threshold
  threshold = 200

  #buat array
  img_threshold = np.zeros(erosion.shape, np.uint8)
  #loop
  for i in range(erosion.shape[0]):
```

```
    for j in range(erosion.shape[1]):
          if erosion[i][j] > threshold:
            img_threshold[i][j] = 0
          else:
            img_threshold[i][j] = 255

  result = img_threshold

  return result

path = os.path.join(DATADIR, 'Cendana')
one_img = os.listdir(path)[1]
testBnw = bnwImg(one_img)
plt.imshow(testBnw, 'gray')
plt.show()
```

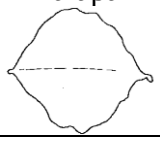
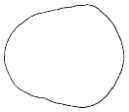
Gambar 3. 4 *Source code*, *grayscale*, *resize*, deteksi tepi

Proses pertama yang dilakukan dalam deteksi tepi yaitu melakukan *resize image*, hal ini dilakukan untuk memudahkan dalam proses komputasi. Image data aslinya mempunyai resolusi 1600 x 1200, kemudian di *resize* dengan ukuran 400x300. Selanjutnya dilakukan *smoothing image* dengan ukuran kernel 3x3menggunakan *library opencv* dengan perintah *cv2.medianBlur(img, 3)*. Selanjutnya yaitu inisialisasi kernel matriks sobel dengan nama *sobel_x* dan *sobel_y* untuk digunakan perhitungan nilai tepi pada *image*, setelah itu dibuat array kosong yang mempunyai ukuran yang sama dengan gambar dengan nama *img_sobel_x*, *img_sobel_y*, *img_prewitt_x*, *dan img_prewitt_y*, array kosong ini nantinya digunakan untuk menyimpan hasil perhitungan deteksi tepi sobel dan prewitt.

Setelah itu, dilakukan perulangan pada setiap piksel untuk setiap image untuk dilakukan perkalian nilai piksel dengan nilai pada kernel matriks sobel dan prewitt dengan ukuran 3x3. Kemudian hasil perkalian tersebut dijumlahkan menggunakan rumus *gradient* pada persamaan 3.4. Setelah itu dilakukan penggabungan hasil deteksi sobel dan prewitt menggunakan fungsi

cv2.addWeighted. Langkah selanjutnya yaitu melakukan dilasi dan erosi untuk menghilangkan *noise* pada image dan mengisi celah-celah piksel yang kosong sehingga hasil tepinya terlihat lebih jelas, implementasi dilasi dan erosi menggunakan fungsi *cv2.dilate()* dan *cv2.erode()* dengan ukuran kernel 3x3. Kemudian, dilakukan inisialisasi nilai *threshold* sebesar 200, lalu dilakukan perulangan untuk menampilkan tepinya dengan kondisi jika piksel image hasil erosi bernilai lebih dari *threshold* maka pikselnya akan bernilai 0, jika nilainya lebih kecil dari threshold maka akan bernilai 255. Tabel 3.1 merupakan hasil deteksi tepi menggunakan kombinasi operator sobel dan operator prewitt.

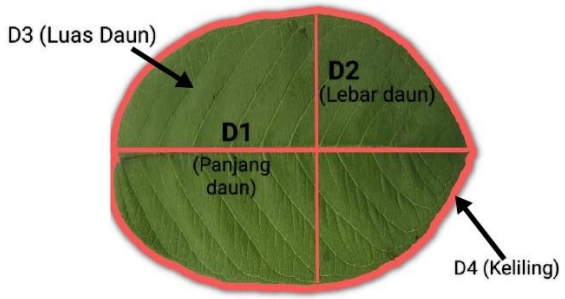
Tabel 3. 1 Hasil Implementasi Deteksi Tepi

Kelor 	Cendana 	Daun Mangga 	Mint 
Daun Jeruk Nipis 	Malapari 	Daun Jambu Biji 	Daun Kersen 
Daun Delima 	Lengkuas 	Bayam 	Ara 
Kari 	Sirih 	Daun Nangka 	Karanda 

3.2.3. Ekstraksi Fitur Bentuk

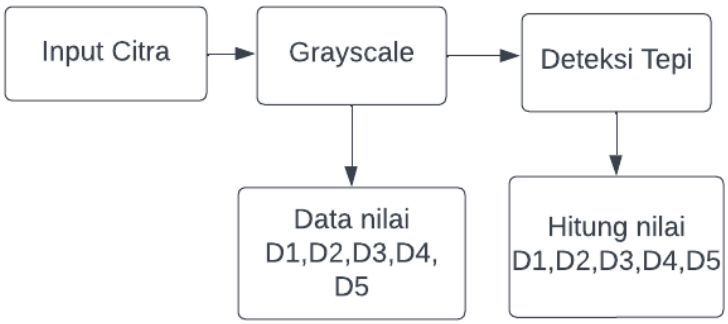
Setelah melakukan deteksi tepi pada citra daun, proses selanjutnya yaitu melakukan ekstraksi fitur, dalam penelitian ini ekstraksi fitur yang digunakan adalah ekstraksi fitur bentuk dengan mencari nilai d1, d2, d3, d4, dan d5. Nilai d1

menyatakan panjang daun, d2 menyatakan lebar daun, d3 menyatakan luas daun, d4 menyatakan perimeter (keliling) daun. Dan d5 menyatakan *slimness* atau kerampingan daun yaitu dengan menghitung rasio antara panjang dan lebar daun. Gambar 3.4 menampilkan pembagian nilai fitur-fitur pada daun.



Gambar 3. 5 Gambar Daun

Alur proses ekstraksi fitur bentuk digambarkan dalam *flowchart* pada gambar 3.4 :



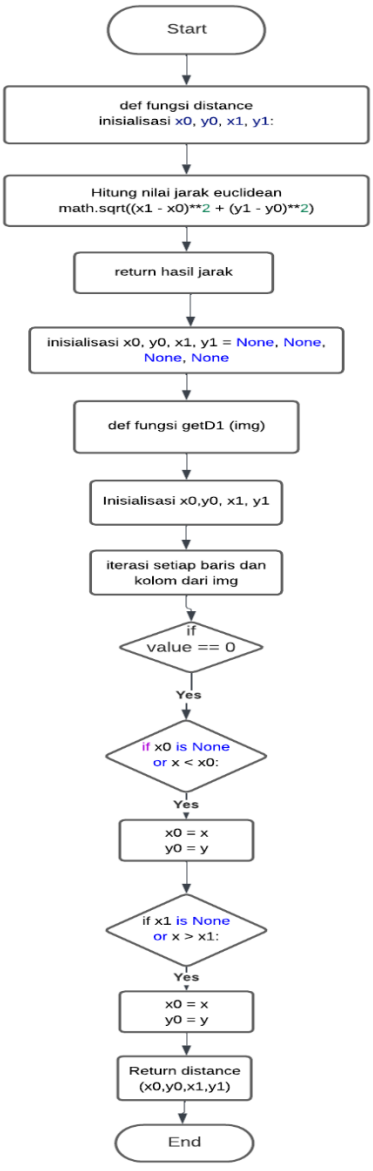
Gambar 3. 6 Alur Proses Ekstraksi Fitur

Nilai d1, d2, d3, d4, d5 tersebut dihitung dengan beberapa rincian metode yaitu:

1. Ukuran daun : Nilai D1 dicari dengan menghitung panjang citra daun dengan mengukur jarak antara titik terjauh pada sumbu x. Rumus dasar yang digunakan yaitu mengunakan rumus jarak *Euclidean* sebagai berikut:

$$d(x,y) = \sqrt{(x2-x1)^2 + (y2-y1)^2} \quad (3.4)$$

Gambar 3.7 menampilkan *flowchart* perhitungan D1 (panjang citra daun) yaitu sebagai berikut:



Gambar 3. 7 *Flowchart* menghitung panjang daun

Berdasarkan *flowchart* pada Gambar 3.6 proses pertama yang dilakukan untuk menghitung nilai D1 (panjang citra daun) yaitu mengimport modul *math* yang digunakan dalam perhitungan algoritma. Setelah itu mendefinisikan fungsi *distance*

dengan memasukkan rumus *euclidean* dalam menghitung jarak antara 2 titik, fungsi *euclidean* akan mengembalikan nilai hasil akar kuadrat dari perhitungan jarak antara titik x dan titik y. Setelah itu, mendefinisikan fungsi *getD1*(*img)* untuk menginisialisasi variabel *x0, y0, x1, y1* dengan nilai *none* untuk menyimpan koordinat titik pada *image*. Kemudian dilakukan iterasi pada setiap baris pada image dengan *for y, row in enumerate(img)* dalam setiap baris lalu dilakukan iterasi setiap elemen x (kolom) untuk dilakukan pengecekan kondisi apakah nilai elemen saat ini bernilai sama dengan 0, jika iya maka dilakukan pengecekan lagi jika nilai *x0* belum diinisialisasi atau jika nilai x saat ini lebih kecil dari *x0* maka nilai *x0* dan *y0* diperbarui dengan nilai *x* dan *y* saat ini.

Kondisi kedua yaitu jika nilai *x1* bernilai none atau nilai *x* saat ini bernilai lebih besar dari *x1*, maka nilai *x1* dan *y1* diperbarui menjadi nilai *x* dan *y* saat ini. Setelah menentukan nilai *x* dan *y* maka dilakukan pemanggilan fungsi *distance* untuk mneghitung jarak antara titik dengan koordinat *x0, y0, x1, y1* yang telah didefinisikan dengan rumus *euclidean*, kemudian hasil perhitungan fungsi *distance* dikembalikan sebagai hasil nilai *getD1*. *Source code* perhitungan nilai D1 (panjang citra daun) ditampilkan pada Gambar 3.8.

```
import math
# rumus jarak (Euclidean)
def distance(x0, y0, x1, y1):
    return math.sqrt((x1 - x0)**2 + (y1 - y0)**2)
#  mendapatkan nilai panjang daun
def getD1(img):
  # mencari koordinat berdasarkan x yng bernilai 0 pada baris
terdekat dan terjauh dari 0
  x0, y0, x1, y1 = None, None, None, None

  # Iterasi melalui setiap baris dari raw_img
  for y, row in enumerate(img):
      # Iterasi melalui setiap elemen dari baris saat ini
      for x, value in enumerate(row):
          # Jika elemen saat ini bernilai 0
```

```
            if value == 0:
                # Jika x0 belum diinisialisasi atau x saat ini
lebih dekat dari x0
                if x0 is None or x < x0:
                    x0 = x
                    y0 = y
                # Jika x1 belum diinisialisasi atau x saat ini
lebih jauh dari x1
                if x1 is None or x > x1:
                    x1 = x
                    y1 = y
    return distance(x0,y0,x1,y1)


# [Test Algoritma] contoh gambar ukuran 5x3, dengan jarak 3
pixel
raw_img = [
    [255, 0, 255, 255, 255],
    [0, 0, 255, 0, 255],
    [255, 255, 255, 255, 255],
]
testImg = np.array(raw_img)
print(getD1(testImg))
```

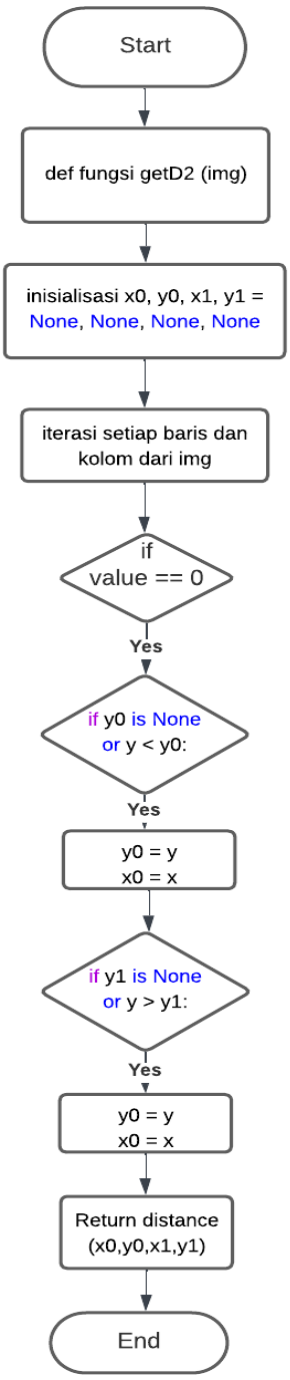
Gambar 3. 8 *Source code* untuk D1 (panjang daun)

Nilai D2 dicari dengan menghitung lebar citra daun dengan cara mengukur jarak antara titik terjauh dan titik terdekat pada sumbu y. Rumus dasar yang digunakan yaitu :

$$Lebar\ daun\ =\ abs\ |\ y\_max\ -\ y\_min\ |\qquad (3.5)$$

Keterangan :
y_max = koordinat *y* dari titik dengan nilai *y* tertinggi
y_min = koordinat *y* dari titik dengan nilai *y* terendah

Gambar 3.7 menampilkan flowchart perhitungan D2 (lebar citra daun) yaitu sebagai berikut:



Gambar 3. 9 *Flowchart* Menghitung lebar Daun

Perhitungan nilai D2 (lebar citra daun) diawali dengan inisialisasi variabel *x0, y0, x1, y1* bernilai *none* untuk meyimpan koordinat titik pada *image*. Kemudian dilakukan iterasi setiap baris *(y)* pada *image*, lalu dalam iterasi tersebut dilakukan

iterasi lagi pada setiap elemen *x* dari baris saat ini. Setelah itu dilakukan pengecekan kondisi jika nilai elemen saat ini bernilai 0 maka dilakukan pengecekan lagi jika nilai *y0* belum diinisialisasi atau nilai *y* lebih kecil dari *y0* maka nilai *y0* dan *x0* akan diperbarui menjadi nilai *y* dan *x* saat ini. Kondisi lain jika *y1* belum diinisialisasi atau nilai *y* saat ini lebih besar dari *y1* maka, nilai *y1* dan *x1* diperbarui menjadi nilai *y* dan *x* saat ini. Setelah proses iterasi dilakukan perhitungan jarak antara 2 titik menggunakan fungsi *distance* yang menerima parameter *x0*, *y0*, *x1*, *y1* yang kemudian hasil perhitungannya dikembalikan sebagai nilai hasil fungsi *getD2*. *Source code* perhitungan nilai D2 (lebar citra daun) ditampilkan pada Gambar 3.10.

```
#  mendapatkan nilai lebar daun
def getD2(img):
   # mencari koordinat berdasarkan x yng bernilai 0 pada baris terdekat dan terjauh dari 0
  x0, y0, x1, y1 = None, None, None, None

  # Iterasi melalui setiap baris dari raw_img
  for y, row in enumerate(img):
      # Iterasi melalui setiap elemen dari baris saat ini
      for x, value in enumerate(row):
          # Jika elemen saat ini bernilai 0
          if value == 0:
             # Jika y0 belum diinisialisasi atau y saat ini lebih dekat dari y0
             if y0 is None or y < y0:
                 y0 = y
                 x0 = x
              # Jika y1 belum diinisialisasi atau y saat ini lebih jauh dari y1
             if y1 is None or y > y1:
                 y1 = y
                 x1 = x
  return distance(x0,y0,x1,y1)


# [Test Algoritma] contoh gambar ukuran 5x3, dengan jarak 2 pixel
raw_img = [
    [255, 0, 255, 255, 255],
    [0, 255, 255, 255, 0],
    [255, 0, 255, 255, 255],
]
testImg = np.array(raw_img)
print(getD2(testImg))
```

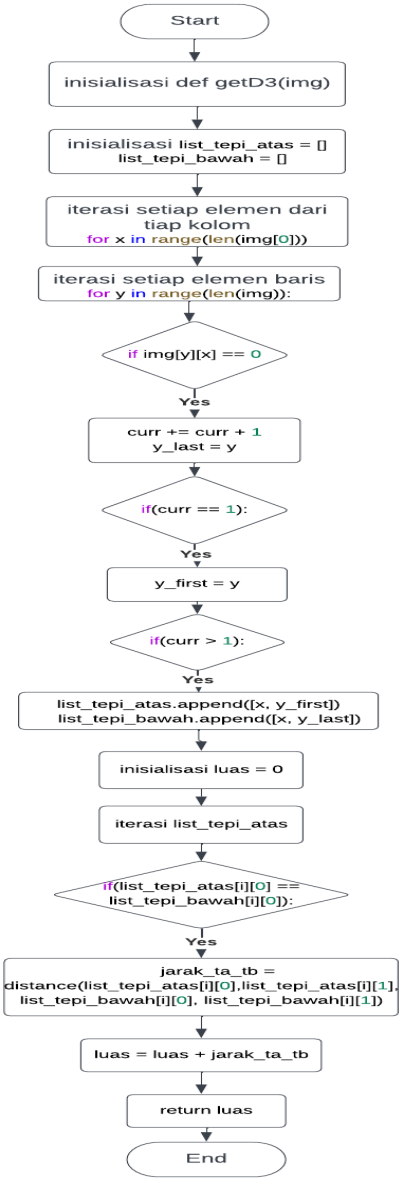
Gambar 3. 10 *Source code* untuk D2 (lebar daun)

Nilai D3 dicari didapatkan dengan menghitung luas daun dengan cara menjumlahkan semua piksel yang berada dalam daun. Rumus dasar yang digunakan yaitu :

$$Luas \ = Jumlah \ pixel \ daun \tag{3.6}$$

Keterangan :
Jumlah piksel daun = jumlah piksel dalam garis tepi daun

Gambar 3.11 menampilkan *flowchart* perhitungan D3 (luas citra daun):



Gambar 3. 11 Flowchart menghitung luas daun

Dalam penerapannya, menghitung luas daun pertama dilakukan dengan mendefinisikan fungsi bernama *getD3* dengan parameter *img*, setelah itu inisialisasi 2 list kosong dengan nama *list_tepi_atas* dan *list_tepi_bawah* untuk menyimpan titik-titik tepi atas dan tepi bawah dari citra daun. Setelah itu dilakukan perulangan setiap elemen kolom pada *image* dengan sintaks *for x in range(len(img[0]))* untuk mengiterasi *index* kolom dari *index* 0 hingga panjang kolom pada *image*, kemudian menginisialisasi variabel *curr*, *y_first*, dan *y_last* bernilai 0. Dalam perulangan tersebut dilakukan iterasi setiap elemen baris pada image mulai dari 0 menggunakan sintaks *for y in range(len(img)).*

Setelah itu, dalam iterasi elemen baris tersebut dilakukan pengecekan nilai, jika nilai elemen image pada baris dan kolom saat ini bernilai 0 maka variabel curr akan di*increment* dengan sintaks *curr += curr +1* untuk menghitung jumlah piksel 0 berturut-turut pada kolom tersebut. Kemudian nilai variabel *y_last* diinisialisasi menjadi nilai *index* baris saat ini dan merupakan *index* terakhir piksel 0 pada kolom tersebut. Namun, kondisi lainnya jika nilai *curr* adalah 1 maka nilai ini adalah piksel 0 pertama pada kolom tersebut dan variabel *y_first* menjadi nilai *index* baris saat ini. Kemudian jika nilai *curr* lebih besar dari 1 dengan kondisi *if(curr > 1)* maka titik koordinat kolom dan baris piksel 0 pertama ditambahkan pada *list_tepi_atas*. Setelah itu, jika titik koordinat kolom dan baris piksel 0 terakhir maka ditambahkan pada *list_tepi_bawah*.

Setelah proses iterasi tersebut dilakukan inisialisasi variabel luas dengan nilai 0, variabel ini akan digunakan sebagi akumulasi nilai luas daun. Kemudian dilakukan iterasi tiap elemen pada *list_tepi atas* untuk mengecek jika nilai kolom

list_tepi_atas sama dengan nilai kolom pada *list_tepi_bawah* maka variabel *jarak_ta_tb* akan memanggil fungsi *distance* untuk menghitung jarak tepi atas dengan tepi bawah pada kolom yang sama dan hasil perhitungan *jarak_ta_tb* akan ditambahkan sebagai hasil nilai luas. *Source code* perhitungan nilai D3 (luas citra daun) ditampilkan pada Gambar 3.12.

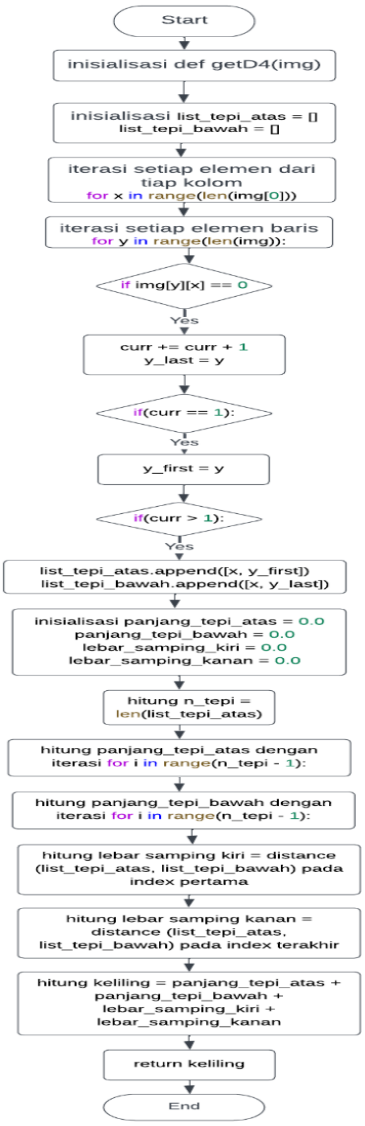
```
#  mendapatkan nilai luas daun
def getD3(img):
  # Iterasi melalui setiap baris dari raw_img
  list_tepi_atas = []
  list_tepi_bawah = []

  # Iterasi melalui setiap elemen dari tiap kolom
  for x in range(len(img[0])):
        # Iterasi melalui setiap elemen dari baris saat ini
        curr = 0
        y_first = None
        y_last = None
        for y in range(len(img)):
          if img[y][x] == 0:
            curr += curr + 1
            y_last = y
            if(curr == 1):
              y_first = y

        if(curr > 1):
          list_tepi_atas.append([x, y_first])
          list_tepi_bawah.append([x, y_last])
  luas = 0
  for i in range(len(list_tepi_atas)):
    if(list_tepi_atas[i][0] == list_tepi_bawah[i][0]):
      jarak_ta_tb =
distance(list_tepi_atas[i][0],list_tepi_atas[i][1],
list_tepi_bawah[i][0], list_tepi_bawah[i][1])
      luas = luas + jarak_ta_tb
  # print(luas)
  return luas


# [Test Algoritma] contoh gambar ukuran 5x3, dengan jarak 2
pixel
raw_img = [
    [255, 0, 255, 0, 255],
    [0, 255, 255, 255, 0],
    [255, 0, 255, 0, 255],
]
testImg = np.array(raw_img)
print(getD3(testImg))
```

Gambar 3. 12 *Source code* hitung D3 (luas daun)

2. Perimeter daun: Nilai D4 didapatkan dengan menghitung jumlah piksel yang merupakan bagian dari garis tepi daun. Rumus dasar yang digunakan bisa menggunakan rumus jarak Euclidean, dimana keliling daun dinyatakan sebagai berikut :

$$Keliling\ daun = Jumlah\ pixel\ pada\ tepi\ daun \quad (3.7)$$

Gambar 3.13 menampilkan *flowchart* perhitungan D4 (keliling citra daun) yaitu:



Gambar 3. 13 *Flowchart* menghitung keliling daun

Dalam menghitung perimeter (keliling daun) pertama dilakukan yaitu mendefinisikan fungsi bernama *getD4* dengan parameter *img*, setelah itu inisialisasi 2 list kosong dengan nama *list_tepi_atas* dan *list_tepi_bawah* untuk menyimpan titik-titik tepi atas dan tepi bawah dari citra daun. Setelah itu dilakukan perulangan setiap elemen kolom pada *image* dengan sintaks *for x in range(len(img[0]))* untuk mengiterasi *index* kolom dari *index* 0 hingga panjang kolom pada *image*, kemudian menginisialisasi variabel *curr*, *y_first*, dan *y_last* bernilai 0. Dalam perulangan tersebut dilakukan iterasi setiap elemen baris pada *image* mulai dari 0 menggunakan sintaks *for y in range(len(img))*.

Setelah itu, dalam iterasi elemen baris tersebut dilakukan pengecekan nilai, jika nilai elemen image pada baris dan kolom saat ini bernilai 0 maka variabel *curr* akan diincrement dengan sintaks *curr += curr +1* untuk menghitung jumlah piksel 0 berturut-turut pada kolom tersebut. Kemudian nilai variabel *y_last* diinisialisasi menjadi nilai index baris saat ini dan merupakan index terakhir piksel 0 pada kolom tersebut. Namun, kondisi lainnya jika nilai *curr* adalah 1 maka nilai ini adalah piksel 0 pertama pada kolom tersebut dan variabel *y_first* menjadi nilai index baris saat ini. Kemudian jika nilai *curr* lebih besar dari 1 dengan kondisi *if(curr > 1)* maka titik koordinat kolom dan baris piksel 0 pertama ditambahkan pada *list_tepi_atas*. Setelah itu, jika titik koordinat kolom dan baris piksel 0 terakhir maka ditambahkan pada *list_tepi_bawah*.

Setelah proses iterasi tersebut dilakukan inisialisasi variabel *panjang_tepi_atas,panjang_tepi_bawah,lebar_samping_kiri,lebar_samping_kanan* bernilai 0. Selanjutnya dilakukan perhitungan panjang tepi atas dengan

mengiterasi setiap elemen pada list_tepi_atas menggunakan sintaks *for i in range(n_tepi -1)*. Pada setiap iterasi dilakukan penjumlahan hasil perhitungan jarak antara titik tepi atas saat ini dengan titik tepi atas berikutnya dengan memanggil fungsi *distance*, hasil perhitungan disimpan dalam variabel *panjang_tepi_atas*.

Perhitungan panjang tepi bawah dilakukan dengan cara serupa dengan menggunakan *list_tepi_bawah* dan hasilnya disimpan dalam variabel *panjang_tepi_bawah*. Perhitungan lebar samping kiri dan lebar samping kanan menggunakan fungsi *distance* untuk menghitung jarak antara titik pada tepi atas dengan titik tepi bawah. Perbedaan perhitungan terletak pada letak indeks, jika menghitung lebar samping kiri menggunakan indeks pertama pada *list_tepi_atas* dan *list_tepi_bawah*, sedangkan untuk menghitung lebar samping kanan menggunakan indeks terakhir pada *list_tepi_atas* dan *list_tepi_bawah*. Tahap terakhir menghitung keliling citra daun dengan menjumlahkan semua nilai panjang tepi atas, panjang tepi bawah, lebar samping kiri, dan lebar samping kanan. Hasil perhitungan keliling disimpan pada variabel keliling yang kemudian dikembalikan sebagai output fungsi *getD4*. *Source code* perhitungan nilai D4 (keliling citra daun) ditampilkan pada Gambar 3.14.

```
#  mendapatkan nilai keliling daun
def getD4(img):
  # Iterasi melalui setiap baris dari raw_img
  list_tepi_atas = []
  list_tepi_bawah = []

  # Iterasi melalui setiap elemen dari tiap kolom
  for x in range(len(img[0])):
        # Iterasi melalui setiap elemen dari baris saat ini
        curr = 0
        y_first = None
        y_last = None
        for y in range(len(img)):
          if img[y][x] == 0:
            curr += curr + 1
```

```
            y_last = y
            if(curr == 1):
              y_first = y

        if(curr > 1):
          list_tepi_atas.append([x, y_first])
          list_tepi_bawah.append([x, y_last])


  panjang_tepi_atas = 0.0
  panjang_tepi_bawah = 0.0
  lebar_samping_kiri = 0.0
  lebar_samping_kanan = 0.0
  n_tepi = len(list_tepi_atas)


  for i in range(n_tepi - 1):
      panjang_tepi_atas = panjang_tepi_atas +
      distance(list_tepi_atas[i][0],list_tepi_atas[i][1],
      list_tepi_atas[i+1][0], list_tepi_atas[i+1][1])

  for i in range(n_tepi - 1):
      panjang_tepi_bawah = panjang_tepi_bawah +
      distance(list_tepi_bawah[i][0],list_tepi_bawah[i][1],
      list_tepi_bawah[i+1][0], list_tepi_bawah[i+1][1])

  lebar_samping_kiri = distance(list_tepi_atas[0][0],
  list_tepi_atas[0][1],
  list_tepi_bawah[0][0],list_tepi_bawah[0][1])
  lebar_samping_kanan = distance(list_tepi_atas[n_tepi - 1][0],
  list_tepi_atas[n_tepi - 1][1], list_tepi_bawah[n_tepi - 1][0],
  list_tepi_bawah[n_tepi - 1][1])

  keliling = panjang_tepi_atas + panjang_tepi_bawah +
  lebar_samping_kiri + lebar_samping_kanan

  return keliling


# [Test Algoritma] contoh gambar ukuran 5x3, dengan jarak 2
pixel
raw_img = [
    [255, 0, 0, 0, 255],
    [255, 0, 255, 0, 255],
    [255, 0, 0, 0, 255],
]
testImg = np.array(raw_img)
print(getD4(testImg))
```

Gambar 3. 14 *Source code* hitung D4 (keliling daun)

3. *Slimness* daun : Nilai D5 dapat dihitung dengan mengukur rasio perbandingan antara panjang daun dan lebar daun. Rumus dasar yang digunakan untuk mengukur kerampingan daun yaitu :

$$Slimness = \frac{P_X}{L_X} \tag{3.8}$$

Keterangan :
P_X= Nilai panjang daun
L_X= Nilai lebar daun

Implementasi perhitungan nilai D5 yaitu dengan mendefinisikan fungsi *getD5* dengan parameter nilai D1 dan D2 kemudian fungsi *getD5* akan mengembalikan hasil pembagian antara nilai D1 dan D2. *Source code* perhitungan nilai D5 (rasio citra daun) ditampilkan pada Gambar 3.15.

```
#  mendapatkan nilai slimness daun
def getD5(d1, d2):
    return d1/d2
```

Gambar 3. 15 *Source code* hitung *slimness* (rasio daun)

Tabel 3.2 merupakan implementasi ekstraksi fitur d1, d2, d3, d4, dan d5 sebagai fitur bentuk pada citra daun tanaman obat.

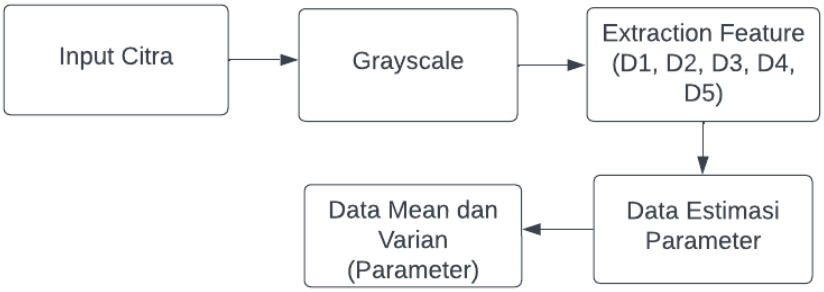
Tabel 3. 2 Hasil Implementasi Ekstraksi Fitur Bentuk

index	d1_panjang	d2_lebar	d3_luas	d4_keliling	d5_rasio	Kelas
0	296	262.6119	53407	925.1705	1.127139	Cendana
50	335.0239	255.7069	66264	970.1302	1.310187	Jambu Biji
100	235.4167	252	42923	828.9277	0.934193	Mint
150	282.1135	235.1361	42833	832.709	1.199788	Bayam Hijau
200	346.0925	260.0308	58859	983.1967	1.330967	Jeruk Nipis
250	379.6972	159.0126	30902	882.1737	2.387844	Mangga
300	281.0071	265.0019	40371	1073.479	1.060397	Ara
350	278.8422	224.9556	40236	814.3119	1.239544	Kari
400	205.7596	179.6274	24943	595.5263	1.14548	Kelor
450	353.5534	200	45526	885.1585	1.767767	Delima
500	316.2278	267.9048	50913	913.9546	1.180373	Malapari
550	337.3781	241.8677	51840	924.3207	1.394887	Sirih
600	340.802	212.7628	46758	886.2388	1.601793	Nangka
650	351.0912	196.0867	41698	903.3676	1.790489	Kersen

index	d1_panjang	d2_lebar	d3_luas	d4_keliling	d5_rasio	Kelas
700	274.2207	251.0319	46490	822.0786	1.092374	Karanda
750	495.6178	493.8117	21513	1066.051	1.003657	Lengkuas

3.2.4. Estimasi Parameter

Proses lanjutan setelah menghitung nilai D1, D2, D3, D4, dan D5 yaitu melakukan estimasi paramater. Estimasi parameter merupakan estimasi yang digunakan untuk melakukan pendugaan terhadap suatu populasi dalam suatu sampel. Dalam ilmu statistika, estimasi merupakan suatu proses yang menggunakan sebuah estimator misalnya dalam statistik (rata-rata, varian, persentase, dan lain-lain) yang digunakan untuk mengestimasi suatu parameter. Pada saat berhadapan dengan data kontinu, asumsi khas yang digunakan yaitu distribusi Gaussian dengan parameter diambil dari nilai rata-rata dan varian pada sampel (Saraswati, 2011). Algoritma yang digunakan dalam penelitian ini yaitu dengan menggunakan Gaussian Naïve Bayes. Berikut alur proses estimasi parameter pada Gambar 3.16 :



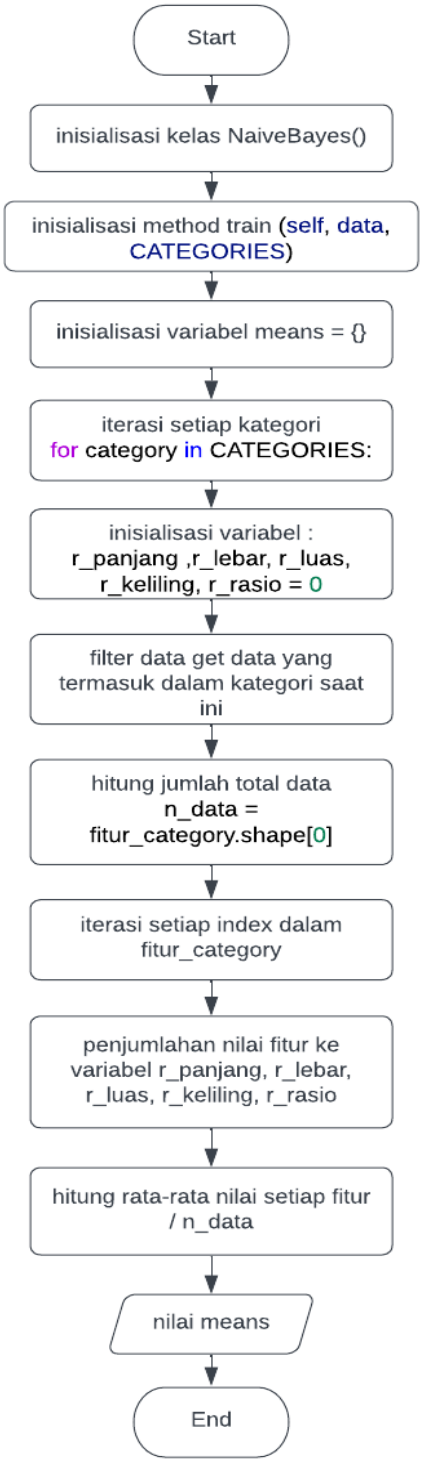
Gambar 3. 16 Alur Proses Estimasi Parameter

Persamaan yang digunakan dalam menghitung nilai *mean* yaitu:

$$\mu_{ic} = \frac{1}{n} \sum_{j=1}^{n} X_{ij} \tag{3.9}$$

Keterangan :
n = jumlah citra untuk tiap kelas
$\sum x_{ij}$= jumlah fitur ke-i hingga n untuk tiap kelas

Gambar 3.17 menampilkan flowchart perhitungan estimasi parameter rata-rata sebagai berikut:



Gambar 3. 17 Flowchart hitung rata-rata

Flowchart pada Gambar 3.17 merupakan bentuk flowchart perhitungan nilai rata-rata, tahap pertama yaitu inisialisasi variabel kamus kosong bernama *self.means* untuk menyimpan nilai rata-rata setiap fitur pada setiap kategori. Selanjutnya, dilakukan iterasi setiap kategori yang di dalam perulangan tersebut sudah diinisialisasi variabel *r_panjang*, *r_lebar*, *r_luas*, *r_keliling*, dan *r_rasio* dengan nilai 0 untuk menjumlahkan nilai fitur-fitur setiap data dalam kategori tersebut. Selanjutnya dilakukan penyaringan data untuk mendapatkan data yang hanya termasuk dalam kategori saat ini menggunakan sintaks *.loc* yang disimpan dalam variabel *fitur_category*. Perhitungan jumlah total data dalam kategori saat ini menggunakan sintaks *.shape[0]* pada *fitur_category* dan hasilnya disimpan dalam variabel *n_data*.

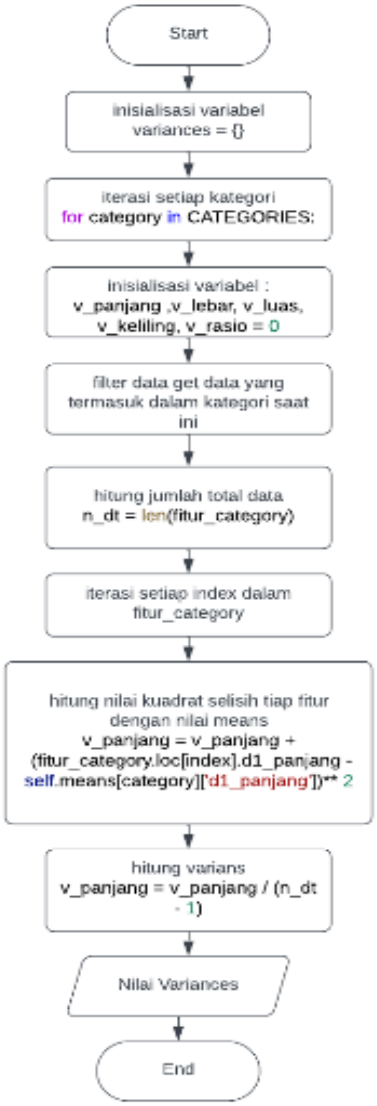
Selanjutnya, dilakukan iterasi setiap indeks data dalam *fitur_category* untuk dilakukan penjumlahan nilai setiap fitur pada setiap kategori, setelah proses iterasi selesai dilakukan perhitungan rata-rata yaitu nilai hasil penjumlahan setiap fitur pada setiap kategori dibagi dengan jumlah total data yang disimpan dalam variabel *n_data*. Hasil perhitungan rata-rata disimpan dalam variabel *r_panjang*, *r_lebar*, *r_luas*, *r_keliling*, dan *r_rasio*. Setelah perhitungan rata-rata selesai, tahap selanjutnya membuat kamus untuk menyimpan informasi fitur dan nilai rata-ratanya untuk setiap kategori dalam variabel *self.means[category]*.

Sedangkan persamaan yang digunakan untuk menghitung nilai *varian* yaitu ditunjukkan pada persamaan 3.10 berikut :

$$\sigma_{iC} = \frac{1}{n-1} \sum_{i=j}^{n} (X_{ij} - \mu_{ic})^2 \qquad (3.10)$$

Keterangan :
n = jumlah citra untuk tiap kelas
$\sum x_{ij}$= jumlah fitur ke-i hingga n untuk tiap kelas
μ_{ic}= *mean* fitur ke-i untuk kelas C

Perhitungan nilai *varian* menggunakan rumus pada persamaan 3.10 ditunjukkan dalam Gambar 3.18 yaitu flowchart menghitung nilai *varian*.



Gambar 3.18 *Flowchart* hitung varian

Flowchart pada Gambar 3.18 merupakan flowchart menghitung *varian*. Langkah pertama yaitu inisialisasi variabel kamus kosong bernama *self.variances*

untuk menyimpan nilai *varian* setiap fitur pada setiap kategori. Selanjutnya, dilakukan iterasi setiap kategori yang di dalam perulangan tersebut sudah diinisialisasi variabel *v_panjang*, *v_lebar*, *v_luas*, *v_keliling*, dan *v_rasio* dengan nilai 0 untuk menjumlahkan nilai fitur-fitur setiap data dalam kategori tersebut. Selanjutnya dilakukan penyaringan data untuk mendapatkan data yang hanya termasuk dalam kategori saat ini menggunakan sintaks *.loc* yang disimpan dalam variabel *fitur_category*. Perhitungan jumlah total data dalam kategori saat ini menggunakan sintaks *len* pada *fitur_category* dan hasilnya disimpan dalam variabel *n_dt*. Selanjutnya, dilakukan iterasi setiap indeks dalam *fitur_category* untuk dilakukan akumulasi nilai kuadrat selisih nilai setiap fitur dalam *fitur_category* dengan nilai rata-rata yang disimpan dalam *self.means*, setelah proses iterasi selesai, kemudian dilakukan perhitungan varian yaitu nilai hasil akumulasi setiap fitur pada setiap kategori dibagi dengan jumlah total data yang disimpan dalam variabel *n_dt*.

Hasil perhitungan varian disimpan dalam variabel *v_panjang*, *v_lebar*, *v_luas*, *v_keliling*, dan *v_rasio*. Setelah perhitungan varian selesai, tahap selanjutnya membuat kamus untuk menyimpan informasi fitur dan nilai variannya untuk setiap kategori dalam variabel *self.variances[category]*. *Source code* perhitungan nilai rata-rata dan varian untuk setiap fitur pada setiap kategori ditampilkan pada Gambar 3.19.

```
class NaiveBayes:
  def train(self, data, CATEGORIES):
    self.means = {} # kode untuk menghitung rata-rata
    for category in CATEGORIES:
      r_panjang = 0
      r_lebar = 0
      r_luas = 0
      r_keliling = 0
      r_rasio = 0
      fitur_category = data.loc[data['y'] == category]
```

```
      n_data = fitur_category.shape[0]

      for index in fitur_category.index:
        r_panjang = r_panjang +
        fitur_category.loc[index]['d1_panjang']
        r_lebar = r_lebar +
fitur_category.loc[index]['d2_lebar']
        r_luas = r_luas + fitur_category.loc[index]['d3_luas']
        r_keliling = r_keliling +
        fitur_category.loc[index]['d4_keliling']
        r_rasio = r_rasio +
fitur_category.loc[index]['d5_rasio']
      r_panjang = r_panjang/n_data
      r_lebar = r_lebar/n_data
      r_luas = r_luas/n_data
      r_keliling = r_keliling/n_data
      r_rasio = r_rasio/n_data
      self.means[category] =  {
          'd1_panjang' : r_panjang,
          'd2_lebar' : r_lebar,
          'd3_luas' : r_luas,
          'd4_keliling' : r_keliling,
          'd5_rasio' : r_rasio,
      }
    self.variances = {} # kode untuk menghitung varians
    for category in CATEGORIES:
      v_panjang = 0
      v_lebar = 0
      v_luas = 0
      v_keliling = 0
      v_rasio = 0
      fitur_category = fitur.loc[fitur['y']== category]
      n_dt = len(fitur_category)
      for index in fitur_category.index:
        v_panjang = v_panjang +
       (fitur_category.loc[index].d1_panjang -
       self.means[category]['d1_panjang'])** 2
        v_lebar = v_lebar+  (fitur_category.loc[index].d2_lebar
       - self.means[category]['d2_lebar'])** 2
        v_luas = v_luas +  (fitur_category.loc[index].d3_luas -
       self.means[category]['d3_luas'])** 2
        v_keliling = v_keliling +
       (fitur_category.loc[index].d4_keliling -
       self.means[category]['d4_keliling'])** 2
        v_rasio = v_rasio +  (fitur_category.loc[index].d5_rasio
       - self.means[category]['d5_rasio'])** 2
      v_panjang = v_panjang / (n_dt - 1)
      v_lebar = v_lebar / (n_dt - 1)
      v_luas = v_luas / (n_dt - 1)
      v_keliling = v_keliling / (n_dt - 1)
      v_rasio = v_rasio / (n_dt - 1)

      self.variances[category] =  {
        'd1_panjang' : v_panjang,
        'd2_lebar' : v_lebar,
        'd3_luas' : v_luas,
```

```
        'd4_keliling' : v_keliling,
        'd5_rasio' : v_rasio,
    }
```

Gambar 3. 19 *Source code* hitung rata-rata dan varians

Tabel 3.3 merupakan implementasi estimasi parameter rata-rata dan *varians* untuk kelas cendana yaitu sebagai berikut :

Tabel 3. 3 Hasil Estimasi Parameter

id	d1_panjang	d2_lebar	d3_luas	d4_keliling	d5_rasio	kelas
0	296	262.6119	53407	925.1705	1.127139	Cendana
1	341.0059	248.8956	49815	965.9474	1.370076	Cendana
2	333.4082	243.0514	51359	930.1416	1.37176	Cendana
3	363.0014	228.0022	54292	987.9581	1.592096	Cendana
4	336.3941	253.16	54594	949.5545	1.328781	Cendana
…	…	…	…	…	…	…
45	312.1939	248.1149	52382	922.8033	1.258263	Cendana
46	340.5877	247.002	57282	968.3018	1.378886	Cendana
47	349.5168	223.0807	49284	1208.949	1.566773	Cendana
48	344.3312	221.822	47911	902.6232	1.552286	Cendana
49	339.3317	236.3049	49156	923.4177	1.435991	Cendana
mean	333.4997	243.6425	51961.73	1050.941	1.376289	
varians	340.2469	249.9426	18808940	401151.4	0.019461	Cendana

3.2.5. Naïve Bayes Classifier

Naïve bayes merupakan salah satu metode klasifikasi yang menggunakan teori terhadap probabilitas dan statistika. Algoritma ini berasumsi bahwa setiap fitur mempunyai distribusi Gaussian atau distribusi normal yang berarti distribusi probabilitas dalam menjalankan data kontinu. Persamaan teorema bayes dilihat pada persamaan 2.5.

$$P(C_i|x_i) = \frac{P(C_i)\,P(x_i|C_i)}{P(x_i)} \tag{3.11}$$

Variabel C_i merepresentasikan sebuah kelas dan $x_i$merepresentasikan fitur-fitur kelas ke -i. Persamaan rumus diatas dapat ditulis sebagai berikut :

$$Posterior = \frac{Prior\ x\ likelihood}{Evidence} \tag{3.12}$$

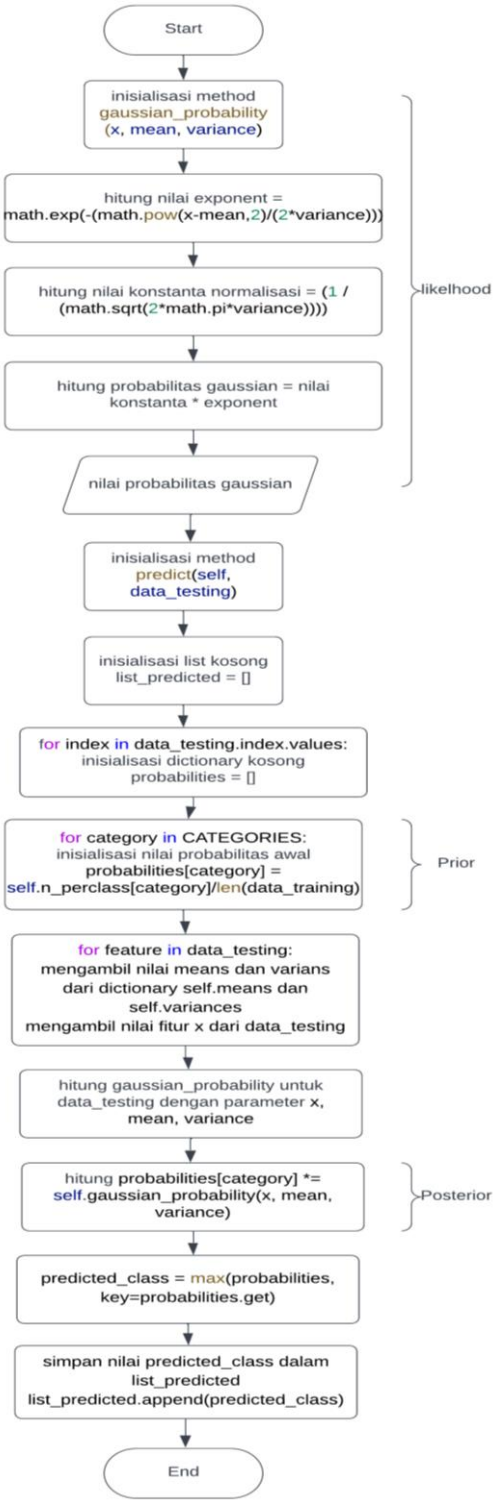
Nilai *evidence* bernilai tetap untuk setiap kelas dalam satu sampel sehingga perhitungan nilai *evidence* tidak perlu dilakukan. Hasil perhitungan nilai posterior digunakan untuk menentukan kelas yang paling mungkin terjadi berdasarkan data yang diberikan. Semakin besar nilai posterior maka semakin mungkin bahwa data x termasuk dalam kelas C_i. Persamaan rumus *gaussian naïve bayes* dapat ditulis seperti berikut (Duin & Tax, 2005) :

$$P(x_i|\mu_{iC}\ ,{\sigma^2}_{iC}) = \mathrm{N}(x_i;\mu_{iC}\ ,{\sigma^2}_{iC})\ = \frac{1}{\sqrt{2\pi\ {\sigma^2}_{iC}}}\ exp\left(-\frac{(x_i-\mu_{iC}\ )^2}{2\ {\sigma^2}_{iC}}\right) \tag{3.13}$$

Proses klasifikasi menggunakan *gaussian naïve bayes* tahap pertama yaitu dilakukan penginputan nilai pada masing-masing fitur yang digunakan kemudian dihitung nilai *mean* dan *varian* pada setiap fitur untuk setiap kelas. Setelah itu dilakukan perhitungan *gaussian* pada rumus 3.13 diatas. Setelah dilakukan perhitungan tersebut maka dilakukan perhitungan posterio*r* dengan dilakukan perhitungan prior atau probabilitas kelas kemudian dikalikan dengan nilai hasil perhitungan *gaussian* yang terdapat pada rumus 3.13. probabilitas kelas pada penelitian ini diasumsikan sama pada setiap kelas. Hasil perhitungan nilai posterior kemudian dilakukan perhitungan argument maksimal untuk dicari data tersebut masuk dalam kelas tertentu menggunakan persamaan 3.8 sebagai berikut :

$$C_i = arg\ max(P_j) \tag{3.14}$$

Proses klasifikasi menggunakan *gaussian naïve bayes* ditunjukkan dalam *flowchart* pada Gambar 3.20



Gambar 3. 20 *Flowchart Naive Bayes*

Tahap klasifikasi menggunakan metode *gaussian naïve bayes* yang pertama dilakukan yaitu mendapatkan nilai rata-rata dan varians pada setiap fitur untuk setiap kategori yang disimpan dalam kamus *self.means* dan *self.variances*. Selanjutnya, membuat *method gaussian_probability* dengan 3 parameter yaitu *x, mean, dan variance*. *Method* ini digunakan dalam perhitungan probabilitas *gaussian* atau probabilitas *likelihood*. Dalam *method gaussian* dilakukan perhitungan nilai eksponen menggunakan rumus $-\frac{(x_i-\mu_{iC})^2}{2\,\sigma^2{}_{iC}}$. Kemudian dilakukan perhitungan konstanta normalisasi dengan rumus $\frac{1}{\sqrt{2\pi\,\sigma^2{}_{iC}}}$ lalu mengembalikan nilai probabilitas *gaussian* dengan mengalikan hasil perhitungan konstanta normalisasi dengan nilai eksponen.

Langkah selanjutnya yaitu mendefinisikan *method predict* yang menerima parameter *data_testing*, kemudian inisialisasi list kosong bernama *list_predicted* untuk menyimpan hasil prediksi. Setelah itu dilakukan iterasi pada setiap indeks dalam *data_testing*, di dalam iterasi dibuat inisialisasi kamus kosong bernama *probabilities* untuk menyimpan probabilitas pada setiap kategori. Setelah itu dilakukan iterasi pada setiap kategori yang kemudian diinisialisasi nilai perhitungan probabilitas awal atau *prior* dengan sintaks *self.n_perclass[category]/len(data_training)*. Sintaks tersebut mengambil nilai jumlah data *training* pada setiap kategori lalu dibagi dengan jumlah total data *training* untuk semua kelas. Selanjutnya dilakukan pengambilan nilai rata-rata dan varian berdasarkan kategori dan fitur saat ini dalam *data_testing* dengan memanggil kamus *self.means* dan *self.variances*. Setelah itu mendefinisikan nilai *x* yang merupakan nilai fitur pada indeks ke *index* dan fitur saat ini.

Langkah selanjutnya yaitu menghitung nilai probabilitas posterior dengan mengalikan nilai probabilitas awal dengan nilai probabilitas *gaussian* untuk setiap fitur menggunakan parameter *x, mean, variance* dan hasilnya disimpan dalam *probabilities[category]*. Selanjutnya setelah ditemukan hasilnya maka dicari kategori yang mempunyai nilai probabilitas tertinggi menggunakan persamaan 3.14 kemudian menambahkan kelas yang diprediksi dalam *list_predicted*. Akhirnya *method predict* akan menghasilkan daftar prediksi berdasarkan data testing yang ditentukan. *Source code* proses klasifikasi menggunakan *gaussian naïve bayes* ditunjukkan pada Gambar 3.21.

```
class NaiveBayes:
  def gaussian_probability(self, x, mean, variance):
    exponent = math.exp(-(math.pow(x-mean,2)/(2*variance)))
    return (1 / (math.sqrt(2*math.pi*variance))) * exponent

  def predict(self, data_testing):
    list_predicted =[]
    for index in data_testing.index.values:
      probabilities = {}
      for category in CATEGORIES:
        probabilities[category] = 50/len(data_training)
        for feature in data_testing:
          mean = self.means[category][feature]
          variance = self.variances[category][feature]
          x = data_testing.iloc[index][feature]
          probabilities[category] *= self.gaussian_probability(x,
         mean, variance)

      predicted_class = max(probabilities, key=probabilities.get)
      list_predicted.append(predicted_class)
    return list_predicted
```

Gambar 3. 21 Source code Gaussian Naive Bayes

BAB IV

UJI COBA DAN PEMBAHASAN

4.1. Langkah-langkah Uji Coba

4.1.1. Data Pengujian

Data uji terdiri dari 16 kelas dengan masing-masing kelas berjumlah 50 data citra daun sehingga total dataset citra daun berjumlah 800 gambar. Dataset yang digunakan didapatkan dari sumber Mendeley. Selanjutnya, dataset tersebut dilakukan beberapa proses seperti deteksi tepi, ekstraksi fitur dan estimasi parameter rata-rata dan varians. Setelah proses pelatihan tersebut dilakukan, data dipisah menjadi data latih dan data uji dengan beberapa variasi rasio 70:30, 80:20, dan 90:10 dengan jumlah data training lebih besar daripada data testing. Tabel 4.1 merupakan contoh dataset yang dikumpulkan :

Tabel 4. 1 Data Penelitian

Kelas	Jumlah
Ara	50 data
Bayam Hijau	50 data
Daun Delima	50 data
Daun Jambu Biji	50 data
Daun Jeruk Nipis	50 data
Kelor	50 data
Kersen	50 data
Karanda	50 data
Kari	50 data
Cendana	50 data
Lengkuas	50 data
Mint	50 data
Daun Mangga	50 data
Malapari	50 data
Nangka	50 data
Sirih	50 data

Proses pembagian data latih dan data uji (split data) dilakukan dengan menginisialisasi variabel rasio untuk variasi pengujian sebesar 0.7, 0.8, dan 0.9. Pembagian dataset ini juga dibagi sama besar jumlahnya pada masing-masing kelas, misalnya untuk rasio data training 70 dan data testing 30, maka pembagian jumlah data per kelasnya adalah 35 data training dan 15 data testing karena jumlah data per kelas maisng-masing total 50. Kemudian untuk rasio 80:20 berarti pembagian data per kelasnya yaitu 40 data training dan 10 data testing. Begitu pula untuk rasio 90:10 pembagian data per kelasnya sebesar 45 data training dan 5 data testing.

Pembagian dataset dibagi untuk beberapa variasi agar dapat memperoleh rasio pembagian data terbaik sehingga nantinya bisa mengoptimalkan hasil akurasi. Setelah itu juga menginisialisasi nilai *random seed* untuk mengacak urutan pembagian data pada dataset. Penulis menggunakan nilai random 5 untuk ketiga variasi pembagian data agar dapat memberikan pembagian yang sama setiap program dijalankan lagi. Tabel 4.2 menunjukkan jumlah pembagian data latih dan data uji untuk ketiga variasi rasio.

Tabel 4. 2 Rasio Pembagian Data

Ratio	**Seluruh Data**		**Per kelas**		**Random Seed**
	Training	**Testing**	**Training**	**Testing**	
0,7	560	240	35	15	5
0,8	640	160	40	10	5
0,9	720	80	45	5	5

Tahap pengujian selain menggunakan variasi rasio pembagian data, proses pengujian juga menggunakan variasi jumlah kelas yang berbeda dan jenis kelas daun yang dipilih secara random untuk mengetahui performa sistem dalam mengidentifikasi jenis tanaman obat berdasarkan jumlah kelas daun. Tabel 4.3 menunjukkan jumlah pembagian jenis kelas.

Tabel 4. 3 Jumlah Pembagian Variasi Kelas Daun

Jumlah Kelas	Nama Kelas Daun
6	Jambu Biji, Bayam Hijau, Jeruk Nipis, Mangga, Ara, Karanda
11	Cendana, Jambu Biji, Bayam Hijau, Jeruk Nipis, Mangga, Delima, Ara, Kelor, Sirih, Kersen, Lengkuas
16	Semua Kelas Daun

Uji coba pada penelitian ini juga melakukan uji normalitas untuk mengetahui kondisi distribusi normal pada masing-masing fitur daun. Fitur daun yang menunjukkan nilai signifikansi yang lebih rendah berdasarkan uji normalitas kemudian dihilangkan dan dilakukan uji coba identifikasi jenis tanaman obat tanpa menggunakan fitur yang dianggap tidak berdistribusi normal.

4.1.2. Menghitung Kinerja Sistem

Dalam menghitung kinerja sistem penelitian ini menggunakan konsep confussion matrix multi class dengan jumlah kelasnya sebanyak 16. Pada tahap ini percobaan dilakukan menggunakan skenario pembagian data manual dengan rasio 70:30, 80:20, dan 90:10 dan skenario k-fold cross validation. Pada saat sistem setiap kali digunakan maka nilai akurasi, presisi, dan recall akan dihitung untuk mengetahui performa terbaik dari sistem yang digunakan berdasarkan skenario diatas. Konsep confusion matrix multi class akan menunjukkan hasil pencocokan antara kelas prediksi dan kelas yang sebenarnya. Nilai-nilai pada confusion matrix memiliki arti yang berbeda, yaitu sebagai berikut :

1. TP (True Positive):Jumlah data dengan nilai sebenarnya positif dan diprediksi benar oleh sistem.
2. TN (True Negative): Jumlah data dengan nilai sebenarnya positif namun diprediksi salah oleh sistem.

3. FP (False Positive) : Jumlah data dengan nilai sebenarnya negative dan diprediksi benar oleh sistem.
4. FN (False Negative) : Jumlah data dengan nilai sebenarnya negative dan diprediksi salah oleh sistem.

Rumus menghitung nilai akurasi, presisi, dan recall dapat menggunakan persamaan berikut :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}\ x\ 100\% \quad (4.1)$$

$$Precision_C = \frac{TP_c}{TP_c + FP_c} x\ 100\% \quad (4.2)$$

$$Recall_C = \frac{TP_c}{TP_c + FN_c} x\ 100\% \quad (4.3)$$

$$F - measure_C = \frac{2\ x\ Precision_c\ x\ Recall_C}{Precision_c + Recall_c} x\ 100\% \quad (4.4)$$

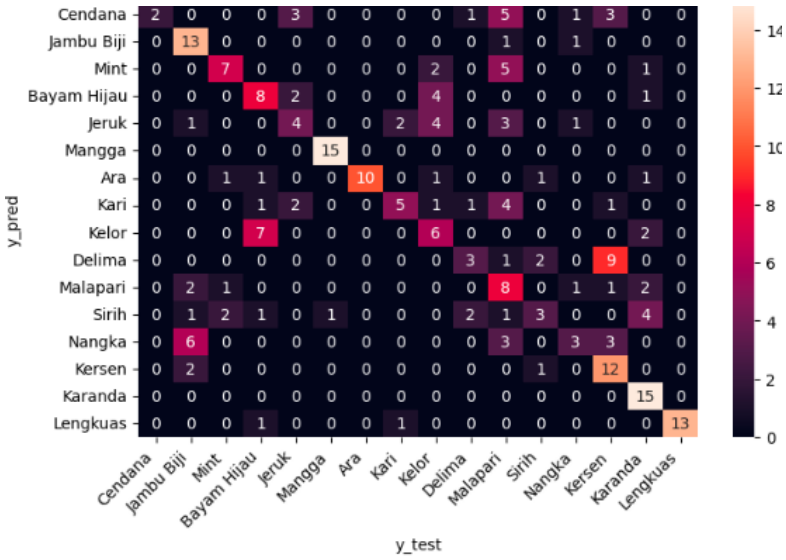
Penelitian ini juga dilakukan pengujian validasi k-fold cross validation menggunakan *10-fold cross validation*. Dataset dibagi menjadi data *training* dan data *testing* dengan jumlah total data awalnya 800, data akan dibagi menjadi *10 fold* yang sama besar kemudian dihitung rata-rata performa model dalam setiap fold sehingga dapat menemukan nilai yang paling sesuai dan kemudian dilakukan evaluasi dengan menghitung akurasi, presisi, dan recall, dan f-measure menggunakan *confussion matrix.* Tabel 4.4 merupakan skenario 10-fold cross validation.

Tabel 4. 4 Skenario 10-fold cross validation

Fold	***10-Fold Cross Validation***									
	1-16	**1-16**	**1-16**	**1-16**	**1-16**	**1-16**	**1-16**	**1-16**	**1-16**	**1-16**
1	80	80	80	80	80	80	80	80	80	80
2	80	80	80	80	80	80	80	80	80	80
3	80	80	80	80	80	80	80	80	80	80
4	80	80	80	80	80	80	80	80	80	80
5	80	80	80	80	80	80	80	80	80	80
6	80	80	80	80	80	80	80	80	80	80
7	80	80	80	80	80	80	80	80	80	80
8	80	80	80	80	80	80	80	80	80	80
9	80	80	80	80	80	80	80	80	80	80
10	80	80	80	80	80	80	80	80	80	80

4.2. Hasil Uji Coba 1

Hasil uji coba 1 skenarionya terletak pada jumlah pembagian rasio data dan variasi jumlah kelas daun. Pada sistem ini dataset dibagi menjadi data training dan data testing secara manual dengan perbandingan 70:30, 80:20, dan 90:10 dengan jumlah data training yang lebih besar dari data testing. Selanjutnya, proses uji coba ini menggunakan 3 variasi jumlah kelas daun, hal ini dilakukan untuk mengetahui seberapa besar pengaruh jumlah kelas daun terhadap performa sistem. Berdasarkan skenario tersebut hasil data testing dari klasifikasi gaussian naïve bayes dengan pembagian data perbandingan 70:30 ditampilkan oleh Gambar 4.1 sebagai berikut.



Gambar 4. 1 Visualisasi Prediksi Data 70:30 pada 16 Kelas

Berdasarkan Gambar 4.1 maka diperoleh hasil perhitungan *confusion matrix multi class* untuk *precision*, *recall*, dan *f-measure* pada 6 kelas jenis daun tanaman obat ditunjukkan pada Tabel 4.5.

Tabel 4. 5 Nilai *Precision, Recall, F-measure* pada 6 kelas

Kelas Daun	***Precision***	***Recall***	***F-Measure***
Jambu Biji	78,94%	100%	88,23%
Bayam Hijau	75%	80%	77,41%
Jeruk	69,23%	60%	64,28%
Mangga	100%	100%	100%
Ara	100%	80%	88,89%
Karanda	93,33%	93,33%	93,33%

Hasil perhitungan *confusion matrix multi class* untuk *precision*, *recall*, dan *f-measure* pada 11 kelas jenis daun tanaman obat ditunjukkan pada Tabel 4.6.

Tabel 4. 6 Nilai *Precision, Recall, F-measure* pada 11 kelas

Kelas Daun	***Precision***	***Recall***	***F-Measure***
Cendana	75%	40%	52,17%
Jambu Biji	75%	100%	85,71%
Bayam Hijau	57,14%	53,33%	55,17%
Jeruk	50%	33,33%	40%
Mangga	100%	93,33%	96,55%
Delima	57,69%	100%	73,17%
Ara	91,67%	73,33%	81,48%
Kelor	54,54%	80%	64,86%

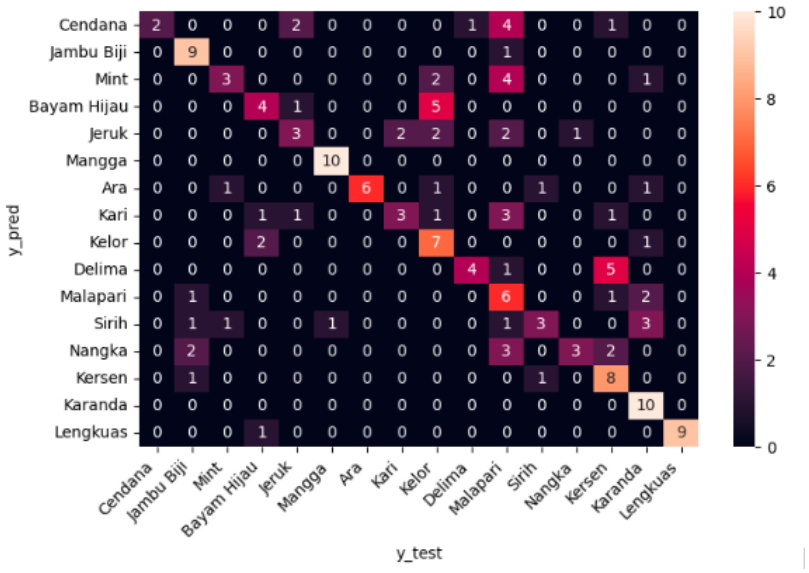
Kelas Daun	*Precision*	*Recall*	*F-Measure*
Sirih	40%	13,33%	20%
Karanda	73,68%	93,33%	82,35%
Lengkuas	100%	100%	100%

Hasil perhitungan *confusion matrix multi class* untuk *precision*, *recall*, dan *f-measure* pada 16 kelas jenis daun tanaman obat ditunjukkan pada Tabel 4.7.

Tabel 4. 7 Nilai *Precision, Recall, F-measure* pada 16 kelas

Kelas Daun	Confusion Matrix		
	Precision	**Recall**	**F-measure**
Cendana	100%	13,33%	23,53 %
Jambu Biji	52%	86,67%	65%
Mint	63,64%	46,67%	53,85%
Bayam Hijau	42,11%	53,33%	47,06%
Jeruk Nipis	36,36%	26,67%	30,77%
Mangga	93,75%	100%	96,77%
Ara	100%	66,67%	80%
Kari	62,50%	33,33%	43,48%
Kelor	33,33%	40%	36,36%
Delima	42,86%	20%	27.27%
Malapari	25,81%	53,33%	34,78%
Sirih	42,86%	20%	27,27%
Nangka	42,86%	20%	27,27%
Kersen	41,38%	80%	54,55%
Karanda	57,69%	100%	73,17%
Lengkuas	100%	86,67%	92,86%

Selanjutnya, Gambar 4.2 menunjukkan hasil identifikasi jenis tanaman obat pada masing-masing kelas dengan pembagian 80% untuk data training dan 20% untuk data testing.



Gambar 4. 2 Visualisasi Prediksi Data 80:20 pada 16 Kelas

Berdasarkan Gambar 4.2 maka diperoleh hasil confusion matrix untuk *precision*, *recall*, dan *f-measure* untuk 6 kelas tanaman obat yang ditunjukkan pada Tabel 4.8.

Tabel 4. 8 Nilai *Precision, Recall, F-measure* pada 6 kelas

Kelas Daun	***Precision***	***Recall***	***F-Measure***
Jambu Biji	76,92%	100%	86,95%
Bayam Hijau	66,67%	80%	72,72%
Jeruk	66,67%	60%	63,15%
Mangga	100%	100%	100%
Ara	100%	80%	88,89%
Karanda	100%	80%	88,89%

Hasil perhitungan *confusion matrix multi class* untuk *precision*, *recall*, dan *f-measure* pada 11 kelas jenis daun tanaman obat ditunjukkan pada Tabel 4.9.

Tabel 4. 9 Nilai *Precision, Recall, F-measure* pada 11 kelas

Kelas Daun	***Precision***	***Recall***	***F-Measure***
Cendana	71,42%	50%	58,82%
Jambu Biji	76,92%	100%	86,95%
Bayam Hijau	50%	40%	44,44%
Jeruk	33,33%	20%	25%
Mangga	90%	90%	90%
Delima	62,50%	100%	76,92%
Ara	87,50%	70%	77,78%
Kelor	43,75%	70%	53,84%

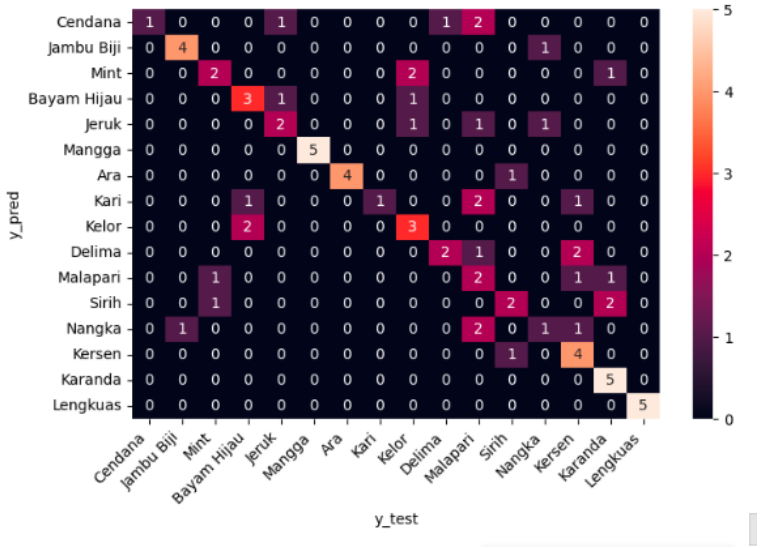
Kelas Daun	*Precision*	*Recall*	*F-Measure*
Sirih	25%	10%	14,28%
Karanda	69,23%	90%	78,26%
Lengkuas	100%	90%	94,73%

Hasil perhitungan *confusion matrix multi class* untuk *precision*, *recall*, dan *f-measure* pada 16 kelas jenis daun tanaman obat ditunjukkan pada Tabel 4.10.

Tabel 4. 10 *Nilai Precision, Recall, F-measure* pada 16 Kelas

Class	Confusion Matrix		
	Precision	**Recall**	**F-measure**
Cendana	**100%**	**20%**	**33,33 %**
Jambu Biji	64,29%	90%	75%
Mint	60%	30%	40%
Bayam Hijau	50%	40%	44,44%
Jeruk Nipis	42,86%	30%	50%
Mangga	90,91%	100%	95,24%
Ara	100%	60%	75 %
Kari	60%	30%	40%
Kelor	38,89%	70%	50%
Delima	80%	40%	53,33%
Malapari	24%	60%	34,29%
Sirih	60%	30%	40%
Nangka	75%	30%	28,57%
Kersen	44,44%	80%	54,55%
Karanda	55,56%	100%	71,43%
Lengkuas	100%	90%	94,74%

Selanjutnya, Gambar 4.3 menunjukkan visualisasi prediksi pada masing-masing kelas dengan pembagian 90% untuk data training dan 10% untuk data testing.



Gambar 4. 3 Visualisasi Prediksi Data 90:10 pada 16 Kelas

Berdasarkan Gambar 4.3 maka diperoleh hasil perhitungan confusion matrix untuk *precision*, *recall*, dan *f-measure* untuk 6 kelas tanaman obat yang ditunjukkan pada Tabel 4.11.

Tabel 4. 11 Nilai *Precision, Recall, F-measure* pada 6 Kelas

Class	*Precision*	*Recall*	*F-Measure*
Jambu Biji	71,42%	100%	83,33%
Bayam Hijau	83,33%	100%	90,91%
Jeruk	100%	60%	75%
Mangga	100%	100%	100%
Ara	100%	80%	88,89%
Karanda	100%	100%	100%

Hasil perhitungan *confusion matrix multi class* untuk *precision*, *recall*, dan *f-measure* pada 11 kelas jenis daun tanaman obat ditunjukkan pada Tabel 4.12.

Tabel 4. 12 Nilai *Precision, Recall, F-measure* pada 11 Kelas

Kelas Daun	*Precision*	*Recall*	*F-Measure*
Cendana	50%	20%	28,57%
Jambu Biji	83,33%	100%	90,91%
Bayam Hijau	50%	20%	28,57%
Jeruk	20%	20%	20%
Mangga	100%	100%	100%
Delima	55,56%	100%	71,42%
Ara	80%	80%	80%
Kelor	44,44%	80%	57,14%

Kelas Daun	*Precision*	*Recall*	*F-Measure*
Sirih	50%	20%	28,57%
Karanda	80%	80%	80%
Lengkuas	100%	100%	100%

Hasil perhitungan *confusion matrix multi class* untuk *precision*, *recall*, dan *f-measure* pada 16 kelas jenis daun tanaman obat ditunjukkan pada Tabel 4.13.

Tabel 4. 13 Nilai *Precision, Recall, F-measure* pada 16 Kelas

	Confusion Matrix		
Kelas Daun	***Precision***	***Recall***	***F-measure***
Cendana	100%	20%	33,33 %
Jambu Biji	80%	80%	80%
Mint	50%	40%	44,44%
Bayam Hijau	50%	60%	54,55%
Jeruk Nipis	50%	40%	44,44%
Mangga	100%	100%	100%
Ara	100%	80%	88,89 %
Kari	100%	20%	33,33%
Kelor	42,86%	60%	50%
Delima	66,67%	40%	50%
Malapari	20%	40%	26,67%
Sirih	50%	40%	44,44%
Nangka	33,33%	20%	25%
Kersen	44,44%	80%	57,14%
Karanda	55,56%	100%	71,43%
Lengkuas	100%	100%	100%

Hasil uji coba menggunakan 3 variasi rasio pembagian data training dan testing untuk 3 variasi jumlah kelas daun ditunjukkan pada Tabel 4.14.

Tabel 4. 14 Hasil Rata-rata *Accuracy, Precision, Recall*, dan *F-measure* Uji Coba 1

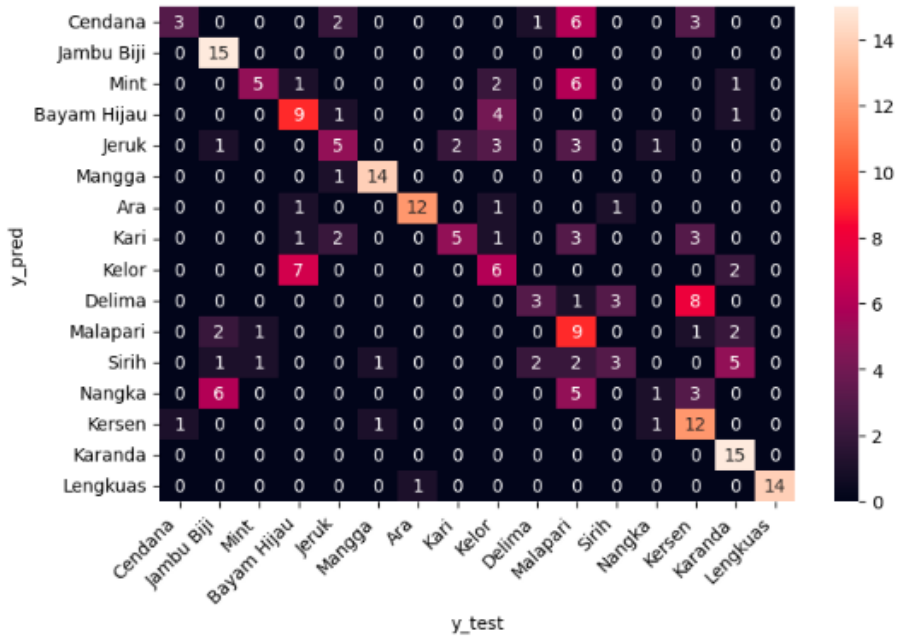
Jumlah Kelas	**Rasio Data**	***Accuracy***	***Precision***	***Recall***	***F-Measure***
6	70:30	85,56%	86,09%	85,56%	85,36%
	80:20	83,33%	85,04%	83,33%	83,44%
	90:10	90%	92,46%	90%	89,69%
11	70:30	70,91%	70,43%	70,91%	68,32%
	80:20	66,36%	64,52%	66,36%	63,73%
	90:10	65,45%	64,85%	65,45%	62,29%

Jumlah Kelas	Rasio Data	*Accuracy*	*Precision*	*Recall*	*F-Measure*
16	70:30	52,92%	58,57%	52,92%	50,87%
	80:20	56,25%	65,37%	56,25%	55,13%
	90:10	57,50%	65,18%	57,50%	56,48%

Berdasarkan hasil uji coba 1 dengan menggunakan 3 variasi rasio data dan variasi 3 pembagian kelas didapatkan hasil terbaik pada rasio pembagian data 90:10 untuk pembagian 6 kelas. Hal ini berarti bahwa sistem mampu melakukan identifikasi jenis tanaman obat secara optimal pada rasio 90% data training dan 10% data testing untuk 6 jenis tanaman obat dengan nilai *accuracy* sebesar 90%. Selanjutnya, pada saat uji coba menggunakan 11 dan 16 kelas daun, nilai *accuracy* mengalami penurunan dengan rasio data terbaik masing-masing terletak pada rasio 90:10. Hal ini menunjukkan bahwa penambahkan jumlah kelas juga mempengaruhi hasil *accuracy* identifikasi tanaman obat.

4.3. Hasil Uji Coba 2

Berdasarkan hasil uji normalitas fitur yang telah dilakukan hasilnya bahwa fitur panjang memiliki nilai signifikansi yang lebih rendah dibandingkan fitur lainnya, sehingga diasumsikan bahwa fitur panjang tidak berdistribusi normal. Selanjutnya, penelitian ini melakukan uji coba identifikasi jenis daun berdasarkan bentuk pada citra daun tanpa menggunakan fitur panjang dengan 3 variasi rasio dan 3 variasi pembagian jumlah kelas daun. Gambar 4.5 menunjukkan visualisasi hasil prediksi jenis tanaman obat tanpa menggunakan fitur panjang daun.



Gambar 4. 4 Visualisasi Uji Coba 2 rasio 70:30 pada 16 Kelas

Berdasarkan Gambar 4.4 maka diperoleh hasil perhitungan confusion matrix untuk *precision*, *recall*, dan *f-measure* untuk 6 kelas tanaman obat yang ditunjukkan pada Tabel 4.15.

Tabel 4. 15 Nilai *Precision, Recall, F-measure* Uji Coba 2 pada 6 Kelas

Kelas Daun	Precision	Recall	F-Measure
Jambu Biji	78,94%	100%	88,23%
Bayam Hijau	68,42%	86,67%	76,47%
Jeruk	80%	53,33%	64%
Mangga	93,75%	100%	96,77%
Ara	100%	86,67%	92,85%
Karanda	100%	86,67%	92,85%

Hasil perhitungan *confusion matrix multi class* untuk *precision*, *recall*, dan *f-measure* pada 11 kelas jenis daun tanaman obat ditunjukkan pada Tabel 4.16.

Tabel 4. 16 Nilai *Precision, Recall, F-measure* Uji Coba 2 pada 11 Kelas

Kelas Daun	Precision	Recall	F-Measure
Cendana	75%	40%	52,17%
Jambu Biji	75%	100%	85,71%
Bayam Hijau	56,25%	60%	58,06%
Jeruk	46,16%	40%	42,85%
Mangga	93,75%	100%	96,77%

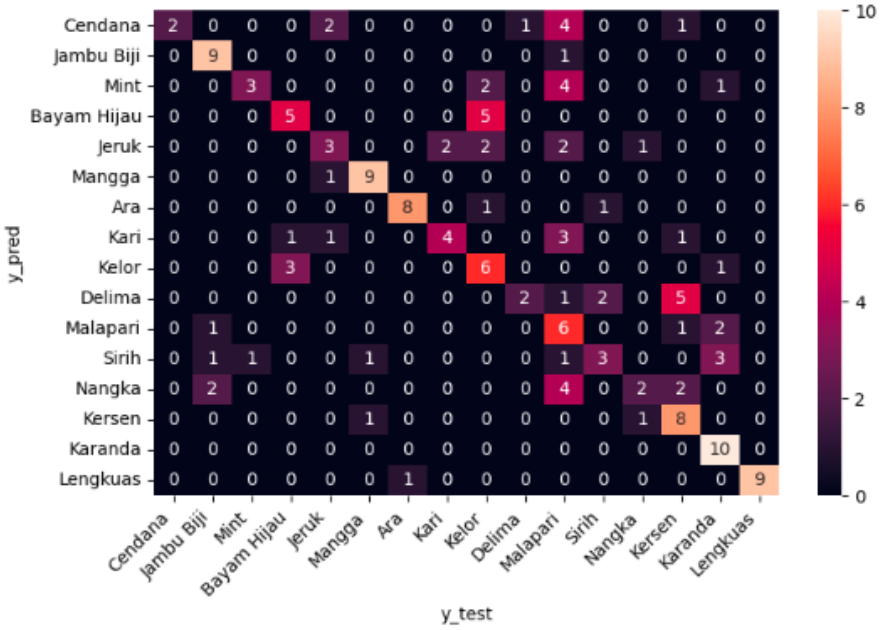
Kelas Daun	Precision	Recall	F-Measure
Delima	58,33%	93%	71,79%
Ara	100%	80%	88,89%
Kelor	57,89%	73%	64,70%
Sirih	75%	20%	31,57%
Karanda	72 %	86,67%	78,78%
Lengkuas	100%	100%	100%

Hasil perhitungan *confusion matrix multi class* untuk *precision*, *recall*, dan *f-measure* pada 16 kelas jenis daun tanaman obat ditunjukkan pada Tabel 4.17.

Tabel 4. 17 Nilai *Precision, Recall, F-measure* Uji Coba 2 pada 16 Kelas

Kelas Daun	Confusion Matrix		
	Precision	Recall	F-measure
Cendana	75%	20%	31,57 %
Jambu Biji	60%	100%	75%
Mint	71,42%	33,33%	45,45%
Bayam Hijau	47,36%	60%	52,94%
Jeruk Nipis	45,45%	33,33%	38,46%
Mangga	87,50%	93,33%	90,32%
Ara	92,30%	80%	85,71%
Kari	71,42%	33,33%	45,45%
Kelor	35,29%	40%	37,50%
Delima	50%	20%	28,57%
Malapari	25,71%	60%	36%
Sirih	42,85%	20%	27,27%
Nangka	33,33%	6,67%	11%
Kersen	40%	80%	53,33%
Karanda	57,69%	100%	73,17%
Lengkuas	100%	93,33%	96,55%

Selanjutnya, Gambar 4.5 menunjukkan hasil identifikasi jenis tanaman obat pada masing-masing kelas dengan pembagian 80% untuk data training dan 20% untuk data testing.



Gambar 4. 5 Visualisasi Uji Coba 2 rasio 80:20 pada 16 Kelas

Berdasarkan Gambar 4.5 maka diperoleh hasil confusion matrix untuk *precision*, *recall*, dan *f-measure* untuk 6 kelas tanaman obat yang ditunjukkan pada Tabel 4.18.

Tabel 4. 18 Nilai *Precision, Recall, F-measure* Uji Coba 2 pada 6 Kelas

Kelas Daun	**Precision**	**Recall**	**F-Measure**
Jambu Biji	76,92%	100%	86,95%
Bayam Hijau	61,53%	80%	69,56%
Jeruk	71,42%	50%	58,82%
Mangga	90,91%	100%	95,23%
Ara	100%	80%	88,89%
Karanda	100%	80%	88,89%

Hasil perhitungan *confusion matrix multi class* untuk *precision*, *recall*, dan *f-measure* pada 11 kelas jenis daun tanaman obat ditunjukkan pada Tabel 4.19.

Tabel 4. 19 Nilai *Precision, Recall, F-measure* Uji Coba 2 pada 11 Kelas

Kelas Daun	***Precision***	***Recall***	***F-Measure***
Cendana	66,67%	40%	50%
Jambu Biji	76,92%	100%	86,95%
Bayam Hijau	50%	40%	44,44%
Jeruk	37,50%	30%	33,33%
Mangga	90,91%	100%	95,23%
Delima	62,50%	100%	76,92%

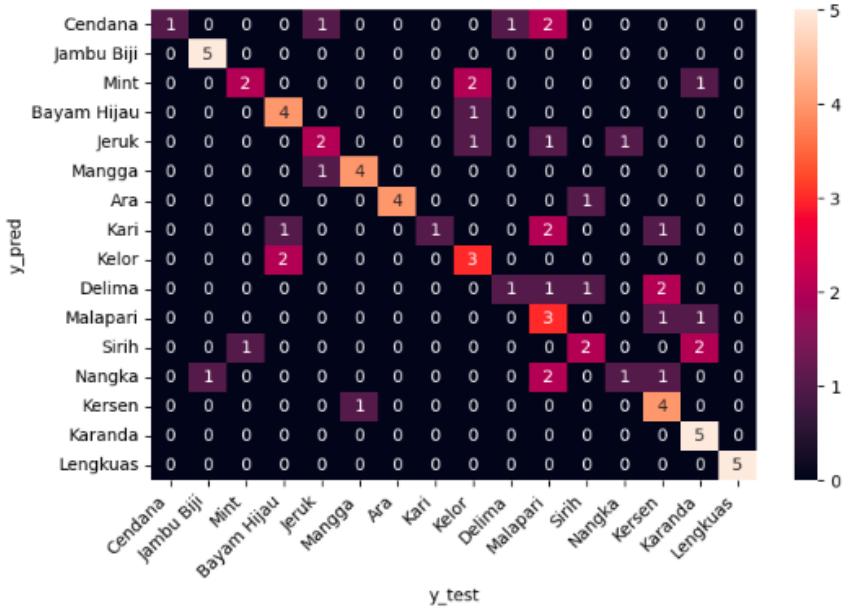
Kelas Daun	*Precision*	*Recall*	*F-Measure*
Ara	100%	80%	88,89%
Kelor	46,67%	70%	56%
Sirih	66,67%	20%	30,76%
Karanda	66,67%	80%	72,73%
Lengkuas	100%	100%	100%

Hasil perhitungan *confusion matrix multi class* untuk *precision*, *recall*, dan *f-measure* pada 16 kelas jenis daun tanaman obat ditunjukkan pada Tabel 4.20.

Tabel 4. 20 Nilai Precision, Recall, F-measure Uji Coba 2 pada 16 Kelas

	Confusion Matrix		
Class	**Precision**	**Recall**	**F-measure**
Cendana	100%	20%	33,33 %
Jambu Biji	69,23%	90%	78,26%
Mint	75%	30%	42,85%
Bayam Hijau	55,56%	50%	52,63%
Jeruk Nipis	42,85%	30%	35,29%
Mangga	81,82%	90%	85,71%
Ara	88,89%	80%	84,21 %
Kari	66,67%	40%	50%
Kelor	37,50%	60%	46,15%
Delima	66,67%	20%	30,76%
Malapari	23,07%	60%	33,33%
Sirih	50%	30%	37,50%
Nangka	50%	20%	28,57%
Kersen	44,44%	80%	57,14%
Karanda	58,82%	100%	74,07%
Lengkuas	100%	90%	94,73%

Selanjutnya, Gambar 4.6 menunjukkan visualisasi prediksi pada masing-masing kelas dengan pembagian 90% untuk data training dan 10% untuk data testing.



Gambar 4. 6 Visualisasi Uji Coba 2 rasio 90:10 pada 16 Kelas

Berdasarkan Gambar 4.6 maka diperoleh hasil perhitungan confusion matrix untuk *precision*, *recall*, dan *f-measure* untuk 6 kelas tanaman obat yang ditunjukkan pada Tabel 4.16.

Tabel 4. 21 Nilai *Precision, Recall, F-measure* Uji Coba 2 pada 6 Kelas

Kelas Daun	***Precision***	***Recall***	***F-Measure***
Jambu Biji	71,42%	100%	83,33%
Bayam Hijau	80%	80%	80%
Jeruk	75%	60%	66,67%
Mangga	100%	100%	100%
Ara	100%	80%	88,89%
Karanda	100%	100%	100%

Hasil perhitungan *confusion matrix multi class* untuk *precision*, *recall*, dan *f-measure* pada 11 kelas jenis daun tanaman obat ditunjukkan pada Tabel 4.22.

Tabel 4.22 Nilai *Precision, Recall, F-measure* Uji Coba 2 pada 11 Kelas

Kelas Daun	***Precision***	***Recall***	***F-Measure***
Cendana	100%	40%	57,14%
Jambu Biji	71,42%	100%	83,33%
Bayam Hijau	80%	80%	80%
Jeruk	33,33%	20%	25%
Mangga	100%	100%	100%
Delima	55,56%	100%	71,42%

Kelas Daun	*Precision*	*Recall*	*F-Measure*
Ara	100%	80%	88,89%
Kelor	66,67%	80%	72,73%
Sirih	66,67%	40%	50%
Karanda	66,67%	80%	72,73%
Lengkuas	100%	100%	100%

Hasil perhitungan *confusion matrix multi class* untuk *precision*, *recall*, dan *f-measure* pada 16 kelas jenis daun tanaman obat ditunjukkan pada Tabel 4.23.

Tabel 4.23 Nilai *Precision, Recall, F-measure* Uji Coba 2 pada 16 Kelas

Kelas Daun	*Confusion Matrix*		
	Precision	*Recall*	*F-measure*
Cendana	100%	20%	33,33 %
Jambu Biji	83,33%	100%	90,91%
Mint	66,67%	40%	50%
Bayam Hijau	57,14%	80%	66,67%
Jeruk Nipis	50%	40%	44,44%
Mangga	80%	80%	80%
Ara	100%	80%	88,89 %
Kari	100%	20%	33,33%
Kelor	42,85%	60%	50%
Delima	50%	20%	28,57%
Malapari	27,27%	60%	37,50%
Sirih	50%	40%	44,44%
Nangka	50%	20%	28,57%
Kersen	44,44%	80%	57,14%
Karanda	55,56%	100%	71,42%
Lengkuas	100%	100%	100%

Hasil uji coba menggunakan 3 variasi rasio pembagian data training dan testing untuk 3 variasi jumlah kelas daun ditunjukkan pada Tabel 4.24.

Tabel 4.24 Hasil Rata-rata *Accuracy, Precision, Recall,* dan *F-measure* 2

Jumlah Kelas	Rasio Data	*Accuracy*	*Precision*	*Recall*	*F-Measure*
6	70:30	85,56%	86,85%	85,56%	85,20%
	80:20	81,67%	83,47%	81,67%	81,39%
	90:10	86,67%	87,74%	86,67%	86,48%
11	70:30	72,12%	73,60%	72,12	70,12%
	80:20	69,09%	69,50%	69,09%	66,84%
	90:10	74,55%	76,39%	74,55%	72,84%
16	70:30	54,58%	58,46%	54,58%	51,78%
	80:20	55,62%	63,16%	55,62%	54,04%
	90:10	58,75%	66,08%	58,75%	56,58%

Berdasarkan hasil uji coba 1 dengan menggunakan 3 variasi rasio data dan variasi 3 pembagian kelas didapatkan hasil terbaik pada rasio pembagian data 90:10 untuk pembagian 6 kelas. Hal ini berarti bahwa sistem mampu melakukan identifikasi jenis tanaman obat secara optimal pada rasio 90% data training dan 10% data testing untuk 6 jenis tanaman obat dengan nilai accuracy sebesar 86,67%. Selanjutnya, pada saat uji coba menggunakan 11 dan 16 kelas daun, nilai *accuracy* mengalami penurunan dengan rasio data terbaik masing-masing terletak pada rasio 90:10. Hal ini menunjukkan bahwa penambahkan jumlah kelas juga mempengaruhi hasil *accuracy* identifikasi tanaman obat.

4.4. Hasil Uji Coba K-Fold Cross Validation

Tahap selanjutnya yaitu dilakukan pengujian validasi silang menggunakan *k-fold cross validation* dengan nilai k sebesar 10 sehingga nantinya akan dilakukan proses iterasi 10 kali untuk membagi data *training* dan *testing*. Hasil pengujian 10-fold cross validation ditunjukkan pada Tabel 4.25.

Tabel 4.25 Hasil *10-Fold Cross Validation*

***Fold* ke-**	***10-fold cross validation* (semua fitur)**				***10-fold cross validation* (tanpa fitur panjang)**			
	Acc	***Prec***	***Rec***	***F1***	***Acc***	***Prec***	***Rec***	***F1***
1	56%	60%	56%	58%	56%	68%	56%	61%
2	59%	55%	59%	57%	60%	56%	60%	58%
3	61%	64%	61%	63%	54%	60%	54%	57%
4	54%	52%	54%	53%	54%	46%	54%	49%
5	53%	52%	52%	52%	49%	46%	49%	48%
6	56%	58%	56%	57%	56%	58%	56%	57%
7	53%	45%	52%	48%	57%	55%	57%	56%
8	56%	68%	56%	62%	55%	62%	55%	58%
9	45%	52%	45%	48%	44%	51%	44%	47%
10	50%	50%	50%	50%	50%	47%	50%	49%
Rata-rata	54,3%	56%	54,1%	55%	53%	55%	53%	54%

Berdasarkan pengujian menggunakan *10-fold cross validation* diatas, hasilnya menunjukkan bahwa nilai akurasi rata-rata sebesar 54%. Hasil akurasi tertinggi saat menggunakan semua fitur terjadi pada *fold* ke-3 dengan hasil akurasi sebesar 61%, hasil *precision* sebesar 64%, hasil *recall* sebesar 61% dan *f-measure* sebesar 63%. Uji coba kedua yaitu uji coba tanpa menggunakan fitur panjang dengan rasio pembagian data 90:10 untuk data training dan data testing dilakukan pengujian validasi silang dengan 10-fold cross validation. Berdasarkan pengujian *10-fold cross validation* tanpa menggunakan fitur panjang diperoleh hasil terbaik terdapat pada *fold* ke 2. Nilai *accuracy* sebesar 60%, *precision* sebesar 56%, *recall* 60%, dan *f-measure* 58% dengan rata-rata hasil akurasi pada semua *fold* yaitu *accuracy* sebesar 53%, *precison* 55%, *recall* 53%, dan *f-measure* 54%. Hasil ini menunjukkan bahwa validasi silang *10-fold cross validation* menysehingga *k-fold cross validation* memiliki pengaruh pada kinerja sistem identifikasi tanaman obat.

4.5. Pembahasan

Berdasarkan hasil uji coba pembagian rasio data dan uji coba pembagian 3 variasi kelas tanaman obat, didapatkan hasil terbaik terletak pada rasio 90:10 dengan nilai *accuracy*, *precision*, *recall*, dan *f-measure* pada variasi 6 kelas masing-masing bernilai 90%, 92,46%, 90%, dan 89,69%, nilai *accuracy*, *precision*, *recall*, dan *f-measure* pada variasi 11 kelas masing-masing bernilai 65,45%, 64,85%, 65,45%, dan 62,29%, dan nilai *accuracy*, *precision*, *recall*, dan *f-measure* pada variasi 16 kelas masing-masing bernilai 57,50%, 65,18%, 57,50%, dan 56,48%. Selanjutnya, penulis mencoba melakukan uji normalitas pada nilai-nilai fitur tersebut menggunakan *Kolmogorov-Smirnov* dan *Shapiro-Wilk* dan pada Lampiran IV hasil uji normalitas fitur bentuk daun yang menunjukkan bahwa fitur panjang memiliki nilai signifikansi yang lebih rendah dibandingkan fitur yang lainnya, hal ini berarti bahwa fitur panjang dapat diasumsikan tidak berdistribusi normal. Selanjutnya hasil uji coba kedua tanpa menggunakan fitur panjang didapatkan hasil terbaik terletak pada rasio 90:10 dengan nilai *accuracy*, *precision*, *recall*, dan *f-measure* pada variasi 6 kelas masing-masing bernilai 86,67%, 87,74%, 86,67%, dan 86,48%, nilai *accuracy*, *precision*, *recall*, dan *f-measure* pada variasi 11 kelas masing-masing bernilai 74,55%, 76,39%, 74,55%, dan 72,84%, dan nilai *accuracy*, *precision*, *recall*, dan *f-measure* pada variasi 16 kelas masing-masing bernilai 58,75%, 66,08%, 58,75%, dan 56,58%. Hasil tersebut disebabkan jumlah data *training* yang lebih besar dibandingkan jumlah data *training* pada rasio yang lainnya sehingga memungkinkan model untuk mempelajari lebih banyak data dan mendapatkan informasi yang lebih bervariasi. Hasilnya bahwa dengan rasio 90:10

dapat memprediksi sistem dengan lebih baik dan menghasilkan nilai akurasi yang tinggi.

Hasil uji coba 1 dan uji coba 2 memiliki hasil yang berbeda, pada uji coba 1 yang menggunakan semua fitur, hasil pengujian terbaik terletak pada rasio 90:10 untuk variasi 6 kelas dengan *accuracy* 90%. Kemudian, hasil terbaik pada uji coba 2 juga terletak pada rasio 90:10 dengan menggunakan 6 kelas, namun nilai *accuracy* nya lebih rendah dibandingkan pengujian menggunakan semua fitur. Namun, saat pengujian menggunakan 11 kelas, hasil *accuracy* uji coba 1 yang menggunakan semua fitur lebih rendah dibandingkan dengan hasil *accuracy* tanpa menggunakan fitur panjang hal yang sama berlaku pada pengujian 16 kelas, hasil *accuracy* mengalami peningkatan jika dibandingkan pengujian uji coba 1 yang menggunakan semua fitur. Hal ini disimpulkan bahwa pemilihan jenis fitur juga mempengaruhi kinerja dalam melakukan identifikasi jenis tanaman obat.

Pada Tabel 4.5 dapat dilihat bahwa evaluasi performa model untuk pembagian rasio data 70:30 pada 6 kelas mangga, ara, dan karanda memiliki performa yang sangat baik sehingga nilai f-measure nya paling tinggi. Selanjutnya, Tabel 4.6 menunjukkan bahwa kelas jambu biji dan lengkuas juga memiliki performa cukup baik dalam mengenali tanaman, namun performa jeruk dan sirih sangat rendah sehingga nilai recall nya sangat rendah dan kinerjanya sangat kurang dalam mengklasifikasikan tanaman. Kemudian Tabel 4.7 menunjukkan bahwa kinerja model dalam mengklasifikasikan kelas mangga, lengkuas, dan daun ara sangat baik dilihat dari nilai *f-measure* nya yang paling tinggi. Sedangkan untuk kelas cendana, delima, sirih dan nangka performa kinerjanya sangat kurang dalam

memprediksi data. Namun, pada kelas cendana, model ternyata sangat baik dalam mengklasifikasikan tanaman yang bukan cendana sebagai tanaman cendana sehingga mempunyai nilai *precision* yang sangat tinggi namun memiliki nilai *f-measure* yang sangat rendah.

Pada Tabel 4.8 ditunjukkan hasil evaluasi performa model untuk pembagian rasio data 80:20 pada 6 kelas, hasilnya semua kelas memiliki kinerja yang sangat baik kecuali kelas jeruk yang performanya paling rendah diantara kelas lainnya, sehingga nilai f-measurenya juga rendah. Pada Tabel 4.9 hasil pengujian pada 11 kelas menunjukkan bahwa kelas jeruk, kelor, dan sirih , dan bayam hijau yang memiliki peforma rendah dalam mengenali tanaman. Selanjutnya pada Tabel 4.10 diketahui bahwa kinerja model sangat baik dalam mengklasifikasikan kelas mangga, lengkuas, dan jambu biji sehingga nilai *f-measure*nya paling tinggi diantara kelas yang lain. Performa model pada kelas yang lain cenderung rata-rata kisaran 40% hingga 50% . Pada pembagian rasio data ini, ternyata kelas jeruk nipis dan malapari yang memiliki performa paling rendah dibandingkan dengan kelas yang lainnya sehingga nilai f-measurenya juga rendah. Kelas jeruk nipis juga memiliki nilai *precision* dan *recall* yang cukup rendah dan dilihat bahwa jeruk nipis lebih banyak melakukan kesalahan dalam memprediksi kelas sebagai negative. Hal ini artinya bahwa model kurang mengenali kelas yang benar sebagai positif.

Pada Tabel 4.9 untuk pembagian rasio data 90:10 dapat dilihat bahwa terdapat beberapa variasi kinerja model dalam melakukan klasifikasi pada setiap kelas. Misalnya pada kelas cendana, kari, dan angka cenderung mengklasifikasikan data yang bukan sebenarnya sebagai bagian dari kelas cendana, kari, dan nangka.

Kelas lainnya seperti kelas mint, bayam hijau, dan jeruk nipis cenderung melakukan klasifikasi pada data yang benar namun diprediksi salah oleh sistem sehingga hasil recallnya cukup rendah. Pada beberapa kelas lain seperti daun jambu biji, ara, daun mangga, daun delima, kersen, karanda dan lengkuas ternyata memiliki kinerja yang cukup baik dalam mengklasifikasikan sampel data sehingga nilai akurasi, presisi dan recall yang didapatkan juga cukup tinggi.

Berdasarkan pada Lampiran II Hasil Estimasi Parameter Rata-rata dapat dilihat bahwa nilai fitur panjang, lebar, dan luas pada setiap kelas daun memiliki selisih nilai yang berdekatan. Misalnya untuk fitur panjang, kelas mint, bayam hijau, ara, dan kari memiliki nilai panjang dengan selisih yang sangat dekat. Selanjutnya untuk fitur lebar kelas mint dan bayam, ara, malapari dan keranda juga memiliki selisih yang sangat dekat bahkan hampir sama. Kemudian untuk fitur luas kelas mint dan jeruk memiliki selisih nilai yang dekat. Namun, pada fitur keliling dan rasio masih memiliki nilai yang cukup representatif pada setiap daun.

Selanjutnya, berdasarkan hasil pengujian 3 variasi pembagian rasio, misalnya rasio 90:10 pada Tabel 4.9 dilihat bahwa kelas lengkuas mempunyai nilai *precision, recall, dan f-measure* paling tinggi diantara kelas yang lain yaitu sebesar 100%. Kemudian dilihat lagi pada Lampiran II Hasil Estimasi Parameter Rata-rata ternyata kelas lengkuas memang memiliki nilai fitur yang selisihnya cukup jauh dibandingkan dengan kelas lainnya sehingga sistem dapat dengan mudah mengenali kelas lengkuas daripada kelas lainnya. Selisih nilai-nilai yang berdekatan dan hampir mirip inilah yang menyebabkan sistem tidak dapat memprediksi kelas

secara optimal sehingga mempengaruhi kinerja sistem dan hasil akurasinya tidak cukup tinggi.

Selanjutnya dilakukan tahap *k-fold cross validation* untuk mengukur kinerja model dengan validasi silang. *K-fold cross validation* pada penelitian ini perlu dilakukan untuk memastikan bahwa performa model tetap stabil pada data yang berbeda. *K-fold cross validation* akan menggunakan seluruh data sebagai data uji dan data training secara bergantian, sehingga persebaran datanya lebih representatif dan memastikan bahwa model yang dibuat tidak hanya mengenali pola data tertentu saja,sehingga memaksimalkan kinerja pada model. Penelitian ini menggunakan *fold* 10 untuk menyamakan rasio pembagian data 90:10 pada uji coba sebelumnya. Hasil uji coba 10-fold cross validation ditunjukan pada Tabel 4.12 dengan menggunakan semua fitur, hasil akurasi tertinggi terdapat pada *fold* ke 3 dengan nilai *accuracy* 59%, *precision* 64%, *recall* 61%, dan *f-measure* 63%. Selanjutnya dilakukan uji coba 10-fold cross validation kedua tanpa menggunakan fitur panjang, hasilnya ditunjukkan pada Tabel 4.13. Berdasarkan pada Tabel 4.13 akurasi tertinggi terdapat pada fold ke 2 dengan nilai *accuracy* 60%, *precision* 56%, *recall* 60%, dan *f-measure* 58%. Hasil ini menunjukkan bahwa pembagian persebaran data pada *k-fold cross validation* memiliki pengaruh pada kinerja sistem identifikasi tanaman obat.

Tujuan dari penelitian ini yaitu mengidentifikasi jenis tanaman obat Indonesia sehingga dapat membantu masyarakat dalam mengenali jenis tanaman obat sehingga dapat mendorong potensi perkembangan obat herbal menggunakan

tanaman obat. Sistem yang dibangun masih perlu banyak perbaikan agar performa sistem semakin baik dan efektif sehingga menjadi bermanfaat bagi masyarakat.

Sebagaimana firman Allah subhanahu wa ta'ala dalam surat Ar-Ra'd ayat 3 berbunyi :

وَهُوَ ٱلَّذِى مَدَّ ٱلْأَرْضَ وَجَعَلَ فِيهَا رَوَٰسِىَ وَأَنْهَٰرًا ۖ وَمِن كُلِّ ٱلثَّمَرَٰتِ جَعَلَ فِيهَا زَوْجَيْنِ ٱثْنَيْنِ ۖ يُغْشِى ٱلَّيْلَ ٱلنَّهَارَ ۚ إِنَّ فِى ذَٰلِكَ لَءَايَٰتٍ لِّقَوْمٍ يَتَفَكَّرُونَ

"Dan Dialah Tuhan yang membentangkan bumi dan menjadikan gunung-gunung dan sungai-sungai diatasnya. Dan padanya Dia menjadikan padanya semua buah-buahan berpasang-pasangan; Dia menutupkan malam kepada siang. Sungguh, pada yang demikian itu terdapat tanda-tanda(kebesaran Allah) bagi orang-orang yang berpikir." (*QS.ar-Ra'd : 3*)

Menurut Tafsir Jalalain, ayat diatas menjelaskan bahwa Allah telah menciptakan alam semesta yang begitu luas dengan adanya gunung yang kokoh dan sungai-sungai. Allah swt juga menciptakan buah-buahan dengan berbagai macam bentuk, bau, dan warna yang beragam sebagai bukti dan tanda-tanda kebesaran Allah swt bagi orang-orang yang berpikir serta merenungkannya. Dalam mencapai tingkat keimanan yang sempurna para manusia senantiasa harus berfikir dan mengambil hikmah atas hasil pemikirannya. Seperti halnya juga pada pembuatan sistem identifikasi jenis tanaman obat Indonesia juga merupakan hasil pemikiran dan pemahaman terhadap ciptaan Allah swt dan memanfaatkannya (mengambil hikmah) dengan tujuan untuk memudahkan manusia dalam mengenali salah satu jenis ciptaan Allah swt yaitu tanaman obat berdasarkan bentuk daunnya berdasarkan ukuran-ukuran tertentu. Hal ini selaras dengan surat Al-Hijr ayat 19 yang berbunyi sebagai berikut:

وَٱلْأَرْضَ مَدَدْنَٰهَا وَأَلْقَيْنَا فِيهَا رَوَٰسِيَ وَأَنۢبَتْنَا فِيهَا مِن كُلِّ شَيْءٍ مَّوْزُونٍ

"Dan Kami telah menghamparkan bumi dan menjadikan padanya gunung-gunung dan Kami tumbuhkan padanya segala sesuatu menurut ukuran"(QS..al-Hijr:19)

Menurut tafsir Kementrian Agama Republik Indonesia, Allah telah menciptakan gunung-gunung yang kokoh serta segala macam tanaman yang beragam berdasarkan ukuran-ukuran tertentu yang tepat demi kemaslahatan sarana kehidupan semua makhluk yang ada di bumi. Ayat tersebut selaras dengan penelitian ini karena sesuai dengan firman Allah bahwa segala tanaman mempunyai ukuran-ukuran dan bentuk yang beragam sehingga dapat dikenali secara berbeda berdasarkan ukurannya.

Dari Abu Hurairah ra, Nabi Muhammad Saw, bersabda:

عَنْ أَبِيْ هُرَيْرَةَ رَضِيَ اللَّهُ عَنْهُ عَنِ النَّبِيِّ صَلَّى اللَّهُ عَلَيْهِ وَسَلَّمَ قَالَ مَنْ نَفَّسَ عَنْ مُؤْمِنٍ كُرْبَةً مِنْ كُرَبِ الدُّنْيَا ، نَفَّسَ اللهُ عَنْهُ كُرْبَةً مِنْ كُرَبِ يَوْمِ الْقِيَامَةِ، وَمَنْ يَسَّرَ عَلَى مُعْسِرٍ ، يَسَّرَ اللهُ عَلَيْهِ فِي الدُّنْيَا وَالْآخِرَةِ ، وَمَنْ سَتَرَ مُسْلِمًا ، سَتَرَهُ اللهُ فِي الدُّنْيَا وَالْآخِرَةِ ، وَاللهُ فِي عَوْنِ الْعَبْدِ مَا كَانَ الْعَبْدُ فِي عَوْنِ أَخِيهِ ،

"Barangsiapa yang melepaskan satu kesusahan seorang mukmin, pasti Allah akan melepaskan darinya satu kesusahan pada hari kiamat. Barangsiapa yang menjadikan mudah urusan orang lain, pasti Allah akan memudahkan urusannya di dunia dan akhirat. Barangsiapa yang menutupi aib seorang muslim, pasti Allah akan menutupi aibnya di dunia dan di akhirat. Allah senantiasa menolong hamba-Nya selama hamba-Nya suka menolong saudaranya."(Hadits Shahih: Diriwayatkan oleh Muslim no 2699).

Hasil implementasi dan pengujian pada penelitian ini diharapkan dapat bermanfaat banyak pihak seperti produsen obat herbal dalam memaksimalkan potensi penggunaan tanaman obat sebagai pengobatan herbal, bagi pelajar di bidang farmakologi, dan bagi akademisi yang membutuhkan penelitian terkait identifikasi tanaman obat.

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan hasil uji coba yang telah dilakukan, terdapat beberapa kesimpulan yang dapat diambil yaitu:

1. Pengujian sistem identifikasi jenis tanaman obat berdasarkan citra daun menggunakan metode deteksi tepi dan *Gaussian Naïve Bayes* dengan pemilihan fitur panjang, lebar, luas, keliling, dan rasio citra daun menggunakan 3 variasi skenario pembagian data dan 3 variasi pembagian jumlah kelas. Pengujian pertama hasil terbaik terletak pada rasio 90:10 dengan nilai *accuracy*, *precision*, *recall*, dan *f-measure* pada variasi 6 kelas masing-masing bernilai 90%, 92,46%, 90%, dan 89,69%, nilai *accuracy*, *precision*, *recall*, dan *f-measure* pada variasi 11 kelas masing-masing bernilai 65,45%, 64,85%, 65,45%, dan 62,29%, dan nilai *accuracy*, *precision*, *recall*, dan *f-measure* pada variasi 16 kelas masing-masing bernilai 57,50%, 65,18%, 57,50%, dan 56,48%. Selanjutnya hasil uji coba kedua tanpa menggunakan fitur panjang didapatkan hasil terbaik terletak pada rasio 90:10 dengan nilai accuracy, precision, recall, dan f-measure pada variasi 6 kelas masing-masing bernilai 86,67%, 87,74%, 86,67%, dan 86,48%, nilai accuracy, precision, recall, dan f-measure pada variasi 11 kelas masing-masing bernilai 74,55%, 76,39%, 74,55%, dan 72,84%, dan nilai accuracy, precision, recall, dan f-measure pada variasi 16 kelas masing-masing bernilai 58,75%, 66,08%, 58,75%, dan 56,58%..

Hal tersebut disimpulkan bahwa identifikasi jenis tanaman obat menggunakan deteksi tepi dan gaussian naïve bayes dapat bekerja secara optimal pada 6 kelas daun.

2. Berdasarkan pengujian pada *10-fold cross validation*, performa terbaik terdapat pada fold ke 3 pada 16 kelas dengan nilai *accuracy* sebesar 61%, *precision* 64%, *recall* 61%, dan *f-measure* 63%. Hal ini dapat disimpulkan bahwa pembagian data pada *k-fold cross validation* yang dapat mempengaruhi hasil performa kinerja sistem. Berdasarkan hasil pengujian dari 3 variasi skenario rasio pembagian data diketahui performa paling baik dalam mengidentifikasi jenis tanaman obat terdapat pada kategori lengkuas. Hal ini terjadi karena nilai rata-rata dan varian setiap fitur pada kategori lengkuas memiliki rentang nilai yang cukup jauh dibandingkan dengan nilai fitur-fitur antar kelas pada kategori lainnya, sehingga sistem lebih baik dalam mengidentifikasi kategori lengkuas. Nilai fitur-fitur lain pada kategori selain lengkuas diketahui mempunyai rentang nilai yang cukup dekat, sehingga performa sistem dalam mengenali kategori selain lengkuas lebih rendah dan mengakibatkan sistem memiliki nilai *accuracy* yang rendah. Hal tersebut dapat disimpulkan bahwa pemilihan jenis fitur bentuk sangat mempengaruhi hasil performa kinerja sistem identifikasi jenis tanaman obat berdasarkan bentuk citra daun. Selanjutnya, hasil uji coba tanpa menggunakan fitur panjang ternyata mampu meningkatkan nilai *accuracy* pada identifikasi 11 dan 16 jenis daun, misalnya identifikasi 16 jenis daun nilai accuracy umenjadi 58,75%, rata-rata *precision* 66,08%,

rata-rata *recall* 58,75%, dan rata-rata *f-measure* sebesar 56,58%. Uji coba tanpa fitur panjang dilakukan setelah menguji normalitas pada masing-masing fitur, disimpulkan bahwa nilai normalitas antar fitur juga mempengaruhi hasil akurasi pada identifikasi jenis tanaman obat menggunakan metode deteksi tepi dan *Gaussian Naïve Bayes.*

5.2. Saran

Dalam pengembangan sistem identifikasi jenis tanaman obat berdasarkan bentuk daun ini, dibutuhkan beberapa perbaikan dalam mencapai hasil yang lebih maksimal. Berikut merupakan saran kepada peneliti selanjutnya :

1. Penggunaan dataset yang kecil dapat menyebabkan hasil *accuracy* dan *precision* yang belum maksimal karena jumlah dataset setiap kategori untuk melatih sistem kurang representatif dan tidak cukup banyak. Oleh karena itu, disarankan untuk menggunakan dataset yang lebih banyak atau juga bisa melakukan teknik augmentasi seperti melakukan rotasi, *cropping*, *resize*, dan lainnya pada penelitian selanjutnya.
2. Pemilihan fitur yang lebih komprehensif dan korelasi antar fitur juga dapat mempengaruhi hasil performa kinerja sistem. Disarankan untuk penelitian selanjutnya dapat menggunakan fitur lain seperti *compactness, roundness* atau mengkombinasikan berbagai fitur bentuk lainnya agar sistem lebih banyak mempelajari lebih banyak informasi dalam melakukan identifikasi.
3. Melakukan percobaan menggunakan metode lain seperti *Neural Network*, *Support Vector Machine*, dan lain sebagain

DAFTAR PUSTAKA

Aulia Annisa Br Bangun, Achmad Fauzi, & Husnul Khair. (2022). Coconut Wood Density Image Processing Techniques Based On Texture Image With Comparison Of Sobel Edge Detection Algorithm And Canny Edge Detection Algorithm. *International Journal of Health Engineering and Technology*, *1*(2), 89–96. https://doi.org/10.55227/ijhet.v1i2.22

Bemando, C., Miranda, E., & Aryuni, M. (2021). Machine-Learning-Based Prediction Models of Coronary Heart Disease Using Naïve Bayes and Random Forest Algorithms. *Proceedings - 2021 International Conference on Software Engineering and Computer Systems and 4th International Conference on Computational Science and Information Management, ICSECS-ICOCSIM 2021*, *August 2021*, 232–237. https://doi.org/10.1109/ICSECS52883.2021.00049

Damayanti, M., & Adi, C. K. (2019). Pengenalan Daun Tanaman Obat Menggunakan Jaringan Syaraf Tiruan Backpropagation. *MEANS (Media Informasi Analisa dan Sistem)*, 98–103. https://doi.org/10.54367/means.v4i2.542

Dileep, M. R., & Pournami, P. N. (2019). AyurLeaf: A Deep Learning Approach for Classification of Medicinal Plants. *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, *2019-Octob*, 321–325. https://doi.org/10.1109/TENCON.2019.8929394

Duin, R. P. W., & Tax, D. M. J. (2005). Statistical pattern recognition. In *Handbook of Pattern Recognition and Computer Vision, 3rd Edition*. https://doi.org/10.1142/9789812775320_0001

Harefa, D. (2020). Pemanfaatan Hasil Tanaman Sebagai Tanaman Obat Keluarga (TOGA). *Madani : Indonesian Journal of Civil Society*, *2*(2), 28–36. https://doi.org/10.35970/madani.v2i2.233

Kurniawan Budhi, R., Prayitno, A., & Elvina, S. (2019). Pengenalan Pola Daun untuk Pendeteksi Dini Penyakit Tanaman Jagung Menggunakan Deteksi Tepi Sobel. *Seminar Nasional APTIKOM*, 340–346.

Liantoni, F., & Hermanto, L. A. (2017). Pengembangan Metode Ant Colony Optimization Pada Klasifikasi Tanaman Mangga Menggunakan K-Nearest Neighbor. *Jurnal Buana Informatika*, *8*(4), 193–200. https://doi.org/10.24002/jbi.v8i4.1443

Muhathir, M., Muliono, R., & Hafni, M. (2022). Image Classification of Autism Spectrum Disorder Children Using Naïve Bayes Method With Hog Feature Extraction. *Journal of Informatics and Telecommunication Engineering*, *5*(2), 494–501. https://doi.org/10.31289/jite.v5i2.6365

Padao, F. R. F., & Maravillas, E. A. (2016). Using Naïve Bayesian method for plant

leaf classification based on shape and texture features. *8th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, HNICEM 2015, December*. https://doi.org/10.1109/HNICEM.2015.7393179

Pratama, H. K. C. A., Suharso, W., & A'yun, Q. (2022). Pengklasifikasian Kanker Payudara Dan Kanker Paru-Paru Dengan Metode Gaussian Naïve Bayes , Multinomial Naïve Bayes , Dan Bernoulli Naïve Bayes. *Jurnal Smart Teknologi*, *3*(4), 350–355. http://jurnal.unmuhjember.ac.id/index.php/JST/article/view/7592/3886

Rahayuda, S. (2016). Identifikasi Jenis Obat Berdasarkan Gambar Logo Pada Kemasan Menggunakan Metode Naive Bayes. *Klik - Kumpulan Jurnal Ilmu Komputer*, *3*(2), 125. https://doi.org/10.20527/klik.v3i2.46

Sumayyah, S., & Nada, S. (2017). Obat tradisional : antara khasiat dan efek samping. *Majalah farmasetika*, *2*(5), 1–4.

LAMPIRAN

LAMPIRAN

Lampiran 1 Dataset dan data fitur daun

id	d1_panjang	d2_lebar	d3_luas	d4_keliling	d5_rasio	kategori
0	296	262.6119	53407	925.1705	1.127139	Cendana
1	341.0059	248.8956	49815	965.9474	1.370076	Cendana
2	333.4082	243.0514	51359	930.1416	1.37176	Cendana
3	363.0014	228.0022	54292	987.9581	1.592096	Cendana
4	336.3941	253.16	54594	949.5545	1.328781	Cendana
5	363.0675	225.1799	51149	1021.527	1.612344	Cendana
6	350.571	245.2937	54866	989.6731	1.429189	Cendana
7	351.7627	261.0939	55276	996.5899	1.347265	Cendana
8	360.1389	219.6725	48929	964.5854	1.639435	Cendana
9	313.6941	244.2949	48021	879.8963	1.28408	Cendana
10	332.0241	226	46053	915.1844	1.469133	Cendana
11	342.0409	232.6908	49346	932.0355	1.469938	Cendana
12	319.3305	250.9661	48303	3311.301	1.272405	Cendana
13	292.0274	259.6247	55877	900.4743	1.124806	Cendana
14	315.1587	249.3391	53514	914.401	1.263976	Cendana
15	318.498	226.0553	46002	873.6619	1.408939	Cendana
16	306.3234	260.5091	52454	921.4372	1.175864	Cendana
17	343.2098	234.062	48753	930.8106	1.466321	Cendana
18	306.0801	251.008	52275	899.2469	1.219404	Cendana
19	347.8577	221.1176	44412	920.3837	1.57318	Cendana
20	308.2337	249.8159	49785	913.4809	1.233843	Cendana
21	323.3033	256.7586	57924	947.492	1.259172	Cendana
22	309.3622	264.1855	55341	933.7859	1.171004	Cendana
23	318.6283	260.9693	49110	935.0793	1.220942	Cendana
24	345.9046	269.2675	51851	3387.047	1.284613	Cendana
25	311.8878	220.1113	42039	846.0501	1.416955	Cendana
26	301.8029	265.0679	58351	928.7668	1.138587	Cendana
27	325.9831	248.582	56888	957.3129	1.311371	Cendana
28	347.1959	255.0961	54801	987.8584	1.36104	Cendana
29	337.1483	228.1074	47644	924.4479	1.478024	Cendana
30	355.4434	212.6805	50092	934.6503	1.671255	Cendana
31	330.3105	250.018	58207	956.0502	1.321147	Cendana
32	351.1994	218.7419	45828	919.3319	1.605543	Cendana
33	334.4264	244.3461	52299	941.8642	1.368659	Cendana
34	329.0866	249.8259	54357	989.8799	1.317264	Cendana
35	342.2309	279.1218	63234	1006.196	1.226099	Cendana

36	359.5066	254.002	51309	940.476	1.415369	Cendana
37	333.4936	249.3592	49478	993.1213	1.337403	Cendana
38	326.2913	246.7752	52684	955.0388	1.322221	Cendana
39	335.1805	247.0729	48266	3958.362	1.356606	Cendana
40	326.5593	263.0969	57343	960.2171	1.241213	Cendana
41	337.2625	230.4886	50133	993.4578	1.46325	Cendana
42	364.0027	253.5508	60408	1023.578	1.435621	Cendana
43	335.872	222.7667	51539	925.2738	1.50773	Cendana
44	313.2699	232.0453	46156	893.3821	1.350038	Cendana
45	312.1939	248.1149	52382	922.8033	1.258263	Cendana
46	340.5877	247.002	57282	968.3018	1.378886	Cendana
47	349.5168	223.0807	49284	1208.949	1.566773	Cendana
48	344.3312	221.822	47911	902.6232	1.552286	Cendana
49	339.3317	236.3049	49156	923.4177	1.435991	Cendana
50	335.0239	255.7069	66264	970.1302	1.310187	Jambu Biji
51	350.1157	282.3473	75547	1031.787	1.240018	Jambu Biji
52	290.441	273.5288	63123	933.2336	1.06183	Jambu Biji
53	361.9351	238.0273	62449	968.762	1.520561	Jambu Biji
54	279.0448	270.2832	63177	928.4725	1.032416	Jambu Biji
55	336.8234	263.0019	67141	974.9988	1.280688	Jambu Biji
56	346.0925	277.0162	75209	1074.548	1.249358	Jambu Biji
57	291.1632	265.3771	62421	930.8667	1.097168	Jambu Biji
58	368.4807	266.3325	69580	1033.121	1.383536	Jambu Biji
59	367.3935	269.0019	71559	1076.297	1.365766	Jambu Biji
60	360.0889	254.2322	67211	1001.083	1.416378	Jambu Biji
61	343.0058	259.8557	62523	954.5856	1.319986	Jambu Biji
62	352.0511	256.1269	66949	988.4475	1.374518	Jambu Biji
63	343.7703	249.97	68099	989.3525	1.375246	Jambu Biji
64	346.325	242.2499	63201	958.173	1.429619	Jambu Biji
65	356.9762	266.0921	68295	1000.503	1.341551	Jambu Biji
66	347.829	236.1228	60927	954.622	1.473085	Jambu Biji
67	327.1101	252.343	61160	932.9117	1.296291	Jambu Biji
68	311.5461	273.5909	67397	964.1669	1.13873	Jambu Biji
69	343.0058	270.1481	69453	1010.633	1.269695	Jambu Biji
70	334.2933	266.0075	66900	973.0918	1.256706	Jambu Biji
71	343.3278	251.3663	68000	990.4597	1.365847	Jambu Biji
72	358.169	260.9904	69692	1001.744	1.372345	Jambu Biji
73	337.5352	258.6987	67875	973.1069	1.304743	Jambu Biji
74	344	270	67817	990.1168	1.274074	Jambu Biji
75	355.2689	224.6175	56594	933.5849	1.581662	Jambu Biji
76	342.3171	235.034	59569	940.5039	1.456458	Jambu Biji

77	337.2136	252.2856	68554	981.8478	1.336635	Jambu Biji
78	364.5847	269.2155	72127	1016.565	1.354248	Jambu Biji
79	350.2799	265.7461	69522	1002.969	1.3181	Jambu Biji
80	333.5386	245.1653	59159	928.4908	1.360464	Jambu Biji
81	366.1502	248.582	67228	1001.64	1.472956	Jambu Biji
82	348.4724	241.1327	60468	962.1596	1.445147	Jambu Biji
83	393.103	268.7936	78450	1083.258	1.462472	Jambu Biji
84	361.0346	266.3175	70982	1021.186	1.355655	Jambu Biji
85	288.1111	264.2726	61135	911.4479	1.090204	Jambu Biji
86	371.9099	276.6966	76024	1044.24	1.344107	Jambu Biji
87	361.6697	237.5079	60741	964.4594	1.522769	Jambu Biji
88	363.446	233.5487	60983	972.6095	1.556189	Jambu Biji
89	363.4969	240.4163	58958	961.7739	1.511948	Jambu Biji
90	390.5125	233.0193	64419	1028.675	1.67588	Jambu Biji
91	343.2463	259.3781	64059	978.0778	1.323343	Jambu Biji
92	303.0264	246.6759	55393	884.0433	1.228439	Jambu Biji
93	375.9814	279.8464	65463	1009.71	1.343528	Jambu Biji
94	350.0914	249.0723	64545	977.5957	1.405582	Jambu Biji
95	367.9348	230.5927	56078	946.1684	1.595605	Jambu Biji
96	362.5162	306.067	72751	1035.786	1.184434	Jambu Biji
97	350.3213	247.1639	63205	967.0737	1.417364	Jambu Biji
98	385.4205	243.1872	67369	1023.94	1.584872	Jambu Biji
99	347.6349	241.6775	63931	963.8844	1.438425	Jambu Biji
100	235.4167	252	42923	828.9277	0.934193	Mint
101	319.5951	261.0479	55928	1021.795	1.224278	Mint
102	271.1549	261.5033	50375	914.0708	1.036908	Mint
103	265.8759	227.9693	43417	840.0557	1.166279	Mint
104	221.3256	216.1874	33294	752.1962	1.023767	Mint
105	258.2344	239.4034	45613	858.3822	1.078658	Mint
106	248.582	254.1515	42456	849.1284	0.978086	Mint
107	253.239	225.938	40402	827.8769	1.120834	Mint
108	262.5509	223.7789	40492	860.3434	1.17326	Mint
109	295.061	255.2744	52372	934.6888	1.155858	Mint
110	220.1454	230.9567	35891	771.5258	0.953189	Mint
111	286.0857	238.4114	48741	920.2056	1.199966	Mint
112	264.1893	259.5092	49713	883.99	1.018035	Mint
113	270.6677	255.3272	46116	901.5522	1.060082	Mint
114	295.332	252.8577	53257	967.7766	1.167977	Mint
115	267.7611	259.6498	47714	904.871	1.031239	Mint
116	232.4414	220.7351	35091	749.9816	1.053033	Mint
117	238.0189	216.3169	38823	792.9697	1.100325	Mint

118	283.0866	259.5862	53075	949.0387	1.09053	Mint
119	283.2896	242.5964	46208	897.4189	1.16774	Mint
120	239.2091	217.6718	38617	816.1313	1.098944	Mint
121	294.2873	236.603	47670	947.62	1.243802	Mint
122	225.1422	211.4001	33758	752.2593	1.065005	Mint
123	280.4568	255.0961	49376	935.8538	1.099416	Mint
124	253.3338	231.6571	42058	848.4029	1.093572	Mint
125	248.0181	205.6502	38875	800.2445	1.20602	Mint
126	210.1166	200.1125	31847	712.8181	1.049993	Mint
127	296.1959	255.5015	50651	940.0293	1.159273	Mint
128	269.3641	220.0023	43814	868.7735	1.224369	Mint
129	222.1441	206.4219	34165	749.5452	1.076165	Mint
130	271.0074	266.0846	51121	899.1608	1.018501	Mint
131	235.4782	241.197	37256	778.9962	0.97629	Mint
132	245.0979	206.497	36865	787.3364	1.186932	Mint
133	260.0942	246.0772	45805	851.9975	1.056962	Mint
134	221.0814	198.0808	32865	734.1241	1.116117	Mint
135	275.2235	248.8875	48088	913.7005	1.105815	Mint
136	302.3326	256.2811	54021	982.0059	1.179691	Mint
137	291.3863	241.0996	46426	904.1108	1.208573	Mint
138	198.0808	207.1714	30250	696.4519	0.95612	Mint
139	256.7976	234.173	43524	851.8597	1.096615	Mint
140	306.3658	229.9652	47299	954.3558	1.332227	Mint
141	240.5203	218.0734	37099	794.5876	1.102933	Mint
142	291.0756	264.017	53966	949.53	1.102488	Mint
143	296.422	232.6908	46596	945.7853	1.273888	Mint
144	264.371	223.0202	40834	831.0663	1.185413	Mint
145	295.0017	243.6658	48819	906.2901	1.210682	Mint
146	276.0652	245.204	48062	894.7243	1.125859	Mint
147	269.0297	247.0992	49323	905.1474	1.088752	Mint
148	255.3272	262.4881	45451	880.4976	0.972719	Mint
149	296.0152	245.9614	49397	942.4726	1.203503	Mint
150	282.1135	235.1361	42833	832.709	1.199788	Bayam Hijau
151	263.2128	224.0357	39057	788.8204	1.17487	Bayam Hijau
152	260	230.8679	36465	779.3686	1.126185	Bayam Hijau
153	249.0642	251.4478	37486	781.6522	0.990521	Bayam Hijau
154	231.7801	233.9444	39376	756.9362	0.990748	Bayam Hijau
155	224.0357	230.0196	31543	708.3814	0.973985	Bayam Hijau
156	247.0729	222.036	37752	743.0502	1.11276	Bayam Hijau
157	225.0089	237.2551	32037	721.5039	0.948384	Bayam Hijau
158	258.745	229.264	36426	785.9053	1.12859	Bayam Hijau

159	305.2868	231.0541	38706	854.3853	1.321278	Bayam Hijau
160	237.6089	234.4803	33077	739.1488	1.013343	Bayam Hijau
161	236.0339	243.2961	35724	760.9637	0.970151	Bayam Hijau
162	259.5573	243.2488	39647	799.4131	1.067044	Bayam Hijau
163	283.4449	267.1516	43597	858.2153	1.060989	Bayam Hijau
164	272.0074	258.0174	42916	838.5567	1.054221	Bayam Hijau
165	310.2805	242.9094	45288	864.0524	1.277351	Bayam Hijau
166	330.097	235.0085	45571	911.4545	1.404617	Bayam Hijau
167	284.271	253.9547	40021	838.881	1.119377	Bayam Hijau
168	215.6965	239	35742	725.0679	0.902496	Bayam Hijau
169	253.0316	239.253	35778	776.8328	1.05759	Bayam Hijau
170	260.1922	234	43030	792.3807	1.111933	Bayam Hijau
171	290.4978	256.1269	44164	854.1356	1.134195	Bayam Hijau
172	247.0506	232.622	35757	756.5128	1.062026	Bayam Hijau
173	219.9113	210.238	30296	686.8512	1.046012	Bayam Hijau
174	238.6483	216.7487	34568	756.0573	1.101037	Bayam Hijau
175	302.0596	250.1619	47786	880.9653	1.207456	Bayam Hijau
176	232.0539	234	32455	728.0686	0.991683	Bayam Hijau
177	255.002	224.183	34266	765.7141	1.137473	Bayam Hijau
178	250.2878	240.0187	36382	759.5739	1.042785	Bayam Hijau
179	243.2961	241.0021	36469	759.6204	1.009519	Bayam Hijau
180	269.0669	223.6001	38739	792.3112	1.20334	Bayam Hijau
181	301.2009	226.0354	43913	835.6326	1.332539	Bayam Hijau
182	287.3917	240.2769	43077	842.3319	1.196085	Bayam Hijau
183	225.3198	233.2145	33385	715.0573	0.966148	Bayam Hijau
184	313.0144	260.3017	49538	901.0103	1.202506	Bayam Hijau
185	307.0016	232.6908	44639	845.0477	1.319354	Bayam Hijau
186	207.2414	229.0087	32856	693.8765	0.90495	Bayam Hijau
187	238.21	231.1385	35563	739.5105	1.030594	Bayam Hijau
188	249.002	236.3578	34031	764.5285	1.053496	Bayam Hijau
189	327.0061	227.1783	41231	862.6955	1.439425	Bayam Hijau
190	311.8589	235.8495	42759	862.6211	1.322279	Bayam Hijau
191	289.0138	256.4254	46798	851.4539	1.127087	Bayam Hijau
192	320.4076	232.1422	41164	858.9567	1.380221	Bayam Hijau
193	255.002	261.0172	39859	790.4577	0.976954	Bayam Hijau
194	232.2606	240.0083	34302	748.459	0.967719	Bayam Hijau
195	299.2207	267.5407	45253	894.4484	1.118412	Bayam Hijau
196	263.0019	242.4644	38837	793.7969	1.084703	Bayam Hijau
197	242.6685	235.0191	36505	752.0934	1.032548	Bayam Hijau
198	263.0304	243.5775	43428	808.3298	1.079863	Bayam Hijau
199	301.2125	233.3624	38983	840.142	1.29075	Bayam Hijau

200	346.0925	260.0308	58859	983.1967	1.330967	Jeruk
201	352.4103	208.4059	44954	922.5311	1.690981	Jeruk
202	257.381	223.3047	38641	770.1887	1.1526	Jeruk
203	340.5305	262.0859	54293	1002.789	1.299309	Jeruk
204	312.578	230.1521	49586	871.285	1.358136	Jeruk
205	235	194.1391	29978	694.0847	1.210472	Jeruk
206	332.0241	259.326	56883	960.9685	1.280335	Jeruk
207	248.002	209.0096	32993	738.3877	1.186558	Jeruk
208	361.3323	225.938	48901	963.0016	1.599254	Jeruk
209	258.0271	196.4714	30121	729.3107	1.313307	Jeruk
210	209.1602	154.0552	20156	598.9911	1.357697	Jeruk
211	224.183	206.4098	31153	689.1477	1.086106	Jeruk
212	319.6248	236.2562	51715	898.4259	1.352874	Jeruk
213	328.2392	249.4795	48572	940.1365	1.315696	Jeruk
214	284.0158	212.0424	39016	805.8555	1.339429	Jeruk
215	291.755	232.0194	44026	856.0939	1.25746	Jeruk
216	260.0019	188.3215	30353	726.2098	1.380628	Jeruk
217	333.9371	253.2686	54376	967.5497	1.31851	Jeruk
218	301.0066	243.8524	52763	872.9013	1.23438	Jeruk
219	330.9637	208.9521	41334	871.054	1.583921	Jeruk
220	315.5139	237.0759	43687	906.9746	1.330856	Jeruk
221	294.4622	230.0087	48346	834.5411	1.280222	Jeruk
222	305.1049	239.2864	45570	884.2295	1.275061	Jeruk
223	316.2673	182.8579	33130	839.9962	1.72958	Jeruk
224	293.5524	231.1471	44674	861.2992	1.269981	Jeruk
225	252.0397	220.1931	37473	777.7794	1.14463	Jeruk
226	286.5659	209.2678	42074	797.2533	1.369374	Jeruk
227	360.5135	258.031	59659	1007.41	1.397171	Jeruk
228	344.1889	240.0104	49980	949.4587	1.434058	Jeruk
229	241.5305	181.604	25715	678.2649	1.329985	Jeruk
230	248.1149	203.6296	34889	769.6721	1.218462	Jeruk
231	304.2006	236.0763	49659	866.6542	1.288569	Jeruk
232	355.3548	265.3771	60133	994.381	1.339056	Jeruk
233	324.9815	256.2226	53095	956.8892	1.268356	Jeruk
234	348.5585	213.565	43447	917.1968	1.632096	Jeruk
235	246.9514	204.0613	33080	750.3694	1.210183	Jeruk
236	247.6065	206.0607	30964	745.0018	1.20162	Jeruk
237	312.0032	239.0188	52032	883.5344	1.30535	Jeruk
238	305.7908	210.0119	39284	851.4465	1.456064	Jeruk
239	312.1025	247.2023	48130	911.8706	1.262539	Jeruk
240	327.0015	256.1425	53214	964.1996	1.276639	Jeruk

241	296.3393	182.5842	34268	796.6491	1.623028	Jeruk
242	342.9402	254.0079	55789	995.4387	1.350117	Jeruk
243	272.1838	226.7973	41886	835.8086	1.200119	Jeruk
244	228.0789	214.0374	33602	728.9253	1.065603	Jeruk
245	280.1446	216.8156	42597	795.0393	1.292087	Jeruk
246	224.8911	183.6137	25115	677.7256	1.224805	Jeruk
247	324.2221	239.0523	49304	941.4747	1.356281	Jeruk
248	353.4586	267.1872	57777	1007.067	1.322888	Jeruk
249	294.2873	211.0379	43185	816.3202	1.394476	Jeruk
250	379.6972	159.0126	30902	882.1737	2.387844	Mangga
251	379.0356	94.00532	25366	853.2525	4.032066	Mangga
252	385.1052	129.7998	26264	872.6702	2.966916	Mangga
253	366.1871	227.706	26553	855.9863	1.608158	Mangga
254	384.0326	97.14422	21458	856.6646	3.953221	Mangga
255	370.0608	113.1371	27545	836.0581	3.270906	Mangga
256	384.0013	121.2023	27650	879.8731	3.168267	Mangga
257	357.14	89.80535	21593	794.8011	3.976824	Mangga
258	368.0489	107.0187	25825	836.134	3.439109	Mangga
259	377.7367	203.9068	24230	857.8474	1.852496	Mangga
260	386.0207	100.1848	22634	856.1148	3.853086	Mangga
261	377.0053	99.40322	24154	838.9432	3.792687	Mangga
262	378.1918	159.7655	31792	892.9692	2.367169	Mangga
263	385.873	167.0269	26171	879.184	2.310244	Mangga
264	375.7047	114.4378	28576	852.0581	3.283048	Mangga
265	387.1343	310.1548	23828	865.4266	1.248197	Mangga
266	336.0729	153.1176	19832	754.0109	2.194868	Mangga
267	373.8369	115.0391	24777	849.7324	3.24965	Mangga
268	378.2235	111.018	27316	858.0844	3.406866	Mangga
269	382.4265	167.2603	26655	867.0926	2.286415	Mangga
270	382.6042	111.1396	24225	849.529	3.442557	Mangga
271	380.3367	120.2082	23883	847.0077	3.163984	Mangga
272	380.6954	129.3561	28483	862.5434	2.943003	Mangga
273	367.0054	100.4042	24718	830.1455	3.65528	Mangga
274	395.9217	390.6981	23527	868.8561	1.01337	Mangga
275	384.2551	109.0046	25712	863.6733	3.525128	Mangga
276	368.3056	107.2007	25495	827.5575	3.435662	Mangga
277	378.8034	145.6915	23069	841.6711	2.600038	Mangga
278	360.0111	222.7846	27576	840.089	1.61596	Mangga
279	385.7305	193.4141	28197	873.1922	1.994325	Mangga
280	364.5888	94.36631	22530	812.6001	3.863549	Mangga
281	383.2623	189.1692	24193	853.4078	2.026029	Mangga

282	416.1118	401.1508	31670	945.7163	1.037295	Mangga
283	380.0118	141.8203	28631	863.3981	2.67953	Mangga
284	399.1115	331.1269	26949	894.4958	1.205313	Mangga
285	374.0214	115.4513	25353	841.8646	3.239647	Mangga
286	374.6465	204.1299	24890	850.5697	1.835334	Mangga
287	364.4448	110.4355	22647	818.2438	3.300069	Mangga
288	368.9173	109.252	23999	827.2504	3.376756	Mangga
289	388.4147	249.8259	28650	880.2175	1.554741	Mangga
290	366.546	98.32599	23328	831.2428	3.727865	Mangga
291	390.2666	170.0735	21643	861.529	2.294693	Mangga
292	379.0053	203.5903	25475	877.3932	1.861608	Mangga
293	381.0682	231.931	26106	854.1905	1.643024	Mangga
294	381.4826	255.0686	24576	856.4332	1.495608	Mangga
295	369.0054	103.0776	24462	831.0231	3.579878	Mangga
296	363.0881	88.28363	20530	807.4382	4.112746	Mangga
297	347.908	185.3483	27382	822.7707	1.87705	Mangga
298	377.7632	133.1803	31625	878.8987	2.836479	Mangga
299	376.4944	207.2342	25805	846.9905	1.816758	Mangga
300	281.0071	265.0019	40371	1073.479	1.060397	Ara
301	270.4163	258.0698	35288	870.6999	1.047842	Ara
302	275.2199	251.0976	36594	876.5223	1.096068	Ara
303	278.4978	270.2665	40372	943.1025	1.030456	Ara
304	260.4496	280.1446	39910	894.6877	0.929697	Ara
305	253.0613	271.249	35266	855.0977	0.932948	Ara
306	268.1865	267.479	36753	1458.822	1.002645	Ara
307	264.282	271.0166	37317	909.146	0.975151	Ara
308	265.4826	267.479	36638	902.5697	0.992536	Ara
309	264.0076	271	39895	896.2561	0.974198	Ara
310	261.3752	266.4226	38613	881.5788	0.981055	Ara
311	253.002	267.03	33447	861.9344	0.947467	Ara
312	273.3587	259.0309	37552	896.372	1.055313	Ara
313	267.0674	270.3128	41243	888.087	0.987994	Ara
314	287.696	261.4307	36285	931.5157	1.100468	Ara
315	243.7417	235	32248	809.754	1.037199	Ara
316	278.0072	256.0957	36041	940.0228	1.08556	Ara
317	257.3908	264.0019	35187	888.8382	0.974958	Ara
318	260.6396	268.0466	38656	890.2273	0.972367	Ara
319	261.4441	250.018	30481	996.0104	1.045701	Ara
320	262.0019	271.4148	39861	894.7433	0.965319	Ara
321	263.7518	266.0019	40303	1739.363	0.991541	Ara
322	294.0017	273.0897	38752	970.8373	1.076575	Ara

323	263.3875	262.2747	35341	896.7454	1.004243	Ara
324	250.06	250.018	32178	839.0388	1.000168	Ara
325	280.3016	271.2232	41739	963.6607	1.033472	Ara
326	293.0683	256.4391	36001	913.8068	1.142838	Ara
327	262.0305	252	33240	923.1037	1.039804	Ara
328	248.2438	265.6803	35758	851.5563	0.934371	Ara
329	268.8438	263.1216	38351	929.5739	1.021747	Ara
330	256.1269	268.1865	38305	856.5774	0.955033	Ara
331	265.7536	275.1472	37444	890.6118	0.96586	Ara
332	241.2509	270	37702	849.427	0.893522	Ara
333	259.9327	257.0311	30698	871.927	1.011289	Ara
334	257.1225	269	34444	870.4439	0.955846	Ara
335	255.049	272.1489	37762	867.2212	0.937167	Ara
336	252.0317	251.7956	35238	830.4237	1.000938	Ara
337	229.4275	257.0175	35291	803.4271	0.892653	Ara
338	257.4374	279.5174	40767	1161.774	0.921006	Ara
339	276.7237	258.2344	39150	910.5648	1.071599	Ara
340	274.8545	278.0288	39209	894.4407	0.988583	Ara
341	233.1051	242.2065	31088	793.7161	0.962423	Ara
342	308.7847	253.0711	40282	911.5718	1.22015	Ara
343	253.3377	238.2541	31376	840.4377	1.063309	Ara
344	245	268.0672	36020	1056.833	0.91395	Ara
345	259.7383	257.07	32683	895.5298	1.01038	Ara
346	252.0516	284.2129	39890	913.891	0.886841	Ara
347	294.8644	243.8217	35254	901.8935	1.209344	Ara
348	245.0082	234.3075	28011	790.1764	1.045669	Ara
349	259.193	265.6803	35622	866.6138	0.975582	Ara
350	278.8422	224.9556	40236	814.3119	1.239544	Kari
351	317.1277	247.0729	49222	892.1171	1.283539	Kari
352	135.6245	184.0027	15398	498.8025	0.737079	Kari
353	307.9951	222.4455	41424	849.133	1.384587	Kari
354	322.6717	188.5099	36697	859.6434	1.711696	Kari
355	114.1271	178.6393	12496	466.1019	0.638869	Kari
356	292.0274	183.1202	28987	761.903	1.594731	Kari
357	281.6842	134.8369	20749	678.3621	2.089073	Kari
358	275.3561	125.1719	19030	660.126	2.199824	Kari
359	335.44	247.0971	52959	951.1282	1.357523	Kari
360	296.3849	231.4584	45249	855.2316	1.28051	Kari
361	289.9948	235.9068	40079	840.0821	1.229277	Kari
362	159.8312	217.3891	18214	577.5911	0.735231	Kari
363	281.8102	179.1787	32580	746.1816	1.572789	Kari

364	101.9657	163.591	10269	414.4281	0.623297	Kari
365	114.2804	161.9938	11344	429.8841	0.705461	Kari
366	370.4187	226.4972	48970	957.1521	1.635422	Kari
367	152.345	270.0667	22317	1151.369	0.564101	Kari
368	191.1256	160.1624	19956	556.1526	1.193324	Kari
369	250.7848	155.8717	15948	600.384	1.608917	Kari
370	277.1804	244.1741	41616	824.6135	1.135175	Kari
371	345.2144	239.0983	56774	981.6486	1.443818	Kari
372	338.0961	164.0488	28259	809.1448	2.060949	Kari
373	333.0961	218.33	46929	901.3838	1.525654	Kari
374	348.0833	252.5173	52013	970.2026	1.378453	Kari
375	309.0955	159.8124	27603	769.6638	1.934114	Kari
376	311.7178	251.0339	48138	903.8569	1.241736	Kari
377	331.7981	238.0462	43865	888.9331	1.393839	Kari
378	327.9146	183.0683	39015	864.7457	1.791215	Kari
379	326.443	233.5209	48640	925.8198	1.397918	Kari
380	184.2715	125.1	14528	503.1697	1.472994	Kari
381	249.3592	172.4471	22905	656.6501	1.446004	Kari
382	243.8237	147.3533	19320	642.6469	1.654688	Kari
383	208.4922	116.619	15594	532.7433	1.787806	Kari
384	262.9449	242.5964	39835	777.7496	1.083878	Kari
385	307.3711	220.2045	41591	872.6572	1.395844	Kari
386	182.7184	141.0035	18041	520.5739	1.295842	Kari
387	279.3313	232.0345	38337	793.0037	1.203835	Kari
388	299.4879	183.3085	25116	742.7072	1.633792	Kari
389	343.0364	241.514	50926	934.3239	1.420359	Kari
390	317.8333	192.0937	36990	823.0026	1.654574	Kari
391	345.0362	238.0525	48595	922.0314	1.449412	Kari
392	325.0738	213.6773	45395	885.4154	1.521331	Kari
393	362.2499	223.6359	54142	980.2149	1.61982	Kari
394	301.735	228.2652	43978	857.5198	1.321862	Kari
395	316.5122	244.806	47299	907.1809	1.29291	Kari
396	323.311	231.0541	49341	913.5697	1.399287	Kari
397	326.5593	264.3936	50459	928.9719	1.235126	Kari
398	304.7655	248.131	46109	896.5327	1.228244	Kari
399	320.3389	231.0779	44913	891.1731	1.386281	Kari
400	205.7596	179.6274	24943	595.5263	1.14548	Kelor
401	193.6621	192.8419	25851	597.206	1.004253	Kelor
402	239.2091	202.0099	32002	685.9887	1.184145	Kelor
403	169.0266	185.3888	22701	554.5565	0.911741	Kelor
404	207.6174	194.0412	30477	642.4259	1.069966	Kelor

405	155.7241	139.0899	15493	461.9172	1.119593	Kelor
406	240.2082	231.2163	38881	734.4071	1.03889	Kelor
407	182.7922	187.3846	24206	575.8189	0.975492	Kelor
408	215.0372	191	27913	629.3868	1.125849	Kelor
409	184.1738	177.8145	23499	565.3011	1.035764	Kelor
410	201.7176	165.003	22076	574.8157	1.222509	Kelor
411	236.0275	219.0023	37008	717.0505	1.07774	Kelor
412	240.7509	232.3639	37892	733.2976	1.036094	Kelor
413	175.3853	155.158	20739	532.2466	1.130366	Kelor
414	252.8735	232.8605	40181	749.0243	1.085944	Kelor
415	185.5586	170.9503	24975	582.0461	1.085454	Kelor
416	228.2652	220	34750	696.3094	1.037569	Kelor
417	233.8568	193.3727	30614	661.0571	1.209358	Kelor
418	246.002	237.1034	41065	752.7705	1.037531	Kelor
419	215.0279	197.0913	29455	640.5159	1.091006	Kelor
420	213.1901	209.0096	31708	656.2406	1.020001	Kelor
421	242.0744	210.1928	35119	701.688	1.151678	Kelor
422	259.85	233.2145	39631	764.699	1.11421	Kelor
423	245.1	242.6685	43851	772.1443	1.01002	Kelor
424	236.1715	216.1134	34434	707.6773	1.092813	Kelor
425	209.6187	204.0882	29916	641.6226	1.027099	Kelor
426	225.8871	206.0097	31319	663.9786	1.096488	Kelor
427	250.2179	225.3908	37863	743.942	1.110152	Kelor
428	248.0726	229.1397	37183	733.4299	1.082626	Kelor
429	272.2646	248.3948	47083	807.3844	1.096096	Kelor
430	209.6139	196	28022	629.0276	1.069459	Kelor
431	250.032	252.1607	42515	783.3493	0.991558	Kelor
432	220.0568	208.137	35007	689.1397	1.057269	Kelor
433	217.175	216.2314	32896	679.2448	1.004364	Kelor
434	247.0182	236.0763	40008	753.0474	1.046349	Kelor
435	229.3142	205.4118	32512	678.1364	1.116363	Kelor
436	267.8246	265.1207	47509	829.696	1.010199	Kelor
437	242.9979	221.0543	40533	745.1872	1.099268	Kelor
438	238.6797	226.8744	36561	721.9851	1.052034	Kelor
439	242.0331	244.1004	43932	776.0851	0.991531	Kelor
440	207.8702	191.7551	29921	640.725	1.08404	Kelor
441	233.0536	228.3156	40543	739.256	1.020752	Kelor
442	236.1038	215.1883	35571	713.6605	1.097196	Kelor
443	234.2691	218.799	39553	731.6884	1.070705	Kelor
444	262.3223	219.7908	39304	752.5047	1.193509	Kelor
445	233.1051	191.5881	30248	665.1732	1.216699	Kelor

446	245.7824	212.8121	32881	703.4566	1.154927	Kelor
447	257.777	225.02	38643	749.3788	1.145574	Kelor
448	209.5376	247.1639	39503	738.3276	0.847768	Kelor
449	268.1641	236.5798	46117	814.526	1.133504	Kelor
450	353.5534	200	45526	885.1585	1.767767	Delima
451	344.3269	211.0924	49513	1026.131	1.631167	Delima
452	358.9596	191.3661	46406	893.927	1.875774	Delima
453	350.0686	211.0095	48046	911.6108	1.659018	Delima
454	358.1396	187.984	43748	941.9162	1.90516	Delima
455	386.598	267.7555	52005	962.1243	1.443847	Delima
456	309.0793	257.1303	50297	1399.334	1.202034	Delima
457	354.1073	152.3975	32855	906.2391	2.323577	Delima
458	349.0458	184.201	43187	868.9803	1.894919	Delima
459	364.0275	240.6824	43888	907.2463	1.512481	Delima
460	357.5598	186.325	45389	890.7878	1.919011	Delima
461	347.5198	216.8986	41672	872.0396	1.602222	Delima
462	367.6955	194.2395	43857	905.1201	1.893	Delima
463	330.6675	172.3514	37269	1135.193	1.918566	Delima
464	350.297	228.0351	48814	902.6388	1.536154	Delima
465	370.1959	208.3195	51915	935.0493	1.777059	Delima
466	381.0472	173.1849	41888	948.6131	2.200234	Delima
467	368.8794	196.0918	47748	926.2902	1.881156	Delima
468	324.5689	227.9825	52541	907.526	1.423658	Delima
469	355.1296	183.807	42831	875.0206	1.932079	Delima
470	355.9284	213.3542	50346	955.4265	1.668251	Delima
471	354.9338	161.0124	40324	877.5707	2.204388	Delima
472	374.7119	207.0411	52664	958.0015	1.809843	Delima
473	367.535	183.9918	45212	902.8484	1.997562	Delima
474	363.0675	176.1391	43327	931.4579	2.061254	Delima
475	356.3664	195.7575	43356	887.3003	1.820448	Delima
476	364.4448	194.8333	46687	2010.903	1.870547	Delima
477	374.477	216.0046	37210	2905.541	1.733653	Delima
478	360.2249	196.3797	44515	904.0659	1.834328	Delima
479	338.8126	182.387	40446	839.2281	1.857658	Delima
480	372.7103	214.1822	45309	2634.379	1.740156	Delima
481	345.5836	212.8967	42620	867.0501	1.623245	Delima
482	367.448	223.0202	49855	929.1899	1.6476	Delima
483	354.391	210.0905	42956	883.5405	1.68685	Delima
484	361.1122	197.7321	47136	904.7083	1.826269	Delima
485	343.2463	172.2905	39761	887.4492	1.992254	Delima
486	304.3846	229.1724	43272	854.4226	1.32819	Delima

487	338.8126	188.9047	42036	1149.822	1.793563	Delima
488	354.2767	174.0259	41613	885.9681	2.035771	Delima
489	377.2347	206.7849	50660	957.0893	1.824285	Delima
490	368.402	206.526	51211	942.5124	1.783804	Delima
491	356.3846	193.3106	43666	896.7074	1.843585	Delima
492	356.2022	166.1475	39805	876.2972	2.143891	Delima
493	343	209.4039	41635	895.4968	1.637983	Delima
494	373.1863	229.4275	51045	955.2096	1.626598	Delima
495	370.1094	196.7257	48363	930.7104	1.881348	Delima
496	346.3697	187.2165	44435	876.4262	1.850103	Delima
497	363.2699	196.367	48027	927.5115	1.849954	Delima
498	367.6425	188.0665	46745	915.8094	1.954854	Delima
499	346.3697	224.2878	54133	928.7623	1.54431	Delima
500	316.2278	267.9048	50913	913.9546	1.180373	Malapari
501	296.076	261.8435	49708	872.7484	1.130737	Malapari
502	290.5581	274.5269	45908	892.188	1.058396	Malapari
503	289.762	275.525	45844	889.2901	1.051672	Malapari
504	351.4598	264.371	60243	987.1837	1.329419	Malapari
505	283.2984	263.947	42137	859.0272	1.073316	Malapari
506	318.0393	265.1207	54453	938.5951	1.199602	Malapari
507	318.509	258.0078	51173	910.2744	1.234494	Malapari
508	340.7007	258.6271	52986	960.1579	1.317343	Malapari
509	348.1436	256.125	57630	970.1668	1.359273	Malapari
510	290.3532	272.7343	48778	880.4956	1.064601	Malapari
511	318.9044	265.4826	51501	921.0695	1.201225	Malapari
512	308.0146	277.9083	55493	922.303	1.108332	Malapari
513	366.2308	250.6472	57434	992.2831	1.461141	Malapari
514	328.3733	275.7481	54182	945.738	1.190845	Malapari
515	288.0278	273.0568	45952	874.4814	1.054827	Malapari
516	349.0014	253.1265	53047	964.4894	1.378763	Malapari
517	343.0714	251	54927	947.2716	1.366818	Malapari
518	345.2448	271.9577	57821	968.7548	1.26948	Malapari
519	360.0014	243.0514	55912	970.7895	1.481174	Malapari
520	346.7636	259.5149	56457	979.6556	1.336199	Malapari
521	330.1833	268.8438	55399	950.8203	1.22816	Malapari
522	333.054	262.0534	52035	943.9231	1.270939	Malapari
523	251.1275	274.0529	41586	813.1563	0.916347	Malapari
524	303.033	281.6416	50552	919.6696	1.075953	Malapari
525	335.0895	257.3286	55492	960.9223	1.302185	Malapari
526	330.1227	254.1594	49163	922.5206	1.298881	Malapari
527	328.0015	245.0979	48338	906.3749	1.338247	Malapari

528	297.6726	280.0643	51050	898.435	1.062873	Malapari
529	377.0212	213.2346	50804	969.7292	1.768105	Malapari
530	304.2367	270.6677	48361	914.335	1.124023	Malapari
531	329.947	261.0172	50841	931.079	1.264081	Malapari
532	345.2086	244.524	54012	936.0021	1.411758	Malapari
533	369.0664	237.0084	53498	978.187	1.557187	Malapari
534	290	273.4118	49188	883.3962	1.060671	Malapari
535	290.7594	272.0092	45040	874.5411	1.068932	Malapari
536	334.0958	269	59089	960.2203	1.241992	Malapari
537	343.0233	266.3175	55292	956.585	1.288024	Malapari
538	299.1321	276.0072	51150	887.848	1.083783	Malapari
539	330.1833	241.1327	47070	912.1982	1.369301	Malapari
540	274.3319	265.9323	49544	854.243	1.031585	Malapari
541	318.1902	274.0657	52164	917.0282	1.161	Malapari
542	312.0785	272.4427	55259	926.2813	1.145483	Malapari
543	339.2123	280.0714	61318	990.6361	1.211164	Malapari
544	368.11	231.8124	53092	963.0094	1.587965	Malapari
545	332.6635	272.3894	50309	949.0822	1.221279	Malapari
546	313.0064	270.0167	56206	931.7532	1.159211	Malapari
547	309.1618	278.4116	56286	930.6523	1.110449	Malapari
548	326.0552	269.8185	53717	947.0981	1.208424	Malapari
549	371.4862	218.2773	51484	963.481	1.7019	Malapari
550	337.3781	241.8677	51840	924.3207	1.394887	Sirih
551	291.2044	266.6927	47074	863.5823	1.09191	Sirih
552	291.2405	263.4862	48093	882.7539	1.105335	Sirih
553	329.4753	245.6176	50867	903.5362	1.341416	Sirih
554	255.1431	270.0667	46413	830.6841	0.944741	Sirih
555	289.3752	255.0078	46289	869.7099	1.13477	Sirih
556	270.2832	257.1945	44531	829.6255	1.05089	Sirih
557	257.4374	249.0502	41018	803.2612	1.033677	Sirih
558	266.0169	273.3587	48778	959.8346	0.973142	Sirih
559	262.0019	272.3105	44583	836.2191	0.962144	Sirih
560	331.9578	240.202	51514	913.9714	1.381994	Sirih
561	283.4237	267.1516	47345	867.7328	1.060909	Sirih
562	290.2482	243.0741	48088	849.083	1.194073	Sirih
563	272.0901	264.282	47948	845.775	1.029544	Sirih
564	229.2968	266.4808	40202	776.3243	0.860463	Sirih
565	268.628	270.5347	47702	852.086	0.992952	Sirih
566	241.814	274.1168	46039	805.2992	0.882157	Sirih
567	266.7508	248.0988	42425	814.7294	1.07518	Sirih
568	279.5085	280.0446	47122	868.1936	0.998086	Sirih

569	257.7014	256.3299	41626	811.6197	1.005351	Sirih
570	246.7144	274.4103	44214	824.1267	0.899071	Sirih
571	200.2623	262.0935	37409	742.5746	0.764087	Sirih
572	328.2986	176.3434	40845	844.8451	1.861701	Sirih
573	362.1988	191.1675	47304	1328.227	1.894668	Sirih
574	388.7261	197.0228	53106	1493.329	1.973	Sirih
575	386.246	182.0687	47944	971.0242	2.121431	Sirih
576	374.0214	168.1071	42983	917.393	2.224899	Sirih
577	350.287	158.1139	36671	885.7405	2.21541	Sirih
578	377.5341	146.7583	37538	967.0641	2.572489	Sirih
579	386.5876	186.1182	49985	966.9351	2.077108	Sirih
580	391.6031	181.5406	50169	992.1905	2.15711	Sirih
581	352.6315	252.127	59652	974.7295	1.398627	Sirih
582	373.2265	141.8767	33971	882.7735	2.63064	Sirih
583	355.1704	146	33494	854.3098	2.432674	Sirih
584	379.4272	177.0254	42962	2931.445	2.143349	Sirih
585	369.6282	183.1748	46080	937.8398	2.017899	Sirih
586	384.3345	192.0339	47887	1661.006	2.001389	Sirih
587	384.6882	176.1391	48237	955.9125	2.184002	Sirih
588	362.7465	173.4157	43846	908.9664	2.091774	Sirih
589	381.3791	173.7872	27783	1524.562	2.194517	Sirih
590	384.2929	172.0727	43162	1036.765	2.233317	Sirih
591	357.1358	145.4957	34135	856.532	2.454614	Sirih
592	356.2752	145.031	33244	857.3051	2.456544	Sirih
593	363.3249	153.0523	37278	875.556	2.373862	Sirih
594	373.6482	172.4181	42912	925.4787	2.167106	Sirih
595	372.2633	166.6763	38056	911.5087	2.233451	Sirih
596	371.4364	169.4344	41965	920.4254	2.192214	Sirih
597	383.5753	198.7285	50284	995.4558	1.930148	Sirih
598	346.8314	178.5497	44149	881.987	1.942492	Sirih
599	396.3395	216.1134	59760	1020.779	1.833942	Sirih
600	340.802	212.7628	46758	886.2388	1.601793	Nangka
601	325.1246	235.9195	54010	898.3468	1.378117	Nangka
602	348.8624	249.0642	59454	958.6842	1.400693	Nangka
603	352.6259	258.8532	60375	1268.077	1.362262	Nangka
604	355.0282	261.49	64641	998.1124	1.357712	Nangka
605	355.2577	268.0019	59377	1548.803	1.325579	Nangka
606	341.7967	231.0022	57999	930.9816	1.479626	Nangka
607	324.2345	252.0397	60605	927.5191	1.286442	Nangka
608	349.0358	272.0607	66018	1000.129	1.282934	Nangka
609	351.1937	237.1181	53205	928.2763	1.481092	Nangka

610	347.7945	235.3317	55233	933.3836	1.477891	Nangka
611	321.1682	250.7449	54683	913.0376	1.280856	Nangka
612	341.4103	271.4719	63475	999.0788	1.257627	Nangka
613	318.9122	245.5891	56802	906.9646	1.29856	Nangka
614	352.3209	240.3518	54503	943.3731	1.465855	Nangka
615	341.0528	240.8527	50194	899.3599	1.416023	Nangka
616	338.5159	209.1602	46750	878.5109	1.618452	Nangka
617	337.7129	252.002	54002	919.5127	1.34012	Nangka
618	340.9076	259.3781	61074	980.5779	1.314327	Nangka
619	346.6713	262.0305	61834	971.521	1.323019	Nangka
620	354.4644	228.7575	50570	932.3749	1.54952	Nangka
621	361.7955	262.9544	64788	1034.502	1.375887	Nangka
622	304.0164	262.4881	54364	958.695	1.15821	Nangka
623	354.2033	264.3728	61420	978.0399	1.339787	Nangka
624	339.2123	270.7397	65371	1006.373	1.252909	Nangka
625	349.0014	268.9108	62891	998.0177	1.297834	Nangka
626	304.2778	257.9399	55901	898.7613	1.179646	Nangka
627	340.1558	240.8319	53048	957.4832	1.41242	Nangka
628	338.3327	258.9537	60189	954.3989	1.306538	Nangka
629	339.1342	239.1339	56728	954.9431	1.418177	Nangka
630	367.9266	253.16	57084	970.0284	1.453336	Nangka
631	391.0051	255.2822	66340	1058.087	1.531658	Nangka
632	339.1784	263.5318	60791	967.0306	1.287049	Nangka
633	349.4639	228.0088	55593	970.4673	1.532677	Nangka
634	368.5987	248.1633	58553	2383.325	1.485307	Nangka
635	351.0057	267.479	64972	980.3298	1.312274	Nangka
636	352.3634	264.3199	65216	984.3531	1.333095	Nangka
637	351.0513	278.9193	65554	990.9045	1.258612	Nangka
638	374.7666	246.5218	60555	996.5069	1.520217	Nangka
639	348.7865	239.2697	52810	935.2149	1.457713	Nangka
640	318.2342	257.0175	57101	925.7528	1.238181	Nangka
641	355.0014	228.2652	53888	943.5211	1.555215	Nangka
642	351.7456	249.5756	61826	971.6461	1.409375	Nangka
643	361.0222	265.2772	69792	1021.621	1.360924	Nangka
644	352.3195	262.4881	61197	980.1939	1.34223	Nangka
645	357.5933	226.6142	52615	964.1893	1.577983	Nangka
646	351.0698	257.9399	63645	980.4357	1.361053	Nangka
647	347.1974	267.0805	64817	997.0675	1.299973	Nangka
648	383.0117	255.0176	68920	1040.942	1.501903	Nangka
649	344.6926	250.056	58739	949.8146	1.378462	Nangka
650	351.0912	196.0867	41698	903.3676	1.790489	Kersen

651	371.8562	215.2812	52258	986.7122	1.727304	Kersen
652	369.4239	209.6592	51325	1002.261	1.76202	Kersen
653	346.9784	198.4943	42661	913.3567	1.748052	Kersen
654	357.8743	195.4329	43491	918.3895	1.831188	Kersen
655	360.4844	223.0807	46895	952.4036	1.615937	Kersen
656	360.4691	209.6521	42266	918.2787	1.719368	Kersen
657	357.1694	191.5646	37150	890.9971	1.864485	Kersen
658	346.0925	193.2149	39028	885.7602	1.791231	Kersen
659	374.4823	217.0023	44862	960.268	1.725707	Kersen
660	385.5723	247.1356	53711	1005.82	1.560165	Kersen
661	356.6791	200.8606	40494	915.7758	1.775754	Kersen
662	357.1064	205.2438	44744	942.8916	1.739914	Kersen
663	366.1106	200.125	42857	947.479	1.82941	Kersen
664	360.867	196.0944	41439	922.3487	1.840272	Kersen
665	352.0014	212.7346	48511	931.7669	1.654651	Kersen
666	367.7662	176.2839	43275	945.0578	2.086216	Kersen
667	365.6788	205.7304	44174	927.6162	1.777466	Kersen
668	348.5585	179.9778	40768	896.2476	1.936675	Kersen
669	379.0844	255.8906	53914	1008.394	1.481432	Kersen
670	377.4652	214.8488	45835	966.4642	1.756888	Kersen
671	362.0055	210.9502	47271	951.1967	1.716071	Kersen
672	359.451	213.244	49551	949.9066	1.685632	Kersen
673	346.5747	191.4628	42115	903.7027	1.810141	Kersen
674	348.415	161.1118	37277	873.2045	2.162567	Kersen
675	381.6622	235.682	48101	987.9351	1.619395	Kersen
676	375.3851	206.1844	46256	983.4725	1.820628	Kersen
677	366.3509	233.2059	41482	933.2335	1.570933	Kersen
678	356.6188	214.6742	45002	932.7617	1.66121	Kersen
679	374.3394	229.0022	47102	976.7514	1.634654	Kersen
680	351.5693	174.1953	42093	902.0483	2.018248	Kersen
681	336.8828	223.58	42720	888.9744	1.506766	Kersen
682	347.5198	202.2474	49372	939.9145	1.718291	Kersen
683	354.408	209.6974	42432	927.9504	1.690092	Kersen
684	352.8867	222.4635	43223	929.5322	1.586268	Kersen
685	348.6101	247.358	47891	932.2402	1.409334	Kersen
686	368.2825	190.6017	40566	925.1813	1.93221	Kersen
687	360.7991	203.4945	47094	933.3382	1.773017	Kersen
688	348.2068	185.2836	39661	893.6105	1.879319	Kersen
689	364.7904	222.6859	48440	970.4826	1.638139	Kersen
690	343.2856	208.4322	46127	933.215	1.646989	Kersen
691	355.0789	205.8009	35416	884.4274	1.725352	Kersen

692	363.2547	221.1425	44773	929.9217	1.642627	Kersen
693	358.1396	206.1674	39353	915.6037	1.73713	Kersen
694	377.3274	253.314	47657	965.1917	1.489564	Kersen
695	337.1202	153.2645	33429	834.2367	2.199597	Kersen
696	338.3371	213.6773	45680	937.6135	1.583402	Kersen
697	350.7506	192.151	43765	915.9392	1.825391	Kersen
698	360.0805	217.8164	42347	911.7228	1.653138	Kersen
699	363.0014	233.5851	49332	963.0413	1.554043	Kersen
700	274.2207	251.0319	46490	822.0786	1.092374	Karanda
701	246.1727	247.0506	39922	763.215	0.996446	Karanda
702	264.0303	261.2317	47383	813.3548	1.010713	Karanda
703	252.6697	252.0179	43761	800.8849	1.002587	Karanda
704	282.0443	258.124	48959	838.1689	1.09267	Karanda
705	256.0352	269.1189	48219	831.0468	0.951383	Karanda
706	261.6907	262.1908	46534	808.461	0.998093	Karanda
707	250.06	261.0939	45612	798.1241	0.95774	Karanda
708	237.3563	255.049	42899	772.1515	0.93063	Karanda
709	213.9743	261.3752	41109	755.1637	0.818648	Karanda
710	290.0569	248.002	46996	832.0143	1.169575	Karanda
711	274.3319	267.1872	48999	843.654	1.02674	Karanda
712	273.4685	270.0667	51067	844.9463	1.012596	Karanda
713	237.4131	262.0305	44871	783.4104	0.906051	Karanda
714	254.0236	258.0078	44275	796.3579	0.984558	Karanda
715	274.6434	259.0309	46223	841.4096	1.060273	Karanda
716	274.3319	255.0314	47497	823.5543	1.075679	Karanda
717	238.1029	252.3351	43413	777.8486	0.943598	Karanda
718	235.3593	253.4443	41321	766.7371	0.928643	Karanda
719	273.0897	270	49566	844.9574	1.011443	Karanda
720	225.8871	268.0019	40867	780.4479	0.842857	Karanda
721	251.5313	263.0076	45350	811.1756	0.956365	Karanda
722	249.2429	271.0664	47403	808.0098	0.91949	Karanda
723	257	249.0502	42889	792.4465	1.03192	Karanda
724	260.6185	257.0953	45529	808.4175	1.013704	Karanda
725	253.9547	251.0498	44165	790.1221	1.011571	Karanda
726	246.5867	263.0684	42615	789.6103	0.937348	Karanda
727	213.115	244.0328	34629	715.9568	0.873305	Karanda
728	255.6912	250.2878	43456	796.0966	1.021589	Karanda
729	261.2757	254.0315	44481	806.9779	1.028517	Karanda
730	211.1919	265.0075	40270	757.069	0.796928	Karanda
731	229.0349	270.15	44380	782.6723	0.847807	Karanda
732	227.0551	269.0297	43871	779.0703	0.843978	Karanda

733	272.5748	267.0075	49403	844.5693	1.020851	Karanda
734	254.0236	258.0174	44278	798.426	0.984521	Karanda
735	225.0555	258.0698	39833	754.2201	0.872073	Karanda
736	235.3593	253.4443	41340	766.7371	0.928643	Karanda
737	273.1172	266.0921	51366	840.6513	1.026401	Karanda
738	275.1818	263.0076	48240	830.3721	1.046288	Karanda
739	244.0922	256.2811	42341	775.139	0.952439	Karanda
740	254.0177	256.2362	45874	805.3774	0.991342	Karanda
741	253.9547	251.0498	44164	790.1221	1.011571	Karanda
742	265.3696	252.446	45343	805.037	1.051193	Karanda
743	274.0164	269.0911	51034	845.2562	1.018304	Karanda
744	254.0079	256.0957	45916	805.0003	0.991848	Karanda
745	226.3184	259.0019	39906	754.9032	0.87381	Karanda
746	246.9818	264.4844	46238	804.703	0.933824	Karanda
747	273.1172	266.0677	51393	841.218	1.026495	Karanda
748	274.0164	270.0463	51021	845.7009	1.014702	Karanda
749	230.3758	273.0568	43180	786.8873	0.843692	Karanda
750	495.6178	493.8117	21513	1066.051	1.003657	Lengkuas
751	468.6246	466.3443	15742	992.971	1.00489	Lengkuas
752	298.7189	300.2932	27626	732.9787	0.994758	Lengkuas
753	323.1548	324.7353	12492	688.9674	0.995133	Lengkuas
754	464.0043	462.65	18434	988.6113	1.002927	Lengkuas
755	456.514	454.8055	19623	973.8364	1.003757	Lengkuas
756	343.9026	340.8284	30657	810.9559	1.00902	Lengkuas
757	342.9869	343.4021	13399	730.1623	0.998791	Lengkuas
758	366.3796	342.1476	30107	850.0353	1.070823	Lengkuas
759	439.4963	438.0148	19861	944.8782	1.003382	Lengkuas
760	366.8079	363.2795	16084	790.9909	1.009712	Lengkuas
761	443.4433	442.1369	21136	950.3351	1.002955	Lengkuas
762	495.0253	490.6007	23185	1068.848	1.009019	Lengkuas
763	447.8326	448.8441	18259	957.6873	0.997746	Lengkuas
764	391.0307	384.194	24605	858.4493	1.017795	Lengkuas
765	481.0509	475.6007	19618	1026.962	1.01146	Lengkuas
766	449.325	448.6424	13494	957.3748	1.001522	Lengkuas
767	468.0865	465.3988	16197	988.7398	1.005775	Lengkuas
768	465.172	455.4086	20937	1003.26	1.021439	Lengkuas
769	457.4505	456.0274	30123	1006.287	1.003121	Lengkuas
770	457.9083	455.8114	27019	991.4902	1.0046	Lengkuas
771	475.6185	472.6034	20901	1020.78	1.00638	Lengkuas
772	451.2483	449.3217	19801	966.9991	1.004288	Lengkuas
773	435.4056	431.356	23789	942.395	1.009388	Lengkuas

774	473.846	470.4051	15437	1009.583	1.007315	Lengkuas
775	448.8574	448.3771	26027	990.224	1.001071	Lengkuas
776	468.8027	463.7036	26088	1010.357	1.010997	Lengkuas
777	486.6395	484.2004	21472	1045.488	1.005037	Lengkuas
778	454.3237	453.2163	19086	966.6139	1.002443	Lengkuas
779	375.1333	340.5877	29769	861.1457	1.101429	Lengkuas
780	430.0291	425.4174	16423	914.4674	1.01084	Lengkuas
781	320.4949	300.5878	25297	754.2344	1.066227	Lengkuas
782	424.0047	422.0427	17082	900.9055	1.004649	Lengkuas
783	466.2027	465.8948	16958	996.5397	1.000661	Lengkuas
784	474.8021	473.6243	15118	1005.796	1.002487	Lengkuas
785	378.7466	373.3751	16846	815.6483	1.014386	Lengkuas
786	331.658	323.1486	31287	800.2962	1.026333	Lengkuas
787	461.4	459.0534	21947	985.582	1.005112	Lengkuas
788	482.4459	478.1266	18126	1028.518	1.009034	Lengkuas
789	462.039	460.1087	21305	985.9763	1.004195	Lengkuas
790	375.6448	370.7991	21049	824.3671	1.013068	Lengkuas
791	388.8817	389.6216	14686	832.5235	0.998101	Lengkuas
792	490.8004	489.8418	19144	1044.53	1.001957	Lengkuas
793	340.6024	343.0015	30232	804.9938	0.993006	Lengkuas
794	443.4433	441.1916	17103	944.1603	1.005104	Lengkuas
795	495.6178	495.4049	14961	1055.634	1.00043	Lengkuas
796	364.56	363.0496	15262	790.078	1.00416	Lengkuas
797	392.45	390.3639	21324	839.014	1.005344	Lengkuas
798	302.0497	270.248	29688	746.0142	1.117676	Lengkuas
799	378.4019	377.0477	14980	808.2745	1.003591	Lengkuas

Lampiran II Hasil Estimasi Parameter Rata-rata

Nama Kelas	d1_panjang	d2_lebar	d3_luas	d4_keliling	d5_rasio
Cendana	333.4997	243.6425	51961.73	1050.941	1.376289
Jambu Biji	347.4635	255.96	65542.56	982.2391	1.363819
Mint	264.4988	237.3904	44485.69	868.8424	1.115068
Bayam Hijau	265.0538	238.6165	39093.64	796.0355	1.111913
Jeruk	297.2676	223.3617	42998.44	849.554	1.334187
Mangga	376.8026	163.2664	25548.6	851.7526	2.696511
Ara	264.3526	261.8606	36401.31	901.2527	1.011229
Kari	275.4872	202.6169	34611.98	781.0566	1.387771
Kelor	226.1014	211.7297	33910.29	685.9209	1.071188
Delima	355.288	200.3392	45221.02	1034.213	1.794805
Malapari	324.7583	262.3767	52537.02	931.3663	1.246302
Sirih	331.602	212.8924	45001.42	994.323	1.679311
Nangka	346.7076	252.3172	59361.78	1013.866	1.379473
Kersen	359.4591	209.2802	44565.47	934.5788	1.731119
Karanda	251.9409	259.956	45018.91	800.2058	0.969914
Lengkuas	420.241	415.7346	21186.98	915.6842	1.012741

Lampiran III Hasil Estimasi Parameter Varians

Nama Kelas	d1_panjang	d2_lebar	d3_luas	d4_keliling	d5_rasio
Cendana	340.2469	249.9426	18808940	401151.4	0.019461
Jambu Biji	607.6186	261.2007	27239771	1789.972	0.019462
Mint	788.4939	370.2954	44474911	5944.492	0.007946
Bayam Hijau	1036.295	152.9633	22251711	3314.906	0.017716
Jeruk	1749.633	685.309	1.01E+08	10742.3	0.020238
Mangga	159.711	5669.878	7684470	838.8994	0.832908
Ara	246.8096	130.3914	10209318	25121.34	0.005181
Kari	4920.609	1707.704	1.94E+08	28570.99	0.129779
Kelor	719.3639	672.368	53415672	6205.158	0.00542
Delima	263.621	533.4942	20676915	161342.3	0.048061
Malapari	767.8353	233.49	18991292	1584.04	0.030785
Sirih	3000.029	2194.43	40886288	112793.9	0.345519
Nangka	276.6004	245.3223	30016108	49672.55	0.012064
Kersen	137.3396	451.227	19567965	1276.954	0.026322
Karanda	375.2824	55.30716	12621948	931.421	0.006167
Lengkuas	3322.809	3682.755	28128368	10865.28	0.0006

Lampiran IV Hasil Uji Normalitas

Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
panjang	.164	50	.002	.913	50	.001
lebar	.085	50	.200*	.974	50	.332
keliling	.086	50	.200*	.981	50	.588
rasio	.072	50	.200*	.981	50	.602
luas	.084	50	.200*	.983	50	.700

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction