BAB III
METODE PENELITIAN

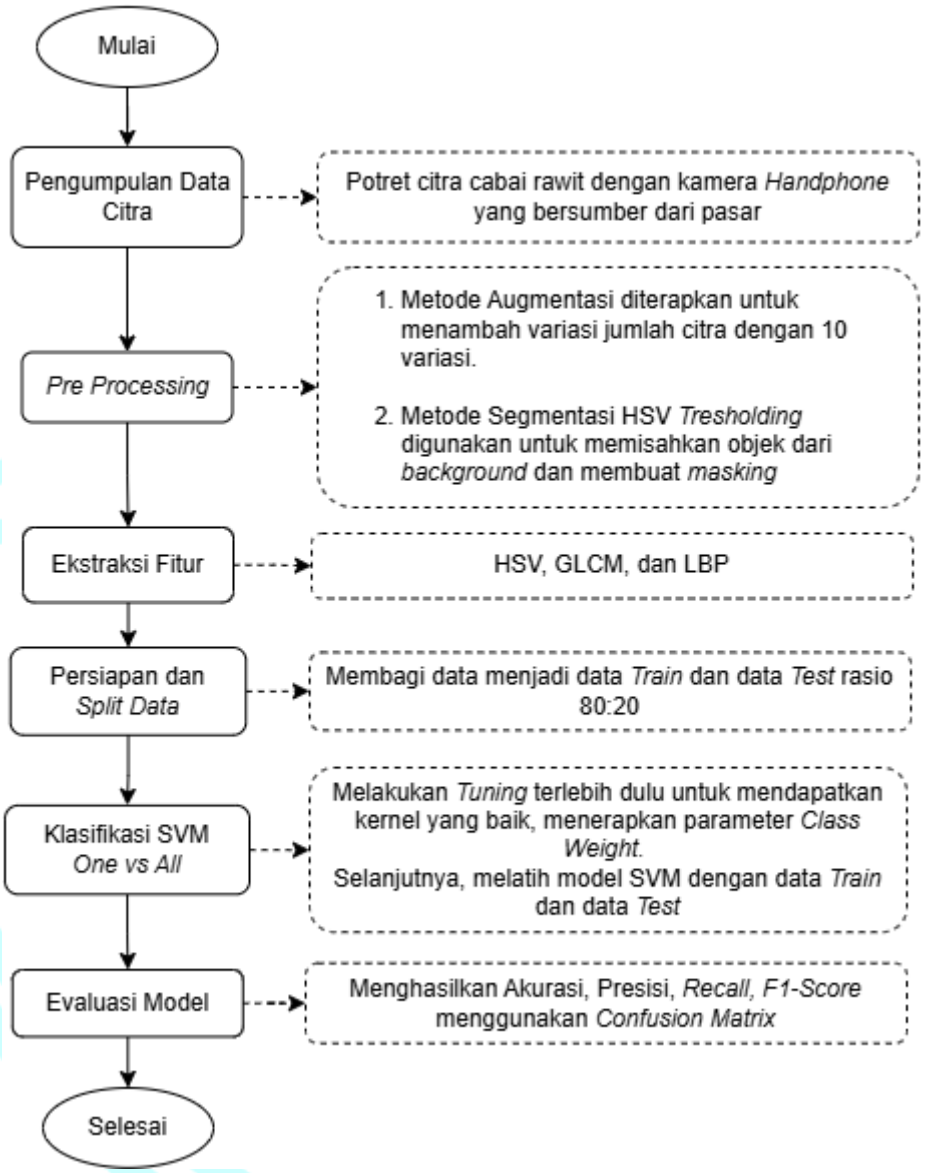
3.1 Objek Penelitian

Pada penelitian ini penulis berfokus pada pengimplementasian algoritma *Support Vector Machine* (SVM) dengan ekstraksi fitur yang diharapkan dapat meningkatkan efektivitas pada sistem klasifikasi kondisi kematangan cabai rawit.

Dataset yang digunakan dalam penelitian ini berupa citra cabai rawit tanpa tangkai, hal ini dikarenakan agar sistem dapat berfokus pada warna buah cabai rawit saja dan mengurangi *noise* yang disebabkan oleh tangkai pada ekstraksi fitur tekstur (Fitri et al., 2020). Menggunakan lebih dari 10 buah cabai rawit setiap kelasnya yang bersumber dari pasar, dan berhasil menghasilkan total 523 citra.

3.2 Prosedur Penelitian

Untuk menghasilkan *ouput* yang diharapkan maka diperlukan alur pengerjaan sehingga dapat menjawab tujuan dari rumusan masalah yang telah disampaikan sebelumnya. Berikut Alur pengerjaannya :

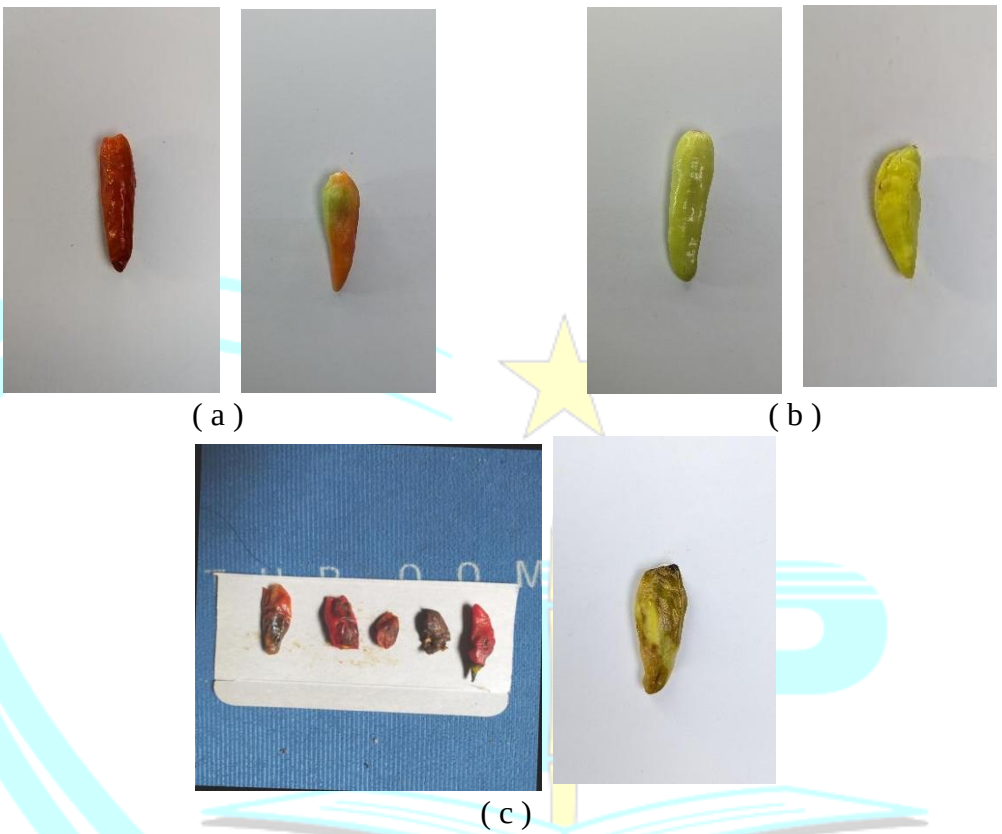


Gambar 3. 1 Tahapan Prosedur Penelitian

3.3 Pengumpulan Data

Dalam penelitian ini, dataset citra cabai rawit diperoleh dengan menggunakan kamera Google Pixel 6 *Pro*, di mana pengambilan citra dilakukan pada jarak sekitar 15 cm dari objek untuk memastikan detail visual yang optimal. Cabai rawit yang digunakan berasal dari pasar tradisional, sehingga mencerminkan kondisi nyata seperti yang biasa ditemui oleh konsumen dalam kehidupan sehari-hari.

Total terdapat 523 citra yang berhasil dikumpulkan, yang terdiri dari tiga kategori utama berdasarkan kondisi kematangan cabai, yaitu: 278 citra cabai rawit matang, 161 citra cabai rawit mentah, dan 84 citra cabai rawit rusak.



(a) (b)

(c)

Gambar 3. 2 Contoh citra dari tiap label kematangan cabai rawit
(a) Cabai Matang, (b) Cabai Mentah, (c) Cabai Rusak

3.4 *PreProcessing* Data

Pada tahap awal, dataset citra akan diunggah ke *Google Drive*. Setelah berhasil dimuat, metode augmentasi akan diterapkan terlebih dahulu, dilanjutkan dengan metode segmentasi.

3.4.1 Augmentasi Data

Metode augmentasi ini diperlukan untuk menambah variasi pada data *train* tanpa mengubah infromasi mendasar yang bertujuan untuk mencegah terjadinya *overfitting* dan meningkatkan generalisasi model serta membantu model mengenali pola yang lebih *robust.*

Tabel 3. 1 Augmentasi Data

Algoritma	: Augmentasi Data
Input	: Dataset Citra Cabai Rawit
Output	: Data hasil augmentasi

1. Membaca Dataset berisi 3 label (Matang, Mentah, dan Rusak)
2. Memutar Citra dari sudut 30º hingga 90º untuk variasi perspektif (*Rotate*)
3. Membalik citra secara horizontal/vertikal (*Flip*)
4. Menambah pencahayaan dan kontras
5. Menambahk *Motion Blur*
6. Menambahkan gangguan digital (*Noise Gaussian*)
7. Menghilangkan sebagian kecil citra
8. Hasil augmentasi

3.4.2 Segmentasi Data

Metode segmentasi di tambahkan guna memisahkan objek cabai dengan *background*nya, ini dapat memberikan bantuan kepada model agar dapat lebih terfokus dalam mengekstraksi objek dan dapat memaksimalkan proses pemodelan.

Tabel 3. 2 Segmentasi Data

Algoritma	: Segmentasi Data
Input	: Data Hasil Augmentasi
Output	: Data hasil Segmentasi dan Masking

1. Input Hasil Augmentasi
2. Menentukan warna dengan nilai HSV yang di tentukan oleh fungsi *Lower* dan *Upper*
3. Mengganti *background* dengan warna hitam yang selanjutnya akan di jadikan *masking*
4. Hasil Segmentasi dan *Masking*

3.5 Ekstraksi Fitur

Tahapan ini cukup krusial karena dapat mempengaruhi hasil prediksi berdasarkan warna dan tekstur dari objek citra cabai itu sendiri. Hasil nilai yang di peroleh dari ekstraksi ini akan menjadi *input* bagi *training* model SVM.

3.5.1 Fitur Ruang Warna HSV

Umumnya RGB adalah komponen warna primer yang terdiri dari *Red*(Merah), *Green*(Hijau), dan *Blue*(Biru) menjadikan RGB memiliki

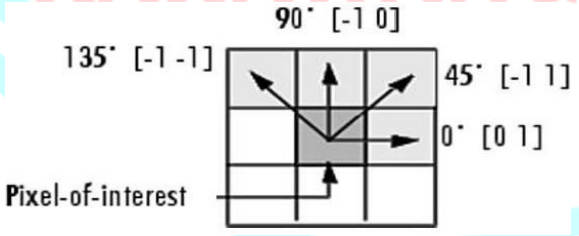
kelebihan sebagai ruang warna yang luas, namun hal ini juga menjadi kelemahan dalam segmentasi. Oleh karena itu diperlukan konversi ke HSV sehingga mampu menghasilkan nilai rata-rata *hue*, *saturation*, dan *value* sebagai ekstraksi fitur warna (Fitri et al., 2020). Adapun rumus konversi RGB terhadap HSV:

Tabel 3. 3 HSV

Algoritma	: HSV
Input	: Citra RGB Hasil Segmentasi
Output	: 3 fitur warna *mean* H, S, dan V
1.	Membaca Dataset Menentukan *Min Max*. R = $\frac{R}{255}$, G = $\frac{G}{255}$, B = $\frac{B}{255}$
2.	Menghitung V dengan mencari nilai *Max* dari RGB
3.	Menghitung Vm = V – *min*(r,g,b)
4.	Menghitung S = $\frac{v-(r,g,b)}{v}$
5.	Menghitung nilai H berdasarkan nilai R, G, dan B
6.	Memisahkan *channel* H, S, dan V
7.	Hitung rata – rata setiap *channel.*

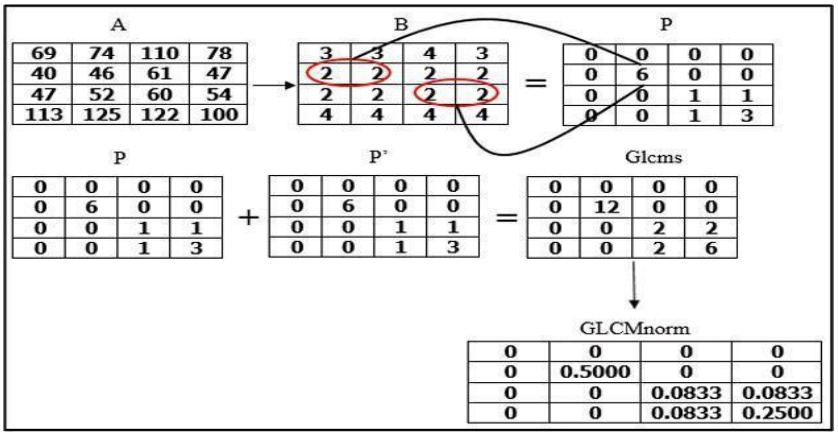
3.5.2 Fitur Tekstur GLCM

Selanjutnya, dilakukan ekstraksi fitur tekstur dengan matriks GLCM. Matriks pada GLCM dapat merepresentasikan hubungan suatu derajat keabuan antara dua piksel dengan intensitas tertentu dalam jarak di simbolkan □ dan arah sudut disimbolkan □, sudut yang biasa digunakan dalam GLCM untuk menghitung *co-occurance* terdapat pada sudut 0º, 45º, 90º, dan 135º.



Gambar 3. 3 Sudut GLCM

Setelah Melakukan penentuan awal matriks berdasarkan dua pasang piksel, dilakukan *tranpose* dengan membalikan nilai dari x (baris) menjadi y (kolom) begitu pula sebaliknya. Hasil *transpose* dinormalisasikan dengan membagi nilai dengan jumlah keseluruhan nilai.



Gambar 3. 4 *Tranpose* dan normalisasi GLCM

Tabel 3. 4 Ekstraksi Tekstur GLCM

Algoritma	: GLCM
Input	: Citra *Greyscale* Hasil Segmentasi
Output	: 6 fitur tekstur

1. Membaca citra yang sudah di konversi ke *Grayscale*
2. Menghitung jarak 1, 3, 5 piksel
3. Menghitung Sudut arah 0º, 45º, 90º, dan 135º
4. Mengekstraksi 5 properti statistik GLCM
5. Menghitung nilai *entropy* GLCM
6. Hasil 6 Fitur *(contrast, energy, homogeneity, correlation, dissimilarity, entropy).*

3.5.3 Fitur Tekstur LBP

Untuk merepresentasikan pola lokal pada citra *grayscale*, *Local Binary Pattern* (LBP) adalah metode ekstraksi fitur tekstur yang sederhana namun efektif. Prosesnya dimulai dengan menerapkan operator LBP pada setiap piksel citra dengan menggunakan radius 1 dan 8 titik tetangga di sekelilingnya. Operator ini melihat intensitas piksel pusat dan piksel tetangganya. Jika intensitas piksel tetangga lebih tinggi atau sebanding dengan piksel pusat, maka nilainya diberikan nilai 1, sedangkan jika intensitasnya lebih rendah, maka nilainya diberikan nilai 0. Hasil perbandingan menghasilkan pola biner berukuran delapan bit yang kemudian dikonversi menjadi nilai desimal untuk menunjukkan fitur lokal (IF ITS, 2020). Nilai fitur harus dinormalisasi agar proporsional dan dapat dibandingkan antar citra. Vektor fitur adalah hasil akhir dari proses ini.

Tabel 3. 5 Ekstraksi Tekstur LBP

Algoritma	: LBP
Input	: Citra *Greyscale* Hasil Segmentasi
Output	: 10 fitur pola lokal
1. Membaca citra yang sudah di konversi ke *Grayscale*	
2. Menerapkan operator LBP dengan radius 1 dengan 8 titik tetangga	
3. Menghitung histogram pola biner	
4. Normalisasi Histogram	

Setelah berhasil mendefinisikan dan mendapatkan nilai fiturnya, langkah selanjutnya adalah mengintegrasikan seluruh fitur seperti berikut:

Tabel 3. 6 Kombinasi Fitur

Algoritma	: Kombinasi Fitur
Input	: *Output* HSV + GLCM + LBP
Output	: Vektor fitur kombinasi (20 Fitur)
1. Menggabungkan semua fitur : ● 3 fitur HSV ● 6 fitur GLCM ● 10 fitur LBP ● 1 fitur *entropy greyscale*	
2. Normalisasikan dengan *StandardScaler*	

3.6 Implementasi Klasifikasi Algoritma SVM

Untuk klasifikasi, algoritma SVM *One-vs-All* digunakan setelah fitur diekstraksi. Sangat penting untuk diingat bahwa sebelum melatih model, dataset citra cabai rawit dibagi menjadi dua bagian: 80% data *train* dan 20% data *test*. Ini dilakukan di awal proses klasifikasi agar model dapat disesuaikan (mencari kernel SVM terbaik) berdasarkan struktur dataset menggunakan data *train* sebelum pelatihan sebenarnya dimulai.

Selanjutnya, pelatihan model dilakukan dengan data *train* yang sudah berisi fitur tekstur. Parameter *Class Weight* digunakan untuk mengatasi masalah ketidakseimbangan data, seperti ketika jumlah citra cabai mentah, matang, dan rusak tidak seimbang. Ini menempatkan kelas minoritas di atas yang memiliki sampel lebih sedikit. Penyesuaian bobot ini dilakukan untuk membuat model lebih peka dan sensitif terhadap kelas minoritas sehingga mereka dapat menggunakan data *train* secara maksimal. Selain itu, penerapan

Class Weight membantu menurunkan bias model terhadap kelas mayoritas dan secara signifikan meningkatkan kinerja model, terutama dalam hal metrik seperti *recall* dan *F1 Score*, yang penting untuk mengukur kemampuan model untuk mengidentifikasi kelas minoritas dengan benar.

Tabel 3. 7 Klasifikasi Algoritma SVM

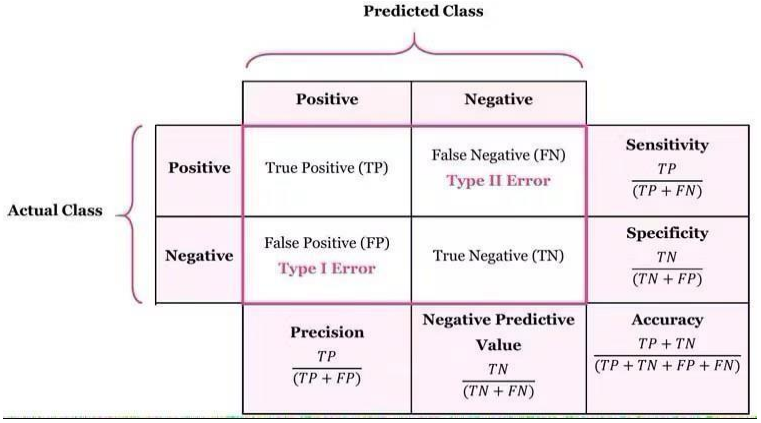
Algoritma	: Klasifikasi SVM
Input	: Data *Train*
Output	: Model Prediksi dan Hasil Prediksi Klasifikasi
1. Membaca data *train*	
2. Melakukan *tuning* pada kernel *linear*, RBF, *Polynomial*, Sigmoid	
3. Mencari *hyperplane* terbaik untuk memisahkan data menggunakan kernel hasil *Tuning*	
4. Penerapan parameter *Class Weight* untuk menyeimbangkan bobot label	
5. Menggunakan metode SVM *One-vs-All*	
6. Hasil prediksi klasifikasi	

3.7 Evaluasi Model

Pada tahap terakhir ini, pengujian dilakukan dengan menggunakan model yang telah dilatih sebelumnya. Setelah itu, proses evaluasi dilakukan menggunakan metode *Confusion Matrix* dan ROC AUC untuk mengevaluasi seberapa baik kinerja model.

3.7.1 *Confusion Matrix*

Pada tahap ini, metode *Confusion Matrix* digunakan untuk mengetahui nilai dari akurasi, presisi, *recall*, dan F1-*Score*. Evaluasi ini sangat penting karena hasil prediksi akan menunjukkan kinerja model yang telah dilatih sehingga mereka dapat menjawab rumusan masalah sebelumnya.



Gambar 3. 5 *Confusion Matrix*

- Akurasi : proporsi prediksi yang benar dari semua prediksi

$$\text{Akurasi} = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (3.1)$$

- Presisi : Proporsi prediksi positif yang benar dari semua prediksi positif.

$$\text{Presisi} = \frac{TP}{(TP+FP)} \quad (3.2)$$

- *Recall* : Proporsi sampel positif yang benar-benar diklasifikasikan sebagai positif.

$$Recall = \frac{TP}{(TP+FN)} \quad (3.3)$$

- *F1-Score* : Nilai harmonik rata-rata antara *precision* dan *recall*.

$$F1 - Score = \frac{2\,(Precision + Recall)}{(Precisiom + Recall)} \quad (3.4)$$

3.7.2 ROC AUC

Evaluasi kinerja model dalam penelitian ini juga dilengkapi dengan pengukuran ROC AUC (*Receiver Operating Characteristic - Area Under Curve*) sebagai metode tambahan untuk menilai performa klasifikasi *multiclass*. ROC AUC digunakan untuk mengevaluasi kemampuan model dalam membedakan antar kelas secara menyeluruh, terutama pada kondisi distribusi kelas yang tidak seimbang. ROC *curve* menggambarkan hubungan antara *True Positive Rate* (TPR) dan *False Positive Rate* (FPR), yang dihitung menggunakan rumus:

- *True Positive Rate (Recall)* : $TPR = \frac{TP}{TP+FN}$ (3.5)
- *False Positive Rate* : $FPR = \frac{FP}{FP+TN}$ (3.6)

Sedangkan AUC (*Area Under Curve*) dihitung dari luas area di bawah kurva ROC menggunakan pendekatan integral numerik. Semakin besar nilai AUC, maka semakin baik kemampuan model dalam membedakan antara kelas positif dan negatif.

Dalam kasus klasifikasi *multiclas* seperti pada penelitian ini, pendekatan *One-vs-Rest* (OvR) digunakan, yaitu dengan membandingkan satu kelas terhadap gabungan dari kelas lainnya untuk memperoleh nilai ROC AUC per kelas. Hasil pengukuran AUC selanjutnya dapat dirata-ratakan menggunakan metode *macro-average* guna memperoleh gambaran kinerja model secara keseluruhan terhadap semua kelas yang ada.