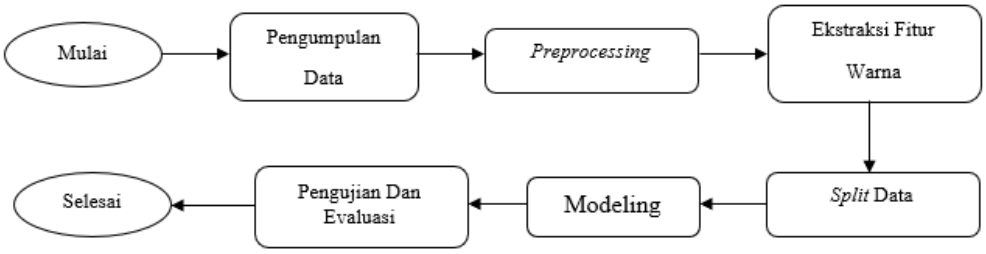
BAB III
METODE PENELITIAN

3.1 Objek Penelitian

Objek dalam penelitian ini merupakan gambar buah pepaya dengan tingkat kematangan mentah, setengah matang, dan matang. Pada tingkat kematangan buah pepaya mentah berwarna hijau. Sedangkan, pepaya yang memiliki tingkat kematangan setengah matang berwana hijau yang memudar dengan kuning. Sementara itu, tingkat kematangan buah pepaya yang sudah matang berwarna oranye.

3.2 Prosedur Penelitian

Prosedur penelitian ini dirancang untuk memberikan langkah-langkah sistematis dalam mengumpulkan dan menganalisis data, guna mencapai tujuan penelitian yang telah ditetapkan. Berikut *Flowchart* prosedur penelitian pada Gambar 3.1.



Gambar 3. 1 Prosedur penelitian

3.2.1 Pengumpulan Data

Prosedur pengumpulan data dalam penelitian ini dimulai dengan mencari dan memilih dataset yang sesuai melalui *platform Kaggle*, adalah sebuah komunitas yang menyediakan berbagai dataset untuk keperluan penelitian dan pengembangan model berbasis data. Dataset yang dipilih berisi 300 gambar buah pepaya 100 gambar kematangan mentah, 100 gambar setengah matang, dan 100 gambar matang. Lalu melakukan *review* jurnal dan buku yang berkaitan dengan klasifikasi tingkat kematangan buah menggunakan algoritma *Logistic Regression* dan *KNN*. Berikut gambar3.2 buah pepaya dari 3 (tiga) tingkat kematangan

Gambar 3. 2 Buah pepaya dengan 3 tingkat kematangan, mentah, mengkal dan matang

3.2.2 *Preprocessing*

Data yang telah dikumpulkan akan menjalani tahap *pre-processing* yang mencakup beberapa langkah penting untuk memastikan kualitas data yang digunakan dalam pelatihan model. Proses *pre-processing* ini bertujuan untuk menyiapkan data agar model dapat memprosesnya dengan efektif. Berikut adalah langkah-langkah yang akan dilakukan :

1. *Resize*

   Gambar-gambar dalam dataset akan diubah ukurannya menjadi seragam misalnya 240 x 240 piksel. Lalu, gambar ditampilkan gambar sebelum di resize dan setelah di *resize*. Berikut merupakan contoh pseudocode untuk Resize (mengubah ukuran gambar).

   Resize

   Mulai

   #Langkah 1 : Ambil dataset yang akan dirubah pikselnya Baca dataset yang berisi gambar dari file atau sumber lainnya

   #Langkah 2: Merubah ukuran semua gambar menjadi 240 x 240 piksel

   #Langkah 3: Menampilkan hasil gambar *resize*

   Menampilkan gambar sebelum dilakukan *resize* dan sesudah dilakukan *resize*

   Selesai

   Hasil :

Gambar 3. 3 *Resize*

2. *Augmentasi*

Teknik *augmentasi* data seperti *Rotated, Flipping, Zooming, Cropping,* dan *Brightness/Contrast Adjustment* diterapkan. Augmentasi ini bertujuan untuk meningkatkan jumlah variasi gambar yang digunakan dalam pelatihan sehingga model dapat belajar mengenali fitur-fitur daun mangga yang terkena hama dengan lebih baik.

1. *Rotated*

Rotated atau rotasi teknik ini akan digunakan untuk memutar citra pada sudut tertentu, misal gambar akan di *rotasi* sebesar 30°. Lalu, ditampilkan menjadi 2 yaitu gambar sebelum di rotasi dan gambar sesudah di *rotasi.* Fungsinya yaitu untuk meningkatkan keragaman data latih. Berikut merupakan contoh pseudocode untuk *Rotated (Rotasi*/Memutar gambar).

Rotated

Mulai

#Langkah 1 : Ambil dataset yang akan dirubah rotasinya 31 Baca dataset yang berisi gambar dari file atau sumber lainnya.

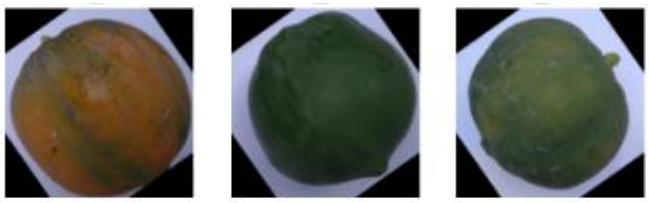
#Langkah 2: Merubah ukuran semua gambar menjadi 30°.

#Langkah 3: Menampilkan hasil gambar rotated.

Menampilkan gambar sebelum dilakukan rotated dan sesudah dilakukan rotated.

Selesai

Hasil :



Gambar 3. 4 Rotasi

2. *Flipping*

Gambar ini akan dilakukan dengan teknik *augmentasi* citra dengan cara membalik gambar secara *horizontal* atau *vertikal*. Untuk garis *horizontal* akan dilakukan dititik tangen 185° dan untuk garis kemiringan *vertikal* 95°. Setelah dilakukan keduanya. Kemudian, gambar asli, gambar *horizontal* dan gambar *vertikal* akan ditampilkan. Berikut merupakan contoh *pseudocode* untuk *flipping* (membalikkan gambar).

Flipping

Mulai

#Langkah 1 : Ambil dataset yang akan dirubah Baca dataset yang berisi gambar dari file atau sumber lainnya

#Langkah 2: Merubah ukuran horizontal semua gambar dirubah menjadi 185°

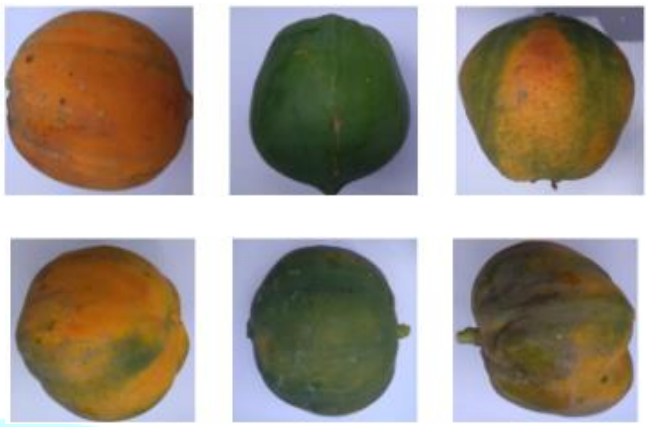
#Langkah 3: Merubah ukuran vertikal semua gambar dirubah menjadi 95°

#Langkah 4: Menampilkan hasil gambar flipping

Menampilkan beberapa gambar sebelum dilakukan *flipping* dan sesudah dilakukan *flipping*

Selesai

Hasil :



Gambar 3. 5 flipping horizontal dan *vertikal*

3. *Zooming*

Gambar yang sudah melalui tahap *flipping* akan dilanjutkan dengan tahap zooming yaitu dengan teknik augmentasi citra untuk memperbesar gambar, gambar akan dilakukan dengan diperbesar 1.5x. Setelah itu gambar sebelum dan sesudah zooming ditampilkan. Berikut merupakan contoh *pseudocode* untuk *zooming* (memperbesar gambar).

Proses mengubah citra dari satu ruang warna ke ruang warna *RGB (Red, Green, Blue).*

Zooming

Mulai

#Langkah 1 : Ambil dataset yang akan dirubah Baca dataset yang berisi gambar dari file atau sumber lainnya

#Langkah 2: Memperbesar gambar dengan 1.5x Semua gambar akan di perbesar sebanyak 1.5x

#Langkah 3: Menampilkan hasil gambar *zooming*

Menampilkan gambar sebelum dilakukan *zooming* dan sesudah dilakukan *zooming*

Selesai

Hasil :

Gambar 3. 6 *zooming*

4. *Crooping*

Cropping dapat menghasilkan variasi baru dari citra yang sama, yang bisa membantu model dalam belajar lebih baik dari berbagai perspektif. Berdasarkan koordinat dan ukuran, gambar akan di potong pada titik koordinat (50, 50) dengan ukuran 150x150 piksel. Lalu, gambar akan ditampilkan yakni gambar sebelum dan sesudah di cropping. Berikut merupakan contoh pseudocode untuk *cropping* (memotong gambar).

Crooping

Mulai

#Langkah 1 : Ambil dataset yang akan dirubah Baca dataset yang berisi gambar dari file atau sumber lainnya

#Langkah 2: Memotong gambar dititik koordinat 50, 50 gambar berukuran 150x150 piksel Semua gambar akan di potong pada titik koordinat 50, 50 dengan ukuran 150x150 piksel

#Langkah 3: Menampilkan hasil gambar *cropping* Menampilkan beberapa gambar sebelum dilakukan *cropping* dan sesudah dilakukan cropping

Selesai

Hasil :

Gambar 3. 7 *crooping*

5. *Brightness/Contrast Adjustment*

Brightness adalah Teknik *augmemntasi* citra untuk mengatur tingkat kecerahan keseluruhan gambar. Seperti dengan menambah nilai 2.0 pada kecerahan dan mengalikan kontras dengan faktor 2.0 untuk meningkatkan perbedaan antara nilai pixel dalam gambar. Berikut merupakan contoh pseudocode untuk *brihtness/contrast adjustment* (mengatur kecerahan/kontras)

Brightness/Contrast Adjustment

Mulai

#Langkah 1 : Ambil dataset yang akan dirubah Baca dataset yang berisi gambar dari file atau sumber lainnya

#Langkah 2: Mengatur kecerahan gambar dengan nilai 2.0 x 2.0 untuk nilai kontras Semua gambar akan di atur kecerahannya dengan nilai kecerahan 2.0 x 2.0 untuk nilai kontras

#Langkah 3: Menampilkan hasil gambar *brightness/contrast adjustment*

Selesai

Hasil :

Gambar 3. 8 *brightness adjusment*

6.Segmentasi *Otsu*

Segmentasi *Otsu* adalah metode *thresholding* otomatis yang digunakan untuk memisahkan objek dari latar belakang dalam citra *grayscale*. *Otsu* memilih nilai *threshold* optimal dengan cara meminimalkan varians intra-kelas (dalam objek dan *background*).

Segmentasi *Otsu*

Mulai

Langkah 1 : Ambil dataset yang akan dirubah

Baca dataset yang berisi gambar dari file atau sumber lainnya

Langkah 2: Mengatur kecerahan gambar dengan nilai 2.0

Set brightness = 2.0

Set contrast = 2.0

Langkah 3: Konversi gambar ke *grayscale*

Ubah gambar berwarna *(RGB/BGR)* menjadi gambar *grayscale*

Langkah 4: Lakukan *Gaussian Blur* untuk mengurangi *noise* (opsional)

Terapkan *Gaussian Blur* ke gambar *grayscale* (ukuran kernel = 5x5)

Langkah 5: Segmentasi menggunakan metode *Otsu*

Hitung nilai *threshold* optimal secara otomatis menggunakan metode *Otsu*

Terapkan *threshold* ke gambar untuk menghasilkan gambar biner

Langkah 6: Tampilkan hasil segmentasi *Otsu*

Tampilkan gambar hasil segmentasi (gambar biner)

Selesai

Hasil :

Gambar 3. 9 Segmentasi *otsu*

3.2.3 Ekstraksi Fitur Warna

Pada tahap ini gambar akan ditentukan arahnya *horizontal* dan *vertikal* jaraknya 1 piksel dan arahnya adalah 90° dengan level ke abu-abuannya 250. Berikut merupakan contoh *pseudocode* untuk tahapan *GLCM.*

Gray Level Co-Occurrence Matrix

Mulai

#Langkah 1 : Ambil dataset yang akan dirubah baca dataset yang berisi gambar dari file atau sumber lainnya.

#Langkah 2 : Merubah ukuran *horizontal* dan *vertikal* semua gambar dirubah menjadi 90° dengan jarak 1 piksel.

#Langkah 3 : Merubah warna *horizontal* dan *vertikal* dengan nilai level ke abu-abu an 250 dengan jarak 1 piksel untuk semua gambar.

#Langkah 4 : Menampilkan hasil gambar dari ekstrasi fitur *GLCM.*

Selesai

Hasil :

Tabel 3. 1 Hasil Ekstraksi Fitur *GLCM*

Tingkat Kematangan	*Contrast*	*Dissim-ilarity*	*Homo - geneity*	*Energy*	*Correlation*	*ASM*
Pepaya Matang	30,44%	2,26%	0,52%	0,04%	0,99%	0,00%
Pepaya Mentah	32,17%	1,96%	0,59%	0,05%	0,99%	0,00%
Pepaya Mengkal	38,44%	2,24%	0,53%	0,04%	0,99%	0,00%

3.2.4 *Split* Data

Tahap pembagian data dalam penelitian ini dilakukan dengan membagi dataset menjadi dua bagian utama, yaitu 80% untuk data latih (*training* data) dan 20% untuk data uji (*testing* data). Pembagian ini bertujuan untuk memastikan bahwa model dapat dilatih menggunakan sebagian data (data latih) dan diuji kinerjanya pada data yang tidak digunakan selama pelatihan (data uji). Proses pembagian dilakukan secara sistematis untuk menjaga pembagian data yang seimbang di setiap kelas dalam dataset, dataset akan bertambah setelah melewati tahap augmentasi yaitu menjadi 1.200.

Mulai

#Langkah 1 : siapkan dataset yang akan di bagi menjadi *training* dan *testing.*

#Langkah 2 : buat dua folder untuk menyimpan data *training* dan data *testing.*

#Langkah 3 : masukan masing-masing dataset yang telah di siapkan ke folder *training* dan *testing.*

Selesai

Hasil :

Tabel 3. 2 *Split* Data

Persentase data		Jumlah	
Training	*Testing*	*Training*	*Testing*
80%	20%	960	240
Total		1.200	

3.2.5 Modeling

Tahap pemodelan merupakan fase penting setelah *preprocessing* data, di mana dilakukan pembentukan model klasifikasi dengan tujuan untuk mengenali pola dari dataset yang telah disiapkan. Pada penelitian ini, proses pemodelan dilakukan dengan menerapkan dua algoritma klasifikasi, yaitu *Logistic Regression* dan *K-Nearest Neighbor (K-NN).* Kedua algoritma ini digunakan untuk membandingkan performa dalam mengklasifikasikan tingkat kematangan buah pepaya berdasarkan fitur citra yang telah diekstraksi dan dipraproses sebelumnya.

Mulai

Langkah 1: *Load dataset*

Baca dataset hasil praproses (fitur dan label)

Langkah 2: *Split dataset*

Pisahkan dataset menjadi data latih dan data uji

Persentase contoh: 80% data latih, 20% data uji

Langkah 3: Inisialisasi model

Inisialisasi algoritma:

1. Model_1 = *Logistic Regression*

2. Model_2 = *K-Nearest Neighbor* (K = nilai tertentu)

Selesai

```
from sklearn.linear_model import LogisticRegression

# Inisialisasi model Logistic Regression
logreg_model = LogisticRegression(random_state=42, max_iter=1000, C=1.0, solver='lbfgs')

# Melatih model dengan data pelatihan
# Gunakan X_train_flat yang sudah di-flatten
logreg_model.fit(X_train_flat, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000, random_state=42)
```

Gambar 3. 10 *Modeling logistic Regression*

```
[ ] from sklearn.neighbors import KNeighborsClassifier

    # Implementasi model knn
    # Menggunakan nilai K terbaik dari data uji
    knn_model = KNeighborsClassifier(n_neighbors=optimal_k_test) # Mengganti optimal_k_accuracy dengan optimal_k_test
    knn_model.fit(X_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=np.int64(1))
```

Gambar 3. 11 Modeling *K-Nearest Neighbor*

3.2.6 Pengujian Dan Evaluasi

Setelah melalui berbagai tahapan, tahap terakhir adalah pengujian dan evaluasi. Proses ini mencakup analisis hasil klasifikasi untuk menilai apakah model telah mencapai tingkat akurasi dan kinerja yang diharapkan. Klasifikasi dilakukan dengan menggunakan algoritma *Logistic Regression* dan *KNN*. Sebelum melakukan pengujian dengan *K-Nearest Neighbor* terlebih dahulu mencari nilai K terbaik.

Mulai

Langkah 1 : Siapkan data latih dan data uji

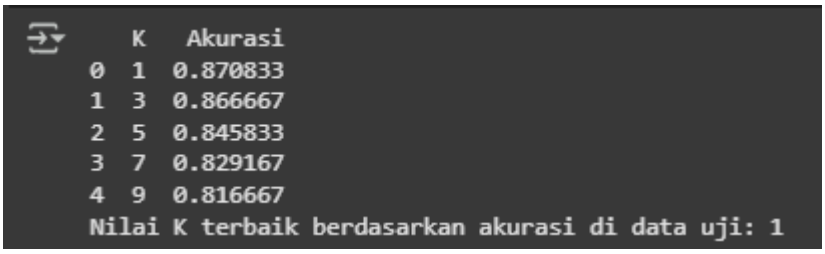
Langkah 2 : Inisialisasi daftar nilai k (misalnya dari 1 sampai 9)

Langkah 3: Buat variabel untuk menyimpan akurasi tertinggi dan nilai k terbaik

Langkah 4 : Tampilkan nilai k terbaik dan akurasi tertingginya

Selesai

Hasil :



	K	Akurasi
0	1	0.870833
1	3	0.866667
2	5	0.845833
3	7	0.829167
4	9	0.816667

Nilai K terbaik berdasarkan akurasi di data uji: 1

Gambar 3. 12 Hasil mencari nilai K terbesar

Logistic Regression

Mulai

Langkah 1 : Persiapkan data untuk *Logistic Regression*

1. Pisahkan dataset menjadi data latih dan data uji dengan rasio 80:20.
2. Normalisasi fitur agar memiliki rentang nilai yang seragam.

Langkah 2 : Latih model *Logistic Regression*

1. Gunakan *Logistic Regression* dengan regularisasi (*L1/L2*, misalnya *L2 Ridge*).
2. Tentukan parameter regularisasi (misalnya C).
3. Latih model menggunakan data latih dan label kematangan buah.

Langkah 3 : Evaluasi model

1. Uji model dengan data uji dan catat hasil prediksi.
2. Hitung metrik evaluasi seperti *akurasi, precision, recall*, dan *F1-score.*

Langkah 4 : Prediksi kematangan buah baru

1. Ekstrak dan normalisasi fitur dari gambar baru.
2. Masukkan fitur ke dalam model *Logistic Regression*.
3. Model akan memprediksi kategori: "Matang", "Mengkal", atau "Mentah".
4. Tampilkan hasil prediksi.

Langkah 5 : Simpan model jika diperlukan

1. Simpan model *Logistic Regression* yang telah dilatih agar dapat digunakan kembali tanpa perlu pelatihan ulang.

Selesai

Hasil pengujian :

Tabel 3. 3 Hasil pengujian algoritma *Logistic Regression* dan *K-Nearest Neighbor*

Model	Tingkat Kematangan	*Accuracy*	*Precision*	*Recall*	*F1-Score*
Logistic Regression	Pepaya matang	73%	69%	85%	76%
Logistic Regression	Pepaya mengkal	73%	74%	79%	76%
Logistic Regression	Pepaya mentah	73%	79%	56%	66%
K-Nearest Neighbor	Pepaya matang	87%	86%	89%	87%
K-Nearest Neighbor	Pepaya mengkal	87%	89%	89%	89%
K-Nearest Neighbor	Pepaya mentah	87%	87%	84%	85%

K-Nearest Neighbor

Mulai

Langkah 1 : Persiapkan data untuk *KNN*

1. Pisahkan dataset menjadi data latih dan data uji dengan rasio 80:20.
2.Normalisasi fitur agar memiliki rentang nilai yang seragam (menggunakan *Standard Scaler* atau *MinMaxScaler*).

Langkah 2 : Latih model *KNN*

1. Pilih jumlah tetangga (k) yang sesuai (misalnya k=1).
2.Tentukan metric jarak yang sesuai (misalnya *Euclidean, Manhattan*, atau *Minkowski*).
3. Latih model *KNN* menggunakan data latih dan label kematangan buah.

Langkah 3 : Evaluasi model

1. Uji model dengan data uji dan catat hasil prediksi.
2. Hitung metrik evaluasi seperti *akurasi, precision, recall,* dan *F1-score.*
3. Hitung confusion matrix untuk mengetahui performa klasifikasi model.

Langkah 4 : Prediksi kematangan buah baru

1. Masukkan fitur gambar baru ke dalam model *KNN.*
2.Model akan memprediksi apakah buah tersebut "Matang", "Mengkal", atau "Mentah".
3. Tampilkan hasil prediksi.

Langkah 5 : Simpan model jika diperlukan

1. Simpan model *KNN* yang telah dilatih agar dapat digunakan kembali tanpa perlu melatih ulang.

Selesai

Hasil pengujian :

Tabel 3. 4 Hasil pengujian algoritma *K-Nearest Neighbor* dan *Logistic Regression*

Model	Tingkat Kematangan	*Accuracy*	*Precision*	*Recall*	*F1-Score*
Logistic Regression	Pepaya matang	73%	69%	85%	76%
Logistic Regression	Pepaya mengkal	73%	74%	79%	76%
Logistic Regression	Pepaya mentah	73%	79%	56%	66%
K-Nearest Neighbor	Pepaya matang	87%	86%	89%	87%
K-Nearest Neighbor	Pepaya mengkal	87%	89%	89%	89%

Model	Tingkat Kematangan	*Accuracy*	*Precision*	*Recall*	*F1-Score*
K-Nearest Neighbor	Pepaya mentah	87%	87%	84%	85%

Confusion Matrix akan digunakan untuk mengevaluasi kinerja model setelah tahap pemodelan selesai. *Confusion Matrix* menilai *performa* model berdasarkan nilai-nilai yang terdapat di dalamnya, yaitu *akurasi, presisi, recall,* dan *f1-score*. Data yang telah dipisahkan kemudian diuji untuk menghitung *akurasi, presisi, recall,* dan *f1-score.* Perhitungan nilai-nilai tersebut dilakukan dengan menggunakan Confusion Matrix yang terdiri dari empat elemen utama: *True Positive (TP), False Positive (FP), True Negative (TN),* dan *False Negative (FN).* Berikut adalah cara untuk menghitung evaluasi tersebut:

1. *Accuracy* : Mengukur proporsi keseluruhan prediksi yang benar, baik untuk kelas positif maupun negatif

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

2. *Precision* : Mengukur seberapa tepat model dalam memprediksi kasus positif.

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

3. *Recall* : Mengukur sejauh mana model mampu mengidentifikasi kasus positif.

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

4. *F1-Score :* Merupakan rata-rata harmonis antara *presisi* dan *recall,* yang memberikan gambaran keseimbangan antara keduanya.

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

Setelah melewati rancangan tahapan pengumpulan data, *preprocessing*, ekstrasi fitur, *split data*, pengujian dan evaluasi, maka hasil pengujian klasifikasi kematangan buah pepaya yang telah diperoleh akan dijelaskan dan dibahas lebih lanjut di bab hasil dan pembahasan.