

Nama : Erdianti Wiga Putri Andini

NIM : 13522053

## Tugas 5 - EB3204 Pembelajaran Mesin dalam Teknik Biomedis

Dataset: [Form Dataset Kematangan Buah PMB 2025 \(Responses\)](#)

Kode:

```
# ripe_classify.py

import cv2
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
import pickle
import os

FEATURE_NAMES = [
    "Mean H", "Mean S", "Mean V",
    "Std H", "Std S", "Std V",
    "Ratio S/H", "Ratio V/S",
    "Entropy H",
    "Prop Kuning", "Prop Hijau",
    "Std Gray"
]

def extract_extended_features(roi):
    hsv = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv)

    mean_h = np.mean(h)
    mean_s = np.mean(s)
    mean_v = np.mean(v)
    std_h = np.std(h)
    std_s = np.std(s)
    std_v = np.std(v)

    ratio_s_h = mean_s / (mean_h + 1e-5)
    ratio_v_s = mean_v / (mean_s + 1e-5)

    hist_h = cv2.calcHist([hsv], [0], None, [180], [0,180])
    hist_h = hist_h / hist_h.sum()
    entropy_h = -np.sum(hist_h * np.log2(hist_h + 1e-10))

    total_pixels = h.size
    prop_kuning = np.sum((h >= 20) & (h <= 40)) / total_pixels
    prop_hijau = np.sum((h >= 50) & (h <= 70)) / total_pixels

    gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    std_gray = np.std(gray)
```

```

    return np.array([
        mean_h, mean_s, mean_v,
        std_h, std_s, std_v,
        ratio_s_h, ratio_v_s,
        entropy_h,
        prop_kuning, prop_hijau,
        std_gray
    ])

def train_classifier(features_df, model_type='rf'):
    if 'Label' in features_df.columns and 'Filename' in features_df.columns:
        X = features_df.drop(['Label', 'Filename'], axis=1)
        y = features_df['Label']
    else:
        X = features_df.iloc[:, :-2]
        y = features_df.iloc[:, -2]

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    X_train, X_test, y_train, y_test = train_test_split(
        X_scaled, y, test_size=0.2, random_state=42
    )

    if model_type == 'rf':
        clf = RandomForestClassifier(n_estimators=100, random_state=42)
    elif model_type == 'gb':
        clf = GradientBoostingClassifier(n_estimators=100, random_state=42)
    elif model_type == 'svm':
        clf = SVC(probability=True, random_state=42)
    elif model_type == 'optimal':
        param_grid = {
            'C': [0.1, 1, 10, 100],
            'gamma': ['scale', 'auto', 0.1, 0.01],
            'kernel': ['rbf', 'poly', 'sigmoid']
        }
        clf = GridSearchCV(SVC(probability=True), param_grid, cv=5)

    clf.fit(X_train, y_train)

    if model_type == 'optimal':
        print(f"Best parameters: {clf.best_params_}")
        clf = clf.best_estimator_

    accuracy = clf.score(X_test, y_test)
    print(f"Model {model_type} accuracy: {accuracy:.4f}")

    if hasattr(clf, 'feature_importances_'):
        feature_importance = pd.DataFrame({
            'Feature': X.columns,
            'Importance': clf.feature_importances_
        }).sort_values('Importance', ascending=False)
        print(feature_importance.head(5))

    with open(f'fruit_classifier_{model_type}.pkl', 'wb') as f:
        pickle.dump((clf, scaler, X.columns.tolist()), f)

    return clf, scaler, X.columns.tolist()

```

```

def real_time_classification(model_type='rf'):
    model_path = f'fruit_classifier_{model_type}.pkl'

    try:
        with open(model_path, 'rb') as f:
            try:
                clf, scaler, feature_names = pickle.load(f)
                print(f"Model {model_type} loaded successfully")
            except ValueError:
                f.seek(0)
                clf, scaler = pickle.load(f)
                feature_names = FEATURE_NAMES

                with open(model_path, 'wb') as f_new:
                    pickle.dump((clf, scaler, feature_names), f_new)
    except Exception as e:
        print(f"Loading model error: {e}")
        print("Training new model...")
        all_features = pd.read_csv('fitur_buah.csv')
        clf, scaler, feature_names = train_classifier(all_features, model_type)

for cam_index in [0, 1, 2]:
    print(f"Testing camera {cam_index}...")
    cap = cv2.VideoCapture(cam_index)
    if cap.isOpened():
        print(f"Camera {cam_index} connected")
        break
    cap.release()
else:
    print("No camera found")
    return

width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
print(f"Resolution: {width}x{height}")

frame_counter = 0

while True:
    ret, frame = cap.read()
    if not ret:
        print(f"Frame grab failed #{frame_counter}")
        if frame_counter > 0:
            cap.release()
            cap = cv2.VideoCapture(cam_index)
            if not cap.isOpened():
                print("Camera reconnection failed")
                break
            continue
        else:
            break

    frame_counter += 1

    cv2.imshow("Camera Feed", frame)

    h, w, _ = frame.shape
    crop_w = int(w * 0.25)
    crop_h = int(h * 0.25)
    x1 = (w - crop_w) // 2

```

```

y1 = (h - crop_h) // 2
x2 = x1 + crop_w
y2 = y1 + crop_h

x1, y1 = max(0, x1), max(0, y1)
x2, y2 = min(w, x2), min(h, y2)

if x1 >= x2 or y1 >= y2 or x2 <= 0 or y2 <= 0:
    continue

roi = frame[y1:y2, x1:x2].copy()

if roi.size == 0:
    continue

cv2.imshow("ROI", roi)

try:
    features = extract_extended_features(roi)
    features_df = pd.DataFrame([features], columns=feature_names)
    features_scaled = scaler.transform(features_df)

    prediction = clf.predict(features_scaled)[0]
    probabilities = clf.predict_proba(features_scaled)[0]
    confidence = max(probabilities) * 100

    if prediction == 'matang':
        color = (0, 255, 255)
    elif prediction == 'mentah':
        color = (0, 255, 0)
    else:
        color = (0, 0, 255)

    overlay = frame.copy()
    cv2.rectangle(overlay, (x1, y1), (x2, y2), color, 3)

    cv2.putText(overlay, f"Prediction: {prediction}", (10, 30),
               cv2.FONT_HERSHEY_SIMPLEX, 0.8, color, 2)
    cv2.putText(overlay, f"Confidence: {confidence:.2f}%", (10, 70),
               cv2.FONT_HERSHEY_SIMPLEX, 0.8, color, 2)

    y_pos = 110
    key_features = ['Prop Kuning', 'Prop Hijau', 'Ratio S/H', 'Ratio V/S']
    for i, name in enumerate(key_features):
        try:
            idx = feature_names.index(name)
            value = features[idx]
        except ValueError:
            idx = {'Prop Kuning': 9, 'Prop Hijau': 10, 'Ratio S/H': 6, 'Ratio V/S': 7}.get(name, 0)
            value = features[idx]

        cv2.putText(overlay, f"{name}: {value:.2f}", (10, y_pos + i*40),
                   cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2)

    cv2.imshow("Fruit Ripeness Classification", overlay)

except Exception as e:
    print(f"Processing error: {e}")
    cv2.imshow("Fruit Ripeness Classification", frame)

```

```

key = cv2.waitKey(1) & 0xFF
if key == 27:
    break

cap.release()
cv2.destroyAllWindows()

if __name__ == "__main__":
    try:
        all_features = pd.read_csv('fitur_buah.csv')
        print(f'Data: {all_features.shape} rows')
    except Exception as e:
        print(f'Data loading error: {e}')
        exit(1)

    model_type = 'optimal'

    model_path = f'fruit_classifier_{model_type}.pkl'
    if os.path.exists(model_path):
        try:
            with open(model_path, 'rb') as f:
                model_data = pickle.load(f)
                if len(model_data) != 3:
                    print("Updating model format...")
                    os.remove(model_path)
        except:
            print("Recreating corrupted model...")
            os.remove(model_path)

    if not os.path.exists(model_path):
        print(f'Training {model_type} model...')
        train_classifier(all_features, model_type)

real_time_classification(model_type)

```

```

# data.py
import pandas as pd
import requests
import re
import io
import os
import time
from urllib.parse import urlparse, parse_qs

def extract_links(combined_links):
    parts = re.split(r'(?=https://)', combined_links)
    links = [part for part in parts if part.startswith('https://')]

    return links

def download_csv_from_drive(file_id):
    try:
        url = f"https://drive.google.com/uc?export=download&id={file_id}"
        response = requests.get(url)

        if response.status_code == 200:
            try:
                return pd.read_csv(io.StringIO(response.content.decode('utf-8')))
            except:
                pass
    except:
        pass

```

```

        except Exception as e:
            print(f"Error parsing CSV with first method: {e}")

url = f"https://drive.google.com/file/d/{file_id}/view"
print(f"Trying alternative method for {url}")

session = requests.Session()
response = session.get(url)

if response.status_code == 200:
    try:
        export_url = f"https://drive.google.com/uc?export=download&id={file_id}"
        response = session.get(export_url)
        return pd.read_csv(io.StringIO(response.content.decode('utf-8')))
    except Exception as e:
        print(f"Error with alternative method: {e}")
        return None
else:
    print(f"Failed to download file with ID {file_id}. Status code: {response.status_code}")
    return None
except Exception as e:
    print(f"Error during download: {e}")
    return None

def download_sheet_from_drive(sheet_id):
    try:
        url = f"https://docs.google.com/spreadsheets/d/{sheet_id}/export?format=csv"
        response = requests.get(url)
        if response.status_code == 200:
            try:
                return pd.read_csv(io.StringIO(response.content.decode('utf-8')))
            except Exception as e:
                print(f"Error parsing sheet: {e}")
                return None
        else:
            print(f"Failed to download sheet with ID {sheet_id}. Status code: {response.status_code}")
            return None
    except Exception as e:
        print(f"Error during sheet download: {e}")
        return None

def extract_file_id(url):
    try:
        if 'drive.google.com/file/d/' in url:
            file_id = url.split('/file/d/')[1].split('/')[0]
            return file_id
        elif 'docs.google.com/spreadsheets/d/' in url:
            sheet_id = url.split('/spreadsheets/d/')[1].split('/')[0]
            return sheet_id
        elif 'drive.google.com/uc?export=download&id=' in url:
            parsed_url = urlparse(url)
            query_params = parse_qs(parsed_url.query)
            return query_params.get('id', [None])[0]
        else:
            print(f"Unsupported URL format: {url}")
            return None
    except Exception as e:
        print(f"Error extracting file ID from {url}: {e}")

```

```

        return None

def process_link(link):
    file_id = extract_file_id(link)
    if not file_id:
        return None

    if 'docs.google.com/spreadsheets' in link:
        return download_sheet_from_drive(file_id)
    else:
        return download_csv_from_drive(file_id)

def save_individual_csv(df, index, folder="individual_csvs"):
    if not os.path.exists(folder):
        os.makedirs(folder)

    filename = os.path.join(folder, f"source_{index}.csv")
    df.to_csv(filename, index=False)
    print(f"Saved individual CSV to {filename}")

def main():
    combined_links =
"""https://drive.google.com/file/d/1i_9ka1G2nGsot-8uhAG5qGoTNG43IJje/view?usp=sharinghttps://drive.google.com/file/d/1YvSR3kFb-LBweOqC9zLQet8r1fgCAbEe/view?usp=sharinghttps://drive.google.com/file/d/1lhciDgpwpmiFM5g1PRjEDaY7qJFxLulk/view?usp=drive_linkhttps://drive.google.com/file/d/1EHb8TBtHs1xcmqhWAQPZ4oaswlsuvUK0/view?usp=drive_linkhttps://drive.google.com/file/d/1ss_2m1UDGkIEQc_kv01Uah0ZARDHio1/view?usp=sharinghttps://drive.google.com/file/d/1VJuJehQj0-bZMTNVN6IrJaRVKPbq67uA/view?usp=drive_linkhttps://drive.google.com/file/d/13nJGU4Vop613ktqiz6mcxydZRF-st-CN/view?usp=sharinghttps://docs.google.com/spreadsheets/d/1sJpf9maa-qDq82w8X7cTelmLOLfjl1bvRG6qjFkPysVI/edit?usp=sharinghttps://docs.google.com/spreadsheets/d/1HJ6byfqNgr3cqUf1mL-LAS9hc8Jwc-Bc0q1zr0i019c/edit?usp=sharinghttps://drive.google.com/file/d/1IPz7Cc1n4By0qUyGf2JHz2Q8n3ZnUSgf/view?usp=sharinghttps://drive.google.com/file/d/1CCwkhQynj2Ii4weNjDjKrK53zD_7yh_f/view?usp=sharinghttps://docs.google.com/spreadsheets/d/10uFAXzjY2Vosj6j3Sm3otFvkAbdkZL11/edit?usp=sharing&ouid=105011830416643335170&rtpof=true&sd=truehttps://drive.google.com/file/d/1DCUrPAsXnnf09NLDCv8zp_Xqkx--dkcn/view?usp=drive_linkhttps://drive.google.com/file/d/10nK_vAmz5ALg4m9Z6r0M1r-K01sUVy8x/view?usp=sharinghttps://drive.google.com/file/d/1Nmre_LgSwpJ52evhdVy4W_4K0MAvpW3b/view?usp=sharinghttps://drive.google.com/file/d/1rmBXz0uaNd3MzMPbd_YFWcar_yEwMQoR/view?usp=sharinghttps://drive.google.com/file/d/1Uqxkpy5BKZxYEoy91jppZLXlzYIcyry_/view?usp=sharinghttps://drive.google.com/file/d/1Hs1Y1caaERD-Vy3rTsnD3IMTDH3Q0js/view?usp=sharinghttps://drive.google.com/file/d/1EM_lTewgsTubjn58mjVi4Mk8MTiqN2cA/view?usp=drive_linkhttps://drive.google.com/file/d/13N28_A6B004XepqvxE2uHfntfdKdtWSa/view?usp=drive_linkhttps://docs.google.com/spreadsheets/d/1gBNMX1QfcDtJK6pjmb5YbPRjk8jKGAVjc4pjBuRR-4Y/edit?usp=sharinghttps://drive.google.com/file/d/1hwUPo2_0d2GJ4p33AClu00PydlBtfzkR/view?usp=sharinghttps://drive.google.com/file/d/1pQZ1YWyDgdW75New-ytiU_yi9Iev7Sh8/view?usp=sharinghttps://drive.google.com/file/d/1Pf49LaypNOXMkQaqoscSJ4gQ8h-82e8/view?usp=sharinghttps://docs.google.com/spreadsheets/d/1HJ6byfqNgr3cqUf1mL-LAS9hc8Jwc-Bc0q1zr0i019c/edit?usp=sharinghttps://drive.google.com/file/d/1Le0aIDGbZEDDbV9Zr4pb4oQoCqgqMaDp/view?usp=sharinghttps://drive.google.com/file/d/1sH5F2tl4BPO2j-TR2GNagFKd7HYkLSW/view?usp=drive_linkhttps://drive.google.com/file/d/16vIxIEN499A0mrFr3MA4-Tv6BeT_nlyr/view?usp=sharinghttps://drive.google.com/file/d/1puYPN8_kVVGDEMyp0XBLraigulQY3uPVK/view?usp=sharinghttps://drive.google.com/file/d/1LW9vkLnhyfb_mYdFQ5qb3BbsOKLdaZx/_view?usp=sharinghttps://drive.google.com/file/d/1NoLtfCLUe2dAr3Ac0fxXqgwxLwqbthQ/view?usp=sharinghttps://drive.google.com/file/d/17FsW-1ltHhqDeQ9DkDNOYy00Bivy3lJS/view?usp=sharinghttps://drive.google.com/file/d/1_y9JG7_di_FFxFzijp4qaF8PNyRfEGi01/view?usp=sharinghttps://docs.google.com/spreadsheets/d/1vgObEQT19RHftKTxju2AlBigReK-WbZ3lo30Ht8b1ueQ/edit?usp=sharinghttps://docs.google.com/spreadsheets/d/1Sj08vcbqQ9WIHAoLstwyz5Bz5k0Y0J53/edit?usp=sharing&ouid=117400838184951060020&rtpof=true&sd=truehttps://drive.google.com/file/d/1cNH1Kp4VTevMyhBVRLrp6rnN192aYs5J/view?usp=sharinghttps://drive.google.com/file/d/1UgUkP_LRjc77VKPHXP5eVERcmseZ7fD/view?usp=sharinghttps://drive.google.com/file/d/1BgWnTZe8bLS6nYSF112VHqyqmxa5L5W2/view?usp=drive_linkhttps://drive.google.com/file/d/1GI_tEx54RfmnL5c9uw7Pq1w89d-IEhju/view?usp=drive_linkhttps://drive.google.com/file/d/18NXzTv64-T_ETdgaujvvoUABtK7DSeo_/view?u
"""

```

```

sp=drive_linkhttps://drive.google.com/drive/folders/1S6SVn4lRdjbj-3cNz8WzQgTBFcyWvzRn?usp=dr
ive_linkhttps://drive.google.com/file/d/1LwaDTWPu7DTYQiJ3k0Y-P_a1yuNGBFbZ/view?usp=sharinght
tps://docs.google.com/spreadsheets/d/1kPFQgEGFT7T17jISfJFbHSFVg8A0zorb7_g1Q2M6La0/edit?usp=d
rive_linkhttps://drive.google.com/file/d/1Gn56Yt9ML9M6H28qquo9hk8vXxA0tG3/view?usp=sharing"
"""

links = extract_links(combined_links)
print(f"Found {len(links)} links")

all_dataframes = []
success_count = 0
failure_count = 0

for i, link in enumerate(links):
    print(f"\nProcessing link {i+1}/{len(links)}: {link}")

    if 'drive.google.com/drive/folders' in link:
        print("Skipping folder link")
        continue

    if i > 0 and i % 3 == 0:
        print("Pausing briefly to avoid rate limiting...")
        time.sleep(2)

    df = process_link(link)

    if df is not None:
        print(f"Successfully downloaded data. Shape: {df.shape}")
        print(f"Columns: {df.columns.tolist()}")

        required_columns = ['Mean H', 'Mean S', 'Mean V', 'Label', 'Filename']
        missing_columns = [col for col in required_columns if col not in df.columns]

        if missing_columns:
            print(f"Warning: Missing required columns: {missing_columns}")
            failure_count += 1
            continue

        save_individual_csv(df, i)
        all_dataframes.append(df)
        success_count += 1
        print(f"Successfully processed dataset {i+1}")
    else:
        print(f"Failed to process link {i+1}")
        failure_count += 1

if all_dataframes:
    combined_df = pd.concat(all_dataframes, ignore_index=True)
    print(f"\nCombined data shape: {combined_df.shape}")

    combined_df.to_csv('fitur_buah.csv', index=False)
    print("Data saved to fitur_buah.csv")

    print(f"\nSummary:")
    print(f"Total links processed: {len(links)}")
    print(f"Successful: {success_count}")
    print(f"Failed: {failure_count}")
    print(f"Total rows in combined dataset: {combined_df.shape[0]}")

    if 'Label' in combined_df.columns:

```

```

label_counts = combined_df['Label'].value_counts()
print("\nClass distribution:")
for label, count in label_counts.items():
    print(f" {label}: {count} samples ({count/combined_df.shape[0]*100:.1f}%)")
else:
    print("\nNo valid data found. Please check the URLs and make sure the files are
accessible.")
    print("Individual CSV files may still be available in the 'individual_csvs'
directory for inspection.")

if __name__ == "__main__":
    main()

```

## Fitur yang Dipilih

Jelaskan fitur yang yang anda pilih dan strategi/alasan yang anda pakai!

### Jawaban:

Kode 'ripe\_classify.py' ini menggunakan 12 fitur untuk menentukan kematangan buah, yaitu:

1. **Mean H, Mean S, Mean V** - Rata-rata nilai warna dalam format HSV (Hue, Saturation, Value). Ini membantu mengenali warna dominan buah.
2. **Std H, Std S, Std V** - Standar deviasi nilai HSV. Ini mengukur variasi warna pada buah.
3. **Ratio S/H, Ratio V/S** - Perbandingan antara komponen warna. Rasio ini bisa menunjukkan perubahan karakteristik warna saat buah matang.
4. **Entropy H** - Mengukur "kekacauan" distribusi warna Hue. Buah dengan warna yang lebih beragam memiliki entropi lebih tinggi.
5. **Prop Kuning, Prop Hijau** - Proporsi piksel dengan warna kuning dan hijau. Ini sangat penting karena banyak buah berubah dari hijau ke kuning saat matang.
6. **Std Gray** - Standar deviasi citra grayscale, mengukur variasi kecerahan.

Alasan saya memilih untuk menggunakan fitur-fitur ini karena:

- Berfokus pada warna (komponen paling penting untuk kematangan)
- Menyertakan proporsi warna spesifik (kuning dan hijau) yang berkaitan langsung dengan proses pematangan
- Menggunakan rasio antar komponen warna untuk menangkap perubahan karakteristik visual
- Menambahkan variasi kecerahan yang bisa menunjukkan perubahan tekstur

## **Langkah Pengembangan**

Jelaskan langkah-langkah pengembangan yang anda lakukan, termasuk trik khusus apabila ada!

### **Jawaban:**

Dalam mengembangkan sistem klasifikasi kematangan buah secara real-time menggunakan webcam, saya melakukan beberapa langkah sebagai berikut:

#### **1. Pengumpulan dan Penggabungan Data**

Menggunakan kode ‘data.py’ untuk mengumpulkan 44 dataset dari berbagai link Google Drive dengan tahapan:

- Mengekstrak link individual dari string gabungan
- Download file CSV atau Google Sheets dari setiap link
- Memeriksa kelengkapan kolom yang dibutuhkan (Mean H, Mean S, Mean V, Label, Filename)
- Menyimpan setiap dataset dalam file terpisah untuk pemeriksaan
- Menggabungkan semua dataset yang valid menjadi satu file 'fitur\_buah.csv'
- Menampilkan statistik distribusi kelas (matang/mentah) untuk memastikan keseimbangan data

#### **2. Ekstraksi Fitur**

Pada kode ‘ripe\_classify.py’ terdapat fungsi extract\_extended\_features() untuk mengambil citra buah dan menghitung 12 fitur yang sudah dijelaskan sebelumnya.

#### **3. Training Model**

Pada kode ‘ripe\_classify.py’ terdapat fungsi train\_classifier() yang melatih model machine learning dengan beberapa opsi:

- Random Forest (rf)
- Gradient Boosting (gb)
- SVM (Support Vector Machine)
- Model ‘optimal’ yang menggunakan GridSearchCV untuk menemukan parameter terbaik

#### **4. Normalisasi Data**

Menggunakan StandardScaler untuk menormalkan fitur sehingga model bekerja lebih baik

## **5. Penyimpanan Model**

Menyimpan model dalam file pickle untuk digunakan kembali

## **6. Klasifikasi Real-time**

Pada kode ‘ripe\_classify.py’ terdapat fungsi `real_time_classification()` untuk menjalankan deteksi langsung menggunakan kamera dengan tahapan:

- Mengambil gambar dari kamera
- Membuat area fokus (ROI) di tengah frame
- Mengekstrak fitur dari ROI
- Memprediksi kematangan buah
- Menampilkan hasil dengan visual yang jelas (kotak warna dan teks)

Trik khusus yang digunakan dalam program ini adalah :

- Penanganan berbagai format Google Drive (file dan spreadsheet) untuk mengumpulkan dataset
- Penanganan kesalahan yang baik jika model tidak ditemukan atau rusak
- Mekanisme anti rate-limiting dengan penundaan antara permintaan download
- Pengujian beberapa indeks kamera (0, 1, 2) untuk menemukan kamera yang berfungsi
- Mekanisme pemulihan jika kamera gagal
- Menampilkan hasil visual yang jelas dengan kode warna (kuning untuk matang, hijau untuk mentah)

## Uji Coba dengan Buah Asli

Lakukan uji coba dan analisa dengan buah secara riil!

### Jawaban:

Untuk uji coba dengan buah asli, program ini melakukan:

1. Memuat model yang sudah dilatih dengan data gabungan dari 44 sumber
2. Mengambil gambar buah melalui kamera
3. Menganalisis proporsi warna kuning dan hijau
4. Menghitung fitur-fitur penting lainnya
5. Memprediksi apakah buah "matang" atau "mentah"
6. Menampilkan hasil

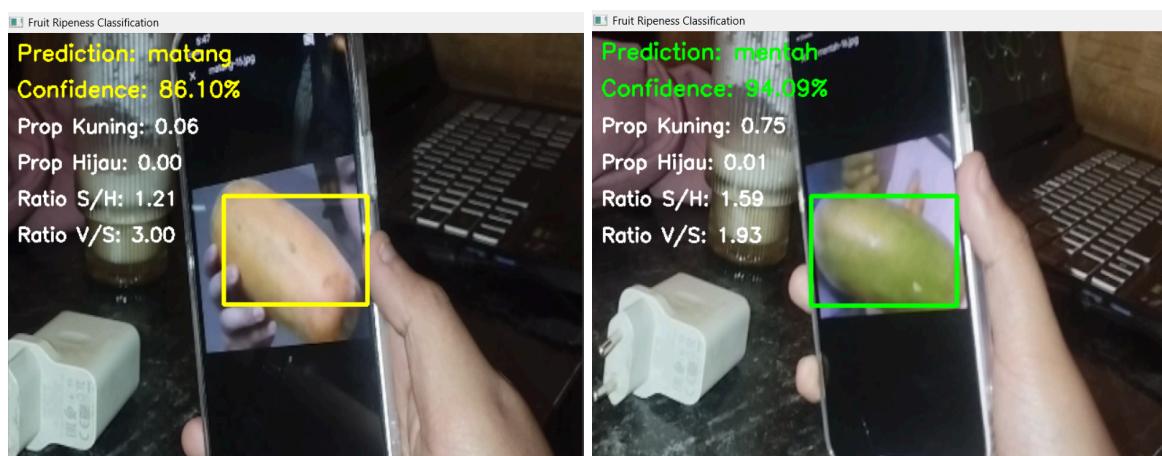
Saat menjalankan program, yang akan terlihat adalah:

- Kotak di tengah layar yang menunjukkan area fokus
- Prediksi kematangan buah (matang/mentah)
- Persentase keyakinan prediksi
- Nilai-nilai fitur penting seperti proporsi warna kuning dan hijau

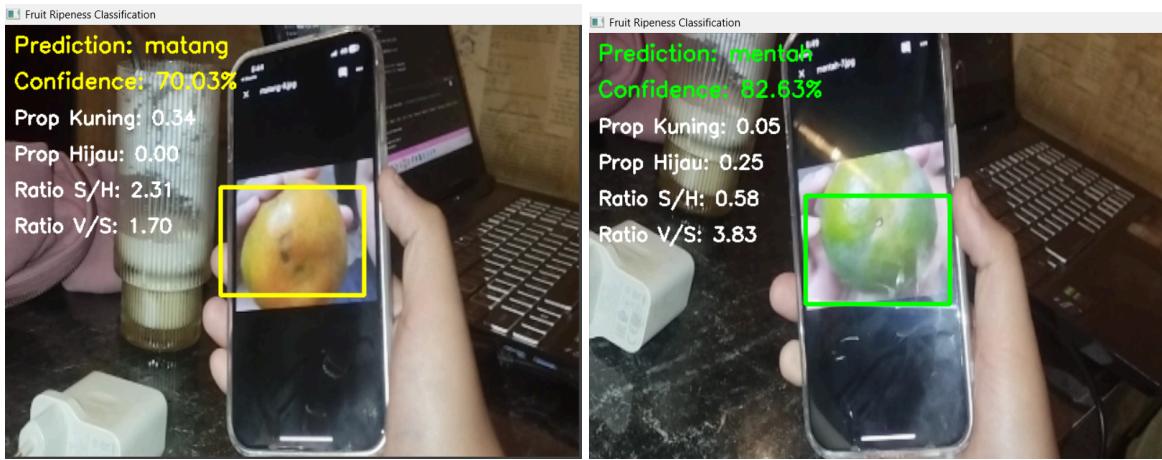
Model yang digunakan adalah model terbaik yang dipilih secara otomatis lewat proses GridSearchCV, jadi hasilnya lebih akurat. Kelebihan sistem ini adalah menggunakan data yang sangat banyak dari berbagai sumber, sehingga model bisa mengenali berbagai jenis buah dan kondisi cahaya dengan lebih baik.

Hasil uji coba program ini terlampir sebagai berikut.

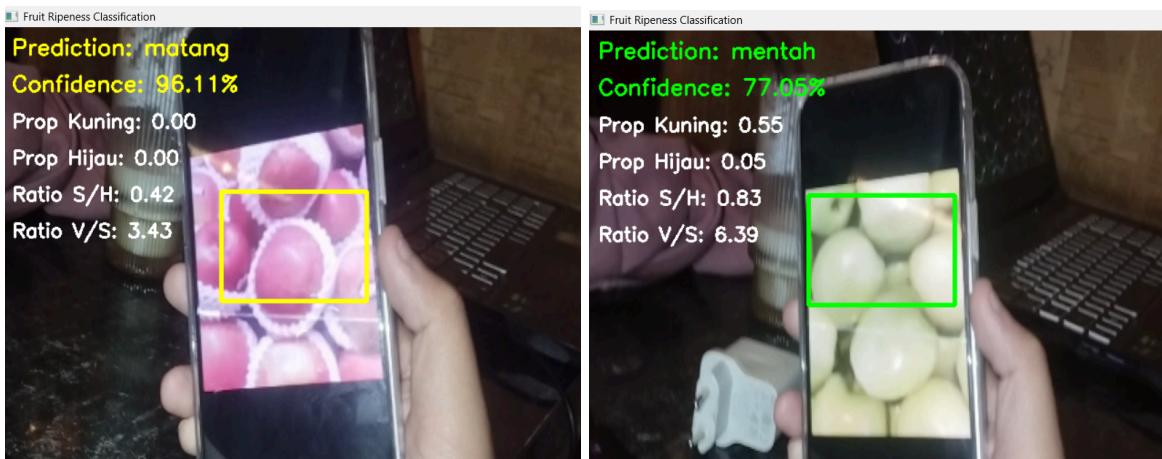
Pepaya



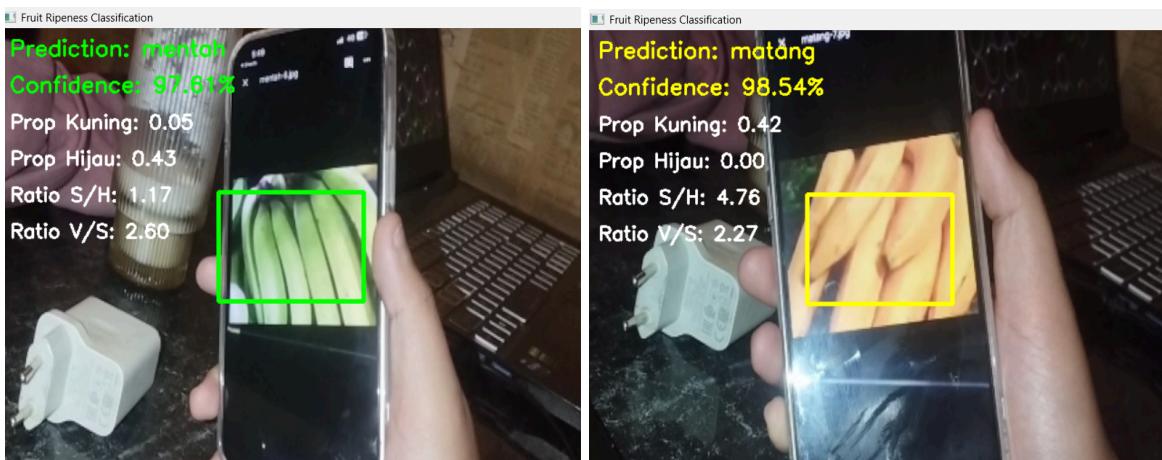
## Jeruk



## Apel



## Pisang



Berdasarkan hasil uji coba tersebut, sistem klasifikasi kematangan buah bekerja dengan baik dalam mengenali buah matang dan mentah. Dari hasil uji coba terlihat:

- Buah hijau (seperti pepaya muda, apel hijau, dan pisang hijau) berhasil dikenali sebagai "mentah" dengan confidence tinggi antara 77-97%.
- Buah kuning dan berwarna (seperti pisang kuning, apel merah, dan buah berwarna ungu) dikenali sebagai "matang" dengan confidence antara 70-98%.
- Sistem tidak hanya bergantung pada warna kuning saja, tapi juga nilai rasio S/H dan V/S untuk menentukan kematangan.
- Proporsi warna hijau selalu tinggi pada buah mentah, sementara buah matang memiliki proporsi hijau mendekati nol.
- Rasio V/S cenderung lebih tinggi pada buah mentah dibanding buah matang.

Kesimpulannya, fitur warna HSV yang dipilih terbukti efektif untuk mengenali kematangan berbagai jenis buah dalam kondisi pencahayaan yang berbeda.