FINAL PROJECT

MATA KULIAH PENGANTAR PEMROSESAN DATA MULTIMEDIA

DISUSUN OLEH:

Ananda Putra	**2108561001**
Ni Putu Suci Paramita	**2108561011**
Hammam Akmal Prathama	**2108561016**

PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS UDAYANA

2023

BAB I
PENDAHULUAN

1.1 Latar Belakang

Image preprocessing adalah serangkaian teknik yang digunakan untuk mempersiapkan data gambar sebelum diolah oleh algoritma machine learning [6]. Tahap preprocessing sangat penting dalam proses training model, karena dapat membantu meningkatkan akurasi dan efisiensi algoritma. Salah satu algoritma yang sering digunakan dalam pengenalan pola pada data gambar adalah algoritma k-Nearest Neighbor (K-NN).

Algoritma K-NN merupakan salah satu algoritma machine learning yang sederhana dan populer dalam klasifikasi data. Algoritma ini bekerja dengan cara mencari k data terdekat dalam ruang fitur berdasarkan jarak Euclidean atau jarak lainnya, lalu melakukan mayoritas pemungutan suara untuk menentukan label kelas dari data yang sedang diprediksi. Dalam konteks pengenalan pola pada data gambar, K-NN dapat digunakan untuk mengklasifikasikan gambar berdasarkan fitur-fitur yang dihasilkan dari preprocessing [8].

Tahap preprocessing dalam penggunaan algoritma K-NN pada data gambar melibatkan beberapa langkah penting. Berikut adalah beberapa langkah umum dalam image preprocessing sebelum menggunakan algoritma K-NN:

1. Resolusi dan Ukuran: Gambar dalam dataset dapat memiliki resolusi dan ukuran yang berbeda-beda. Langkah pertama adalah menyamakan ukuran dan resolusi gambar agar memiliki dimensi yang seragam. Hal ini dapat dilakukan dengan melakukan resizing, cropping, atau padding pada gambar.
2. Normalisasi: Normalisasi gambar adalah proses mengubah intensitas piksel agar memenuhi rentang yang diinginkan. Misalnya, dapat dilakukan normalisasi dengan mengubah rentang intensitas piksel menjadi antara 0 hingga 1 atau -1 hingga 1. Hal ini dilakukan untuk memudahkan perhitungan jarak antar piksel pada algoritma K-NN.
3. Pemrosesan Warna: Jika gambar berwarna, maka dapat dilakukan pemrosesan warna untuk mengubah gambar menjadi skala abu-abu atau mengambil saluran warna tertentu yang lebih relevan untuk klasifikasi.

Pemrosesan warna juga dapat melibatkan ekstraksi fitur warna seperti histogram warna atau metode ekstraksi ciri lainnya.

4. Ekstraksi Fitur: Ekstraksi fitur adalah langkah penting dalam preprocessing gambar untuk klasifikasi menggunakan algoritma K-NN. Fitur-fitur yang relevan diekstraksi dari gambar untuk mewakili karakteristik unik yang membedakan kelas. Contoh metode ekstraksi fitur yang umum digunakan adalah metode ekstraksi ciri tekstur, seperti Local Binary Patterns (LBP) atau metode ekstraksi ciri bentuk seperti Histogram of Oriented Gradients (HOG) serta Gray-Level Co-occurrence Matrix (GLCM).
5. Reduksi Dimensi: Dalam beberapa kasus, gambar memiliki dimensi fitur yang besar. Reduksi dimensi dapat dilakukan untuk mengurangi kompleksitas perhitungan dan mempercepat proses training.

Setelah langkah-langkah preprocessing di atas selesai, data gambar siap digunakan untuk training algoritma K-NN. Dalam tahap training, K-NN akan menghitung jarak antara fitur-fitur gambar yang telah diekstraksi dengan fitur-fitur gambar pada dataset latihan. Kemudian, algoritma akan menentukan kelas yang paling sering muncul dari k data terdekat dan menggunakannya sebagai prediksi untuk gambar yang sedang diproses.

Dengan melakukan langkah-langkah preprocessing yang tepat, algoritma K-NN dapat memberikan hasil klasifikasi yang lebih baik pada data gambar. Preprocessing membantu dalam mengurangi keberisikan, menormalisasi data, dan mengekstraksi fitur yang relevan, sehingga meningkatkan kemampuan algoritma untuk memahami dan mengklasifikasikan gambar dengan lebih baik.

1.2 Problem Komputasi

1. Feature Extraction dengan metode berbasis tekstur menggunakan metode GLCM (Gray-Level Co-occurrence Matrix) untuk menentukan 6 nilai fitur dari data image: dissimilarity, correlation, homogeneity, contrast, ASM, energy, dengan 5 sudut (0, 45, 90, 135, dan 180). Total ada 30 nilai fitur (6 x 5) untuk setiap data image.

2. Training dilaksanakan untuk menghasilkan model klasifikasi yang terbaik. Untuk metode KNN training dilakukan dengan mencoba beberapa nilai k yang ganjil (contoh k=3, atau 5, atau 7, atau 9).
3. Model yang dihasilkan di deploy ke sistem aplikasi berbasis web dengan fitur utama adalah user menginput satu atau beberapa data dan outputnya adalah hasil sentimen atau identifikasi emosi.

1.3 Tujuan

Berikut tujuan dari penugasan final project

1. Untuk mengidentifikasi emosi dari sebuah citra ekspresi wajah dengan 2 sentimen yaitu happy dan sad.
2. Untuk mendapatkan hasil dari tahap image preprocessing dengan ekstraksi fitur menggunakan metode GLCM (Gray-Level Co-occurrence Matrix).
3. Untuk metode KNN training dilakukan dengan mencoba beberapa nilai k yang ganjil (contoh k=1, atau 3, atau 5, atau 7, atau 9).

1.4 Manfaat

Adapun manfaat yang didapatkan dari penugasan final project ini

1. Mendapatkan hasil identifikasi dari sebuah gambar (sad atau happy).
2. Untuk mendapatkan hasil dari tahap image preprocessing dengan ekstraksi fitur menggunakan metode GLCM (Gray-Level Co-occurrence Matrix).
3. Mendapatkan hasil dari akurasi training machine learning dengan algoritma K-NN.

BAB II
ISI

2.1 Landasan Teori

2.1.1 Image Processing

Image processing adalah metode pengolahan atau manipulasi gambar dua dimensi [2]. Image processing bertujuan untuk memperbaiki kesalahan data sinyal citra yang terjadi akibat transmisi dan deteksi sinyal, serta memperbaiki tampilan citra sehingga dapat diinterpretasikan oleh sistem visual manusia melalui manipulasi dan lebih mudah. Citra grayscale merupakan gambar yang menggunakan pada tingkatan warna abu – abu [1]. Warna Abu-abu adalah satu-satunya warna dalam ruang RGB di mana komponen merah, hijau, dan biru memiliki intensitas yang sama. Untuk gambar grayscale, hanya menentukan nilai intensitas untuk setiap piksel sebagai satu nilai, sedangkan untuk gambar berwarna, Anda menentukan tiga nilai intensitas untuk setiap piksel. Intensitas gambar skala abu-abu disimpan sebagai bilangan bulat 8-bit dan menawarkan 256 kemungkinan dari level 0 hingga 255 (0 untuk hitam dan 255 untuk putih, di antaranya adalah level abu-abu).

2.1.2 GLCM

Grayscale Co-Occurrence Matrix (GLCM) adalah metode analisis tekstur/ekstraksi fitur. GLCM adalah matriks yang menggambarkan frekuensi pasangan dua piksel dengan intensitas tertentu pada jarak dan arah tertentu dalam gambar [7]. Prinsip dasar GLCM adalah menghitung frekuensi pasangan nilai intensitas piksel yang terjadi pada jarak dan arah tertentu pada citra grayscale. Dengan membangun matriks koeksistensi, kita dapat menganalisis hubungan distribusi dan kejadian dari sepasang nilai intensitas yang memberikan informasi tentang tekstur citra.

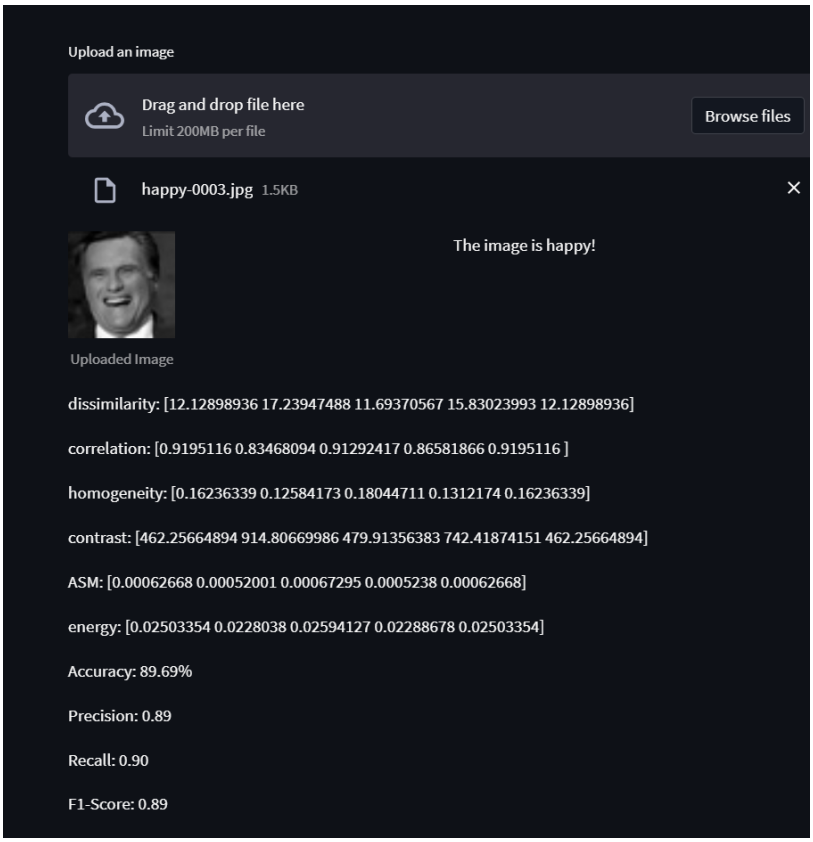
2.1.3 KNN

K-Nearest Neighbor (K-NN) merupakan salah satu metode klasifikasi yang tersedia dalam penambangan data dan termasuk dalam kelompok pembelajaran berbasis instan. KNN dilakukan dengan mencari k objek pada data training yang paling dekat (mirip) dengan objek pada data test [4]. KNN merupakan algoritma supervised learning dimana hasil dari query

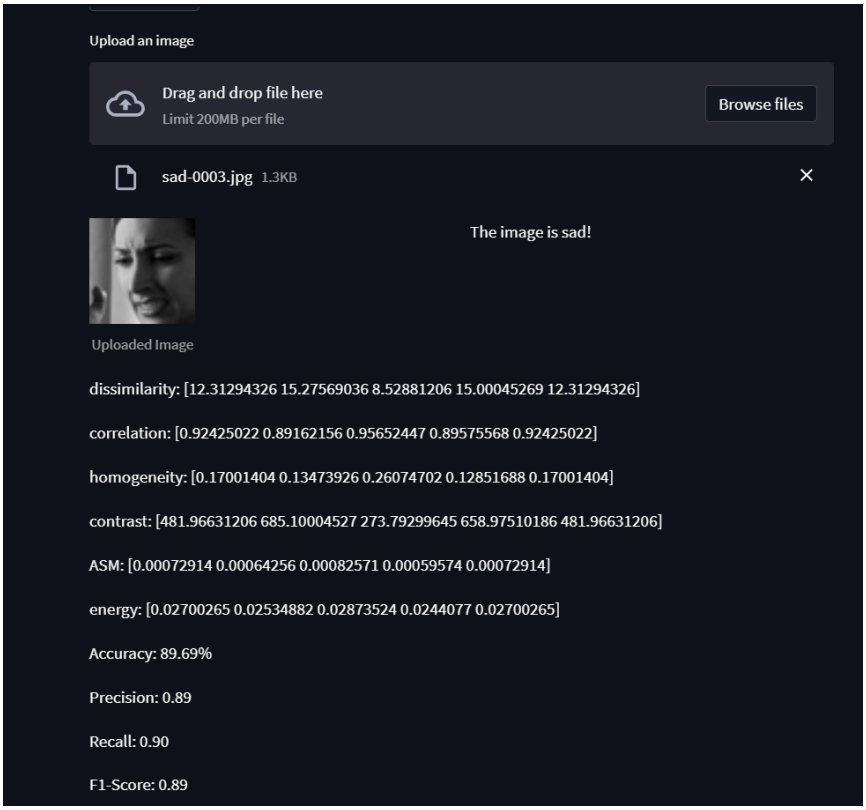
instance yang baru diklasifikan berdasarkan mayoritas dari kategori pada algoritma KNN. Dimana kelas yang paling banyak muncul yang nantinya akan menjadi kelas hasil dari klasifikasi [5].

2.2 Fitur Sistem

2.2.1 Tampilan Output Program



Gambar 1 Hasil Image Happy



Gambar 2 Hasil Image Sad

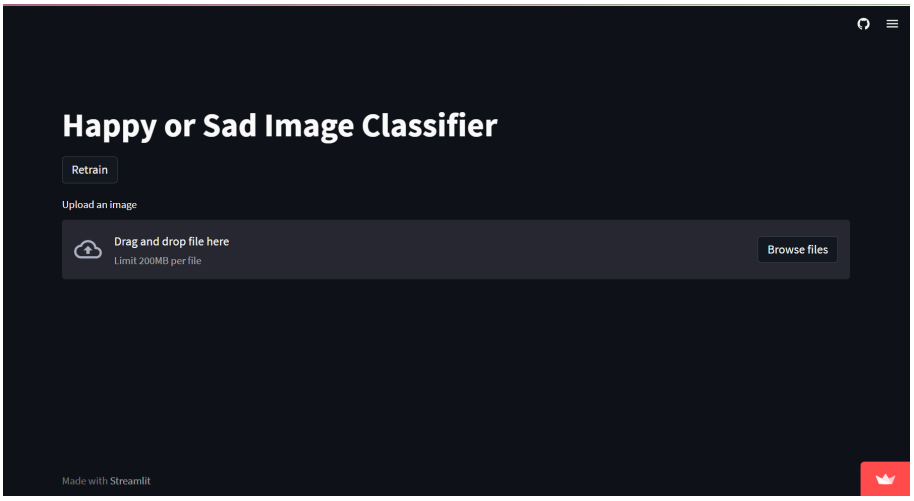
Tampilan output program akan seperti pada gambar 1 dan 2, pada gambar file yang di unggah dan hasil prediksi emosi. Jawaban dari hasil prediksi terdapat 2 kemungkinan yaitu *happy* dan *sad.*

2.2.2 Hasil Program dan Penjelasan

Tahapan Processing Image dengan metode berbasis tekstur menggunakan metode GLCM untuk menentukan 6 nilai fitur dari data image: dissimilarity, correlation, homogeneity, contrast, ASM, energy, dengan 5 sudut (0, 45, 90, 135, dan 180). Total ada 30 nilai fitur (6 x 5) untuk setiap data image. Kami menggunakan parameter nilai k yang ganjil (k=1, atau 3, atau 5, atau 7, atau 9). Dari 5 nilai parameter yang digunakan, hasil terbaik yang didapatkan adalah nilai parameter K = 1 memiliki akurasi yang paling tinggi sekitar 89.69%.

2.3 Tampilan Antar Muka Aplikasi

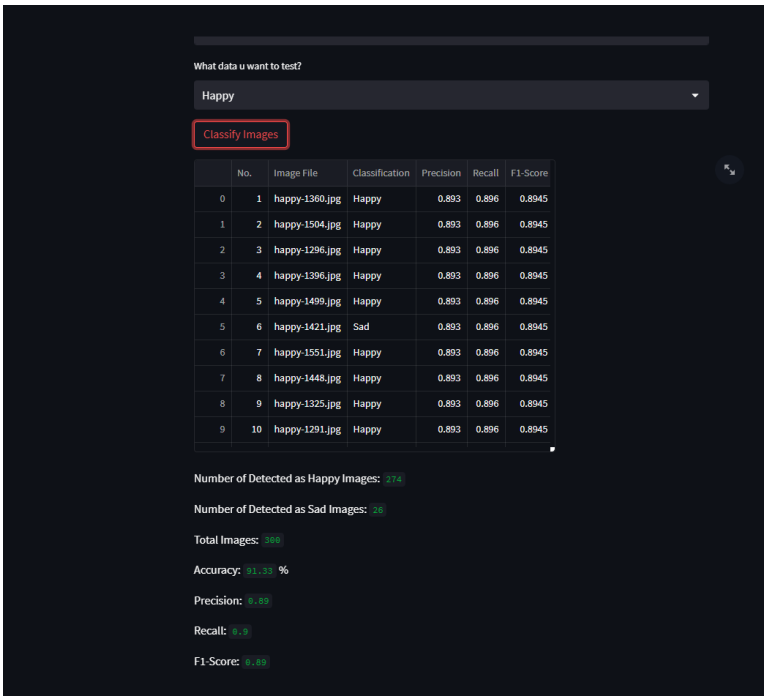
2.3.1 Halaman Utama



Gambar 3 Tampilan Awal Aplikasi

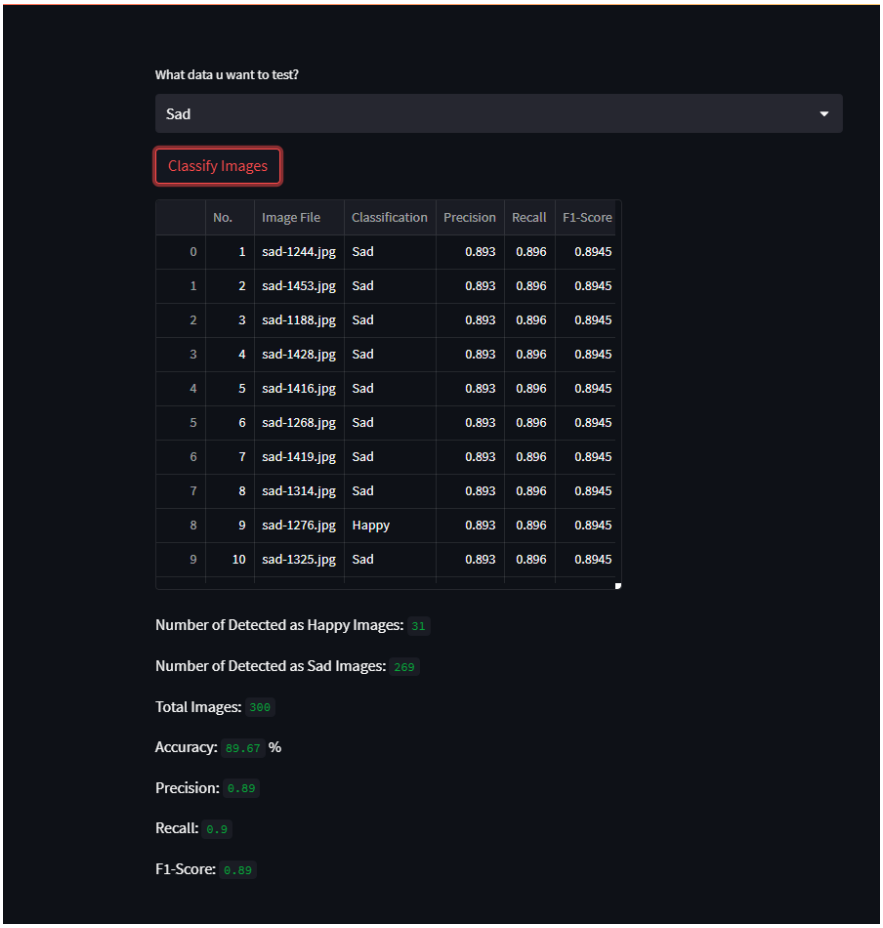
Tampilan awal program seperti gambar 3, pengguna dapat meng inpukan file gambar yang ingin di idenfikasi. Caranya pertama klik “Browser fie” kemudia pilih file gambar atau bisa melakukan drag and drop file gambar ke dalam kotak inputan, maksimal file yang di inputkan adalah 200 MB dengan format file JPG, JPEG atau PNG.

2.3.2 Perhitungan Tuning Hyper Parameter



	No.	Image File	Classification	Precision	Recall	F1-Score
0	1	happy-1360.jpg	Happy	0.893	0.896	0.8945
1	2	happy-1504.jpg	Happy	0.893	0.896	0.8945
2	3	happy-1296.jpg	Happy	0.893	0.896	0.8945
3	4	happy-1396.jpg	Happy	0.893	0.896	0.8945
4	5	happy-1499.jpg	Happy	0.893	0.896	0.8945
5	6	happy-1421.jpg	Sad	0.893	0.896	0.8945
6	7	happy-1551.jpg	Happy	0.893	0.896	0.8945
7	8	happy-1448.jpg	Happy	0.893	0.896	0.8945
8	9	happy-1325.jpg	Happy	0.893	0.896	0.8945
9	10	happy-1291.jpg	Happy	0.893	0.896	0.8945

Gambar 4 Perhitungan Tuning Hyper Parameter untuk Data Happy



Gambar 5 Perhitungan Tuning Hyper Parameter untuk Data Sad

Pada Gambar 4 , perhitungan tuning hyper parameter nya yaitu : Dari 300 data gambar Happy, terdapat 274 data gambar yang terdeteksi “Happy” dan 26 data gambar yang terdeteksi “Sad”.

Pada gambar 5, perhitungan tuning hyper Parameter nya yaitu : Dari 300 data gambar Sad, terdapat 31 data gambar yang terdeteksi “Happy” dan 269 data gambar yang terdeteksi “Sad” .

2.4 Implementasi (Coding) dan Penjelasannya

1. Import Library

```
import os
import glob
import numpy as np
from skimage.feature import graycomatrix, graycoprops
import cv2
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn import metrics
import joblib
import streamlit as st
```

Pada bagian import Library ini adalah proses import modul pustaka yang akan digunakan dalam program. Berikut penjelasan program :

- import os: Mengimpor modul os yang menyediakan fungsi-fungsi untuk berinteraksi dengan sistem operasi, seperti mengakses file dan direktori, menjalankan perintah system.
- import glob: Mengimpor modul glob yang digunakan untuk mencocokkan pola dan mencari file atau direktori dalam direktori tertentu.
- import numpy as np: Mengimpor modul numpy dan memberikan alias np. numpy adalah pustaka yang populer untuk komputasi numerik di Python.
- import pandas as pd: Mengimpor modul pandas dan memberikan alias pd. pandas adalah pustaka yang digunakan untuk manipulasi dan analisis data terstruktur, terutama dalam bentuk dataframe.
- from skimage.feature import graycomatrix, graycoprops: Mengimpor fungsi graycomatrix dan graycoprops dari modul skimage.feature. Modul skimage adalah modul dari pustaka scikit-image yang menyediakan berbagai alat pengolahan gambar.
- import cv2: Mengimpor modul cv2 yang merupakan OpenCV, pustaka populer untuk pengolahan citra dan komputer visi.
- from sklearn.model_selection import train_test_split: Mengimpor fungsi train_test_split dari modul sklearn.model_selection. Fungsi ini digunakan untuk membagi data menjadi set pelatihan dan pengujian dalam pembelajaran mesin.
- from sklearn.neighbors import KNeighborsClassifier: Mengimpor kelas KNeighborsClassifier dari modul sklearn.neighbors. Kelas ini digunakan untuk membangun model klasifikasi menggunakan algoritma K-Nearest Neighbors.
- from sklearn import metrics: Mengimpor modul metrics dari sklearn. Modul ini berisi berbagai metrik evaluasi yang digunakan untuk mengukur kinerja model pembelajaran mesin.

- import joblib: Mengimpor modul joblib yang digunakan untuk menyimpan dan memuat model yang dilatih.
- import streamlit as st: Mengimpor modul streamlit dan memberikan alias st. streamlit adalah kerangka kerja untuk membangun aplikasi web interaktif dengan mudah menggunakan Python.

2. Mendefinisikan beberapa variable data agar lebih mudah di proses

```
# Global variables
happy_dir = 'happy'
sad_dir = 'sad'
dataset_file = 'dataset.npy'
model_file = 'model.joblib'
```

Pada bagian #Global Variable ini ada proses mendefinisiskam beberapa variable data agar lebih mudah di proses. Berikut penjelasan programnya :

- happy_dir = 'happy': Variabel happy_dir adalah string yang menyimpan nama direktori yang mengandung data gambar yang dikategorikan sebagai "happy" atau senang.
- sad_dir = 'sad': Variabel sad_dir adalah string yang menyimpan nama direktori yang mengandung data gambar yang dikategorikan sebagai "sad" atau sedih.
- dataset_file = 'dataset.npy': Variabel dataset_file adalah string yang menyimpan nama file yang akan digunakan untuk menyimpan dataset yang telah diproses atau siap untuk digunakan. File ini menggunakan format .npy, yang merupakan format file Numpy untuk menyimpan array multidimensi.
- model_file = 'model.joblib': Variabel model_file adalah string yang menyimpan nama file yang akan digunakan untuk menyimpan model yang telah dilatih. File ini menggunakan format .joblib, yang merupakan format file yang digunakan oleh modul joblib untuk menyimpan model Python.

3. Menghitung fitur GLCM pada sebuah gambar

```
# Calculate GLCM features
def calculate_glcm_features(img):
    angles = [0, np.pi/4, np.pi/2, 3*np.pi/4, np.pi]
```

```
            properties = ['dissimilarity', 'correlation', 'homogeneity',
        'contrast', 'ASM', 'energy']
            glcm = graycomatrix(img, [1], angles, 256, symmetric=True,
        normed=True)

            features = []

            for prop in properties:
                feature = graycoprops(glcm, prop).ravel()
                features.append(feature)

                st.write(f"{prop}: {feature}")

    return np.concatenate(features)
```

Pada bagian # Calculate GLCM features ini adalah proses menghitung GLCM pada sebuah gambar. Berikut penjelasan programnya :

- angles = [0, np.pi/4, np.pi/2, 3*np.pi/4, np.pi]: Variabel angles adalah daftar sudut yang digunakan untuk menghitung matriks GLCM. Sudut-sudut ini menentukan arah di mana hubungan antara pasangan nilai piksel dianalisis.
- properties = ['dissimilarity', 'correlation', 'homogeneity', 'contrast', 'ASM', 'energy']: Variabel properties adalah daftar fitur GLCM yang akan dihitung. Fitur-fitur ini mencakup dissimilarity (ketidaksamaan), correlation (korelasi), homogeneity (homogenitas), contrast (kontras), ASM (Angular Second Moment), dan energy (energi).
- glcm = graycomatrix(img, [1], angles, 256, symmetric=True, normed=True): Menghitung matriks GLCM dari citra img dengan menggunakan sudut-sudut yang ditentukan dalam angles. graycomatrix adalah fungsi dari modul skimage.feature yang menghasilkan matriks GLCM dengan jarak 1, menggunakan 256 level keabuan, dengan simetri simetris, dan dinormalkan (normed=True).
- features = []: Membuat sebuah daftar kosong features untuk menyimpan hasil fitur-fitur GLCM.

- for prop in properties: ...: Melakukan iterasi melalui setiap fitur dalam properties.
- feature = graycoprops(glcm, prop).ravel(): Menghitung nilai fitur GLCM yang diberikan (prop) dari matriks GLCM glcm menggunakan fungsi graycoprops dari modul skimage.feature. Hasil fitur ini kemudian diubah menjadi array satu dimensi menggunakan metode ravel().
- features.append(feature): Menambahkan array fitur yang dihitung ke dalam daftar features.
- return np.concatenate(features): Menggabungkan semua array fitur dalam features menjadi satu array tunggal menggunakan fungsi np.concatenate. Array ini kemudian dikembalikan sebagai hasil fungsi.

4. Untuk melatih dataset gambar atau label yang digunakan

```
# Train the model
def train_model(k):
    st.write("Training Model...")

    # Load image files and labels
    happy_files = glob.glob(os.path.join(happy_dir, '*'))
    sad_files = glob.glob(os.path.join(sad_dir, '*'))
    files = happy_files + sad_files
    labels = [0] * len(happy_files) + [1] * len(sad_files)  # 0 for happy, 1
for sad
```

Pada bagian #Train the model dan #Load image files and labels ini adalah proses untuk melatih dataset atau label yang digunakan. Berikut penjelasan program :

- st.write("Training Model..."): Ini adalah pernyataan yang menampilkan teks "Training Model..." menggunakan modul streamlit yang telah diimpor sebelumnya. Fungsi write() dari modul streamlit digunakan untuk menampilkan teks atau objek lainnya di antarmuka aplikasi web yang dibangun menggunakan Streamlit.
- happy_files = glob.glob(os.path.join(happy_dir, '*')): Menggunakan fungsi glob.glob() dari modul glob, variabel happy_files akan

menyimpan daftar jalur lengkap (path) ke semua file di dalam direktori happy_dir yang diwakili dengan karakter '*' (yang berarti semua file).

- sad_files = glob.glob(os.path.join(sad_dir, '*')): Menggunakan fungsi glob.glob(), variabel sad_files akan menyimpan daftar jalur lengkap ke semua file di dalam direktori sad_dir
- files = happy_files + sad_files: Menggabungkan daftar jalur file happy_files dan sad_files menjadi satu daftar files yang berisi jalur lengkap ke semua file dalam direktori "happy" dan "sad".
- labels = [0] * len(happy_files) + [1] * len(sad_files): Membuat daftar labels yang mengandung label untuk setiap file dalam files. Setiap file dalam happy_files diberi label 0 (representasi "happy"), sedangkan setiap file dalam sad_files diberi label 1 (representasi "sad"). Jumlah elemen dalam daftar label akan sama dengan jumlah file di direktori "happy" ditambah jumlah file di direktori "sad".

5. Memproses gambar dan menghasilkan fitur GLCM serta melatih model KNN dan menyimpan model serta dataset yang telah diproses.

```
# Preprocess images and extract GLCM features
    features = [calculate_glcm_features(cv2.resize(cv2.imread(file, 0), (48, 48))) for file in files]

    # Ensure features and labels are of the same size
    features, labels = np.array(features), np.array(labels)
    assert features.shape[0] == labels.shape[0], "Mismatch in features and labels sizes"

    # Train KNN classifier
    X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = metrics.accuracy_score(y_test, y_pred)
    precision = metrics.precision_score(y_test, y_pred)
    recall = metrics.recall_score(y_test, y_pred)
```

```
            f1_score = metrics.f1_score(y_test, y_pred)

            st.write(f"Model accuracy: {accuracy*100:.2f}%")
            st.write(f"Precision: {precision:.2f}")
            st.write(f"Recall: {recall:.2f}")
            st.write(f"F1-Score: {f1_score:.2f}")

            # Save the model
            joblib.dump(model, model_file)
            st.write("Model saved!")

            # Save the dataset
            dataset = np.column_stack((features, labels))
            np.save(dataset_file, dataset)
    st.write("Dataset saved!")
```

Pada bagian table diatas ini merupakan proses gambar dan menghasilkan fitur GLCM serta melatih model KNN dan menyimpan model serta dataset yang telah diproses. Berikut penjelasan program :

- features = [calculate_glcm_features(cv2.resize(cv2.imread(file, 0), (48, 48))) for file in files]: Variabel features berisi hasil fitur GLCM yang diekstraksi dari setiap file dalam daftar files. Setiap file dibuka menggunakan cv2.imread(), diubah menjadi citra grayscale dengan 0 sebagai parameter kedua, kemudian diubah ukurannya menjadi (48, 48) menggunakan cv2.resize(). Selanjutnya, fitur GLCM diekstraksi menggunakan fungsi calculate_glcm_features() yang mengambil citra tersebut sebagai argumen.
- features, labels = np.array(features), np.array(labels): Variabel features dan labels dikonversi menjadi array Numpy menggunakan np.array(). Ini memastikan bahwa features dan labels memiliki format yang sesuai untuk proses pelatihan.
- assert features.shape[0] == labels.shape[0], "Mismatch in features and labels sizes": Pernyataan assert digunakan untuk memeriksa apakah jumlah baris dalam features dan labels sama. Jika jumlahnya tidak sama, maka pesan error "Mismatch in features and labels sizes" akan muncul.

- X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42): Data fitur dan label dibagi menjadi set pelatihan (X_train dan y_train) dan set pengujian (X_test dan y_test) menggunakan fungsi train_test_split() dari sklearn.model_selection. Set pengujian sebesar 20% dari data digunakan dalam pengujian, sementara 80% digunakan dalam pelatihan.
- model = KNeighborsClassifier(n_neighbors=k): Membuat objek model KNN (KNeighborsClassifier) dengan parameter n_neighbors yang ditentukan oleh k.
- model.fit(X_train, y_train): Melatih model KNN dengan menggunakan data pelatihan (X_train dan y_train) menggunakan metode fit().
- y_pred = model.predict(X_test): Memprediksi label kelas untuk data pengujian (X_test) menggunakan model yang dilatih dengan metode predict()
- accuracy = metrics.accuracy_score(y_test, y_pred): Menghitung akurasi prediksi model dengan membandingkan label yang sebenarnya (y_test) dengan label yang diprediksi (y_pred) menggunakan fungsi accuracy_score() dari sklearn.metrics.
- precision = metrics.precision_score(y_test, y_pred): Menghitung presisi prediksi model menggunakan fungsi precision_score() dari sklearn.metrics.
- recall = metrics.recall_score(y_test, y_pred): Menghitung recall prediksi model menggunakan fungsi recall_score() dari sklearn.metrics.
- f1_score = metrics.f1_score(y_test, y_pred): Menghitung F1-Score prediksi model menggunakan fungsi f1_score() dari sklearn.metrics
- st.write(f"Model accuracy: {accuracy*100:.2f}%"): Menampilkan akurasi model yang telah dilatih dalam format persentase menggunakan st.write() dari modul streamlit.
- st.write(f"Precision: {precision:.2f}"): Menampilkan presisi model dalam format desimal dengan dua angka di belakang koma menggunakan st.write().

- st.write(f"Recall: {recall:.2f}"): Menampilkan recall model dalam format desimal dengan dua angka di belakang koma menggunakan st.write().
- st.write(f"F1-Score: {f1_score:.2f}"): Menampilkan F1-Score model dalam format desimal dengan dua angka di belakang koma menggunakan st.write().
- joblib.dump(model, model_file): Menyimpan model yang telah dilatih ke dalam file dengan nama yang ditentukan menggunakan joblib.dump() dari modul joblib.
- st.write("Model saved!"): Menampilkan pesan "Model saved!" menggunakan st.write()
- dataset = np.column_stack((features, labels)): Menggabungkan array fitur features dan array label labels secara horizontal menggunakan np.column_stack() untuk membentuk dataset yang berisi fitur-fitur dan label-label yang sesuai.
- np.save(dataset_file, dataset): Menyimpan dataset ke dalam file dengan nama yang ditentukan menggunakan np.save() dari modul numpy
- st.write("Dataset saved!"): Menampilkan pesan "Dataset saved!" menggunakan st.write().

6. Untuk menapilkan Halaman Utama

```
# Streamlit UI - Home Page
  def home():
      st.title('Happy or Sad Image Classifier')

  # Check if model and dataset files exist
      model_exists = os.path.exists(model_file)
      dataset_exists = os.path.exists(dataset_file)

  # Add "Train Model" button
      if st.button("Train Model"):
          st.session_state.page = 'Train'

    # Image classification
```

```
        image_folder = st.text_input("Enter the path to the image folder:")
        now = st.selectbox("What data u want to test?", ["Happy", "Sad"])
        if st.button("Classify Images"):
            if not os.path.exists(image_folder):
                st.write("Invalid image folder path!")
                return

    # Load the model
            if model_exists:
                model = joblib.load(model_file)
    # Classify images and create a dataframe
            results = []
            for idx, file in enumerate(image_files):
                img = cv2.resize(cv2.imread(file, 0), (48, 48))
                features = calculate_glcm_features(img)
                prediction = model.predict([features])[0]
                result = {
                    "No.": idx + 1,
                    "Image File": os.path.basename(file),
                    "Classification": "Happy" if prediction == 0 else "Sad"
                }
        # Calculate metrics
        if dataset_exists:
            dataset = np.load(dataset_file)
            X_test, y_test = dataset[:, :-1], dataset[:, -1]
            y_pred = model.predict(X_test)
            precision = metrics.precision_score(y_test, y_pred)
            recall = metrics.recall_score(y_test, y_pred)
            f1_score = metrics.f1_score(y_test, y_pred)
            result["Precision"] = precision
             result["Recall"] = recall
              result["F1-Score"] = f1_score

results.append(result)
```

```
            df = pd.DataFrame(results)
            st.write(df)

        # Count the number of happy and sad images
        count = df['Classification'].value_counts()
        st.write("Number of Detected as Happy Images:", count["Happy"])
        st.write("Number of Detected as Sad Images:", count["Sad"])
        st.write("Total Images:", len(image_files))

        if now == "Happy":
            st.write("Accuracy:", round(count["Happy"] / len(image_files) * 100, 2),
        "%")
        else:
            st.write("Accuracy:", round(count["Sad"] / len(image_files) * 100, 2), "%")

        st.write("Precision:", round(df["Precision"].mean(), 2))
        st.write("Recall:", round(df["Recall"].mean(), 2))
        st.write("F1-Score:", round(df["F1-Score"].mean(), 2))
```

Pada bagian table di atas merupakan proses untuk menampilkan halaman utama. Berikut penjelasan program :

- st.title('Happy or Sad Image Classifier'): Menampilkan judul "Happy or Sad Image Classifier" menggunakan st.title() dari modul streamlit.
- model_exists = os.path.exists(model_file): Memeriksa apakah file model (model_file) ada di sistem menggunakan fungsi os.path.exists() dari modul os. Hasilnya disimpan dalam variabel model_exists.
- dataset_exists = os.path.exists(dataset_file): Memeriksa apakah file dataset (dataset_file) ada di sistem menggunakan os.path.exists(). Hasilnya disimpan dalam variabel dataset_exists.
- if st.button("Train Model"): ...: Menampilkan tombol "Train Model" menggunakan st.button() dari modul streamlit. Jika tombol ditekan, maka variabel st.session_state.page akan diatur menjadi 'Train'. Ini

memberi sinyal untuk mengarahkan aplikasi ke halaman pelatihan model.

- image_folder = st.text_input("Enter the path to the image folder:"): Meminta pengguna memasukkan jalur folder gambar yang ingin diklasifikasikan menggunakan st.text_input().
- now = st.selectbox("What data u want to test?", ["Happy", "Sad"]): Menampilkan kotak pilihan yang memungkinkan pengguna memilih jenis data yang ingin diuji (Happy atau Sad) menggunakan st.selectbox().
- if st.button("Classify Images"): ...: Menampilkan tombol "Classify Images" menggunakan st.button(). Jika tombol ditekan, maka langkah-langkah berikutnya akan dijalankan untuk mengklasifikasikan gambar-gambar dalam folder yang dimasukkan oleh pengguna.
- if not os.path.exists(image_folder): ...: Memeriksa apakah folder gambar yang dimasukkan oleh pengguna (image_folder) ada di sistem menggunakan os.path.exists(). Jika folder tidak valid, maka akan ditampilkan pesan "Invalid image folder path!" menggunakan st.write().
- image_files = glob.glob(os.path.join(image_folder, '*')): Menggunakan glob.glob(), variabel image_files akan menyimpan daftar jalur lengkap ke semua file di dalam folder gambar yang dimasukkan oleh pengguna.
- if len(image_files) == 0: ...: Memeriksa apakah tidak ada gambar yang ditemukan dalam folder yang dimasukkan oleh pengguna. Jika tidak ada gambar, maka akan ditampilkan pesan "No images found in the specified folder!" menggunakan st.write().
- if model_exists: ...: Memeriksa apakah file model (model_file) ada di sistem. Jika ada, maka model akan dimuat menggunakan joblib.load() dan disimpan dalam variabel model.
- results = []: Membuat sebuah daftar kosong results untuk menyimpan hasil klasifikasi dan metrik untuk setiap gambar.
- for idx, file in enumerate(image_files): ...: Melakukan iterasi melalui setiap file dalam image_files untuk melakukan klasifikasi dan menghitung metrik.

- img = cv2.resize(cv2.imread(file, 0), (48, 48)): Membaca gambar menggunakan cv2.imread() dan mengubahnya menjadi citra grayscale dengan parameter kedua 0. Selanjutnya, citra tersebut diubah ukurannya menjadi (48, 48) menggunakan cv2.resize(), dan disimpan dalam variabel img.
- features = calculate_glcm_features(img): Menghitung fitur GLCM dari citra menggunakan fungsi calculate_glcm_features() yang mengambil img sebagai argumen. Fitur-fitur ini disimpan dalam variabel features.
- prediction = model.predict([features])[0]: Melakukan prediksi kelas untuk fitur features menggunakan model yang telah dimuat. Prediksi ini disimpan dalam variabel prediction.
- result = { ... }: Membuat sebuah objek result yang berisi nomor gambar, nama file gambar, dan klasifikasinya berdasarkan prediksi.
- if dataset_exists: ...: Memeriksa apakah file dataset (dataset_file) ada di sistem. Jika ada, maka akan dihitung metrik (precision, recall, dan F1-Score) dengan membandingkan prediksi dengan label sebenarnya menggunakan dataset yang telah diproses.
- results.append(result): Menambahkan objek result ke dalam daftar results.
- df = pd.DataFrame(results): Membentuk dataframe dari daftar results menggunakan pd.DataFrame() dari modul pandas. DataFrame ini akan berisi hasil klasifikasi dan metrik untuk setiap gambar.
- st.write(df): Menampilkan dataframe hasil klasifikasi dan metrik menggunakan st.write().
- count = df['Classification'].value_counts(): Menghitung jumlah gambar yang terdeteksi sebagai "Happy" dan "Sad" dalam dataframe menggunakan value_counts().
- st.write("Number of Detected as Happy Images:", count["Happy"]): Menampilkan jumlah gambar yang terdeteksi sebagai "Happy" menggunakan st.write().
- st.write("Number of Detected as Sad Images:", count["Sad"]): Menampilkan jumlah gambar yang terdeteksi sebagai "Sad"

7. Untuk menampilakan halaman proses pada nilai K yang digunakan

```
# Streamlit UI - Train Page
        def train():
            st.title('Train Model')

            # Add "Back" button
            if st.button("Back"):
                st.session_state.page = 'Home'

            # Select K value
            k = st.selectbox("Select K value", [1, 3, 5, 7, 9])

            # Train the model
            if st.button("Train", key="train_button"):
      train_model(k)

# Streamlit App
        def main():
            # Initialize Streamlit session state
            if 'page' not in st.session_state:
                st.session_state.page = 'Home'

            # Page navigation
            if st.session_state.page == 'Home':
                home()
            elif st.session_state.page == 'Train':
                train()

        if __name__ == "__main__":
  main()
```

Pada bagian table diatas merupakan proses untuk menampilkan halaman proses nilai K yang digunakan. Berikut penjelasan program :

- def train(): ...: Fungsi train() merupakan halaman pelatihan model yang ditampilkan saat pengguna menekan tombol "Train Model" di halaman utama. Ini menampilkan antarmuka pengguna untuk pelatihan model.

- st.title('Train Model'): Menampilkan judul "Train Model" menggunakan st.title() dari modul streamlit.
- if st.button("Back"): ...: Menampilkan tombol "Back" menggunakan st.button(). Jika tombol ditekan, maka variabel st.session_state.page akan diatur kembali ke 'Home', mengarahkan aplikasi kembali ke halaman utama.
- k = st.selectbox("Select K value", [1, 3, 5, 7, 9]): Menampilkan kotak pilihan yang memungkinkan pengguna memilih nilai K untuk model KNN yang akan dilatih menggunakan st.selectbox().
- if st.button("Train", key="train_button"): ...: Menampilkan tombol "Train" menggunakan st.button(). Jika tombol ditekan, maka fungsi train_model(k) akan dipanggil untuk melatih model dengan nilai K yang dipilih.
- def main(): ...: Fungsi main() adalah fungsi utama yang mengatur navigasi halaman dan inisialisasi sesi Streamlit.
- if 'page' not in st.session_state: ...: Memeriksa apakah variabel page belum ada di dalam st.session_state. Jika belum ada, maka variabel tersebut diinisialisasi dengan nilai 'Home'.
- if st.session_state.page == 'Home': ...: Jika page dalam st.session_state adalah 'Home', maka halaman utama (home()) akan ditampilkan.
- elif st.session_state.page == 'Train': ...: Jika page dalam st.session_state adalah 'Train', maka halaman pelatihan (train()) akan ditampilkan.
- if __name__ == "__main__": ...: Bagian ini akan menjalankan fungsi main() saat script dieksekusi secara langsung.

2.5 Tuning Hyper Parameter

Dalam tuning hyper-parameter, disini akan dilakukan tes pengujian per satu dari 300 data happy dan 300 data sad. Untuk pengujian akan dilakukan secara berurutan dari K=1 sampai K=9, penjelasan lebih lengkapnya terdapat pada video presentasi. Dapat dilihat hasilnya sebagai berikut:

1. Hasil dari tuning hyper-parameter untuk K=1:

K = 1	**Happy**	**Sad**
Accuracy	91.33 %	89.67 %
Precision	0.89	0.89
Recall	0.9	0.9
F1-Score	0.89	0.89

2. Hasil dari tuning hyper-parameter untuk K=3:

K = 3	**Happy**	**Sad**
Accuracy	72.67 %	65.33 %
Precision	0.7	0.7
Recall	0.69	0.69
F1-Score	0.69	0.69

3. Hasil dari tuning hyper-parameter untuk K=5:

K = 5	**Happy**	**Sad**
Accuracy	74.33 %	64.33 %
Precision	0.67	0.67
Recall	0.66	0.66
F1-Score	0.66	0.66

4. Hasil dari tuning hyper-parameter untuk K=7:

K = 7	Happy	Sad
Accuracy	72.33 %	59.0 %
Precision	0.64	0.64
Recall	0.62	0.62
F1-Score	0.63	0.63

5. Hasil dari tuning hyper-parameter untuk K=9:

K = 9	Happy	Sad
Accuracy	73.67 %	58.67 %
Precision	0.64	0.64
Recall	0.61	0.61
F1-Score	0.63	0.63

Setelah dilakukan pengujian tuning hyper-parameter dan mendapatkan hasil seperti diatas, maka dapat kita simpulkan untuk hasil dari tuning hyper-parameter sebagai berikut:

	K = 1	K = 3	K = 5	K = 7	K = 9
Accuracy	90,47 %	69%	69,33%	65 %	66,17 %
Precision	0.89	0.7	0.67	0.64	0,64
Recall	0.9	0.69	0.66	0,62	0,61
F1-Score	0.89	0.69	0.66	0,63	0,63

Dari hasil tuning parameter diatas, dapat dilihat bahwa untuk hasil terbaiknya didapatkan oleh K=1 dengan akurasi 90,47%, kemudian untuk hasil moderate didapatkan oleh K=3 dengan akurasi 69% dan untuk hasil terjelek didapatkan oleh K=7 dengan akurasi 65%.

BAB III
PENUTUP

3.1 Kesimpulan

Dengan melakukan langkah-langkah preprocessing menggunakan, algoritma K-NN dapat memberikan hasil klasifikasi yang lebih baik pada data gambar. Preprocessing membantu dalam mengurangi keberisikan, menormalisasi data, dan mengekstraksi fitur yang relevan, sehingga meningkatkan kemampuan algoritma untuk memahami dan mengklasifikasikan gambar dengan lebih baik. Dari 5 nilai parameter yang digunakan, hasil yang didapatkan adalah nilai parameter K = 1 memiliki akurasi yang paling tinggi sekitar 89.69%.

DAFTAR PUSTAKA

[1] d. Sutayo, *Teori Pengolahan Citra Digital. Andi. Yogyakarta,* 2009.

[2] A. Usman, *Pengolahan Citra Digital & Teknik Pemrogramnnya,* 2005.

[3] S. Indra, *Simulasi Pengaturan Lalu lintas Bedasarkan Data Image Processing Kepadatan Kendaraan Berbasis Mikrokontroler Atmega 16 ,* 2014.

[4] X. &. K. Wu, *The Top Ten Algorithms in Data Mining,* 2009.

[5] B. Avelita, *Klasifikasi_K-Nearest_Neighbor,* 2016.

[6] AnbiDev, *Belahar Machine Learning - Image Processing ,* 2020.

[7] A. W. W. A. S. Restu Widodo, *Pemanfaatan Ciri Gray Level Co-Occurrence Matrix (GLCM) Citra Buah Jeruk Keprok (Citrus reticulata Blanco) untuk Klasifikasi Mutu,* vol. 2, 2018.

[8] Trivusi, *Yuk Kenali Apa itu Algoritma K-Nearest Neighbors (KNN) ,* 2022.