FINAL PROJECT

PENGANTAR PEMROSESAN DATA MULTIMEDIA

Disusun Oleh Kelompok 4B:

Matthew Novan Sidharta (2108561012)

Anak Agung Ngurah Mahadana Apta Gotra (2108561066)

I Gusti Agung Gede Ary Mahayasa (2108561087)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS UDAYANA

2023

BAB I
PENDAHULUAN

1.1 Latar Belakang

Teknologi terus mengalami perkembangan dari masa ke masa. Pengaruhnya dalam kehidupan sehari-hari juga semakin besar. Mulai dari pekerjaan kantor, pembelajaran di sekolah, transaksi, belanja barang, dan masih banyak yang lainnya. Teknologi yang paling dekat dengan masyarakat global tentunya adalah smartphone, laptop, dan komputer. Jika berbicara terkait komputer, maka kita akan mengetahui bahwa ada sekumpulan sistem yang membangunnya. Sistem merupakan sekumpulan komponen yang dihubungkan untuk mempermudah aliran informasi demi mencapai suatu tujuan (Mulachela, 2022). Sistem dalam komputer dan/atau aplikasi memiliki begitu banyak variasi tergantung dari metode atau algoritma penyusunnya. Algoritma sendiri merupakan sederet aturan, tata cara, dan panduan yang dapat digunakan untuk memecahkan sebuah masalah dalam sistem atau aplikasi (Hidayati, 2022). Algoritma sendiri memiliki banyak varian, salah satunya adalah k-nearest neighbor (KNN). KNN bekerja dengan cara mengklasifikasikan data berdasarkan similarity atau kemiripan atau kedekatannya terhadap data lainnya (Afifah, 2020).

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

Berdasarkan problem komputasi diatas, maka terdapat beberapa tujuan sebagai berikut:

1. Mengimplementasikan teknik-teknik *pre-processing image* menggunakan Python.
2. Mengimplementasikan GLCM menggunakan Python.

1.4 Manfaat

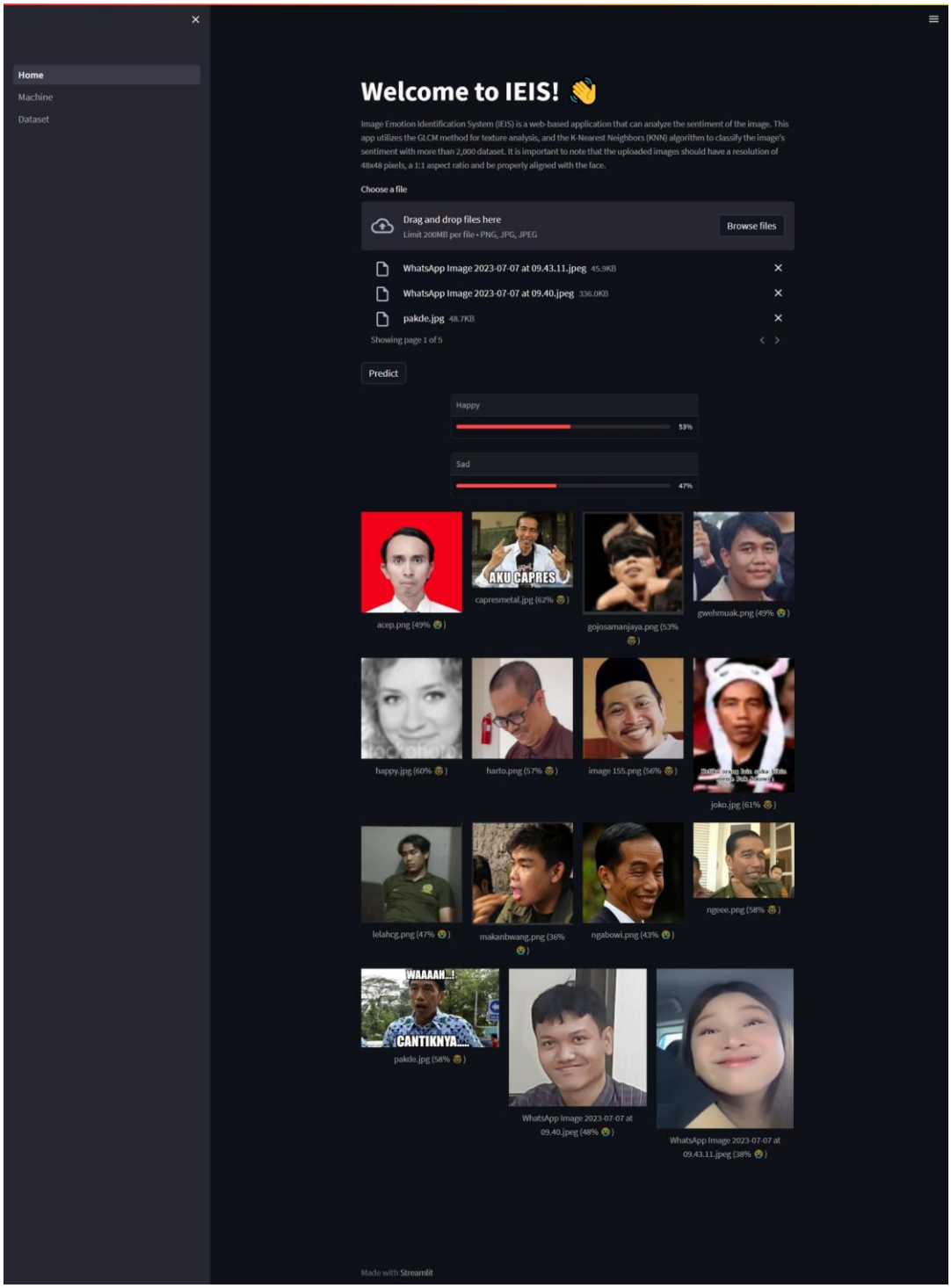
Berdasarkan tujuan diatas, maka terdapat beberapa manfaat yang kita dapatkan sebagai berikut:

1. Memahami cara mengimplementasikan teknik-teknik *pre-processing image* menggunakan Python.
2. Memahami cara mengimplementasikan GLCM menggunakan Python.

BAB II

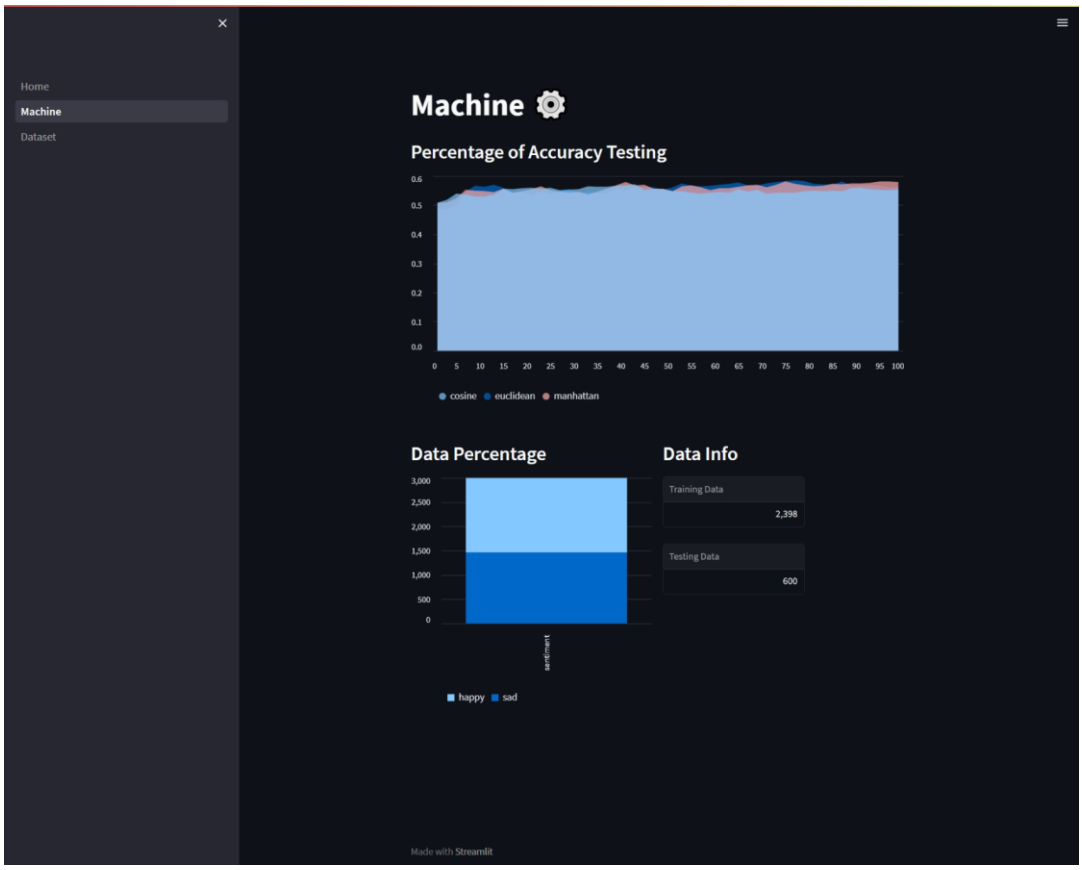
ISI

2.1 Manual Aplikasi



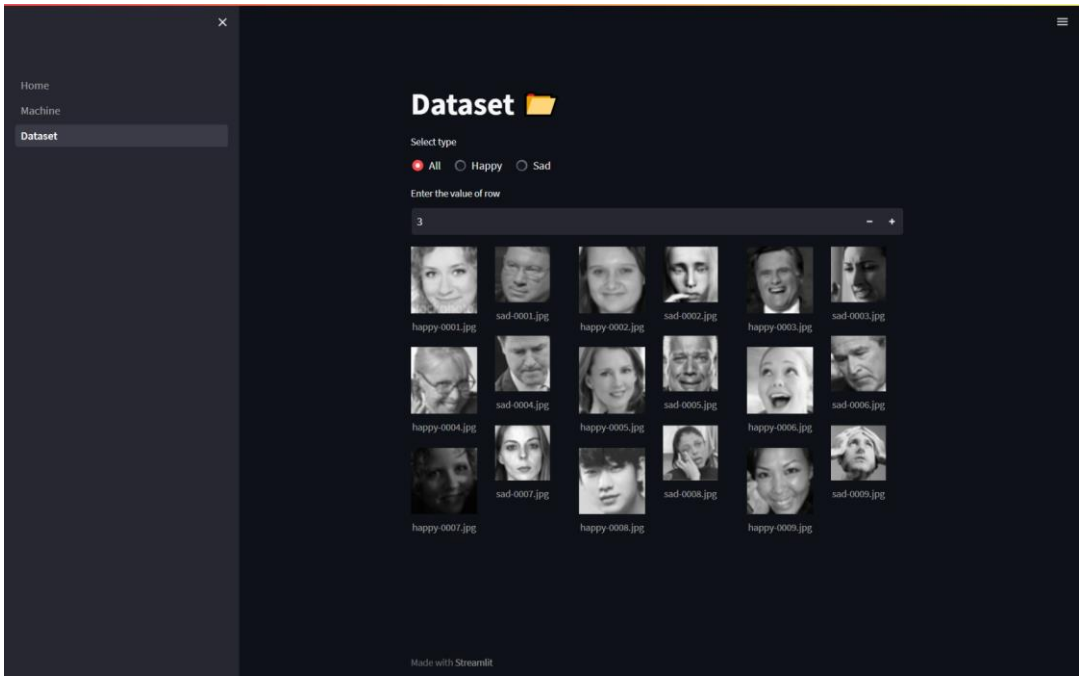
Gambar 1 Home Page

Gambar di atas merupakan tampilan ketika program sudah melakukan klasifikasi. Terdapat kolom persentase happy dan sad yang merupakan representasi dari keseluruhan gambar happy yang diiinputan dan juga keseluruhan gambar sad yang diinputkan. Di sebelah nama gambar terdapat persentase yang menunjukan tingkat happy ataupun sad dari gambar tersebut. Serta terdapat emoji pada nama gambar untuk menginformasikan sentiment dari gambar tersebut (happy = 😳, sad = 😭).



Gambar 2 Machine Page

Gambar di atas merupakan tampilan dari halaman yang memberikan informasi seputar komponen-komponen pembangun program, seperti nilai k, rumus jarak, jumlah data set, jumlah data testing, dan jumlah data training.



Gambar 3 Dataset Page

Gambar di atas merupakan tampilan halaman yang memberikan informasi tentang dataset yang digunakan dalam program. Tampilannya dapat diatur berdasarkan kategori ataupun jumlah yang ingin di tampilkan.

2.2 Source Code

Program ini dibangun dalam 5 file inti yang digunakan untuk menjalankan aplikasinya. File “model.ipynb” berfungsi untuk membuat model-model KNN. File “Backend.py” berisi source code untuk menyiapkan gambar sebelum diproses, menghitung GLCM-nya dan menganalisis sentimennya. Sedangkan, file “Home.py”, “1_Machine.py”, dan “2_Dataset.py” berisi source code untuk menampilkan seluruh informasi agar dapat dilihat oleh pengguna, dengan kata lain front-end. Berikut adalah penjelasan tiap filenya secara lebih rinci.

1. model.ipynb

```
from skimage.feature import graycomatrix, graycoprops

# Hitung fitur GLCM
def compute_glcm(image, angles):
    glcm = graycomatrix(image, distances=[1], angles=angles,
levels=256, symmetric=True, normed=True)
    return glcm

import numpy as np
```

```
# Hitung matriks GLCM
def glcm_matrix(image):
    matrix = []
    angles = [0, np.pi/4, np.pi/2, 3*np.pi/4, np.pi]
    metric_texture = ['dissimilarity', 'correlation',
'homogeneity', 'contrast', 'ASM', 'energy']
    for i in metric_texture:
        row = []
        for j in angles:
            row.append(graycoprops(compute_glcm(image, [j]),
prop=i)[0][0])
        matrix.append(row)
    return np.array(matrix).flatten()
```

Kode di atas berfungsi untuk menghitung GLCM dari *dataset*.

```
X = []  # Features
y = []  # Labels

import os
from skimage import io, color, util

# Load citra dengan ekspresi happy
positive_images = os.listdir("dataset/happy/")
for img_path in positive_images:
    image = io.imread("dataset/happy/" + img_path)
    features = glcm_matrix(image)
    X.append(features)
    y.append(1)  # Sentimen positif

# Load citra dengan ekspresi sad
negative_images = os.listdir("dataset/sad/")
for img_path in negative_images:
    image = io.imread("dataset/sad/" + img_path)
    features = glcm_matrix(image)
    X.append(features)
    y.append(0)  # Sentimen negatif
```

Kode di atas berfungsi untuk membaca seluruh *dataset* dan menghitung GLCM-nya lalu memasukannya ke dalam *list*.

```
from sklearn.model_selection import train_test_split

# Pisahkan data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Kode di atas berfungsi untuk membuat data *training* dan data *testing* (80%, 20%).

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import precision_score, recall_score,
f1_score
```

```
import pandas as pd

# Melatih klasifikasi KNN dengan nilai k dan metrik jarak
yang berbeda
k_values = list(range(1, 100, 2))  # Percobaan nilai k
distance_metrics = ['euclidean', 'manhattan', 'cosine']  #
Percobaan rumus jarak

results = []
accuracy_data = []

for distance_metric in distance_metrics:
    row = []
    for k in k_values:
        knn = KNeighborsClassifier(n_neighbors=k,
metric=distance_metric).fit(X_train, y_train)
        y_pred = knn.predict(X_test)
        accuracy = knn.score(X_test, y_test)
        precision = precision_score(y_test, y_pred)
        recall = recall_score(y_test, y_pred)
        f1 = f1_score(y_test, y_pred)
        row.append(accuracy)

        results.append({
            'k': k,
            'distance_metric': distance_metric,
            'accuracy': accuracy,
            'precision': precision,
            'recall': recall,
            'f1_score': f1
        })

    accuracy_data.append(row)

num_moderate = results
num_moderate.sort(key=lambda x: x['accuracy'])
moderate_index = (len(num_moderate) - 1) // 2
moderate_result = results[moderate_index]

best_index = max(range(len(results)), key=lambda i:
results[i]['accuracy'])
best_result = results[best_index]

worst_index = min(range(len(results)), key=lambda i:
results[i]['accuracy'])
worst_result = results[worst_index]

data = {
    'The Best': [best_result['k'],
best_result['distance_metric'], round(best_result['accuracy']
* 100),
                 round(best_result['precision'], 2),
round(best_result['recall'], 2),
round(best_result['f1_score'], 2)],
```

```
    'The Moderate': [moderate_result['k'],
moderate_result['distance_metric'],
round(moderate_result['accuracy'] * 100),
                round(moderate_result['precision'], 2),
round(moderate_result['recall'], 2),
round(moderate_result['f1_score'], 2)],
    'The Worst': [worst_result['k'],
worst_result['distance_metric'],
round(worst_result['accuracy'] * 100),
                round(worst_result['precision'], 2),
round(worst_result['recall'], 2),
round(worst_result['f1_score'], 2)]
}
df = pd.DataFrame(data, index=['k', 'Distance Metric',
'Accuracy', 'Precision', 'Recall', 'F1-Score'])
df
```

Kode di atas berfungsi untuk membuat model berdasarkan nilai k dan rumus jarak.

```
import joblib

# Menyimpan model
best_model =
KNeighborsClassifier(n_neighbors=best_result['k'],
metric=best_result['distance_metric']).fit(X_train, y_train)
joblib.dump(best_model, "best_model")

df = pd.DataFrame(accuracy_data, columns=k_values,
index=distance_metrics)
df.to_csv('models.csv', index=False)
df
```

Kode di atas berfungsi untuk meyimpan seluruh akurasi yang telah di cari dan juga menyimpan model terbaik.

```
import matplotlib.pyplot as plt

for i, distance_metric in enumerate(distance_metrics):
    plt.plot(k_values, accuracy_data[i],
label=distance_metric)

plt.xlabel('k')
plt.ylabel('Accuracy (%)')
plt.legend()
plt.show()
```

Kode di atas berfungsi untuk menampilkan grafik akurasi dari setiap model yang telah di buat

2. Backend.py

```
import os
import joblib
import numpy as np
from skimage import io, color, util, transform
from skimage.feature import graycomatrix, graycoprops
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

def initialization(image):
    image = io.imread(image)
    # Periksa jika gambar tidak memiliki resolusi 48x48
    if (image.shape[0] != 48) or (image.shape[1] != 48):
        image = transform.resize(image, (48, 48))
    # Periksa jika gambar memiliki saluran Alpha
    if len(image.shape) == 3:
        image = image[:, :, :3]
        image = color.rgb2gray(image)
    image = util.img_as_ubyte(image)
    return image
```

Kode di atas berfungsi untuk menyiapkan gambar sebelum di proses. Seperti mengatur ukurannya dan mengubah warnanya menjadi *grayscale*.

```
# Hitung fitur GLCM
def compute_glcm(image, angles):
    glcm = graycomatrix(image, distances=[1], angles=angles,
levels=256, symmetric=True, normed=True)
    return glcm

# Hitung matriks GLCM
def glcm_matrix(image):
    matrix = []
    angles = [0, np.pi/4, np.pi/2, 3*np.pi/4, np.pi]
    metric_texture = ['dissimilarity', 'correlation',
'homogeneity', 'contrast', 'ASM', 'energy']
    for i in metric_texture:
        row = []
        for j in angles:
            row.append(graycoprops(compute_glcm(image, [j]),
prop=i)[0][0])
        matrix.append(row)
    return np.array(matrix).flatten()
```

Kode di atas berfungsi untuk menghitung GLCM dengan 6 fitur dan 5 sudut.

```
# Prediksi model
def model_predict(features):
    model = joblib.load("best_model")
    prediction = model.predict([features])
    if prediction[0] == 1:
        positive_proba =
model.predict_proba([features])[0][1]
        percentage = positive_proba * 100
    else:
```

```
        negative_proba =
model.predict_proba([features])[0][0]
        percentage = (1 - negative_proba) * 100
    return prediction[0], round(percentage)
```

Kode di atas berfungsi untuk memprediksi sentimen gambar, berdasarkan model terbaik yang telah di buat.

```
def get_sentiment(image):
    image = initialization(image)
    new_features = glcm_matrix(image)
    prediction, percentage = model_predict(new_features)
    return prediction, percentage
```

Kode di atas berfungsi untuk memanggil semua fungsi sehingga seluruh proses untuk mendapatkan sentimen gambarnya berjalan.

3. Home.py

```
import streamlit as st
import joblib
import pandas as pd
import Backend as be

st.set_page_config(page_title="IEIS", page_icon="🖼️")
st.title("Welcome to IEIS! 👋")
st.caption("Image Emotion Identification System (IEIS) is a
web-based application that can analyze the sentiment of the
image. This app utilizes the GLCM method for texture
analysis, and the K-Nearest Neighbors (KNN) algorithm to
classify the image's sentiment with more than 2,000 dataset.
It is important to note that the uploaded images should have
a resolution of 48x48 pixels, a 1:1 aspect ratio and be
properly aligned with the face.")
```

Kode di atas berfungsi untuk konfigurasi halaman seperti penentuan judul halaman, ikon halaman, dan juga untuk menyediakan area mengunggah file gambar.

```
if len(uploaded_file) == 0:
    button = st.button("Predict", disabled=True)
```

Kode diatas berfungsi untuk mengunci tombol “predict” pada saat pengguna belum melakukan *upload file.*

```
else:
    button = st.button("Predict", disabled=False)

    if button:
```

```
        sentiment = []
        emoji = []

        for image in uploaded_file:
            analysis = be.get_sentiment(image)
            sentiment.append(analysis[0])
            if analysis[0] == 1:
                emoji.append(f"({analysis[1]}% 🥹)")
            else:
                emoji.append(f"({analysis[1]}% 😭)")
```

Kode di atas berfungsi untuk melakukan prediksi pada setiap gambar yang diunggah ketika pengguna menekan tombol “predict”. Hasil prediksi sentimen disimpan dalam daftar sentiment, dan dilakukan penentuan emoji yang sesuai berdasarkan hasil prediksi.

```
        happy = sentiment.count(1)
        sad = sentiment.count(0)
        happy_percentage = (happy / len(sentiment)) * 100
        sad_percentage = (sad / len(sentiment)) * 100

        col1, col2, col3 = st.columns([1, 3, 1])
        with col2:

            data_df1 = pd.DataFrame({"happy":
[happy_percentage]})
            st.data_editor(
                data_df1,
                column_config={
                    "happy": st.column_config.ProgressColumn(
                        label="Happy",
                        help="The number of happy images",
                        width="large",
                        format="%.0f%%",
                        min_value=0,
                        max_value=100,
                    ),
                },
                disabled=True,
                hide_index=True,
            )

            data_df2 = pd.DataFrame({"sad":
[sad_percentage]})
            st.data_editor(
                data_df2,
                column_config={
                    "sad": st.column_config.ProgressColumn(
                        label="Sad",
                        help="The number of sad images",
                        width="large",
                        format="%.0f%%",
                        min_value=0,
                        max_value=100,
```

```
                    ),
                },
                disabled=True,
                hide_index=True,
            )

        num_images = len(uploaded_file)
        max_columns = 4

        for i in range(0, num_images, max_columns):
            cols = st.columns(min(num_images - i,
max_columns))
            for j in range(min(num_images - i, max_columns)):
                with cols[j]:
                    st.image(
                        uploaded_file[i + j],
                        caption=uploaded_file[i + j].name + "
" + emoji[i + j],
                        use_column_width=True
                    )
```

Kode di atas berfungsi untuk menampilkan gambar-gambar yang diunggah beserta caption yang terdiri dari nama gambar dan emoji yang sesuai.

4. 1_Machine.py

```
import streamlit as st
import re
import os
import joblib
import pandas as pd

st.set_page_config(page_title="IEIS - Machine",
page_icon="")
st.title("Machine ")
```

Kode di atas berfungsi untuk konfigurasi halaman seperti penentuan judul halaman dan ikon halaman.

```
def real_k():
    st.subheader('Percentage of Accuracy Testing')
    k_values = list(range(1, 100, 2))
    distance_metrics = ['euclidean', 'manhattan', 'cosine']
    df = pd.read_csv('models.csv').T
    chart_data = pd.DataFrame(df.values,
columns=distance_metrics, index=k_values)
    st.area_chart(chart_data)
```

Kode di atas berfungsi untuk menampilkan grafik area yang menunjukkan persentase akurasi pengujian model yang telah dibuat.

```
def fixed_data():
    positive_images = os.listdir("dataset/happy/")
    negative_images = os.listdir("dataset/sad/")
    col1, col2 = st.columns(2)
    with col1:
        st.subheader('Data Percentage')
        chart_data = pd.DataFrame(
            {'happy': [len(positive_images)],
            'sad': [len(negative_images)]},
            index=['sentiment']
        )
        st.bar_chart(chart_data)
    with col2:
        st.subheader('Data Info')
        all_data = len(positive_images) +
len(negative_images)
        data_train = round(all_data * 0.8)
        data_test = round(all_data * 0.2)
        data_df1 = pd.DataFrame({"train": [data_train]})
        data_df2 = pd.DataFrame({"test": [data_test]})
        st.data_editor(
            data_df1,
            column_config={
                "train": st.column_config.NumberColumn(
                    label="Training Data",
                    help="The number of data training",
                    width="medium",
                )
            },
            disabled=True,
            hide_index=True,
        )
        st.data_editor(
            data_df2,
            column_config={
                "test": st.column_config.NumberColumn(
                    label="Testing Data",
                    help="The number of data testing",
                    width="medium",
                ),
            },
            disabled=True,
            hide_index=True,
        )
```

Kode di atas berfungsi untuk menampilkan informasi terkait data yang digunakan dalam model seperti, jumlah *dataset happy* dan *sad*, jumlah *data training* dan *data testing*.

```
real_k()
fixed_data()
```

Kode diatas berfungsi untuk memanggil fungsi real_k() dan fixed_data() dipanggil dan menampilkan visualisasi dan informasi terkait data pada aplikasi *web*.

5. 2_Dataset.py

```
import streamlit as st
import re
import os
import math
import pandas as pd

st.set_page_config(page_title="IEIS - Dataset",
page_icon="")
st.title("Dataset ")
```

Kode di atas berfungsi untuk konfigurasi halaman seperti penentuan judul halaman dan ikon halaman.

```
image_folder_happy = "dataset/happy/"
image_folder_sad = "dataset/sad/"

positive_images = os.listdir(image_folder_happy)
negative_images = os.listdir(image_folder_sad)

happy_image_files = [os.path.join(image_folder_happy, img)
for img in positive_images]
sad_image_files = [os.path.join(image_folder_sad, img) for
img in negative_images]
```

Kode di atas berfungsi untuk membaca seluruh dataset happy dan sad.

```
image_files = []
file_names = []

type = st.radio("Select type", ('All', 'Happy', 'Sad'),
horizontal=True)

if type == 'All':
    for happy_img, sad_img in zip(happy_image_files,
sad_image_files):
        image_files.extend([happy_img, sad_img])
        file_names.extend([os.path.basename(happy_img),
os.path.basename(sad_img)])
elif type == 'Happy':
    image_files.extend(happy_image_files)
    file_names.extend(positive_images)
elif type == 'Sad':
    image_files.extend(sad_image_files)
    file_names.extend(negative_images)
```

Kode diatas berfungsi untuk memilih gambar-gambar yang akan ditampilkan berdasarkan tipe yang diinginkan pengguna.

```
row = st.number_input("Enter the value of row", min_value=1,
max_value=math.ceil(len(image_files) / 6))
```

```
columns = st.columns(6)
for i in range(row):
    for j in range(6):
        index = i * 6 + j
        if index < len(image_files):
            with columns[j]:
                st.image(image_files[index],
caption=file_names[index])
```

Kode di atas berfungsi untuk menampilkan gambar-gambar sejumlah baris yang diinginkan pengguna.

BAB III
PENUTUP

Pada akhirnya, dapat diketahui bahwa website untuk mengidentifikasi emosi pada gambar dapat dibangun menggunakan framework streamlit. Di bagian frontend terdapat fitur untuk menginputkan gambar dan kolom hasil prediksi atau hasil identifikasinya. Sedangkan, pada bagian backend terdapat serangkaian fungsi dalam program yang digunakan untuk menganalisis tekstur (GLCM) serta mengklasifikasikan data gambar menggunakan algoritma KNN. Dengan demikian, pengguna dapat mengecek gambar yang ingin diidentifikasi jenis emosinya.

DAFTAR PUSTAKA

Afifah, L. (2020). Algoritma K-Nearest Neighbor (KNN) untuk Klasifikasi. Retrieved from https://ilmudatapy.com/algoritma-k-nearest-neighbor-knn-untuk-klasifikasi/

Hidayati, K. F. (2022). Algoritma: Pengertian, Perkembangan, Ciri-Ciri, dan Komponennya. Retrieved from https://glints.com/id/lowongan/algoritma-adalah/

Mulachela, H. (2022). Sistem Adalah Suatu Kesatuan, Berikut Teori dan Cirinya. Retrieved from https://katadata.co.id/safrezi/berita/61f37503ef773/sistem-adalah-suatu-kesatuan-berikut-teori-dan-cirinya