KNN IMAGE FINAL PROJECT

PENGANTAR PEMROSESAN DATA MULTIMEDIA

Kelas D

Anggota Kelompok:

Yehezkiel Batara Lumbung (2108561048)

Ni Wayan Sani Utari Dewi (2108561089)

I Made Sudarsana Taksa Wibawa (2108561109)

Dosen Pengampu:

Dr. Anak Agung Istri Ngurah Eka Karyawati, S.Si., M.Eng.

PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM UNIVERSITAS UDAYANA

JIMBARAN

2023

BAB I
PENDAHULUAN

1.1 Latar Belakang

Klasifikasi gambar adalah proses mengidentifikasi dan mengkategorikan gambar ke dalam kelas atau label yang telah ditentukan sebelumnya. Tujuan dari klasifikasi gambar adalah untuk mengembangkan model atau sistem yang dapat secara otomatis mengenali atau mengklasifikasikan gambar-gambar baru berdasarkan pola atau fitur yang terdapat dalam gambar tersebut.

Klasifikasi gambar menggunakan metode K-Nearest Neighbors (KNN) melibatkan penggunaan algoritma KNN untuk mengklasifikasikan gambar berdasarkan kesamaan fitur dengan tetangga terdekat. KNN bekerja dengan prinsip bahwa objek yang memiliki fitur yang mirip cenderung berada dalam kelas yang sama

Sebelum melakukan klasifikasi gambar dengan K-Nearest Neighbors (KNN), ekstraksi fitur dari gambar diperlukan untuk mengubah gambar menjadi representasi numerik yang dapat digunakan oleh algoritma KNN. Salah satu metode ekstraksi fitur yang biasa digunakan adalah Gray Level Co-occurrence Matrix (GLCM).

1.2 Problem Komputasi

Membangun sistem aplikasi untuk mengidentifikasi sentimen/emosi dari sebuah/beberapa ulasan, citra ekspresi wajah, atau suara orang. Terdapat dua sentimen yaitu sentimen positif (atau emosi happy) dan sentimen negatif (atau emosi sad).

Dataset merupakan data Image (citra grayscale ekspresi wajah orang). Semua data dibagi menjadi 2 bagian: 80% untuk dataset training dan 20% untuk dataset testing. Terdapat 3 tahapan utama proses komputasi untuk menghasilkan model klasifikasi yaitu: 1) Preprocessing; 2) Training; dan 3) Testing.

Data image terbagi menjadi 2 label emosi: happy dan sad.

Tahapan preprocessing untuk data image:

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.

Tahapan Training:

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).

Tahapan Testing:

Ukuran evaluasi yang digunakan: akurasi, precision, recall, dan F1-Score.

Tahapan Deployment:

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

Tujuan dari dibuatnya laporan ini adalah sebagai berikut:

1. Menjelaskan dan menerapkan secara rinci Feature Extraction dengan metode berbasis tekstur menggunakan metode GLCM (Gray-Level Co-occurrence Matrix) dalam analisis image.
2. Menjelaskan dan menerapkan secara rinci model K-Nearest Neighbour sebagai model klasifikasi emosi gambar.
3. Menerapkan model yang dibuat dengan implementasi model pada bentuk website.

1.4 Manfaat

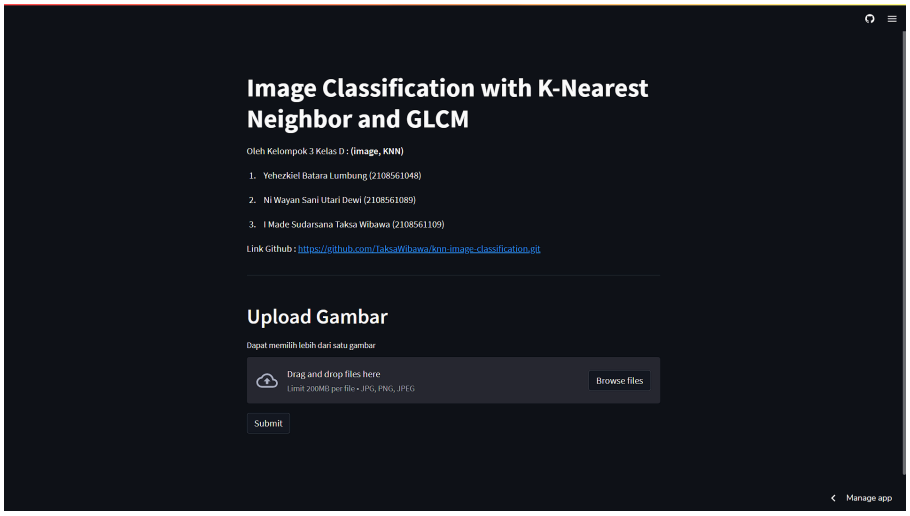
Manfaat dari dibuatnya laporan ini adalah sebagai berikut:

1. Memberikan wawasan yang lebih dalam tentang proses preprocessing untuk data image menggunakan GLCM.
2. Memberikan wawasan yang lebih dalam tentang K-Nearest Neighbour sebagai model klasifikasi emosi gambar.

BAB II
ISI

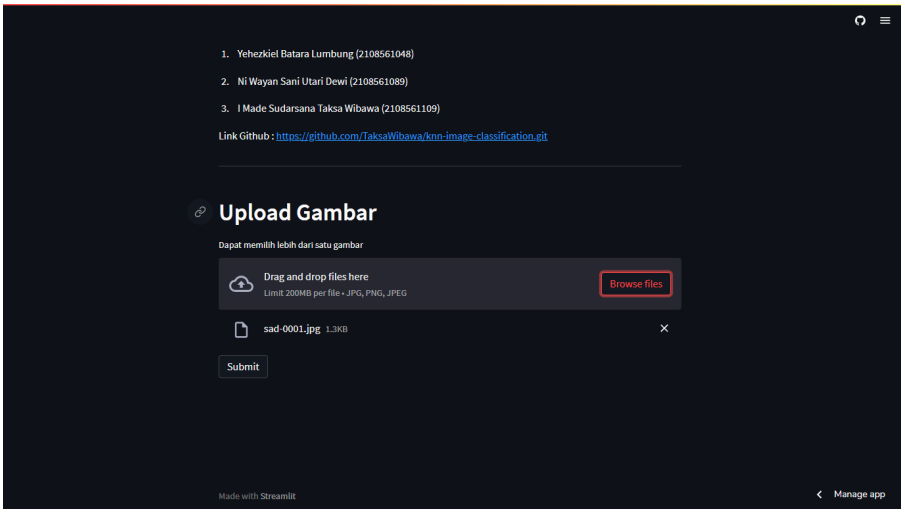
2.1 Manual Aplikasi

2.1.1 Fitur Sistem dan Antar Muka



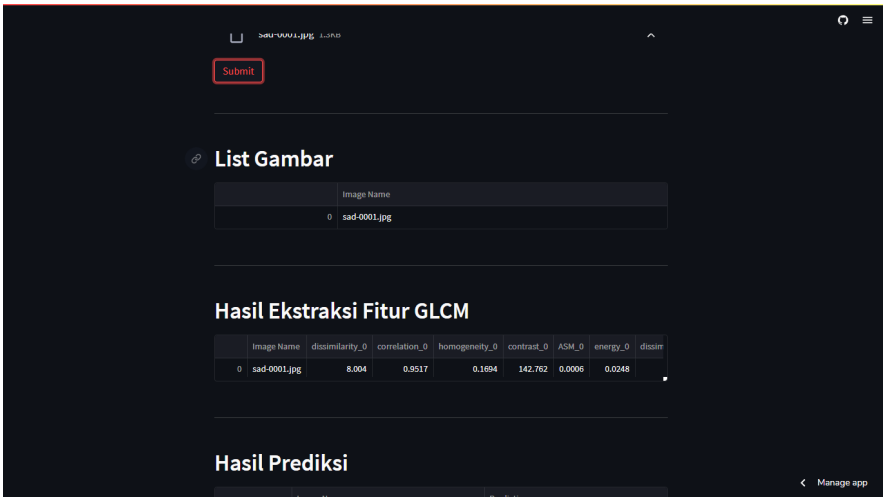
Tampilan awal web

Berikut merupakan tampilan awal aplikasi. User dapat menginputkan image



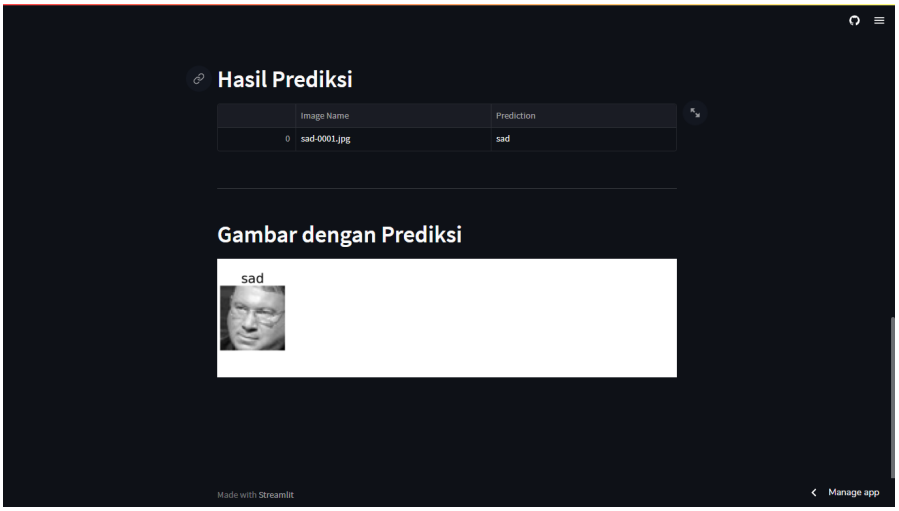
Tampilan Saat Upload Gambar

Berikut merupakan tampilan setelah upload gambar. User dapat submit image



List Gambar dan Hasil Ekstraksi Fitur dengan GLCM

Setelah user menginput file image, maka program akan secara otomatis melakukan identifikasi sentimen dari file tersebut. Menampilkan list gambar yang telah odi submit, image name, dissimilarity, correlation, homogeneity, contrast, ASM, energy.



Hasil Prediksi dari Gambar

Setelah tampilan List gambar dan hasil ekstraksi fitur. Di bawah tampilan itu adakan menampilkan hasil prediksi dari gambar yaitu, nama gambar dan hasil prediksi sentimen beserta gambar yang terkait.

2.2 Source Code Modul

2.2.1 Image_Preprocessing

Pertama akan membaca masukan gambar - gambar yang akan digunakan sebagai dataset pelatihan model. Dataset yang digunakan di sini meliputi 1537 gambar *happy* dan 1463 gambar *sad*.

```python
parent_folder = "FacialExpression/"
subfolder_names = ["happy", "sad"]
df = pd.DataFrame(columns=['Image Name', 'Category'])

df_list = []
for subfolder in subfolder_names:
    subfolder_path = os.path.join(parent_folder, subfolder)
    image_list = os.listdir(subfolder_path)
    image_names = [os.path.splitext(image)[0] for image in image_list]
    category = [subfolder] * len(image_names)
    image_df = pd.DataFrame(
        {"Image Name": image_names, "Category": category})
    df_list.append(image_df)
df = pd.concat(df_list, ignore_index=True)
print(df['Category'].value_counts())

#Output
#happy    1537
#sad      1463
```

Gambar - gambar tersebut akan diubah bentuknya menjadi sebuah DataFrame yang di mana memiliki dua kolom, yaitu Image Name dan Category.

Selanjutnya, gambar - gambar tersebut akan melalui proses preprocessing yang dimana akan dilakukan seperti berikut ini:

```python
def preprocess_image(image_path, target_size):
    # Read the image
    image = cv2.imread(image_path)

    # Convert image to grayscale
    grayscale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Resize image while maintaining aspect ratio
    height, width = grayscale_image.shape[:2]
    if height > width:
        new_height = target_size
        new_width = int(width * (target_size / height))
    else:
        new_width = target_size
        new_height = int(height * (target_size / width))
    resized_image = cv2.resize(grayscale_image, (new_width, new_height))

    return resized_image
```

Gambar akan dibaca pikselnya dengan bantuan library openCV. Selanjutnya gambar yang telah dibaca pikselnya akan diubah menjadi grayscale. Terakhir, akan dilakukan pengubahan ukuran gambar, yang di mana ukuran ini tetap mempertahankan rasio dari gambar, sehingga kami mematok ukuran lebarnya yaitu sebesar 128 (target_size) dan untuk panjangnya akan menyesuaikan gambar. Sehingga pada tahap ini akan mengembalikan gambar yang telah di grayscale dan juga pengubahan ukuran gambar.

2.2.2 Tahap Feature Extraction dengan GLCM

Pada tahap ekstraksi fitur, di sini kami menggunakan GLCM, yang di mana setiap gambarnya dalam kondisi normal akan memberikan fitur sebanyak 6 fitur, yaitu dissimilarity, correlation, homogeneity, contrast, ASM, dan energy.

Selain itu, untuk memperbanyak fitur yang dapat digunakan untuk pemodelan nanti, kami akan melakukan perhitungan GLCM berdasarkan *angle* rotasi dari gambar, yaitu sudut 0, 45, 90, 135, dan 180.

```
angles = [0, np.pi/4, np.pi/2, 3*np.pi/4, np.pi]
```

Setelah menentukan sudut gambar yang ingin diekstraksi fiturnya, per gambarnya kami akan melakukan perhitungan GLCM untuk setiap sudut yang ada, seperti gambar di bawah ini.

```
def extract_features(image):
    features = []
    for angle in angles:
        glcm = graycomatrix(image, [1], [angle], levels=256, symmetric=True, normed=True)
        dissimilarity = graycoprops(glcm, 'dissimilarity').ravel()
        correlation = graycoprops(glcm, 'correlation').ravel()
        homogeneity = graycoprops(glcm, 'homogeneity').ravel()
        contrast = graycoprops(glcm, 'contrast').ravel()
        asm = graycoprops(glcm, 'ASM').ravel()
        energy = graycoprops(glcm, 'energy').ravel()
        angle_features = np.concatenate((dissimilarity, correlation, homogeneity, contrast, asm, energy))
        features.extend(angle_features)

    return np.array(features)
```

Untuk setiap sudut yang ada, pertama akan perhitungan matriks GLCM di dalam variabel glcm. Kemudian, dari perhitungan matriks tadi akan perhitungan tekstur (6 fitur yang disebutkan tadi) dan menyimpannya sesuai dengan variabel yang ditentukan. Terakhir, fitur tadi akan diubah dan disimpan sebagai matriks dua dimensi untuk mempermudah pemodelan nanti.

2.2.3 Tahap Pemodelan K-Nearest Neighbour

```python
for (i, imagePath) in enumerate(df['Image Name']):
    label = df['Category'][i]
    path = os.path.join(parent_folder, label + '/' + imagePath + ".jpg")
    try:
        image = preprocess_image(path, 128)
        feat = extract_features(image)
        features.append(feat)
        labels.append(label)
    except:
        print("File corrupted: {}".format(imagePath))

    # show an update every 200 images until the last image
    if i > 0 and ((i + 1)% 200 == 0 or i == len(imagePath)-1):
        print("[INFO] processed {}/{}".format(i+1, len(df)))

#Output
# [INFO] processed 10/3000
# [INFO] processed 200/3000
# [INFO] processed 400/3000
# [INFO] processed 600/3000
# [INFO] processed 800/3000
# File corrupted: happy-0974
# [INFO] processed 1000/3000
# [INFO] processed 1200/3000
# [INFO] processed 1400/3000
# [INFO] processed 1600/3000
# [INFO] processed 1800/3000
# [INFO] processed 2000/3000
# [INFO] processed 2200/3000
# [INFO] processed 2400/3000
# File corrupted: sad-0967
# [INFO] processed 2600/3000
# [INFO] processed 2800/3000
# [INFO] processed 3000/3000
```

Sebelum masuk ke dalam pemodelan KNN, dari preprocessing dan feature extraction yang telah dilakukan, terdapat dua gambar yang tidak bisa dibaca atau *corrupt*, sehingga kita tidak dapat menggunakan gambar - gambar tersebut sebagai data pelatihan.

Kemudian dataset yang telah dibaca akan dibagi menjadi training data dan testing data dengan rasio pembagian sebesar 80% untuk training dan 20% untuk testing.

```python
(trainFeat, testFeat, trainLabels, testLabels) = train_test_split(
    features, labels, test_size=0.2, random_state=4)
```

Kami juga melakukan inisialisasi terhadap nilai K berapa saja yang akan diuji coba untuk mendapatkan model yang terbaik. K di sini merupakan salah satu parameter penting dalam melakukan pemodelan KNN, untuk menentukan berapa banyak ketetanggaan yang ada.

```python
n_neighbors = [3, 5, 7, 9, 11]
```

Selanjutnya, pelatihan model akan dilakukan secara iterasi, yang di mana yang diiterasikan di sini adalah nilai K agar dapat dibandingkan satu dengan lainnya model dengan nilai K berapa yang memliki performa terbaik.

```
for k in n_neighbors:
    print("[INFO] evaluating feature accuracy for k={}...".format(k))
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(trainFeat, trainLabels)
```

Untuk data yang dimasukan ke model pada tahap training adalah data fitur tadi dan juga label - label dari gambar yaitu *happy* dan *sad*.

Selanjutnya dilakukan testing model untuk menghitung akurasi dari pilihan tetangga yang digunakan pada model.

```
pred_feat = model.predict(testFeat)
acc = accuracy_score(testLabels, pred_feat)
```

Untuk melihat performa dari model, kamu menggunakan fungsi classification_report yang disediakan oleh library sci-kit learn untuk menampilkan akurasi, precision, recall, dan F1-Score dari model.

```
print("[INFO] k-NN classifier: k={}".format(k))
print("[INFO] feature accuracy: {:.2f}%".format(acc*100))
report = classification_report(testLabels, pred_feat, target_names=["happy",
"sad"])
print(report)

#Output
# [INFO] evaluating feature accuracy for k=3...
# [INFO] k-NN classifier: k=3
# [INFO] feature accuracy: 52.83%
#               precision    recall  f1-score   support

# happy             0.55      0.55      0.55       314
# sad               0.51      0.50      0.50       286

# accuracy                              0.53       600
# macro avg         0.53      0.53      0.53       600
# weighted avg      0.53      0.53      0.53       600

# ....
```

2.2.4 Tahap Evaluasi Model

Dikarenakan belum menemukan hasil model yang terbaik, maka di sini kami melakukan hyperparameter tuning, yang dilakukan seperti berikut ini:

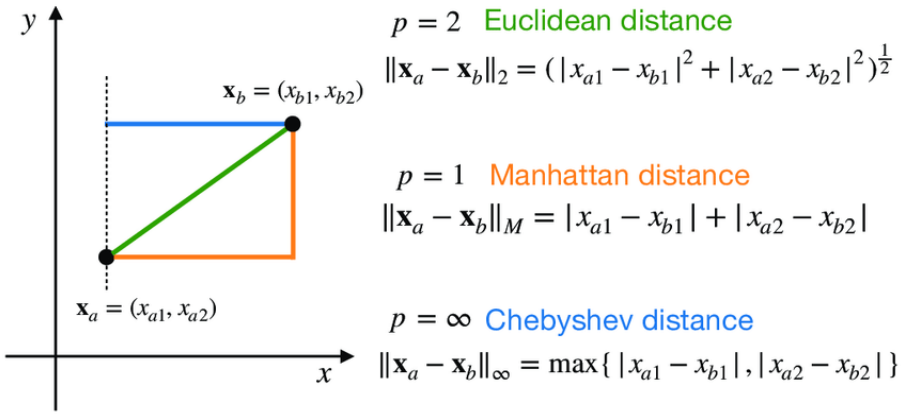
```python
from sklearn.model_selection import GridSearchCV

#List Hyperparameters that we want to tune.
n_neighbors = list(range(1,30,2))
p=[1,2]
#Convert to dictionary
hyperparameters = dict(n_neighbors=n_neighbors, p=p)
#Create new KNN object
knn_2 = KNeighborsClassifier()
#Use GridSearch
clf = GridSearchCV(knn_2, hyperparameters)

#Fit the model
best_model = clf.fit(trainFeat, trainLabels)

#Print The value of best Hyperparameters
print('Best p:', best_model.best_estimator_.get_params()['p'])
print('Best n_neighbors:', best_model.best_estimator_.get_params()['n_neighbors'])
```

Pada tahap ini akan dilakukan *hyperparameter tuning* berdasarkan dua parameter, yaitu nilai k dan p nya. Nilai k yang dipakai adalah rentang nilai dari 1 sampai 30 dan bilangan ganjil, sehingga terdapat 15 nilai k yang akan dicoba. Kemudian untuk nilai p di sini adalah metode distance yang digunakan, yang meliputi nilai 1 dan 2. Untuk lebih jelasnya mengenai nilai p dapat dilihat pada gambar di bawah ini:



Nilai K	Metode Distance	Akurasi
1	Manhattan	54,17023
1	Euclidean	54,29515
3	Manhattan	53,87909
3	Euclidean	53,08542
5	Manhattan	54,33803
5	Euclidean	55,42215
7	Manhattan	55,08812
7	Euclidean	54,75391

9	Manhattan	55,13005
9	Euclidean	54,3369
11	Manhattan	54,33794
11	Euclidean	54,21207
13	Manhattan	53,96233
13	Euclidean	55,04662
15	Manhattan	54,29645
15	Euclidean	54,25461
17	Manhattan	54,96355
17	Euclidean	54,88118
19	Manhattan	55,04662
19	Euclidean	55,04715
21	Manhattan	54,42119
21	Euclidean	54,00418
23	Manhattan	55,50583
23	Euclidean	54,0876
25	Manhattan	55,29741
25	Euclidean	55,04689
27	Manhattan	53,25374
27	Euclidean	54,96312
29	Manhattan	54,25452
29	Euclidean	54,58812

Dari tuning yang dilakukan, didapatkan bahwa hasil kombinasi parameter terbaik yaitu kombinasi antara nilai k yaitu 23 dengan metode distance manhattan atau nilai p sama dengan 1.

2.2.5 Tahap Deployment

Untuk tahap deployment, kami menggunakan Streamlit untuk mengimplementasikan model kami ke dalam bentuk website. Website ini akan melakukan prediksi atas model terbaik yang telah kami uji cobakan.

```
import pickle

filename = "knn_model.sav"
pickle.dump(model, open(filename, 'wb'))
```

Untuk menyimpan model, kami menggunakan library pickle di sini.

BAB III
PENUTUP

3.1 Kesimpulan

Dalam sistem ini, digunakan teknik ekstraksi fitur dari GLCM mengidentifikasi sentimen/emosi dari sebuah/beberapa ulasan, citra ekspresi wajah. Terdapat dua sentimen yaitu sentimen positif (atau emosi happy) dan sentimen negatif (atau emosi sad). Dataset citra grayscale ekspresi wajah dibagi menjadi 80% untuk dataset training dan 20% untuk dataset testing. Pembagian dataset ini umum dilakukan dalam pembelajaran mesin untuk melatih model pada data training dan menguji kinerjanya pada data testing yang belum pernah dilihat sebelumnya. Sistem ini mencapai tingkat akurasi sebesar 54,33% dalam memprediksi sentimen/emosi dari citra ekspresi wajah. Akurasi tersebut mengindikasikan kemampuan model dalam mengklasifikasikan citra sebagai sentimen positif (happy) atau negatif (sad).

Pada website menampilkan pilihan user untuk submit foto yang akan di processing, setelahnya sistem menampilkan list gambar, hasil ekstraksi fitur, dan hasil prediksi gambar tergolong sentimen happy atau sad. Sistem ini memberikan pengguna kemampuan untuk mengunggah foto mereka sendiri dan melihat hasil ekstraksi fitur serta prediksi sentimen/emosi dari gambar tersebut. Ini memberikan pengalaman interaktif kepada pengguna dalam melihat hasil analisis dari gambar yang diunggah.

DAFTAR PUSTAKA

Yanosma, D., Johar, A., & Anggriani, K. (2016). Implementasi Metode K-Nearest Neighbor (KNN) dan Simple Additive Weighting (SAW) dalam Pengambilan Keputusan Seleksi Anggota PASKIBRAKA. Rekursif: Jurnal Informatika, 4(2).

Winda Ika Praseptiana , Agus Wahyu Widodo , & Muh Arif Rahman (2019). Pemanfaatan Ciri Gray Level Co-occurrence Matrix (GLCM) Untuk Deteksi Melasma Pada Citra Wajah.

Rifki Kosasih (2020). Kombinasi Metode KNN Pada Image Processing Untuk Pengenalan Wajah.

M. ZAINI F, RIZKY IBRAHIM S, Hafez Ar R, Anna B (2022). Merancang Sistem Deteksi Pola Gambar Menggunakan Metode K-Nearest Neighbor Classifier