

**Technical Report**  
**Image Classification of Anemia Detection using K-Nearest  
Neighbors (KNN)**



**Disusun Oleh:**

**Kelompok 4**

**Muhammad Zacky Said      (105222046)**

**Muhammad Zaky Tabrani      (105222025)**

**FAKULTAS SAINS DAN ILMU KOMPUTER**  
**PROGRAM STUDI ILMU KOMPUTER**  
**UNIVERSITAS PERTAMINA**

## 1. Pendahuluan

Klasifikasi gambar merupakan salah satu penerapan penting dalam bidang *machine learning*, terutama dalam pemrosesan digital untuk kebutuhan medis dan diagnostik. Dalam *mini project* ini, dilakukan implementasi model klasifikasi gambar menggunakan model *K-Nearest Neighbors* (KNN) untuk membedakan gambar yang menunjukkan kondisi *anemic* dan *nonanemic*. Pemilihan model KNN didasarkan pada kemampuannya dalam menangani klasifikasi sederhana berbasis jarak dan efektif digunakan pada dataset berskala kecil hingga menengah.

## 2. Tujuan

Tujuan dari *mini project* ini adalah membangun model klasifikasi gambar yang mampu mengenali perbedaan antara dua kategori gambar, yaitu *anemic* dan *nonanemic*, menggunakan model *K-Nearest Neighbors* (KNN). Selain itu, dilakukan juga eksplorasi data secara visual dan statistik untuk mendukung proses *preprocessing* dan meningkatkan akurasi model klasifikasi.

## 3. Dataset

Dataset terdiri dari dua kelas utama, yaitu *anemic* dan *nonanemic*, yang masing-masing berisi sejumlah gambar dalam format .png. Dataset diorganisasi dalam folder terpisah berdasarkan kelasnya. Setiap gambar dibaca menggunakan *library* PIL, dan informasi mengenai ukuran (lebar dan tinggi) dikumpulkan dalam sebuah struktur data `class_data`.

Jumlah total gambar berdasarkan kelas:

- Anemic: 2563 gambar
- Nonanemic: 1714 gambar

Beberapa sampel gambar dari masing-masing kelas juga ditampilkan untuk memastikan format dan kualitas gambar, disertai informasi dimensi dari masing-masing gambar.

anemic | Anemic-001 (12).png | Ukuran: (430, 225)



Gambar 3.1 Sampel anemic

nonanemic | Non-Anemic-002 (2).png | Ukuran: (214, 271)



Gambar 3.2 Sampel nonanemic

#### 4. Preprocessing

a. Eksplorasi Awal

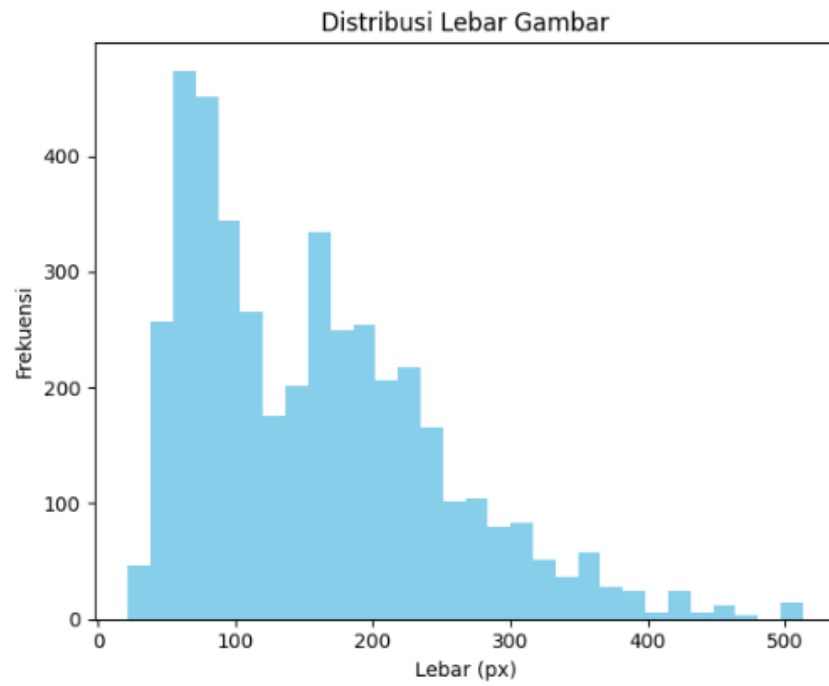
Jumlah gambar untuk masing-masing kelas dihitung, dan ditampilkan menggunakan bar chart untuk melihat distribusi kelas. Kemudian, dilakukan perhitungan statistik deskriptif terhadap ukuran gambar (lebar dan tinggi), serta rasio aspek (lebar/tinggi) dari gambar di setiap kelas.

**Tabel Statistik Ukuran Gambar:**

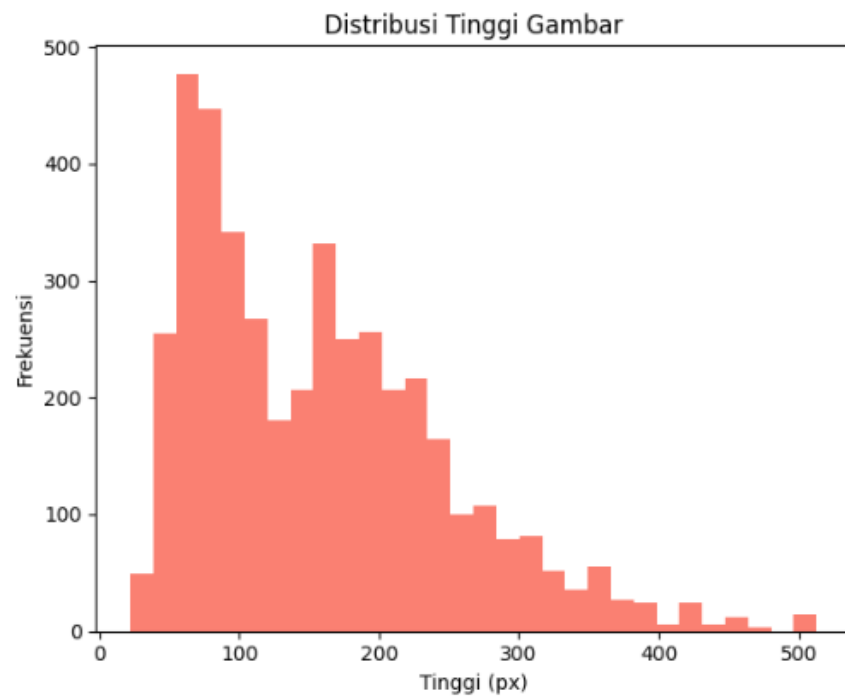
Kelas	Rata-rata lebar	Rata-rata tinggi	Rasio aspek rata-rata	Min/Max lebar	Min/Max tinggi
<i>Anemic</i>	151.80px	151.74px	1.53	22/462	22/462
<i>Nonanemic</i>	168.20px	167.67px	1.59	22/513	22/513

b. Visualisasi Ukuran Gambar

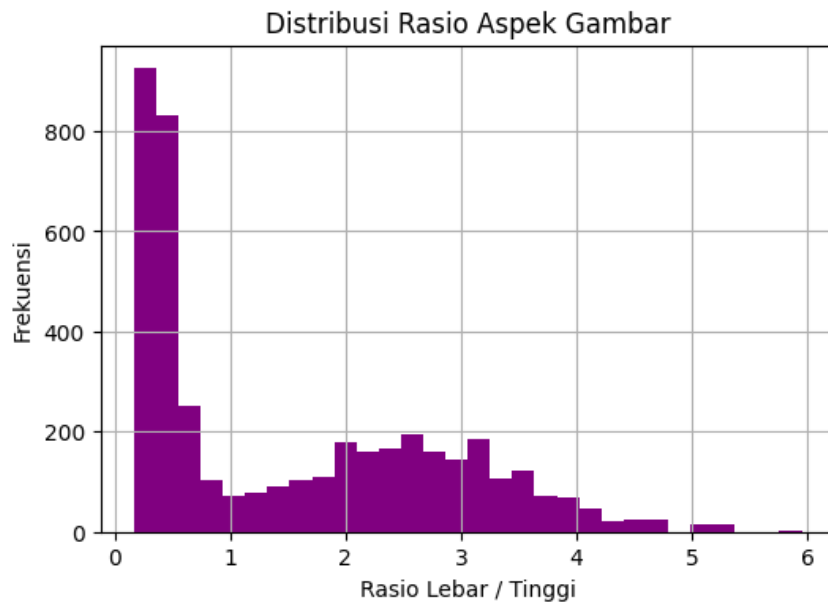
Distribusi ukuran gambar divisualisasikan menggunakan histogram untuk lebar, tinggi, dan rasio aspek. Selain itu, dibuat juga heatmap 2D (menggunakan hexbin plot) untuk menunjukkan sebaran ukuran gambar (lebar vs tinggi), yang membantu memutuskan apakah perlu dilakukan proses *resize*.



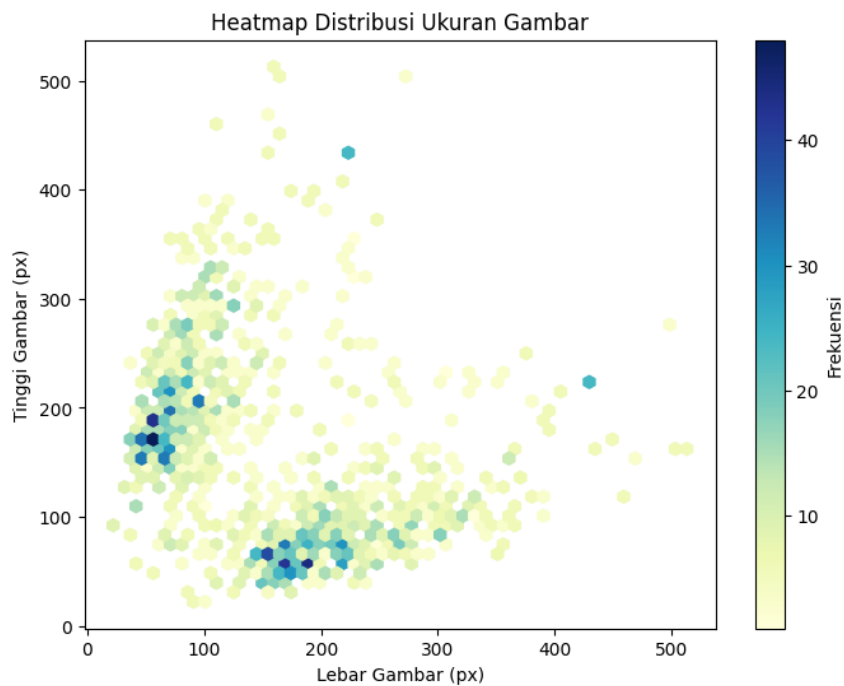
Gambar 4.1 Histogram distribusi lebar gambar



Gambar 4.2 Histogram distribusi tinggi gambar



Gambar 4.3 Histogram distribusi rasio aspek gambar



Gambar 4.4 Heatmap 2D ukuran gambar (lebar vs tinggi)

### c. *Preprocessing* Gambar

Seluruh gambar diubah ukurannya menjadi 64x64 piksel dan dikonversi ke format RGB. Setelah itu, gambar dinormalisasikan dengan membagi nilai piksel terhadap 255, dan kemudian di-*flatten* menjadi vektor 1 dimensi agar dapat digunakan dalam pelatihan model. Label untuk setiap gambar diubah ke format numerik (*anemic* = 0, *nonanemic* = 1).

```

IMG_SIZE = (64, 64)
X, y = [], []

for class_name in classes:
    folder_path = os.path.join(base_path, class_name)
    for file_name in os.listdir(folder_path):
        if file_name.endswith(('.jpg', '.jpeg', '.png')):
            img_path = os.path.join(folder_path, file_name)
            try:
                img = Image.open(img_path).convert('RGB')
                img = img.resize(IMG_SIZE)
                img_array = np.array(img) / 255.0
                X.append(img_array.flatten())
                y.append(class_name)
            except:
                continue

X = np.array(X)
y = np.array(y)
le = LabelEncoder()
y_encoded = le.fit_transform(y)

```

Gambar 4.5 Kode preprocessing

## 5. Metodologi

Sebelum pelatihan, dilakukan reduksi dimensi menggunakan PCA (*Principal Component Analysis*) dengan jumlah komponen sebanyak 100 untuk menyederhanakan representasi fitur gambar. Setelah itu, dilakukan penyeimbangan data menggunakan metode SMOTE (*Synthetic Minority Oversampling Technique*) untuk mengatasi ketidakseimbangan kelas dan meningkatkan performa klasifikasi.

```

pca = PCA(n_components=100)
X_pca = pca.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_pca, y_encoded, test_size=0.2, random_state=42)

sm = SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train)

```

Gambar 5.1 Kode PCA dan SMOTE

## 6. Implementasi

Model yang digunakan adalah *K-Nearest Neighbors* (KNN) dari *library* sklearn, dengan parameter `n_neighbors = 3`. Dataset dibagi menjadi data latih dan data uji dengan perbandingan 80:20. Model dilatih menggunakan data hasil *preprocessing* dan reduksi dimensi.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

model = KNeighborsClassifier(n_neighbors=3)

● model.fit(X_train_res, y_train_res)
✓ 0.0s
```

KNeighborsClassifier ⓘ ?

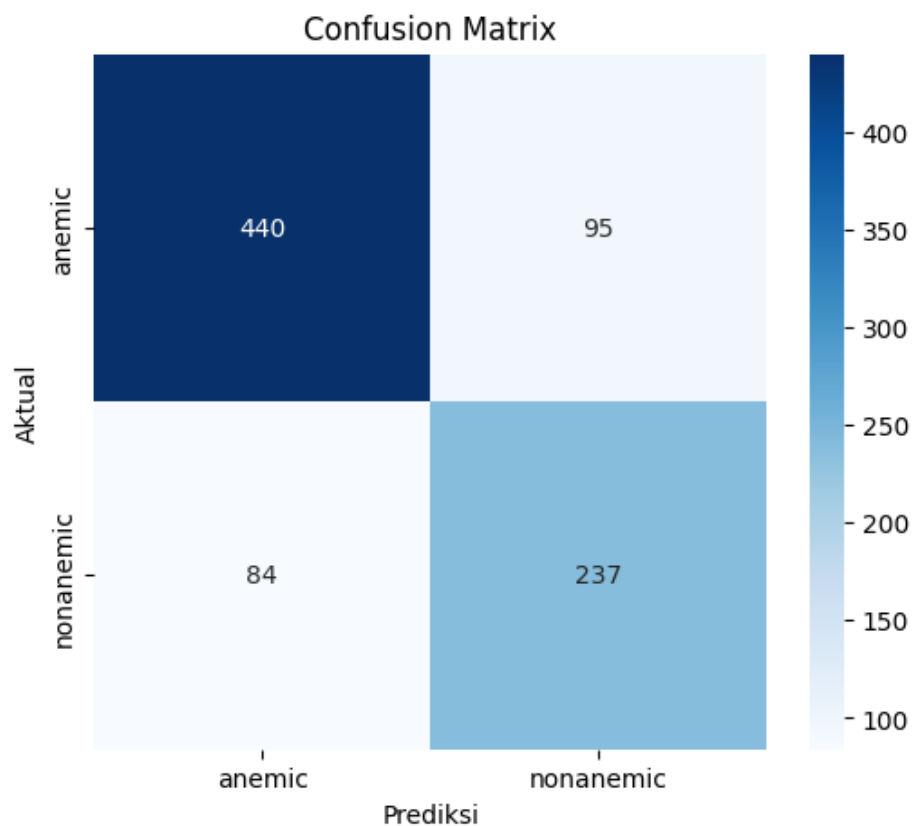
KNeighborsClassifier(n\_neighbors=3)

Gambar 6.1 Kode pelatihan model

## 7. Hasil

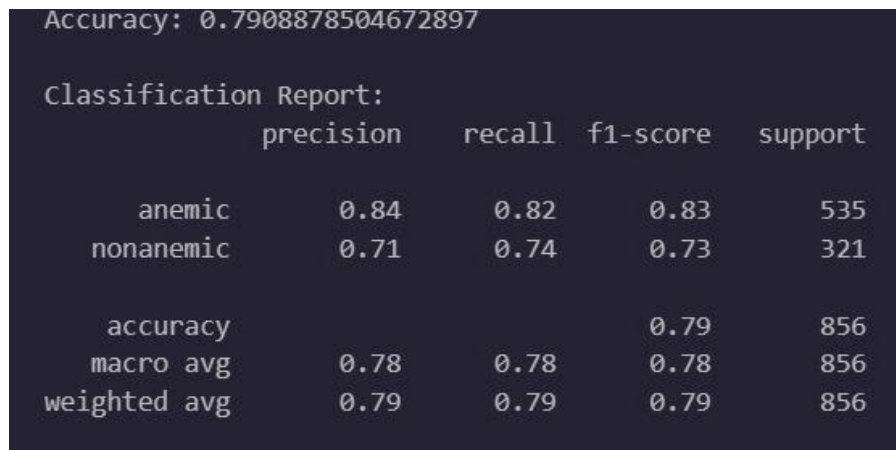
Evaluasi model dilakukan menggunakan *confusion matrix* dan *classification report* dari sklearn. Hasil evaluasi menunjukkan akurasi model sebesar 0.79, yang tergolong cukup baik untuk model sederhana seperti KNN. *Confusion matrix* juga menunjukkan distribusi prediksi terhadap kelas yang benar dan salah.

*Confusion Matrix*:



Gambar 7.1 Confusion matrix

### Classification Report:

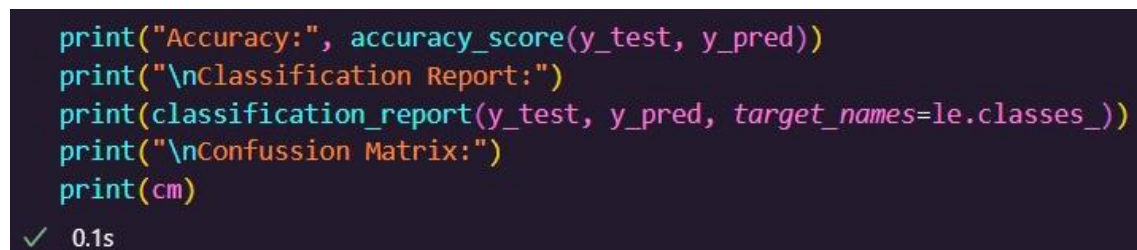


```
Accuracy: 0.7908878504672897
```

	precision	recall	f1-score	support
anemic	0.84	0.82	0.83	535
nonanemic	0.71	0.74	0.73	321
accuracy			0.79	856
macro avg	0.78	0.78	0.78	856
weighted avg	0.79	0.79	0.79	856

Gambar 7.2 Classification Report

### Evaluasi Model:



```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=le.classes_))
print("\nConfussion Matrix:")
print(cm)
```

✓ 0.1s

Gambar 7.3 Kode evaluasi model

## 8. Kesimpulan

Berdasarkan hasil yang diperoleh, model KNN mampu digunakan untuk klasifikasi gambar dengan tingkat akurasi yang cukup baik, khususnya untuk dataset dengan jumlah sedang dan fitur yang direduksi secara tepat. Proses EDA dan *preprocessing* terbukti penting dalam menyiapkan data untuk pelatihan model. Untuk pengembangan lebih lanjut, dapat dipertimbangkan penggunaan metode klasifikasi berbasis deep learning seperti CNN (*Convolutional Neural Network*) untuk meningkatkan akurasi dan menangkap pola visual yang lebih kompleks.