TECHNICAL REPORT

PRAKTIKUM MACHINE LEARNING

KLASIFIKASI GAMBAR MENGGUNAKAN MODEL SUPPORT VECTOR MACHINE (SVM)

Disusun oleh :

1. **Sheva Haya Milano (105222036)**
2. **Fitria Nurhaliza (105222035)**

PROGRAM STUDI ILMU KOMPUTER

UNIVERSITAS PERTAMINA

2025

BAB I

PENDAHULUAN

1.1 LATAR BELAKANG

Klasifikasi Gambar adalah salah satu tugas dalam pengenalan visual yang bertujuan untuk memahami dan mengkategorikan gambar secara keseluruhan dibawah label tertentu. Dengan meningkatnya jumlah data visual yang dihasilkan setiap hari, kemampuan untuk melakukan klasifikasi secara otomatis menjadi sangat penting dalam berbagai aplikasi, seperti pengenalan wajah, pengawasan, dan diagnosis medis [1].

Salah satu algoritma yang banyak digunakan untuk klasifikasi adalah *Support Vector Machine* (SVM). SVM adalah algoritma pembelajaran mesin untuk mengatasi masalah klasifikasi dan juga regresi. Algoritma *Support Vector Machine* sendiri yaitu *Supervised Learning*. Metode ini bekerja dengan memetakan data ke dalam ruang fitur dengan dimensi tinggi. Pemetaan ini nantinya akan membuat titik data dapat dikategorikan meskipun data mungkin tidak bisa dipisahkan secara linear. Setelah pemisah data ditemukan, maka data dapat di transformasikan sehingga pemisah data nantinya dapat digambarkan dengan hyperlane [2]. Kelebihan SVM ini adalah kemampuannya untuk menangani data yang tidak seimbang dan memberikan hasil yang baik meskipun dengan jumlah data yang terbatas.

Pada proyek ini akan diterapkan model SVM, *Exploratory Data Analysis* (EDA), dan Preprocessing data sangat penting untuk memahami karakteristik dataset serta menyiapkan data agar dapat digunakan dalam model. EDA membantu mengidentifikasi pola, anomali, dan distribusi data, sedangkan preprocessing meliputi tahapan seperti normalisasi, augmentasi data dan pengurangan dimensi untuk meningkatkan performa model.

1.2 TUJUAN

Adapun tujuan dari proyek ini yaitu :

- Melakukan EDA untuk memahami karakteristiknya, seperti distribusi ukuran gambar, jumlah gambar dalam setiap kategori, dan visualisasi contoh gambar.
- Preprocessing data untuk mempersiapkan data sebelum dimasukkan ke dalam model SVM
- Mengembangkan model SVM untuk melakukan klasifikasi gambar fitur yang diekstrak dari dataset.
- Melakukan evaluasi model menggunakan metrik evaluasi seperti akurasi, precision, recall, dan confusion matrix untuk menilai performa model SVM yang dibangun.

BAB II

DASAR TEORI

2.1 DEFINISI KLASIFIKASI GAMBAR

Klasifikasi gambar adalah proses pengelompokan gambar ke dalam kategori atau kelas tertentu berdasarkan fitur-fitur yang ada dalam gambar tersebut. Dalam Machine Learning, klasifikasi gambar merupakan jenis pembelajaran terawasi dimana model dilatih menggunakan dataset berlabel. Klasifikasi gambar merupakan disiplin ilmu visi komputer yang telah berlangsung selama beberapa decade [3]. Model belajar mengenali pola dan karakteristik yang membedakan satu kelas dari kelas lainnya, ini memungkinkan prediksi untuk gambar baru yang belum pernah dilihat sebelumnnya. Aplikasi klasifikasi gambar sangat luas, mencakup pengenalan wajah, deteksi objek, dan analisis citra medis, Dimana akurasi dan kehandalan adalah kunci keberhasilan.

Dalam Machine Learning, pendekatan yang digunakan untuk klasifikasi gambar dapat dibagi menjadi dua, yaitu *Supervised Learning* dan *Unsupervised Learning*, algoritma *Supervised learning* sendiri menggunakan set data sampel untuk melatih dirinya sendiri guna membuat prediksi, yang secara berulang menyesuaikan diri untuk meminimalkan error. Set data ini diberi label sesuai konteks, memberikan nilai output yang diinginkan untuk memungkinkan model memberikan jawaban yang "benar". Sedangkan, Algoritma *Unsupervised Learning* bekerja secara independen untuk mempelajari struktur yang melekat pada data tanpa panduan atau instruksi khusus, cukup hanya memberikan data input tanpa label dan membiarkan algoritma mengidentifikasi pola yang terjadi secara alami dalam set data [4]. Proses klasifikasi gambar juga melalui tahap penting seperti *Explanatory Data Analysis* (EDA).

2.2 *EXPLANATORY DATA ANALYSIS* (EDA)

Explanatory Data Analysis (EDA) adalah proses analisis awal data yang bertujuan untuk memahami karakteristik, struktur, dan komponen penting dari dataset sebelum melakukan analisis statistik atau pemodelan prediktif lebih lanjut. EDA tidak hanya berfokus pada pengujian hipotesis atau pemodelan statistik, tetapi juga eksplorasi awal seperti data cleaning, analisis statistika deskriptif, visualisasi data, dan penilaian kualitas data. EDA sangat penting dilakukan karena dapat memahami data secara mendalam, identifikasi dan penanganan outlier, Mengurangi risiko kesalahan analisis, dan lainnya [5].

EDA dapat dibagi menjadi beberapa teknik, yaitu :

- **Analisis Bivariant :** proses menganalisis dua variabel untuk mengevaluasi hubungan sebab akibat, korelasi, dan ketergantungan di antara mereka. Alat visual yang sering digunakan dalam analisis bivariat yaitu *scatter plot, bar chart,* dan matriks korelasi
- **Analisis Univariat :** analisis ini berfokus pada satu variabel tunggal. Tujuannya yaitu untuk menggambarkan dan merangkum data tersebut. alat yang umum digunakan dalam analisis univariat yaitu histogram, *box plot*, serta statistik deskriptif seperti mean, median, dan modus
- **Analisis Multivariat** : Teknik ini melibatkan analisis lebih dari dua variabel untuk memahami hubungan dan interaksi. Ini mengidentifikasi pola, tren, serta korelasi yang tidak terlihat dalam analisis univariat atau bivariat. Metode yang digunakan yaitu *Principal Component Analysis* (PCA), analisis kluster, dan model regresi multivariant.
- **Time-series analysis:** Metode analisis untuk data yang dikumpulkan secara berkala seiring waktu. Tujuannya untuk mengidentifikasi tren dan pola musiman, dalam data. Ini penting dalam peramalan dan pemodelan ekonometrik. Teknik yang sering digunakan yaitu timeline plot dan model ARIMA.
- **Analisis outlier:** Outlier adalah nilai yang sangat berbeda dari sebagian besar data. Analisis ini bertujuan untuk mengidentifikasi dan menangani outlier. Box plot, scatter plot, dan metode statistik seperti *Z-score* dipakai untuk mendeteksi outlier.
- **Missing data analysis:** Teknik ini diterapkan untuk menangani masalah data yang hilang dalam dataset. Ini melibatkan identifikasi pola data yang hilang dan menerapkan metode seperti imputasi, penghapusan baris, atau model statistik untuk mengatasi masalah.

- **Visualisasi data:** Aspek penting dari EDA yang melibatkan penggunaan grafik, diagram, dan peta untuk memvisualisasikan data. Visualisasi membantu dalam memahami data secara intuitif dan mengidentifikasi pola atau anomali yang tidak terlihat dalam analisis numerik. Alat visualisasinya yaitu *bar graph, pie chart, heatmap,* dan dashboard interaktif [5].

2.3 PRE-PROCESSING

Preprocessing merupakan salah satu tahapan untuk menghilangkan beberapa permasalahan yang bisa menganggu saat pemrosesan data. Melalui data preprocessing, proses mining akan berjalan dengan lebih efektif dan efisien. Karena data yang telaj melalui pra-pemrosesan data, adalah data yang sudah melalui beberapa tahap pembersihan, tahap-tahap dalam preprocessing yaitu :

- **Data Cleaning (Pembersihan Data),** yaitu data mentah yang telah diperoleh perlu diseleksi Kembali. Kemudian dihapus atau hilangkan data-data yang tidak lengkap, tidak relevan, dan tidak akurat.
- **Data Integration,** Melakukan pengecekan data-data yang datang dari berbagai sumber tersebut supaya memiliki format sumber yang sama, karena data preprocessing akan menggabungkan beberapa data dalam satu dataset.
- **Transformasi Data,** Melakukan penyamaan seluruh data yang terkumpul supaya dapat mempermudah proses analisis data.
- **Data Reduction,** mengurangi sampel data yang diambil, tetapi tidak akan mengubah hasil analisis data.

  Terdapat 3 teknik yang dapat diterapkan saat melakukan pengurangan data, yaitu dimensionality reduction, numerosity reduction, data compression [6].

2.4 SUPPORT VECTOR MACHINE (SVM)

Support Vector Machine (SVM) adalah salah satu metode dalam supervised learning yang biasanya digunakan untuk klasifikasi (Support Vector Classification) dam regresi (Support Vector Regression). SVM ini dapat mengatasi masalah klasifikasi dan regresi dengan linear maupun non linear. SVM juga digunakan untuk mencari hyperlane terbaik dengan memaksimalkan jarak antar kelas. Hyperlane adalah sebuah fungsi yang dapat digunakan untuk pemisah antar kelas, hyperlane dikenal juga dengan fungsi yang digunakan untuk klasifikasi di dalam ruang kelas dimensi yang lebih tinggi, memungkinkan klasifikasi non-linear [7].

Hyperplane dipilih karena untuk memaksimalkan margin, yaitu jarak antara hyperplane dan titik data terdekat dari masing-masing kelas, yang disebut support vectors. Kelebihan SVM yaitu kemampuannya untuk bekerja dengan data yang besar dan kompleks, serta kinerja yang baik dalam situasi di mana jumlah fitur jauh lebih besar daripada jumlah sampel. SVM banyak digunakan dalam berbagai aplikasi, termasuk pengenalan wajah dan klasifikasi teks.

BAB III

HASIL DAN IMPLEMENTASI

3.1 EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) dilakukan untuk memahami karakteristik awal dataset citra yang digunakan, yang terbagi menjadi dua kategori: *anemic* dan *non-anemic*. Beberapa langkah EDA yang dilakukan:

3.1.1 Menghitung jumlah gambar per kategori:

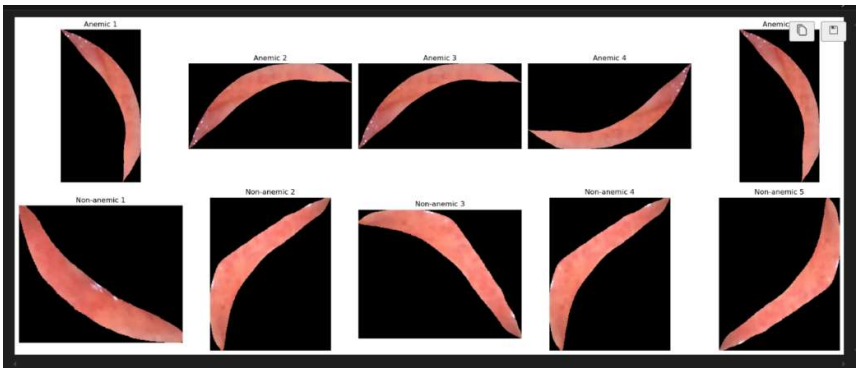
```
Jumlah gambar anemic: 2563
Jumlah gambar non-anemic: 1714
    category  count  height_mean  height_std  width_mean  width_std  \
0     anemic   2563   151.737417   87.199527  151.801014  87.488780
1  nonanemic   1714   167.669778   95.412631  168.202450  95.612155

   min_height  max_height  min_width  max_width
0          22         462         22        462
1          22         513         22        513
```

- **Jumlah Gambar**: Terdapat 2.563 gambar *anemic* dan 1.714 gambar *non-anemic*.
- **Dimensi Gambar**:
    - ***Anemic***: Rata-rata tinggi dan lebar sekitar 151.74 piksel (std ~87.2).
    - ***Non-anemic***: Rata-rata tinggi dan lebar sekitar 167.67 piksel (std ~95.4), lebih besar daripada *anemic*.
- **Rentang Dimensi**:
    - ***Anemic***: Tinggi/lebar minimum 22 piksel, maksimum 462 piksel.
    - ***Non-anemic***: Tinggi/lebar minimum 22 piksel, maksimum 513 piksel.

Gambar *non-anemic* cenderung lebih besar dengan variasi dimensi yang lebih tinggi dibandingkan gambar *anemic*.

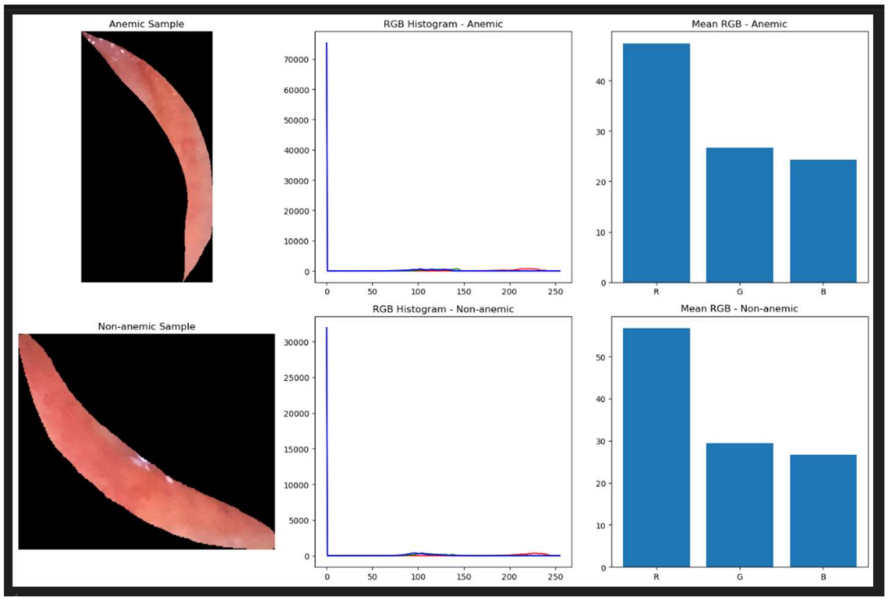
3.1.2 Analisis ukuran gambar

Gambar-gambar bervariasi ukuran dan resolusinya, namun sebagian besar berada pada rentang tinggi dan lebar antara 200–500 piksel.



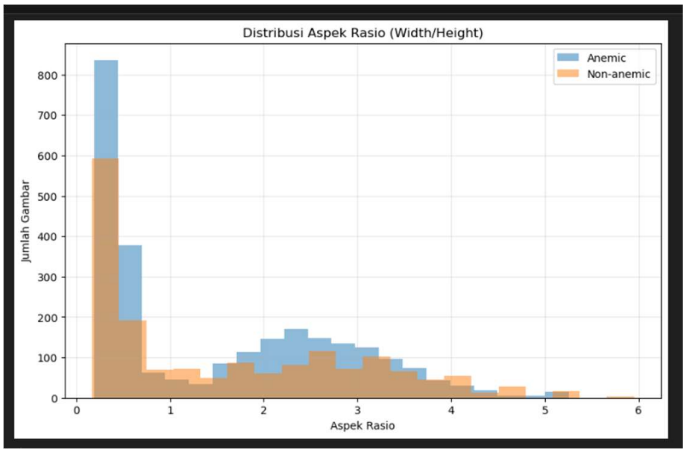
3.1.3 Visualisasi sampel gambar

Ditampilkan masing-masing 5 contoh gambar dari kedua kategori untuk observasi visual langsung.



3.1.4 Analisis distribusi aspek rasio gambar

Diperoleh histogram aspek rasio (width/height) yang memperlihatkan sebaran rasio cukup seragam di kedua kategori.



3.2 PREPROCESSING DATA

Preprocessing data adalah tahap penting dalam pengembangan model machine learning untuk meningkatkan kualitas data dan memfasilitasi proses pembelajaran yang lebih efektif. Berikut tahapan preprocessing yang dilakukan pada dataset gambar *anemic* dan *non-anemic*:

3.2.1 Resize Gambar

Untuk memastikan keseragaman ukuran input, semua gambar diubah ke ukuran yang sama yaitu 128x128 piksel. Hal ini dilakukan dengan menggunakan fungsi cv2.resize().

3.2.2 Konversi ke Grayscale

Setiap gambar dikonversi ke grayscale untuk mengurangi kompleksitas warna dan memfokuskan model pada pola bentuk daripada informasi warna. Konversi ini dilakukan dengan fungsi cv2.cvtColor() dengan parameter cv2.COLOR_BGR2GRAY.

3.2.3 Perbaikan Kontras dengan CLAHE

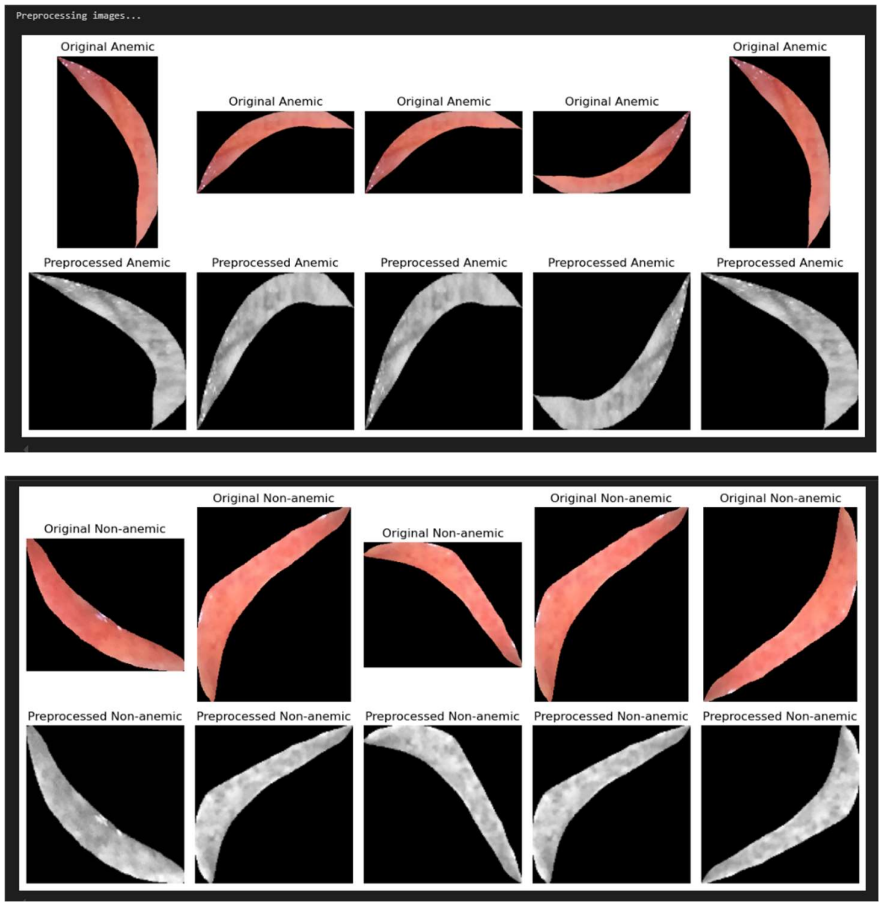
Untuk meningkatkan kontras dan detail gambar, digunakan teknik Contrast Limited Adaptive Histogram Equalization (CLAHE). Teknik ini membantu menonjolkan fitur-fitur penting pada gambar dengan cara yang adaptif berdasarkan area lokal. Parameter yang digunakan adalah clipLimit=2.0 dan tileGridSize=(8, 8).

3.2.4 Normalisasi

Normalisasi data dilakukan dengan membagi nilai piksel dengan 255 sehingga semua nilai piksel berada dalam rentang [0,1]. Hal ini membantu dalam konvergensi model yang lebih cepat dan stabil selama proses pelatihan.

```python
# Fungsi preprocessing
def preprocess_images(images, target_size=(128, 128)):
    processed_images = []

    for img in images:
        # Resize ke ukuran yang sama
        resized = cv2.resize(img, target_size)

        # Konversi ke grayscale
        gray = cv2.cvtColor(resized, cv2.COLOR_BGR2GRAY)

        # Perbaikan kontras dengan CLAHE
        clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
        enhanced = clahe.apply(gray)

        # Normalisasi nilai piksel ke range [0,1]
        normalized = enhanced / 255.0

        processed_images.append(normalized)

    return np.array(processed_images)

print("Preprocessing images...")
X_anemic = preprocess_images(anemic_images)
X_nonanemic = preprocess_images(nonanemic_images)

# Visualisasi beberapa gambar hasil preprocessing
plt.figure(figsize=(12, 6))
for i in range(min(5, len(anemic_images))):
    # Gambar original
    plt.subplot(2, 5, i+1)
    plt.imshow(cv2.cvtColor(anemic_images[i], cv2.COLOR_BGR2RGB))
    plt.title('Original Anemic')
    plt.axis('off')

    # Gambar hasil preprocessing
    plt.subplot(2, 5, i+6)
    plt.imshow(X_anemic[i], cmap='gray')
    plt.title('Preprocessed Anemic')
    plt.axis('off')

plt.tight_layout()
plt.show()
```

3.2.5 Hasil Preprocessing



Hasil preprocessing menunjukkan perbedaan yang signifikan antara gambar asli dan gambar yang telah diproses. Pada gambar yang diproses:

- Warna telah dikonversi ke grayscale
- Kontras telah ditingkatkan sehingga fitur-fitur penting lebih terlihat
- *Noise* telah berkurang
- Fitur sel darah lebih jelas terlihat

Seperti yang terlihat pada gambar output, perbedaan antara gambar *anemic* dan *non-anemic* setelah preprocessing menunjukkan bahwa proses ini berhasil menonjolkan fitur-fitur penting yang dapat membantu model dalam melakukan klasifikasi dengan lebih baik.

3.2.6 Reshaping Data

Setelah preprocessing pada level gambar, data perlu direshape untuk menyesuaikan format input yang dibutuhkan oleh model machine learning. Proses ini melibatkan flattening data gambar 2D menjadi vektor 1D:

```
    y_anemic = np.ones(len(X_anemic))
    y_nonanemic = np.zeros(len(X_nonanemic))

    X = np.concatenate((X_anemic, X_nonanemic), axis=0)
    y = np.concatenate((y_anemic, y_nonanemic), axis=0)

    # Reshape untuk model ML tradisional (flatten gambar)
    X_flat = X.reshape(X.shape[0], -1)
    print(f"Shape data setelah flatten: {X_flat.shape}")
13]

..  Shape data setelah flatten: (4277, 16384)
```

Hasil output menunjukkan dimensi dataset setelah proses flattening adalah (4277, 16384), yang berarti terdapat 4277 sampel gambar yang telah diubah menjadi vektor dengan 16384 fitur untuk setiap gambar.

3.3 PEMILIHAN MODEL & TRAINING

3.3.1 Pemilihan Model

Setelah melakukan preprocessing data dan memahami karakteristik dataset melalui EDA, langkah selanjutnya adalah pemilihan model yang sesuai untuk klasifikasi gambar *anemic* dan *non-anemic*. Dalam penelitian ini, model *Support Vector Machine* (SVM) dipilih sebagai model klasifikasi dengan pertimbangan bahwa :

1. SVM efektif untuk kasus klasifikasi biner (*anemic* dan *non-anemic*)
2. SVM dapat menangani data dengan dimensi tinggi (16.384 fitur per gambar)
3. SVM mampu membuat batasan keputusan yang optimal untuk memisahkan kelas
4. SVM relatif tahan terhadap *overfitting*, terutama pada dataset dengan jumlah fitur yang besar

SVM bekerja dengan mencari hyperplane optimal yang memaksimalkan margin antara dua kelas. Untuk kasus ini, digunakan SVM dengan kernel RBF (*Radial Basis Function*) yang dapat menangani kasus di mana data tidak dapat dipisahkan secara linear di ruang fitur asli.

3.3.2 Pembagian Data

Sebelum melatih model, dataset dibagi menjadi data training dan data testing menggunakan fungsi train_test_split dari library scikit-learn. Pembagian data dilakukan dengan perbandingan 70:30 untuk memastikan model memiliki cukup data untuk training sekaligus menyediakan subset data yang memadai untuk evaluasi.

```
# Split data menjadi training dan testing
X_train, X_test, y_train, y_test = train_test_split(
    X_flat, y, test_size=0.3, random_state=42, stratify=y
)

print(f"Jumlah data training: {X_train.shape[0]}")
print(f"Jumlah data testing: {X_test.shape[0]}")
```

Hasil output menunjukkan pembagian dataset menjadi dua, yaitu data training dan data testing:

- Data training: 2.993 sampel
- Data testing: 1.284 sampel

Pembagian ini memastikan bahwa model memiliki cukup data untuk belajar mengenali pola dan fitur dalam data. Parameter stratify=y digunakan untuk memastikan distribusi kelas dalam data training dan testing tetap proporsional dengan distribusi dalam dataset asli.

3.3.3 Standardisasi Fitur

Standardisasi fitur merupakan tahap penting dalam pelatihan model SVM karena model ini sensitif terhadap skala fitur. Standardisasi dilakukan menggunakan teknik StandardScaler dari scikit-learn yang mengubah nilai fitur sehingga memiliki mean 0 dan standar deviasi 1.

```
# Standardisasi fitur
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print(f"Shape data training: {X_train_scaled.shape}")
print(f"Shape data testing: {X_test_scaled.shape}")
```

Hasil output standardisasi:

- Shape data training: (2993, 16384)
- Shape data testing: (1284, 16384)

Standardisasi dilakukan dengan menggunakan fit_transform pada data training dan transform pada data testing. Hal ini penting untuk menghindari data leakage, di mana informasi dari data testing tidak boleh digunakan dalam proses standardisasi data training.

3.3.4 Training Model SVM

```
print("\n--- Training SVM Model ---")
svm_model = SVC(kernel='rbf', random_state=42)
svm_model.fit(X_train_scaled, y_train)
```

Model SVM dikonfigurasi dengan parameter sebagai berikut:

- **kernel='rbf'**: Menggunakan kernel *Radial Basis Function* yang cocok untuk menangani hubungan non-linear antara fitur dan kelas target
- **random_state=42**: Menetapkan seed untuk memastikan hasil yang konsisten dan reproduktibilitas

Proses pelatihan model SVM melibatkan:

1. Pembentukan *hyperplane* optimal yang memisahkan kelas *anemic* dan *non-anemic*
2. Optimisasi margin antara kedua kelas
3. Identifikasi *support vector* (sampel yang berada dekat dengan boundary decision)

3.4 EVALUASI MODEL

3.4.1 Evaluasi Model Awal

Pada bagian ini, akan dijelaskan hasil dan pembahasan mengenai evaluasi model klasifikasi *Support Vector Machine* (SVM) yang telah dilatih untuk mengklasifikasikan pasien ke dalam kategori "*Anemic*" dan "*Non-anemic*". Evaluasi dilakukan menggunakan beberapa metrik kinerja dan visualisasi yang umum digunakan dalam klasifikasi biner.

```
y_pred = svm_model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")

Accuracy: 0.7516
```

Hasil evaluasi model SVM menunjukkan tingkat akurasi sebesar 0.7516 atau 75.16%, yang berarti model berhasil mengklasifikasikan dengan benar sekitar 75.16% dari total data pengujian. Untuk memahami lebih detail tentang kinerja model, *confusion matrix* digunakan sebagai alat visualisasi yang menunjukkan distribusi prediksi benar dan salah.

```
# Confusion Matrix
print("\nConfusion Matrix:")
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

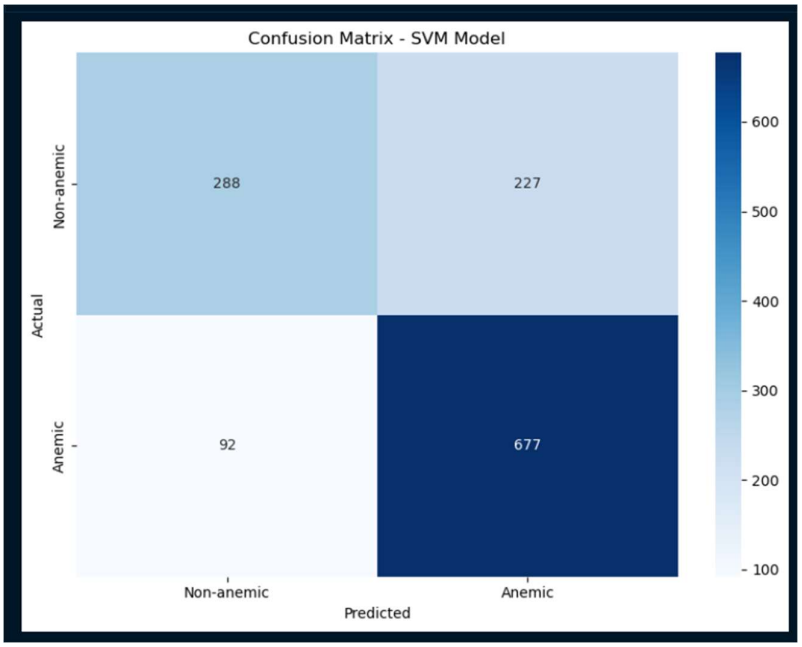
Hasil confusion matrix berupa:

```
Confusion Matrix:
[[288 227]
 [ 92 677]]
```

Dari confusion matrix di atas, dapat diinterpretasikan:

- *True Positives* (TP): 677 — jumlah prediksi benar untuk kelas positif (*Anemic*).
- *True Negatives* (TN): 288 — jumlah prediksi benar untuk kelas negatif (*Non-anemic*).
- *False Positives* (FP): 227 — jumlah prediksi salah untuk kelas negatif (model memprediksi positif padahal negatif).
- *False Negatives* (FN): 92 — jumlah prediksi salah untuk kelas positif (model memprediksi negatif padahal positif).

Visualisasi confusion matrix menggunakan *heatmap* memberikan representasi yang lebih jelas tentang distribusi prediksi model:



Hasil output diatas adalah visualisasi dari confusion matrix untuk model SVM (sumbu x mewakili prediksi model, sedangkan sumbu y mewakili kelas aktual)

Kelas "*Non-anemic*" (baris atas):
- 288: Prediksi benar untuk kelas "*Non-anemic*" (True Negatives, TN).
- 227: Prediksi salah untuk kelas "*Non-anemic*" (False Positives, FP).

Kelas "*Anemic*" (baris bawah):
- 92: Prediksi salah untuk kelas "*Anemic*" (False Negatives, FN).
- 677: Prediksi benar untuk kelas "*Anemic*" (True Positives, TP).

Warna di dalam plot menunjukkan jumlah prediksi, warna yang lebih gelap menunjukkan nilai yang lebih tinggi yang dapat membantu dalam mengidentifikasi seberapa baik model dalam mengklasifikasikan setiap kelas.

Selanjutnya, dilakukan pengecekan klasifikasi report.

```
# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=['Non-anemic', 'Anemic']))
```

Hasil laporan klasifikasi:

```
Classification Report:
              precision    recall  f1-score   support

  Non-anemic       0.76      0.56      0.64       515
     Anemic        0.75      0.88      0.81       769

    accuracy                           0.75      1284
   macro avg       0.75      0.72      0.73      1284
weighted avg       0.75      0.75      0.74      1284
```

Berdasarkan laporan di atas, dapat dianalisis:

1. Precision:
    a. *Non-anemic*: 0.76 (76%) - Ketika model memprediksi kelas *Non-anemic*, prediksi tersebut benar 76% dari waktu.
    b. *Anemic*: 0.75 (75%) - Ketika model memprediksi kelas *Anemic*, prediksi tersebut benar 75% dari waktu.
2. Recall:
    a. *Non-anemic*: 0.56 (56%) - Model berhasil mengidentifikasi 56% dari total kasus *Non-anemic* yang sebenarnya.
    b. *Anemic*: 0.88 (88%) - Model berhasil mengidentifikasi 88% dari total kasus *Anemic* yang sebenarnya.
3. F1-score:
    a. *Non-anemic*: 0.64 - Rata-rata harmonik dari precision dan recall untuk kelas *Non-anemic*.
    b. *Anemic*: 0.81 - Rata-rata harmonik dari precision dan recall untuk kelas *Anemic*.
4. Support:
    a. *Non-anemic*: 515 sampel
    b. *Anemic*: 769 sampel

Model ini menunjukkan performa yang baik pada kelas *anemic* dengan recall tinggi (0.88), tetapi kurang optimal pada kelas *non-anemic* dengan recall yang lebih rendah (0.56).

3.4.2 Hyperparameter Tuning Model

Untuk meningkatkan performa model, teknik hyperparameter tuning dilakukan menggunakan *GridSearchCV*. Proses ini bertujuan untuk menemukan kombinasi parameter yang memberikan kinerja terbaik:

```
# Hyperparameter Tuning dengan GridSearchCV
print("\n--- Performing Hyperparameter Tuning ---")
param_grid = {
    'C': [0.1, 1],
    'gamma': ['scale'],
    'kernel': ['rbf']
}

grid_search = GridSearchCV(
    SVC(random_state=42),
    param_grid,
    cv=3,
    verbose=1,
    n_jobs=1
)

grid_search.fit(X_train_scaled, y_train)
```

Hasil tuning menunjukkan:

```
--- Performing Hyperparameter Tuning ---
Fitting 3 folds for each of 2 candidates, totalling 6 fits
▸ GridSearchCV
▸ estimator: SVC
    ▸ SVC
```

Proses tuning hyperparameter melibatkan:

- *3 folds*: Data dibagi menjadi tiga bagian (folds) untuk validasi silang. Setiap fold akan digunakan sebagai data pengujian satu kali, sementara dua fold lainnya digunakan untuk pelatihan.
- *2 candidates*: Ada dua kombinasi atau set parameter yang sedang dievaluasi.
- *Totalling 6 fits*: Jumlah total percobaan pelatihan yang dilakukan adalah 6, dihitung sebagai 3 folds dikali 2 candidates.

Parameter terbaik yang diperoleh dari proses tuning adalah:

```
# Best parameters & Best Model accuracy
print("\nBest Parameters:")
print(grid_search.best_params_)
```

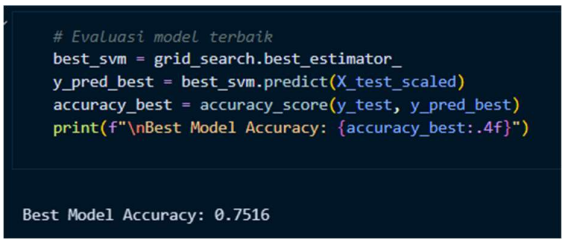
```
Best Parameters:
{'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
```

Parameter tersebut dapat diinterpretasikan sebagai:

- C=1: Parameter ini mengontrol *trade-off* antara mendapatkan margin yang lebih besar dan menghindari kesalahan klasifikasi.
- gamma='scale': Parameter ini mengatur seberapa jauh pengaruh dari satu titik data terhadap titik data lainnya.
- kernel='rbf': Parameter ini menentukan jenis fungsi kernel yang digunakan dalam model SVM. Kernel RBF (*Radial Basis Function*) efektif untuk data non-linear, karena dapat menangkap hubungan yang kompleks antara fitur-fitur dalam data.

3.4.3 Evaluasi Model Terbaik

Setelah mendapatkan parameter terbaik, model SVM dilatih kembali dengan parameter tersebut:

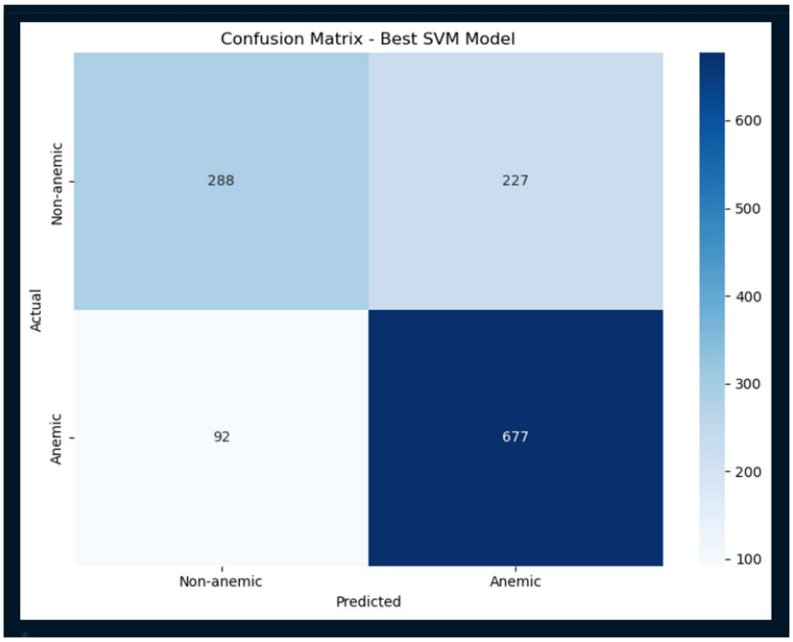


```
# Evaluasi model terbaik
best_svm = grid_search.best_estimator_
y_pred_best = best_svm.predict(X_test_scaled)
accuracy_best = accuracy_score(y_test, y_pred_best)
print(f"\nBest Model Accuracy: {accuracy_best:.4f}")
```

```
Best Model Accuracy: 0.7516
```

Hasil akurasi model terbaik tetap 0.7516 atau 75.16%, yang mengindikasikan bahwa parameter default sudah cukup optimal untuk kasus ini.

Hasil *Confusion Matrix* setelah hyperparameter tuning juga sama dengan hasil *confusion matrix* pada evaluasi awal.

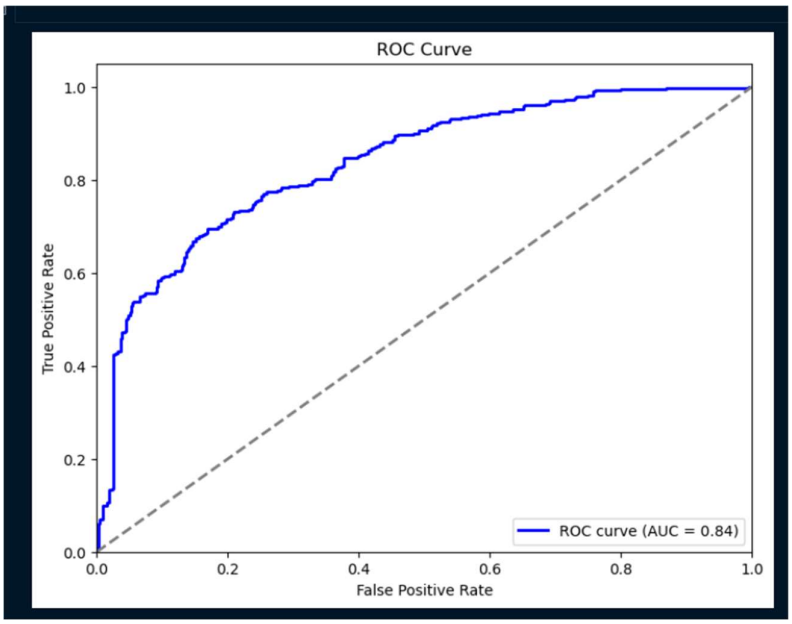


3.4.4 Kurva ROC

Untuk analisis lebih lanjut, kurva ROC *(Receiver Operating Characteristic)* dan nilai AUC *(Area Under Curve)* dihitung:

```
# Prediksi probabilitas untuk ROC
y_scores = best_svm.decision_function(X_test_scaled)
fpr, tpr, thresholds = roc_curve(y_test, y_scores)
roc_auc = auc(fpr, tpr)

# Plot kurva ROC
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.savefig('roc_curve.png')
plt.show()
```

Hasil analisis ROC menunjukkan:



- AUC = 0.84: Nilai ini menunjukkan model memiliki kemampuan yang baik dalam membedakan antara kelas positif dan negatif.
- Kurva ROC: Kurva ini menggambarkan hubungan antara True Positive Rate (sensitivitas) dan False Positive Rate (1-spesifisitas) pada berbagai ambang batas klasifikasi. Kurva yang mendekati sudut kiri atas menunjukkan model yang lebih baik.

Nilai AUC berkisar antara 0 hingga 1:

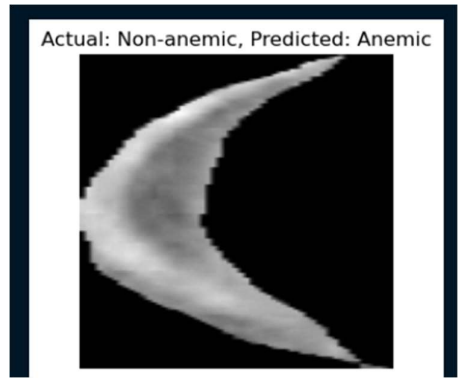
- Nilai 1: model sempurna
- Nilai 0.5: model tidak lebih baik dari tebakan acak
- Nilai 0.84: model memiliki kemampuan yang baik dalam membedakan antara kelas positif dan negatif.

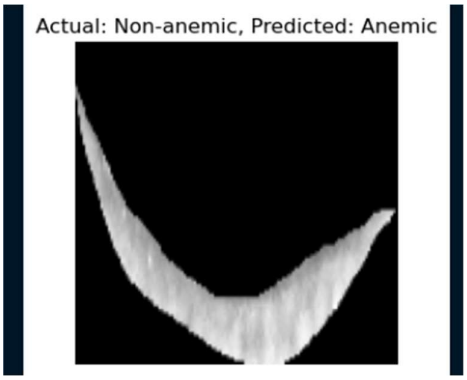
3.4.5 Analisis Kesalahan

Setelah evaluasi model menggunakan metrik seperti akurasi, *precision, recall*, dan *confusion matrix*, analisis kesalahan dilakukan untuk menilai *misclassifications* atau kesalahan prediksi.

```python
# Analisis Kesalahan Klasifikasi
misclassified_indices = np.where(y_test != y_pred_best)[0]
print(f"\nJumlah sampel yang salah diklasifikasi: {len(misclassified_indices)}")

# Jika ada gambar yang salah diklasifikasi, tampilkan beberapa sebagai contoh
if len(misclassified_indices) > 0:
    # Mendapatkan indeks asli dari data test
    test_indices = np.arange(len(X_test))

    # Jika ingin menampilkan gambar yang salah klasifikasi
    # Catatan: perlu dikembalikan ke bentuk gambar asli
    n_samples = min(5, len(misclassified_indices))
    plt.figure(figsize=(15, 3*n_samples))

    for i, idx in enumerate(misclassified_indices[:n_samples]):
        img_flat = X_test[idx]
        img_2d = img_flat.reshape(128, 128)  # Sesuaikan dengan ukuran gambar saat preprocessing

        actual = "Anemic" if y_test[idx] == 1 else "Non-anemic"
        predicted = "Anemic" if y_pred_best[idx] == 1 else "Non-anemic"

        plt.subplot(n_samples, 1, i+1)
        plt.imshow(img_2d, cmap='gray')
        plt.title(f"Actual: {actual}, Predicted: {predicted}")
        plt.axis('off')

    plt.tight_layout()
    plt.savefig('misclassified_samples.png')
    plt.show()
```

Dari 1.284 sampel pengujian, terdapat 319 sampel yang salah diklasifikasikan oleh model, dengan sebagian besar kesalahan terjadi pada gambar yang seharusnya masuk ke dalam kategori "*Non-anemic*" tetapi diprediksi sebagai "*Anemic*". Beberapa contoh gambar yang salah diklasifikasikan ditampilkan di bawah ini, dengan label "Actual" menunjukkan kategori yang benar, dan "Predicted" menunjukkan hasil prediksi model.





Model menunjukkan akurasi 75.16%, namun mengalami kesalahan klasifikasi, terutama pada kelas "*Non-anemic*". Hal ini kemungkinan disebabkan oleh ketidakseimbangan data (lebih banyak gambar "*Anemic*" dibandingkan "*Non-anemic*") dan fitur yang mirip antara kedua kelas. Langkah yang dapat diambil adalah meningkatkan data kelas minoritas atau menggunakan teknik yang lebih efektif untuk mengatasi ketidakseimbangan ini.

3.5 Dokumentasi Github

Dokumentasi lengkap mengenai proyek ini, termasuk kode sumber, dataset, dan instruksi penggunaan, dapat diakses melalui link berikut:
https://github.com/FitriaaN/PrakML_miniproject.git

BAB IV

PENUTUP

Dalam proyek ini, telah dilakukan klasifikasi gambar menggunakan algoritma *Support Vector Machine* (SVM) untuk membedakan gambar antara kategori *anemic* dan *non-anemic*. Berdasarkan hasil evaluasi, model yang diterapkan menunjukkan akurasi 75.16%, dengan kinerja yang lebih baik pada kelas "*Anemic*" (recall 0.88), meskipun performa pada kelas "*Non-anemic*" (recall 0.56) masih kurang optimal.

Analisis kesalahan mengungkapkan bahwa ketidakseimbangan jumlah data antara kedua kelas dan kesamaan fitur pada beberapa gambar menjadi tantangan utama dalam klasifikasi. Meskipun demikian, hasil ini menunjukkan bahwa SVM dapat menjadi alat yang efektif dalam tugas klasifikasi gambar medis. Sebagai langkah perbaikan, perlu dilakukan teknik pengolahan data lebih lanjut, seperti oversampling pada kelas minoritas dan eksperimen dengan berbagai teknik tuning hyperparameter untuk meningkatkan performa model, terutama pada kelas dengan data lebih sedikit. Secara keseluruhan, proyek ini memberikan pemahaman yang lebih dalam.

DAFTAR PUSTAKA

[1] Papers with Code, “Image Classification.” [Online]. Available: https://paperswithcode.com/task/image-classification. [Accessed: 21-Apr-2025].

[2] ITBox, “Support Vector Machine: Definisi, Metode, dan Cara Kerja,” 29-Mar-2022. [Online]. Available: https://itbox.id/blog/support-vector-machine-definisi-metode-dan-cara-kerja/. [Accessed: 21-Apr-2025].

[3] Kili Technology, “What is Image Classification?,” 2023. [Online]. Available: https://kili-technology.com/data-labeling/computer-vision/image-annotation/what-is-image-classification. [Accessed: 21-Apr-2025].

[4] Google Cloud, “Supervised vs. Unsupervised Learning.” [Online]. Available: https://cloud.google.com/discover/supervised-vs-unsupervised-learning?hl=id. [Accessed: 21-Apr-2025].

[5] RevoU, “EDA (Exploratory Data Analysis).” [Online]. Available: https://www.revou.co/kosakata/eda. [Accessed: 21-Apr-2025].

[6] Program Studi Manajemen Informatika BINUS University, “Teknik Pre-Processing dan Classification dalam Data Science,” 26-Aug-2022. [Online]. Available: https://mie.binus.ac.id/2022/08/26/teknik-pre-processing-dan-classification-dalam-data-science/. [Accessed: 21-Apr-2025].

[7] E. Samsudin, “Penjelasan Sederhana Tentang Apa itu SVM,” 08-May-2020. [Online]. Available: https://medium.com/@samsudiney/penjelasan-sederhana-tentang-apa-itu-svm-149fec72bd02. [Accessed: 21-Apr-2025].