Tugas Akhir Pengenalan Pola

Klasifikasi Detak Jantung menggunakan data EKG

Dengan MFCC dan metode Support Vektor Machine

Disusun Oleh:

Ida Bagus Made Surya Widnyana 2008561092

PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS UDAYANA

DESEMBER

2023

BAB I PEMBAHASAN

1.1 Langkah-Langkah Program

Pada penelitian ini, dilakukan klasifikasi jenis-jenis detak jantung dengan data EKG, yaitu normal, extrastole, dan murmur, menggunakan dataset Heartbeat Sounds yang terdiri dari dua sumber, A dan B. Dataset ini mencakup file audio dalam format wav yang bervariasi durasinya. Melalui pengumpulan data dari aplikasi iStethoscope Pro iPhone (Set A) dan uji klinis rumah sakit dengan stetoskop digital DigiScope (Set B), tujuan utama penelitian adalah mengklasifikasikan detak jantung secara otomatis. Pendekatan yang digunakan melibatkan metode MFCC (Mel-Frequency Cepstral Coefficients) untuk ekstraksi fitur suara, dan SVM (Support Vector Machine) sebagai model klasifikasi. Metode ini dirancang untuk memanfaatkan karakteristik unik dari suara detak jantung guna mencapai akurasi klasifikasi yang optimal.

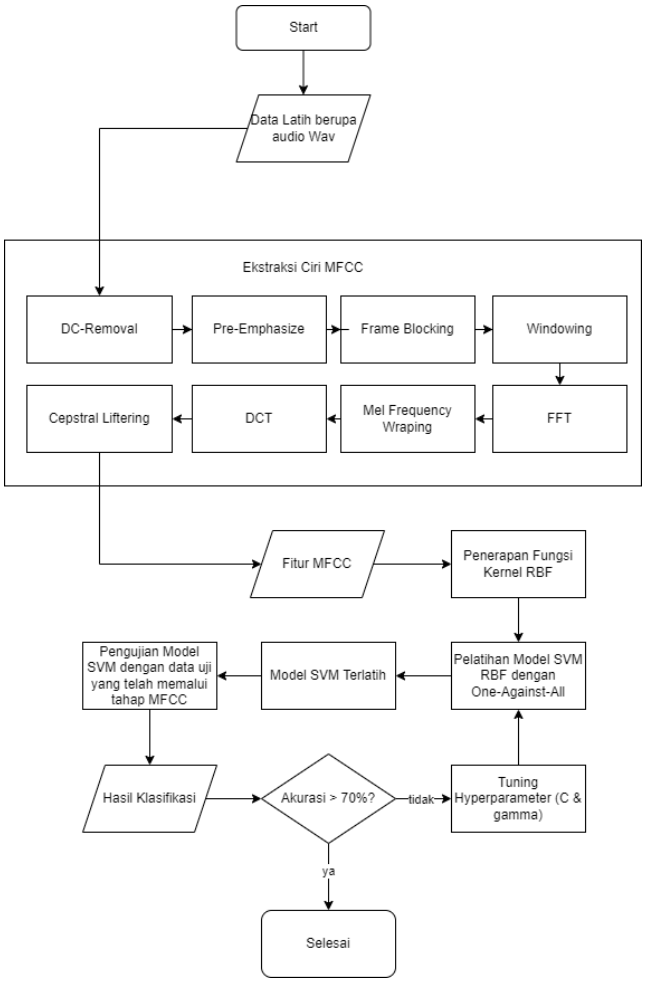
1.1.1 Pengumpulan data

Dataset penelitian ini diperoleh dari Heartbeat Sounds, menggabungkan data dari aplikasi iStethoscope Pro iPhone yang dikumpulkan dari masyarakat umum (Set A) dan uji klinis rumah sakit dengan stetoskop digital DigiScope (Set B). Dataset ini terbagi menjadi dua sumber, A dan B, dengan file audio dalam format wav. Durasi file berkisar antara 1 hingga 30 detik, beberapa di antaranya telah dipangkas untuk mengurangi kebisingan dan memusatkan perhatian pada fragmen yang signifikan dari suara. Inspirasi dari dataset ini adalah untuk secara otomatis memisahkan detak jantung normal dan abnormal, termasuk suara extrastole dan murmur. Tujuan utama penelitian ini adalah mengidentifikasi lokasi suara jantung dan mengklasifikasikannya, menjadikan dataset ini sebagai sumber potensial untuk pengembangan pemantauan kesehatan kardiovaskular melalui teknik machine learning. Dalam konteks ini, data latih sebanyak 70% atau setara dengan 150 file audio dipilih, sedangkan data uji sebanyak 30% atau setara dengan 45 file audio untuk memastikan representasi yang baik dalam pengujian model.

Tabel 1.1 Dataset Penelitian

Label	Data latih	Data Uji
murmur	50	15
normal	50	15
extrastole	50	15
total	150	45

1.1.2 Alur Program



Gambar 1.1 Alur Program

1. Library

   Library yang digunakan pada penelitian ini adalah

```
import os
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
import librosa
from scipy.fftpack import dct
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
import pandas as pd
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV
```

2. Metode MFCC

Fungsi ekstraksi MFCC (Mel-Frequency Cepstral Coefficients) ini merupakan serangkaian langkah untuk mendapatkan representasi fitur khas dari suara detak jantung yang direkam dalam file audio WAV.

- DC Removal
  Tujuan dari proses DC Removal adalah untuk menghitung nilai rata-rata dari data sampel suara dan kemudian mengurangi nilai masing-masing sampel suara dengan nilai rata-rata tersebut.

```
def dc_removal(signal):
    mean = np.mean(signal)
    return signal - mean
```

- Pre-Emphasize
  Proses pre-emphasis memfilter sinyal suara yang masuk, mengurangi frekuensi sinyal sehingga hanya sinyal berfrekuensi tinggi yang dapat melewati filter. Tujuan dari proses ini adalah untuk mengurangi noise dari suara, sehingga sistem hanya dapat menangkap data sinyal suara yang sebenarnya.

```
def pre_emphasis(signal):
    pre_emphasis = 0.97
    signal_emphasis = np.append(signal[0], signal[1:] -
pre_emphasis * signal[:-1])
    return signal_emphasis
```

- Frame Blocking
  Proses frame blocking akan memotong sampel suara menjadi frame yang lebih pendek. Karena sampel suara memiliki sinyal suara yang tidak stabil, akan sulit untuk mengidentifikasi karakteristik suara. Namun, dengan memotong sampel menjadi bagian-bagian yang lebih kecil, sinyal suara yang ada akan lebih stabil, yang membuatnya lebih mudah untuk mengidentifikasi karakteristik suara. Untuk menghindari informasi yang hilang selama proses berikutnya, frame yang dibuat akan berlubang dengan frame lainnya.

```
def frame_blocking(signal_emphasis, sample_rate):
    frame_size = 0.025
    frame_stride = 0.01
    frame_length = int(frame_size * sample_rate)
    frame_step = int(frame_stride * sample_rate)
    signal_length = len(signal_emphasis)
    frames_overlap = frame_length - frame_step

    num_frames = (np.abs(signal_length - frames_overlap)
// np.abs(frame_length - frame_step)).astype(int)
    rest_samples = np.abs(signal_length -
frames_overlap) % np.abs(frame_length - frame_step)
```

```
    pad_signal_length = int(frame_length - rest_samples)
    z = np.zeros((pad_signal_length))
    pad_signal = np.append(signal_emphasis, z)

    indices = np.tile(np.arange(0, frame_length),
(num_frames, 1)) + np.tile(
        np.arange(0, num_frames * frame_step,
frame_step), (frame_length, 1)
    ).T

    frames = pad_signal[indices.astype(np.int32,
copy=False)]
    return frames, frame_length
```

- Windowing
  Proses windowing dilakukan pada setiap frame untuk mengurangi diskontinyu sinyal di awal dan akhir frame.

```
def windowing(frames, frame_length):
    frames = frames * (np.hamming(frame_length))
    return frames
```

- FFT
  Konversi dari domain waktu frekuensi diperlukan karena sinyal suara saat ini masih berada dalam domain waktu selama proses. Untuk mendapatkan sinyal dalam domain frekuensi dari sebuah sinyal diskrit, Fast Fourier Transform (FFT) adalah algoritma perhitungan Discrete Fourier Transform (DFT) yang efisien. Oleh karena itu, proses perhitungan Discrete Fourier Transform (DFT) akan menjadi lebih cepat, terutama ketika nilai yang digunakan cukup besar. FFT diterapkan pada setiap frame sinyal yang telah diwindowing.

```
def fft(frames):
    NFFT = 512
    mag_frames = np.absolute(np.fft.rfft(frames, NFFT))
    pow_frames = ((1.0 / NFFT) * ((mag_frames) ** 2))
    return pow_frames, NFFT
```

- Mel Frequency Wraping dan DCT
  Proses ini melakukan filtering dari spektrum setiap frame yang telah dikumpulkan dari tahap sebelumnya. Sebelum proses filtering dimulai, batas atas dan bawah ditetapkan untuk memastikan bahwa nilai yang berada di luar batas tidak akan difilter. Kemudian, kedua batas tersebut diubah ke dalam skala mel. Dalam proses ini, nilai ceptrum MFCC yang masih berada dalam domain frekuensi harus diubah menjadi domain waktu karena ciri akan mengacu pada urutan waktu. Akibatnya, mel frekuensi harus diubah kembali ke domain waktu untuk mendapatkan nilai ceptrum MFCC yang masih berada dalam domain frekuensi.

```
def filter_bank(pow_frames, sample_rate, NFFT):
    nfilt = 40
    low_freq_mel = 0
```

```
    high_freq_mel = (2595 * np.log10(1 + (sample_rate /
2) / 700))  # Convert Hz to Mel
    mel_points = np.linspace(low_freq_mel,
high_freq_mel, nfilt + 2)  # Equally spaced in Mel scale
    hz_points = (700 * (10 ** (mel_points / 2595) -
1))  # Convert Mel to Hz
    bin = np.floor((NFFT + 1) * hz_points / sample_rate)

    fbank = np.zeros((nfilt, int(np.floor(NFFT / 2 +
1))))
    for m in range(1, nfilt + 1):
        f_m_minus = int(bin[m - 1])  # left
        f_m = int(bin[m])  # center
        f_m_plus = int(bin[m + 1])  # right

        for k in range(f_m_minus, f_m):
            fbank[m - 1, k] = (k - bin[m - 1]) / (bin[m]
- bin[m - 1])
        for k in range(f_m, f_m_plus):
            fbank[m - 1, k] = (bin[m + 1] - k) / (bin[m
+ 1] - bin[m])

    filter_banks = np.dot(pow_frames, fbank.T)
    filter_banks = np.where(filter_banks == 0,
np.finfo(float).eps, filter_banks)  # Numerical
Stability
    filter_banks = 20 * np.log10(filter_banks)  # dB

    return (filter_banks / np.amax(filter_banks)) * 255
```

- Cepstral Liftering
  Proses ini meningkatkan kualitas pengenalan ekstraksi ciri. Ini disebabkan oleh fakta bahwa cepstral coefficient sangat sensitif terhadap slope spektral pada order low, sementara sangat sensitif terhadap noise pada order high.

```
def cepstral_liftering(filter_banks):
    num_ceps = 12
    cep_lifter = 11
    mfcc = dct(filter_banks, type=2, axis=1,
norm='ortho')[:, :(num_ceps)]
    (nframes, ncoeff) = mfcc.shape
    n = np.arange(ncoeff)
    lift = 1 + (cep_lifter / 2) * np.sin(np.pi * n /
cep_lifter)
    mfcc = (np.mean(mfcc, axis=0) + 1e-8)
    return mfcc
```

- Fungsi MFCC

  Seluruh proses ini diimplementasikan dalam fungsi-fungsi seperti dc_removal, pre_emphasis, frame_blocking, windowing, fft, filter_bank, dan cepstral_liftering. Fungsi extract_mfcc kemudian menggabungkan langkah-langkah ini untuk mengekstrak fitur MFCC dari file audio detak jantung dalam format WAV.

```
def extract_mfcc(file_path):
    try:
        # Baca file audio WAV
        audio_data, sample_rate =
librosa.load(file_path, sr=None)

        # DC Removal
        dc_removed_signal = dc_removal(audio_data)

        # Pre-Emphasis
        pre_emphasized_signal =
pre_emphasis(dc_removed_signal)

        # Frame Blocking
        frames, frame_length =
frame_blocking(pre_emphasized_signal, sample_rate)

        # Windowing
        windowed_frames = windowing(frames,
frame_length)

        # FFT, Mel Frequency Wrapping, dan DCT
        pow_frames, NFFT = fft(windowed_frames)
        filter_banks = filter_bank(pow_frames,
sample_rate, NFFT)
        mfcc_features = cepstral_liftering(filter_banks)

        return mfcc_features
    except Exception as e:
        print(f"Error processing {file_path}: {e}")
        return None
```

- Menyesuaikan Panjang fitur MFCC

  Sementara itu, fungsi pad_mfcc_features digunakan untuk menyesuaikan panjang fitur MFCC dengan cara menyesuaikan atau memotong fitur-fitur tersebut sesuai kebutuhan atau batasan panjang tertentu.

```
def pad_mfcc_features(features, max_length):
    num_features = len(features)
```

```
    feature_shape = features[0].shape[1] if
len(features[0].shape) > 1 else 1

    # Menyesuaikan panjang fitur tanpa DTW
    padded_features = np.zeros((num_features,
max_length, feature_shape))

    for i, feature in enumerate(features):
        feature_length = len(feature)

        if feature_length < max_length:
            # Membuat pad dengan menggunakan nilai nol
            padded_feature = np.zeros((max_length,
feature_shape))
            padded_feature[:feature_length, :] =
feature  # Mengisi dengan fitur yang sebenarnya
            padded_features[i] = padded_feature
        else:
            # Jika panjang fitur sudah cukup, gunakan
fitur tanpa modifikasi
            padded_features[i, :feature_length, :] =
feature.reshape((feature_length, feature_shape))  #
Reshape here

    return padded_features
```

3. Mendapatkan dataset

Pada tahap ini memiliki peran utama dalam proses pengumpulan dataset untuk keperluan klasifikasi detak jantung. Fungsi ini berfungsi untuk menyusun dataset dari folder yang mengandung file audio dengan label-label khusus, seperti extrastole, normal, dan murmur. Setiap file audio dalam folder tersebut diolah menggunakan fungsi extract_mfcc, yang bertanggung jawab untuk menghasilkan fitur-fitur Mel-Frequency Cepstral Coefficients (MFCC) dari setiap file audio. Fitur-fitur MFCC tersebut kemudian disertakan ke dalam dataset bersama dengan label yang sesuai, menciptakan dataset yang siap digunakan untuk melatih dan menguji model klasifikasi detak jantung. Dengan langkah-langkah ini, dataset menjadi dasar penting untuk pengembangan model klasifikasi dengan menggunakan teknik machine learning SVM.

```
def get_dataset(base_folder, labels):
    # Mendapatkan dataset
    dataset = {"features": [], "labels": []}

    for label in labels:
        folder_path = os.path.join(base_folder, label)
        files = os.listdir(folder_path)
```

```
        for file in files:
            file_path = os.path.join(folder_path, file)
            features = extract_mfcc(file_path)

            if features is not None:
                dataset["features"].append(features)
                dataset["labels"].append(label)

    return dataset
```

4. Support Vector Machine Model untuk data latih

   Pada tahap ini pelatihan model klasifikasi detak jantung menggunakan fitur Mel-frequency Cepstral Coefficients (MFCC) dan Support Vector Machine (SVM), pertama-tama dataset latih dikumpulkan dari folder yang berisi file audio dengan label extrastole, normal, dan murmur. Setiap file audio diubah menjadi fitur-fitur MFCC, dan panjang fitur dinormalisasi tanpa Dynamic Time Warping (DTW). Padding ditambahkan pada setiap sampel untuk menyamakan panjang fitur. Matriks fitur dua dimensi diubah menjadi satu dimensi, kemudian fitur-fitur MFCC dinormalisasi menggunakan metode StandardScaler.

   Model klasifikasi dibuat dengan menggunakan SVM melalui pipeline, diatur dengan kernel radial basis function (RBF) dan parameter khusus serta menggunakan metode One-Against-All. Selanjutnya, model dilatih menggunakan data latih yang telah diproses, dan informasi penting terkait pelatihan, seperti jenis kernel yaitu RBF, nilai C yaitu 20, dan nilai gamma yaitu 0.01, dicetak untuk evaluasi. Dengan menjalankan langkah-langkah ini, model siap untuk mengklasifikasikan jenis detak jantung pada data uji.

```
if __name__ == "__main__":
    base_folder = "/content/drive/MyDrive/Dataset EKG /Data
Latih EKG"

    labels = ["extrastole", "normal", "murmur"]
    print(f"Analisis Kinerja MFCC dan SVM untuk Klasifikasi
Aritmia Jantung")
    # Mendapatkan dataset
    dataset = get_dataset(base_folder, labels)

    # Normalisasi panjang fitur MFCC tanpa DTW
    max_length = max(len(feature) for feature in
dataset["features"])

    # Sisipkan padding pada setiap sampel
    padded_features = pad_mfcc_features(dataset["features"],
max_length)
```

```
    # Ubah matriks dua dimensi
    num_samples, max_length, num_features =
padded_features.shape
    reshaped_features =
padded_features.reshape((num_samples, max_length *
num_features))

    # Normalisasi fitur MFCC menggunakan StandardScaler
    scaler = StandardScaler()
    reshaped_features_scaled =
scaler.fit_transform(reshaped_features)

    # Menggunakan pipeline untuk menggabungkan normalisasi
dan klasifikasi
    model = make_pipeline(StandardScaler(),
OneVsRestClassifier(SVC(kernel='rbf', C=20, gamma=0.01)))

    # Melatih model
    model.fit(reshaped_features_scaled, dataset["labels"])

    print("\n----------------Model Training Information-----
----------")
    print(f"Kernel:
{model.named_steps['onevsrestclassifier'].estimators_[0].ker
nel}")
    print(f"C:
{model.named_steps['onevsrestclassifier'].estimators_[0].C}"
)
    print(f"Gamma:
{model.named_steps['onevsrestclassifier'].estimators_[0].gam
ma}")
```

5. Support Vektor Machine Model untuk data uji

   Setelah memperoleh model dari data pelatihan, langkah selanjutnya adalah menguji kinerja model pada data uji menggunakan metode Support Vector Machine (SVM). Proses evaluasi dimulai dengan menginisialisasi variabel seperti correct_predictions (untuk mencatat jumlah prediksi yang benar), total_samples (untuk total sampel), serta dua list untuk menyimpan label sebenarnya (true_labels) dan label prediksi (predicted_labels). Selain itu, set unique_samples digunakan untuk menyimpan sampel-sampel unik yang telah dievaluasi. Selama iterasi pada setiap label pada dataset uji, fitur MFCC diekstraksi dan dinormalisasi panjangnya. Model SVM digunakan untuk memprediksi label pada sampel data uji, dan hasil prediksi dibandingkan dengan label sebenarnya untuk menghitung akurasi prediksi.

```
    # Evaluasi model menggunakan data uji
    correct_predictions = 0
    total_samples = 0
    true_labels = []
    predicted_labels = []

    unique_samples = set()

    for label in labels:
        test_folder =
os.path.join("/content/drive/MyDrive/Dataset EKG /Data Uji
EKG", label)
        files = os.listdir(test_folder)

        for file in files:
            test_audio_file = os.path.join(test_folder,
file)
            # print(f"Test Audio File: {test_audio_file}")

            # Ekstraksi fitur MFCC
            mfcc_features_test =
extract_mfcc(test_audio_file)

            if mfcc_features_test is not None:
                # Menampilkan audio waveform
                sample_rate, audio_data_test =
librosa.load(test_audio_file, sr=None)

                # Normalisasi panjang fitur MFCC pada data
uji
                mfcc_features_test =
pad_mfcc_features([mfcc_features_test], max_length)[0]

                # Ratakan dimensi kedua dan ketiga dari
fitur MFCC
                mfcc_features_test_flat =
mfcc_features_test.reshape((1, -1))

                # Normalisasi fitur MFCC uji menggunakan
StandardScaler
                mfcc_features_test_flat_scaled =
scaler.transform(mfcc_features_test_flat)

                # Prediksi label menggunakan model yang
telah dilatih
                recognized_label =
model.predict(mfcc_features_test_flat_scaled)
```

```
                # Uji akurasi
                total_samples += 1
                true_labels.append(label)
                predicted_labels.append(recognized_label[0])
                unique_samples.add(file)

                # Perbarui correct_predictions
                if recognized_label[0] == label:
                    correct_predictions += 1

    # Hitung total sampel unik
    total_samples = len(unique_samples)
```

6. Optimasi Dan Evaluasi Model

   Setelah semua sampel dievaluasi, dilakukan perhitungan metrik evaluasi seperti akurasi, presisi, recall, dan F1-Score. Evaluasi ini memberikan gambaran tentang kemampuan model SVM dalam mengklasifikasikan sampel-sampel dari data uji, termasuk informasi mengenai keberhasilan model dalam mengenali kelas-kelas yang berbeda. Apabila nilai akurasi yang diperoleh masih rendah maka akan dilakukan optimasi dengan melakukan tuning parameter SVM berupa nilai C (penalty) dan gamma (kernel coefficients) yang terus dilakukan hingga mendapatkan nilai akurasi terbaik.

```
   # Menghitung metrik evaluasi
    accuracy = accuracy_score(true_labels, predicted_labels)
    precision = precision_score(true_labels,
predicted_labels, average='weighted')
    recall = recall_score(true_labels, predicted_labels,
average='weighted')
    f1 = f1_score(true_labels, predicted_labels,
average='weighted')
    print(f"\n----------------Model Evaluation--------------
--")
    print(f"Total Samples: {total_samples}")
    print(f"Correct Predictions: {correct_predictions}")
    print(f"Accuracy: {accuracy * 100:.2f}%")
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1-Score: {f1:.2f}")
```

7. Simpan dataframe

   Pada tahap ini, hasil evaluasi model, termasuk label sebenarnya (True_Label) dan label prediksi (Predicted_Label) dari data uji, disimpan dalam suatu struktur data DataFrame menggunakan library pandas. Data ini kemudian diperkaya dengan menyertakan fitur-fitur MFCC yang telah dinormalisasi dan diratakan pada dimensi kedua dan ketiga (reshaped_features_scaled). DataFrame yang

telah dibentuk kemudian disimpan dalam format CSV untuk keperluan analisis lebih lanjut atau dokumentasi.

```
    df = pd.DataFrame({'True_Label': true_labels,
'Predicted_Label': predicted_labels})
    df_new = pd.concat([df,
pd.DataFrame(reshaped_features_scaled)], axis=1)
    df_new.to_csv("/content/drive/MyDrive/Dataset EKG
/df_result.csv", index=False)
```

1.2 Hasil

Dalam penelitian ini, dilakukan pengembangan model klasifikasi detak jantung menggunakan ekstraksi fitur Mel-Frequency Cepstral Coefficients (MFCC) dan metode Support Vector Machine (SVM). Penggunaan data melibatkan alokasi sebanyak 70%, atau setara dengan 150 file audio, dari dataset sebagai data latih, sementara 30%, atau setara dengan 45 file audio, diambil sebagai data uji. Proporsi ini dipilih untuk memastikan representasi yang baik dalam pelatihan dan pengujian model, dengan tujuan mencapai keseimbangan dan keberagaman data. Model ini difokuskan untuk mengklasifikasikan detak jantung ke dalam tiga kategori: normal, extrastole, dan murmur. Evaluasi model menunjukkan keberhasilan dengan akurasi mencapai 73.33%. Dari total 45 sampel pada dataset uji, sebanyak 33 prediksi dinyatakan benar. Metrik evaluasi lainnya, seperti presisi (0.76), recall (0.73), dan F1-Score (0.74), memberikan gambaran tentang kinerja model dalam mengklasifikasikan jenis detak jantung dengan tepat.

```
----------Model Evaluation----------
Total Samples: 45
Correct Predictions: 33
Accuracy: 73.33%
Precision: 0.76
Recall: 0.73
F1-Score: 0.74
```

Gambar 1.2 kinerja model

Dengan hasil perbandingan true label dan predict label sebagai berikut.

True_Label	Predicted_Label
extrastole	extrastole
extrastole	extrastole
extrastole	normal
extrastole	normal
extrastole	extrastole
extrastole	extrastole
extrastole	extrastole
extrastole	extrastole
extrastole	extrastole
extrastole	extrastole
extrastole	extrastole
extrastole	normal
extrastole	extrastole
extrastole	normal
extrastole	normal
normal	extrastole
normal	normal
normal	normal
normal	normal
normal	normal
normal	normal
normal	normal
normal	extrastole
normal	normal
normal	normal
normal	normal
normal	normal
normal	extrastole
normal	normal
normal	normal
murmur	extrastole
murmur	murmur
murmur	murmur
murmur	murmur
murmur	murmur
murmur	murmur
murmur	normal
murmur	murmur
murmur	murmur
murmur	murmur
murmur	murmur
murmur	murmur
murmur	extrastole
murmur	extrastole
murmur	murmur

Gambar 1.3 Perbandingan true label dan predict label

1.3 Analisis

Dalam menganalisis hasil evaluasi model klasifikasi detak jantung berdasarkan ekstraksi fitur Mel-Frequency Cepstral Coefficients (MFCC) dan metode Support Vector Machine (SVM), beberapa temuan penting muncul. Model ini, setelah melibatkan 70% dari dataset sebagai data latih dan 30% sebagai data uji, berhasil memprediksi detak jantung dengan tingkat akurasi sebesar 73.33%. Dari total 45 sampel pada dataset uji, sebanyak 33 prediksi dinyatakan benar. Pencapaian ini menunjukkan kemampuan model dalam mengklasifikasikan detak jantung ke dalam tiga kategori: normal, extrastole, dan murmur. Meskipun kinerja model umumnya baik, terdapat beberapa sampel extrastole yang salah diklasifikasikan sebagai normal, mengindikasikan potensi untuk meningkatkan prediksi pada kategori ini. Dalam melakukan analisis, terlihat bahwa konfigurasi hyperparameter model (C=20, gamma=0.01) memberikan hasil yang seimbang antara presisi dan recall.

Pemahaman ini dapat digunakan sebagai dasar untuk eksplorasi lebih lanjut terhadap peningkatan model, seperti penyesuaian hyperparameter lebih lanjut atau penanganan khusus pada kesalahan yang mungkin terjadi. Evaluasi ini memberikan gambaran menyeluruh tentang kinerja model, dan melalui analisis lebih lanjut, dapat ditemukan strategi untuk meningkatkan keakuratannya.

1.4 Kesimpulan

Penelitian ini bertujuan untuk mengembangkan model klasifikasi detak jantung menggunakan ekstraksi fitur Mel-Frequency Cepstral Coefficients (MFCC) dan metode Support Vector Machine (SVM). Dalam penelitian ini, data latih sebanyak 70% dari total dataset digunakan untuk melatih model, sedangkan 30% sisanya diambil sebagai data uji. Proporsi ini dipilih untuk memastikan representasi yang baik dalam pelatihan dan pengujian model. Model ini difokuskan pada klasifikasi detak jantung ke dalam tiga kategori utama: normal, extrastole, dan murmur. Hasil evaluasi model menunjukkan tingkat akurasi sebesar 73.33%, di mana dari total 45 sampel pada dataset uji, sebanyak 33 prediksi dinyatakan benar. Selain akurasi, metrik evaluasi lainnya seperti presisi, recall, dan F1-Score memberikan gambaran tentang kinerja model dalam mengklasifikasikan jenis detak jantung dengan tepat. Melalui analisis nilai hyperparameter, konfigurasi model dengan C=20 dan gamma=0.01 memberikan hasil yang seimbang antara presisi dan recall. Namun, terdapat beberapa sampel yang salah diklasifikasikan, terutama pada kategori extrastole yang cenderung diidentifikasi sebagai detak jantung normal. Model ini menunjukkan potensi untuk mendukung diagnosis aritmia jantung dengan tingkat akurasi yang relatif tinggi. Namun, perlu dilakukan peningkatan lebih lanjut pada model, seperti penanganan khusus pada jenis kesalahan tertentu atau eksplorasi lebih lanjut terhadap hyperparameter, guna memperbaiki kinerja model dan membuatnya lebih dapat diandalkan dalam praktik klinis.