**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

## 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.

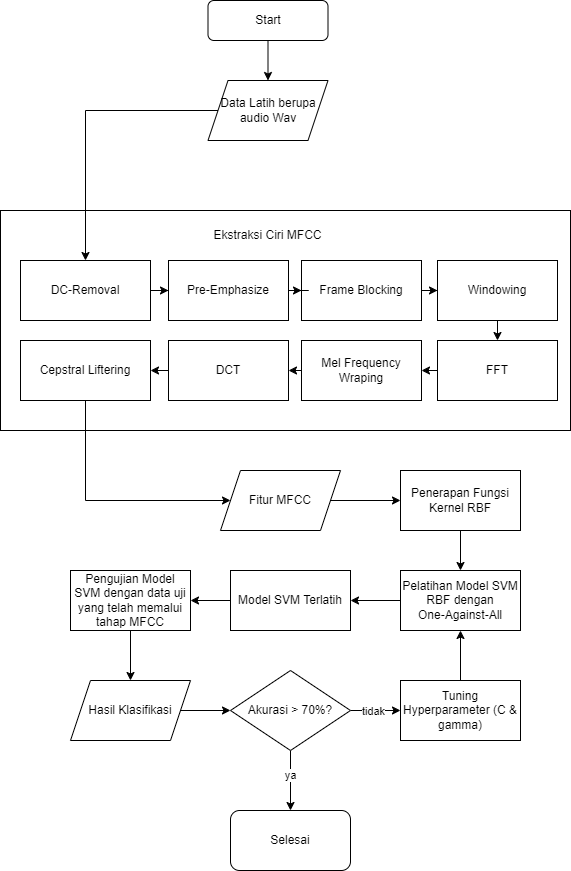
## 1.1.1 Pengumpulan data

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

1. 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

1. 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

1. 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].kernel}")

    print(f"C: {model.named\_steps['onevsrestclassifier'].estimators\_[0].C}")

    print(f"Gamma: {model.named\_steps['onevsrestclassifier'].estimators\_[0].gamma}")

1. 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)

1. 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}")

1. 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.

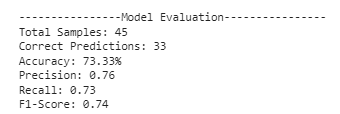
    df = pd.DataFrame({'True\_Label': true\_labels, 'Predicted\_Label': predicted\_labels})

    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)

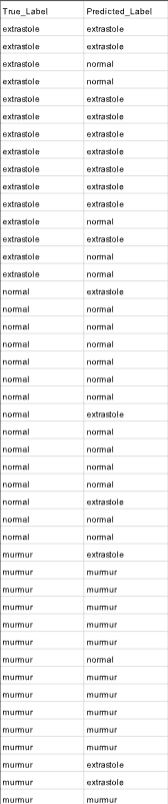
* 1. Hasil

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.



Gambar 1.2 kinerja model

Dengan hasil perbandingan true label dan predict label sebagai berikut.



Gambar 1.3 Perbandingan true label dan predict label

* 1. 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. 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.

### 

# 

# 