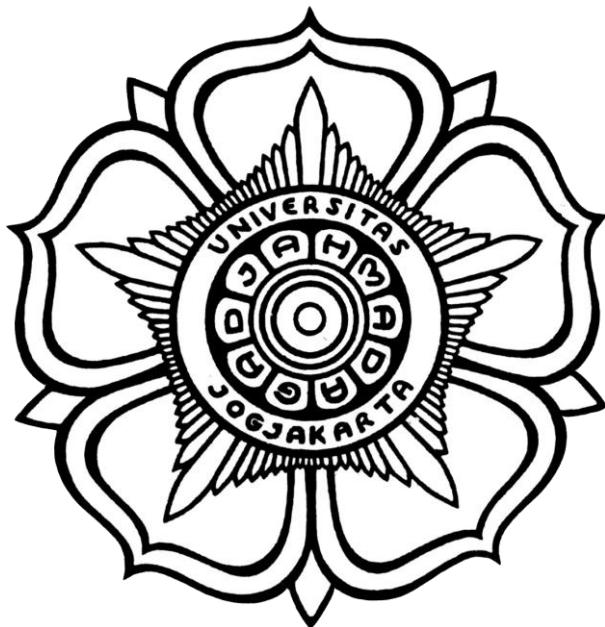


**LAPORAN KLASIFIKASI AUDIO MENGGUNAKAN MFCC DAN
MODEL MACHINE LEARNING**

LAPORAN TUGAS PENGENALAN POLA



Oleh:
Pramitha Witawacita Astadewi – 24/554318/NPA/19979

**PROGRAM STUDI ELEKTRONIKA DAN INSTRUMENTASI
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2025**

DAFTAR ISI

LAPORAN KLASIFIKASI AUDIO MENGGUNAKAN MFCC DAN MODEL MACHINE LEARNING	1
1. Pendahuluan	3
2. Deskripsi Dataset	3
3. Ekstraksi Fitur.....	4
3.1 Fungsi extract_mfcc	4
4. Persiapan Data dan Label	5
4.1 Membaca dan Ekstraksi Fitur dari Dataset.....	5
4.2 Pengkodean Label dan Downsampling Data	6
4.3 Split Data dan Reshape	7
5. Visualisasi Distribusi Kelas	8
6. Pelatihan dan Evaluasi Model	9
6.1 Pelatihan Model.....	9
6.2 Perbandingan Hasil Evaluasi (Recall & F1-Score).....	13
7. Prediksi pada File Audio Acak	15
Fungsi predict_audio.....	15
8. Kesimpulan	17
9. Penutup.....	17

1. Pendahuluan

Laporan ini menjelaskan proses pengolahan dan klasifikasi sinyal audio menggunakan fitur Mel-Frequency Cepstral Coefficients (MFCC). Dataset yang digunakan merupakan infant cry audio corpus yang telah diunduh dari Kaggle. Tiga model klasifikasi – K-Nearest Neighbors (KNN), Naive Bayes, dan Support Vector Machine (SVM) – digunakan untuk mengenali kelas-kelas suara (misalnya: *hungry, discomfort, burping, tired, belly_pain*). Laporan ini mencakup penjelasan kode, ruang untuk output, dan analisis dari setiap fungsi yang diimplementasikan.

Dataset: <https://www.kaggle.com/datasets/warcoder/infant-cry-audio-corpus/code>

2. Deskripsi Dataset

- **Sumber Dataset:** Dataset diunduh dari [Kaggle Infant Cry Audio Corpus](#) dan disimpan di direktori `"/content/drive/MyDrive/datasets/"`.
- **Struktur Folder:** Dataset tersusun dalam subfolder yang masing-masing merepresentasikan label (contoh: `hungry, discomfort, burping, tired, belly_pain`).
- **Format File:** File audio berekstensi `.wav`.
- **Proses Downsampling:** Untuk mengatasi ketidakseimbangan jumlah data per kelas, dilakukan downsampling sehingga diperoleh 27 sampel per kelas.

Output Dataset:

Class: hungry - 382 file(s)

Class: belly_pain - 16 file(s)

Class: tired - 24 file(s)

Class: burping - 8 file(s)

Class: discomfort - 27 file(s)

Class: tired - 24 file(s).

3. Ekstraksi Fitur

3.1 Fungsi extract_mfcc

```
# MFCC extraction function

def extract_mfcc(file_path, n_mfcc=100, fixed_length=100):

    try:

        # Read audio file

        y, sr = librosa.load(file_path, sr=None)

        # Extract MFCC

        mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc)

        # Padding or trimming to a fixed length

        if mfcc.shape[1] < fixed_length:

            pad_width = fixed_length - mfcc.shape[1]

            mfcc = np.pad(mfcc, pad_width=((0, 0), (0, pad_width)), mode='constant')

        else:

            mfcc = mfcc[:, :fixed_length]

    return mfcc

except Exception as e:

    print(f"Error processing {file_path}: {e}")

    return None
```

Penjelasan:

- **Tujuan:** Mengambil fitur MFCC dari file audio.
- **Proses:**
 - File audio dibaca menggunakan `librosa.load`.
 - Fitur MFCC diekstrak dengan jumlah koefisien yang ditentukan (default 100).

- Jika jumlah frame kurang dari `fixed_length` (default 100), dilakukan padding dengan nol; jika lebih, dilakukan trimming.
 - **Output:** Matriks MFCC dengan dimensi tetap untuk setiap file audio.
-

4. Persiapan Data dan Label

4.1 Membaca dan Ekstraksi Fitur dari Dataset

```
import os
import librosa
import numpy as np

base_dir = "/content/drive/MyDrive/datasets/donateacry_corpus"

mfcc_list = []
labels = []

fixed_length = 100

for subdir, _, files in os.walk(base_dir):
    for file in files:
        if file.endswith(".wav"):
            file_path = os.path.join(subdir, file)
            mfcc_features = extract_mfcc(file_path,
fixed_length=fixed_length)
            if mfcc_features is not None:
                mfcc_list.append(mfcc_features)
                labels.append(os.path.basename(subdir))

mfcc_array = np.array(mfcc_list)
print(f"MFCC array shape: {mfcc_array.shape}")
```

Penjelasan:

- **Fungsi `os.walk`:** Menelusuri folder dataset untuk mencari file `.wav`.
- **Penyimpanan Data:**
 - `mfcc_list` menyimpan matriks MFCC.
 - `labels` menyimpan label yang diperoleh dari nama folder.
- **Output:** Informasi bentuk array MFCC (misalnya: `(total_samples, 100, 100)`).

Output:

MFCC array shape: (457, 100, 100)

4.2 Pengkodean Label dan Downsampling Data

```
# prepare data
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(labels)

print(f"Labels shape: {y.shape}")

# Downsample to 27 samples per class
target_samples = 27
downsampled_mfcc_list = []
downsampled_labels = []

for label in np.unique(labels):
    indices = [i for i, l in enumerate(labels) if l == label]
    selected_indices = random.sample(indices, min(target_samples, len(indices)))
    downsampled_mfcc_list.extend(mfcc_array[selected_indices])
    downsampled_labels.extend([label] * len(selected_indices))

downsampled_mfcc_array = np.array(downsampled_mfcc_list)
downsampled_labels = np.array(downsampled_labels)

print(f"Downsampled MFCC array shape: {downsampled_mfcc_array.shape}")
```

Penjelasan:

- **Label Encoding:** Mengubah label berbentuk string ke nilai numerik agar dapat diproses oleh model.
- **Downsampling:** Mengambil secara acak 27 sampel per kelas untuk menyeimbangkan distribusi data.
- **Output:** Bentuk array downsampled yang ditampilkan.

Output:

Labels shape: (457,)

Downsampled MFCC array shape: (102, 100, 100)

4.3 Split Data dan Reshape

```
X_train, X_test, y_train, y_test = train_test_split(downscaled_mfcc_array,  
label_encoder.transform(downscaled_labels), test_size=0.2, random_state=42)
```

```
print(f"X_train shape: {X_train.shape}")  
print(f"X_test shape: {X_test.shape}")  
print(f"y_train shape: {y_train.shape}")  
print(f"y_test shape: {y_test.shape}")
```

```
# Ubah data dari 3 dimensi menjadi 2 dimensi
```

```
n_samples, n_mfcc, n_frames = X_train.shape  
X_train_2d = X_train.reshape(n_samples, n_mfcc * n_frames)  
X_test_2d = X_test.reshape(X_test.shape[0], n_mfcc * n_frames)  
  
print(f"X_train_2d shape: {X_train_2d.shape}")  
print(f"X_test_2d shape: {X_test_2d.shape}")
```

Penjelasan:

- **Train-Test Split:** Data dipecah menjadi data latih (80%) dan data uji (20%).
- **Reshape:** Data MFCC yang awalnya 3D diubah menjadi 2D untuk kesesuaian dengan input model scikit-learn.
- **Output:** Bentuk data latih dan uji yang dicetak.

Output:

X_train shape: (81, 100, 100)

X_test shape: (21, 100, 100)

y_train shape: (81,)

y_test shape: (21,)

X_train_2d shape: (81, 10000)

X_test_2d shape: (21, 10000)

5. Visualisasi Distribusi Kelas

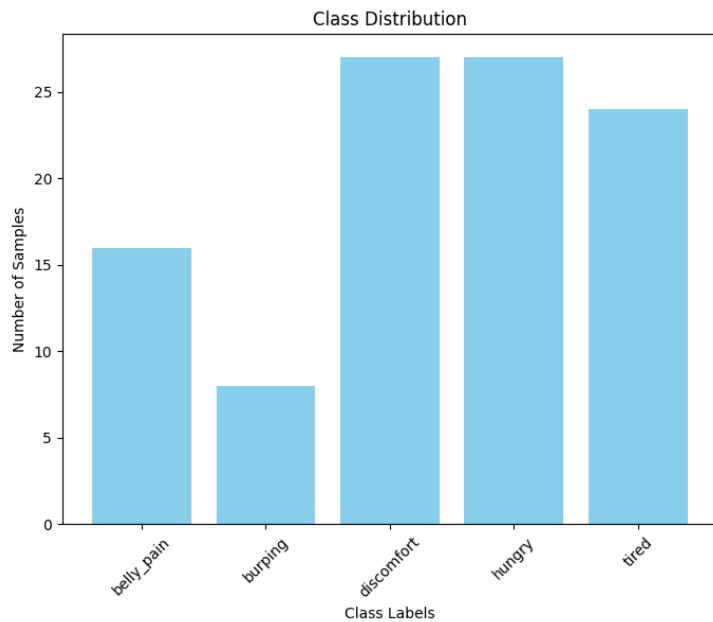
```
print(f"Total samples: {len(.downsampled_labels)}")  
unique_labels, counts = np.unique(.downsampled_labels, return_counts=True)  
print(f"Number of classes: {len(unique_labels)}")  
print("Class distribution:")  
for label, count in zip(unique_labels, counts):  
    print(f" - {label}: {count}")  
  
plt.figure(figsize=(8, 6))  
plt.bar(unique_labels, counts, color='skyblue')  
plt.title("Class Distribution")  
plt.xlabel("Class Labels")  
plt.ylabel("Number of Samples")  
plt.xticks(rotation=45)  
plt.show()
```

Penjelasan:

- **Penghitungan Distribusi:** Menampilkan jumlah sampel per kelas.
- **Grafik Batang:** Visualisasi distribusi data per kelas membantu mengidentifikasi ketidakseimbangan.

- **Output :**

```
Total samples: 102
Number of classes: 5
Class distribution:
- belly_pain: 16
- burping: 8
- discomfort: 27
- hungry: 27
- tired: 24
```



Gambar 1-Visualisasi-jumlah-data-setiap-class

6. Pelatihan dan Evaluasi Model

6.1 Pelatihan Model

```
# KNN
```

```
knn_model = KNeighborsClassifier(n_neighbors=2)
knn_model.fit(X_train_2d, y_train)
knn_metrics = evaluate_model(knn_model, X_test_2d, y_test, label_encoder.classes_)
```

```
# Naive Bayes
```

```
nb_model = GaussianNB()
nb_model.fit(X_train_2d, y_train)
nb_metrics = evaluate_model(nb_model, X_test_2d, y_test, label_encoder.classes_)
```

```
# SVM
```

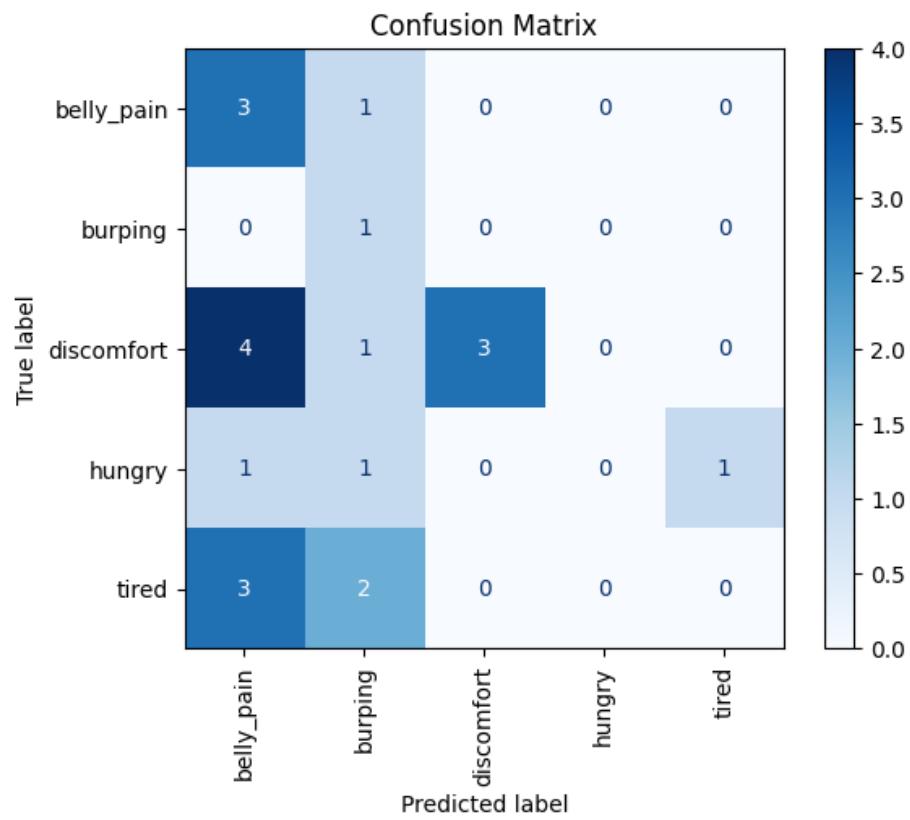
```
svm_model = SVC(kernel="linear", random_state=42)
svm_model.fit(X_train_2d, y_train)
```

```
svm_metrics = evaluate_model(svm_model, X_test_2d, y_test, label_encoder.classes_)
```

Penjelasan:

- **KNN:** Model dilatih dengan 2 tetangga terdekat.
- **Naive Bayes:** Menggunakan GaussianNB untuk klasifikasi probabilistik.
- **SVM:** Model dengan kernel linear dilatih untuk memisahkan data secara linear.
- **Output:** Metrik evaluasi masing-masing model beserta confusion matrix ditampilkan :
- **K-Nearest Neighbors (KNN)**
 - **Akurasi ±33%** dan **F1-Score ±30%**.
 - Cenderung lebih baik dibandingkan dua model lainnya pada dataset ini. Beberapa kelas seperti *burping* dan *hungry* relatif lebih mudah dikenali, sedangkan kelas seperti *discomfort* dan *tired* masih sering tertukar.
- **Naive Bayes**
 - **Akurasi ±29%** dan **F1-Score ±27%**.
 - Hampir mendekati performa KNN, tetapi masih sedikit di bawahnya. Kelas *hungry* terkласifikasi dengan baik, namun *discomfort*, *belly_pain*, dan *tired* sering salah prediksi. Asumsi independensi fitur pada Naive Bayes kemungkinan kurang sesuai untuk data audio yang memiliki korelasi antarframe.
- **SVM (Kernel Linear)**
 - **Akurasi ±19%** dan **F1-Score ±16%**.
 - Terendah di antara ketiga model. Beberapa kelas seperti *belly_pain*, *discomfort*, dan *tired* tidak terprediksi dengan benar sama sekali. Hal ini menunjukkan bahwa pemisahan linear tidak cocok atau data terlalu sedikit sehingga model sulit membedakan kelas-kelas yang mirip.

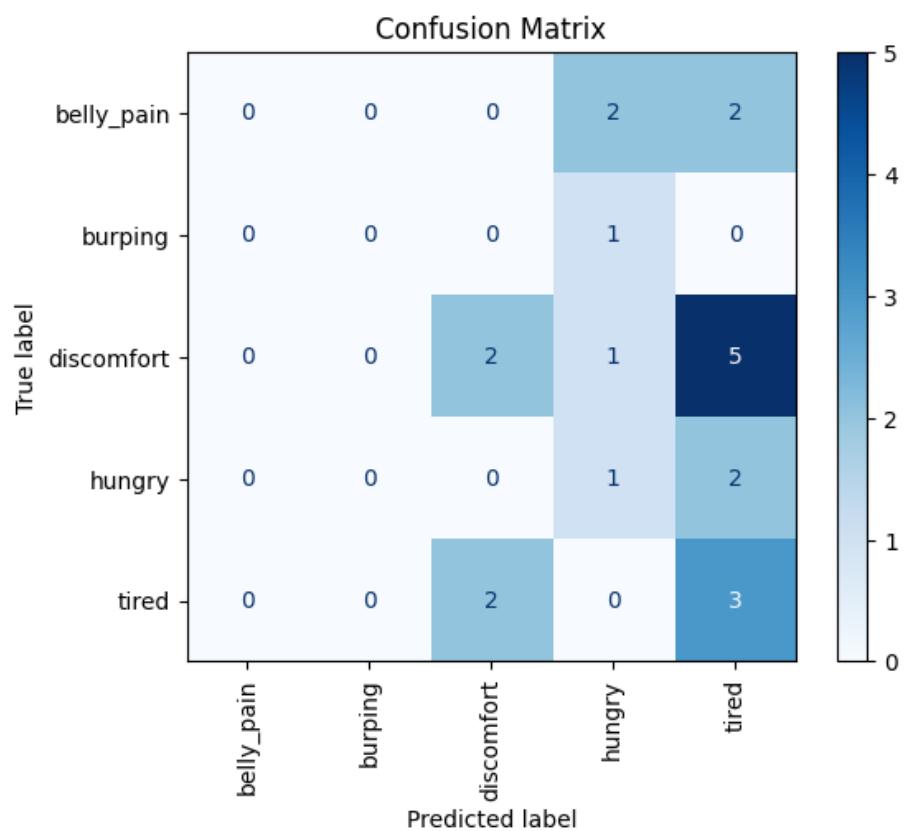
Secara keseluruhan, **KNN** lebih unggul pada dataset terbatas ini, sementara **Naive Bayes** berada di posisi kedua, dan **SVM** (dengan kernel linear) kurang optimal.



Accuracy: 0.33
Precision: 0.44
Recall: 0.33
F1-Score: 0.30

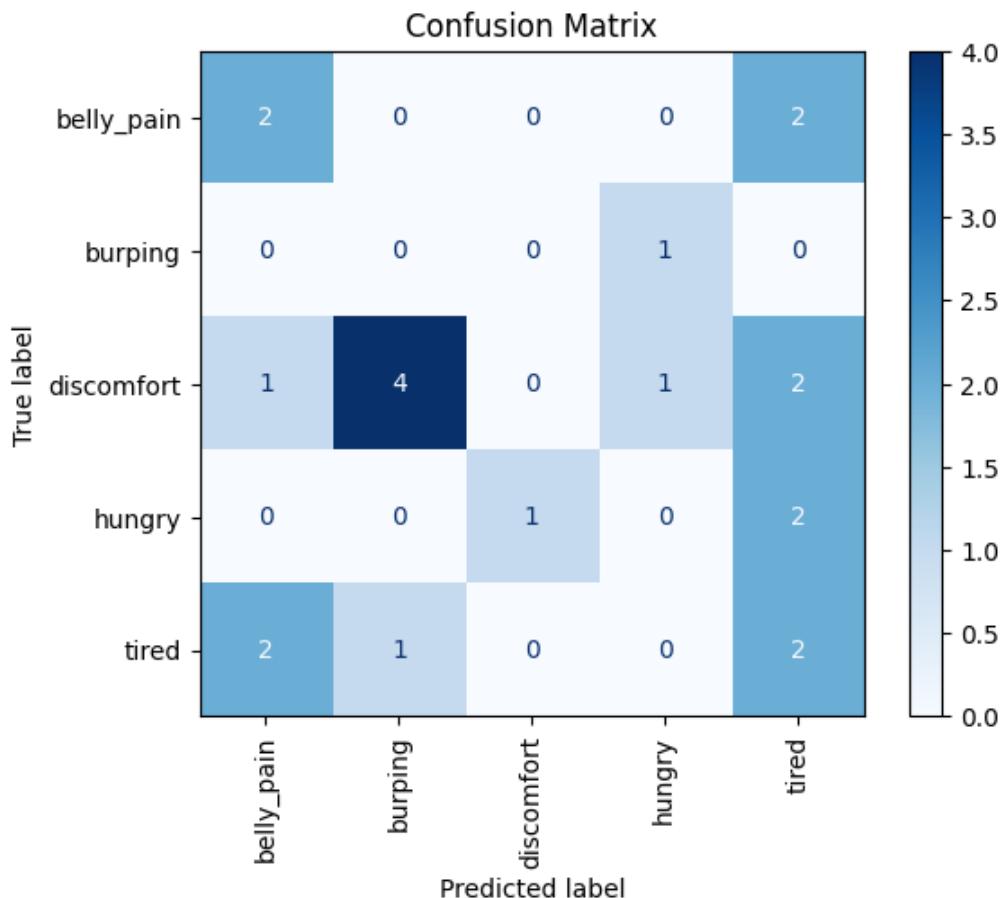
Gambar 2-Visualisasi-ConfusionMatrix-KKN

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: _warn_prf(average, modifier, f'{metric.capitalize()} is', len(result))
```



Accuracy: 0.29
Precision: 0.28
Recall: 0.29
F1-Score: 0.25

Gambar 3-Visualisasi-ConfusionMatrix-Naive



Gambar 4-Visualisasi-ConfusionMatrix-SVM

6.2 Perbandingan Hasil Evaluasi (Recall & F1-Score)

```
import matplotlib.pyplot as plt
import numpy as np

# Misal, knn_metrics, nb_metrics, dan svm_metrics sudah dihitung dengan fungsi evaluate_model
sebelumnya

# Mengumpulkan metrik dari setiap model
models = ["KNN", "Naive Bayes", "SVM"]
recall_scores = [knn_metrics["Recall"], nb_metrics["Recall"], svm_metrics["Recall"]]
f1_scores = [knn_metrics["F1-Score"], nb_metrics["F1-Score"], svm_metrics["F1-Score"]]
```

```

# Menampilkan metrik di konsol
print("Perbandingan Recall dan F1-Score untuk masing-masing model:")
for model, rec, f1 in zip(models, recall_scores, f1_scores):
    print(f"{model} => Recall: {rec:.2f}, F1-Score: {f1:.2f}")

# Visualisasi dalam bentuk grafik batang
x = np.arange(len(models)) # label posisi untuk tiap model
width = 0.35 # lebar batang

fig, ax = plt.subplots(figsize=(8,6))
rects1 = ax.bar(x - width/2, recall_scores, width, label='Recall')
rects2 = ax.bar(x + width/2, f1_scores, width, label='F1-Score')

# Tambahkan label, judul, dan keterangan sumbu
ax.set_ylabel('Skor')
ax.set_title('Perbandingan Recall dan F1-Score untuk Setiap Model')
ax.set_xticks(x)
ax.set_xticklabels(models)
ax.legend()

# Menambahkan nilai di atas tiap batang
def add_labels(rects):
    for rect in rects:
        height = rect.get_height()
        ax.annotate(f'{height:.2f}',
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3), # jarak 3 titik ke atas
                    textcoords="offset points",
                    ha='center', va='bottom')

add_labels(rects1)
add_labels(rects2)

fig.tight_layout()
plt.show()

```

Penjelasan:

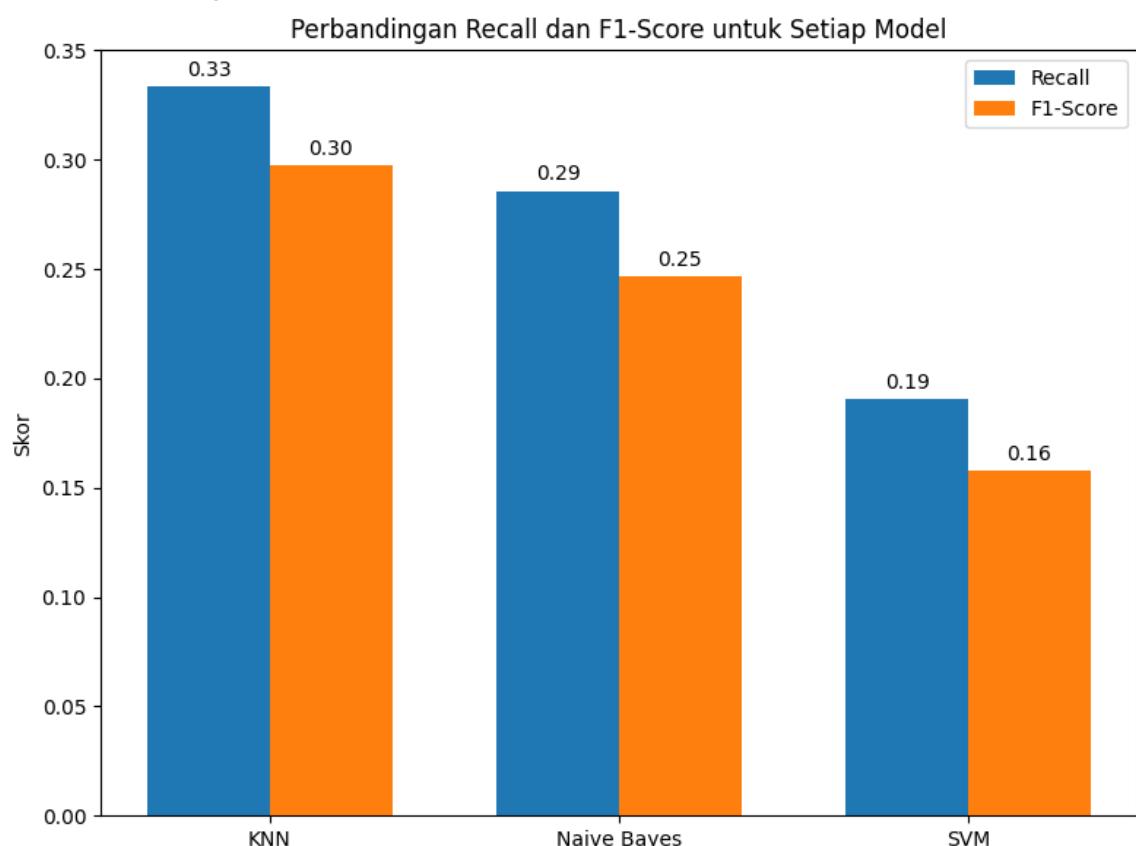
- **Pengumpulan Metrik:** Nilai recall dan F1-Score masing-masing model disimpan ke dalam list.
- **Visualisasi:** Grafik batang dibuat untuk membandingkan performa setiap model dalam hal recall dan F1-Score.
- **Output:** Grafik perbandingan yang menunjukkan nilai metrik untuk KNN, Naive Bayes, dan SVM.

Perbandingan Recall dan F1-Score untuk masing-masing model:

KNN => Recall: 0.33, F1-Score: 0.30

Naive Bayes => Recall: 0.29, F1-Score: 0.25

SVM => Recall: 0.19, F1-Score: 0.16



Gambar 5-grafik perbandingan Recall dan F1-Score.

7. Prediksi pada File Audio Acak

Fungsi `predict_audio`

```
def predict_audio(model, file_path, label_encoder, fixed_length=100):
```

```

# Ekstraksi fitur MFCC dari file audio
mfcc_features = extract_mfcc(file_path, fixed_length=fixed_length)

if mfcc_features is None:
    print(f"❌ Gagal membaca file: {file_path}")
    return

# Ubah bentuk fitur agar sesuai dengan model (reshape menjadi 2D)
mfcc_features = mfcc_features.reshape(1, -1)

# Prediksi kelas
predicted_label = model.predict(mfcc_features)[0]

# Konversi dari angka ke nama label
predicted_class = label_encoder.inverse_transform([predicted_label])[0]

print(f"✅ File: {file_path}")
print(f"⌚ Prediksi Kelas: {predicted_class}")

return predicted_class

```

Penjelasan:

- **Tujuan:** Melakukan prediksi pada file audio yang dipilih secara acak.
- **Proses:**
 - Mengekstrak fitur MFCC dari file audio.
 - Mengubah bentuk data agar sesuai dengan model.
 - Melakukan prediksi dan mengubah hasil numerik menjadi label asli.
- **Output:** Menampilkan file yang diuji dan hasil prediksi.

- Prediksi dengan KNN:
 - ✓ File: /content/drive/MyDrive/datasets/donateacry_corpus/hungry/a7b13b65-da2c-48ec-8aed-46adcaba705f-1430823489236-1.7-f-26-hu.wav
 - ⌚ Prediksi Kelas: hungry
- Prediksi dengan Naive Bayes:
 - ✓ File: /content/drive/MyDrive/datasets/donateacry_corpus/hungry/a7b13b65-da2c-48ec-8aed-46adcaba705f-1430823489236-1.7-f-26-hu.wav
 - ⌚ Prediksi Kelas: hungry
- Prediksi dengan SVM:
 - ✓ File: /content/drive/MyDrive/datasets/donateacry_corpus/hungry/a7b13b65-da2c-48ec-8aed-46adcaba705f-1430823489236-1.7-f-26-hu.wav
 - ⌚ Prediksi Kelas: hungry
 - 'hungry'

Gambar 6-Hasil-Prediksi-Kelas

8. Kesimpulan

Berdasarkan eksperimen dan evaluasi yang dilakukan, dapat disimpulkan:

- **Performa Model:**

- Model KNN memberikan hasil evaluasi terbaik (nilai akurasi, recall, dan F1-Score lebih tinggi dibandingkan Naive Bayes dan SVM) pada dataset yang relatif kecil dan downsample.
- Model Naive Bayes menunjukkan performa yang cukup baik, sedangkan SVM dengan kernel linear kurang optimal, kemungkinan karena data tidak sepenuhnya linear terpisah.
-

- **Tantangan:**

- Ukuran dataset yang kecil dan distribusi kelas yang tidak seimbang menyebabkan model cenderung bias pada kelas mayoritas.

Berdasarkan eksperimen dan evaluasi yang telah dilakukan, KNN memberikan performa terbaik (dengan akurasi, recall, dan F1-Score yang lebih tinggi) dibandingkan Naive Bayes dan SVM, kemungkinan karena data yang relatif kecil dan pola yang mudah dikenali oleh pendekatan berbasis tetangga terdekat. SVM dengan kernel linear cenderung kurang optimal karena data tidak sepenuhnya linear dan jumlah sampel terbatas. Tantangan utama yang dihadapi adalah ukuran dataset yang kecil serta distribusi kelas yang tidak seimbang, yang dapat memicu bias terhadap kelas mayoritas. Untuk meningkatkan performa, disarankan menambah jumlah data atau melakukan augmentasi, mencoba tuning hyperparameter pada masing-masing model, dan mempertimbangkan fitur tambahan selain MFCC agar representasi data lebih kaya.

9. Penutup

Dalam laporan ini, telah diuraikan secara menyeluruh tahapan mulai dari ekstraksi fitur MFCC, pengolahan dan persiapan data, pelatihan model, hingga evaluasi dan prediksi pada file audio secara acak. Melalui hasil output serta visualisasi yang disajikan, pembaca dapat memperoleh gambaran jelas mengenai kinerja masing-masing model dan tantangan yang muncul, seperti keterbatasan data dan distribusi kelas yang tidak seimbang. Pendekatan ini diharapkan dapat menjadi dasar yang kuat untuk

pengembangan lebih lanjut, dengan penyesuaian metode dan peningkatan jumlah data guna mencapai performa yang lebih optimal.