A PBL-3 Project Report on

ScreamMonitor: AI for Crime Prevention

Submitted to Manipal University Jaipur

Towards the partial fulfillment for the Award of the Degree of

BACHELORS OF TECHNOLOGY

In Computers Science and Engineering (AIML)

2024-2025

By

Jaspreet Kaur-229302307

Khushi Gupta-229310145

Shreya-229310154

MANIPAL UNIVERSITY
JAIPUR

Mr. Harish Sharma

Department of Artificial Intelligence and Machine Learning

School of Computer Science and Engineering

Manipal University Jaipur

Jaipur, Rajasthan

A PBL-3 Project Report on

ScreamMonitor: AI for Crime Prevention

Submitted to Manipal University Jaipur

Towards the partial fulfillment for the Award of the Degree of

BACHELORS OF TECHNOLOGY

In Computers Science and Engineering (AIML)

2024-2025

By

Jaspreet Kaur-229302307

Khushi Gupta-229310145

Shreya-229310154

MANIPAL UNIVERSITY
JAIPUR

Mr. Harish Sharma

Harish Sharma
MUJ-0736, Dept of AIML, MUJ

Department of Artificial Intelligence and Machine Learning

School of Computer Science and Engineering

Manipal University Jaipur

Jaipur, Rajasthan

MANIPAL UNIVERSITY
JAIPUR
(University under Section 2(f) of the UGC Act)

CERTIFICATE

This is to certify that the project report (PBL-3) entitled **Scream Monitor: AI for Crime Prevention** submitted by **Jaspreet Kaur(229302307)**,Department of Artificial Intelligence and Machine Learning (AIML), School of Computer Science and Engineering Manipal University Jaipur, Rajasthan for the award of the degree of *Bachelor of Technology* is a record of bonafide work carried out by him/her under my supervision, as per the code of academic and research ethics of Manipal University Jaipur, Rajasthan.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The report fulfills the requirements and regulations of the University and in my opinion, meets the necessary standards for submission.

Harish Sharma
MUJ-0736 Dept of AIML, MUJ

Place: Manipal University Jaipur

Signature

Date: 27/11/2024

Mr. Harish Sharma

MANIPAL UNIVERSITY JAIPUR
(University under Section 2(f) of the UGC Act)

CERTIFICATE

This is to certify that the project report (PBL-3) entitled **Scream Monitor: AI for Crime Prevention** submitted by **Khushi Gupta(22931O145)**,Department of Artificial Intelligence and Machine Learning (AIML), School of Computer Science and Engineering Manipal University Jaipur, Rajasthan for the award of the degree of *Bachelor of Technology* is a record of bonafide work carried out by him/her under my supervision, as per the code of academic and research ethics of Manipal University Jaipur, Rajasthan.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The report fulfills the requirements and regulations of the University and in my opinion, meets the necessary standards for submission.

Harish Sharma
MUJ-0736, Dept of AIML, MUJ

Place: Manipal University Jaipur — Signature

Date: 27/11/2024 — **Mr. Harish Sharma**

MANIPAL UNIVERSITY
JAIPUR
(University under Section 2(f) of the UGC Act)

CERTIFICATE

This is to certify that the project report (PBL-3) entitled **Scream Monitor: AI for Crime Prevention** submitted by **Shreya(229310154)**,Department of Artificial Intelligence and Machine Learning (AIML), School of Computer Science and Engineering Manipal University Jaipur, Rajasthan for the award of the degree of *Bachelor of Technology* is a record of bonafide work carried out by him/her under my supervision, as per the code of academic and research ethics of Manipal University Jaipur, Rajasthan.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The report fulfills the requirements and regulations of the University and in my opinion, meets the necessary standards for submission.

Harish Sharma
MUJ-0736, Dept of AIML, MUJ

Place: Manipal University Jaipur

Signature

Date: 27/11/2024

Mr. Harish Sharma

Abstract

Human screams are crucial auditory indicators of distress, danger, or violence in emergency situations. However, traditional security systems are predominantly focused on visual monitoring and lack the ability to detect auditory cues, leaving a significant gap in real-time threat detection. This project, Human Scream Detection and Analysis for Controlling Crime Rate, aims to bridge this gap by developing an advanced scream detection system using machine learning algorithms. The system is designed to automatically identify and differentiate screams from environmental sounds, triggering alerts for potential emergencies. The methodology involves creating a user-friendly interface using the Kivy framework, curating a dataset of scream and non-scream audio samples, and extracting Mel Frequency Cepstral Coefficients (MFCCs) as features using the librosa library. Two machine learning models—Support Vector Machine (SVM) and Multilayer Perceptron Neural Network (MPN)—are trained to classify audio signals accurately. The system employs a dual-risk alert mechanism, generating high-risk alerts when both models detect a scream and medium-risk alerts when only one model identifies a scream. This application-based project provides a scalable and robust solution for integrating real-time sound recognition into existing safety infrastructures, enhancing response times and improving public safety. Its potential applications span high-crime areas, schools, residential complexes, and public spaces, offering a transformative approach to emergency detection and crime prevention.

Table of Contents

1. Introduction

In emergency situations, human screams are vital auditory signals that often indicate distress, violence, or imminent danger. The recognition of these sounds in real time can significantly improve safety measures by enabling prompt responses from law enforcement or emergency services. However, traditional security systems, such as surveillance cameras and alarm systems, are predominantly designed for visual monitoring and fail to account for critical auditory cues like human screams. This limitation presents a gap in real-time threat detection, particularly in scenarios where visual input is restricted or absent. Addressing this gap is crucial to enhancing safety in diverse environments such as public spaces, educational institutions, and residential areas.

The proposed project, Human Scream Detection and Analysis for Controlling Crime Rate, is an application-based initiative aimed at developing an advanced sound recognition system that automatically identifies human screams in audio recordings. By employing machine learning algorithms, the system seeks to distinguish screams from other environmental noises, enabling swift and accurate detection of emergencies. Such a solution not only strengthens the capabilities of existing safety systems but also provides a scalable framework for integrating auditory analysis into broader crime prevention strategies.

The project methodology encompasses a comprehensive approach to scream detection, starting with the design of an intuitive user interface (UI) using the Kivy framework. This UI will facilitate real-time audio input and interaction with the system. A key aspect of the project involves curating a diverse dataset of scream and non-scream audio samples sourced from online repositories. This dataset will serve as the foundation for training machine learning models. Sound features are extracted from the audio samples using the librosa library, with a focus on Mel Frequency Cepstral Coefficients (MFCCs), which are widely recognized for their effectiveness in audio analysis.

Two machine learning models—Support Vector Machine (SVM) and Multilayer Perceptron Neural Network (MPN)—will be developed and trained on the extracted features. These models will operate collaboratively to ensure accurate classification of audio signals. The detection process includes a dual-risk alert mechanism: a high-risk alert is triggered when both models confirm a scream, while a medium-risk alert is activated if only one model detects a scream. This layered risk assessment ensures reliability and minimizes false alarms.

The proposed system offers applications in various contexts, including high-crime urban areas, schools, hospitals, and residential complexes. It is designed to seamlessly integrate with existing safety infrastructure, such as CCTV systems, to provide a holistic emergency response framework. By enabling real-time detection and classification of screams, the system has the potential to improve response times, reduce human intervention, and enhance individual security.

This application-driven project holds promise for significant societal impact, particularly in enhancing public safety and preventing crimes. Beyond its immediate application, the framework can be adapted to broader use cases, including the detection of other distress sounds like gunshots or explosions. As urban areas continue to expand, the implementation of sound-based emergency detection systems will play a critical role in fostering safer communities.

By combining sound recognition technologies with state-of-the-art machine learning models, this project represents a step forward in the development of intelligent safety systems. Its scalability and robustness make it a viable solution for addressing vulnerabilities in existing security mechanisms, ultimately contributing to a safer and more secure society.

3. Objective

In today's world, ensuring public safety requires innovative solutions that leverage technology for rapid threat detection and response. Traditional surveillance systems rely heavily on visual cues, often overlooking auditory indicators like screams, which are critical in signaling emergencies. This project addresses this gap by creating a system that can detect and analyze human screams in real time, offering timely alerts to emergency responders or law enforcement personnel. By integrating machine learning and advanced sound recognition techniques, the project aims to redefine security frameworks, making them more responsive and effective in preventing crimes and safeguarding individuals.

The primary objectives of this project are:

1. **Real-time Scream Detection and Emergency Alerts**
   The system aims to identify human screams instantly, triggering immediate notifications to law enforcement or emergency responders. This capability is vital for preventing violent incidents, aiding victims in distress, or mitigating risks in environments such as public spaces, educational institutions, and high-crime neighborhoods.

2. **Minimizing False Alarms Through Advanced Differentiation**
   Leveraging machine learning algorithms, the system will classify screams based on their context—such as distress versus excitement—thereby reducing false alarms. This ensures that alerts are generated only for genuine threats, enhancing the reliability and credibility of the system.

3. **Integration with Broader Safety Frameworks**
   The project envisions incorporating the scream detection system into larger safety infrastructures, including smart surveillance systems, mobile apps, and emergency response networks. Features like location-based monitoring and real-time data analysis will further enhance its utility by enabling targeted and localized safety measures.

4. **Scalable and Global Safety Solution**
   Designed to be scalable and versatile, the system can adapt to a wide range of environments, from urban centers and parks to transportation hubs and residential areas. Over time, it can evolve into a comprehensive crime prevention tool, offering customized safety solutions tailored to the unique needs of different regions and communities worldwide.

By meeting these objectives, the project seeks to contribute significantly to global safety efforts, creating a proactive tool that not only identifies threats but also facilitates swift interventions, ultimately reducing crime rates and enhancing public security.

2. Literature Survey

Comparative analysis of existing software's

Software/Tool	Purpose	Strengths	Limitations
YAMNet (TensorFlow)	Pre-trained model for general sound classification (includes scream detection).	- Detects over 521 classes of sounds, including screams. - Easy integration into TensorFlow workflows.	- Limited to pre-trained classes. - Requires fine-tuning for specific use cases like crime prevention.
Librosa	Python library for audio and music analysis.	- Ideal for feature extraction (MFCC, spectral features). - Highly customizable for specific applications.	- Requires integration with machine learning tools for classification. - No GUI.
Kivy	Framework for creating cross-platform GUI applications.	- Ideal for developing user-friendly interfaces for scream detection applications. - Open-source.	- GUI-only framework, requires integration with backend detection models.
SoX (Sound eXchange)	Command-line tool for sound processing.	- Great for batch audio processing (e.g., format conversion, noise filtering). - Lightweight.	- No real-time capabilities. - Limited to preprocessing tasks without classification models.
WaveSurfer	Open-source tool for audio visualization and annotation.	- Easy to use for segmenting and annotating audio files. - Supports various audio formats.	- Not real-time. - Requires manual processing and lacks classification features.
Praat	Software for speech analysis and phonetics research.	- High-level tools for analyzing voice intensity and pitch. - Widely used in linguistics.	- Not real-time. - No integrated machine learning capabilities for scream classification.
Edge Impulse	Platform for developing AI models for edge devices (audio included).	- Allows creating lightweight models for real-time deployment. - Built-in tools for data collection and feature extraction.	- Requires edge devices for deployment. - Limited flexibility compared to traditional TensorFlow/Keras pipelines.
TARS (Custom ML Framework)	Customizable ML platform for sound detection (e.g., gunshots, screams).	- Can be tailored for specific sound detection tasks. - Scalable for crime prevention applications.	- Requires significant development effort. - Needs training on large datasets.

Technologies used

Library/Tool	Purpose	Key Use Cases
SoundDevice	Real-time audio input/output for recording and playback.	Capturing audio in real-time for scream detection; playback for testing results.
SciPy (scipy.io)	Library for scientific computations, including audio file I/O.	Reading and writing WAV files for audio data processing and feature extraction.
Pandas	Data manipulation and analysis.	Storing, cleaning, and analyzing tabular data, such as audio metadata, results, and logs.
NumPy	Fundamental package for numerical computation in Python.	Array manipulations, signal processing, and computations during audio data preprocessing.
PyTZ	Time zone handling for datetime objects.	Adding precise timestamps to audio recordings across different time zones.
Requests	Library for making HTTP requests.	Fetching external data (e.g., weather, crime statistics) or sending results to a web server or API.
Kivy	Framework for building multitouch GUI applications.	Developing cross-platform GUIs to display scream detection results and provide user controls.
Kivy CheckBox	Widget for creating interactive checkboxes in Kivy.	Adding user-configurable options in the app, such as enabling/disabling features.
OpenPyXL	Library for working with Excel spreadsheets.	Exporting analysis results or logs to Excel for structured reporting and easy sharing.
SoundFile	Library for handling various audio file formats.	Reading and writing FLAC, OGG, and WAV files used in audio dataset creation or storage.
Wave	Simplified library for working with WAV files.	Reading simple WAV audio files and converting audio for further processing.
Shutil	High-level file operations library.	Organizing, copying, or archiving datasets or log files.
TensorFlow	Open-source library for machine learning and deep learning.	Building and deploying machine learning models for scream detection using neural networks.

Keras	High-level API built on TensorFlow for deep learning.	Creating and training deep learning models quickly and efficiently.
Sequential (Keras)	Layered approach for defining neural networks.	Building a feedforward neural network for audio classification and scream detection.
Dense (Keras)	Fully connected layers in neural networks.	Designing and implementing layers in deep learning models for classification tasks.
PyDub	Audio editing and manipulation library built on FFmpeg.	Audio conversion, segmentation, and feature extraction (e.g., slicing audio into smaller chunks).
Matplotlib	Python library for plotting and data visualization.	Visualizing waveforms, spectrograms, and performance metrics (e.g., training accuracy).
Seaborn	Statistical data visualization library.	Creating advanced plots like heatmaps for confusion matrices and spectrogram intensity charts.
OS Module	Provides OS interaction capabilities.	Handling file paths, directories, and automating file-related tasks during data preprocessing.

4. Planning of Work

The **Human Scream Detection and Analysis for Controlling Crime Rate** project focuses on using sound-based detection systems to identify screams, potentially indicating distress or a crime, and automatically notifying authorities or alert systems. Below is the detailed methodology of this project, based on the general principles of sound detection, feature extraction, machine learning, and deployment.

4.1 Requirement Engineering (SRS) / Research Methodology

4.1.1 Objective

The primary goal is to develop a machine learning-based system to:

1. Detect human screams from audio files or live recordings.
2. Classify the severity of detected screams (e.g., high, medium, or safe).
3. Trigger appropriate alerts based on the classification.

4.1.2 Functional Requirements

1. **Audio Input**:
    - The system should handle both pre-recorded .wav files and live microphone input.
    - Ensure compatibility with mono-channel .wav files of sufficient duration (minimum 3 seconds).
2. **Audio Preprocessing**:
    - Validate audio file properties (e.g., channel count, duration).
    - Extract features (e.g., amplitude values, metadata) for model training and prediction.
3. **Model Training**:
    - Train a neural network model for binary classification (scream vs. non-scream).
    - Evaluate the model's performance using accuracy and loss metrics.
4. **Real-time Detection**:
    - Process live or uploaded audio files.
    - Predict and classify audio data in real-time.

5. **Alert System**:
    - Trigger visual alerts (pop-ups) based on the severity of predictions.

4.1.3 Non-Functional Requirements

1. **Performance**:
    - High classification accuracy to reduce false positives and negatives.
    - Real-time detection capability for live inputs.
2. **Scalability**:
    - Handle multiple audio streams simultaneously in the future.
3. **User Experience**:
    - Provide an intuitive, Kivy-based user interface for easy interaction.

4.1.4 Research Methodology

1. **Data Collection**:
    - Collect scream and non-scream audio samples from public datasets or manual recordings.
    - Ensure data diversity (e.g., different tones, pitches, and background noises).
2. **Feature Extraction**:
    - Use raw amplitude values as features for model training.
    - Handle data imbalance by augmenting under-represented classes.
3. **Machine Learning Approach**:
    - Employ a Multi-Layer Perceptron (MLP) for binary classification.
    - Use binary_crossentropy loss for optimization and sigmoid activation for output.
4. **Evaluation**:
    - Use metrics like accuracy, precision, recall, and F1-score to evaluate the model's performance.
    - Conduct cross-validation to ensure robustness.

4.2 Design

4.2.1 System Architecture

The system comprises four main modules:

1. **Data Preparation Module**:
    - Prepares a labelled dataset from raw audio files.
    - Extracts and validates audio metadata (e.g., duration, channels).
2. **Model Training Module**:
    - Trains the neural network model using pre-processed data.
    - Saves the trained model for deployment.
3. **Real-time Processing Module**:
    - Processes live audio recordings or uploaded files for classification.
    - Interfaces with the trained model to predict scream occurrences.
4. **Alert System**:
    - Generates alerts based on the classification output.

4.2.2 Data Flow

1. **Input**:
    - Raw audio files or live microphone recordings.
2. **Processing**:
    - Validate and preprocess audio data (e.g., padding, normalization).
    - Extract features and make predictions using the trained model.
3. **Output**:
    - Binary classification result (1 for scream, 0 for non-scream).
    - Visual alerts based on the result.

4.2.3 Key Interactions

- **User Interaction**: Users can upload files or record audio using the Kivy UI.
- **Model Integration**: The trained model processes input data and returns predictions.
- **Alert Triggering**: Visual alerts are displayed based on prediction results.

4.3 Coding / Implementation

4.3.1 Data Preparation

1. **Script**: datasetmaker.py
   - Ensures folder structure: positive, negative, testing.
   - Assigns labels:
     - 1 for screams in the positive folder.
     - 0 for non-screams in the negative folder.
   - Validates files:
     - Removes stereo files.
     - Moves files shorter than 3 seconds to a dustbin folder.
   - Creates a labelled dataset (resources.csv) and shuffles it for training and testing(datasetmaker).

```
files = os.listdir('testing')
for i in files:
    if i.startswith("1"):
        num = float(1)
    else:
        num = float(0)

    i = 'testing/' + i
    try:
        data, rs = read(i)
    except:
        print("Removed " + i)
        os.remove(i)
        continue
    try:
        rs = rs.astype(float)
        rs = np.insert(rs, 0, num)
        a = pd.Series(rs)
        arr.append(a)
        self.ctr += 1
    except:
        pass
```

Fig. datasetmaker.py

2. **Metadata Extraction**:
   - Script: fileinfo.py
   - Extracts metadata (e.g., filename, channels, sample rate) from audio files.
   - Saves metadata in an Excel file (infofile.xlsx) for manual inspection(fileinfo).

4.3.2 Model Training

1. **Script**: sound_classifier_neural.py
   - Reads the prepared dataset (newresources.csv) and splits it into features (X) and labels (y).
   - Builds a neural network:
     - Input layer: Matches the dataset's feature dimensions.
     - Hidden layers: Fully connected layers with ReLU activation.

    - Output layer: Single neuron with sigmoid activation for binary classification.
- Compiles the model:
    - Loss: binary_crossentropy.
    - Optimizer: adam.
- Trains the model for 150 epochs with a batch size of 50(sound_classifier_nueral).

```
# # define the keras model
model = Sequential()
model.add(Dense(12, input_dim=total_number_of_column_required_for_prediction, activation='relu')
model.add(Dense(8, activation='relu'))

model.add(Dense(10, activation='relu'))
model.add(Dense(5, activation='relu'))
model.add(Dense(3, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# compile the keras model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# fit the keras model on the dataset
history = model.fit(X, y,validation_split=0.33, epochs=150, batch_size=50
|   |   |   |   |   )


# evaluate the keras model
_, accuracy = model.evaluate(X, y)
print('Accuracy: %.2f' % (accuracy * 100))

# make probability predictions with the model
predictions = model.predict(X2)
```

Fig. sound_classifier_neural.py

```
Epoch 148/150
2/2 [==============================] - 0s 68ms/step - loss: 0.1228 - accuracy: 0.9684 - val_loss: 263.3119 - val_accuracy: 0.5417
Epoch 149/150
2/2 [==============================] - 0s 70ms/step - loss: 0.1226 - accuracy: 0.9684 - val_loss: 263.3117 - val_accuracy: 2/2 [==
1226 - accuracy: 0.9684 - val_loss: 263.3117 - val_accuracy: 0.5417
Epoch 150/150
2/2 [==============================] - 0s 74ms/step - loss: 0.1225 - accuracy: 0.9684 - val_loss: 263.3116 - val_accuracy: 0.5417
5/5 [==============================] - 0s 8ms/step - loss: 88.4657 - accuracy: 0.8252
Accuracy: 82.52
2/2 [==============================] - 0s 74ms/step - loss: 0.1225 - accuracy: 0.9684 - val_loss: 263.3116 - val_accuracy: 0.5417
5/5 [==============================] - 0s 8ms/step - loss: 88.4657 - accuracy: 0.8252
Accuracy: 82.52
0.5417
5/5 [==============================] - 0s 8ms/step - loss: 88.4657 - accuracy: 0.8252
Accuracy: 82.52
Accuracy: 82.52
1/1 [==============================] - 0s 194ms/step
predicted value is[1, 1, 1, 0, 1]
predicted value is[1, 1, 1, 0, 1]
actual value was[1.0, 1.0, 0.0, 0.0, 1.0]
```

Fig. Output of sound_classifier_neural

4.3.3 Real-time Processing

1. **Script**: modelloader.py
   - Processes .wav audio files:
     - Pads or truncates files to match the model's input dimensions.
     - Handles missing values by replacing them with zeros.
   - Loads the trained model and predicts the classification for the input file.
   - Returns a binary output (1 for scream, 0 for non-scream)(modelloader).

```
# Validate input shape
if X2.shape[1] != model.input_shape[1]:
    print(f"Invalid input shape for prediction. Expected {model.input_shape[1]} columns, but got {X2.shape[1]}")
    return False

# Make predictions
try:
    predictions = model.predict(X2)
    rounded = [round(x[0]) for x in predictions]
    print(f"Predictions: {predictions}")
except Exception as e:
    print(f"Error during prediction: {e}")
    return False

return rounded == [1.0]
```

Fig. modelloader.py

2. **Integration in the App**:
   - Script: main.py
   - Provides a Kivy-based user interface:
     - Allows users to record audio or upload files for classification.
     - Displays visual alerts (e.g., high risk, medium risk) based on predictions.
   - Interacts with multiple models (e.g., MLP, SVM) to ensure robust classification(main).

4.3.4 Alert System

1. **Risk Levels**:
   - **High Risk**: If both models (MLP and SVM) predict a scream.
   - **Medium Risk**: If either model predicts a scream.

- **Safe**: If neither model predicts a scream.

2. **Implementation**:
    - Alerts are displayed using Kivy pop-ups with appropriate risk levels.

4.3.5 Workflow Summary

1. **Audio Data Collection**:
    - Collect and preprocess data using datasetmaker.py and fileinfo.py.
2. **Model Training**:
    - Train the neural network using sound_classifier_neural.py.
3. **Real-time Processing**:
    - Process live or uploaded audio files using modelloader.py.
4. **User Interaction**:
    - Enable seamless interaction via the Kivy app (main.py).
5. **Alert Generation**:

- Trigger risk-based alerts depending on classification results.

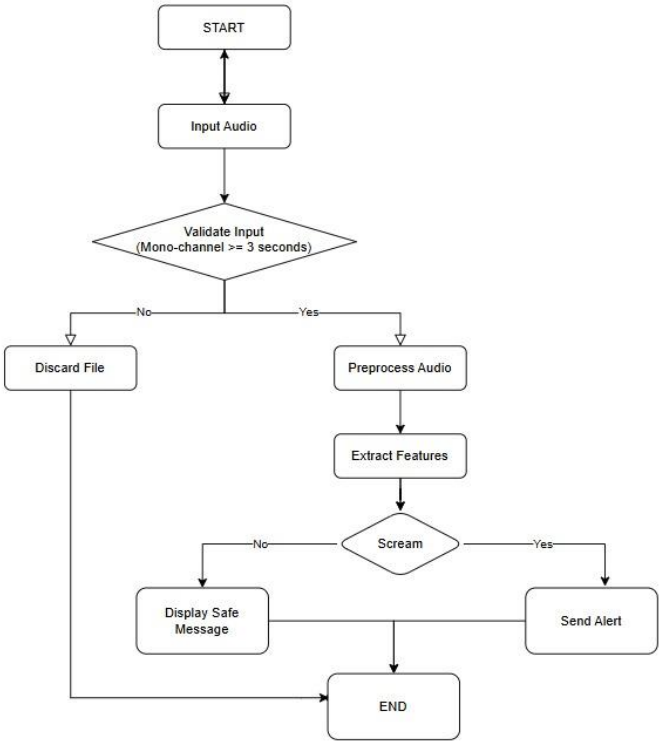

Fig. Flowchart

Conclusion

This methodology outlines a detailed and systematic approach to building a robust scream detection and analysis system. By integrating data preprocessing, model training, real-time processing, and user interaction, the project achieves its objectives effectively.