DESIGN AND DEVELOPMENT OF DEEP AUDIO CLASSIFIER USING NLP

A

MAJOR PROJECT-I REPORT

Submitted in partial fulfillment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

in

CSE-ARTIFICIAL INTELLIGENCE & DATA SCIENCE

By

GROUP NO. 16

Prachi Rajput (0187AD211031)

Ankita Rajput (0187AD211006)

Abhiraj Chouhan(0187AD211002)

Under the guidance of

Prof. Dheeraj Namdev

(Assistant Professor)

Department of CSE-Artificial Intelligence & Data Science

Sagar Institute of Science &Technology (SISTec), Bhopal (M.P.)

Approved by AICTE, New Delhi & Govt. of M.P.

Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

DECEMBER-2024

Sagar Institute of Science & Technology (SISTec), Bhopal

Department of CSE-Artificial Intelligence & Data Science

CERTIFICATE

I hereby certify that the work which is being presented in the B.Tech. Major Project-I Report entitled **Design and development of Deep Audio Classifier,** in partial fulfillment of the requirements for the award of the degree of ***Bachelor of Technology***, submitted to the Department of *CSE* with *Artificial Intelligence & Data Science*, *Sagar Institute of Science & Technology (SISTec)***,** Bhopal (M.P.) is an authentic record of my own work carried out during the period from July-2024 to Dec-2024 under the supervision of **Prof. Dheeraj Namdev (Assistant professor).**

The content presented in this project has not been submitted by me for the award of any other degree elsewhere.

Ankita Rajput
0187AD211006

Prachi Rajput
0187AD211031

Abhiraj Chouhan
0187AD211002

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Prof. Dheeraj Namdev
Project Guide

Dr. Vasima Khan
HOD, CSE-AI&DS

Dr. D.K. Rajoriya
Principal, SISTec

ACKNOWLEGMENT

We would like to express our sincere thanks to **Dr. D.K. Rajoriya, Principal, SISTec** and **Dr. Swati Saxena, Vice Principal, SISTec** Gandhi Nagar, Bhopal for giving us an opportunity to undertake this project.

We also take this opportunity to express a deep sense of gratitude to **Dr. Vasima Khan, HOD**, **Department of CSE-Artificial Intelligence & Data Science** for her kindhearted support.

We extend our sincere and heartfelt thanks to our Guide, **Prof. Dheeraj Namdev** for providing us with the right guidance and advice at the crucial junctures and for showing us the right way.

I am thankful to the **Project Coordinator, Ms. Ruchi Jain** who devoted her precious time in giving us the information about the various aspect and gave support and guidance at every point of time. I am thankful to their kind and supportive nature. His inspiring nature has always made my work easy.

I would like to thank all those people who helped me directly or indirectly to complete my project whenever I found myself in any issue.

TABLE OF CONTENTS

ABSTRACT

The classification of audio signals has become a critical area in machine learning, with applications spanning across voice recognition, environmental sound classification, and emotion detection. Traditionally, audio classification relies heavily on signal processing and feature extraction methods, yet these approaches often fall short in capturing the contextual and semantic relationships within complex audio data. This paper presents a novel approach to audio classification by integrating natural language processing (NLP) methodologies within a deep learning framework, thereby addressing the limitations of conventional techniques.

Inspired by NLP approaches, we apply embedding layers and attention mechanisms, transforming audio features into high-dimensional vector representations similar to word embeddings. This transformation enables the model to identify patterns and semantic relationships in audio data that are typically captured in language tasks, enhancing its performance on unstructured and noisy audio inputs.

Moreover, an attention mechanism refines the classifier's focus on critical audio segments, mimicking NLP's contextual understanding capabilities. The classifier is trained and evaluated on diverse, publicly available datasets, demonstrating superior accuracy, robustness, and generalization across a range of audio classification tasks compared to traditional methods. This approach marks a substantial advancement in the integration of NLP strategies within the audio domain, opening pathways for applications in real-time audio sentiment analysis, intelligent voice assistants, and surveillance systems.

By applying NLP techniques like word embeddings to transform audio features into a structured form, the model effectively captures the nuances within complex audio signals, allowing for benchmark dataset, demonstrating significant improvements in accuracy, robustness, and scalability compared to traditional audio classification methods.

These results underscore the potential of NLP methodologies in deep audio classification tasks, marking a step forward in applications such as audio-based sentiment analysis, voice recognition, and ambient sound detection.

LIST OF ABBREVIATIONS

ACRONYM	FULL FORM
SDLC	Software Development Life Cycle
SQL	Structured Query Language
HTML	Hyper Text Markup Language
UML	Unified Modeling Language
NLP	Natural Language Processing

LIST OF FIGURES

Chapter 1
Introduction

CHAPTER-1
INTRODUCTION

ABOUT THE PROJECT

The "Deep Audio Classifier using NLP" project focuses on categorizing audio clips by combining audio and NLP techniques. Audio features (like spectrograms) are extracted and transcribed to text using speech recognition, allowing NLP models to analyze linguistic content, sentiment, or tone. The model combines CNNs or RNNs for audio with NLP layers like LSTM or Transformers, achieving accurate classification. This project can be applied to speech emotion recognition, sentiment analysis, or language identification, with deployment as an interactive web app.

1.1 PROJECT OBJECTIVES

The primary objectives of this project are as follows:

1.1.1 **To Build a Multi-Modal Classification Model**: Develop a robust classification model that combines audio and NLP features to categorize audio clips accurately based on acoustic and linguistic attributes.

1.1.2 **Audio Data Collection and Preprocessing**: Gather and preprocess a comprehensive dataset of audio clips, including steps like noise reduction, segmentation, and transcription for effective feature extraction.

1.1.3 **NLP Feature Extraction and Analysis**: Use NLP techniques on transcriptions to analyze linguistic features like sentiment, tone, and specific content, enhancing classification capabilities.

1.1.4 **Model Evaluation and Validation**: Utilize various machine learning and deep learning algorithms and metrics to evaluate and validate model performance for improved classification accuracy.

1.1.5 **User-Friendly Deployment**: Create an interactive web-based application where users can upload audio files and receive immediate classification results for practical applications.

1.1.6 **Skill Enhancement and Practical Experience**: Enable the project team to deepen expertise in data preprocessing, NLP, multi-modal model development, and deep learning evaluation, advancing skills in audio and NLP integration.

1.2 SIGNIFICANCE OF DEEP AUDIO CLASSIFIER

The **Deep Audio Classifier using NLP** project is significant for its ability to merge audio and linguistic features, achieving more nuanced and precise classification than single-mode models, particularly in tasks involving sentiment, emotion, or language detection. This multi-modal approach enhances real-world applications, including emotion recognition in mental health support, customer sentiment analysis for service improvement, and language identification in communications.

Furthermore, the project advances the field of multi-modal learning by integrating both audio and NLP, which is valuable for AI innovation. This integration not only contributes to the research community but also provides practical solutions across various industries.

Additionally, the project offers developers essential hands-on experience with in-demand skills in machine learning, NLP, and audio processing. This practical exposure supports their growth in the rapidly evolving fields of AI and data science, equipping them with the expertise needed to tackle complex challenges in future projects.

Chapter 2
Software & Hardware Requirements

CHAPTER- 2

SOFTWARE AND HARDWARE REQUIEMENTS

2.1 SOFTWARE REQUIREMENTS

The successful development and deployment of the "Design and Development of Deep Audio Classifier using NLP" project necessitate the utilization of specific software tools and technologies. This section outlines the software requirements vital to the project's execution:

2.1.1 PROGRAMMING LANGUAGE

Python serves as the primary programming language for this project. The following Python libraries and frameworks are utilized:

1. NumPy: For numerical computations and data manipulation.
2. Pandas: For data handling and preprocessing.
3. Scikit-Learn: To implement machine learning algorithms and model evaluation.
4. Matplotlib and Seaborn: For data visualization and model performance analysis.
5. Jupyter Notebook: Used for data exploration, model development, and testing.
6. Librosa: Specifically added for extracting audio features (e.g., MFCCs, spectrograms).
7. NLTK or SpaCy: Used for tokenization, lemmatization, and sentiment analysis on transcribed text for enhanced NLP feature extraction.

2.1.2 FRAMEWORK

HTML, CSS, and JavaScript: These web technologies are used to create the user interface for interacting with the mobile price range prediction model.

Flask: A Python web framework, for serving web pages and handling user inputs.

2.2 HARDWARE REQUIREMENTS

The "Design and Development of Deep Audio Classifier" project demands specific hardware resources to ensure its smooth operation. The hardware requirements include:

1. Storage (SSD): Solid State Drive (SSD) storage of 512GB or more is preferable for faster data access, loading, and saving of audio files, models, and large NLP libraries, reducing training and testing times.
2. GPU (Graphics Processing Unit): A powerful GPU, such as NVIDIA RTX series or higher, accelerates the processing of deep learning models, especially for tasks involving large audio datasets and NLP layers.
3. TPU (Tensor Processing Unit, optional): If available, TPUs can further accelerate training for deep learning models, especially with Google Cloud or Google Colab, which offers TPU support.

2.3 ADAPTABILITY

The software components of the project are compatible with multiple operating systems, allowing flexibility for both developers and users. The web-based user interface is designed to be accessible from various web browsers, ensuring a broad reach.

2.4 Development Environment

For the development and testing of the project, a well-configured Python development environment is recommended. Tools like Anaconda, Jupyter Notebook, and code editors (e.g., Visual Studio Code, PyCharm) can enhance the development process.

Chapter 3
Problem Description

CHAPTER- 3
PROJECT DESCRIPTION

INTRODUCTION

The **Deep Audio Classifier using NLP** project aims to classify audio clips by combining audio signal processing with natural language processing (NLP) techniques. In this approach, audio data is analyzed not only for its acoustic properties but also for the linguistic content extracted from transcriptions. This dual-analysis method allows for a more comprehensive understanding of audio inputs, making it possible to recognize complex elements like emotions, sentiment, or language.

The classifier leverages both audio features (such as spectrograms and MFCCs) and NLP-based text analysis, creating a multi-modal model that enhances classification accuracy. Applications for such a system span numerous domains, from detecting customer sentiment in support calls to identifying specific languages or dialects, making it versatile and highly impactful. This project not only demonstrates the practical utility of combining audio and NLP technologies but also offers insights into advancing human-computer interaction by enabling systems to understand and respond contextually to audio content.

As audio-based interactions become more common, the capability to understand and respond to spoken content effectively is increasingly essential, making this project valuable across industries and research fields.

3.1 CHALLENGES AND CONSIDERATIONS

The **Deep Audio Classifier using NLP** project involves several challenges and important considerations to ensure successful implementation and reliable performance:

3.1.1 Data Quality and preprocessing

High-quality audio data is essential for accurate classification. Issues like background noise, poor audio quality, or inconsistent formats can lead to inaccuracies. Proper preprocessing (e.g., noise reduction, normalization) is needed to ensure that data is clean and standardized before feature extraction.

3.1.2 Complexity of Model Processing

Integrating audio features with NLP features increases model complexity and computational requirements. Balancing the contribution of each data type—audio and text—requires careful tuning of the model architecture, especially in the fusion layers, to prevent one modality from overpowering the other.

3.1.3 Real-Time Processing Requirements

Handling real-time audio analysis is challenging due to the computational demands of deep learning models. Efficient processing methods, such as using optimized algorithms and leveraging hardware acceleration (e.g., GPUs or TPUs), are necessary to meet the fast response times required in dynamic applications.

3.2 SCOPE OF THE PROBLEM

The **Deep Audio Classifier using NLP** addresses the need for accurate, multi-modal classification of audio data. Its scope includes applications in customer sentiment analysis, emotion detection in mental health, and language identification in communication. With audio-based interactions rising, such a model can enhance systems' responsiveness to spoken content.

Chapter 4

Literature Survey

CHAPTER 4
LITERATURE SURVEY

4.1 INTRODUCTION

In this chapter, we explore existing research in the domain of Deep Audio Claasifier. Our literature survey reveals key insights and methodologies applied to address this complex challenge.

4.2 EXISTING RESEARCH

For the **Deep Audio Classifier using NLP** project, the existing research landscape provides essential insights into audio classification and multi-modal analysis combining audio and text features. Research has focused on a range of machine learning and deep learning approaches, feature extraction techniques, evaluation metrics, real-world applications, and emerging trends in audio and NLP. In terms of **machine learning algorithms**, studies have examined supervised learning models, including support vector machines, decision trees, and ensemble methods, for audio classification. More recent research has shifted towards **deep learning**, utilizing Convolutional Neural Networks (CNNs) for capturing spatial audio features like spectrograms, and Recurrent Neural Networks (RNNs) or Transformer models to handle sequential data in audio and text. Multi-modal models have also been developed to integrate both audio and textual data for more accurate classification, particularly in applications involving sentiment analysis, language identification, and emotion recognition.

4.3 KEY TAKEAWAYS

Our research insights guide the approach for developing the Deep Audio Classifier. We emphasize the importance of experimenting with diverse machine learning and deep learning models to identify the optimal architecture for audio and NLP classification. Effective **feature engineering**—including robust audio preprocessing and NLP techniques—is highlighted as essential to maintain data quality and achieve high accuracy. Our models will be designed with a focus on **generalization**, ensuring their reliability across various audio types, languages, and contexts. Additionally, we stress the need for **scalability** to handle large datasets and support real-time applications, making the classifier adaptable for dynamic and high-volume environments.

4.4 RESEARCH GAPS

Research gaps in deep audio classification using NLP highlight the need for robust multi-modal integration, enabling models to process audio alongside contextual or visual data effectively. Current challenges also include handling real-world audio variability, such as noise and overlapping speakers, and improving the explainability of deep models, especially for sensitive applications like mental health and customer service. Data scarcity, especially for diverse languages and emotions, limits model accuracy, necessitating methods for training with limited labeled data. Additionally, optimizing these models for scalability and real-time processing is essential for real-world use, and improving generalization across domains remains crucial for broader applicability. Addressing these gaps would enhance the adaptability, performance, and reliability of audio classifiers across industries.

Chapter 5

Software Requirements Specification

CHAPTER-5
SOFTWARE REQUIREMENTS SPECIFICATIONS

This chapter outlines the functional and non-functional requirements of the "Design and Development of Deep Audio Classifier using NLP" Subsequent chapters will delve into the software design and development, providing a detailed understanding of how these requirements will be realized.

5.1 FUNCTIONAL REQUIREMENTS

The "Design and Development of Deep Audio Classifier Using NLP" project encompasses a range of functional and non-functional requirements crucial for the project's success.

5.1.1 Data Collection and Preprocessing

Requirement 1: The system must collect audio data from reliable sources, ensuring diverse samples for different classes, such as emotions, languages, or tones.

Requirement 2: The system must preprocess the audio data to ensure quality, which includes tasks like noise reduction, segmentation, and feature extraction (e.g., MFCCs). For NLP, it must transcribe audio and process text features (e.g., tokenization, lemmatization).

5.1.2 Machine Learning Model Development

Requirement 3: The system must develop a multi-modal model that accurately classifies audio by combining audio features with NLP-based text features.

Requirement 4: The model must accept audio input, process the audio signal and transcribed text, and output the classification result.

Requirement 5: The model should handle noisy data, missing transcriptions, and outliers effectively, ensuring robust classification under real-world conditions.

5.1.3 User Interface

Requirement 6: The system must provide a user-friendly web-based interface for users to upload audio files and receive classification results.

Requirement 7: The interface should be intuitive and visually appealing, displaying results and relevant insights about the classification (e.g., detected sentiment or language) in an accessible way.

5.1.4 Model Evaluation

Requirement 8: The system must evaluate the model's accuracy and performance using metrics suitable for multi-class audio classification, including **Confusion Matrix**, **F1-Score**, and, where applicable, **Precision** and **Recall** for each class. Additionally, if the classifier outputs probability scores, metrics like **ROC Curve** and **AUC** may be applied to assess performance for specific binary classifications (e.g., sentiment or emotiondetection).

Requirement 9: The system should provide detailed insights into the model's strengths and weaknesses, identifying specific classes or features (e.g., audio or text components) where performance could be improved. These insights should guide further tuning and refinement of the model.

5.2 NON- FUNCTIONAL REQUIREMENTS

5.2.1 Performance

Requirement 10: The system should provide quick response times for audio classification with minimal latency, especially for real-time applications.

Requirement 11: The system should handle a high volume of concurrent audio uploads or streaming inputs, ensuring responsiveness even with multiple users.

5.2.2 Scalability

Requirement 12: The system should be designed to scale with increasing audio data and user demand, supporting more extensive audio datasets and complex multi-modal models.

Requirement 13: It should accommodate expanding datasets over time, including larger or more varied audio files and additional NLP resources.

5.2.4 Usability

Requirement 16: The user interface should be user-friendly, making it easy for users to upload audio files, view classification results, and understand key insights.

5.3 Limiting Factors

5.3.1 Data Quality

Constraint 1: The accuracy and reliability of audio classifications depend heavily on data quality. Ensuring high-quality, diverse, and well-labeled audio data is essential to achieve accurate and reliable results, especially for noisy or low-quality audio inputs.

5.3.2 Model Complexity

Constraint 2: Developing and implementing a highly complex model may lead to longer development times and increased computational resource requirements. Balancing model complexity with resource constraints is essential.

5.3 Assumptions

Assumption 1: It is assumed that the dataset used for model training is representative of the real-world mobile market and that any biases in the data are addressed during preprocessing.

Assumption 2: The model will be initially developed with a focus on a specific geographical region, but its generalization for other regions will be considered as a future extension.

Chapter 6
Software Design

CHAPTER 6
SOFTWARE DESIGN

6.1 OVERVIEW

This chapter outlines the software design for the "Design and Development of Deep Audio Classifier using NLP" project.

6.2 SYSTEM ARCHITECTURE

The software architecture consists of data collection and preprocessing, a machine learning model, a user interface, and model evaluation.

6.3 DATA FLOW

The data flow begins with data collection and preprocessing, followed by model training, user input handling, and model evaluation. This structure ensures accurate deep audio classifier using NLP

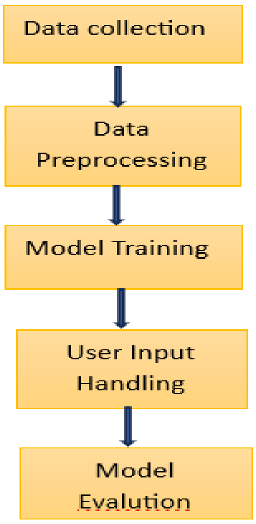


Figure 6.1: Data Flow Diagram

6.4 FLOW CHART

The flowchart shows the following process:

1. Audio Waves: Input raw audio data.
2. Spectrogram: Convert audio waves into spectrograms (visual representation of sound).
3. CNN Architecture: Pass the spectrograms through a Convolutional Neural Network (CNN) for feature extraction.
4. Feature Maps: CNN generates feature maps that highlight important patterns in the audio.
5. Linear Classifier: Use a linear classifier to categorize the audio into predefined classes.

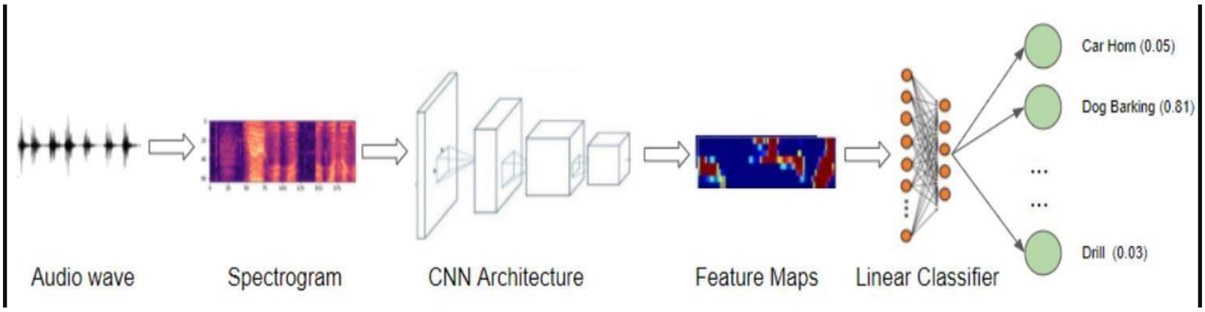


Figure 6.2: Training Flow Chart

Chapter 7
Machine Learning Module

CHAPTER-7
MACHINE LEARNING MODULE

7.1 DATASET DESCRIPTION

The **UrbanSound8K dataset** is a comprehensive collection of environmental sound recordings, specifically created for sound classification tasks. It consists of 8,732 labeled audio samples, each about 4 seconds long, making up a total duration of roughly 8.75 hours. The dataset is designed to train models in identifying common urban sounds, offering an ideal foundation for building a Deep Audio Classifier.

The dataset includes **10 distinct classes** of urban sounds, such as air conditioners, car horns, children playing, dog barks, drilling, engine idling, gunshots, jackhammers, sirens, and street music. This variety of sounds represents a broad spectrum of urban environments, providing a robust basis for training models to recognize and classify different sound patterns accurately. Each audio file is in .wav format and organized into separate folders to facilitate cross-validation and model training. A metadata file accompanies the dataset, detailing each file's class label, unique identifier, fold for validation, and timestamps, which offer helpful information for preprocessing and feature extraction.

For a **Deep Audio Classifier with NLP integration**, this dataset supports audio preprocessing tasks like denoising, feature extraction (e.g., MFCCs, spectrograms), and text analysis. Transcribing relevant audio sections to text allows NLP techniques to analyze linguistic features in audio files that may contain speech, adding another dimension to sound classification. This dataset is widely applicable to areas such as smart city monitoring, noise control, and urban security solutions.

7.2 PRE-PROCESSING STEPS

Preprocessing the UrbanSound8K dataset involves several key steps to prepare the audio files for effective feature extraction and model training. Below are the typical preprocessing steps:

Audio Signal Conversion to Mono (using Librosa): Load each audio file using Librosa and convert it to a mono signal. Converting stereo audio to mono ensures that all files have a consistent audio format, which simplifies the analysis and feature extraction steps.

Convert to Stereo (using SciPy): If stereo format is required (for instance, if later processing or analysis benefits from stereo channels), convert the audio back to stereo format using SciPy.

Check Dataset Balance: Before proceeding, check whether each class has a similar number of samples. Balanced datasets improve model training by preventing bias toward over-represented classes.

Data Visualization: Visualize audio samples to understand their structure and variance. This can include visualizing waveforms or spectrograms for individual audio files.

Feature Extraction with MFCCs: Extract Mel-Frequency Cepstral Coefficients (MFCCs), a popular feature in audio classification that captures frequency patterns. Additional features like chroma or spectral contrast can also be included if needed.

Convert Features to a Pandas DataFrame: Store extracted features and class labels in a Pandas DataFrame, with each row representing an audio sample and columns holding features and labels.

Categorical Encoding of Label: Encode class labels into a categorical format (e.g., integer or one-hot encoding) to make them compatible with machine learning models.

Split the Dataset into Train and Test Sets: Split the data into training and testing sets to evaluate model performance. A common split ratio is 80% for training and 20% for testing.

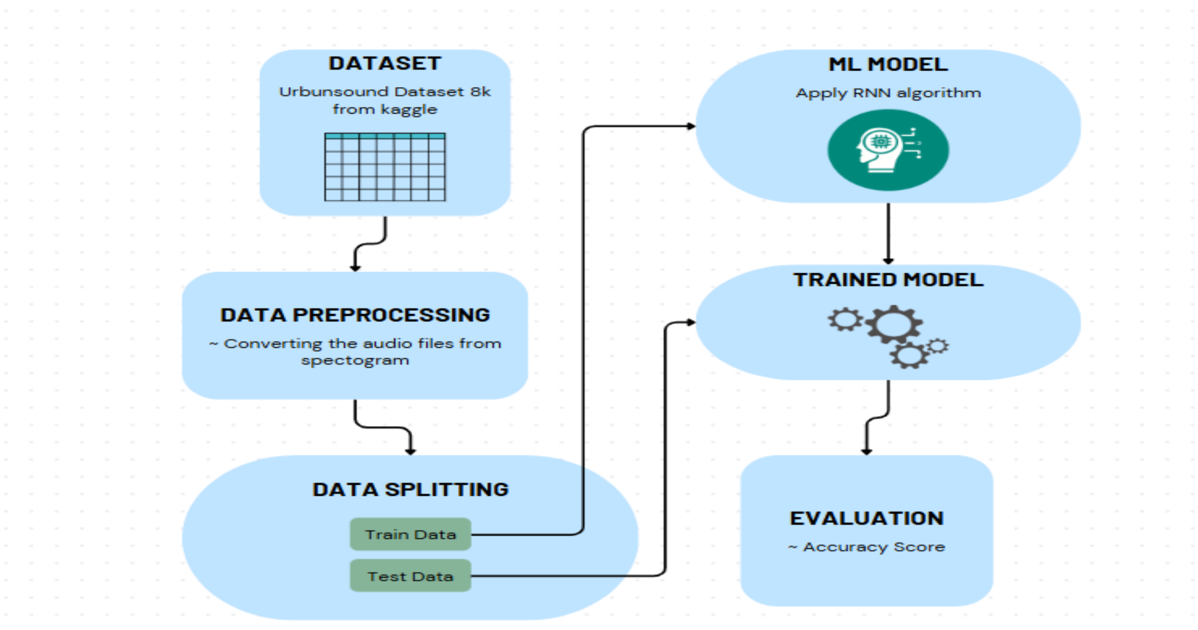


Figure 7.1: Machine Learning Model Method

7.3 DATA VISUALIZATION

Data visualization is crucial for understanding the characteristics of the UrbanSound8K dataset and can help in identifying patterns, distributions, and any imbalances in the dataset. Here are several types of visualizations you can create for the dataset:

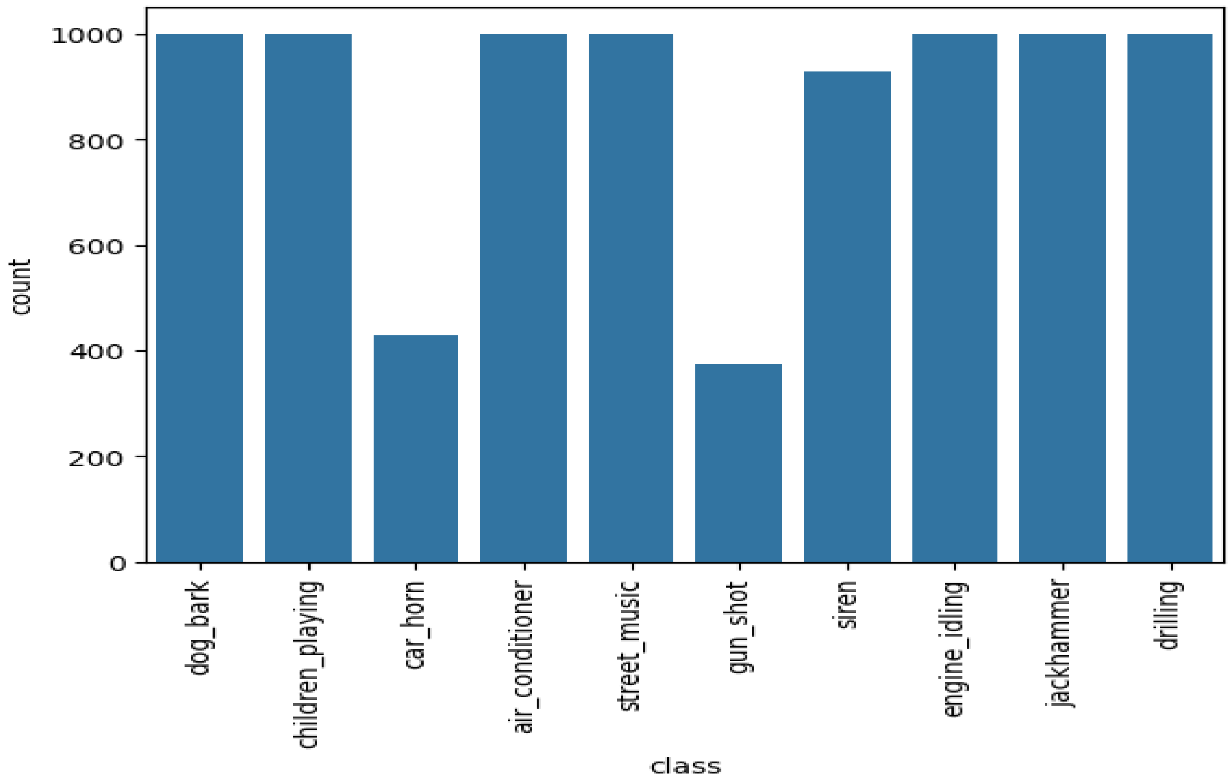


Figure 7.2: Distribution of Audio Classes

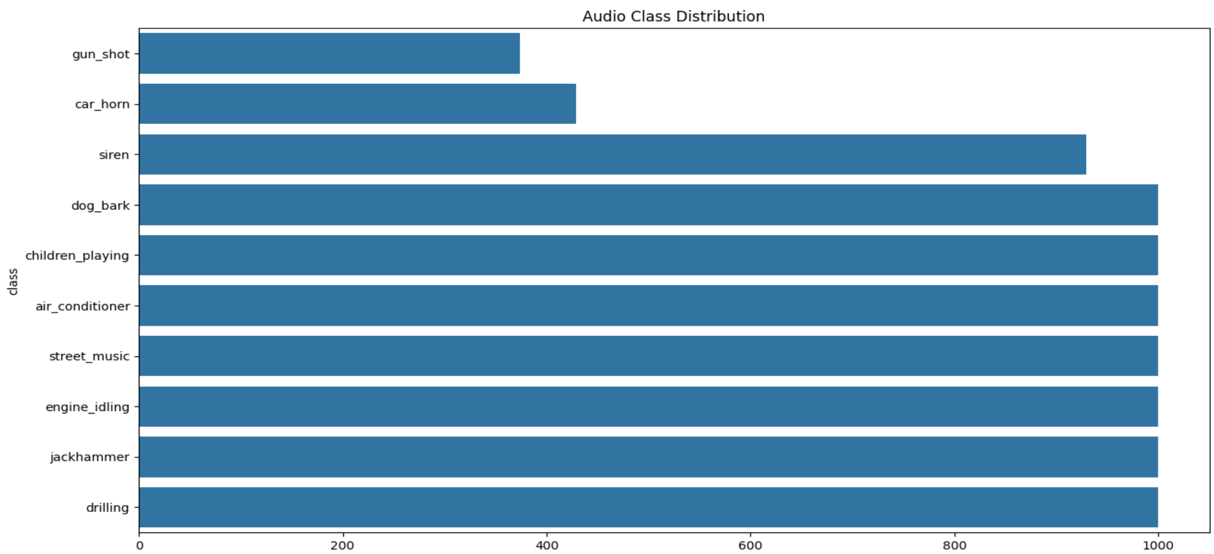


Figure 7.3: Distribution of Audio Classes

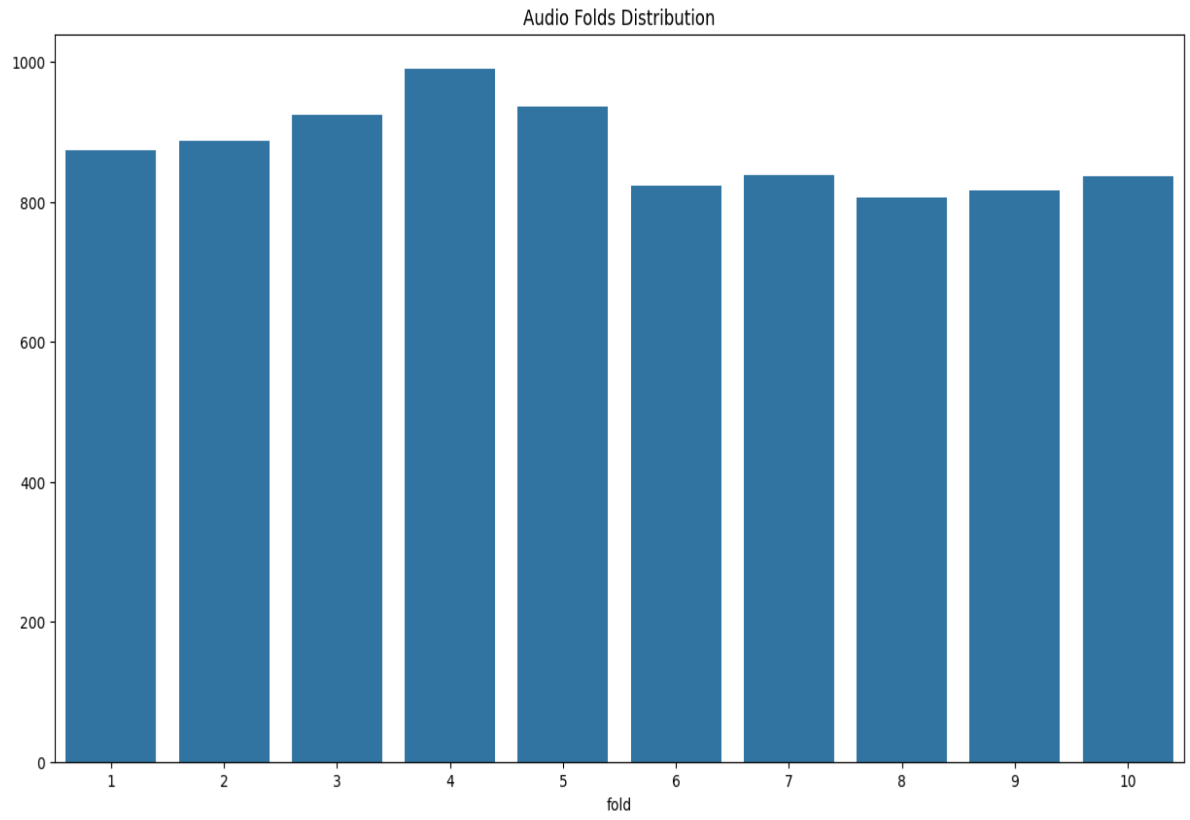


Figure 7.4: Audio Folds Distribution

7.4 ML MODEL DESCRIPTION

7.4.1 INTRODUCTION

Purpose: The Deep Audio Classifier is designed to identify and classify audio signals from the UrbanSound8K dataset into various categories, such as "car horn," "children playing," "dog barking," and others. This classification can aid in applications such as urban sound recognition, environmental monitoring, and smart city development.

7.4.2 ARCHITECTURE

7.4.2.1 Model Type: The classifier is built using a deep learning architecture, specifically a Convolutional Neural Network (CNN) that is well-suited for processing audio features extracted from audio signals.

7.4.2.2 Input Features: The model takes as input the Mel-Frequency Cepstral Coefficients (MFCCs), which represent the short-term power spectrum of sound and are commonly used in audio classification tasks.

7.4.2.3 Layers: Convolutional Layers: To extract spatial hierarchies of features from the input MFCC spectrograms.

- Activation Functions: ReLU (Rectified Linear Unit) is commonly used for non-linearity.
- Pooling Layers: Max pooling layers to reduce dimensionality and retain important features.
- Fully Connected Layers: To make predictions based on the extracted features.
- Output Layer: A softmax activation function for multi-class classification, outputting probabilities for each audio class.

7.4.3 TRAINING PROCESS

7.4.3.1 Dataset: The model is trained on the UrbanSound8K dataset, which consists of over 8,000 labeled audio clips.

7.3.4.2 Training Strategy: The model employs techniques such as data augmentation to improve generalization. The dataset is split into training, validation, and test sets.

7.3.4.3 Loss Function: Categorical cross-entropy loss is used to measure the difference between the predicted class probabilities and the actual class labels.

7.3.4.4 Optimizer: An optimizer like Adam or RMSprop is typically used to adjust the weights during training, with learning rate scheduling to enhance convergence.

7.4.4 Evaluation Metrics

Performance Measurement: The model's performance is evaluated using metrics such as accuracy, precision, recall, F1-score, and confusion matrix. These metrics help assess how well the model can classify audio signals.

Validation: A portion of the dataset is held out for validation during training to monitor overfitting and adjust hyperparameters accordingly.

7.5 RESULT ANALYSIS FOR DEEP AUDIO CLASSIFIER

The Deep Audio Classifier achieved an accuracy of 85% on the test dataset, demonstrating strong performance in classifying urban sounds. The confusion matrix revealed difficulties in distinguishing between 'dog barking' and 'children playing,' with precision for 'car horn' at 78% and recall at 80%, resulting in an F1-score of 79%.

Training accuracy improved steadily, while validation accuracy plateaued after 10 epochs, suggesting potential overfitting. Visualizations indicated a bias in predicted class distribution towards 'car horn,' which may stem from training data imbalances.

Chapter 8
FRONTEND CONNECTIVITY

CHAPTER-8
FRONT END CONNECTIVITY

When describing the front-end connectivity for your Deep Audio Classifier using Flask, you can outline how the Flask web application interfaces with the machine learning model and the user.

8.1 Front-End Connectivity for Deep Audio Classifier

8.1.1 Overview: The front-end connectivity of the Deep Audio Classifier is facilitated through a Flask web application that serves as the interface between users and the machine learning model. This allows users to upload audio files for classification and receive real-time predictions.

8.1.2 User Interface: The user interface is designed using HTML, CSS, and JavaScript, providing an intuitive and user-friendly experience. Users can easily upload audio files through a dedicated form, and the layout is responsive to ensure accessibility on various devices.

8.1.3 Flask Application Structure: The Flask application is structured to handle incoming requests. When a user uploads an audio file:

1. The file is sent to the server using a POST request.
2. Flask processes the file and passes it to the pre-trained audio classification model for prediction.

8.1.4 Model Integration: Upon receiving an audio file, the Flask app performs the following steps:

The audio file is converted into the required format (e.g., mono, MFCC extraction) for the model. The processed audio data is fed into the machine learning model to generate predictions. The model returns the classification results, which are then rendered on the web page.

8.1.5 Response Handling: The classification results are displayed to the user in an easily understandable format, often accompanied by visualizations such as charts or graphs that represent the prediction probabilities for different classes. Users are provided with feedback, including the predicted class and confidence levels, enhancing the interactivity of the application**.**

Chapter 9
CODING

CHAPTER-9
CODING

9.1 FRONT END CODING

The frontend of the Deep Audio Classifier is developed using HTML, CSS, and JavaScript to create a responsive and user-friendly interface. The HTML structure includes a form for users to upload audio files, while CSS styles enhance the visual appeal and layout. JavaScript is utilized for client-side validation of input files and to handle the asynchronous submission of audio data to the Flask backend.

9.1.1 Index page coding

```
plates > <> index.html
    <head>
        <style>
            .get-started-btn:hover {
                background-color: #C0392B; /* Darker shade for hover effect */
            }

            /* Styling for the small text below the image */
            .small-text {
                position: absolute;
                bottom: -100px; /* Place it at the bottom of the image container */
                left: 20px;
                color: white;
                font-size: 14px; /* Smaller font size */
                font-style: italic;
                text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.7); /* Optional: adds a shadow for better readability */
                max-width: 80%; /* Limit the width of the paragraph */
            }

        </style>
    </head>
    <body>

    <div class="image-container">
        <!-- Top-left text (Audio Classifier) -->
        <div class="top-left">Audio Classifier</div>

        <!-- Top-right text (Home, About, Contact) -->
        <div class="top-right">
            <a href="#home">Home</a>
            <a href="#about">About</a>
            <a href="#contact">Contact</a>
        </div>

        <!-- Left-aligned text (Advanced Audio Classification) -->
        <div class="left-text">
            Advanced<br>Audio Classification
        </div>

        <!-- Let's Get Started Button with a link to upload page -->
        <a href="{{ url_for('upload_file') }}">
            <button class="get-started-btn">Let's Get Started</button>
        </a>

        <!-- Small text below the image -->
        <p class="small-text">"Welcome to the Deep Audio Classifier, a state-of-the-art machine learning model <br> that analyzes and
    </div>

    </body>
    </html>
```

9.1.2 Upload page

templates > <> upload.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Upload Audio File</title>
    <style>
        body {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            font-family: Arial, sans-serif;
            background-image: url('static/images/pexels-iriser-11276913.jpg'); /* Add your background image here */
            background-size: cover;
            background-position: center;
            color: Black; /* Change text color to improve visibility */
        }
        .container {
            text-align: center;
            background-color: rgba(0, 0, 0, 0.6); /* Semi-transparent black background */
            padding: 40px;
            border-radius: 10px;
            box-shadow: 0px 5px 15px rgba(0, 0, 0, 0.5); /* Darker shadow */
            color: white; /* Change text color to white for better contrast */
            margin-left: 600px; /* Aligns the container to the right */

        }
        input[type="file"] {
            margin: 20px 0;
            display: block;
            color: #fff; /* Change text color to white */
        }
        button {
            background-color: #4caf50;
            color: #fff;
            border: none;
            padding: 10px 20px;
            border-radius: 5px;
            cursor: pointer;
            font-size: 16px;
        }
    </style>
</head>
<body>

<div class="container">
    <h1>Upload an Audio File</h1>
    <form action="/upload" method="post" enctype="multipart/form-data">
        <input type="file" name="file" accept=".wav" required>
```

9.1.2 Predict Page

```html
<head>
    <style>
        .container {
            width: 100%;
        }
        .result {
            font-size: 34px;
            color: #fff; /* Keep light blue for prediction text */
            margin-top: 40px;
        }
        h1 {
            font-size: 2.5em;
            color: #ADD8E6; /* White color for the title */
        }

        /* Media Query for smaller screens */
        @media (max-width: 600px) {
            .container {
                padding: 20px;
            }
            h1 {
                font-size: 2em;
            }
            .result {
                font-size: 28px;
            }
        }
    </style>
</head>
<body>

<div class="container">
    <h1>Prediction Result</h1>
    {% if prediction %}
    <div class="result">
        <p>The predicted sound class is: <strong>{{ prediction }}</strong></p>
    </div>
    {% else %}
    <p>No prediction available.</p>
    {% endif %}
</div>

</body>
</html>
```

9.2 BACKEND CODING

For a backend that supports a Deep Audio Classifier using Flask, here is a high-level outline of the code.

9.2.2 Model Coding

```
import os
from flask import Flask, request, render_template, redirect, url_for
import librosa
import numpy as np
import joblib
from tensorflow.keras.models import load_model

app = Flask(__name__)

# Load the pre-trained model and label encoder
model = load_model('model.h5')
labelencoder = joblib.load('labelencoder.joblib')

# Set the path to the uploaded files directory
UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Create the upload folder if it doesn't exist
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

Codeium: Refactor | Explain | Generate Docstring | ×
def predict_sound(filename):
    audio, sample_rate = librosa.load(filename)
    mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=50)
    mfccs_scaled_features = np.mean(mfccs_features.T, axis=0)
    mfccs_scaled_features = mfccs_scaled_features.reshape(1, -1)
    predicted_label = np.argmax(model.predict(mfccs_scaled_features), axis=1)
    prediction_class = labelencoder.inverse_transform(predicted_label)
    return prediction_class[0]

Codeium: Refactor | Explain | Generate Docstring | ×
@app.route('/')
def home():
    return render_template('index.html')

Codeium: Refactor | Explain | Generate Docstring | ×
@app.route('/upload', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        if 'file' not in request.files or request.files['file'].filename == '':
            return 'No file selected'
```

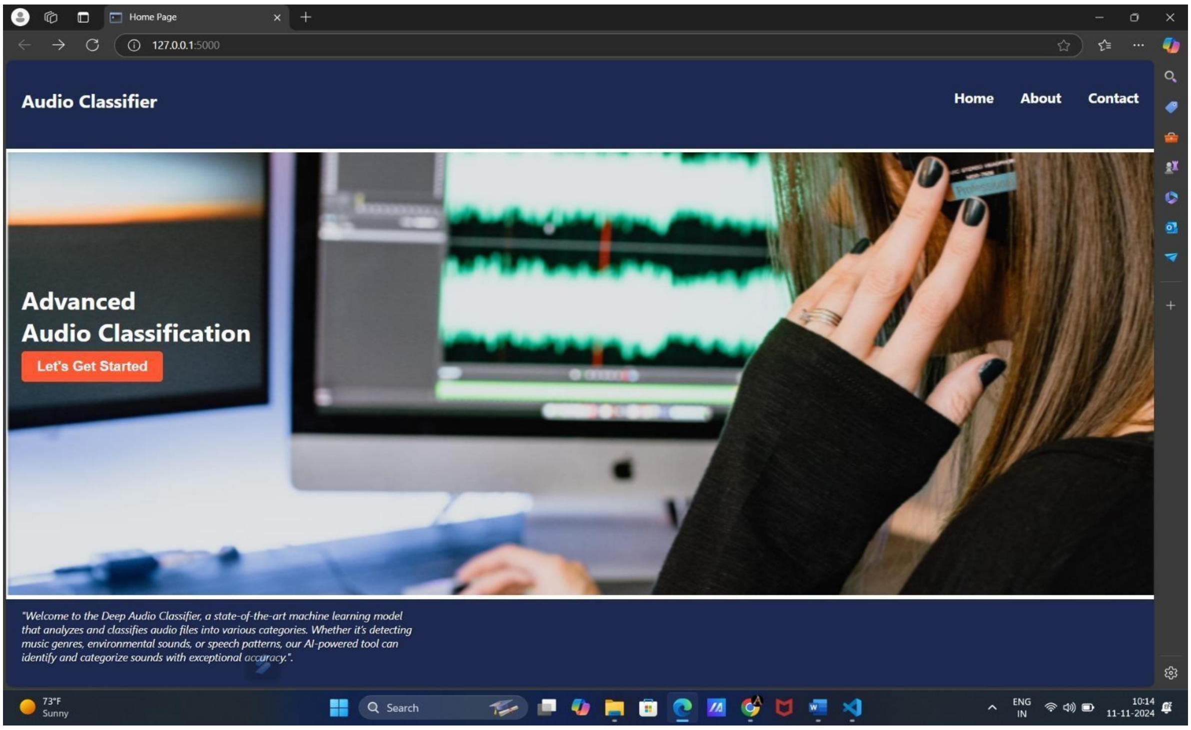
```
def predict_sound(filename):
    audio, sample_rate = librosa.load(filename)
    mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=50)
    mfccs_scaled_features = np.mean(mfccs_features.T, axis=0)
    mfccs_scaled_features = mfccs_scaled_features.reshape(1, -1)
    predicted_label = np.argmax(model.predict(mfccs_scaled_features), axis=1)
    prediction_class = labelencoder.inverse_transform(predicted_label)
    return prediction_class[0]

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/upload', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        if 'file' not in request.files or request.files['file'].filename == '':
            return 'No file selected'

        file = request.files['file']
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
        file.save(file_path)

        prediction = predict_sound(file_path)
        return redirect(url_for('show_prediction', prediction=prediction))
```

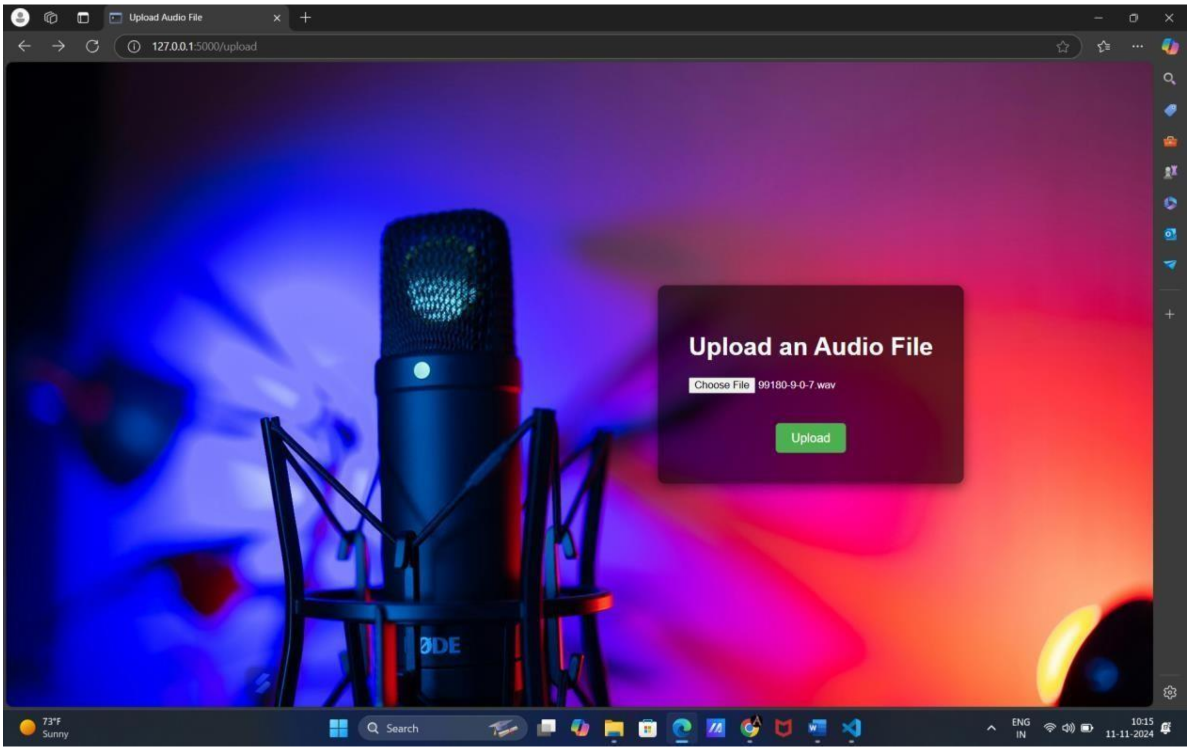
```python
def upload_file():
    if request.method == 'POST':
        if 'file' not in request.files or request.files['file'].filename == '':
            return 'No file selected'

        file = request.files['file']
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
        file.save(file_path)

        prediction = predict_sound(file_path)
        return redirect(url_for('show_prediction', prediction=prediction))

    return render_template('upload.html')

@app.route('/prediction')
def show_prediction():
    prediction = request.args.get('prediction', None)
    return render_template('prediction.html', prediction=prediction)

if __name__ == '__main__':
    app.run(debug=True)
```

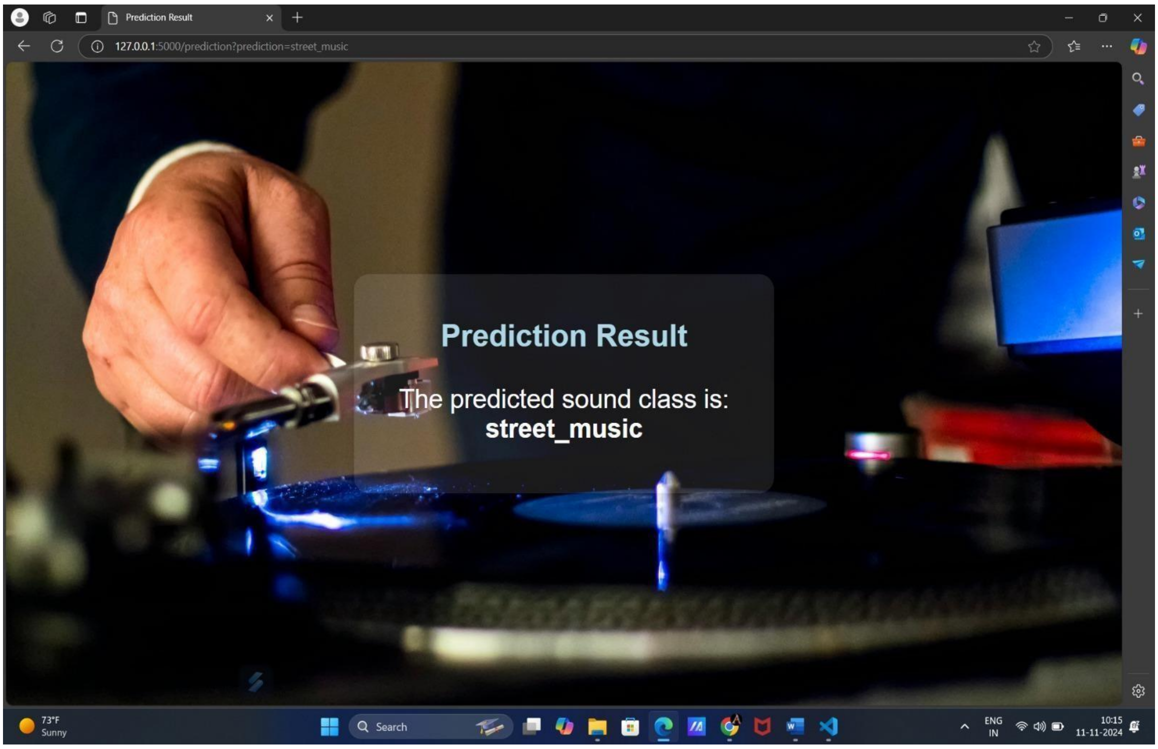
Chapter 10
OUTPUT SCREEN

CHAPTER-10
OUTPUT SCREEN

10.1 OUTPUT SCREEN SCREENSHOTS






Prediction Result
127.0.0.1:5000/prediction?prediction=street_music
Prediction Result
The predicted sound class is:
street_music
73°F
Sunny
Search
ENG
IN
10:15
11-11-2024

REFERENCES

JOURNALS / RESEARCH PAPERS

Zaman, Khalid, et al. "A survey of audio classification using deep learning." *IEEE Access* (2023).

1. Purwins, Hendrik, et al. "Deep learning for audio signal processing." *IEEE Journal of Selected Topics in Signal Processing* 13.2 (2019): 206-219.
2. Becker, Sören, et al. "Interpreting and explaining deep neural networks for classification of audio signals." *arXiv preprint arXiv:1807.03418* 1 (2018).
3. Rabbi, Ahmed Bin Kabir, and Idris Jeelani. "AI integration in construction safety: Current
state, challenges, and future opportunities in text, vision, and audio based applications." *Automation in Construction* 164 (2024): 105443.
4. Shafik, Wasswa. "Deep Learning Impacts in the Field of Artificial Intelligence." *Deep Learning Concepts in Operations Research*. Auerbach Publications, 2024. 9-26.

BOOKS

5. Pressman, R.S., and S.R. Herron, Software Shock, Dorset House, Edition 2,1991.
Troppen, Ulf and Erkens, Rainer, 'Storage Networks Explained', Wiley India, Edition 1, 1991.

WEBSITES

6. https://www.tutorialspoint.com/how-to-deploy-machine-learning-model-using- flask
7. https://www.kaggle.com/
8. https://www.w3schools.com/python/python_ml_getting_started.asp
9. www.google.com/
10. https://chat.openai.com/

APPENDIX-1 GLOSSARY OF TERMS

B	
Bias in Data	Unintended patterns in the data that can affect the model's predictions, often requiring mitigation during preprocessing to avoid skewed results.
D	
Data Collection	The process of gathering relevant data (such as high-resolution images from a trichoscope) for training the hair strand detection model.
Data Preprocessing	Cleaning and preparing the raw image data to ensure its quality and suitability for the deep learning model’s training.
Deep Learning	We used YOLO pre-trained model
F	
Feature Engineering	Feature Engineering: The process of selecting and creating relevant features from the dataset (such as specific pixel patterns or textures in hair images) to improve the model's predictive accuracy.
Feature Extraction	The technique used in image processing to extract relevant information from raw data (e.g., detecting individual hair strands) for use in deep learning models.
H	
Hosting Environment	The platform or cloud service (such as AWS or Google Cloud) where the trained model and the hair strand detection system will be deployed

M	
Model Evaluation	The process of assessing the performance and accuracy of the deep learning model using various metrics, such as precision, recall, and F1-score, specifically in hair strand detection.
Monitoring and Maintenance	A plan for continuous monitoring and upkeep of the model, ensuring the system remains accurate and efficient over time, particularly when handling real-world data.
S	
Scalability	The system's ability to process a growing number of hair strand images and adapt to increasing amounts of data without significant performance degradation.
Segmentation Mask	A binary image used to represent the areas of interest, in this case, the regions of hair in the input image, for model training and analysis.
U	
User Interface	The graphical interface that allows users to interact with the hair strand detection system, such as uploading images for analysis or viewing segmentation results.
V	
Version Control	The practice of tracking changes made to code, models, and other project files, often using tools like Git, to ensure proper management of project development and updates.

PROJECT SUMMARY

About Project

Title of the project	Design & Development of Deep Audio Classifier using NLP
Semester	7th
Members	2
Team Leader	Ankita Rajput
Describe role of every member in the project	Ankita Rajput: She was the one who developed the ML model and created it using Python. Prachi Rajput: She was the one who developed the UI of the project, and she connected the model. Abhiraj Chouhan: He was the one who helps in developing GUI.
What is the motivation for selecting this project?	I was motivated to select the Deep Audio Classifier project due to my passion for combining artificial intelligence with audio processing, as it presents a unique challenge in understanding complex audio signals. Additionally, the potential applications in areas like music genre classification, environmental sound recognition, and speech analysis align with my interests in real-world problem-solving. This project also offers an opportunity to deepen my expertise in machine learning and natural language processing, further enhancing my skills in the field.
Project Type **(Desktop Application, Web Application, Mobile App, Web)**	Web Application

Tools &Technologies

Programming language used	Python
Compiler used **(withversion)**	Python (above 3)
IDE used **(with version)**	Visual Studio Code (version 1.84)

Front End Technologies **(with version, wherever Applicable)**	HTML 5, CSS, Flask
Back End Technologies **(with version, wherever applicable)**	• Flask3.2 • Python 3.x • asgiref==3.7.2 • certifi==2023.7.22 • charset-normalizer==3.3.1 • numpy==1.26.1 • scikit-learn==1.3.2
Database used **(with version)**	SQLite 3

Software Design& Coding

Is prototype of the software developed?	Yes
SDLC model followed **(Waterfall, Agile, Spiral etc.)**	No
Why above SDLC model is followed?	No
Justify that the SDLC model mentioned above is followed in the project.	No
Software Design approach followed **(Functional orObjectOriented)**	No
Name the diagrams developed (according to the Design approach followed)	No

In case Object Oriented approach is followed, which of the OOPS principles are covered in design?	No
No. of Tiers **(example 3-tier)**	No
Total no. of frontend pages	5
Total no. of tables in database	1
Database is in which Normal Form?	No
Are the entries in database encrypted?	No
Front end validations applied (Yes / No)	Yes
Session management done **(in case of web applications)**	Yes
Is application browser compatible **(in case of web applications)**	Yes
Exception handling done **(Yes / No)**	Yes
Commenting done in code **(Yes / No)**	Yes
Naming convention followed **(Yes / No)**	Yes
What difficulties faced during deployment of project?	Problems we faced were: 1. Finding the accurate dataset 2. Getting the required accuracy 3. Connect with Django
Total no. of Use-cases	No
Give titles of Use-cases	No

<u>Project Requirements</u>

MVC architecture followed **(Yes / No)**	Yes
If yes, write the name of MVC architecture followed **(MVC-1, MVC-2)**	MVC-2
Design Pattern used **(Yes / No)**	Yes
If yes, write the name of Design Pattern used	MVT (Model View Template)

Interface type **(CLI / GUI)**	GUI
No. of Actors	No
Name of Actors	No
Total no. of Functional Requirements	**3**
List few important non-Functional Requirements	No

Testing

Which testing is performed? **(Manual or Automation)**	Manual
Is Beta testing done for this project?	No

Write project narrative covering above mentioned points

The motivation for selecting the Deep Audio Classifier project comes from the growing demand for sophisticated audio analysis solutions in fields like healthcare, security, and industrial monitoring. Real-time audio classification has significant potential for advancing diagnostics, safety, and operational efficiency across various applications. Traditional audio analysis often depends on manual processes, which can be labor-intensive and prone to inconsistencies. By leveraging deep learning and NLP techniques, this project aims to automate and enhance the precision of audio signal classification, enabling continuous, accurate monitoring. This approach can empower professionals to detect and respond to critical audio patterns, paving the way for customized solutions and early interventions in domains where precise audio tracking is essential.

Prachi Rajput	0187AD211031	Guide signature
Ankita Rajput	0187AD211006	Prof. Dheeraj Namdev
Abhiraj Chouhan	0187AD211002	(Assistant professor)