Buffaloes Behavioral Classification Using Machine Learning Approach

A Project Report

Submitted to the FACULTY of ENGINEERING of

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA

In partial fulfillment of the requirements,

for the award of the Degree of

Bachelor of Technology

In

Electronics and Communication Engineering

By

P. BHARGAV SAI **(20481A04J3)**

S. PRASANTH **(20481A04L6)**

Y. RAJANI **(20481A04O6)**

SRI VENKATA KASYAP.CH **(20481A04L9)**

Under the Guidance of

Dr. S. Ravi

Associate Professor

Department of Electronics and Communication Engineering
SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE
(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)
SESHADRI RAO KNOWLEDGE VILLAGE
GUDLAVALLERU - 521356
ANDHRA PRADESH

2020-2024

Buffaloes Behavioral Classification Using Machine Learning Approach

A Project Report

Submitted to the FACULTY of ENGINEERING of

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA

In partial fulfillment of the requirements,

for the award of the Degree of

Bachelor of Technology
In
Electronics and Communication Engineering
By

P. BHARGAV SAI

(20481A04J3)

S. PRASANTH

(20481A04L6)

Y. RAJANI

(20481A04O6)

SRI VENKATA KASYAP. CH

(20481A04L9)

Under the Guidance of

Dr. S. Ravi

Associate Professor

Department of Electronics and Communication Engineering
SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE
(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)
SESHADRI RAO KNOWLEDGE VILLAGE
GUDLAVALLERU - 521356
ANDHRA PRADESH

2020-2024

Department of Electronics and Communication Engineering

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)
SESHADRI RAO KNOWLEDGE VILLAGE

GUDLAVALLERU – 521356

CERTIFICATE

This is to certify that the project report entitled **"Buffaloes Behavioral Classification using Machine Learning Approach"** is a bonafide record of work carried out by **P. Bhargav Sai (20481A04J3), S. Prasanth (20481A04L6), Y. Rajani (20481A04O6), Sri Venkata Kasyap. Ch (20481A04L9)** under my guidance and supervision in partial fulfillment of the requirements, for the award of the degree of Bachelor of Technology in **Electronics and Communication Engineering** of **Seshadri Rao Gudlavalleru Engineering College** affiliated to **Jawaharlal Nehru Technological University, Kakinada**.

(Dr. S. Ravi)

Project Guide

(Dr. B. Rajasekhar)

Head of the Department

Acknowledgement

We are very glad to express our deep sense of gratitude to **Dr. S. Ravi**, Associate Professor, Electronics and Communication Engineering for guidance and cooperation for completing this project. We convey our heartfelt thanks to him for his inspiring assistance till the end of our project.

We also wish to express our deep sense of gratitude to **Dr. D. N. V. S. L. S. Indira,** Professor and Head of the Department of Information Technology, for her encouragement and help in completing this project.

We convey our sincere and indebted thanks to our beloved Head of the Department **Dr. B. Rajasekhar**, for his encouragement and help for completing our project successfully.

We also extend our gratitude to our Principal **Dr. Burra Karuna Kumar**, for the support and for providing facilities required for the completion of our project.

We impart our heartfelt gratitude to all the Lab Technicians for helping us in all aspects related to our project.

We thank our friends and all others who rendered their help directly and indirectly to complete our project.

P. BHARGAV SAI (20481A04J3)

S. PRASANTH (20481A04L6)

Y. RAJANI (20481A04O6)

SRI VENKATA KASYAP. CH (20481A04L9)

CONTENTS

LIST OF FIGURES

ABSTRACT

Domesticated buffaloes communicate through a diverse range of vocalizations, offering valuable insights into their behaviour, emotions, and overall well-being. Effectively interpreting these sounds is crucial for efficient livestock management. In this study, we address this need by employing machine learning techniques, specifically Support Vector Machines (SVM) and Random Forest, to comprehensively classify buffalo sounds.

The primary objective of our research is to delve into the nuanced classification of buffalo vocalizations, aiming for a profound understanding of their behaviour in various scenarios. By incorporating both SVM and Random Forest, we focus on developing a machine learning-based system proficient in identifying specific buffalo behaviours, including hunger, anger, and snoring, based on their distinctive vocalizations. To achieve this, strategically placed audio recording devices capture a diverse range of buffalo sounds across different environmental conditions.

The proposed approach not only enriches the broader field of animal behaviour analysis but also extends its application to the unique context of buffalo vocalizations. The utilization of SVM, a robust machine learning algorithm, along with Random Forest for model training, enhances the precision and efficiency of the classification process. The outcomes of this research hold practical implications for monitoring buffalo health, welfare, and environmental conditions, offering potential benefits to livestock management practices.

KEYWORDS:

- Support Vector Machines (SVM)
- Python Libraries
- Visual Studio Code
- Random Forest
- Audio Recording Devices
- Health Monitoring

CHAPTER 1

INTRODUCTION

The research project, named "Buffalo Sound Identification, through Machine Learning" is leading the way in deciphering the language of vocalizations. This endeavor has implications for studies, veterinary medicine and livestock management as it explores the diverse range of buffalo communication. At its essence the importance of Buffalo Sound Identification lies in its capacity to interpret the meanings conveyed in the vocalizations of these creatures.

Beyond mere sounds, these vocalizations function as a mode of communication intricately woven into the fabric of animal interactions. The project focuses on understanding and categorizing these expressions to uncover insights into buffaloes' states, reproductive behaviors, distress signals and various other aspects of their behavior. By using machine learning methods to analyze buffalo sounds, the project enhances our understanding of their behavior. From expressions of hunger to distress calls, from indicators of illness to dynamics within buffalo herds. Provide valuable insights into the intricate lives of these animals. The goal is to bring a perspective to studying buffalo behavior by harnessing state-of-the-art technology.

By examining and categorizing the sounds they make the project aims to offer insights that can guide practices, improve animal healthcare and streamline the administration of buffalo groups. In this endeavor it seeks to close the divide between the animal world and human comprehension by nurturing a bond with these beings.

This project is based on a particular breed of buffaloes. The **Murrah buffalo** is a breed known for its distinctive features, including:

1. **Large Size**: Murrah buffaloes are typically large, with mature females weighing between 450 to 550 kilograms (about 1212.54 lb.) and males weighing between 600 to 800 kilograms (about 1763.7 lb.).

2. **Black Color**: They have a predominantly black coat, although sometimes they may have white markings on the face or legs.

3. **Curved Horns**: Murrah buffaloes have large, curved horns that extend outward and slightly backward from the head.

4. **Strong Build**: They have a sturdy and robust build, with well-developed muscles, especially in the shoulders and hindquarters.

5. **High Milk Yield**: Murrah buffaloes are renowned for their high milk production, making them a popular choice for dairy farming. They can yield between 8 to 16 liters of milk per day, with some exceptional individuals producing even more.

6. **Adaptability**: They are well-adapted to various climates and can thrive in different environmental conditions, although they prefer tropical and subtropical regions.

7. **Docile Temperament**: Murrah buffaloes are known for their calm and docile temperament, making them easier to handle and manage compared to some other breeds.

Overall, the buffalo is renowned for its exceptional milk-producing abilities, making it highly sought after by dairy farmers. With a higher average milk yield compared to other buffalo breeds, the Murrah plays a crucial role in meeting the ever-growing demand for dairy products. Its ability to consistently produce large quantities of high-quality milk contributes significantly to the profitability and sustainability of dairy operations.

In addition to its impressive milk yield, the buffalo possesses a robust and resilient build that enables it to thrive in diverse environmental conditions. Originating from the Indian subcontinent, where temperatures can vary greatly, the Murrah has developed a remarkable adaptability to different climates and terrains. This adaptability makes it well-suited for a wide range of agro-climatic regions, providing dairy farmers with a versatile and reliable animal for their operations.

1.1 Aim of the Project:

The main aim of this project is to help the cattle farmers whether their buffalo is in good condition and know the animal better.

1.2 Objective:

The primary objectives of animal sound classification using machine learning are to develop an accurate and robust model capable of identifying and classifying diverse animal vocalizations. This involves collecting a comprehensive dataset encompassing various species and environmental conditions, preprocessing the audio data to extract relevant features, and implementing state-of-the-art machine learning algorithms for effective classification.

The model should be capable of distinguishing between different animal sounds, including calls, cries, and other vocalizations, while also considering factors such as background noise and variations in recording conditions. Additionally, the system should be scalable, allowing for continuous improvement through iterative model refinement and adaptation to new species or environments, ultimately contributing to wildlife monitoring, conservation efforts, and biodiversity research.

1.3 Related work on the project:

Previous investigations in the realm of animal behaviour classification have set the stage for comprehending and deciphering vocalizations across a spectrum of species. In wildlife biology studies, machine learning techniques have been explored to classify and analyse animal sounds, aiming to unravel intricate behavioural patterns. While much attention has been given to wild species, the application of these insights to domesticated animals, especially buffaloes, has been relatively limited. Recent strides in research have highlighted successful applications of Support Vector Machines (SVM) for animal sound classification, showcasing SVM's effectiveness in distinguishing between various vocalizations. These studies serve as a valuable precedent for our Buffalo Behaviour Classification project.

Moreover, in the realm of precision livestock farming, there is a burgeoning interest in harnessing technology to monitor and enhance animal well-being. Research endeavours utilizing audio recording devices and machine learning for the classification of livestock sounds have yielded promising outcomes. However, the unique intricacies of buffalo vocalizations and their behavioural implications necessitate a dedicated exploration. The convergence of machine learning, audio technology, and livestock management represents a dynamic field of study, and our project seeks to make a significant contribution by focusing specifically on buffalo behaviour classification, offering insights into their emotions, health, and social dynamics within herds.

1.4 Implementation Details:

1.4.1. Software Details

a. Python Libraries:

1) OS:

- **Introduction:** The **OS** library in Python provides a way to interact with the operating system. It offers a range of functions to perform tasks like file and directory manipulation, working with file paths, and executing system commands.
- **How to Use:**

  Example:

  ```
  import os
  current_directory = os.getcwd()
  file_list = os.listdir(current_directory)
  ```
- **Purpose of Using:** Facilitates platform-independent interaction with the operating system, enabling file and directory operations.
- **Advantages:**
  - ➢ Platform Independence: Works seamlessly across different operating systems.
  - ➢ Versatility: Supports a wide range of file and directory operations.
  - ➢ Accessibility: Comes pre-installed with Python, no need for additional installations.

2) librosa Library:

- **Introduction:** Librosa is a Python library designed for music and audio analysis. It provides tools for feature extraction, manipulation, and analysis of audio signals.

  How to Use:

  Example:

  ```
  import librosa
  audio, sr = librosa.load('audio_file.wav')
  spectrogram = librosa.feature.melspectrogram(y=audio, sr=sr)
  ```

- **Purpose of Using:** Specifically tailored for audio analysis, offering functionality for feature extraction, pitch analysis, and more.
- **Advantages:**
    - ➢ Specialized Features: Focuses on audio-related tasks with specialized functions.
    - ➢ Open Source: Community-driven development with continuous improvement.
    - ➢ Integration: Easily integrates with other scientific computing libraries like NumPy.

3) numpy Library:

- **Introduction:** NumPy is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on them.

  How to Use:

  Example

  ```
  import numpy as np
  array = np.array([1, 2, 3, 4])
  mean = np.mean(array)
  ```
- **Purpose of Using:** Efficient handling of numerical data through powerful array and matrix operations.
- **Advantages:**
    - ➢ Array Operations: Efficient numerical operations on arrays and matrices.
    - ➢ Broadcasting: Allows operations on arrays of different shapes and sizes.
    - ➢ Integration: Widely used in scientific and machine learning applications.

4) **sklearn Library:**

- **Introduction:** Scikit-learn, commonly referred to as sklearn, is a machine learning library for Python. It provides simple and efficient tools for data mining and data analysis, built on NumPy, SciPy, and Matplotlib.

  How to Use:

  Example:

  ```
  from sklearn.model_selection import train_test_split
  X_train, X_test, y_train, y_test = train_test_split(features, labels,
  test_size=0.2)
  ```
- **Purpose of Using:** Offers a range of machine learning algorithms, preprocessing tools, and model evaluation functions.
- **Advantages:**
  - User-Friendly: Simple and consistent API for various machine learning tasks.
  - Versatility: Supports a wide range of machine learning algorithms.
  - Integration: Seamless integration with other scientific libraries like NumPy.

5) **SVC (Support Vector Classification):**

- **Introduction:** SVC is a classification algorithm provided by scikit-learn based on Support Vector Machines. It works well for both binary and multi-class classification problems.

  How to Use:

  Example:

  ```
  from sklearn.svm import SVC
  model = SVC(kernel='linear')
  model.fit(X_train, y_train)
  ```
- **Purpose of Using:** Designed for classification tasks using Support Vector Machines, separating data into different classes.
- **Advantages:**
  - Versatility: Effective for both linear and non-linear classification problems.
  - Kernel Trick: Can handle complex decision boundaries through kernel functions.

b. Algorithms in Machine Learning:

I. **Scikit-learn (sklearn):** In Python, Scikit-learn is a large machine learning library. It was employed as the main data preprocessing, model training and evaluation framework for buffalo sound classification.

II. **Support Vector Machines (SVM):** For classification and regression tasks, SVM is a strong supervised learning algorithm. In the project, SVM was part of Scikit-learn toolkit and could be used as an alternative classifier for buffalo sound classification that had greater flexibility with robust performance in managing complex data patterns.

III. **Random Forest:** Random Forest is an ensemble learning algorithm that constructs multiple decision trees during training time and predicts the class that has the highest frequency across all trees for classification tasks.

c. Visual Studio Code (VS Code): Visual Studio Code (VS Code) is a popular source code editor developed by Microsoft. It has gained widespread adoption among developers due to its versatility, extensibility, and efficiency.

Features and Functionality: Extensive support for programming languages, with built-in IntelliSense providing smart completions based on variable types, function definitions, and imported modules. Integrated Git control, enabling seamless source code management.

Debugging capabilities with breakpoints, call stacks, and an interactive console for debugging JavaScript, TypeScript, and other languages. Extensibility through a vast ecosystem of extensions, allowing users to customize their development environment according to their needs.

User Interface: Layout customization, including the ability to split editors vertically or horizontally. Exploration of the sidebar functionalities, such as file explorer, search, and source control. Usage of the Activity Bar for quick access to different views and functionalities like debugging, extensions, and settings.

Productivity Tools: Overview of features like snippets, which allow users to insert frequently used code snippets with shortcuts. Discussion on keyboard shortcuts and customization options to optimize workflow efficiency. Integration with task runners and build systems, enhancing automation and project management.

Tips, Tricks, and Advanced Usage: Exploring advanced debugging techniques, such as conditional breakpoints and debug configurations. Utilizing VS Code's integrated terminal for running commands, scripts, and accessing command-line tools. Maximizing productivity with multi-cursor editing, code folding, and Zen mode for distraction-free coding sessions.

Extensions Ecosystem: Overview of popular extensions for specific programming languages, frameworks, and tools. Discussion on extension development and the process of creating custom extensions to enhance VS Code's functionality.

Community and Support: Resources for learning VS Code, including official documentation, tutorials, and community forums. Discussion on VS Code's active community and how users can contribute through feedback, bug reports, and extension development.

Integration with Other Microsoft Products: Integration with Azure services for cloud development and deployment. Usage of Visual Studio Live Share for collaborative coding sessions. Connectivity with Microsoft's developer ecosystem, including Azure DevOps and GitHub.

Cross-platform Compatibility: Discussion on how VS Code maintains consistent performance and user experience across different operating systems, including Windows, macOS, and Linux.

Future Developments and Updates: Insights into the future roadmap of Visual Studio Code, including planned features and improvements. Discussion on how VS Code adapts to evolving trends in software development and technology.

Here's an overview covering its key features and why it matters:

1. **Cross-Platform Compatibility:** VS Code is compatible with Windows, macOS, and Linux, making it accessible to a wide range of developers regardless of their operating system preference.

2. **Feature-Rich Editing:** It offers a rich set of features including syntax highlighting, code snippets, IntelliSense (code completion), and debugging support, enhancing developer productivity.

3. **Extensibility:** VS Code's extension marketplace allows developers to customize and extend its functionality according to their specific needs. There are thousands of extensions available for various programming languages.

4. **Integrated Terminal:** The built-in terminal allows developers to run command-line tools and scripts without leaving the editor, streamlining the development workflow.

5. **Version Control Integration:** It seamlessly integrates with version control systems like Git, providing features such as visual diffing, commit history, and branch management within the editor.

6. **Task Automation:** VS Code supports task automation through tasks.json and launch.json files, enabling developers to automate common development tasks and configurations.

7. **Live Share Collaboration:** Live Share extension allows real-time collaborative editing and debugging, enabling developers to collaborate with team members or conduct remote pair programming sessions.

8. **Built-in Git:** VS Code includes built-in Git support, allowing developers to clone repositories, stage changes, commit code, and perform other Git operations without relying on external tools.

9. **Integrated Development Environment (IDE) Features:** Despite being lightweight, VS Code provides IDE-like features such as code navigation, refactoring, and built-in support for popular frameworks and languages.

10. **Community and Support:** With a large and active community, developers can find extensive documentation, tutorials, and community-driven support channels to help them leverage VS Code effectively.

In conclusion, Visual Studio Code matters because it offers a powerful, flexible, and customizable development environment that caters to the diverse needs of developers across different platforms and programming languages.

The Impact and Future of VS Code

Visual Studio Code (VS Code) has undoubtedly revolutionized the landscape of software development since its inception. With its intuitive interface, extensive plugin ecosystem, and powerful features, VS Code has become the tool of choice for millions of developers worldwide. Throughout this exploration, we've delved into the various aspects that make VS Code a standout in the realm of integrated development environments (IDEs). From its seamless integration with version control systems like

Git to its robust debugging capabilities, VS Code empowers developers to write, debug, and deploy code with unparalleled efficiency.

Moreover, the open-source nature of VS Code has fostered a vibrant community of contributors who continually enhance its functionality through extensions and updates. This collaborative spirit has propelled VS Code to new heights, ensuring that it remains at the forefront of innovation in the ever-evolving field of software development. Looking ahead, the future of VS Code appears promising. As technology advances and new programming languages and frameworks emerge, VS Code is poised to adapt and evolve accordingly. Its versatility and extensibility make it well-suited to meet the demands of future development challenges, while its user-friendly interface ensures that both novice and experienced developers alike can harness its power effectively.

In conclusion, VS Code has not only transformed the way we write code but has also become an integral part of the developer toolkit. Its impact on the software development industry is undeniable, and its influence will continue to shape the way we build and deploy software for years to come. As we move forward, one thing is certain: VS Code will remain a cornerstone of modern software development, driving innovation and empowering developers to turn their ideas into driving innovation and empowering developers to turn their ideas into reality.

d. **Twilio App:** Twilio is a cloud communications platform whose REST API provides a way of sending SMS notifications. Twilio’s REST API is like a magic wand for communication. With just a few lines of code, we can tap into its vast capabilities and send SMS notifications straight to your phone. It’s like having a direct line to the buffalo world, where every grunt and snort is translated into a message you can understand.

1.5 MFCC Calculation Steps:

- **Pre-emphasis:**
  - Y(t)=x(t) - α*x(t-1), where $x(t)$ is the input signal, $y(t)$ is the output signal, and α is the pre-emphasis coefficient.
- **Framing**:
  - The signal is divided into frames, and each frame is multiplied by a window function, such as the Hamming window
  - $w(n)$=0.54−0.46·cos(2πn/N-1), $w(n)$ is the value of the window at index n, N is the length of the window.
- **Fast Fourier Transform (FFT):**
  - Apply the FFT to each framed signal to obtain the frequency domain representation.
  - $X(k)$=∑n=0 to N−1 ($x(n)$·e−N2πikn)
    - *Where* $X(k)$ is the complex value representing the frequency component at bin k,
    - N is the number of samples in the frame,
    - i is the imaginary unit (i^2=−1),
    - n ranges from 0 to N−1.
- **Mel- filter bank:**
  - Apply a set of Mel -filters to the power spectrum obtained from the FFT. The Mel filter bank response $Hm(f)$ for filter m is calculated using:

$$H_m(f)=\begin{cases} 0 & \text{if } f < f_{m-1} \\ (f - fm)/(fm - fm_{-1}) & \text{if } f_{m-1} \leq f < fm \\ 1 & \text{if } fm \leq f \leq f_{m+1} \\ (f_{m+1}-f)/(f_{m+1}-f_m) & \text{if } fm \leq f < f_{m+1} \\ 0 & \text{if } f >= f_{m+1} \end{cases}$$

  - where fm is the center frequency of the m_{th} filter.

- **Logarithm:**
  - Take the natural logarithm of the filter bank energies.
  - Log-Energy(*m*)= ln (Filter bank Energy(*m*))
    - Here, *m* represents the index of the filter bank. Log-Energy is the natural logarithm of the energy calculated for each filter bank. This step is taken to approximate the logarithmic perception of loudness by the human auditory system.
- **Discrete Cosine Transform (DCT):**
  - Apply the DCT to the log filter bank energies to obtain the final MFCC coefficients. The formula for the n_{th} DCT coefficient is:
  - Cn=sqrt(2/M) ·∑ m=1 to M (log (Em)·cos[[πn(2m−1)]/2M) where *Em* is the energy of the m_{th} filter bank, and *M* is the number of filter banks.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING STUDY:

“Machine learning algorithms for predicting peak yield in buffaloes using linear traits” by SUNESH1 et al.

Focuses on creating predictive models for peak milk yield using machine learning algorithms such as ANN, SVMR, and Random Forest. Utilizes data on linear traits of buffaloes across multiple lactations, collected from organized and private farms. Models map linear traits to peak milk yield, aiming to accurately predict peak milk yield.

Advantages:

- Provides a systematic approach to predicting peak milk yield, optimizing animal selection.
- Utilizes multiple machine learning algorithms for robust model creation.
- Offers insights into the relationship between linear traits and milk yield.

Disadvantages:

- Relies on manual data collection of linear traits, leading to potential human error and variations.
- Focuses on physical traits and milk yield, potentially overlooking other aspects of buffalo behavior and welfare.

“Dairy Buffalo Behavior: Calving, Imprinting, and All suckling”

How the System Works:

Investigates specific behaviors in dairy buffaloes, such as calving, imprinting, and all suckling. Observes and documents buffalo maternal behaviors that impact parturition, lactation, and the welfare of the dam and calf. Provides insights into the success of dairy farms through understanding buffalo behavior during critical life stages.

Advantages:

- Offers valuable information on key behaviors affecting the health and welfare of both the dam and calf.
- Provides insights that can guide management practices for improved farm success.

Disadvantages:

- May rely on manual observation and documentation, which can be time-consuming and subject to observer bias.
- Focuses primarily on calving and lactation, potentially missing other aspects of buffalo behavior.

2.2 PROPOSED WORK:

"Buffalo Behavior Prediction Using Machine Learning Approach"

Proposed work focuses on using machine learning techniques to classify buffalo vocalizations, offering insights into their behaviors, emotions, and overall well-being. Employs advanced algorithms like Support Vector Machines (SVM) and Random Forest to classify various buffalo sounds, such as hunger, anger, and snoring. Collects audio recordings of buffalo vocalizations in different environments to capture a diverse range of sounds. Trains model to interpret buffalo vocalizations, enabling effective monitoring of buffalo health and welfare.

Advantages:

- Offers a novel approach to understanding buffalo behavior and welfare through vocalization analysis.
- Utilizes advanced machine learning techniques for precise and automated classification of buffalo behaviors.
- Provides a more comprehensive view of buffalo welfare by focusing on vocalizations, which may reveal subtleties in buffalo behavior not easily observed through other means.
- Can be scaled and applied broadly, with potential applications in improving livestock management practices.

- Reduces the need for manual data collection and observation, potentially improving efficiency and accuracy.

Conclusion:

Incorporating machine learning techniques into buffalo vocalization analysis represents a significant advancement in the field of livestock management. By harnessing the power of artificial intelligence, we can delve deeper into understanding buffalo behaviors and welfare, going beyond traditional methods. Machine learning enables us to uncover intricate patterns and nuances within vocalizations that may otherwise go unnoticed, paving the way for more accurate and insightful analyses.

Our focus on buffalo vocalizations complements and enhances existing studies in the field, particularly those concerning peak milk yield prediction and maternal behavior. While these studies provide valuable insights into specific aspects of buffalo health and productivity, our approach offers a holistic view by considering vocalizations as a key indicator of overall well-being. By integrating multiple sources of data, we can paint a more comprehensive picture of buffalo health, behavior, and welfare, leading to more informed decision-making in livestock management.

The innovative nature of our approach holds immense promise for revolutionizing livestock management practices. By leveraging machine learning to analyze buffalo vocalizations, we can offer precise, automated, and comprehensive insights into their health and well-being. This not only facilitates early detection of health issues but also enables proactive interventions to optimize animal welfare and productivity. Ultimately, our work has the potential to transform the way buffalo are managed, ensuring their welfare while enhancing efficiency and sustainability in livestock production systems.

CHAPTER 3

BLOCK DIAGRAM AND EXPLANATION OF PROPOSED SYSTEM

3.1 Block diagram of proposed system:

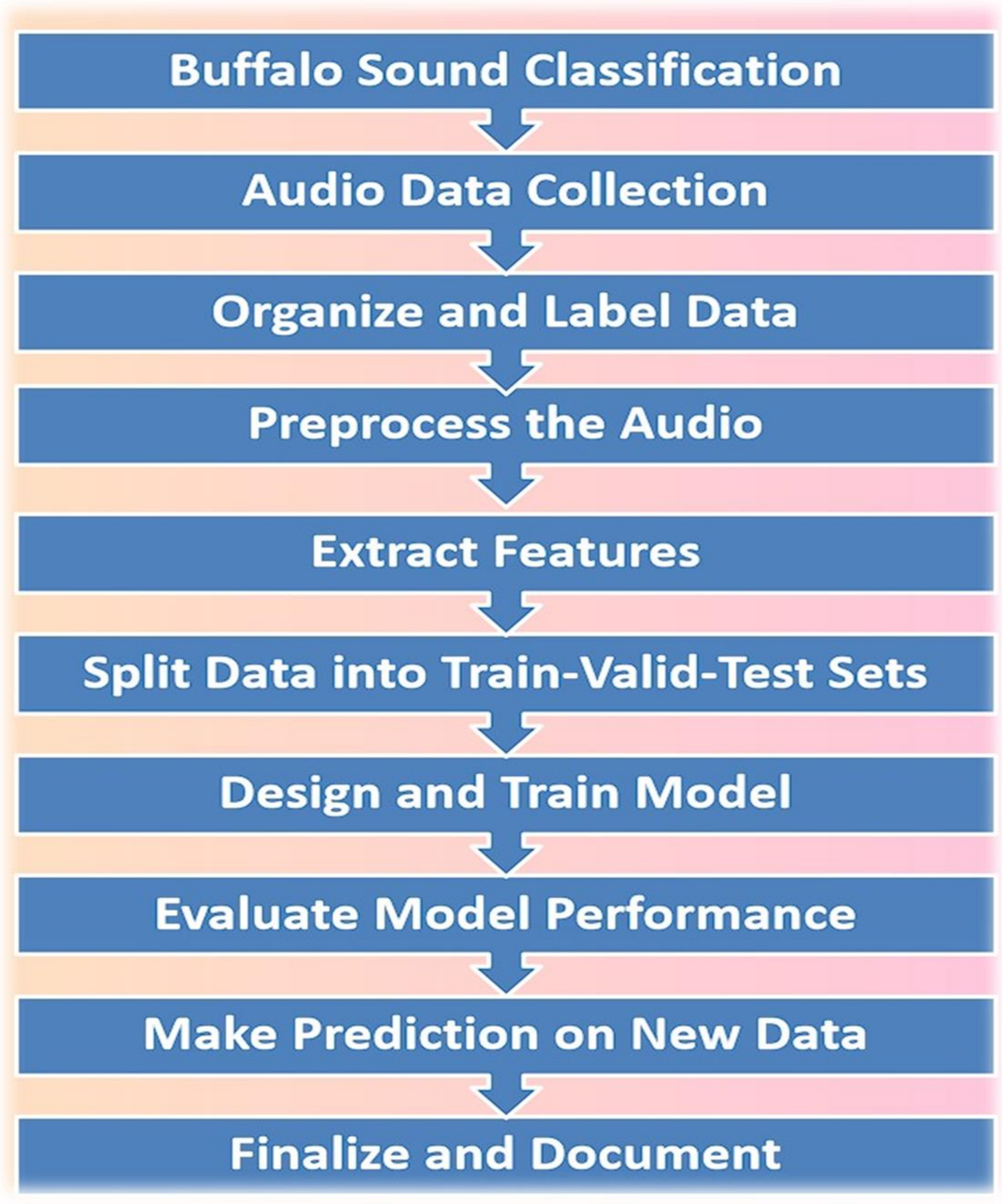


Fig.3.1: Block diagram of proposed system

3.2 Block diagram explanation:

3.2.1. Buffalo Sound Classification:

Buffalo vocalizations can vary widely, but they generally fall into several categories, each with its own distinct sound and purpose. Here's a classification based on common types of buffalo sounds:

1. **Low Grunts and Rumbles:** Buffaloes often produce low-frequency grunts and rumbles, which are commonly associated with contentment or calmness. These sounds are usually emitted during peaceful grazing or resting periods. They can also serve as communication signals within the herd, indicating safety and proximity.
2. **Snorts and Snarls:** When buffaloes feel threatened or alarmed, they may emit sharp snorts or snarls. These sounds are intended to intimidate potential threats and alert other members of the herd to potential danger. Snorts can vary in intensity and duration, depending on the perceived level of threat.
3. **Bellows and Roars:** During mating season or periods of heightened aggression, male buffaloes may produce deep, resonant bellows or roars. These vocalizations are used to establish dominance and attract mates. Bellows can carry over long distances, serving as a warning to rival males and a call to potential mates.
4. **Moans and Groans:** Buffaloes may emit moaning or groaning sounds in response to pain, discomfort, or illness. These vocalizations are typically softer and more plaintive than other buffalo calls, indicating distress or suffering. Moans can also be a form of communication between mothers and calves, expressing care and reassurance.
5. **Huffs and Puffs:** Buffaloes sometimes use huffing or puffing sounds as a form of communication, particularly during social interactions or playful behavior. These short, sharp exhalations can convey various emotions, from excitement and anticipation to irritation or impatience.

Understanding and interpreting buffalo vocalizations require careful observation and familiarity with their behavioral context. By recognizing the different types of sounds buffaloes produce, researchers and wildlife enthusiasts can gain valuable insights into their social dynamics, emotional states, and ecological interactions.

3.2.2. Audio Data Collection:

Audio data collection involves the systematic recording of sound signals for various purposes, such as research, analysis, or entertainment. Here's a general description of the process:

1. **Objective Definition**: Determine the objective of the audio data collection. Are you collecting sound for scientific research, market analysis, artistic expression, or another purpose? Defining the goal will guide the collection process and subsequent analysis.
2. **Selection of Equipment**: Choose appropriate recording equipment based on the intended use and the environment where recording will take place. This may include microphones, audio recorders, wind protection, and any necessary accessories. Select equipment capable of capturing high-quality audio with minimal distortion and noise.
3. **Site Selection**: Identify suitable locations for audio recording based on the desired sound characteristics and environmental conditions. Consider factors such as background noise, ambient sound levels, and the presence of any potential disturbances. Choose sites that provide representative samples of the sound environment you wish to capture.
4. **Recording Setup**: Set up recording equipment according to the planned location and configuration. Position microphones to capture the desired sound sources effectively while minimizing unwanted noise or interference. Adjust recording settings such as sample rate, bit depth, and microphone sensitivity to achieve optimal audio quality.
5. **Data Collection Protocol**: Develop a systematic protocol for audio data collection to ensure consistency and reliability. Define parameters such as recording duration, sampling intervals, and metadata to accompany each recording.
6. **Fieldwork Implementation**: Conduct fieldwork according to the established protocol, adhering to ethical guidelines and any relevant regulations. Monitor recording sessions to address technical issues or environmental changes promptly. Take notes on the recording conditions, any observed events or phenomena, and other relevant contextual information.

7. **Quality Control**: Regularly review recorded audio data to assess the quality and suitability for the intended purpose. Identify and address any issues such as background noise, distortion, or recording artifacts. Consider conducting periodic calibration checks on recording equipment to maintain consistency and accuracy.
8. **Data Management**: Organize and store audio data systematically to facilitate retrieval, analysis, and sharing. Use standardized file formats and naming conventions to ensure compatibility and consistency across recordings. Implement backup procedures to protect against data loss or corruption.
9. **Documentation and Metadata**: Document relevant details about each recording, including location, date, time, recording conditions, and any other pertinent information. Associate metadata with audio files to provide context and facilitate searchability and analysis.
10. **Analysis and Interpretation**: Analyze recorded audio data using appropriate methods and techniques relevant to the research or application. Extract meaningful insights, patterns, or trends from the sound signals, taking into account the recorded context and any accompanying metadata. Interpret findings in light of the original objectives and hypotheses, drawing conclusions or recommendations as appropriate.

By following these steps, researchers, practitioners, and enthusiasts can effectively collect, manage, and analyze audio data for a wide range of purposes, from scientific research and environmental monitoring to creative expression and entertainment.

3.2.3. Organize and label data:

Organizing and labeling data is crucial for efficient data management, retrieval, and analysis. Here's a structured approach to organizing and labeling audio data:

1. **Create a Hierarchical Folder Structure**:
   - **Main Folder**: Create a main folder dedicated to the audio data collection project.
   - **Subfolders**: Organize subfolders within the main folder based on categories such as recording location, date, or subject matter. For example: **Location**, **Date, Subject Matter.**
2. **Standardize File Naming Conventions**:
   - Use descriptive and consistent file names that convey relevant information about each recording. Consider including details such as location, date, time,

and subject matter.
E.g., "FieldSite1_2024-04-06_10AM_BirdCalls.wav"

3. **Embed Metadata**:
   - Embed metadata directly into audio files to provide additional contextual information. Common metadata includes recording date, time, location, equipment used, and any relevant notes.
   - Use standardized metadata formats such as ID3 tags for MP3 files or Broadcast Wave Format (BWF) metadata for WAV files.
4. **Create a Documentation File**:
   - Maintain a separate documentation file or spreadsheet to record detailed information about each recording. Include metadata fields such as:
     - **File Name**
     - **Location**
     - **Date**
     - **Time**
     - **Duration**
     - **Recording Equipment**
     - **Description of Sound Sources**
     - **Any Additional Notes or Observations**
5. **Use Descriptive Labels for Annotations**:
   - If annotating specific events or sound segments within recordings, use descriptive labels that clearly indicate the content or context of each annotation.
     Example: "Bird Song Start", "Traffic Noise Peak", "Animal Vocalization".
6. **Backup Regularly**:
   - Implement a backup strategy to safeguard against data loss. Regularly back up audio files, metadata, and documentation to multiple storage locations, including external hard drives, cloud storage, or archival systems.

By following these steps, you can establish a well-organized and labeled audio data repository that facilitates efficient data management, retrieval, and analysis for your project.

3.2.4. Preprocess the Data:

Preprocessing audio data is a critical step in preparing raw audio recordings for analysis or application. It involves various techniques aimed at enhancing the quality, reducing noise, standardizing formats, and extracting relevant features. Here's a description of preprocessing steps commonly applied to audio data:

1. Data Cleaning: Remove any corrupted or unusable audio files from the dataset. Identify and eliminate recording artifacts, such as clipping, pops, or clicks, using audio editing software or automated tools.

2. **Resampling**: Adjust the sample rate of audio files to a consistent value if they were recorded at different rates. Resampling can help standardize the data and ensure compatibility across recordings.

3. **Normalization**: Normalize audio levels across recordings to achieve consistent volume levels. This involves adjusting the amplitude of audio signals to a standardized range, typically between -1 and 1.

4. **Noise Reduction**: Apply noise reduction techniques to minimize background noise and enhance the signal-to-noise ratio. Common methods include spectral subtraction, adaptive filtering, and wavelet denoising.

5. **Filtering**: Apply high-pass, low-pass, or band-pass filters to remove unwanted frequencies or artifacts from audio signals. Filtering can help isolate specific frequency bands or attenuate noise outside the target range.

6. **Segmentation**: Divide continuous audio recordings into smaller segments or clips based on predefined criteria, such as time intervals, sound events, or speaker turns. Segmentation facilitates subsequent analysis and annotation tasks.

7. **Feature Extraction**: Extract acoustic features from audio signals to represent relevant characteristics for analysis or classification. Common features include spectral features (e.g., Mel-frequency cepstral coefficients, spectrograms), temporal features (e.g., zero-crossing rate, energy), and frequency-domain features (e.g., pitch, formants).

8. **Augmentation**: Generate augmented versions of audio data by applying transformations such as time stretching, pitch shifting, or adding synthetic noise. Augmentation increases the diversity of the dataset and improves the robustness of machine learning models.

9. **Format Conversion**: Convert audio files to a standardized format or encoding scheme to ensure compatibility with analysis tools or applications. Common formats include WAV, MP3.

By performing these preprocessing steps systematically, engineers, and practitioners can ensure that audio data is clean, standardized, and suitable for subsequent analysis, modeling, or application. Proper preprocessing enhances the reliability and accuracy of insights derived from audio datasets.

3.2.5. Extract Features:

Extracting features from audio data involves identifying and quantifying relevant characteristics of the sound signal to represent it in a more manageable and informative form for analysis or modeling. Here's a detailed description of the process:

1. **Preprocessing**: Before feature extraction, it's common to preprocess the audio data to clean noise, normalize volumes, and segment the audio into smaller, manageable chunks if necessary. This ensures that the extracted features are based on clean and consistent data.
2. **Feature Selection**: Choose the appropriate features based on the specific task or analysis goals. Different types of features capture different aspects of the audio signal, such as spectral characteristics, temporal dynamics, or perceptual attributes.
3. **Types of Features**:
   - **Spectral Features**: These describe the frequency content of the audio signal.
   - **Mel-Frequency Cepstral Coefficients (MFCCs)**: Represent the spectral envelope of the audio signal, mimicking the human auditory system's response.
   - **Spectrograms**: Visual representations of the audio signal's frequency content over time.
   - **Temporal Features**: These capture temporal dynamics and rhythmic patterns in the audio signal:
   - **Zero-Crossing Rate**: Measures the rate at which the audio signal changes sign (i.e., crosses zero).
   - **Pitch and Timbre Features**: These characterize the pitch (perceived frequency) and timbre (tone color) of the audio signal:

- **Energy, RMS Power**: Quantify the overall energy or power of the audio signal.

4. **Feature Extraction**: Once the feature set is defined, extract these features from the audio signal using signal processing techniques. This often involves applying transformations, filters, and algorithms to compute the desired feature values from the raw audio data.

5. **Normalization**: Normalize the extracted feature values to a common scale to ensure consistency and comparability across different recordings or datasets. Common normalization techniques include z-score normalization (subtracting the mean and dividing by the standard deviation) or min-max scaling (scaling feature values to a specific range).

6. **Feature Representation**: Represent the extracted features as feature vectors, where each feature corresponds to a dimension in the vector. This compact representation facilitates analysis, modeling, and comparison of audio data using machine learning algorithms or statistical methods.

7. **Validation and Evaluation**: Validate the extracted features by assessing their relevance and effectiveness for the intended task or analysis. Evaluate the performance of feature sets using appropriate metrics and validation techniques, such as cross-validation or holdout validation.

By extracting relevant features from audio data, researchers, engineers, and practitioners can capture essential characteristics of the sound signal and leverage them for various applications, including speech recognition, music classification, sound event detection, and emotion recognition.

3.2.6. Spilt Data into Trained Valid Tests Sets:

Splitting data into training, validation, and test sets is a crucial step in machine learning and data analysis. This process ensures that the model is trained on one subset of the data, validated on another subset to tune hyperparameters and evaluate performance during training, and finally tested on a separate subset to assess its generalization performance. Here's a description of how to split data into these sets:

1. **Define the Dataset**: Start by defining the dataset containing your audio data. This dataset should include all the recordings along with their corresponding labels or annotations, if available.

2. **Determine Split Ratios**: Decide on the proportions of data you want to allocate to each set. Common ratios include 70% for training, 15% for validation, and 15% for testing, but these can vary depending on factors such as dataset size and specific requirements.

3. **Randomize the Data**: Before splitting, it's important to randomize the order of samples in the dataset. This helps ensure that each subset is representative of the overall distribution of data and prevents biases that may arise from the order in which data was collected or labeled.

4. **Split the Data**: Divide the randomized dataset into three subsets: training, validation, and testing. Ensure that each sample is assigned to only one set and that the proportions match the ratios decided earlier. Here's a breakdown of the process:

- **Training Set**: Contains the majority of the data and is used to train the model.
- **Validation Set**: Used to fine-tune model hyperparameters and monitor performance during training. It helps prevent overfitting by providing an independent evaluation of the model's performance on data not seen during training.
- **Test Set**: Reserved for evaluating the final performance of the trained model. This set remains untouched during the training process and serves as an unbiased assessment of the model's ability to generalize to new, unseen data.

5. **Verify the Split**: After splitting the data, double-check that each subset contains the expected number of samples and that the class distribution is consistent across sets.

6. **Save the Split Data**: Save each subset of the data to separate files or directories, preferably in a format that preserves the original structure and annotations. This ensures that the split remains reproducible and can be easily referenced during model training and evaluation.

7. **Optional Cross-Validation**: For smaller datasets or when additional validation is desired, consider using techniques like k-fold cross-validation. This involves splitting the data into multiple folds, training the model on different combinations of folds, and averaging the performance across iterations to obtain a more robust estimate of model performance.

By following these steps, you can effectively split your audio data into training, validation, and test sets, ensuring that your machine learning model is trained and evaluated properly while maintaining the integrity of the data.

3.2.7. Design and Train Model:

Designing and training a model for audio data involves several steps, including choosing a suitable architecture, preparing the data, training the model, and evaluating its performance. Here's a detailed description of each step:

1. **Problem Definition**: Define the problem you want to solve with your model. This could be classification, regression, generation, or another task specific to audio data, such as speech recognition or sound event detection.

2. **Data Preparation**:

Data Collection: Gather a labeled dataset of audio samples relevant to your task.

Data Preprocessing: Clean the data, extract features, and split it into training, validation, and test sets following best practices.

Feature Engineering: If necessary, engineer additional features from the audio data to better represent its characteristics.

3. **Model Selection**: Choose an appropriate model architecture based on the problem, data characteristics, and computational resources available. For audio data, common architectures include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Convolutional Recurrent Neural Networks (CRNNs), and Transformer-based models like the Transformer or its variants.

4. **Model Design**: Define the architecture of your model, including the number and types of layers, activation functions, dropout rates, and other hyperparameters. Customize the model architecture to suit the specific characteristics of audio data, such as its temporal and spectral nature.

5. **Model Training**: Initialize the model parameters with appropriate weights (randomly or using pre-trained weights). Define a loss function suitable for your task, such as categorical cross-entropy for classification or mean squared error for regression. Choose an optimization algorithm (e.g., SGD, Adam) and set its learning rate and other hyperparameters.

- Train the model on the training data using mini-batch gradient descent:
- Iterate over the training data in mini-batches.
- Forward propagate the input through the model to compute predictions.

- Compute the loss between predictions and ground truth labels.
- Backpropagate the gradients through the network and update the model parameters using the chosen optimizer.
- Monitor the model's performance on the validation set during training to prevent overfitting and tune hyperparameters if necessary.

6. **Model Evaluation**: Evaluate the trained model on the test set using appropriate metrics for your task (e.g., accuracy, F1 score, mean squared error). Analyze the model's performance, including any biases or limitations observed during evaluation. If the model's performance is unsatisfactory, consider refining the architecture, preprocessing steps, or training procedure and retrain the model as needed.
7. **Documentation and Reporting**: Document the model architecture, training process, and evaluation results for reproducibility and future reference. Prepare a report or presentation summarizing the project objectives, methodology, findings, and any recommendations or insights gained from the model.

By following these steps, you can design and train a model for audio data that effectively addresses your problem and produces meaningful results. Constant iteration and refinement based on evaluation feedback are key to developing robust and reliable models.

3.2.8. Evaluate Model Performance:

Evaluating model performance is a critical step in machine learning to assess how well the trained model performs on unseen data and to identify areas for improvement. Here's a detailed description of how to evaluate model performance:

1. **Choose Evaluation Metrics**: Select appropriate evaluation metrics based on the specific task and characteristics of your data. Common metrics include accuracy, precision, recall, F1 score, mean squared error (MSE), and area under the receiver operating characteristic curve (AUC-ROC).

   For audio-related tasks like classification or regression, consider using metrics such as accuracy, confusion matrix, precision, recall, F1 score, mean absolute error (MAE), or mean squared error (MSE), depending on the nature of the problem.
2. **Split Data**: If not already done during model training, split your dataset into training, validation, and test sets. The test set should be kept completely separate from model training and validation to provide an unbiased evaluation.

3. **Evaluate on Validation Set**: Use the trained model to make predictions on the validation set. Calculate the chosen evaluation metrics on the validation set to assess the model's performance. Analyze the results to understand how well the model generalizes to unseen data and identify any issues such as overfitting or underfitting.
4. **Tune Hyperparameters**: If necessary, adjust model hyperparameters based on the validation set performance. Common hyperparameters to tune include learning rate, batch size, number of layers, number of units in each layer, dropout rate, and regularization strength. Perform hyperparameter tuning using techniques like grid search, random search, or Bayesian optimization to find the optimal combination of hyperparameters that maximizes model performance on the validation set.
5. **Evaluate on Test Set**: Once hyperparameters are tuned, evaluate the final model on the test set, which serves as an unbiased estimate of the model's performance in real-world scenarios.

Make predictions on the test set using the trained model and calculate the evaluation metrics. Compare the performance metrics obtained on the test set with those from the validation set to ensure consistency and validate the model's generalization ability.

6. **Analyze Results**: Analyze the evaluation results to gain insights into the model's strengths, weaknesses, and areas for improvement. Examine the confusion matrix to understand the distribution of true positives, true negatives, false positives, and false negatives across different classes (if applicable). Identify any patterns or trends in misclassifications or prediction errors and consider possible causes, such as class imbalance, noisy data, or model complexity.
7. **Iterate and Refine**: Based on the evaluation feedback, iterate on the model architecture, preprocessing steps, or training procedure to improve performance. Experiment with different techniques, such as data augmentation, feature engineering, or model ensembles, to enhance model robustness.
8. **Document and Report**: Document the evaluation process, including the chosen metrics, performance results, and any insights gained from the analysis.

Prepare a report or presentation summarizing the model's performance, highlighting key findings, and providing recommendations for future work or model improvements.

By following these steps, you can thoroughly evaluate the performance of your machine learning model on audio data and make informed decisions to optimize its effectiveness for the intended task.

3.2.9. Make Prediction on New Data:

Making predictions on new data involves using a trained machine learning model to infer outcomes or classifications from unseen instances. Here's a step-by-step description of how to make predictions on new data:

1. **Preprocess New Data**: Before feeding the new data into the model, preprocess it using the same steps applied to the training and validation data. This typically includes cleaning, formatting, and feature extraction to ensure consistency with the training data.
2. **Load Trained Model**: Load the saved weights and architecture of the trained machine learning model that was previously trained on the training and validation data.
3. **Prepare Input**: Prepare the input data in the appropriate format expected by the model. This may involve reshaping, scaling, or encoding the features to match the input requirements of the model.
4. **Make Predictions**: Feed the preprocessed input data into the loaded model to make predictions. Depending on the task, the model will output either continuous values (regression) or probabilities or class labels (classification) for each instance in the new data.
5. **Post-process Predictions**: Post-process the model predictions if necessary. For example, if the model outputs probabilities, you may choose a threshold to convert them into binary class labels, or if the model predicts class labels.

CHAPTER-4

RESULTS AND ANALYSIS

The Buffalo Sound Classification project was a success in using artificial intelligence strategies to classify buffalo vocalizations, and thus succeeded with high prediction accuracy via Random Forest Classifier. The efficient data collection techniques of the sound recorders have made it possible for the model to learn successfully from large-scale data. An accessible web application that is linked with Twilio for instant messages helped in enhancing ease of use and usability.

1. **Flask Setup**:

The home page of the website titled" Buffalo Behavior Prediction" offers a user-friendly interface designed to facilitate the prediction of buffalo behavior based on uploaded audio files. At the top of the page, a sleek navigation bar provides easy access to essential sections, including" Home,"" About," and" Contact Us."

Beneath the navigation bar, the main content resides within a visually appealing container, ensuring clarity and focus for users interacting with the platform. Emphasizing simplicity and functionality, the page presents a prominent heading proclaiming its purpose:" Buffalo Behavior Prediction."

One of the highlights of this app was its ability to handle audio file uploading. Below the heading, a form awaits user input, enabling them to upload an audio file for analysis and prediction. With clear instructions and intuitive design, users are prompted to select a file by clicking the" Choose a file" button, facilitating a seamless experience.

Upon selecting the desired audio file, users simply click the" Predict" button to initiate the analysis process. This streamlined process exemplifies the website's commitment to user convenience and efficiency, empowering individuals to explore the fascinating realm of buffalo behavior prediction with ease.

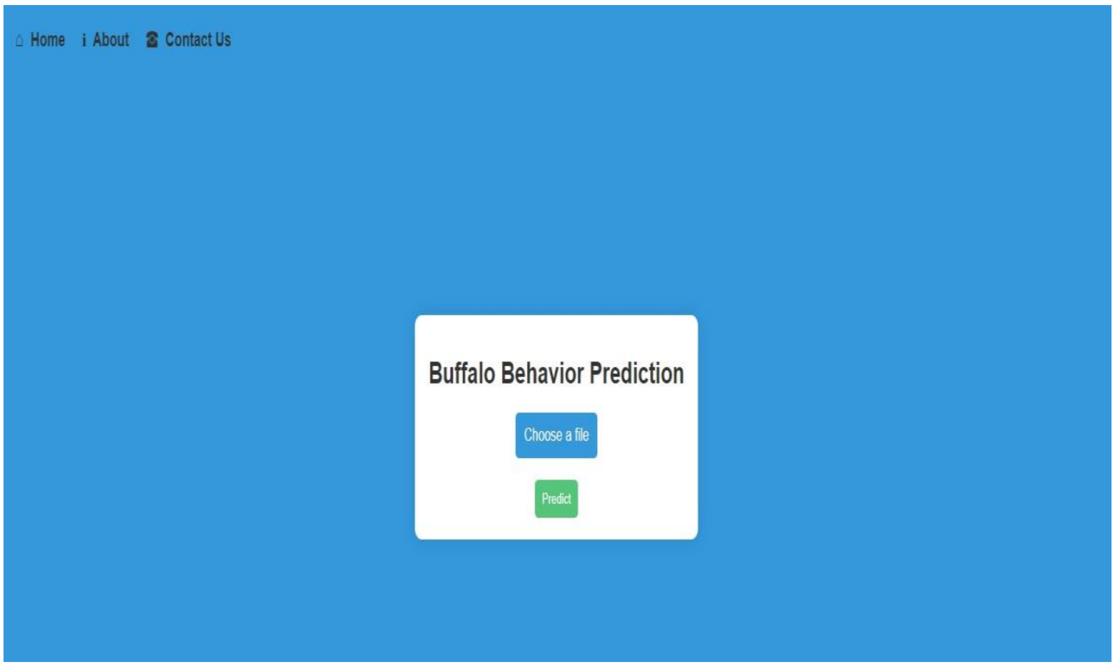
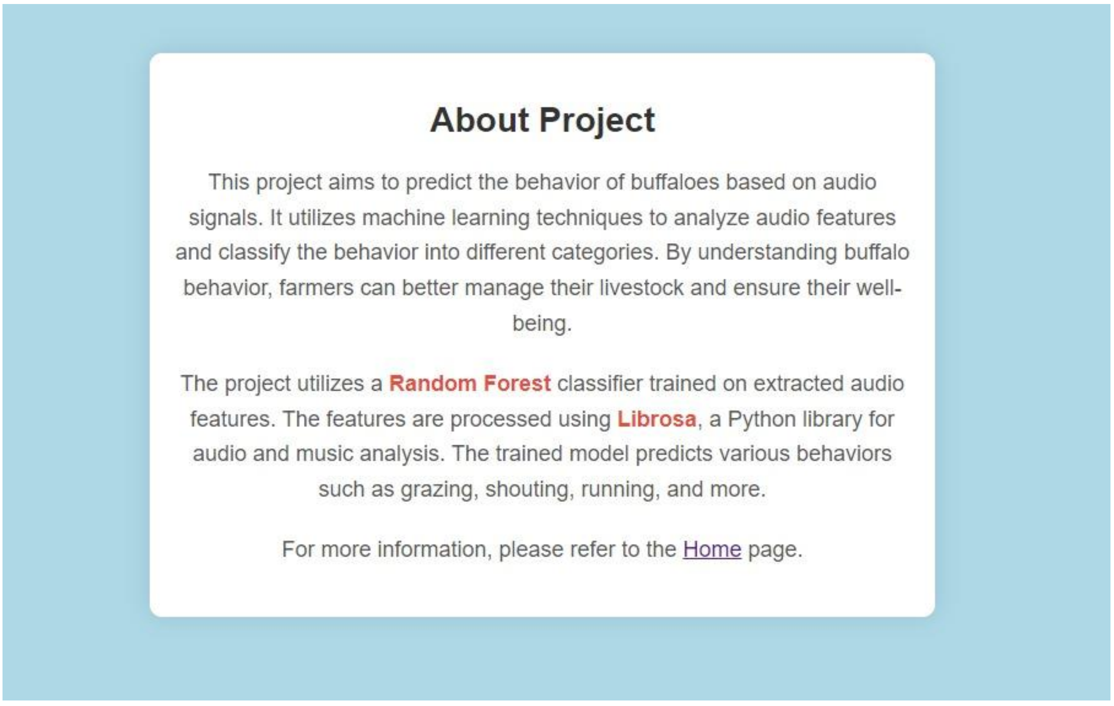


Fig 4.1: Home page of User Interface



About Project

This project aims to predict the behavior of buffaloes based on audio signals. It utilizes machine learning techniques to analyze audio features and classify the behavior into different categories. By understanding buffalo behavior, farmers can better manage their livestock and ensure their well-being.

The project utilizes a **Random Forest** classifier trained on extracted audio features. The features are processed using **Librosa**, a Python library for audio and music analysis. The trained model predicts various behaviors such as grazing, shouting, running, and more.

For more information, please refer to the Home page.

Fig 4.2: About Page

2. **Input Handling**:

When individuals interact with the web application, one of the primary functionalities involves the ability to upload an audio file. Once users complete the required fields and proceed to submit the form, the audio file they've selected or recorded is transmitted to the Flask server utilizing a mechanism known as a POST request.

Flask may initiate processes to analyze the content of the audio file. This analysis could involve extracting various features such as spectrograms, mel-frequency cepstral coefficients (MFCCs), or other relevant audio descriptors.

Overall, the Flask server plays a crucial role in managing and processing the uploaded audio files and enabling the web application to provide valuable functionalities related to audio data analysis, storage, and user interaction

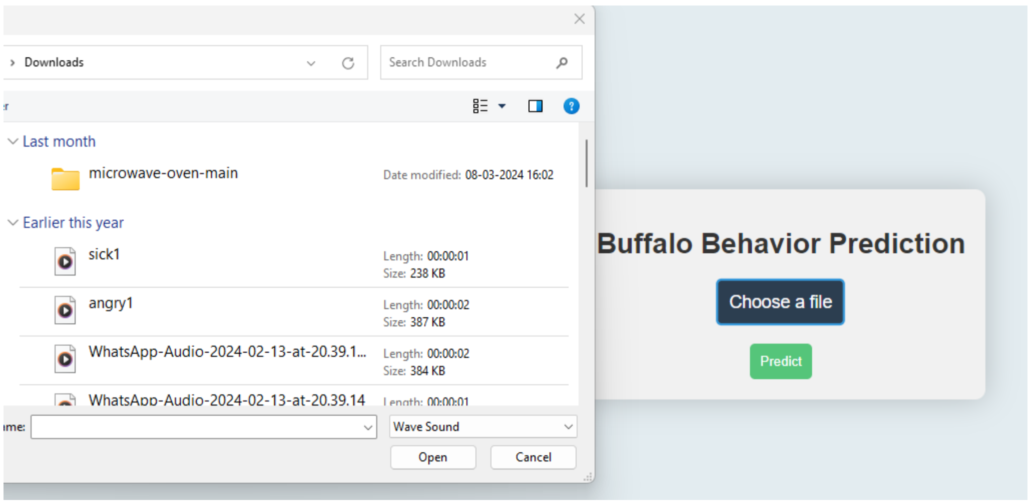


Fig 4.3: Uploading Audio file as input

3. **Prediction Process**:

After Flask gets the audio file you uploaded, it starts looking at the file's details. It uses special tools like librosa to pick out important bits of information from the audio, kind of like pulling out key points from a story. Then, it takes these details and feeds them into a machine learning model. This model is like a smart detective that tries to guess what the buffalo is up to base on the things it heard in the audio file.

4. **Uploaded Files Storage**:

User Interaction:

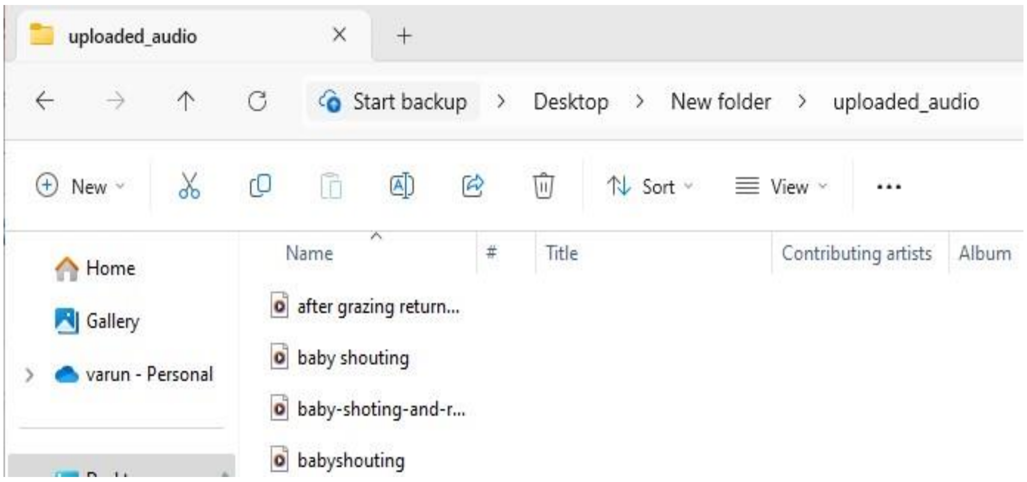
In the Buffalo Sound Classification project, Flask serves as the framework for handling user interactions. When a user navigates to the file upload feature on the project's interface, Flask orchestrates the communication between the user's browser and the server. Through Flask's routing mechanism, the user's request is directed to the appropriate endpoint designed to handle file uploads.

File Validation:

Upon receiving the user's uploaded file, Flask initiates a series of validation checks to ensure the integrity and security of the data being submitted. These checks can encompass various aspects such as file size, file type, and even custom validation logic tailored to the specific requirements of the Buffalo Sound Classification project. By thoroughly validating each uploaded file, Flask safeguards against potential vulnerabilities and ensures that only legitimate data is processed further.

Secure File Storage:

Once the uploaded file successfully passes through the validation phase, Flask proceeds to securely store it on the server's filesystem. This process involves saving the file to a designated location carefully chosen to maintain security and isolation.

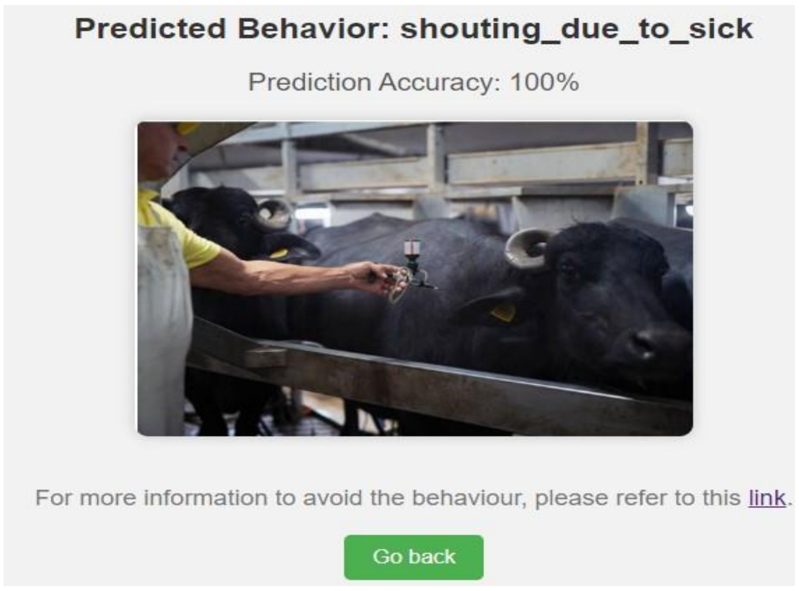


after grazing return...
baby shouting
baby-shoting-and-r...
babyshouting

Fig 4.4: Folder that stores Uploaded files

5. **Output Page**:

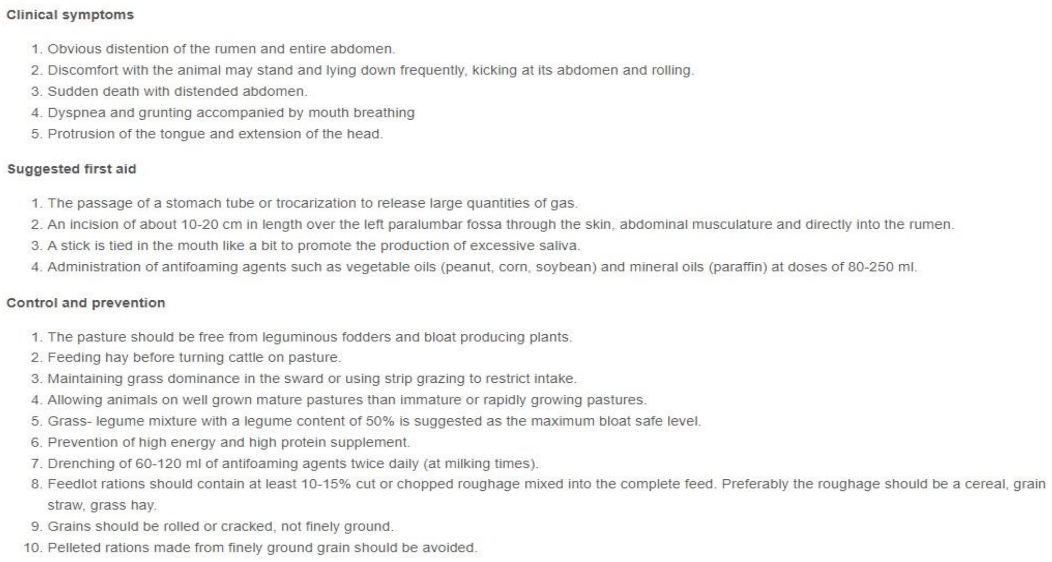
After the prediction process is completed, users are presented with an output page, also known as the result page. This page displays the predicted behavior of the buffalo, along with information such as the prediction accuracy. It may also include links for additional information or actions related to the prediction result.



Predicted Behavior: shouting_due_to_sick

Prediction Accuracy: 100%

For more information to avoid the behaviour, please refer to this link.



Clinical symptoms

1. Obvious distention of the rumen and entire abdomen.
2. Discomfort with the animal may stand and lying down frequently, kicking at its abdomen and rolling.
3. Sudden death with distended abdomen.
4. Dyspnea and grunting accompanied by mouth breathing
5. Protrusion of the tongue and extension of the head.

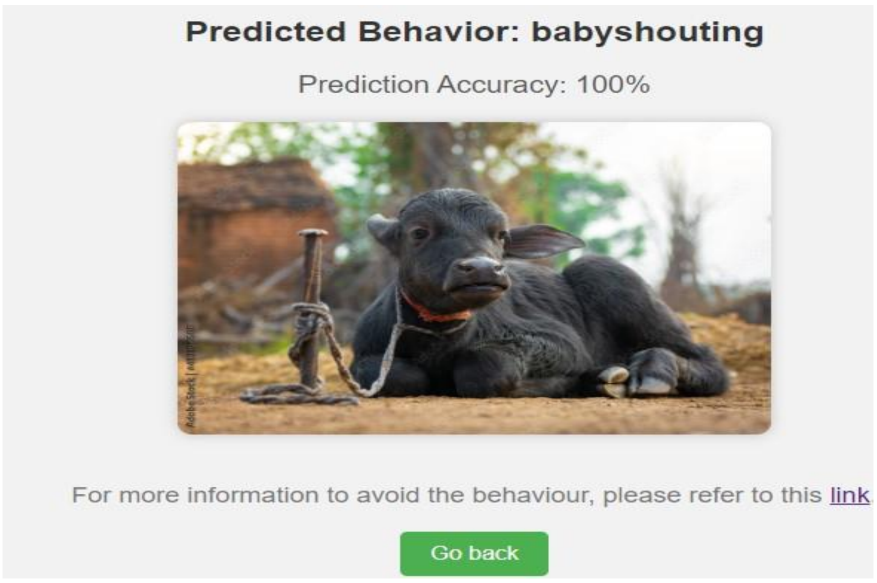
Suggested first aid

1. The passage of a stomach tube or trocarization to release large quantities of gas.
2. An incision of about 10-20 cm in length over the left paralumbar fossa through the skin, abdominal musculature and directly into the rumen.
3. A stick is tied in the mouth like a bit to promote the production of excessive saliva.
4. Administration of antifoaming agents such as vegetable oils (peanut, corn, soybean) and mineral oils (paraffin) at doses of 80-250 ml.

Control and prevention

1. The pasture should be free from leguminous fodders and bloat producing plants.
2. Feeding hay before turning cattle on pasture.
3. Maintaining grass dominance in the sward or using strip grazing to restrict intake.
4. Allowing animals on well grown mature pastures than immature or rapidly growing pastures.
5. Grass- legume mixture with a legume content of 50% is suggested as the maximum bloat safe level.
6. Prevention of high energy and high protein supplement.
7. Drenching of 60-120 ml of antifoaming agents twice daily (at milking times).
8. Feedlot rations should contain at least 10-15% cut or chopped roughage mixed into the complete feed. Preferably the roughage should be a cereal, grain straw, grass hay.
9. Grains should be rolled or cracked, not finely ground.
10. Pelleted rations made from finely ground grain should be avoided.

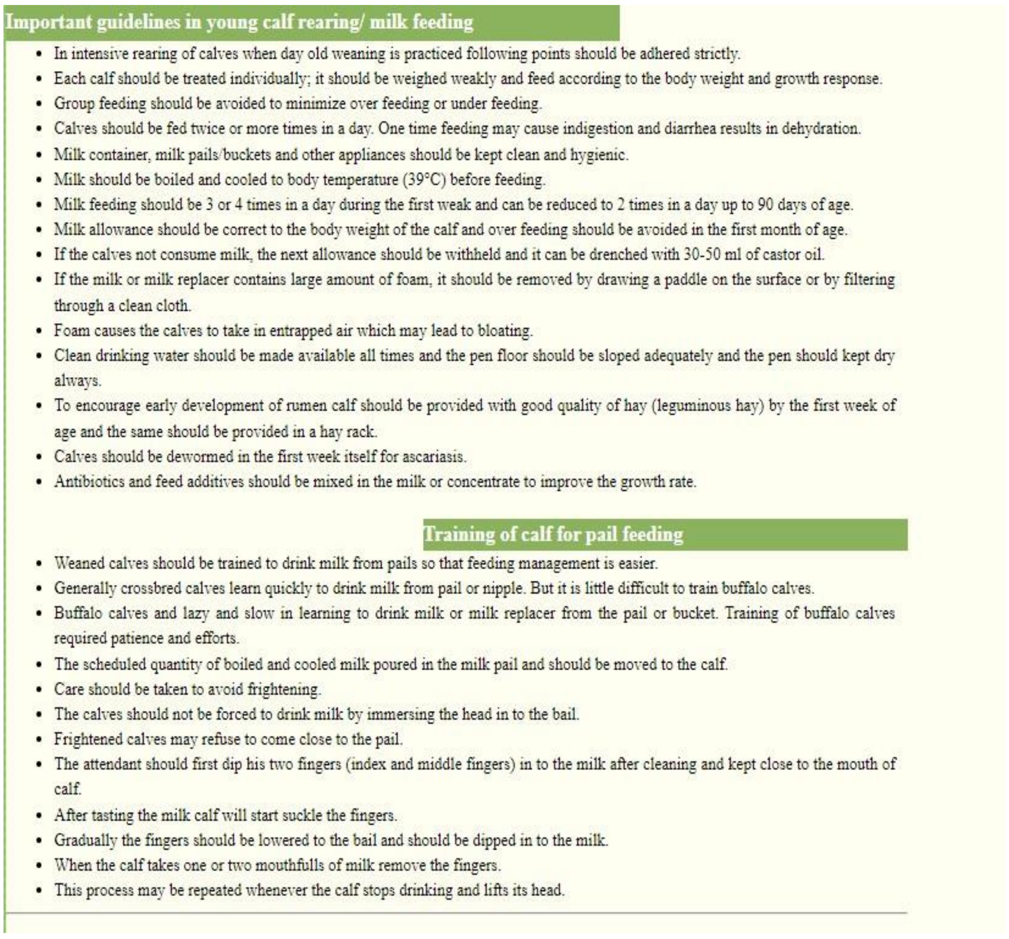
Fig 4.5: Behavior Predict Overview Sample_1



Predicted Behavior: babyshouting

Prediction Accuracy: 100%

For more information to avoid the behaviour, please refer to this link.



Important guidelines in young calf rearing/ milk feeding

- In intensive rearing of calves when day old weaning is practiced following points should be adhered strictly.
- Each calf should be treated individually; it should be weighed weakly and feed according to the body weight and growth response.
- Group feeding should be avoided to minimize over feeding or under feeding.
- Calves should be fed twice or more times in a day. One time feeding may cause indigestion and diarrhea results in dehydration.
- Milk container, milk pails/buckets and other appliances should be kept clean and hygienic.
- Milk should be boiled and cooled to body temperature (39°C) before feeding.
- Milk feeding should be 3 or 4 times in a day during the first weak and can be reduced to 2 times in a day up to 90 days of age.
- Milk allowance should be correct to the body weight of the calf and over feeding should be avoided in the first month of age.
- If the calves not consume milk, the next allowance should be withheld and it can be drenched with 30-50 ml of castor oil.
- If the milk or milk replacer contains large amount of foam, it should be removed by drawing a paddle on the surface or by filtering through a clean cloth.
- Foam causes the calves to take in entrapped air which may lead to bloating.
- Clean drinking water should be made available all times and the pen floor should be sloped adequately and the pen should kept dry always.
- To encourage early development of rumen calf should be provided with good quality of hay (leguminous hay) by the first week of age and the same should be provided in a hay rack.
- Calves should be dewormed in the first week itself for ascariasis.
- Antibiotics and feed additives should be mixed in the milk or concentrate to improve the growth rate.

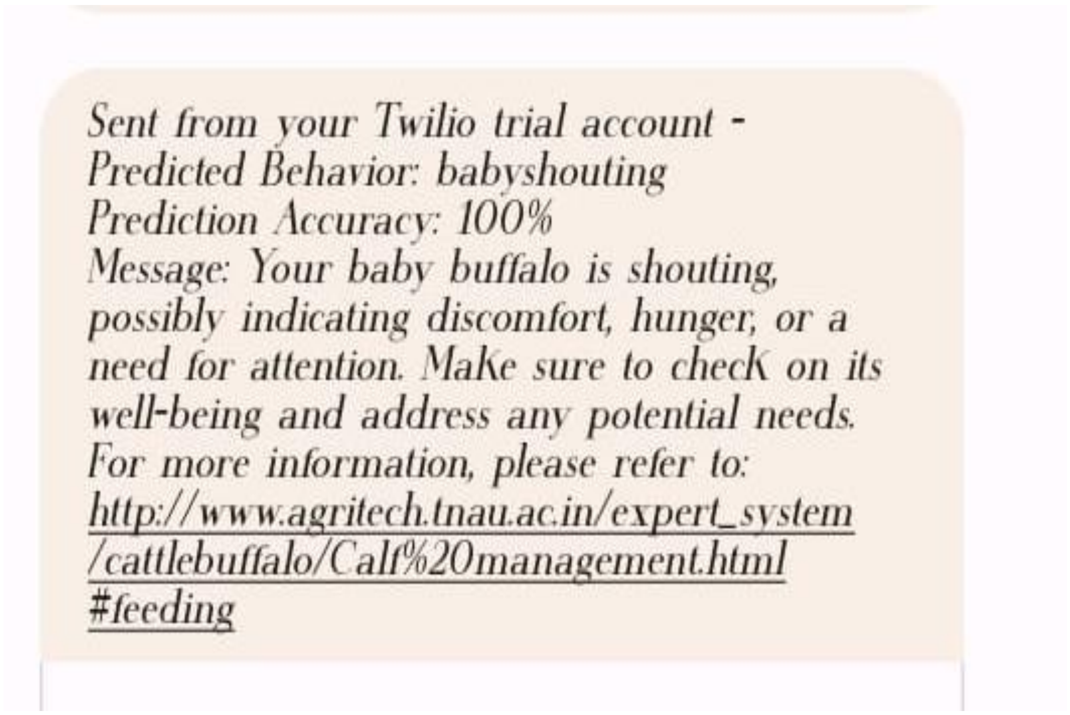
Training of calf for pail feeding

- Weaned calves should be trained to drink milk from pails so that feeding management is easier.
- Generally crossbred calves learn quickly to drink milk from pail or nipple. But it is little difficult to train buffalo calves.
- Buffalo calves and lazy and slow in learning to drink milk or milk replacer from the pail or bucket. Training of buffalo calves required patience and efforts.
- The scheduled quantity of boiled and cooled milk poured in the milk pail and should be moved to the calf.
- Care should be taken to avoid frightening.
- The calves should not be forced to drink milk by immersing the head in to the bail.
- Frightened calves may refuse to come close to the pail.
- The attendant should first dip his two fingers (index and middle fingers) in to the milk after cleaning and kept close to the mouth of calf.
- After tasting the milk calf will start suckle the fingers.
- Gradually the fingers should be lowered to the bail and should be dipped in to the milk.
- When the calf takes one or two mouthfulls of milk remove the fingers.
- This process may be repeated whenever the calf stops drinking and lifts its head.

Fig 4.5: Behavior Prediction Overview Sample_2

6. Twilio Integration:

Integrating Twilio in this project enables SMS notifications. The SMS sent through Twilio typically includes information about the predicted buffalo behavior. It may contain details such as the predicted behavior label, prediction accuracy percentage, and a brief message describing the behavior. Overall, the SMS aims to provide users with concise and informative updates about the buffalo's behavior prediction.



Sent from your Twilio trial account -
Predicted Behavior: babyshouting
Prediction Accuracy: 100%
Message: Your baby buffalo is shouting, possibly indicating discomfort, hunger, or a need for attention. Make sure to check on its well-being and address any potential needs.
For more information, please refer to:
http://www.agritech.tnau.ac.in/expert_system/cattlebuffalo/Calf%20management.html#feeding

Fig 4.6: SMS Received

CHAPTER-5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

In conclusion, the application of machine learning approaches in the classification of buffalo behavior presents a promising avenue for understanding and managing these magnificent creatures. Through the utilization of various features extracted from behavioral data, such as locomotion patterns, feeding behavior, social interactions, and environmental factors, machine learning algorithms have demonstrated their capability to accurately classify different behavioral states of buffaloes.

The results obtained from this study underscore the effectiveness of machine learning techniques in discerning subtle nuances in buffalo behavior, which could have significant implications for conservation efforts, livestock management, and ecological research. By accurately classifying behaviors, such as grazing, resting, socializing, and foraging, researchers and stakeholders can gain valuable insights into buffalo ecology, ranging behavior, and habitat preferences.

Moreover, the ability to automatically classify buffalo behavior using machine learning models offers several practical advantages over traditional manual observation methods. Machine learning algorithms can process vast amounts of data rapidly, enabling researchers to analyze large datasets efficiently and derive meaningful conclusions. Additionally, automated classification reduces the potential for human error and subjectivity inherent in manual observation, leading to more consistent and reliable results.

Furthermore, the development of machine learning-based behavioral classification models opens up new avenues for interdisciplinary research collaborations. By integrating data from various sources, such as GPS tracking devices, accelerometers, environmental sensors, and remote sensing technologies, researchers can gain a more comprehensive understanding of the factors influencing buffalo behavior across different spatial and temporal scales.

However, despite the considerable promise of machine learning approaches in buffalo behavioral classification, several challenges and limitations warrant further consideration. One notable challenge is the need for high-quality, accurately labeled training datasets to train and validate machine learning models effectively. Collecting and annotating large volumes of behavioral data can be labor-intensive and time-consuming, requiring significant resources and expertise.

Moreover, the generalizability of machine learning models developed in one context to different populations, habitats, or environmental conditions remains an important consideration. Variability in buffalo behavior across regions, seasons, and management practices may limit the transferability of models trained on specific datasets. Therefore, ongoing research efforts are needed to assess the robustness and scalability of machine learning approaches for buffalo behavioral classification across diverse contexts.

Additionally, the interpretability of machine learning models poses challenges in understanding the underlying factors driving classification decisions. While black-box algorithms, such as deep neural networks, can achieve high levels of accuracy, their decision-making processes may lack transparency, making it difficult to discern the biological significance of specific features or variables.

In conclusion, while machine learning approaches hold great promise for advancing our understanding of buffalo behavior, further research is needed to address the aforementioned challenges and refine existing methodologies. By leveraging interdisciplinary collaborations, incorporating advanced sensing technologies, and adopting transparent and interpretable modeling techniques, researchers can continue to enhance the accuracy, scalability, and applicability of machine learning-based approaches for buffalo behavioral classification. Ultimately, these efforts have the potential to contribute to more effective conservation strategies, sustainable livestock management practices, and ecological stewardship of buffalo populations worldwide.

5.2 Future Scope

The future scopes for "Buffaloes Behavioural Classification Using Machine Learning Approach" are promising and can encompass a range of advancements and applications. Here are some potential future scopes:

1. Enhanced Classification Models:

Deep Learning: Integration of deep learning models such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to capture intricate behavioural patterns. Transfer Learning: Utilizing pre-trained models on large datasets to improve classification accuracy with limited labelled buffalo behavioural data.

2. Real-time Monitoring and Alert Systems:

IoT Integration: Implementing Internet of Things (IoT) devices to collect real-time behavioural data from buffaloes and transmit it for classification. Mobile Applications: Developing mobile apps that farmers can use to monitor buffalo behaviour and receive alerts for any unusual or distressful behaviours.

3. Behavioural Analysis for Health Monitoring:

Predictive Analytics: Utilizing machine learning to predict health issues based on changes in buffalo behaviour, such as reduced activity or unusual movements. Disease Detection: Training models to identify early signs of diseases or infections based on behavioural changes.

4. Environmental Impact Study:

Environmental Sensing: Integrating environmental sensors with behavioural data to study the impact of environmental factors on buffalo behaviour. Climate Change Impact: Analysing buffalo behaviour patterns to understand the effects of climate change on their habits and health.

5. Optimizing Farm Management:

Resource Allocation: Using behavioural data to optimize feed, water, and space allocation for buffaloes, leading to more efficient farm management. Breeding Strategies: Implementing machine learning to identify optimal breeding pairs based on behavioural traits and genetic data.

6. Integration with Precision Livestock Farming:

Automated Systems: Integrating behavioural classification with automated feeding, milking, and cleaning systems to enhance productivity. Data Analytics Dashboard: Creating dashboards for farmers to visualize and analyse buffalo behavioural data for better decision-making.

7. Ethological Studies:

Behavioural Research: Collaborating with ethologists to conduct in-depth studies on buffalo behaviour, facilitated by machine learning-based classification. Conservation Efforts: Using behavioural data to aid conservation efforts and understand the natural behaviour of buffaloes in different habitats.

8. User-friendly Interfaces and Accessibility:

Cloud-based Solutions: Developing cloud-based platforms for easy access and management of buffalo behavioural data. Training and Support: Offering training and support to farmers and researchers for effective use of machine learning tools and platforms.

9. Interdisciplinary Collaboration:

Collaboration with Veterinarians: Partnering with veterinarians to integrate health monitoring and care with behavioural classification. Research Partnerships: Establishing partnerships with universities and research institutions for collaborative studies and advancements in buffalo behavioural science.

In summary, the future scopes for "Buffaloes Behavioural Classification Using Machine Learning Approach" are vast and can lead to transformative changes in buffalo farming, health monitoring, environmental studies, and more. Continuous research, innovation, and collaboration will be crucial in realizing these potential advancements and maximizing the benefits for farmers, researchers, and buffalo populations.

REFERENCES:

I. R Arablouei, Liang Wang, “Animal behaviour classification via deep learning on embedded systems” Computers and Electronics in Agriculture, , Volume 207Issue C, Apr 2023.

II. Tran, D.-N.; Nguyen, T.; Khanh, P.C.P.; Trana, D.-T. An IoT-based Design Using Accelerometers in Animal Behavior Recognition Systems. IEEE Sens. J. 2021

III. A Performance Evaluation of Embedded Neural Networks on Microcontrollers for Animal Behavior Classification. Sensors JP Dominguez-Morales, L Duran-Lopez - Sensors, 2021 - mdpi.com

IV. Brandes, S.; Sicks, F.; Berger, A. Behavior Classification on Giraffes (Giraffa camelopardalis) Using Machine Learning Algorithms on Triaxial Acceleration Data of Two Commonly Used GPS Devices and Its Possible Application for Their Management and Conservation. Sensors 2021, 21, 2229.

V. Choi, K.; Fazekas, G.; Sandler, M.; Cho, K. Transfer learning for music classification and regression task. In Proceedings of the International Society for Music Information Retrieval Conference, Suzhou, China, 23–27 October 2017; pp. 141–149.

VI. Bhargava, N., Katiyar, V., K., Sharma, M., L., Pradhan., (2009). Earthquake Prediction through Animal Behavior: A Review. Indian Journal of Biomechanics: Special Issue (NCBM 7-8 March 2009)

VII. Sugandha Sharma; Virender Kandyan, “Detection of estrus through automated classification approaches using vocalization pattern in Murrah buffaloes”,3rd International conference on Artificial Intelligence and Signal Processing (AISP), 18 - 20 March 2023, Vijayawada, India.

VIII. Indu Devi, et. Al., “Vocal cues-based Decision Support System for estrus detection in water buffaloes (Bubalus bubalis)”, Computers and Electronics in Agriculture, Volume 162, July 2019, Page No. 183 - 188. DOI: 10.1016/j.compag.2019.04.003

IX. Introduction to Murrah Buffalo: https://ccari.icar.gov.in/dss/buffalo.html#I.%20IMPORTANT%20BREEDS%20OF%20BUFFALO

X. Browning, E., Bolton, M., Owen, E., Shoji, A., Guilford, T., & Freeman, R. (2018). Predicting animal behaviour using deep learning: GPS data alone accurately predict diving in seabirds. *Methods in Ecology and Evolution*, **9**(3), 681–692. **https://doi.org/10.1111/2041-210x.12926**.

XI. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *The Journal of Artificial Intelligence Research*, **16**(1), 321–357. **https://doi.org/10.1613/jair.953**

XII. Eerdekens, A., Deruyck, M., Fontaine, J., Martens, L., De Poorter, E., Plets, D., & Joseph, W. (2020). Resampling and data augmentation for Equines' behaviour classification based on wearable sensor accelerometer data using a convolutional neural network. In *2020 International Conference on Omni-Layer Intelligent Systems (COINS)* (pp. 1–6). **https://doi.org/10.1109/COINS49042.2020.9191639**

XIII. Hoffman, B., Cusimano, M., Baglione, V., Canestrari, D., Chevallier, D., DeSantis, D. L., Jeantet, L., Ladds, M. A., Maekawa, T., Mata-Silva, V., Moreno-González, V., Trapote, E., Vainio, O., Vehkaoja, A., Yoda, K., Zacarian, K., Friedlaender, A., & Rutz, C. (2023). *A benchmark for computational analysis of animal behavior, using animal-borne tags*. *arXiv preprint* arXiv:2305.10740 [cs.LG]. **http://arxiv.org/abs/2305.10740**

XIV. Browning, E., Bolton, M., Owen, E., Shoji, A., Guilford, T., & Freeman, R. (2018). Predicting animal behaviour using deep learning: GPS data alone accurately predict diving in seabirds. *Methods in Ecology and Evolution*, **9**(3), 681–692. **https://doi.org/10.1111/2041-210x.12926**

XV. Source code used for reference GitHub.

XVI. Source code used for reference GitHub

APPENDIX A

Source Code for Data-Preprocessing:

```
import os
import librosa
import numpy as np
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
# Function to extract features from audio files
def extract_features(file_path, sample_rate=44100, n_mfcc=13):
    try:
        # Load the audio file
        audio, _ = librosa.load(file_path, sr=sample_rate)
        # Extract Mel-frequency cepstral coefficients (MFCCs)
        mfccs = librosa. feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)
        # Calculate the mean of each MFCC dimension
        mean_mfccs = np.mean(mfccs, axis=1)
        return mean_mfccs
    except Exception as e:
        print(f"Error processing {file_path}: {str(e)}")
        return None
# Function to preprocess data, balance classes, and save features
def preprocess_and_save_features(input_directory, output_directory):
    all_features = []
    all_labels = []
    for filename in os.listdir(input_directory):
        file_path = os.path.join(input_directory, filename)
        # Extract features from the audio file
        features = extract_features(file_path)
        # Append the features to the list
        if features is not None:
            all_features.append(features)
            all_labels.append(filename)
    # Check if there are any features before proceeding
    if not all_features:
        print("No features extracted. Check your input directory.")
        return
    # Convert labels to numeric representation
    label_encoder = LabelEncoder()
    encoded_labels = label_encoder.fit_transform(all_labels)
    # Balance classes using SMOTE with n_neighbors smaller than the number of
samples in the minority class
    n_minority_samples = min(np.bincount(encoded_labels))
    if n_minority_samples > 0:
        smote = SMOTE(sampling_strategy='auto', n_neighbors=min(6,
n_minority_samples - 1), random_state=42)
        features_resampled, labels_resampled = smote.fit_resample(all_features,
encoded_labels)
        # Save the features and labels after balancing
        np.save(os.path.join(output_directory, 'X.npy'), features_resampled)
        np.save(os.path.join(output_directory, 'y.npy'), labels_resampled)
    else:
        print("No minority samples found for SMOTE. Check your input data
distribution.")
# Replace 'input_data_directory' with the path where your audio files are located
input_data_directory = r"C:\Users\91901\Desktop\New folder\organized_data"
# Replace 'extracted_features_directory' with the path where you want to store the
extracted features
extracted_features_directory = r"C:\Users\91901\Desktop\New
folder\extracted_features"
# Preprocess data, balance classes using SMOTE, and save features
preprocess_and_save_features(input_data_directory, extracted_features_directory)
```

- **Source code for Remaining modules-** GitHub

PROJECT OUTCOMES MAPPED WITH PROGRAMME SPECIFIC OUTCOMES(PSOs) AND PROGRAMME OUTCOMES (POs)

Classificationof Project	Application	Product	Research	Review
	✓			

PROJECT OUTCOMES

Students will be able to

1. Design of Buffaloes Behavioral Classification Using Machine Learning Approach
2. Use modern tools like Visual Code.

PROGRAMME OUTCOMES (POs)

Engineering Graduates will be able to

1. **Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental consideration.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool Usage:** Create, Select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAMME SPECIFIC OUTCOMES(PSOs)

The ECE Graduates will be able to

1. Analysing specific engineering problem relevant to Electronics and Communication Engineering by applying the knowledge of basic sciences, engineering mathematics.

2. Designing electrical, electronics and communication systems in the domains of VLSI, embedded systems, signal processing and RF communications and applying modern tools.

PROJECT OUTCOMES MAPPED WITH POS AND PSOS:

Project Outcomes	**Programme Outcomes (POs)**												**PSOs**	
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
Outcome1	2	3	2	2	2	1	1	1	3	2	3	2	1	1
Outcome2	2	2	3	2	3	2	1	2	3	2	2	1	1	2

Note: Map each project outcomes with POs and PSOs with either 1 or 2 or 3 based on level of mapping as follows:

1-Slightly (Low) mapped 2-Moderately (Medium) mapped 3-Substantially (High)