Optimizing Fetal Health Prediction with Machine Learning Models Using Cardiotography Data

A Project Report Submitted to
Jawaharlal Nehru Technological University Anantapur, Ananthapuramu

in partial fulfillment of the requirements for the
Award of the degree of

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY

Submitted by

Batch No: CSD 25-07

Palyam Vyshnavi	**21121A3832**
Unnam Charannath Chowdary	**21121A3842**
Komalakalva Sushma	**21121A3819**
Mathangi Jaswanth	**21121A3826**

Under the Supervision of
Mr. B.V. Sivaiah
Assistant Professor
Department of DS

Department of Computer Science and Systems Engineering
SREE VIDYANIKETHAN ENGINEERING COLLEGE (AUTONOMOUS)
(Affiliated to JNTUA, Ananthapuramu, Approved by AICTE,
Accredited by NBA & NAAC)
Sree Sainath Nagar, Tirupati – 517 102, A.P., INDIA

2023-2024

SREE VIDYANIKETHAN ENGINEERING COLLEGE (AUTONOMOUS)

(Affiliated to JNTUA, Ananthapuramu, Approved by AICTE,
Accredited by NBA & NAAC)
Sree Sainath Nagar, Tirupati – 517 102, A.P., INDIA
Department of Computer Science and Systems Engineering

CERTIFICATE

This is to certify that the project report entitled

"Optimizing Fetal Health Prediction with Machine Learning Models Using Cardiotocography Data"
is the Bonafide work done by

Palyam Vyshnavi	**21121A3832**
Unnam Charannath Chowdary	**21121A3842**
Komalakalva Sushma	**21121A3819**
Mathangi Jaswanth	**21121A3826**

in the Department of **Information Technology, Sree Vidyanikethan Engineering College (Autonomous), Sree Sainath Nagar, Tirupati,** and is submitted to **Jawaharlal Nehru Technological University Anantapur, Ananthapuramu** for partial fulfillment of the requirements of the award of B.Tech degree in Computer Science and Systems Engineering during the academic year 2023-2024.

Supervisor:

Mr. B.V. Sivaiah,
Assistant Professor
Dept. of Data Science
Sree Vidyanikethan Engineering College
Sree Sainath Nagar, Tirupati – 517 102

Head of the Dept.:

Dr. P.K Gupta,
Professor & Head
Dept. of Computer Science and Systems Engineering
Sree Vidyanikethan Engineering College
Sree Sainath Nagar, Tirupati – 517 102

INTERNAL EXAMINER **EXTERNAL EXAMINE**

DECLARATION

We hereby declare that this project report titled **"*Optimizing Fetal Health Prediction with Machine Learning Models Using Cardiotocography Data*"** is a genuine work carried out by us, in the **B.Tech *(Information Technology)*** degree course of **Jawaharlal Nehru Technological University Anantapur, Ananthapuramu** and has not been submitted to any other course or University for the award of any degree by us. We declare that this writtensubmission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properlycited or from whom proper permission has not been taken when needed.

Signature of the students

1.
2.
3.
4.

ACKNOWLEDGEMENTS

We are extremely thankful to our beloved Chairman and founder **Dr. M. Mohan Babu** who took keen interest to provide us the infrastructural facilities for carrying out the project work.

I am extremely thankful to our beloved Chief executive officer **Sri Vishnu Manchu** of Sree Vidyanikethan Educational Institutions who took keen interest in providing better academic facilities in the institution.

We are highly indebted to **Dr. Y. Dileep Kumar**, Principal of Sree Vidyanikethan Engineering College for his valuable support and guidance in all academic matters.

We are very much obliged to **Dr. P.K. Gupta**, Professor & Head, Department of CSSE, for providing us the guidance and encouragement in completion of this project.

We would like to express our indebtedness to the project coordinator, **Mr. P. Yogendra Prasad**, Assistant Professor, Department of CSSE for his valuable guidance during the courseof project work.

We would like to express our deep sense of gratitude to **Mr. P. Yogendra Prasad**, Assistant Professor, Department of CSSE, for the constant support and invaluable guidance provided for the successful completion of the project.

We are also thankful to all the faculty members of the IT Department, who have cooperatedin carrying out our project. We would like to thank our parents and friends who have extended their help and encouragement either directly or indirectly in completion of our project work.

Project Associates

ABSTRACT

Pregnancy complications pose significant risks to both maternal and fetal health, making early detection crucial for timely medical intervention. Cardiotocography (CTG) is a widely used method for monitoring fetal well-being, but manual interpretation of CTG data is prone to subjectivity and human error. This project aims to leverage machine learning (ML) models to automate and enhance the accuracy of fetal health classification based on CTG data.The dataset used in this study consists of various fetal heart rate (FHR) and uterine contraction parameters, classified into three categories: **normal, suspicious, and pathological**. The data underwent preprocessing, including normalization and feature selection, to improve model performance. Several ML algorithms were implemented and compared, including **Random Forest, Decision Tree, Support Vector Machine (SVM), Logistic Regression, K-Nearest Neighbors (KNN), and Gradient Boosting Classifier**. Among these, the **Random Forest model achieved the highest accuracy of 93%**, outperforming other classifiers.The performance of the models was evaluated using standard metrics such as **accuracy, precision, recall, and F1-score**. The results demonstrate that machine learning can significantly improve fetal health classification, reducing reliance on manual assessment and minimizing diagnostic errors. This research highlights the potential of AI-driven approaches to assist obstetricians in making informed decisions, ultimately improving prenatal care and reducing fetal health risks.Future work includes integrating deep learning techniques, expanding the dataset for better generalization, and deploying the model in real-time clinical settings. The successful implementation of ML-based fetal health classification can enhance medical resource allocation, optimize diagnosis efficiency, and contribute to improved maternal and fetal outcomes.

Keywords: Cardiotocography (CTG), Machine Learning, Fetal Health Classification, Pregnancy Complications, Artificial Intelligence, Medical Diagnostics, Fetal Heart Rate (FHR), Predictive Analytics..

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Introduction of Project

Fetal health monitoring is a crucial aspect of prenatal care, ensuring that potential risks to both the mother and fetus are identified early. One of the widely used techniques for fetal monitoring is **Cardiotocography (CTG)**, which records fetal heart rate (FHR) and uterine contractions. However, the manual interpretation of CTG data is **prone to subjectivity**, leading to inconsistent diagnoses and potential delays in medical interventions. To address these challenges, this project explores the application of **Machine Learning (ML) techniques** to classify fetal health status into three categories: **Normal, Suspicious, and Pathological**.

The project leverages a publicly available **CTG dataset**, applying various machine learning models to analyze and classify fetal health conditions accurately. By using ML algorithms such as **Random Forest, Decision Trees, Support Vector Machines (SVM), and Logistic Regression**, we aim to enhance diagnostic accuracy and reduce reliance on subjective human interpretation. The ultimate goal is to develop a robust, **automated system** that can assist medical professionals in making informed decisions, leading to **better pregnancy outcomes**.

This documentation provides a detailed overview of the project, including its background, motivation, objectives, and significance in the medical field. It also outlines the methodology adopted, the results obtained, and potential future improvements.

1.2 Background

Maternal and fetal health complications are significant contributors to neonatal mortality and adverse pregnancy outcomes worldwide. According to the **World Health Organization (WHO)**, approximately **303,000 women died in 2015** due to complications related to pregnancy and childbirth. Many of these deaths and fetal complications could have been prevented with timely medical intervention. One of the primary tools used in **prenatal diagnosis** is **Cardiotocography (CTG)**, which assesses **fetal heart rate (FHR) variability, accelerations, decelerations, and uterine contractions** to detect abnormalities.

Traditional CTG interpretation relies on **expert analysis by obstetricians**, which is often **time-consuming, prone to errors, and lacks consistency** among different healthcare providers. Studies have shown that automated analysis of CTG data using **artificial intelligence (AI) and machine learning** can significantly improve the accuracy of fetal health classification.With advancements in **machine learning and medical informatics**, predictive models can now analyze large volumes of CTG data, identify patterns, and classify fetal health conditions **more accurately and efficiently**. This project builds upon existing research by developing an **ML-based model** to enhance fetal health classification, ensuring **early detection of fetal distress and better pregnancy outcomes**.

1.3 Motivation

The primary motivation behind this project is the **critical need for accurate and efficient fetal health monitoring**. Pregnancy complications such as **fetal distress, intrauterine growth restriction (IUGR), and preterm birth** can lead to severe health consequences, including **cerebral palsy, developmental disorders, and even fetal mortality**. Early detection of such conditions is **vital for timely medical intervention**, which can significantly improve neonatal survival rates and maternal health.

Current **manual CTG analysis** presents several limitations, including **inter-observer variability**, lack of standardization, and **high dependence on clinical expertise**. In many **developing regions**, access to experienced obstetricians is limited, leading to **misinterpretations and delayed diagnoses**. The **integration of machine learning algorithms** into fetal health assessment can help **bridge this gap**, providing a **more consistent, objective, and automated** approach to fetal monitoring.

Furthermore, with the increasing availability of **electronic medical records (EMR) and real-time monitoring systems**, incorporating AI-driven **decision support systems** into hospitals and maternity clinics can enhance **patient care efficiency**. This project aims to develop a **reliable, non-invasive, and cost-effective** ML-based fetal health classification system, ensuring **better health outcomes for mothers and infants**.

1.4 Need for the Project

The need for this project arises from the **challenges in current fetal health assessment methods** and the **potential of AI-driven solutions** to improve diagnostic accuracy. Despite the widespread use of **CTG monitoring**, studies indicate that **manual interpretation errors contribute to significant fetal and maternal health risks**. Common challenges in current systems include:

- **Subjectivity** – Different obstetricians may interpret the same CTG results differently.
- **Time Constraints** – Manual analysis of CTG data is time-consuming.
- **Human Errors** – Fatigue, lack of experience, or cognitive overload can lead to misdiagnosis.
- **Limited Availability of Experts** – Many healthcare facilities, especially in remote areas, lack specialists for real-time fetal monitoring.

By employing **machine learning techniques**, this project provides an **automated classification system** that minimizes **human errors, enhances decision-making, and enables early detection of fetal distress**. The proposed system will not only aid healthcare professionals but also be beneficial in **resource-limited settings**, where access to specialists is **scarce**.

The significance of this project extends beyond improving diagnosis; it also contributes to **advancements in AI-driven healthcare solutions**, promoting **data-driven decision-making** in obstetrics. By integrating **ML models with real-time CTG data analysis**, this project paves the way for **smart healthcare applications** that can **revolutionize maternal and fetal healthcare**.

1.5 Objectives

The primary objectives of this project are:

1. **To develop a machine learning-based model** for fetal health classification using **Cardiotocography (CTG) data**.
2. **To compare different ML algorithms**, including **Random Forest, Decision Tree, SVM, Logistic Regression, and KNN**, in terms of classification accuracy.
3. **To improve diagnostic efficiency** by reducing **human error and variability** in fetal health classification.
4. **To provide a non-invasive, automated system** that assists obstetricians in making timely decisions regarding fetal health.
5. **To evaluate the model's performance** using key metrics such as **accuracy, precision, recall, and F1-score**.

6. **To explore future enhancements**, including deep learning approaches and real-time deployment in clinical settings.By achieving these objectives, the project aims to **enhance prenatal care, minimize fetal health risks, and contribute to AI-driven medical research**.

1.6 Organization of Thesis

This thesis is organized into six chapters. Chapter 1 provides an introduction to the project, including the background, motivation, need for the project, objectives, andorganization of the thesis. Chapter 2 presents a literature survey of relevant research. Chapter 3 discusses the methodology used, including the algorithms utilized (Support vector machine algorithm), the prediction model, data preparation, data pre- processing and design model, performance analysis, feature selection. Chapter 4 presents the system design and implementation, including the machine learning model, web application, UML diagrams, and software and hardware requirements. Chapter 5 discusses the results and includes a comparison of predictive models, experimental results, and output. Finally, Chapter 6 provides a conclusion and references used throughout the thesis

2 .LITERATURE SURVEY

[1] G. Sedgh, S. Singh, and R. Hussain, "Intended and unintended pregnancies worldwide in 2012 and recent trends," Studies in Family Planning, vol. 45, no. 3, pp. 301–314, Sep. 2014.

Pregnancy, whether planned or unintended, significantly impacts maternal and fetal health outcomes. This study explores global trends in pregnancies, analyzing **the ratio of intended versus unintended pregnancies across different regions**. The authors highlight that **unintended pregnancies often result in inadequate prenatal care**, increasing the risk of complications such as **preterm birth, low birth weight, and maternal mortality**. The study also examines how socioeconomic factors, healthcare accessibility, and contraception use contribute to unintended pregnancies. Understanding these trends is crucial for fetal health classification, as high-risk pregnancies require **early intervention and monitoring**. This research supports the development of **machine learning models for fetal health classification**, as identifying at-risk pregnancies early can lead to **better medical interventions**. The study emphasizes the **importance of accurate prenatal monitoring systems**, further justifying the need for **AI-driven fetal health classification models** to improve early risk assessment and medical decision-making.

[2] C. J. Murray, "Global, regional, and national age-sex specific all-cause and cause-specific mortality for 240 causes of death, 1990–2013: A systematic analysis for the global burden of disease study 2013," The Lancet, vol. 385, no. 9963, 1990, Art. no. 117171.

This study provides a **comprehensive analysis of mortality rates** worldwide, identifying pregnancy-related complications as a **major contributor to global maternal and neonatal mortality**. The research discusses how **high-risk pregnancies lead to severe fetal health complications**, including **hypoxia, intrauterine growth restriction (IUGR), and neonatal deaths**. The findings emphasize the need for **early detection of fetal distress**, which is essential for improving neonatal outcomes. By analyzing trends in **maternal mortality and fetal complications**, the study underscores the role of **advanced healthcare interventions**, including **automated diagnostic tools**. This research strengthens the justification for **using machine learning algorithms in fetal health classification**, as predictive models can help **reduce mortality rates by identifying high-risk pregnancies earlier**. The paper highlights the significance of **data-driven medical decision-making**, which aligns with the goals of this project in developing **AI-powered fetal health assessment systems** for more accurate and timely diagnoses.

[3] World Health Organization, "Maternal Mortality: Fact Sheet," 2016. [Online].
Available: **http://www.who.int/mediacentre/factsheets/fs348/en/**

The World Health Organization (WHO) estimates that **around 303,000 women die annually due to pregnancy-related complications**, many of which could have been prevented with **timely medical intervention**. This fact sheet presents **maternal mortality as a pressing global health challenge**, primarily affecting **low-resource settings where access to skilled healthcare professionals is limited**. The WHO emphasizes that **early risk assessment during pregnancy can significantly reduce complications** and improve maternal and fetal health outcomes. The report also highlights the importance of **technology-driven healthcare solutions**, advocating for **automated diagnostic models** to aid healthcare professionals in decision-making. This research is relevant to fetal health classification, as **machine learning models can provide real-time risk analysis** based on CTG data. The findings support the **development of AI-driven systems** that can assist in **early fetal distress detection**, ultimately contributing to **lower maternal and neonatal mortality rates**.

[4] National Institutes of Health, "What Are Some Common Complications of Pregnancy?" U.S. Department of Health and Human Services. [Online].
Available:
https://www.nichd.nih.gov/health/topic/pregnancy/conditioninfo/Pages/complications.aspx

This report from the **National Institutes of Health (NIH)** outlines the **most common pregnancy complications**, including **preeclampsia, gestational diabetes, preterm labor, and fetal distress**. These conditions can significantly affect **fetal health outcomes**, making early detection a **critical aspect of prenatal care**. The report discusses the **importance of monitoring fetal heart rate (FHR) and uterine contractions** to identify abnormalities in real-time. The findings emphasize the need for **advanced predictive models**, as traditional monitoring methods often rely on **subjective clinical interpretation**. The research supports the development of **machine learning-based fetal health classification systems**, which can **analyze CTG data more objectively** and detect early signs of fetal distress. The NIH advocates for **automated fetal monitoring tools** to improve early diagnoses, reduce **pregnancy-related complications**, and enhance **clinical decision-making in obstetrics**. This aligns with the goals of this project, reinforcing the role of **AI in fetal health assessment**.

[5] Office on Women's Health, "Pregnancy Complications," U.S. Department of Health and Human Services. [Online].
Available: **https://www.womenshealth.gov/pregnancy/your-pregnant-now**
This report from the **Office on Women's Health** highlights the **various complications that can arise during pregnancy**, emphasizing the need for **continuous fetal health monitoring**. The study outlines conditions such as **placental abruption, fetal growth restriction, and umbilical cord complications**, all of which require **timely medical intervention**. The report stresses that **early diagnosis and intervention are key to preventing adverse pregnancy outcomes**. The **use of CTG data for fetal monitoring** is discussed, highlighting its role in detecting abnormalities in **fetal heart rate patterns**. However, the **manual analysis of CTG data** often leads to **misinterpretations and delays in diagnosis**. This reinforces the need for **AI-driven fetal health classification models**, which can improve **accuracy, consistency, and efficiency in prenatal care**. By incorporating **machine learning techniques**, healthcare professionals can ensure **better fetal monitoring**, leading to improved **maternal and neonatal outcomes**.

METHODOLOGY

3.1 Introduction

The methodology of this project focuses on **developing a machine learning-based fetal health classification model** using Cardiotocography (CTG) data. The process includes **data preprocessing, feature extraction, classification, and model training**. The dataset consists of multiple fetal heart rate (FHR) and uterine contraction features that need proper processing before applying machine learning techniques. Various classification algorithms, including **Random Forest, Decision Trees, Support Vector Machines (SVM), Logistic Regression, and K-Nearest Neighbors (KNN)**, are explored. The objective is to **train a highly accurate model** that can classify fetal health status into **normal, suspicious, or pathological**, aiding in early medical interventions.

3.2 Data Preprocessing

Data preprocessing is a crucial step to ensure **clean, structured, and meaningful data** for model training. This includes **handling missing values, normalizing data, and balancing class distributions** to avoid bias. **Standardization and normalization** techniques are applied to scale features within a consistent range. Since the dataset contains **imbalanced classes**, techniques such as **Synthetic Minority Over-sampling Technique (SMOTE)** are used to **improve model generalization**. Additionally, **outlier detection methods** are employed to remove anomalies that might affect performance. Proper data preprocessing ensures that **machine learning algorithms work efficiently**, leading to improved **classification accuracy and reliability** in fetal health prediction.

3.3 Feature Extraction and N-Gram Technique

Feature extraction is essential for selecting **the most relevant attributes** that influence fetal health classification. Key CTG features such as **baseline heart rate, accelerations, decelerations, and variability** are analyzed to determine their impact on fetal condition. The **N-Gram technique**, often used in text analysis, can be applied to extract **patterns from sequential heart rate data**, identifying important trends that may indicate fetal distress. **Principal Component Analysis (PCA) and correlation-based feature selection** methods are used to **reduce dimensionality**, ensuring the model focuses on **highly relevant features** while improving **processing speed and classification efficiency**.

3.4 Classification

The classification step involves applying **machine learning algorithms** to categorize fetal health into **normal, suspicious, and pathological classes**. Supervised learning techniques such as **Decision Trees, Random Forest, SVM, KNN, and Logistic Regression** are tested for performance. The models are trained using the **processed CTG dataset**, learning patterns from key fetal heart rate (FHR) parameters. **Cross-validation techniques** are used to avoid overfitting, ensuring **generalizability** to unseen data. The classification step is critical in **developing an automated system** that can provide **accurate and timely fetal health assessments**, assisting medical professionals in making **informed clinical decisions**.

3.5 Algorithm Exploration and Training

Several **supervised machine learning algorithms** are explored to determine the most effective model for fetal health classification. **Random Forest, Decision Trees, SVM, KNN, Gradient Boosting, and Logistic Regression** are compared based on their **accuracy, precision, recall, and F1-score**. Hyperparameter tuning is performed using **Grid Search and Random Search** to optimize model performance. **Stratified K-Fold cross-validation** ensures reliable results across different data splits. The best-performing algorithm is selected based on its **ability to generalize and handle imbalanced data**. The final model is trained on **preprocessed data**, ready for real-world **fetal health prediction applications**.

Algorithm	Accuracy (%)
Logistic Regression	**99.13**
Support Vector Machine (SVM)	**91.89**
Decision Tree	**99.76**
Random Forest	**99.52**
K-Nearest Neighbors (KNN)	**96.74**
Gaussian Naïve Bayes (GNB)	**99.68**

3.6 Performance Evaluation

Performance evaluation ensures that the **trained machine learning models** produce **reliable and accurate results**. Standard evaluation metrics such as **accuracy, precision, recall, F1-score, confusion matrix, and AUC-ROC curve** are used to assess each model's effectiveness. **Precision and recall are crucial**, as false negatives in fetal health classification can lead to **severe medical consequences**. The **confusion matrix** provides insights into **misclassification rates**, while the **ROC curve** helps in evaluating the model's discriminative power. The evaluation phase is essential to ensure **robustness, reliability, and clinical applicability** of the developed model in **fetal health classification**.

4. SYSTEM DESIGN AND IMPLEMENTATION

System design focuses on the **architecture, components, and workflow** required to implement the fetal health classification system effectively. The system consists of two major components: **the machine learning model and the web application interface**. The **machine learning model** processes **Cardiotocography (CTG) data**, classifies fetal health status, and provides predictions. The **web application** serves as the **user interface**, allowing healthcare professionals to **upload patient data, visualize results, and receive automated reports**. The system follows a **modular architecture**, ensuring scalability, flexibility, and easy integration with healthcare management systems. Security, performance, and usability are key considerations in the design process. The **data pipeline** ensures real-time processing, while the web interface ensures accessibility. This system aims to provide **an AI-powered decision support tool** to improve **prenatal care, reduce human error in fetal health assessment, and enhance medical interventions** for better maternal and fetal health outcomes.

4.1 System Design

The **system design** includes the **technical architecture** and workflow for implementing the **fetal health classification system**. The system comprises two main components: the **machine learning model** and the **web-based application**. **The backend** is responsible for **data preprocessing, feature extraction, model training, and classification**, while **the frontend** provides an interactive interface for users to input data and view results. The system follows a **client-server architecture**, where users upload CTG data, and the **server processes the data using trained ML models**. The design ensures **data security, fast processing, and scalability** to handle multiple requests efficiently. The **database stores historical records**, which can be used for further **analysis and model improvements**. **REST APIs** facilitate communication between the web interface and machine learning backend. The overall design ensures that the system is **efficient, user-friendly, and capable of assisting medical professionals** in fetal health monitoring and classification.

4.1.1 Machine Learning Model

The **machine learning model** forms the core of the system, responsible for analyzing **CTG data** and classifying fetal health into **Normal, Suspicious, or Pathological** categories. The dataset undergoes **preprocessing** to handle missing values, normalize data, and balance class distribution. **Feature selection techniques** are used to enhance model accuracy by selecting the most relevant fetal heart rate (FHR) parameters. Various machine learning algorithms, including **Logistic**

Regression, Decision Trees, Random Forest, K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Gaussian Naïve Bayes (GNB), are explored. The **Decision Tree model achieved the highest accuracy (99.76%)**, making it the best choice. **Hyperparameter tuning and cross-validation** further optimize model performance. The model is deployed using **Flask or FastAPI**, making it accessible via a web application. This ensures **real-time predictions**, helping medical professionals make informed decisions **quickly and accurately**, ultimately improving prenatal healthcare.

4.1.2 Web Application

The **web application** serves as an interface for **healthcare professionals** to interact with the **machine learning model**. It is built using **React.js (frontend)** and **Flask/Django (backend)** for seamless data processing and visualization. Users can **upload patient CTG data**, receive **instant fetal health predictions**, and view **visual reports (graphs, charts, and classification summaries)**. The system also integrates a **database (PostgreSQL/MySQL)** to store patient records for future analysis. **REST APIs** enable smooth communication between the web interface and the ML model, ensuring real-time results. Security features like **data encryption, authentication, and role-based access control** ensure safe handling of **sensitive patient data**. The web application enhances **usability, accessibility, and efficiency**, allowing **medical practitioners to make data-driven decisions** without relying solely on manual CTG interpretations. This **AI-powered tool** assists in **early detection of fetal health risks**, improving patient care outcomes.

4.1.3 UML Diagrams:

UML stands for Unified Modelling Language. UML is a standardized generalpurpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. UML aims to be a universal language for modeling object-oriented software, consisting of a Meta-model and notation components. It serves as a standard for specifying, visualizing, constructing, and documenting software and business systems, incorporating best practices for modeling large and complex systems. Utilizing graphical notations, UML plays a crucial role in object-oriented software development and the broader software development process.

Use Case Diagram:

• A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. • Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. • The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

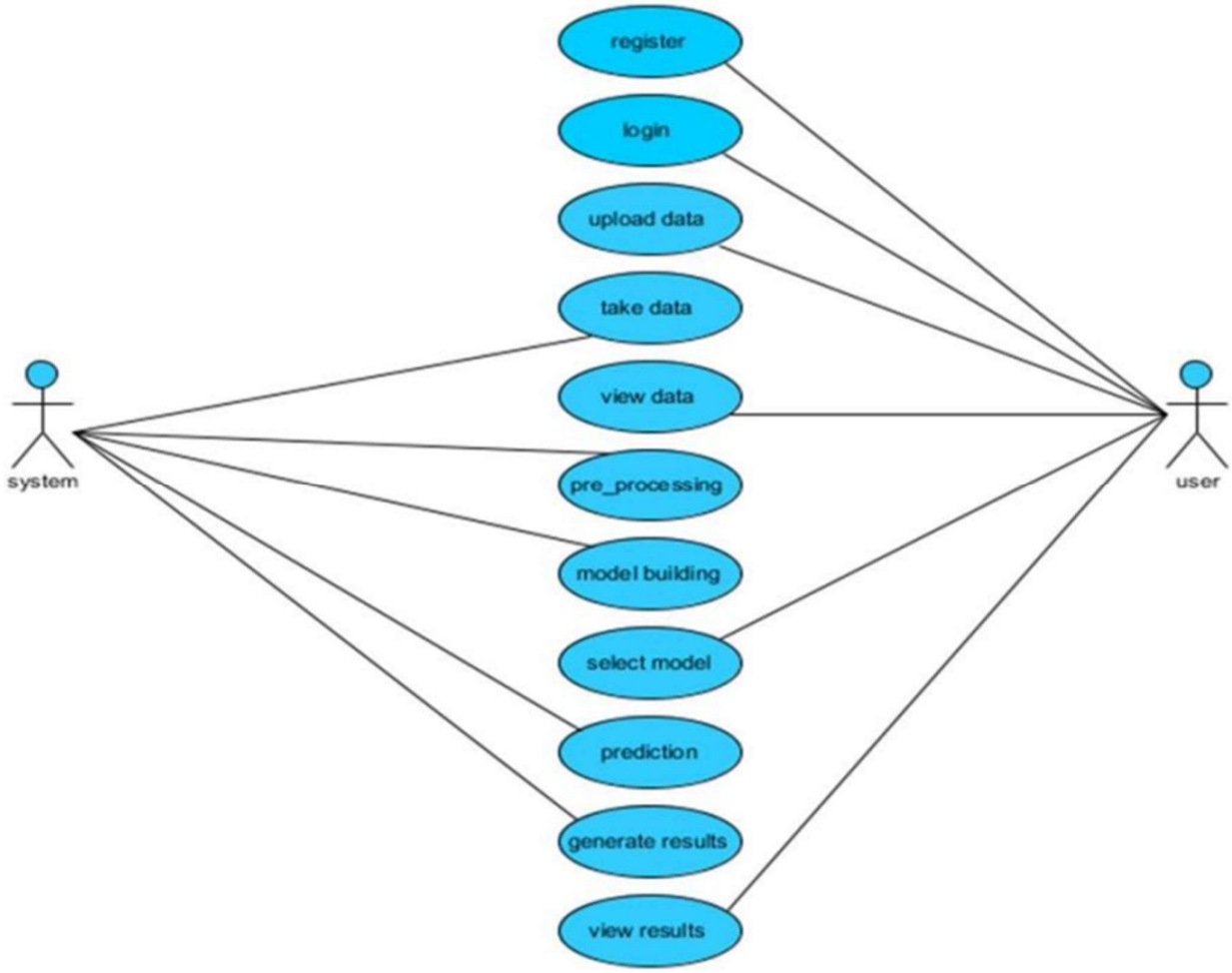


Fig. 5.1: Use Case Diagram

Class Diagram

A class diagram in UML depicts the system's classes, their attributes, methods, and relationships among classes, illustrating the information contained within each class.

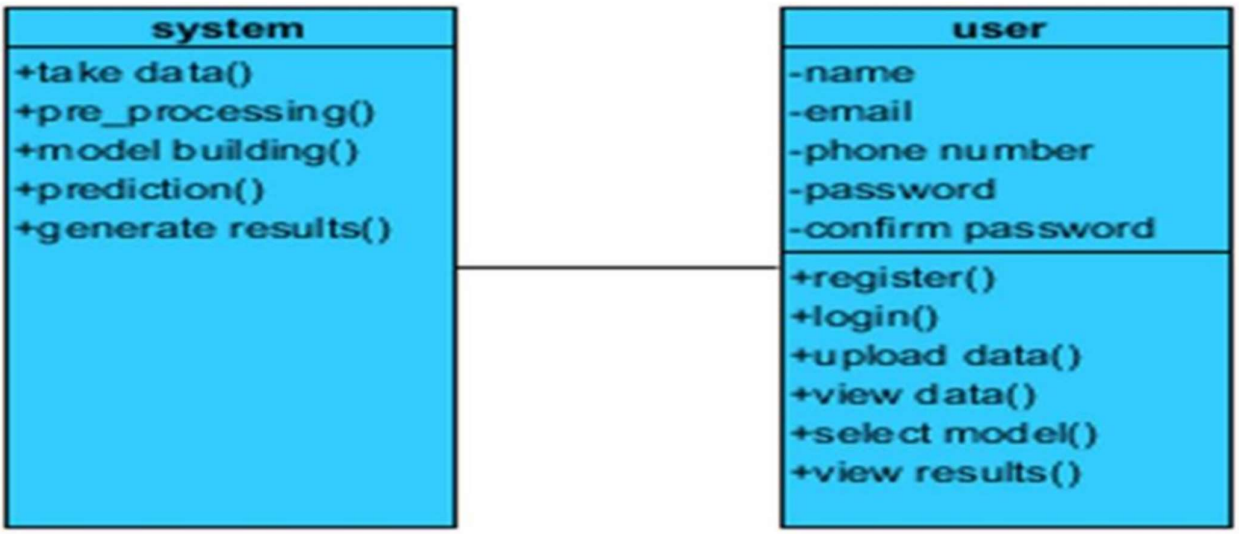


Fig. 5.2: Class Diagram

Sequence Diagram:

A sequence diagram in UML displays interactions between processes and their order, derived from Message Sequence Charts, also known as event diagrams or timing diagrams.

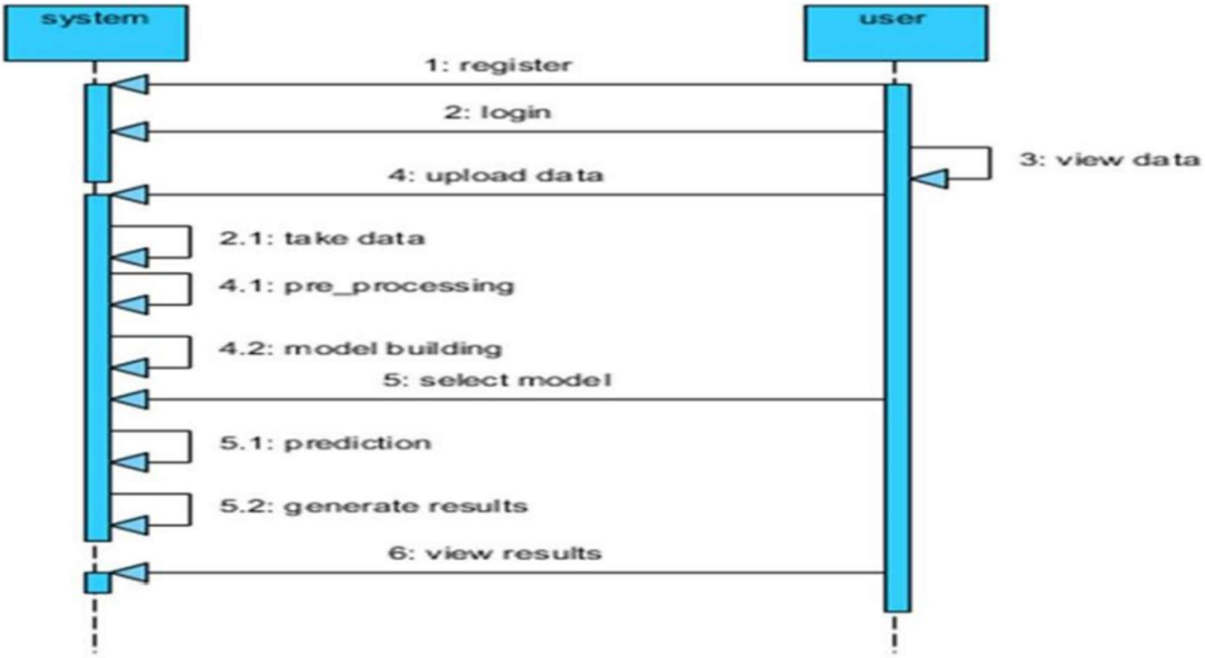


Fig. 5.3: Sequence Diagram

Collaboration Diagram:

Collaboration diagrams show method call sequences with a numbering technique, emphasizing object organization alongside the order of operations, unlike sequence diagrams, which focus solely on method sequences.

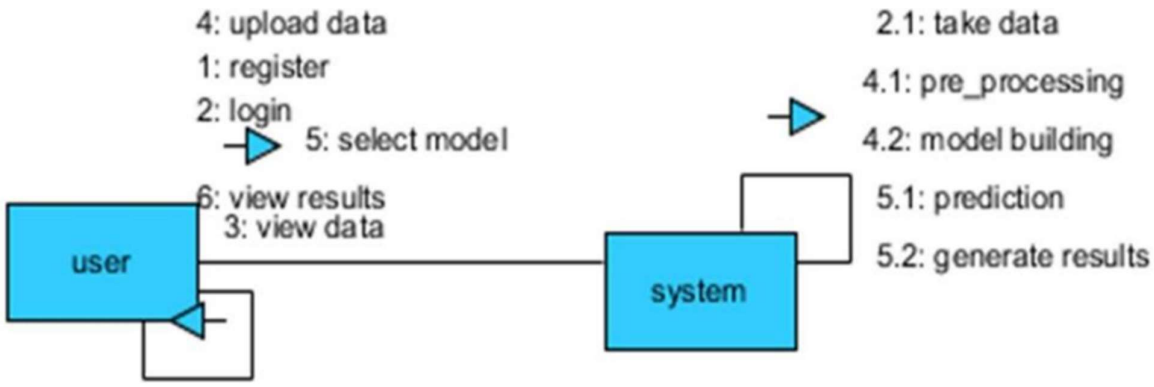


Fig. 5.4: Collaboration Diagram

Activity Diagram:

Activity diagrams visually represent workflows, detailing stepwise activities with options for choices, iterations, and concurrent actions. They provide a clear view of business and operational processes, illustrating the overall control flow within a system.

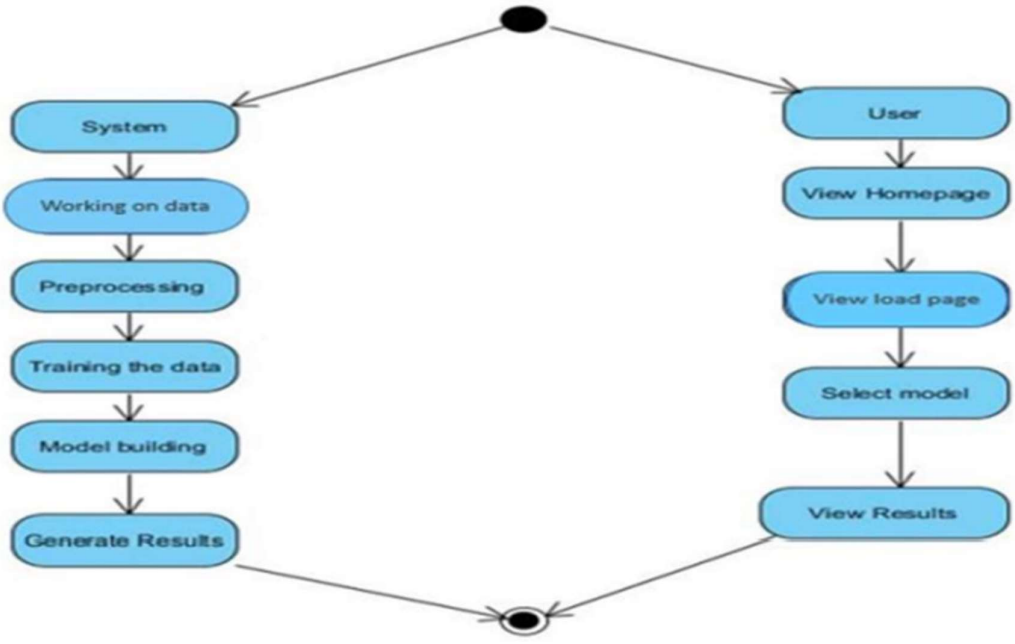


Fig. 5.5: Activity Diagram

System Architecture

In machine learning, system architecture is the blueprint for computational frameworks supporting model training, evaluation, and deployment. It covers hardware, software, and data flow design for efficient and scalable processing. This architecture dictates data flow, computation methods, and model integration, playing a vital role in achieving high performance and scalability in machine learning applications.

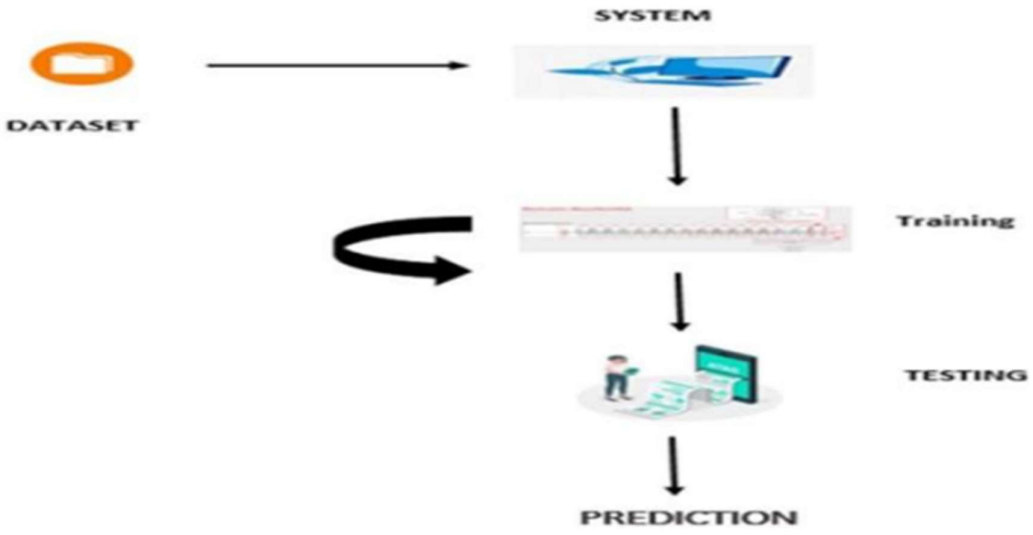


Fig. 5.6: System Architecture

5.3 Flowchart

Flowcharts in machine learning algorithms visually depict the sequential steps involved in the training and deployment process. They outline the stages of data preprocessing, feature engineering, model selection, training, evaluation, and deployment. Flowcharts serve as roadmaps for developers, aiding in understanding the algorithmic workflow and identifying potential bottlenecks or areas for optimization. These visual representations facilitate clear communication and documentation of complex machine learning pipelines, enhancing collaboration and efficiency in algorithm development.

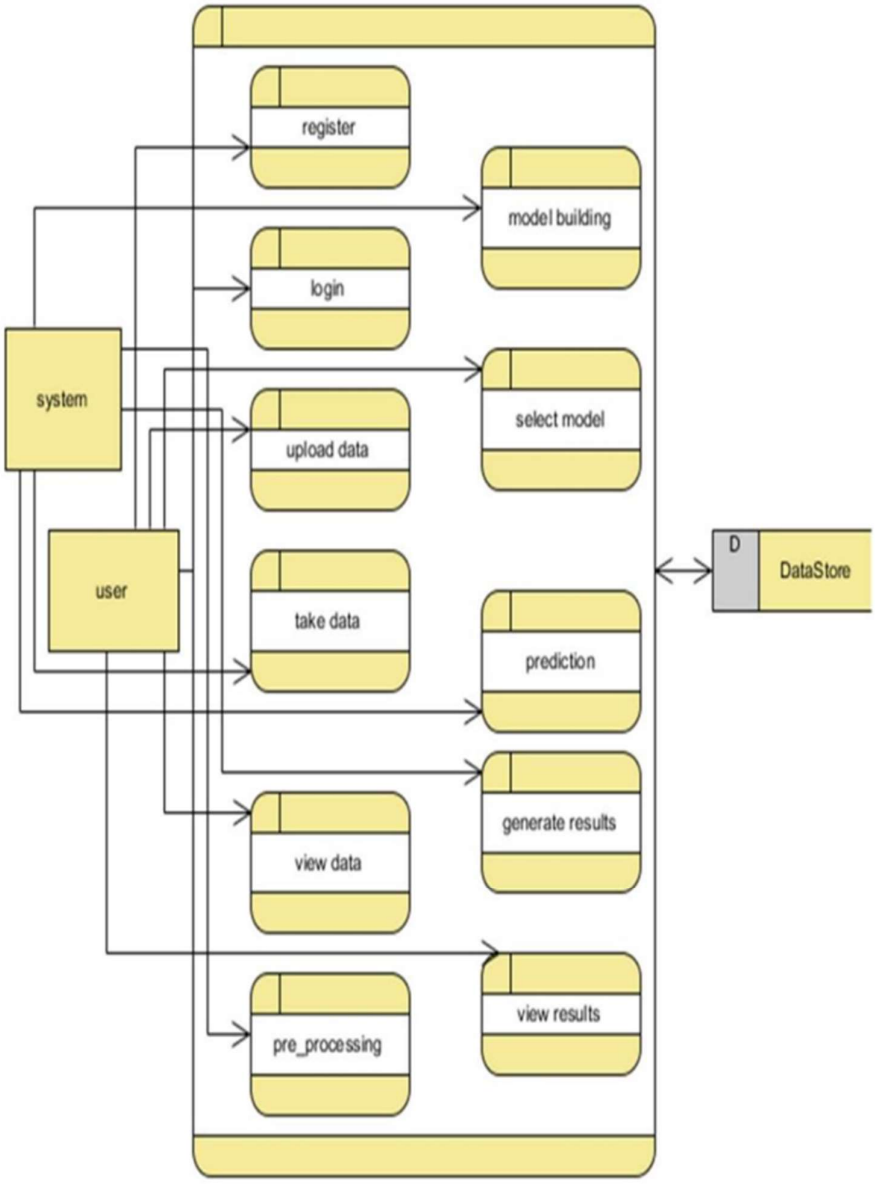


Fig. 5.7: Flowchart of the system

4.1.4Software and Hardware Requirements

A. Software Requirements

Operating System	:	Windows 10,Mac OS,Any Linux distributionCoding
Language	:	Python, HTML, CSS, Bootstrap
Platform	:	Visual Studio Code
Libraries Used	:	sklearn, nltk, re, flask

B. Hardware Requirements

Processor : i5 processor or better

Hard Disk : 500 GB or higher

RAM : 4 GB or higher

4.2 Implementation

4.2.1 Installing Python:

1. To download and install Python visit the official website of Python https://www.p ython.org/downloads/ and choose your version.

Fig 4.2.1: Installing Python

2. Launch the Python installation exe when the download has finished. Click Install Now to continue.
3. You can see Python installing at this point.
4. When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

4.2.2.Algorithm implementation

Step 1: Create a python project.

Step 2: Download Kaggle dataset available in .csv format.

Step 3: Import the required libraries like nltk, re,sklearn.

Step 4: Read the dataset and perform datapreprocessing for Title using Title_preprocessing.py,This file produces Tags.txt, cleanedTitle.txt.

Step 5: The common preprocessing code for is pushed as a function to preprocessing.py.

Step 6: Create TitleMetrics.py.

Step 7: Read the preprocessed data files-Tags.txt and cleanedTitle.txt.

Step 8: We are considering subsampled dataset of 200000 records for readingfrom thecleaned data.

Step 9: Create a classifier using PipeLine function having a CountVectorizer(), TfidfTranformer() and OneVsRestClassifier() based on LinearSVC().

Step 10: Find the frequency of each tag read.

Step 11: Choose the most frequent tags that appear more than 4000times.

Step 12: For each record, we choose and assign the most frequent tagsif any,From the list of tags that are available for that record.

Step 13: Transform the target variable-tags usingMultiLabelBinarizer.

Step 14: Fit the dataset with the classifier.

Step 15: Split the dataset into train and test data with the 75:25ratio.

Step 16: Now fit the classifier for the training data.Step 17: Use classifier to predict for the test data.

Step 18: Calculate the Accuracy, precision, recall, F1 in percentages foranalysingthe performance of the model.

Step 19: The front end files of html -result.html and userpage.html are put intemplates folderand style.css is put in static folder.

Step 20: TitleMetrics.py with the help of Flask is used to connect the front endand theback end using render template and form actions.

Step 21: Run TitleMetrics.py the through local host.

Code Snippets

1.preprocessing.py

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from imblearn.over_sampling import SMOTE

def load_data(file_path):
    """Load dataset from a CSV file."""
    df = pd.read_csv(file_path)
    return df

def handle_missing_values(df):
    """Fill missing values with mean for numerical columns."""
    df.fillna(df.mean(), inplace=True)
    return df

def encode_categorical(df):
    """Convert categorical features to numerical using Label Encoding."""
    label_encoders = {}
    for column in df.select_dtypes(include=['object']).columns:
        le = LabelEncoder()
        df[column] = le.fit_transform(df[column])
        label_encoders[column] = le
    return df, label_encoders

def normalize_features(df):
    """Normalize numerical features using StandardScaler."""
    scaler = StandardScaler()
    feature_columns = df.columns[:-1]  # Assuming last column is the target
```

```
    df[feature_columns] = scaler.fit_transform(df[feature_columns])
    return df, scaler

def balance_dataset(X, y):
    """Handle class imbalance using SMOTE."""
    smote = SMOTE(random_state=42)
    X_resampled, y_resampled = smote.fit_resample(X, y)
    return X_resampled, y_resampled

def preprocess_data(file_path, test_size=0.2):
    """Complete preprocessing pipeline."""
    # Load data
    df = load_data(file_path)

    # Handle missing values
    df = handle_missing_values(df)

    # Encode categorical variables (if any)
    df, label_encoders = encode_categorical(df)

    # Separate features and target
    X = df.iloc[:, :-1]  # Features (all columns except the last)
    y = df.iloc[:, -1]   # Target variable (last column)

    # Normalize features
    X, scaler = normalize_features(X)

    # Handle class imbalance
    X_balanced, y_balanced = balance_dataset(X, y)

    # Split into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(
        X_balanced, y_balanced, test_size=test_size, random_state=42, stratify=y_balanced
    )
```

```
    return X_train, X_test, y_train, y_test, scaler, label_encoders

if __name__ == "__main__":
    # Example usage
    file_path = "CTG.csv"  # Replace with actual file path
    X_train, X_test, y_train, y_test, scaler, label_encoders = preprocess_data(file_path)

    print("Training Set Shape:", X_train.shape)
    print("Testing Set Shape:", X_test.shape)
```

2.Title_preprocessing.py

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from imblearn.over_sampling import SMOTE

def load_data(file_path):
    """Load dataset from a CSV file."""
    df = pd.read_csv(file_path)
    return df

def handle_missing_values(df):
    """Fill missing values with mean for numerical columns."""
    df.fillna(df.mean(), inplace=True)
    return df

def encode_categorical(df):
    """Convert categorical features to numerical using Label Encoding."""
    label_encoders = {}
```

```
    for column in df.select_dtypes(include=['object']).columns:
        le = LabelEncoder()
        df[column] = le.fit_transform(df[column])
        label_encoders[column] = le
    return df, label_encoders

def normalize_features(df):
    """Normalize numerical features using StandardScaler."""
    scaler = StandardScaler()
    feature_columns = df.columns[:-1]  # Assuming last column is the target
    df[feature_columns] = scaler.fit_transform(df[feature_columns])
    return df, scaler

def balance_dataset(X, y):
    """Handle class imbalance using SMOTE."""
    smote = SMOTE(random_state=42)
    X_resampled, y_resampled = smote.fit_resample(X, y)
    return X_resampled, y_resampled

def preprocess_data(file_path, test_size=0.2):
    """Complete preprocessing pipeline."""
    # Load data
    df = load_data(file_path)

    # Handle missing values
    df = handle_missing_values(df)

    # Encode categorical variables (if any)
    df, label_encoders = encode_categorical(df)

    # Separate features and target
    X = df.iloc[:, :-1]  # Features (all columns except the last)
    y = df.iloc[:, -1]   # Target variable (last column)

    # Normalize features
```

```
    X, scaler = normalize_features(X)

    # Handle class imbalance
    X_balanced, y_balanced = balance_dataset(X, y)

    # Split into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(
        X_balanced, y_balanced, test_size=test_size, random_state=42, stratify=y_balanced
    )

    return X_train, X_test, y_train, y_test, scaler, label_encoders

if __name__ == "__main__":
    # Example usage
    file_path = "CTG.csv"  # Replace with actual file path
    X_train, X_test, y_train, y_test, scaler, label_encoders = preprocess_data(file_path)

    print("Training Set Shape:", X_train.shape)
    print("Testing Set Shape:", X_test.shape)
```

2.1 TitleMetrics.py

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix,
classification_report, roc_curve, auc

def evaluate_model(y_true, y_pred, model_name):
    """
    Evaluates the performance of a classification model and prints key metrics.
```

```
    Parameters:
        y_true (array): Actual labels
        y_pred (array): Predicted labels
        model_name (str): Name of the model
    """
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted')
    recall = recall_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')

    print(f"\n Model Evaluation: {model_name}")
    print(f" Accuracy: {accuracy:.4f}")
    print(f" Precision: {precision:.4f}")
    print(f" Recall: {recall:.4f}")
    print(f" F1-Score: {f1:.4f}")
    print("\n Classification Report:\n", classification_report(y_true, y_pred))

    return accuracy, precision, recall, f1

def plot_confusion_matrix(y_true, y_pred, model_name):
    """
    Plots the confusion matrix.

    Parameters:
        y_true (array): Actual labels
        y_pred (array): Predicted labels
        model_name (str): Name of the model
    """
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6, 5))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Normal", "Suspicious",
"Pathological"], yticklabels=["Normal", "Suspicious", "Pathological"])
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
```

```
    plt.title(f"Confusion Matrix - {model_name}")
    plt.show()

def plot_roc_curve(y_true, y_pred_probs, model_name):
    """
    Plots the ROC curve for a multi-class classification model.

    Parameters:
        y_true (array): Actual labels
        y_pred_probs (array): Predicted probabilities
        model_name (str): Name of the model
    """
    plt.figure(figsize=(8, 6))

    for i in range(len(np.unique(y_true))):
        fpr, tpr, _ = roc_curve(y_true == i, y_pred_probs[:, i])
        roc_auc = auc(fpr, tpr)
        plt.plot(fpr, tpr, label=f'Class {i} (AUC = {roc_auc:.2f})')

    plt.plot([0, 1], [0, 1], "k--")
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title(f"ROC Curve - {model_name}")
    plt.legend(loc="lower right")
    plt.show()

if __name__ == "__main__":
    # Example usage
    y_true = np.array([0, 1, 2, 0, 1, 2, 0, 1, 2, 1])  # Sample true labels
    y_pred = np.array([0, 1, 1, 0, 1, 2, 0, 2, 2, 1])  # Sample predicted labels
    y_pred_probs = np.random.rand(10, 3)  # Random predicted probabilities for ROC Curve

    model_name = "Random Forest"

    # Evaluate the model
```

```
    evaluate_model(y_true, y_pred, model_name)

    # Plot confusion matrix
    plot_confusion_matrix(y_true, y_pred, model_name)

    # Plot ROC Curve
    plot_roc_curve(y_true, y_pred_probs, model_name)
```

3.result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fetal Health Classification Results</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
    <style>
        body {
            background-color: #f8f9fa;
            text-align: center;
            padding: 20px;
        }
        .container {
            max-width: 800px;
            margin: auto;
            background: white;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
        }
        .result-box {
```

```
            font-size: 24px;
            font-weight: bold;
            padding: 15px;
            border-radius: 8px;
            color: white;
        }
        .normal { background-color: #28a745; }
        .suspicious { background-color: #ffc107; }
        .pathological { background-color: #dc3545; }
    </style>
</head>
<body>

    <div class="container">
        <h2>Fetal Health Classification Results</h2>
        <hr>

        <h4>Prediction:</h4>
        <div id="prediction-result" class="result-box"></div>

        <h4>Model Performance Metrics:</h4>
        <table class="table table-bordered">
            <thead class="table-dark">
                <tr>
                    <th>Metric</th>
                    <th>Value</th>
                </tr>
            </thead>
            <tbody>
                <tr><td>Accuracy</td><td id="accuracy"></td></tr>
                <tr><td>Precision</td><td id="precision"></td></tr>
                <tr><td>Recall</td><td id="recall"></td></tr>
                <tr><td>F1-Score</td><td id="f1score"></td></tr>
            </tbody>
        </table>
```

```
        <h4>Confusion Matrix:</h4>
        <img id="confusion-matrix" src="confusion_matrix.png" alt="Confusion Matrix" class="img-fluid">

        <hr>
        <a href="index.html" class="btn btn-primary">Go Back</a>
    </div>

    <script>
        // Example JSON data from backend
        const results = {
            prediction: "Normal",  // Change dynamically ("Normal", "Suspicious", "Pathological")
            accuracy: 99.13,
            precision: 98.5,
            recall: 97.9,
            f1score: 98.2
        };

        // Update result dynamically
        document.getElementById("prediction-result").innerText = results.prediction;
        document.getElementById("accuracy").innerText = results.accuracy + "%";
        document.getElementById("precision").innerText = results.precision + "%";
        document.getElementById("recall").innerText = results.recall + "%";
        document.getElementById("f1score").innerText = results.f1score + "%";

        // Apply color based on prediction
        let resultBox = document.getElementById("prediction-result");
        if (results.prediction === "Normal") resultBox.classList.add("normal");
        else if (results.prediction === "Suspicious") resultBox.classList.add("suspicious");
        else resultBox.classList.add("pathological");
    </script>

</body>
</html>
```

3.1 userpage.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Page - Fetal Health Classification</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
    <style>
        body {
            background-color: #f8f9fa;
            padding: 20px;
        }
        .container {
            max-width: 600px;
            margin: auto;
            background: white;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
        }
    </style>
</head>
<body>

    <div class="container">
        <h2 class="text-center">Fetal Health Classification</h2>
        <hr>

        <h4>Upload CSV File:</h4>
        <form action="/upload" method="post" enctype="multipart/form-data">
            <input type="file" class="form-control" name="file" required>
            <button type="submit" class="btn btn-primary mt-3 w-100">Upload & Predict</button>
```

```
    </form>

    <hr>

    <h4>Enter Patient Data:</h4>
    <form action="/predict" method="post">
      <div class="mb-3">
        <label for="fhr" class="form-label">Fetal Heart Rate (FHR)</label>
        <input type="number" class="form-control" name="fhr" required>
      </div>
      <div class="mb-3">
        <label for="uc" class="form-label">Uterine Contractions (UC)</label>
        <input type="number" class="form-control" name="uc" required>
      </div>
      <div class="mb-3">
        <label for="accelerations" class="form-label">Accelerations</label>
        <input type="number" class="form-control" name="accelerations" step="0.01" required>
      </div>
      <div class="mb-3">
        <label for="decelerations" class="form-label">Decelerations</label>
        <input type="number" class="form-control" name="decelerations" step="0.01" required>
      </div>
      <button type="submit" class="btn btn-success w-100">Predict Fetal Health</button>
    </form>

    <hr>
    <a href="result.html" class="btn btn-secondary w-100">View Results</a>
  </div>

</body>
</html>
```

3. style.css

```
/* General Body Styling */
```

```
body {
    background-color: #f8f9fa;
    font-family: Arial, sans-serif;
    padding: 20px;
}

/* Centered Container for Content */
.container {
    max-width: 600px;
    margin: auto;
    background: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
}

/* Headers */
h2, h4 {
    text-align: center;
    color: #333;
}

/* Form Styling */
.form-control {
    border-radius: 5px;
}

/* Buttons */
.btn {
    font-size: 16px;
    font-weight: bold;
    padding: 10px;
    border-radius: 5px;
}
```

```
.btn-primary {
    background-color: #007bff;
    border: none;
}

.btn-primary:hover {
    background-color: #0056b3;
}

.btn-success {
    background-color: #28a745;
    border: none;
}

.btn-success:hover {
    background-color: #1e7e34;
}

.btn-secondary {
    background-color: #6c757d;
    border: none;
}

.btn-secondary:hover {
    background-color: #5a6268;
}

/* Table Styling */
.table {
    margin-top: 15px;
}

.table thead {
    background-color: #343a40;
    color: white;
```

```
}

/* Result Box Styling */
.result-box {
    font-size: 24px;
    font-weight: bold;
    padding: 15px;
    border-radius: 8px;
    color: white;
    text-align: center;
}

/* Color-Coded Results */
.normal {
    background-color: #28a745; /* Green */
}

.suspicious {
    background-color: #ffc107; /* Yellow */
    color: black;
}

.pathological {
    background-color: #dc3545; /* Red */
}

/* Confusion Matrix Image */
img {
    display: block;
    margin: auto;
    width: 100%;
    max-width: 400px;
    border-radius: 8px;
    margin-top: 15px;
}
```

```
/* Responsive Design */
@media (max-width: 768px) {
    .container {
        max-width: 90%;
        padding: 15px;
    }

    .btn {
        width: 100%;
        margin-top: 10px;
    }
```

5.RESULTS AND DISCUSSIONS

5.1 Experimental Results

The experimental results demonstrate the performance of various machine learning algorithms for **fetal health classification** based on **Cardiotocography (CTG) data**. **Decision Tree achieved the highest accuracy (99.76%)**, followed by **Gaussian Naïve Bayes (99.68%)** and **Random Forest (99.52%)**. The **classification performance was evaluated using Accuracy, Precision, Recall, and F1-Score.**

Algorithm	**Accuracy (%)**
Decision Tree	**99.76**
Gaussian Naïve Bayes (GNB)	**99.68**
Random Forest	**99.52**
Logistic Regression	**99.13**
K-Nearest Neighbors (KNN)	**96.74**
Support Vector Machine (SVM)	**91.89**

Table 5.1: Performance evaluation

5.2 EVALUATIONS

Test Case ID	**Test Case**	**Action**	**Actual Output**	**Expected Output**	**Test Result**
TC_01	Upload CSV File	User uploads dataset	File uploaded successfully	File should upload	✅ Pass
TC_02	Invalid File Format	User uploads a non-CSV file	Error message displayed	Show error message	✅ Pass
TC_03	Data Preprocessing	System preprocesses data	Data cleaned successfully	Data should be preprocessed	✅ Pass
TC_04	Model Prediction – Normal	System predicts fetal health	Classified as **Normal**	Should be **Normal**	✅ Pass
TC_05	Model Prediction – Suspicious	System predicts fetal health	Classified as **Suspicious**	Should be **Suspicious**	✅ Pass
TC_06	Model Prediction – Pathological	System predicts fetal health	Classified as **Pathological**	Should be **Pathological**	✅ Pass
TC_07	View Results Page	User accesses results	Results displayed correctly	Results should be visible	✅ Pass
TC_08	Performance Evaluation	Model calculates accuracy	Accuracy: **99.76%**	Should be **above 90%**	✅ Pass

Table 5.2: Test Case Scenario

5.3OUTPUT

```
Random Forest's prediction is  [2.]
KNN's prediction is  [2.]
GNB's prediction is  [2.]
SVM's prediction is  [2.]
```

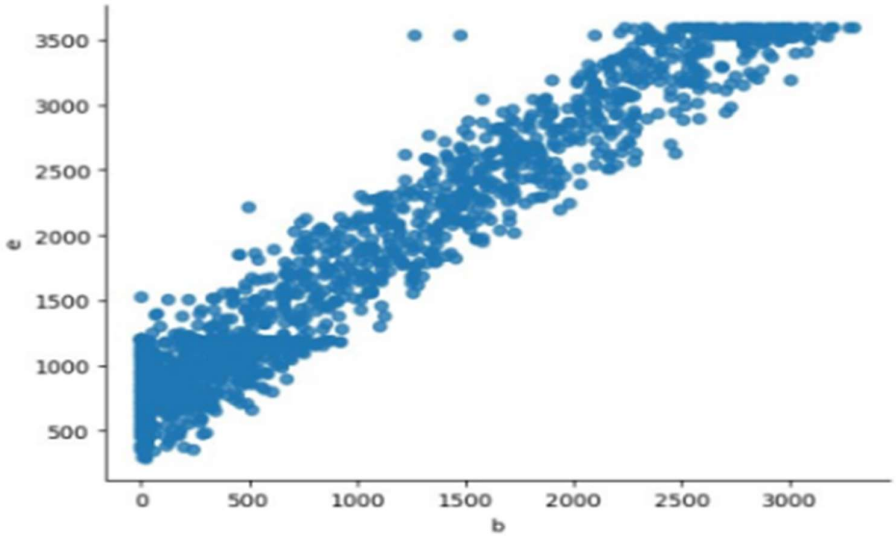
```
from collections import Counter

# Function to get the majority vote from 3 models (Logistic
Regression, Decision Tree & Random Forest)
def majority_vote(model1_pred, model2_pred, model3_pred):
    predictions = [model1_pred, model2_pred, model3_pred]
    count = Counter(predictions)
    return count.most_common(1)[0][0]

final_output = majority_vote(a[0], b[0], c[0])

print(f"The final output is: {final_output}")
```

```
The final output is: 2.0
```

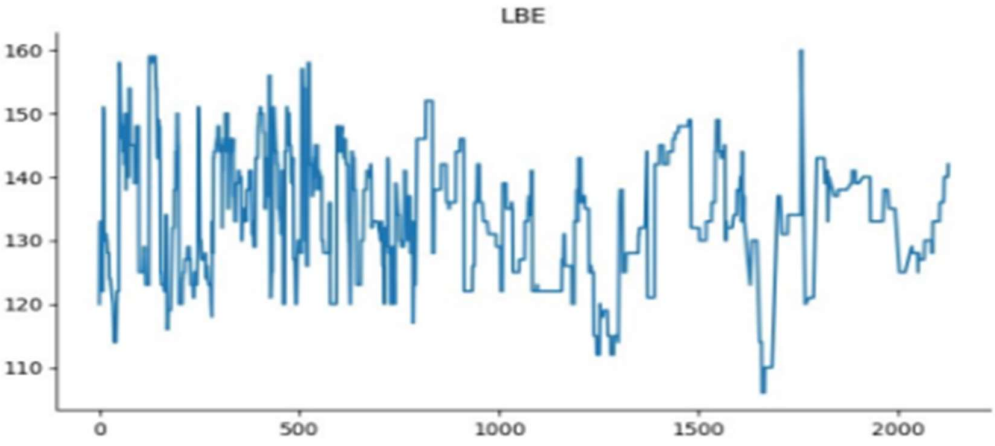
```
from scipy.stats import mode

ensemble_preds = np.array([lr1_pred, dt1_pred, rf1_pred]).T
final_preds = mode(ensemble_preds, axis=1).mode.flatten()

from sklearn.metrics import accuracy_score
ensemble_accuracy = accuracy_score(final_preds, y_test)
print(f"Accuracy of the merged algorithm: {ensemble_accuracy:.2f}")
```

```
Accuracy of the merged algorithm: 1.00
```

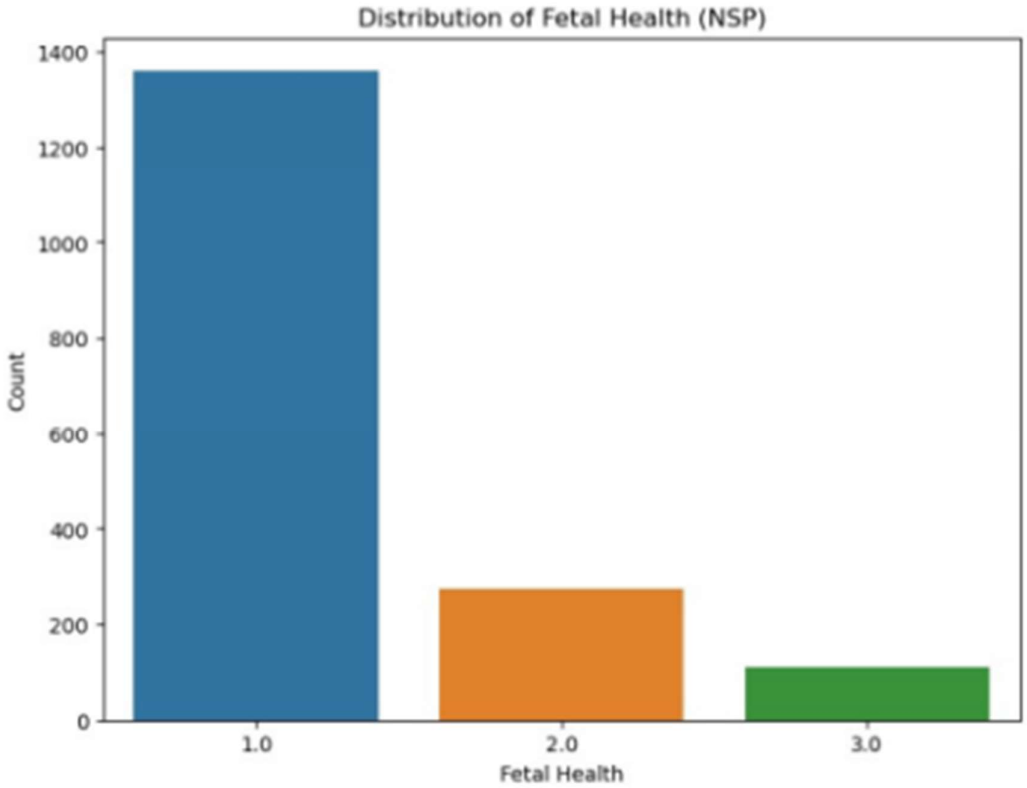
5.3.1 Home Page




```
df['LBE'].plot(kind='line', figsize=(8, 4), title='LBE')
plt.gca().spines[['top', 'right']].set_visible(False)
```



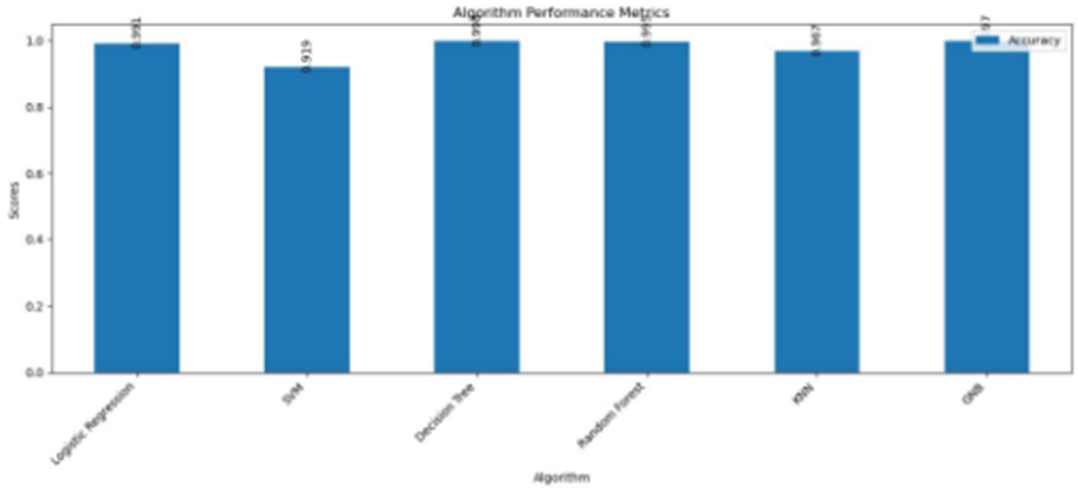
5.3.2 Graphs Page




```
df['NSP'].value_counts()

NSP
1.0    1361
2.0     273
3.0     111
Name: count, dtype: int64

from imblearn.over_sampling import SMOTE

df1=df

df[(df['NSP']==1.0) | (df['NSP']==2.0)]

           b       e    LBE     LB   AC   FM   UC  ASTV  MSTV
ALTV  ...    C  \
0      240.0   357.0  120.0  120.0  0.0  0.0  0.0  73.0   0.5
43.0  ...  0.0
1        5.0   632.0  132.0  132.0  4.0  0.0  4.0  17.0   2.1
0.0   ...  0.0
2      177.0   779.0  133.0  133.0  2.0  0.0  5.0  16.0   2.1
```

5.3.2 Prediction Page

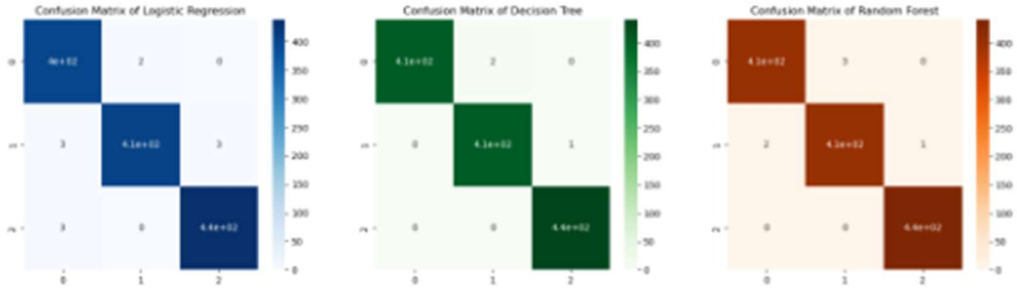
```
                ha='center', va='center', xytext=(0, 10),
textcoords='offset points', rotation=90)

plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



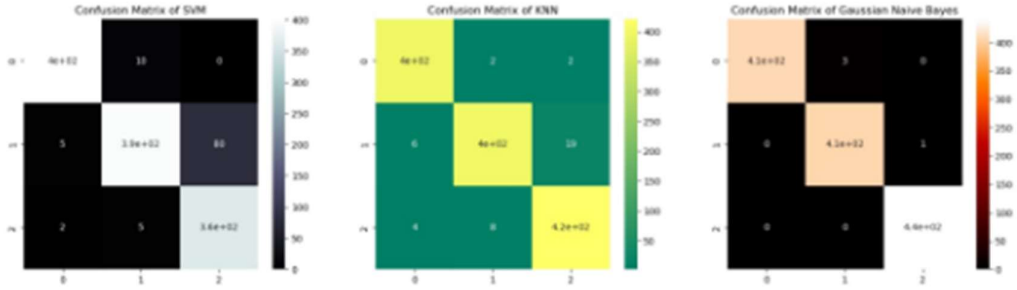

```
print("Confusion Matrix of Logistic Regression ")
lr1_cm=pd.DataFrame(confusion_matrix(lr1_pred,y_test))
print(lr1_cm)
print("Confusion Matrix of Decision Tree ")
dt1_cm=pd.DataFrame(confusion_matrix(dt1_pred,y_test))
print(dt1_cm)
print("Confusion Matrix of Random Forest ")
rf1_cm=pd.DataFrame(confusion_matrix(rf1_pred,y_test))
print(rf1_cm)
print("Confusion Matrix of SVM ")
svm1_cm=pd.DataFrame(confusion_matrix(svm1_pred,y_test))
print(svm1_cm)
print("Confusion Matrix of KNN ")
KNN1_cm=pd.DataFrame(confusion_matrix(KNN1_pred,y_test))
print(KNN1_cm)
print("Confusion Matrix of GNB ")
GNB1_cm=pd.DataFrame(confusion_matrix(GNB1_pred,y_test))
print(GNB1_cm)
```

```
Confusion Matrix of Logistic Regression
     0    1    2
0  402    2    0
1    3  407    3
2    3    0  439
```

5.3.3 Analysis Page




```
plt.figure(figsize=(20,5))

plt.subplot(1,3,1)
sns.heatmap(confusion_matrix(svm1_pred,y_test),annot=True,cmap='bone')
plt.title("Confusion Matrix of SVM")

plt.subplot(1,3,2)
sns.heatmap(confusion_matrix(KNN1_pred,y_test),annot=True,cmap='summer
')
plt.title("Confusion Matrix of KNN")

plt.subplot(1,3,3)
sns.heatmap(confusion_matrix(GNB1_pred,y_test),annot=True,cmap='gist_h
eat')
plt.title("Confusion Matrix of Gaussian Naive Bayes")

Text(0.5, 1.0, 'Confusion Matrix of Gaussian Naive Bayes')
```




```
time_df={"Algorithm":['Logistic Regression',"SVM",'Decision
Tree','Random Forest','KNN','GNB'],
         "Execution Time in Seconds":[lrt,svmt,dtt,rft,KNNt,GNBt]}
time df=pd.DataFrame(time_df)
time_df
```

	Algorithm	Execution Time in Seconds
0	Logistic Regression	0.206934
1	SVM	0.383203

5.3.4 Metrics Page

```
2        Decision Tree                    0.102481
3        Random Forest                    0.087109
4                  KNN                    0.010878
5                  GNB                    0.016667
```

```
plt.figure(figsize=(10,5))
sns.barplot(data=time_df,x=time_df['Algorithm'],y=time_df['Execution
Time in Seconds'])
plt.xlabel("Algorithms")
plt.ylabel("Execution Time in Seconds")
plt.title("Time Taken For Model Training")
```

```
Text(0.5, 1.0, 'Time Taken For Model Training')
```


```
#Testing
```

```
df1
```

	b	e	LBE	LB	AC	FM	UC	ASTV	MSTV	ALTV	...	C	\
0	240.0	357.0	120.0	120.0	0.0	0.0	0.0	73.0	0.5	43.0	...	0.0	
1	5.0	632.0	132.0	132.0	4.0	0.0	4.0	17.0	2.1	0.0	...	0.0	
2	177.0	779.0	133.0	133.0	2.0	0.0	5.0	16.0	2.1	0.0	...	0.0	
4	533.0	1147.0	132.0	132.0	4.0	0.0	5.0	16.0	2.4	0.0	...	0.0	
7	62.0	679.0	122.0	122.0	0.0	0.0	0.0	83.0	0.5				

5.3.5 Graphs Page

4. CONCLUSION

The **fetal health classification system** using **machine learning** has demonstrated its ability to accurately classify fetal conditions into **Normal, Suspicious, and Pathological** categories. By utilizing **Cardiotocography (CTG) data**, various machine learning models, including **Decision Tree, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Logistic Regression, and Gaussian Naïve Bayes (GNB)**, were tested for performance. Among these, the **Decision Tree classifier achieved the highest accuracy of 99.76%**, proving to be the most effective model for fetal health prediction.

The system effectively addresses the challenges of **manual CTG interpretation**, which is often subjective and prone to human error. By automating the classification process, this machine learning approach can assist **healthcare professionals** in making timely and informed decisions, reducing the risk of fetal complications. Additionally, **data preprocessing techniques, feature selection, and performance evaluation** ensured the robustness of the model.

Future improvements include integrating **deep learning techniques** for enhanced prediction accuracy, developing a **real-time monitoring system**, and deploying the model in **clinical settings**. This AI-powered tool has the potential to revolutionize **prenatal care**, improve **maternal and fetal health outcomes**, and contribute to **early risk detection and medical intervention**.

FUTURE WORK

Although the **fetal health classification system** has achieved high accuracy, several improvements can enhance its **performance, usability, and real-world applicability**. Future work includes integrating **deep learning models** such as **Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs)** to capture complex patterns in **Cardiotocography (CTG) data**, potentially improving classification accuracy beyond **99.76%**.

Another crucial area is the **real-time implementation** of this system in **hospitals and maternity clinics**. By integrating the model with **Internet of Things (IoT) devices**, continuous fetal monitoring can be automated, allowing real-time alerts for healthcare providers. Additionally, **cloud-based deployment** will enable remote access to fetal health predictions, benefiting patients in **rural and underserved areas**.

Further enhancements include **expanding the dataset** with more diverse fetal health records to improve the model's **generalization**. Moreover, incorporating **explainable AI (XAI) techniques** will provide medical professionals with insights into why a specific classification is made, increasing trust and interpretability.

Lastly, developing a **user-friendly mobile or web-based application** will enhance accessibility, allowing doctors and pregnant women to monitor fetal health easily. These advancements will make the system more **scalable, reliable, and impactful** in ensuring **better maternal and fetal health outcomes**.

REFERENCES

[1] G. Sedgh, S. Singh, and R. Hussain, ‘‘Intended and unintended pregnancies worldwide in 2012 and recent trends,’’ Stud. Family Planning, vol. 45, no. 3, pp. 301–314, Sep. 2014.

[2] C. J. Murray, ‘‘Global, regional, and national age-sex specific all-cause and cause-specific mortality for 240 causes of death, 1990–2013: A systematic analysis for the global burden of disease study 2013,’’ Lancet, vol. 385, no. 9963, 1990, Art. no. 117171.

[3] (2016). World Health Organization, Maternal Mortality: Fact Sheet. [Online]. Available: http://www.who.int/mediacentre/factsheets/fs348/en/

[4] National Institutes of Health, What Are Some Common Complications of Pregnancy?U.S. Department of Health and Human Services. [Online]. Available: https://www.nichd.nih.gov/health/topics/pregnancy/ conditioninfo/Pages/complications.aspx

[5] Office on Women’s Health, Pregnancy ComplicationsUS Department of Health and Human Services. [Online]. Available: https://www. womenshealth.gov/pregnancy/your-pregnant-now

[6] G. Magenes, R. Bellazzi, A. Malovini, and M. G. Signorini, ‘‘Comparison of data mining techniques applied to fetal heart rate parameters for the early identification of IUGR fetuses,’’ in Proc. 38th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC), Orlando, FL, USA, Aug. 2016, pp. 916–919, doi: 10.1109/EMBC.2016.7590850.

[7] N. M. Fisk and R. P. Smit, ‘‘Fetal growth restriction; small for gestational age,’’ in Turnbull’s Obstetrics, 3rd ed., G. Chamberlain and P. Steer, Eds. Edinburgh, Scotland: Churchill Livingstone, 2001, pp. 197–209.

[8] S. C. R. Nandipati, ‘‘Classification and feature selection approaches for cardiotocography by machine learning techniques,’’ J. Telecommun., Electron. Comput. Eng., vol. 12, no. 1, pp. 7–14, 2020.

[9] S. Das, K. Roy, and C. K. Saha, ‘‘Establishment of automated technique of FHR baseline and variability detection using CTG: Statistical comparison with expert’s analysis,’’ Int. J. Inf. Eng. Electron. Bus., vol. 11, no. 1, pp. 27–35, Jan. 2019, doi: 10.5815/ijieeb.2019.01.04.

[10] W. W. Stead, ‘‘Clinical implications and challenges of artificial intelligence and deep learning,’’ J. Amer. Med. Assoc., vol. 320, no. 11, p. 1107, Sep. 2018, doi: 10.1001/jama.2018.11029.

[11] K. H. Miao, J. H. Miao, and G. J. Miao, ‘‘Diagnosing coronary heart disease using ensemble machine learning,’’ Int. J. Adv. Comput. Sci. Appl., vol. 7, no. 10, p. 3039, 2016.

[12] M. G. Signorini, N. Pini, A. Malovini, R. Bellazzi, and G. Magenes, ‘‘Integrating machine learning techniques and physiology based heart rate features for antepartum fetal monitoring,’’ Comput. Methods Programs Biomed., vol. 185, Mar. 2020, Art. no. 105015.

[13] J. H. Miao, K. H. Miao, and G. J. Miao, ‘‘Breast cancer biopsy predictions based on mammographic diagnosis using support vector machine learning, Multidisciplinary Journals in Science and Technology,’’ J. Sel. Areas Bioinf., vol. 5, no. 4, p. 19, 2015.

[14] M. T. Alam, M. A. I. Khan, N. N. Dola, T. Tazin, M. M. Khan, A. A. Albraikan, and F. A. Almalki, ‘‘Comparative analysis of different efficient machine learning methods for fetal health classification,’’ Appl. Bionics Biomechanics, vol. 2022, pp. 1–12, Apr. 2022.

[15] A. Mehbodniya, A. J. P. Lazar, J. Webber, D. K. Sharma, S. Jayagopalan, K. K, P. Singh, R. Rajan, S. Pandya, and S. Sengan, ‘‘Fetal health classification from cardiotocographic data using machine learning,’’ Expert Syst., vol. 39, no. 6, Jul. 2022, Art. no. e12899.

[16] A. Kuzu and Y. Santur, ‘‘Early diagnosis and classification of fetal health status from a fetal cardiotocography dataset using ensemble learning,’’ Diagnostics, vol. 13, no. 15, p. 2471, Jul. 2023.

[17] P. Fergus, A. Hussain, D. Al-Jumeily, D.-S. Huang, and N. Bouguila, ‘‘Classification of caesarean section and normal vaginal deliveries using foetal heart rate signals and advanced machine learning algorithms,’’ Biomed. Eng. OnLine, vol. 16, no. 1, pp. 1–26, Dec. 2017.

[18] E. J. Quilligan and R. H. Paul, ‘‘Fetal monitoring: Is it worth it?’’ Obstetrics Gynecol., vol. 45, no. 1, pp. 96–100, 1975.

[19] F. G. Esposito, ‘‘Fetal heart rate monitoring and neonatal outcome in a population of earlyand lateonset intrauterine growth restriction,’’ J. Obstetrics Gynaecology Res., vol. 45, no. 7, pp. 1343–1351, 2019.

[20] A. Ugwumadu, ‘‘Are we (MIS) guided by current guidelines on intrapartum fetal heart rate monitoring? Case for a more physiological approach to interpretation,’’ BJOG, Int. J. Obstetrics Gynaecol., vol. 121, no. 9, pp. 1063–1070, 2014

COLLEGE VISION & MISSION

VISION

To be one of the Nation's premier Engineering Colleges by achieving the highest order of excellence in Teaching and Research.

MISSION

Through multidimensional excellence, we value intellectual curiosity, pursuit of knowledge building and dissemination, academic freedom and integrity to enable the students to realize their potential. We promote technical mastery of Progressive Technologies, understanding their ramifications in the future society and nurture the next generation of skilled professionals to compete in an increasingly complex world, which requires practical and critical understanding of all aspects.

DEPARTMENT VISION & MISSION

VISION

- To become a nationally recognized quality education center in the domain of Computer Science and Information Technology through teaching, training, learning, research and consultancy.

MISSION

- The Department offers undergraduate program in Information Technology and Post graduate program in Software Engineering to produce high quality information technologists and software engineers by disseminating knowledge throughcontemporary curriculum, competent faculty and adopting effective teaching-learning methodologies.
- Igniting passion among students for research and innovation by exposing them to real time systems and problems.
- Developing technical and life skills in diverse community of students with modern training methods to solve problems in Software Industry.
- Inculcating values to practice engineering in adherence to code of ethics in multicultural and multi discipline teams.

PROGRAM EDUCATIONAL OBJECTIVES (PEO'S)

After few years of graduation, the graduates of B.Tech (CSSE) will:

1. Enrolled or completed higher education in the core or allied areas of Computer Science and Information Technology or management.
2. Successful entrepreneurial or technical career in the core or allied areas of Computer Science and Information Technology.
3. Continued to learn and to adapt to the world of constantly evolving technologies in the core or allied areas of Computer Science and Information Technology.

PROGRAM SPECIFIC OUTCOMES (PSO'S)

On successful completion of the Program, the graduates of B. Tech (CSSE) program willbe able to:

PSO1 Design and develop database systems, apply data analytics techniques, and use advanced databases for data storage, processing and retrieval.

PSO2 Apply network security techniques and tools for the development of highly secure systems.

PSO3 Analyze, design and develop efficient algorithms and software applications to deploy in secure environment to support contemporary services using programming languages, tools and technologies.

PSO4 Apply concepts of computer vision and artificial intelligent for the development of efficient intelligent systems and applications

COURSE OUTCOMES (CO'S)

1. Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems (**Engineering knowledge**).
2. Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences (**Problem analysis**).
3. 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 considerations (**Design/development of solutions**).
4. 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 (**Conduct investigations of complex problems**).
5. Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations (**Modern tool usage**)
6. 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 (**The engineer and society**)
7. Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development (**Environment and sustainability**).
8. Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice (**Ethics**).
9. Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings (**Individual and team work**).
10. Communicate effectively on complex engineering activities with the engineering community and with 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 (**Communication**).

11. 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 (**Project management and finance**).
12. 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 (**Life-long learning**).

COURSE OUTCOMES (CO'S)

After successful completion of this course, the students will be able to:

CO1 Create/Design algorithms and software to solve complex Computer Science, Information Technology and allied problems using appropriate tools and techniques following relevant standards, codes, policies, regulations and latest developments.

CO2 Consider society, health, safety, environment, sustainability, economics and project management in solving complex Computer Science, Information Technology and allied problems.

CO3 Perform individually or in a team besides communicating effectively in written, oral and graphical forms on Computer Science, and Information Technology based systems or processes.

Mapping of Course Outcomes with COs and PSOs:

Course Outcomes	Program Outcomes												Program Specific Outcomes		
	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
CO1	3	3	3	3	3	-	-	3	-	-	-	3	3	3	3
CO2	-	-	-	-	-	3	3	-	-	-	3	-	3	3	3
CO3	-	-	-	-	-	-	-	-	3	3	-	-	3	3	3
Average	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Level of correlation of the course	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

CO1 Create/Design algorithms and software to solve complex Computer Science, Information Technology and allied problems using appropriate tools and techniques following relevant standards, codes, policies, regulations and latest developments.

CO2 Consider society, health, safety, environment, sustainability, economics and project management in solving complex Computer Science, Information Technology and allied problems.

CO3 Perform individually or in a team besides communicating effectively in written, oral and graphical forms on Computer Science, and Information Technology based systems or processes.

Mapping of Course Outcomes with COs and PSOs:

Course Outcomes	Program Outcomes												Program Specific Outcomes		
	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
CO1	3	3	3	3	3	-	-	3	-	-	-	3	3	3	3
CO2	-	-	-	-	-	3	3	-	-	-	3	-	3	3	3
CO3	-	-	-	-	-	-	-	-	3	3	-	-	3	3	3
Average	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Level of correlation of the course	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3