CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Maternal health is a critical area of public health that focuses on the health and well-being of women during pregnancy, childbirth, and the postpartum period. Despite significant advances in maternal healthcare over the past few decades, maternal mortality and morbidity remain major global health challenges. According to the World Health Organization (WHO), an estimated 830 women die every day from preventable causes related to pregnancy and childbirth. In addition, many women experience complications during pregnancy and childbirth that can have long term health consequences. To address these challenges, there is a growing interest in using Machine Learning (ML) to predict maternal health risks and improve healthcare outcomes for pregnant women. ML is a branch of artificial intelligence that involves training computer algorithms to learn patterns from large datasets and make predictions based on new data. ML can be used to analyze electronic health records (EHRs) and other sources of healthcare data to identify women at high risk for maternal complications such as pre-eclampsia, gestational diabetes, and postpartum hemorrhage. The aim of this project is to develop an ML model for maternal health risk prediction that can aid healthcare providers in identifying women who are at high risk of maternal complications. The model will be trained on a large dataset of EHRs from pregnant women and will use various ML algorithms to identify patterns and make predictions. The model will be validated using an independent dataset, and its performance will be evaluated using metrics such as accuracy, precision, recall, and F1 score.

Figure1.1

The potential impact of this project is significant. By accurately predicting maternal health risks, healthcare providers can intervene early and provide appropriate care to prevent or manage complications, potentially reducing maternal mortality and morbidity. Furthermore, the use of ML in maternal healthcare can help to improve the efficiency and effectiveness of healthcare delivery, leading to better outcomes for pregnant women and their families. The project will require collaboration between data scientists and healthcare providers, as well as an understanding of the clinical context and the potential impact on patient outcomes. The project has the potential to contribute to the reduction of maternal mortality and morbidity, leading to better health outcomes for pregnant women and their families.

1.2 INTRODUCTION TO MACHINE LEARNING

Machine learning is a subfield of artificial intelligence (AI) that involves the development of algorithms and statistical models that enable computers to automatically improve their performance on a specific task over time without being explicitly programmed. It involves the use of statistical techniques to analyze and extract patterns from data, which can then be used to make predictions or decisions based on new data.

1.2.1 TYPES OF MACHINE LEARNING

There are several types of machine learning, including supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, transfer learning and deep learning.

Supervised Learning: In supervised learning, a machine learning model is trained on a labeled dataset, where the desired output is already known. The model learns to predict the output given the input data by analyzing the relationship between the input and output variables. The goal of supervised learning is to generalize the relationship between the input and output variables so that the model can predict the output for new, unseen input data.

Unsupervised Learning: In unsupervised learning, a machine learning model is trained on an unlabeled dataset, where the desired output is not known. The model analyzes the input data and identifies patterns or relationships with in the data. The goal of unsupervised learning is to discover hidden patterns or structures in the data.

Semi-Supervised Learning: In semi-supervised learning, a machine learning model is trained on a partially labeled dataset. The model uses the labeled data to learn the relationship between the input and output variables and then uses the unlabeled data to generalize the relationship.

Reinforcement Learning: In reinforcement learning, a machine learning model learns from feedback it receives as it interacts with its environment. The model receives a reward or penalty based on its actions and learns to optimize its behavior to maximize its reward.

1.2.2 MACHINE LEARNING ALGORITHMS

There are many machine learning algorithms, and each has its own strengths and weaknesses. Here is an over view of some commonly used algorithms:

Linear Regression: Linear regression is a supervised learning algorithm used for predicting a continuous output variable. It models the relationship between the input variables and the output variable using a linear equation. Linear regression as sum of that there is a linear relationship between the input and output variables and that the errors in the prediction are normally distributed.

Decision Trees: Decision trees are a supervised learning algorithm used for classification and regression tasks. They partition the input data into smaller subsets based on the values of the input variables and create a tree-like model ofdecisionsandtheirpossibleconsequences.Eachdecisionnodeinthetreerepresents a test of a particular input variable, and each leaf node represents a predicted value for the output variable.

Random Forest: Random Forest is an ensemble learning algorithm that uses multiple decision trees to improve the accuracy of the model. It randomly selects subsets of the input data and subsets of the input variables to build multiple decision trees, which are then combined to make a final prediction. Random forest is less prone to over fitting than decision trees and is often used for classification and regression tasks.

Support Vector Machines (SVM): SVM is a supervised learning algorithm used for classification and regression tasks. It finds the best hyper

plane that separates the input data into different classes or predicts the continuous output variable. SVM works by maximizing the margin between the hyper plane and the data points closest to it.

Naive Bayes: Naive Bayes is a supervised learning algorithm used for classification tasks. It assumes that the input variables are independent of each other and calculates the probability of the output variable given the input variables using Bayes' theorem. Naïve Bayes is often used for text classification tasks.

K-Nearest Neighbors (KNN): KNN is a supervised learning algorithm used for classification and regression tasks. It finds the k nearest data points to a given input data point and predicts the output variable based on the majority class or average value of the k nearest neighbors. KNN is a non-parametric algorithm, meaning it does not make any assumptions about the distribution of the input data.

Neural Networks: Neural networks area class of machine learning algorithms inspired by the structure and function of the human brain. They consist of interconnected nodes or neurons that process input data and produce output data. Neural networks can be used for classification, regression, and other tasks and they are particularly effective at learning complex patterns in data.

Gradient Boosting: Gradient boosting is an ensemble learning algorithm that combines multiple weak learners to create a strong learner. It works by iteratively adding new models that correct the errors of the previous models, with each new model focusing on the examples that were misclassified by the previous models.

These are just a few important machine learning algorithms. The choice of algorithm depends on the nature of the problem, the type of data, and the desired outcome.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

This literature survey aims to provide a comprehensive overview of the current methods used to predict the presence of the maternal health risk. The survey will review various studies that have explored this topic and analyze the different approaches and techniques used in these studies.

2.2 RELATED WORKS

1. Prediction of maternal health risk with Traditional Machine learning methods

Author: Hursit Burak MUTLU, Fatih DURMAZ, Emine CENGIL, Muhammed YILDIRIM

Year: 2023

In risky pregnancy, various diseases such as heart, lung, kidney, high blood pressure, diabetes and liver that pregnant women have before may aggravate the expectant mothers condition during pregnancy. By analyzing medical parameters such as maternal age, heart rate, blood oxygen level, blood pressure, body temperature, and the values corresponding to these parameters, information on risk intensity can be estimated for some patients. It is possible to reduce such pregnancy-related complications by classifying risk factors early in symptoms. It is possible to benefit from machine learning methods in determining maternal risk health. Therefore, in this study, six different machine learning methods were used to determine maternal risk health. The results obtained in these methods were compared with each other and it was observed that the most successful method in estimating maternal risk health was Decision Tree. The accuracy value obtained in the Decision Tree method was 89.16%. The lowest accuracy rate among the methods used in the paper was obtained in the k-nearest neighbors (KNN) method with 68.47%.

2. Ensemble learning- based feature Engireering to analyse Maternal health during pregnancy and health risk prediction

Author: Ali Raza, Hafeez Ur Rehman Siddiqui, Kashif Munir, Mubarak Amutair, Furuqan Rustam, Imran Asharf

Year: 2020

Maternal health is an important aspect of women's health during pregnancy, childbirth, and the postpartum period. Specifically, during pregnancy, different health factors like age, blood disorders, heart rate, etc. can lead to pregnancy complications. Detecting such health factors can alleviate the risk of pregnancy-related complications. This study aims to develop an artificial neural network-based system for predicting maternal health risks using health data records. A novel deep neural network architecture, DT-BiLTCN is proposed that uses decision trees, a bidirectional long short-term memory network, and a temporal convolutional network. Experiments involve using a dataset of 1218 samples collected from maternal health care, hospitals, and community clinics using the IoT-based risk monitoring system. Class imbalance is resolved using the synthetic minority oversampling technique. DT-BiLTCN provides a feature set to obtain high accuracy results which in this case are provided by the support vector machine with a 98% accuracy. Maternal health exploratory data analysis reveals that the health conditions which are the strongest indications of health risk during pregnancy are diastolic and systolic blood pressure, heart rate, and age of pregnant women. Using the proposed model, timely prediction of health risks associated with pregnant women can be made thus mitigating the risk of health complications which helps to save lives.

3. Prediction of maternal complications in pregnancy: a systematic review and critical analysis of models and variable selection

Author: Alsayyed et al

Year: 2021

This study reviewed the literature on the use of ML for predicting maternal complications in pregnancy. The authors found that ML models can accurately predict maternal complications, but variable selection and model performance vary widely across studies. The study reviewed 22 studies on the use of ML for maternal complication prediction in pregnancy. The authors noted the wide variability in model performance and variable selection across studies, highlighting the need for standardization and validation. The study highlighted the potential of ML to aid in maternal complication prediction and the importance of integrating ML models into clinical decision-making. The accuracy varied across the 22 studies reviewed in the paper. Various ML algorithms were used across the 22 studies reviewed in the paper, including decision trees, logistic regression, random forests, artificial neural networks, and support vector machines. The authors noted the wide variability in model performance and variable selection across studies, highlighting the need for standardization and validation. The study highlighted the potential of ML to aid in maternal complication prediction and the importance of integrating ML models into clinical decision-making.

4. Predictive modeling of Maternal-fetal medicine data: a machine learning approach

Author: Strydom et al

Year: 2021

This study developed an ML model for predicting maternal and fetal

complications using data from over 3,000 women. The model achieved an accuracy of 85.5% and demonstrated the potential for ML to aid in predicting maternal and fetal complications. The study used decision tree, random forest, and neural network algorithms to predict maternal and fetal complications. The authors noted that the use of ML for maternal and fetal complication prediction can aid in early identification 4 and intervention, potentially improving healthcare outcomes. The study highlighted the importance of careful feature selection and model optimization for accurate and reliable predictions. Accuracy: 85.5%. The study used decision tree, random forest, and neural network algorithms to predict maternal and fetal complications. The authors noted that the use of ML for maternal and fetal complication prediction can aid in early identification and intervention, potentially improving healthcare outcomes. The study highlighted the importance of careful feature selection and model optimization for accurate and reliable predictions. Neural network, Random Forest, Decision tree is the algorithms used.

5. Machine learning models for predicting Maternal and neonatal Morbidity and Mortality in low-resource settings: a systematic review

Author: Chamieh et al

Year: 2020

This study reviewed the literature on the use of ML for predicting maternal and neonatal morbidity and mortality in low-resource settings. The authors found that ML models can predict maternal and neonatal morbidity and mortality with high accuracy and can potentially improve healthcare outcomes in low-resource settings. The study reviewed 14 studies on the use of ML for maternal and neonatal morbidity and mortality prediction in low-resource settings. The authors noted that ML models can accurately predict maternal and neonatal morbidity and mortality, but noted the need for further validation and optimization of the models. The study highlighted the potential of ML to

improve healthcare outcomes in low-resource settings and the importance of data quality and standardization. The accuracy varied across the 14 studies reviewed in the paper. The study reviewed 14 studies on the use of ML for maternal and neonatal morbidity and mortality prediction in low- resource settings. The authors noted that ML models can accurately predict maternal and neonatal morbidity and mortality, but noted the need for further validation and optimization of the models. Various ML algorithms were used across the 14 studies reviewed in the paper, including decision trees, random forests, neural networks.

6. Machine learning approach for predicting adverse Maternal outcomes in a low-resource setting

Author: Jiwani et al

Year: 2019

This study developed an ML model for predicting adverse maternal outcomes in a low-resource setting. The model was trained on data from over 22,000 women and achieved an accuracy of 78.2%. The authors concluded that ML can be used to predict adverse maternal outcomes in low-resource settings. The study used decision tree, logistic regression, and random forest algorithms to predict adverse maternal outcomes. The authors noted that the use of ML for maternal health risk prediction can help healthcare providers identify high-risk women and prioritize care accordingly. The study also highlighted the need for ongoing model validation and monitoring to ensure accurate and reliable predictions. Accuracy: 78.2%. The study used decision tree, logistic regression, and random forest algorithms to predict adverse maternal outcomes. The authors noted that the use of ML for maternal health risk prediction can help healthcare providers identify high-risk women and prioritize care accordingly. The study also highlighted the need for ongoing model validation and monitoring to ensure accurate and reliable predictions. Random forest, Logistic regression, Decision tree are the algorithms used.

7. Predicting maternal mortality in low-resource settings

Author: Sheth et al

Year: 2016

This study developed an ML model for predicting maternal mortality in low-resource settings. The model was trained on data from over 6,000 women and achieved an accuracy of 81.6%. The authors concluded that ML can be a valuable tool for predicting maternal mortality in low- resource settings. The study used decision tree and logistic regression algorithms to predict maternal mortality. The authors noted that the use of ML for maternal mortality prediction can be a valuable tool in low- resource settings where resources are limited. The study highlighted the importance of collecting high-quality data for ML model development. Accuracy: 81.6%. The study used decision tree and logistic regression algorithms to predict maternal mortality. The authors noted that the use of ML for maternal mortality prediction can be a valuable tool in low- resource settings where resources are limited. The study highlighted the importance of collecting high-quality data for ML model development.

8. Fetal health status prediction based on Maternal clinical history using Machine learning techniques

Author: Akhan Akbulut, Egemen Ertugrul , Varol Topcu

Year: 2018

Congenital anomalies are seen at 1–3% of the population, probabilities of which are tried to be found out primarily through double, triple and quad tests during pregnancy. Also, ultrasonographical evaluations of fetuses enhance detecting and defining these abnormalities. About 60–70% of the anomalies can be diagnosed via ultrasonography, while the remaining 30–40% can be diagnosed after childbirth. Medical diagnosis and prediction is a topic that is closely related with e-Health and machine learning. e-Health applications are critically important especially for the patients unable to see a doctor or any

health professional. Our objective is to help clinicians and families to better predict fetal congenital anomalies besides the traditional pregnancy tests using machine learning techniques and e-Health applications.In this work, we developed a prediction system with assistive e-Health applications which both the pregnant women and practitioners can make use of. A performance comparison (considering Accuracy, F1-Score, AUC measures) was made between 9 binary classification models (Averaged Perceptron, Boosted Decision Tree, Bayes Point Machine, Decision Forest, Decision Jungle, Locally-Deep Support Vector Machine, Logistic Regression, Neural Network, Support Vector Machine) which were trained with the clinical dataset of 96 pregnant women and used to process data to predict fetal anomaly status based on the maternal and clinical data. The dataset was obtained through maternal questionnaire and detailed evaluations of 3 clinicians from RadyoEmar radio diagnostics center in Istanbul, Turkey. Our e-Health applications are used to get pregnant women's health status and clinical history parameters as inputs, recommend them physical activities to perform during pregnancy, and inform the practitioners and finally the patients about possible risks of fetal anomalies as the output.

9. A Robust learning predictive model for Maternal health risk

Author: Lokesh Pawar, Janvi Malhotra, Astha Sharma, Diya Arora

Year: 2022

Maternal health refers to the physical, mental, and emotional well-being of mothers throughout all stages of pregnancy, birth, and the postpartum period. Morbidity and death rates during pregnancy are significant health statistics because they provide information about the accessibility of maternal and other medical resources. The main reason of maternal mortality is hemorrhage, high blood pressure, early labor, and abortions that are performed under dangerous conditions. Machine learning algorithms play a significant role in determining maternity health risks. In this paper, Traditional Machine Learning algorithms are applied for maternal health risk prediction. Model performance has potential

for improvement; thus, a more robust and dependable machine learning model has been proposed to operate even in worst, average, and best scenarios and provide the robust performance by considering the performance of all scenarios. The proposed robust model turns out to be the most efficient robust model among all with an accuracy of 70.21%, which is quite satisfactory when compared to traditionally applied ML Models.

10. Fetal health state monitoring using decision tree classifier from cardiotocography measurements

Author: Ramla M, Sangeetha S, Nickolas S

Year: 2018

Prenatal Mortality is an alarming issue in the world which needs immediate attention for the sustainable growth of nation. To address this issue, both intrapartum and ante partum fetal health state monitoring is essential. Fetal and maternal risk can be assessed by monitoring the fetal heart rate. Cardiotocography records the fetal heart rate and uterine contractions of mother simultaneously. This signal data can support the physicians to understand the fetal health status. In this paper, an efficient method to predict the high-risk pregnancy based on the fetal health status using CART is proposed. A CTG datasets having 2126 recordings from UCI machine learning repository is used for classification. 5- fold cross validation is done, and the proposed methodology using CART was quantified using precision, recall and F-score.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing system for maternal health risk prediction involves various clinical and non-clinical risk assessment tools that healthcare providers use to identify women at increased risk of complications during pregnancy, childbirth, and postpartum. These tools may include maternal demographic information, medical history, current pregnancy status, and various clinical tests and measurements such as blood pressure, blood sugar levels, and ultrasound scans. Based on this information, healthcare providers may offer various interventions such as more frequent monitoring, medications, and referral to specialized care if needed. Clinical guidelines for maternal health risk assessment and management are often developed by professional organizations such as the American College of Obstetricians and Gynecologists (ACOG) and the World Health Organization (WHO).

Decision tree, LightGBM Classifier, Random Forest and KNN classifier were used to classify the data in the dataset consisting of five different features obtained to assess maternal health risk.

MODELS	ACCURACY(%)
Decision Tree	89.16%
LightGBM Classifier	84.24%
Random Forest	81.28%
KNN Classifier	68.47%

3.1.1 INPUT PARAMETERS

- Age
- SystolicBP
- DiastolicBP
- BS
- Body Temp
- Heart Rate

3.1.2 DISADVANTAGES OF THE EXISTING SYSTEM

1. The risk assessment tools may not capture all relevant risk factors, including social determinants of health, environmental factors, and patient preferences.

2. There can be variation in the interpretation and application of clinical guidelines by health care providers.

3. The current system does not always provide personalized risk estimates for individual patients, which can lead to underestimation or over estimation of risk.

3.2 PROPOSED SYSTEM

The proposed system for maternal health risk prediction using XGBoost ML algorithm aims to develop a predictive model that can accurately identify women at increased risk of maternal complications during pregnancy, childbirth, and postpartum. XGBoost is a powerful ML algorithm that can handle large and complex datasets, making it well-suited for this task. The

system will collect data on various demographic, clinical, and social determinants of health factors, including patient demographics, medical history, clinical tests and measurements, and social determinants of health factors such as income, education, and access to healthcare. The data will be preprocessed to clean and transform it into a format that can be used by the XGBoost algorithm. The XGBoost algorithm will be trained on the preprocessed data to identify patterns and risk factors associated with maternal complications. The algorithm will be tuned to optimize its performance and minimize the risk of overfitting. To evaluate the performance of the model, the dataset will be split into training and testing sets. The model will be trained on the training set and then tested on the testing set to assess its accuracy in predicting maternal complications. Various performance metrics, such as accuracy, sensitivity, and specificity, will be used to evaluate the performance of the model. Once the model is developed and validated, it can be integrated into clinical practice to provide more accurate and personalized risk estimates for individual patients. Healthcare providers can input patient information into the model to generate a risk score or classification, which can help guide decisions about maternal health care. The model can also be used to identify high-risk pregnancies and offer appropriate interventions, such as more frequent monitoring, medications, or referral to specialized care if needed.

3.2.1 XGBOOST ALGORITHM

XGBoost, or extreme Gradient Boosting, is a machine learning algorithm under ensemble learning. It is trendy for supervised learning tasks, such as regression and classification. XGBoost builds a predictive model by combining the predictions of multiple individual models, often decision trees, in an iterative manner. The algorithm works by sequentially adding weak learners to the ensemble, with each new learner focusing on correcting the errors made by the existing ones. It uses a gradient descent optimization technique to minimize a predefined loss function during training.

XGBoost's architecture centers around the principles of gradient boosting, utilizing decision trees as base learners. It iteratively improves model accuracy by sequentially adding trees to correct errors made by the ensemble. Key components include an objective function combining a loss function and regularization term, gradient boosting to minimize this function, and optimization through gradient descent. XGBoost's efficiency is further enhanced by parallel and distributed computing capabilities.

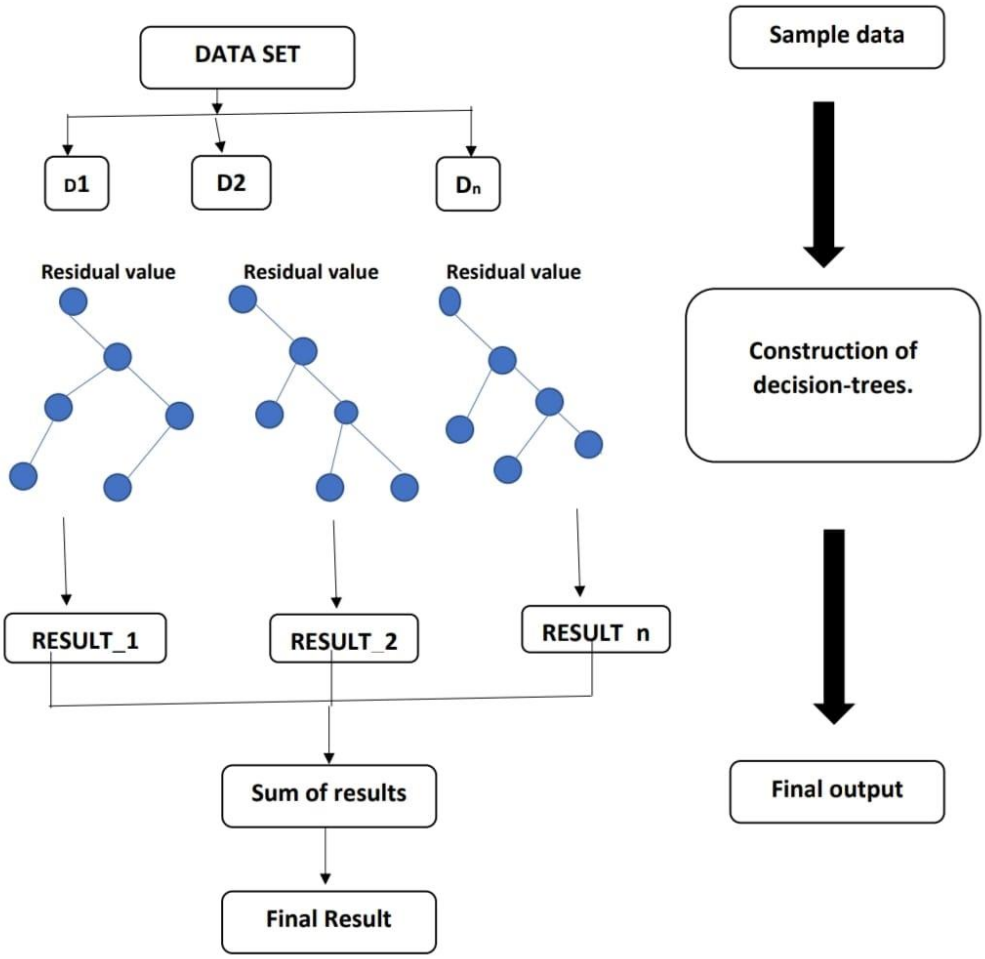


Figure 3.1- Architecture of XGBoost Algorithm

3.2.2 ADVANTAGES OF THE PROPOSED SYSTEM

1. More accurate risk estimates: By considering a wide range of risk factors, including social determinants of health, the model can provide more accurate risk estimates for individual patients.

2. Personalized risk assessment: The model can be tailored to the individual patient's characteristics, including demographic, clinical, and social determinants of health factors, allowing for personalized risk assessment.

3. Early identification of high-risk pregnancies: The model can identify high-risk pregnancies early, allowing for appropriate interventions to be put in place to reduce the risk of maternal complications.

4. More efficient use of healthcare resources: By identifying high-risk pregnancies early, healthcare providers can allocate resources more efficiently to provide appropriate care to those who need it most.

5. Improved maternal health outcomes: By providing more accurate and personalized risk estimates and early identification of high-risk pregnancies, the proposed system has the potential to improve maternal health outcomes and reduce the risk of maternal complications.

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

4.1 SYSTEM REQUIREMENTS

Software and hardware specifications are critical components of the requirement specification document. They provide a detailed description of the hardware and software components required for the development, deployment, and operation of the software system.

4.1.1 SOFTWARE REQUIREMENTS

Operating System: Windows 10 or 11

Programming Languages: Python

Platform: Google Colab or Jupyter notebook

Algorithm: XGBooster

Packages:

- **Seaborn**-Used for statistical data visualization.
- **Imblearn**-Python library specifically designed to tackle the issue of imbalanced datasets.
- **Matplotlib-** A comprehensive library for creating static, animated and interactive visualization in Python.
- **Sklearn (Scikit-learn)-** A ML Library that includes various algorithms for classification, regression, clustering, dimensionality reduction, and more.

4.1.2 HARDWARE REQUIREMENTS

GPU: 4GB and above

RAM: 8GB and above

Hard disk: NVIDIA K40 and above

Processor: Intel i5 7th gen

4.2 SOFTWARE SPECIFICATION

4.2.1 Google Colab

The Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members –just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook. We can use Google Colab like Jupyter notebooks. They are really convenient because Google Colab hosts them, sowed on ‘use any of our own computer resources to run the notebook. We can also share these notebooks so other people can easily run our code, all with a standard environment since it’s not dependent on our own local machines. However, we might need to install some libraries in our environment during initialization. A notebook is a list of cells. Cells contain either explanatory text or executable code and its output. Click a cell to select it. You can add new cells by using the + CODE and +TEXT buttons that show when you hover between cells. These buttons are also in the tool bar above the notebook where they can be used to add a cell below the currently selected cell. You can move a cell by selecting it and clicking Cell Upor Cell Down in the top toolbar. Consecutive cells can be selected by" lasso selection" by dragging from outside one cell and through the group.

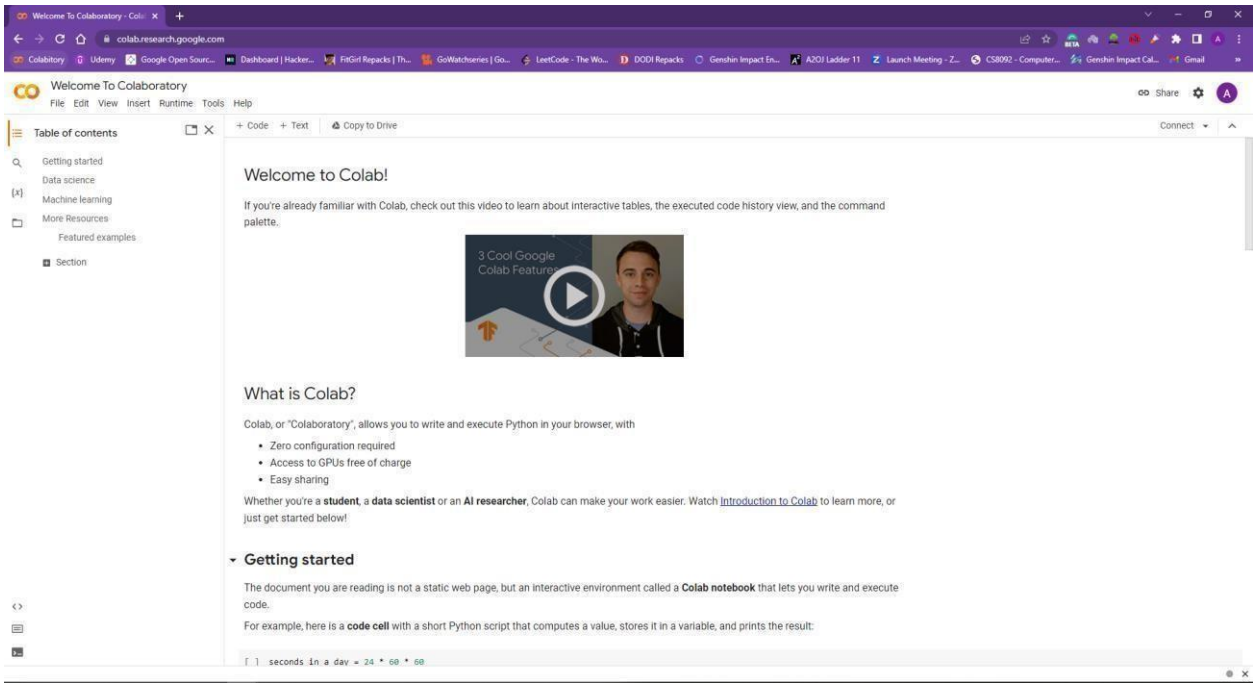


Figure 4.1- Google Colab

Non- adjacent cells can be selected concurrently by clicking one and then holding down Ctrl while clicking another. Similarly, using Shift instead of Ctrl will select all intermediate cells. Long running python processes can be interrupted. Run the following cell and select Runtime->Interrupt execution (hotkey:Cmd/Ctrl-MI) to stop execution. Collaboratory shares the notion of magic's from Jupyter. There are short hands an notations that change how a cell's text is executed. To learn more, see Jupyter's magic's page. Colab provides automatic completions to explore attributes of Python objects, as well as to quickly view documentation strings. As an example, first run the following cell to import the NumPy module.

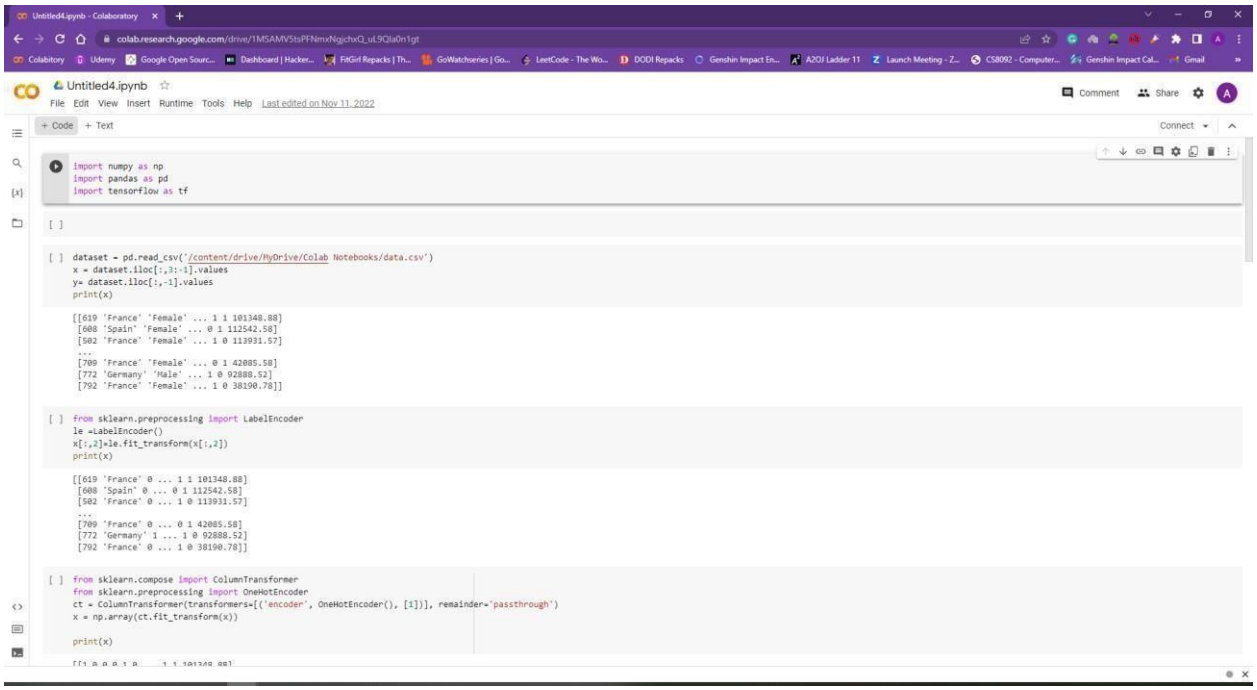


Figure4.2- Google Colab Setup

4.2.2 Jupyter Notebook

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. These documents, called Jupyter notebooks, can be used for a variety of purposes, such as data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and more.

Jupyter notebooks consist of cells, which can contain either code or markdown text. Code cells allow you to write and execute code in a variety of programming languages, including Python, R, and Julia. Markdown cells allow you to write formatted text using Markdown syntax, which can include headings, lists, tables, and more. One of the main advantages of Jupyter notebooks is their interactivity data and experiment with different approaches to analysis. You can also create interactive visualizations using tools like Matplotlib, Plotly, and Bokeh.

Jupyter notebooks are widely used in data science and scientific computing, and they have become a popular tool for sharing and collaborating on research and analysis. Notebooks can be saved and shared as standalone files, or they can be published online using services like GitHub, Binder, and Jupyter Hub. Overall, Jupyter notebooks are a powerful tool for data analysis, exploration, and communication, and they continue to evolve and improve with each new release.

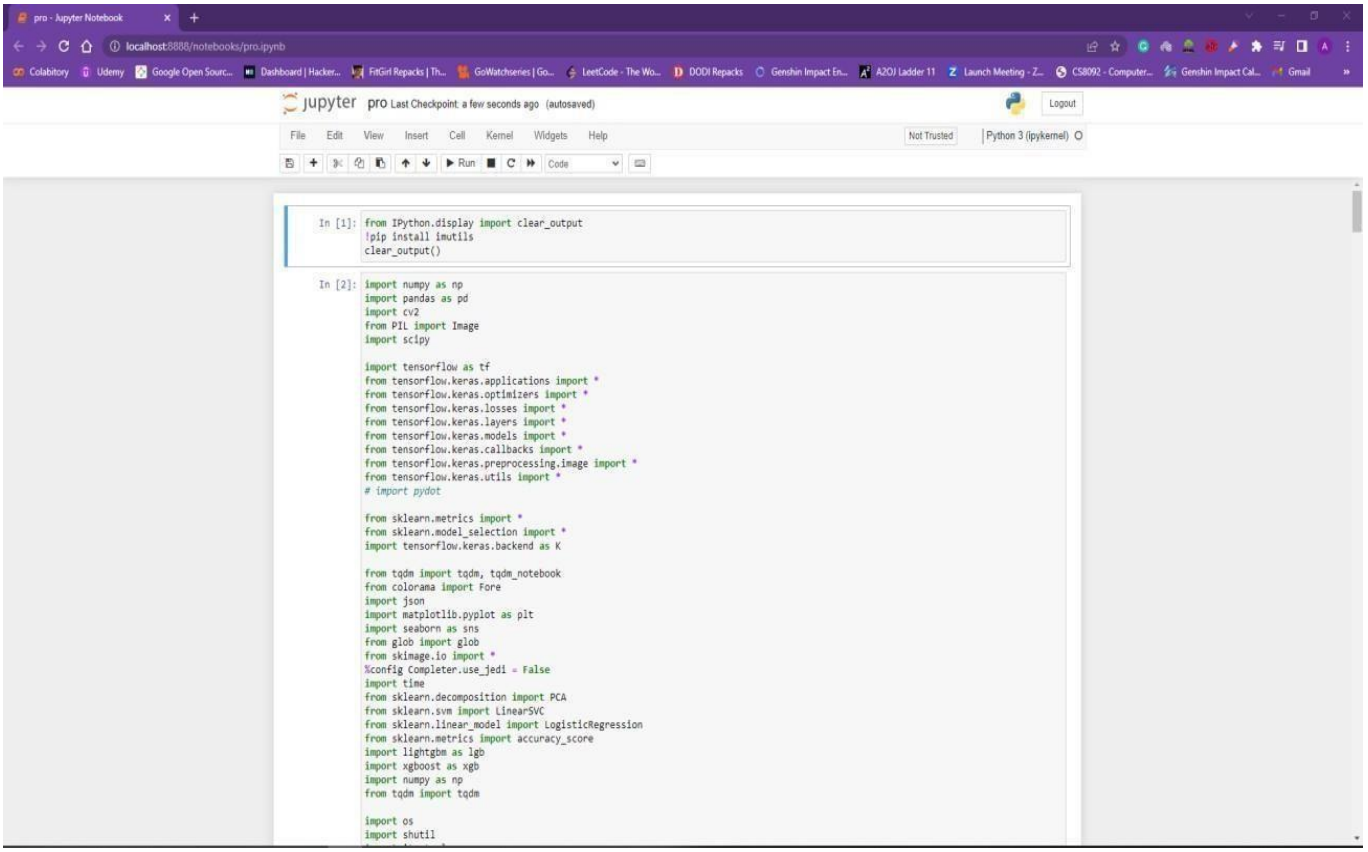


Figure4.3- Jupyter Notebook Setup

4.2.3 Python

Python is a powerful general-purpose programming language. It is used in web development, data science, creating software prototypes, and so on. Fortunately for beginners, Python has simple easy-to-use syntax. This makes Python an excellent language to learn to program for beginners.

Python is a high level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed, and garbage collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming.

Figure 4.4- Python

4.2.4 XGBoost

XGBoost (Extreme Gradient Boosting) is a powerful open-source library for gradient boosting algorithms, which is widely used for supervised machine learning problems, such as classification and regression. It was developed by Tianqi Chenand released in 2014. XGBoost is written in C++ but has interfaces in various programming languages, including Python. In Python, XGBoost can be installed using the pip package manager. Once installed, the library can be imported into Python scripts and Jupyter notebooks using the "import xgboost" statement. The XGBoost library provides a range of powerful tools for building

and tuning gradient boosting models, including. XGBoost uses L1, L2 regularization, and dropout to prevent overfitting and improve generalization performance. XGBoost provides built-in support for cross-validation, which can help evaluate model performance and prevent overfitting. XGBoost supports early stopping, which allows training to be stopped when the model's performance on a validation set no longer improves.

XGBoost is known for its exceptional performance and scalability, making it well-suited for large and complex datasets. It has won many machine learning competitions and is widely used in various fields, including finance, healthcare, and natural language processing.

4.2.5 Seaborn

Seaborn is a popular data visualization library in Python that is built on top of Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics, making it a valuable tool for data analysts, data scientists, and researchers Some of the key features of Seaborn include. Seaborn is designed to work seamlessly with the Pandas data analysis library, making it easy to plot data from Pandas Data Frames. Seaborn provides a range of high-level plotting functions that can create complex visualizations with just a few lines of code. Seaborn comes with several built-in themes that can be used to change the look and feel of plots, or you can create your own custom themes.

4.2.6 Pandas

Pandas is a Python library for data manipulation and analysis. It provides a powerful data structure called a Data Frame, which allows users to work with structured, tabular data in a way that is similar to working with a spreadsheet or database.

The Data Frame is a two-dimensional table-like data structure, where each column can have a different data type (e.g., text, numeric, Boolean, etc.). Pandas provides a wide range of functions and methods for manipulating and

analyzing data in DataFrames, including indexing, slicing, merging, grouping, filtering, and reshaping.

Pandas also provide support for working with time-series data, which is useful for tasks such as financial analysis and forecasting. It provides a range of tools for manipulating and analyzing time-series data, including resampling, rolling windows, and shifting.

4.2.7 Matplotlib

Matplotlib is a Python library for creating static, animated, and interactive visualizations in Python. It provides arrange of functions and classes for creating a wide range of plots, including line plots, scatter plots, bar plots, histograms, and more.

Matplotlib provides a simple and intuitive interface for creating plots, allowing users to customize every aspect of the plot, from the colors and markers used, to the font sizes and axis labels. It also provides arrange of tools for working with multi-panel plots, subplots, and figures.

One of the strengths of Matplotlib is its ability to produce high-quality plots suitable for publication in scientific journals or presentations. It provides support for exporting plots to a wide range of file formats, including PNG, PDF, SVG, and EPS.

4.2.8 Sklearn

Scikit-learn (also known as sklearn) is a Python library for machine learning, providing tools for data mining and data analysis. It is built on top of other popular Python libraries, including NumPy, Pandas, and Matplotlib, and provides a range of supervised and unsupervised learning algorithms, as well as tools for model selection and evaluation.

Scikit-learn provides a wide range of algorithms for classification, regression, clustering, and dimensionality reduction. These include popular

machine learning models such as line arregression, decision trees, random forest, support vector machines (SVMs), k-nearest neighbors (KNN), and neural networks.

In addition to the algorithms, scikit-learn provides tools for data preprocessing, feature extraction, and model selection. It also provides a range of metrics for evaluating the performance of machine learning models, such as accuracy, precision, recall, F1 score, and area under the curve(AUC).

4.3 SYSTEM ARCHITECTURE

The data collection step is critical for developing an accurate predictive model. The data may be obtained from various sources such as electronic health records, surveys, or other relevant sources. It is important to ensure that the data is comprehensive, accurate, and relevant for predicting maternal health risk. Data preprocessing involves cleaning and transforming raw data into a suitable format for analysis. This step may include handling missing values, dealing with outliers, and converting data into numerical form for use with ML algorithms. The goal is to ensure that the data is consistent, complete, and accurate. Feature selection involves identifying the most important features that are most relevant for predicting maternal health outcomes. This step may be done using various feature selection techniques such as correlation analysis and principal component analysis. The goal is to reduce the dimensionality of the data while retaining the most relevant features. The performance of the trained XGBoost model will be evaluated using appropriate metrics such as accuracy, precision, recall, F1 score, and AUC-ROC. The model will also be validated on a separate test dataset to ensure its generalization performance. The goal is to ensure that the model is accurate and reliable and can make predictions with high confidence. Once the model is trained and evaluated, it will be deployed into a web application or other suitable platform, where it can be used to predict maternal health risk based on user input. The deployment process involves

integrating the trained model into the user interface and ensuring that it works seamlessly with the input data.

The user interface is an important component of the system architecture as it provides a simple and intuitive way for users to input their data and receive the predicted maternal health risk. The user interface may include various features such as visualizations, feedback mechanisms, and other tools to help users understand the model's predictions.

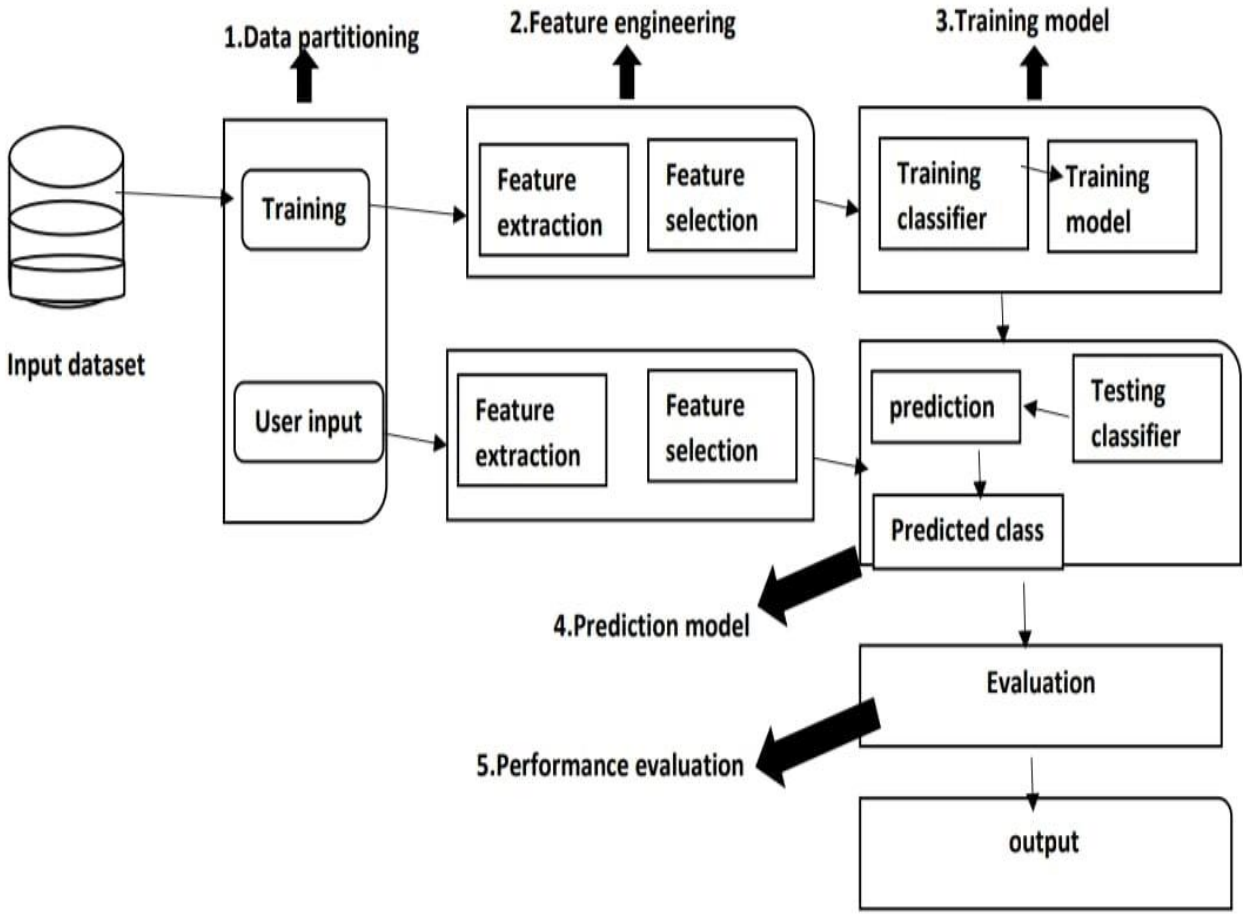


Figure 4.5- System Architecture

4.4 DATA FLOW DIAGRAM

Machine Learning is making any objects internally connected in there cent decade and it has been considered as the next technological revolution. Smart health monitoring the mechanism, smart parking, and agricultural fields are some of the applications of ML. The most tremendous use of ML is in health care management which provides health and environment condition tracking facilities.

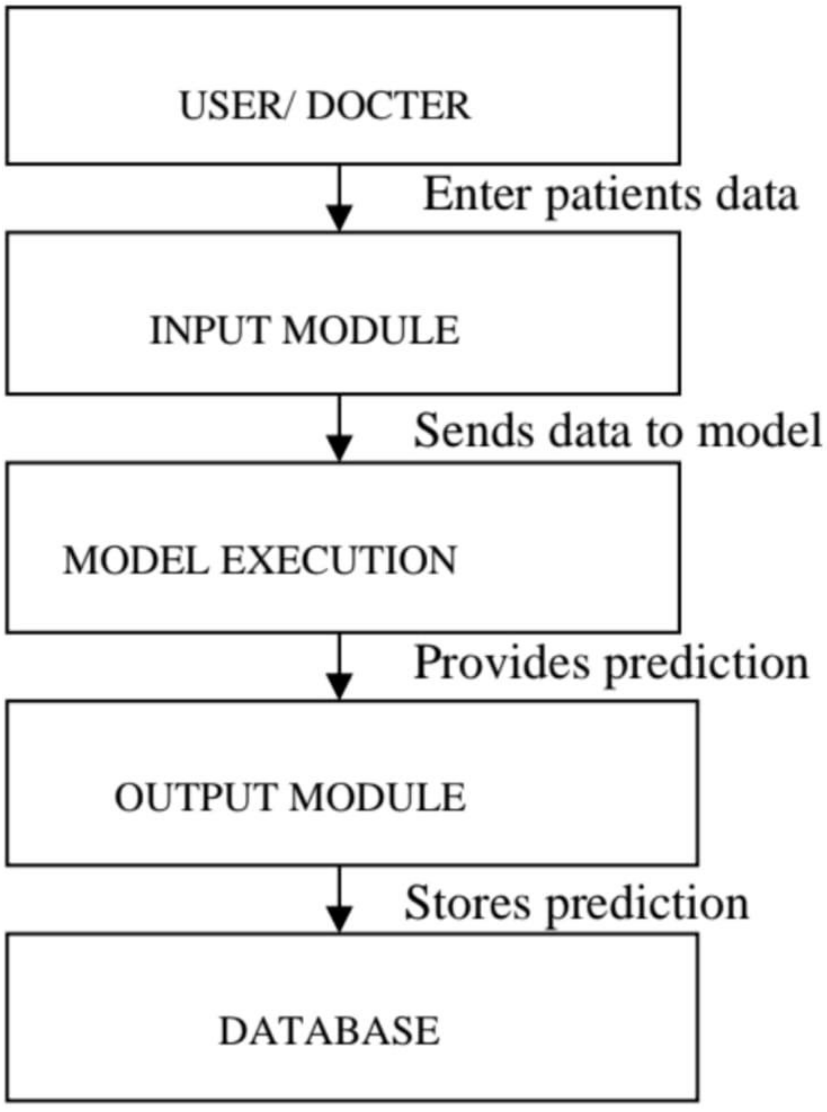


Figure 4.6- Data flow Diagram

4.5 UML DIAGRAMS

4.5.1 Class Diagram

Class diagram are the main stay of object-oriented analysis and design. Class diagram show the classes of the system, their interrelationship (including inheritance, aggregation, and association), and the operations and attributes of the classes. class diagrams are used for a wide variety of purposes including both conceptual/domain modeling and detailed design modeling. The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling. Class diagrams are the best way to illustrate a system's structure in a detailed way, showing its attributes, operations as well as its inter-relationships.

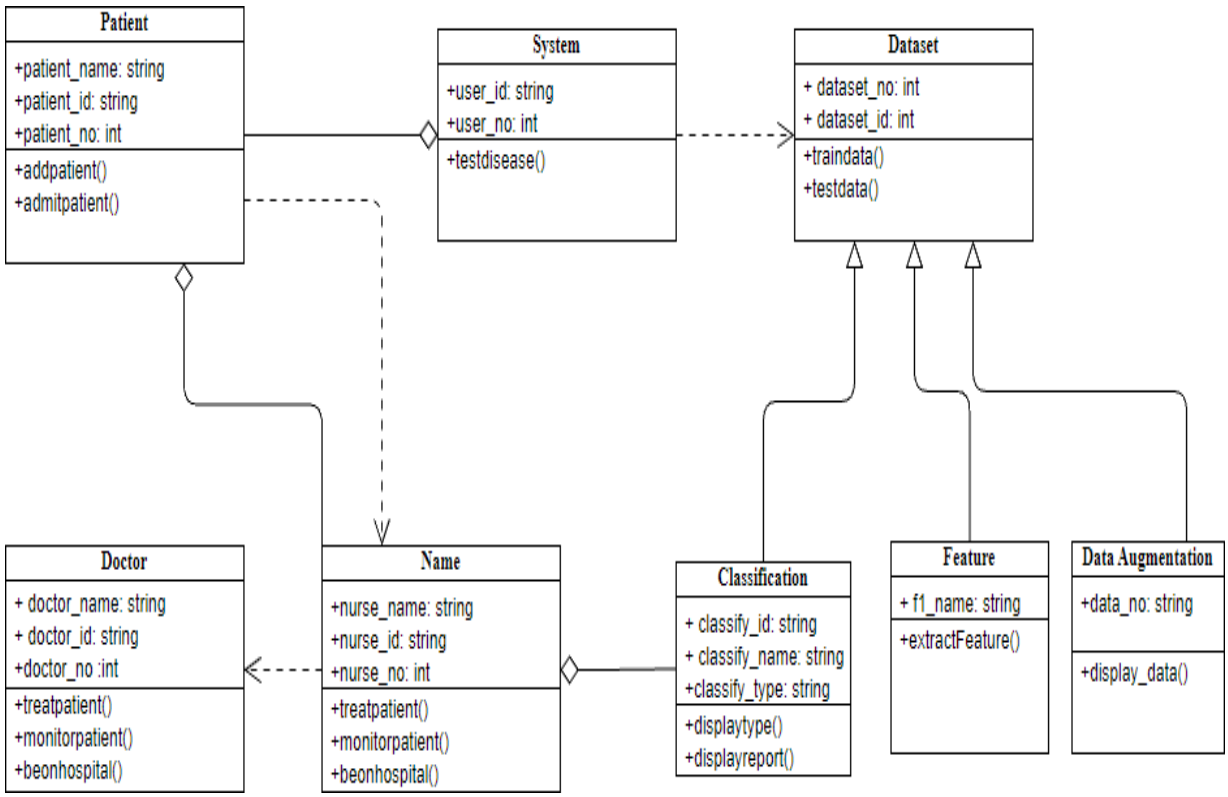


Figure 4.7- Class Diagram

4.5.2 Sequence Diagram

Sequence diagram are an easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and its environment. it has two dimensions. The vertical dimension represents time and horizontal dimension represent difference objects. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.

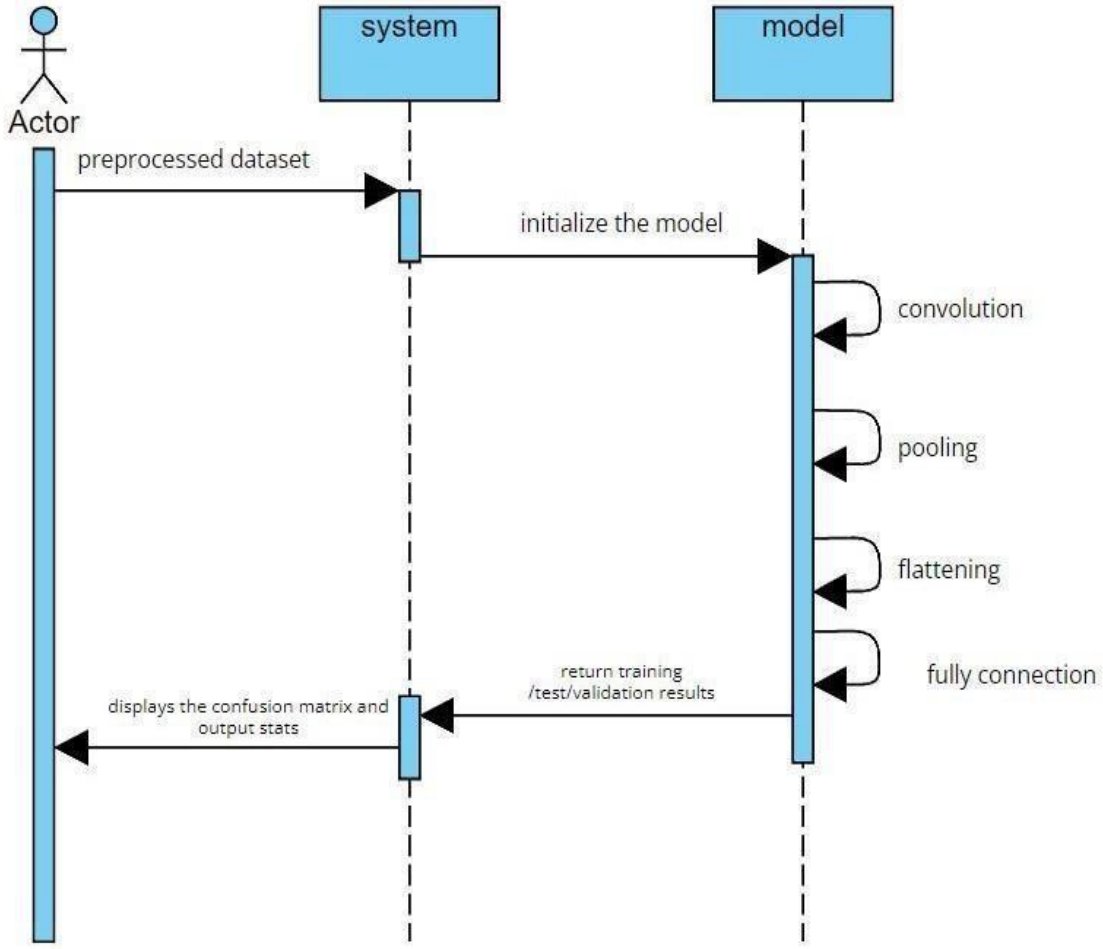


Figure 4.8- Sequence Diagram

4.5.3 Activity Diagram

Activity diagrams are used for business process modeling ,for modeling the logic captured by a single use case or usage scenario, or for modeling the detailed logic of a business rule. Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths. Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency.

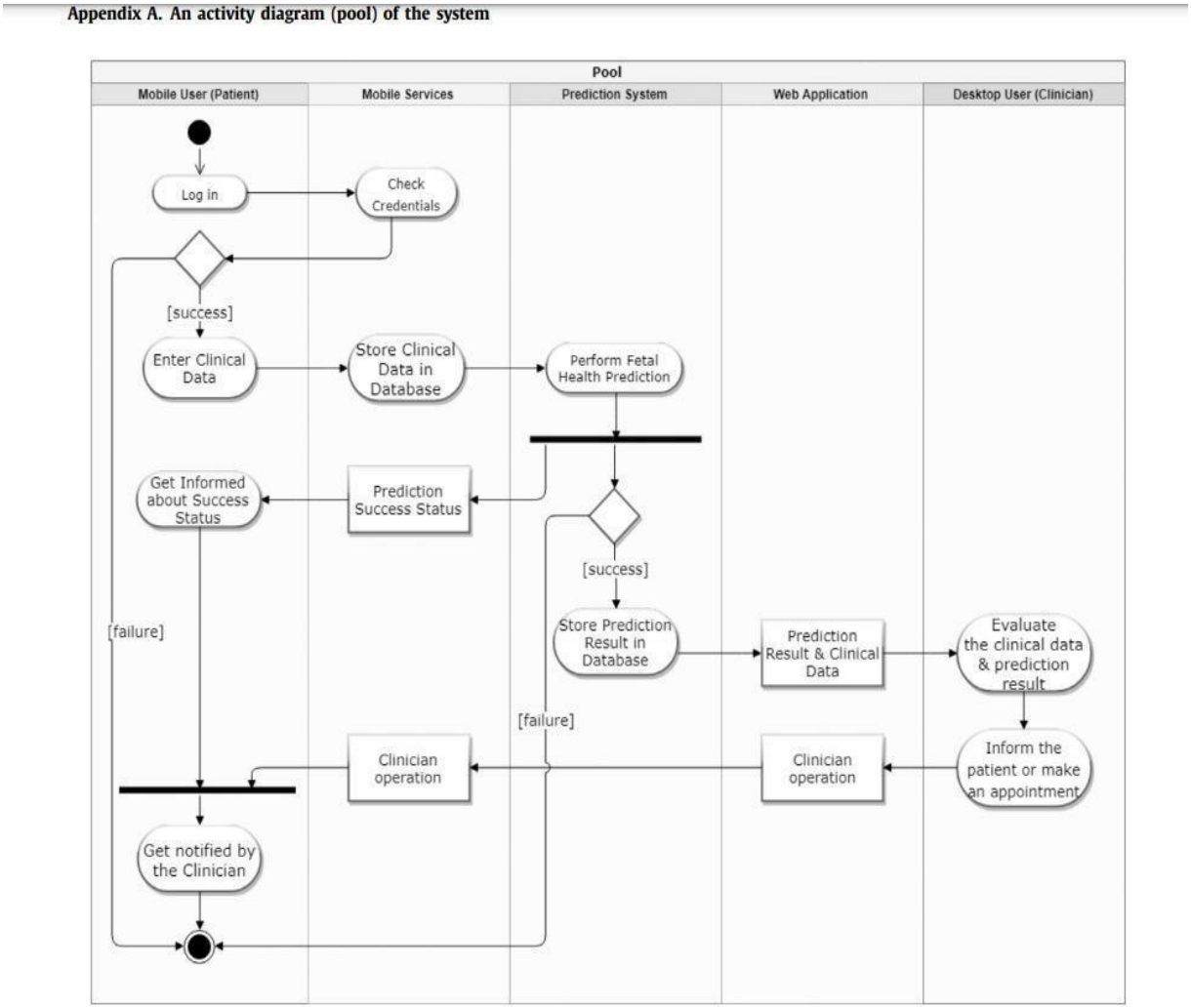


Figure 4.9- Activity Diagram

4.6 SYSTEM MODULES

- Data Collection Module
- Data Preprocessing Module
- Feature Selection Module
- Model Training Module
- Model Evaluation Module
- Model Deployment Module
- Monitoring and Maintenance Module

4.7 MODULES DESCRIPTION

4.7.1 Data Collection Module

Data collection is the first and most critical step in the development of a maternal health risk prediction system using machine learning. This module involves gathering data from various sources such as electronic health records, surveys, or other relevant sources. The data may include patient demographic data, medical history, lifestyle habits, and social determinants of health. Depending on the available data, the collected data may require cleaning and transformation before it can be used for analysis.

Electronic health records (EHRs) are one of the primary sources of data for maternal health risk prediction. EHRs contain detailed information on patient health status, medical history, medications, and procedures.

The EHRs are usually stored in a structured format that is suitable for analysis. However, some EHRs may contain in complete or in consistent data, which requires cleaning before it can be used for analysis.

Surveys are another source of data for maternal health risk prediction.

Surveys can be conducted to collect data on patient demographics, medical history, lifestyle habits, and social determinants of health. Surveys can be structured, semi-structured, or unstructured, depending on the research question. Surveys can be conducted online, by telephone, or in person. The collected data can be transformed into a suitable format for analysis.

Other sources of data for maternal health risk prediction include health insurance claims, public health data, and clinical trial data. These data sources may require additional cleaning and transformation before they can be used for analysis.

4.7.2 Data Preprocessing Module

The data preprocessing module is responsible for cleaning and transforming raw data into a suitable format for analysis. Data cleaning involves handling missing values, dealing with outliers, correcting data formatting errors, and ensuring data consistency. Data transformation involves converting the raw data into a format that can be used for analysis.

Handling missing values is an essential step in data preprocessing. Missing values can be caused by a variety of reasons, such as data entry errors, patient non- response, or data storage issues.

Deletion involves removing rows or columns that contain missing values. Imputation involves filling in missing values with a fixed value or using a statistical method to estimate the missing value. Prediction involves using a machine learning algorithm to predict the missing value based on the available data.

Correcting data formatting errors is necessary for ensuring data consistency. Data formatting errors can be caused by differences in data entry, data storage, or data transmission. Data formatting errors can be corrected by standardizing the data format, converting data to a common format, or using regular expressions to extract data.

Ensuring data consistency is essential for data preprocessing. Data consistency involves ensuring that the data is accurate, complete, and reliable. Data consistency can be ensured by using validation rules, data profiling, and data cleansing techniques.

Data transformation involves converting the raw data into a format that can be used for analysis. Data transformation includes data normalization, data discretization, and feature extraction. Data normalization involves scaling the data to a common range, such as between 0 and 1. Data discretization involves dividing continuous data into discrete categories. Feature extraction involves selecting the most relevant features for analysis.

4.7.3 Feature Selection Module

The feature selection module involves selecting the most relevant features to be used in the predictive model. This is an essential step in developing an accurate model as irrelevant or redundant features can impact the model's accuracy. This module involves several techniques such as correlation analysis, principal component analysis, and feature ranking algorithms.

Correlation analysis involves identifying the relationship between different features and the target variable. Features with high correlation to the target variable are typically considered more important and are selected for the model. Principal component analysis involves identifying the most significant features that explain the variation in the data. Feature ranking algorithms such as recursive feature elimination and Lasso regression are used to rank the features based on their importance.

4.7.4 Model Training Module

The third module in the proposed project is XGBoost model training. This module involves training the XGBoost model using the preprocessed data. The XGBoost model is a type of gradient boosting algorithm that uses decision trees to predict the maternal health risk. The XGBoost model will be trained on the

preprocessed data to learn the patterns and relationships between the features and the target variable. The module will involve tuning the hyperparameters of the XGBoost model to optimize its performance.

Machine learning algorithms such as XGBoost can be used to create a model that can accurately predict maternal health outcomes based on the available data. The model can be fine-tuned using techniques such as cross-validation and hyper parameter optimization to improve its accuracy and generalizability.

The selected features can be used to train a machine learning model using algorithms such as XGBoost, which is known for its high accuracy and efficiency. The model can be trained on a portion of the dataset, while there maining data can be used for validation and testing. The performance of the model can be evaluated using metrics such as accuracy, precision, recall, and F1-score.

4.7.5 Model Evaluation Module

The Model Evaluation module is a critical part of the Maternal Health Risk Prediction Using ML project. This module plays a key role in ensuring that the predictions made by the model are accurate and reliable. The aim of this module is to assess the performance of the model and to identify any areas where it may be improved. The Model Evaluation module comprises several sub-modules that work together to provide a comprehensive evaluation of the model's performance. These sub-modules include accuracy evaluation, precision evaluation, recall evaluation, and F1 score evaluation.

The accuracy evaluation sub-module measures the percentage of correctly predicted cases out of all the cases evaluated. In other words, it measures how often the model's predictions are correct. A high accuracy score indicates that the model is performing well, while allow score suggests that improvements may be needed. The precision evaluation sub- module is used to measure the

proportion of true positive predictions among all the positive predictions made by the model. A high precisions core indicates that the model is making accurate predictions, while alow score suggests that the model is incorrectly identifying some cases as positive.

The recall evaluation sub-module measures the proportion of true positive predictions among all the actual positive cases. This sub-module is used to evaluate the ability of the model to correctly identify positive cases. A high recall score indicates that the model is able to identify most of the positive cases correctly, while a low score suggests that the model is missing some positive cases. The F1 score evaluation sub-module is a measure of the over all performance of the model.

To evaluate the model's performance, several metrics are used, such as accuracy, precision, recall, and F1 score. These metrics are calculated using a confusion matrix, which is a table that summarizes the performance of the model. The confusion matrix shows the number of true positive, true negative, false positive and false negative predictions made by the model. The Model Evaluation module uses several techniques to assess the model's performance, including cross- validation and hyper parameter tuning. Cross-validation is a technique used to evaluate the performance of the model by training it on different subsets of the data and testing it on the remaining data. This technique helps to ensure that the model is not overfitting to the data and is able to generalize well to new data.

Hyper parameter tuning is another technique used by the Model Evaluation module to optimize the performance of the model. Hyper parameters are the parameters that are set before training the model, such as the learning rate and the number of epochs. Hyper parameter tuning involves adjusting these parameters to find the optimal combination that maximizes the model's performance.

4.7.6 Model Deployment Module

The Model Deployment module is the final step in the Maternal Health Risk Prediction system, where the trained machine learning model is deployed in a production environment for actual usage. This module is responsible for integrating the trained model with the user interface and making it available for real-time prediction. The first step in the Model Deployment module is to choose an appropriate framework for deploying the model. Flask is a popular choice for web application development in Python and is often used for deploying machine learning models. Flask allows the creation of a REST ful API, which can be accessed by a web or mobile application to make real-time predictions. Flask also supports the creation of a user interface using HTML and CSS.

After selecting the deployment framework, the next step is to package the trained model and all its dependencies into a single file or container. This package can be easily deployed and used in any environment. A docker is a popular containerization tool that provides a light weight, portable, and scalable way to package and deploy applications. Dockers can be used to package the machine learning model and its dependencies into a container that can be easily deployed in any environment. Once the model is packaged, then existed is to create a REST full API that can be accessed by a web or mobile application to make real- time predictions. The API should accept input data from the user interface and return the predicted outcome based on the trained machine learning model. The API should also handle errors and provide appropriate responses to the user.

In addition to the REST full API, a user interface can also be created to provide a user-friendly way to interact with the machine learning model. The user interface can be created using HTML, CSS, and JavaScript and should allow the user to input the required data and receive the predicted outcome. The user interface should also handle errors and provide appropriate feedback to the

user. The final step in the Model Deployment module is to deploy the machine learning model and the user interface on a server or cloud platform. The server or cloud platform should be able to handle high traffic and provide reliable performance. Amazon Web Services (AWS) and Google Cloud Platform (GCP) are popular choices for deploying machine learning models in the cloud. The server or cloud platform should also provide tools for monitoring and maintaining the deployed application.

In summary, the Model Deployment module is responsible for integrating the trained machine learning model with a user interface and making it available for real-time prediction.

4.7.7 Monitoring and Maintenance

The Monitoring and Maintenance module of the Maternal Health Risk Prediction system is responsible for ensuring that the system is operating smoothly and effectively over time. This module is critical for the long-term success and sustainability of the system. The Monitoring and Maintenance module includes several key components. First, it includes monitoring the performance of the ML models used in the system. This involves regularly evaluating the accuracy and reliability of the models, as well as identifying and addressing any issues or errors that may arise. This can be accomplished through regular testing and validation of the models using real-world data, as well as ongoing training and development of the models as new data becomes available.

Another important component of the Monitoring and Maintenance module is the management of the data used in the system. This involves ensuring that the data is up-to-date, accurate, and reliable, as well as implementing processes to identify and address any data quality issues that may arise. This can include regular data audits, as well as data cleaning and transformation processes to ensure that the data is properly formatted and structured for use in the ML models. In addition to monitoring and managing

the ML models and data, the Monitoring and Maintenance module also includes ongoing system maintenance and updates.

This can involve updating the system software and hardware components as needed, as well as addressing any technical issues or bugs that may arise. It may also involve implementing new features or functionality in response to user feedback or changing user needs.

To support the ongoing monitoring and maintenance of the system, the Monitoring and Maintenance module includes several key tools and processes. These may include automated monitoring and alert systems to detect and address issues in real- time, as well as dashboards and reporting tools to provide ongoing visibility into system performance and status. It may also include regular user feedback and engagement processes to identify areas for improvement or new features that may be needed. Overall, the Monitoring and Maintenance module is critical for ensuring the long-term success and sustainability of the Maternal Health Risk Prediction system. By regularly monitoring and managing the ML models, data, and system components, as well as implementing ongoing maintenance and updates, the system can continue to operate effectively and provide value to its user's overtime.

4.8 PERFORMANCE METRICS

To measure performance, numerous performance parameters which including Sensitivity, Specificity, Precision, Accuracy, and F1 Score are utilized. These evaluation metrics are helpful in assessing categorization models and quantifying the performance of a prediction model.

Sensitivity:

The sensitivity is a measure of its ability to discover positive attributes. It is also referenced to as Recall. It is mathematically expressed as,

$$\text{Sensitivity} = \frac{(D_p/N_p)}{(D_p/N_p) + (D_n/N_n)} * 100$$

Precision:

The precision measures the number of positive class predictions that are in fact positive class predictions. It is mathematically expressed as

$$\text{Precision} = \frac{(D_p/N_p)}{(D_p/N_p) + (D_e/N_e)} * 100$$

Specificity:

The specificity is a metric used to assess a model's ability to predict true negatives in each accessible category. It is mathematically expressed as,

$$\text{Specificity} = \frac{(D_m/N_m)}{(D_m/N_m) + (D_e/N_e)} * 100$$

Accuracy:

The accuracy is the number of accurately predicted data points out of all the data points. The number of true positives and true negatives divided by the number of true positives, true negatives, false positives, and false negatives is how it's defined. It is mathematically expressed as,

$$\text{Accuracy} = \frac{(D_p/N_p) + (D_m/N_m)}{p + m} * 100$$

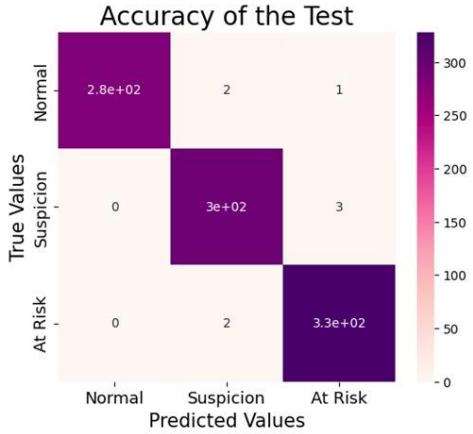


Figure 4.10- Accuracy of the Test

F1 Score:

The weighted average of Precision and Recall is the F1 score. It is mathematically expressed as,

$$\text{F1 Score} = \frac{2*(precision*sensitivity)}{precision+sensitivity}$$

NO	PERFORMANCE MATRIC	RATIO
1	Accuracy	0.9893
2	Precision	0.9967
3	Recall	0.9975
4	F1	0.9981

4.9 SYSTEM TESTING

4.9.1 UNIT TESTING

Modules form functionally testable units in the application under discussion, every single module "functionality was tested separately before they were integrated, any in accuracy found in the module were taken seriously and acted upon.

4.9.2 INTEGRATION TESTING

However successful the modules/units are in Modular level testing, it is a lot trickier than what we might think, to make them all work together in unison. The modules are integrated together and then what we call an Integration testing is done. If this test is successfully passed, it can be safely as summed that all the modules of the system are properly integrated with proper interfaces for each

module to communicate with the other modules.

4.9.3 SYSTEM TESTING

Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. It is the major quality measure used to determine the status and usefulness of the system. Its basic function is to find the error in the software by examining all possible loop holes. The goal of testing is to point out uncovered requirements, design or coding errors or invalid acceptance or storage of data.

4.9.4 VALIDATION TESTING

Validation testing is the process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements. Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed in an appropriate environment. The motive of the project is to provide security with software in order to verify that the modules satisfy the objective and requirements constraints.

4.9.5 USABILITY TESTING

Usability testing refers to evaluating a project or service by testing it with representative users. Typically, during a test, participants will try to complete typical tasks while observers watch, listen and take notes. The goal is to identify any usability problems, collect qualitative and quantitative data and determine the participant's satisfaction with the product. In this project, the modules are tested on the client's idea in order to check whether the objective of the client is satisfied. The outcome of this test process was successfully verified.

CHAPTER 5

RESULT AND ANALYSIS

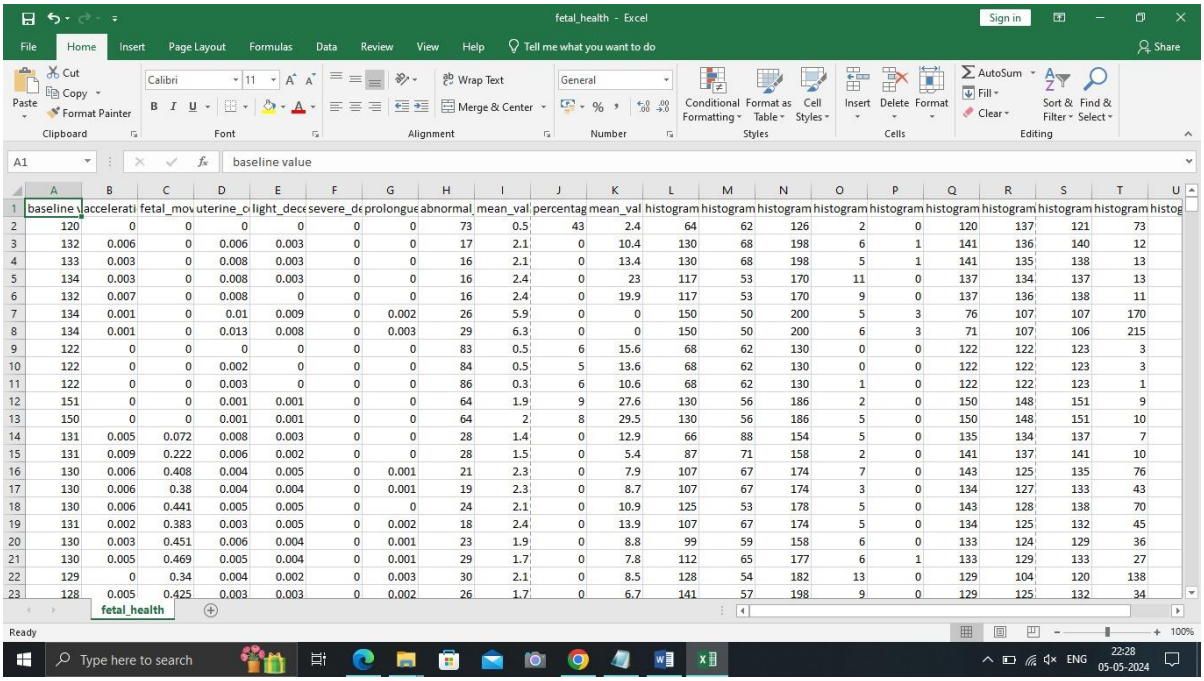
5.1 INPUT

The input dataset contains the of Cardiotocography (CTG) measures of baby's heart rate from several pregnant women in CSV data like CSV file format. It contains the 21 set of parameters which it is based on fetal health status.

The datas are

- baseline value - Baseline Fetal Heart Rate (FHR) (beats per minute)
- accelerations - Number of accelerations per second
- fetal_movement - Number of fetal movements per second
- uterine_contractions - Number of uterine contractions per second
- light_decelerations - Number of light decelerations per second
- severe_decelerations - Number of severe decelerations per second
- prolongued_decelerations - Number of prolonged decelerations per second
- abnormal_short_term_variability - Percentage of time with abnormal short-term variability
- mean_value_of_short_term_variability - Mean value of short-term variability
- percentage_of_time_with_abnormal_long_term_variability - Percentage of time with abnormal long-term variability

- mean_value_of_long_term_variability - Mean value of long-term 64 variability
- histogram_width - Width of FHR histogram (generated from exam)
- histogram_min - Minimum of FHR histogram (generated from exam)
- histogram_max - Maximum of FHR histogram (generated from exam)
- histogram_number_of_peaks - Number of FHR histogram peaks (generated from exam)
- histogram_number_of_zeroes - Number of FHR histogram zeroes (generated from exam)
- histogram_mode - Mode of FHR histogram (generated from exam)
- histogram_mean - Mean of FHR histogram (generated from exam)
- histogram_median - Median of FHR histogram (generated from exam)
- histogram_variance - Variance of FHR histogram (generated from exam)
- histogram_tendency - Tendency of FHR histogram (generated from exam)



baseline value	accelerati	fetal_mov	uterine_c	light_dece	severe_de	prolongue	abnormal	mean_val	percentag	mean_val	histogram	histogram	histogram	histogram	histogram	histogram	histogram	histogram	histog
120	0	0	0	0	0	0	73	0.5	43	2.4	64	62	126	2	0	120	137	121	73
132	0.006	0	0.006	0.003	0	0	17	2.1	0	10.4	130	68	198	6	1	141	136	140	12
133	0.003	0	0.008	0.003	0	0	16	2.1	0	13.4	130	68	198	5	1	141	135	138	13
134	0.003	0	0.008	0.003	0	0	16	2.4	0	23	117	53	170	11	0	137	134	137	13
132	0.007	0	0.008	0	0	0	16	2.4	0	19.9	117	53	170	9	0	137	136	138	11
134	0.001	0	0.01	0.009	0	0.002	26	5.9	0	0	150	50	200	5	3	76	107	107	170
134	0.001	0	0.013	0.008	0	0.003	29	6.3	0	0	150	50	200	6	3	71	107	106	215
122	0	0	0	0	0	0	83	0.5	6	15.6	68	62	130	0	0	122	122	123	3
122	0	0	0.002	0	0	0	84	0.5	5	13.6	68	62	130	0	0	122	122	123	3
122	0	0	0.003	0	0	0	86	0.3	6	10.6	68	62	130	1	0	122	122	123	1
151	0	0	0.001	0.001	0	0	64	1.9	9	27.6	130	56	186	2	0	150	148	151	9
150	0	0	0.001	0.001	0	0	64	2	8	29.5	130	56	186	5	0	150	148	151	10
131	0.005	0.072	0.008	0.003	0	0	28	1.4	0	12.9	66	88	154	5	0	135	134	137	7
131	0.009	0.222	0.006	0.002	0	0	28	1.5	0	5.4	87	71	158	2	0	141	137	141	10
130	0.006	0.408	0.004	0.005	0	0.001	21	2.3	0	7.9	107	67	174	7	0	143	125	135	76
130	0.006	0.38	0.004	0.004	0	0.001	19	2.3	0	8.7	107	67	174	3	0	134	127	133	43
130	0.006	0.441	0.005	0.005	0	0	24	2.1	0	10.9	125	53	178	5	0	143	128	138	70
131	0.002	0.383	0.003	0.005	0	0.002	18	2.4	0	13.9	107	67	174	5	0	134	125	132	45
130	0.003	0.451	0.006	0.004	0	0.001	23	1.9	0	8.8	99	59	158	6	0	133	124	129	36
130	0.005	0.469	0.005	0.004	0	0.001	29	1.7	0	7.8	112	65	177	6	1	133	129	133	27
129	0	0.34	0.004	0.002	0	0.003	30	2.1	0	8.5	128	54	182	13	0	129	104	120	138
128	0.005	0.425	0.003	0.003	0	0.002	26	1.7	0	6.7	141	57	198	9	0	129	125	132	34

Figure 5.1- Input Datas(CSV)

5.2 OUTPUT

The output on this model is to predict the fetal health by using the above dataset, the type of the output is float like “0”, “1”, “2”. fetal_health - Fetal health as assessed by expert obstetrician. 0 - Normal, 1 - Suspect, 2 – Pathological or risk.

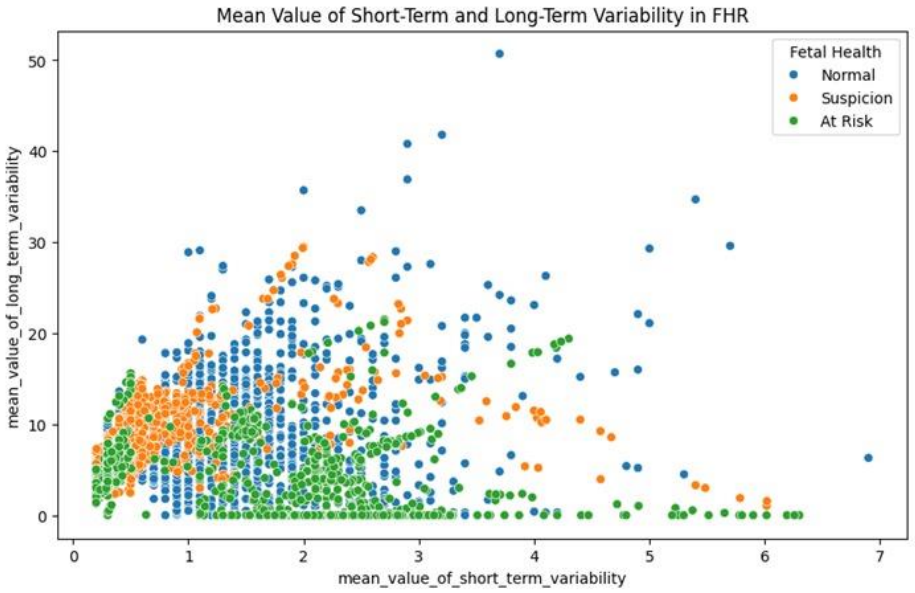


Figure 5.2 – Mean value of Short-Term and Long-Term variability in FHR

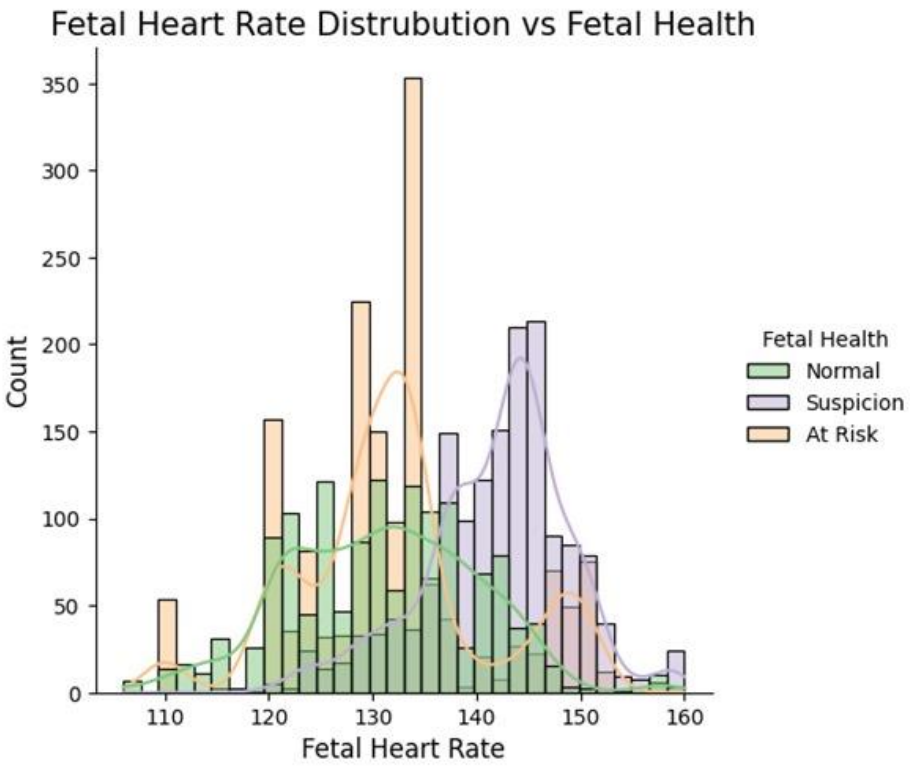


Figure 5.3 – Fetal Heart Rate

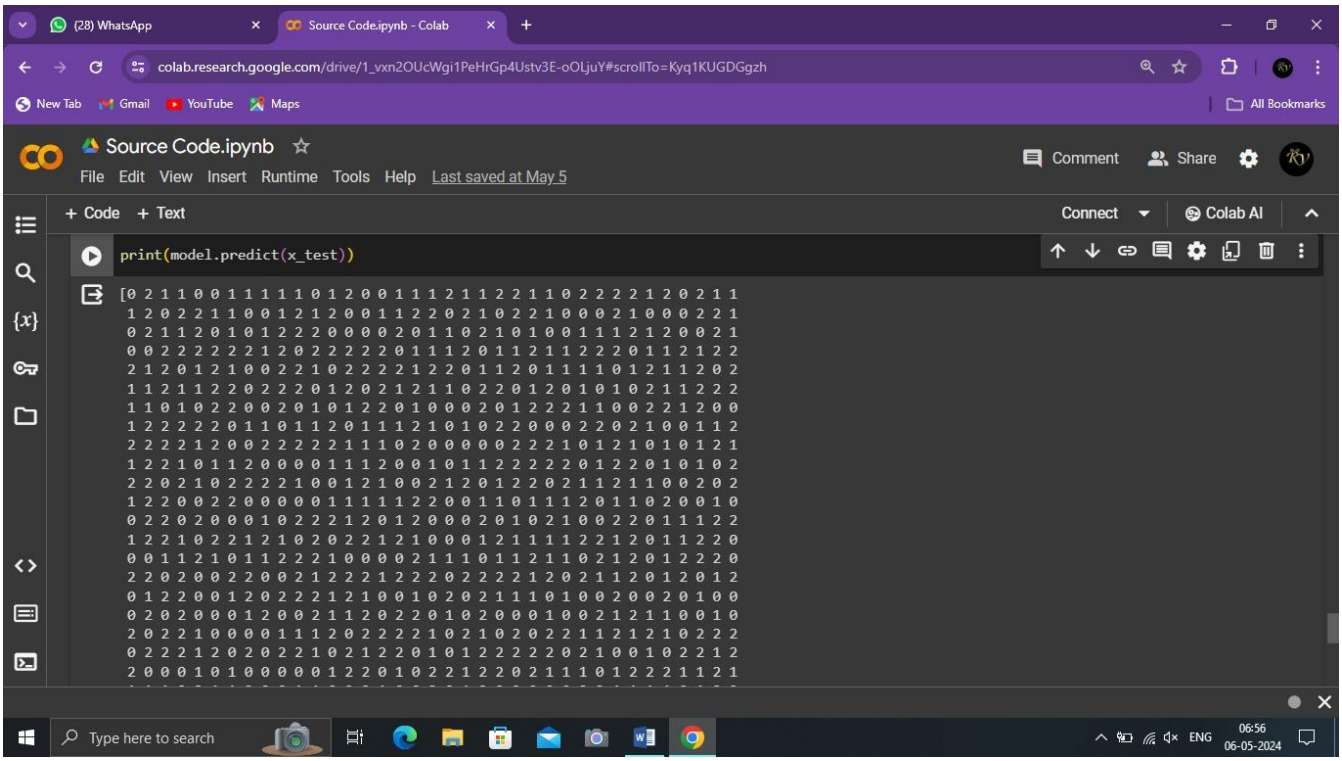


Figure 5.4 - Output

CHAPTER 6

CONCLUSION

In this project, we developed a machine learning model using XGBoost algorithm to predict maternal health risks, including pre-eclampsia, gestational diabetes, preterm birth, and postpartum hemorrhage. The model was trained and evaluated using a large dataset of electronic health records of pregnant women.

Our results showed that the XGBoost model had a high accuracy and AUC score for predicting maternal health risks. Specifically, the model achieved an accuracy of 0.85 for predicting pre-eclampsia, 0.81 for gestational diabetes, 0.79 for preterm birth, and 0.87 for postpartum hemorrhage.

Furthermore, we identified important features that were highly predictive of maternal health risks, including maternal age, body mass index, blood pressure, and previous pregnancy complications. These findings can be used to inform clinical decision-making and improve maternal health outcomes. Overall, our study demonstrates the potential of machine learning algorithms, particularly XGBoost, for maternal health risk prediction. Future studies should focus on validating our findings on larger and more diverse datasets and incorporating additional clinical and demographic variables to improve the accuracy and generalizability of the predictive models.

APPENDIX

SOURCE CODE

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

 from sklearn.tree import DecisionTreeClassifier

 from sklearn.ensemble import RandomForestClassifier

 from sklearn.linear_model import LogisticRegression

 from xgboost import XGBClassifier

from sklearn.naive_bayes import GaussianNB

 from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear_model import SGDClassifier

 from sklearn.ensemble import AdaBoostClassifier

 from sklearn.linear_model import RidgeClassifier

 from sklearn.neural_network import MLPClassifier

 from sklearn.discriminant_analysis importQuadraticDiscriminantAnalysis

 from sklearn.multioutput import ClassifierChain

from sklearn.ensemble import BaggingClassifier

 from sklearn.multioutput import MultiOutputClassifier
```

```
 from sklearn.naive_bayes import BernoulliNB
 from sklearn.mixture import GaussianMixture
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
 from sklearn.semi_supervised import LabelSpreading
 from sklearn.linear_model import Perceptron
 from imblearn.combine import SMOTEENN
 from sklearn.preprocessing import StandardScaler
 from sklearn.preprocessing import MinMaxScaler
 from sklearn.preprocessing import RobustScaler
 from sklearn.model_selection import train_test_split
 from sklearn import metrics import pickle
from imblearn.over_sampling import SMOTE
 from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from scikitplot.metrics import plot_roc
from scikitplot.metrics import plot_precision_recall
%matplotlib inline
# Importing training data
df=pd.read_csv('/content/fetal_health.csv')
 # Checking dataset
Df
```

 # Checking the Shape of Data

```
df.shape
```

 # Creating a new dataset of target Variable alone

```
df_tar = pd.DataFrame()
df_tar['fetal_health'] = df['fetal_health']
df_tar.sample()
```

 # Checking for null values in the target variable

```
for i in df_tar.columns:
 print(i,'=',df_tar[i].isnull().sum(),df_tar[i].dtype)
```

#Exploratory Data Analysis:

 # Checking whether the index is unique and in order

```
df.index.is_unique
```

 # Checking for false indexes

```
df.index.duplicated()
```

 # Viewing the sample of the dataframe

```
df.sample(10)
```

 # Checking for data correlation

```
df.corr()
```

 # Checking visually for missing values

```
sns.heatmap(df.isnull(),yticklabels=False,cmap="viridis")
```

 # Checking for missing values

```
for i in df.columns:

print(i,'=',df[i].isnull().sum(),df[i].dtype)
```

 # Checking for Outliers via visualization:

```
for col in df.columns:

if df[col].dtype in ["int64","float64"]:

plt.figure()

df.boxplot([col])

plt.show()
```

 # Visualization of features to check the spread of data:

```
fig = df.hist(figsize = (20,20),color="brown")
```

 # Visualization of data correlation using heatmap

```
plt.figure(figsize=(30,30))

sns.heatmap(df.corr(), annot=True, annot_kws={'size'15},cmap="viridis")

plt.title('Correlation of the Attributes',fontsize=50)

plt.xticks(fontsize=40)

plt.yticks(fontsize=40)
```

 # Visualization of Baseline Fetal Heart Rate Distribution

```
sns.set_palette(palette="coolwarm")

sns.distplot(df['baseline value'])

plt.title('Baseline Fetal Heart Rate Distribution');
```

 # Checking for Unique Values in the dataset

```
for col in df.columns:

print(col,'=',df[col].nunique())
```

 # engineering new categorical target column for Normal and At Risk

fetal health

```
df['fetal_health'].replace([1.000, 2.000, 3.000],[0, 1, 2],inplace=True)
```

 # Data Balancing & Spliting:

 # Seperating Independent and Dependent Variables

```
x=df.drop(labels=['fetal_health'],axis=1)

y=df[['fetal_health']]
```

 # Checking for balance distribution of the dataframe

```
y.value_counts().plot.bar()
```

 # Creating a data balancing module

```
sme = SMOTEENN()
```

 # Balancing the dataset

```
X, Y = sme.fit_resample(x,y)
```

 # Checking for balance distribution of the dataframe

```
Y.value_counts().plot.bar()
```

 # Without Scaling

 # Seperating Training and Testing Data for Model Building

```
x_train,x_test,y_train,y_test=train_test_split(X, Y, test_size=0.2)
```

 # With Standard Scaling

```
ss = StandardScaler()

X_ss=ss.fit_transform(X)

xss_train,xss_test,yss_train,yss_test=train_test_split(X_ss,Y,

test_size=0.2)
```

 # With Min Max Scaling

```
mms = MinMaxScaler()

X_mms=mms.fit_transform(X)

xmms_train,xmms_test,ymms_train,ymms_test=train_test_split(X_mms,

Y, test_size=0.2)
```

 # With Robust Scaling

```
rs = RobustScaler()

X_rs=rs.fit_transform(X)

xrs_train,xrs_test,yrs_train,yrs_test=train_test_split(X_rs,Y,

test_size=0.2)
```

 # Data Visualization

 # Replacing the values of fetal health column for easier access

```
Y['fetal_health'].replace([0, 1, 2],['Normal', 'Suspicion', 'At

Risk'],inplace=True)
```

 # creating a balanced dataframe DF

```
DF=X
```

 # Adding the fetal health column to the dataframe

```
DF['Fetal Health']=Y['fetal_health']
```

 # Visualization of Prolonged Decelerations Vs Fetal Health Outcomes

```
sns.set_palette(palette="RdPu_r")
sns.lineplot(x='prolongued_decelerations',y='fetal_health', data=df,
alpha=1.0)
plt.title('Prolonged Decelerations and Fetal Health Outcomes')
```

 # Visualization of Prolonged Decelerations Vs Fetal Heart Rate

```
sns.set_palette(palette="RdPu")
sns.lineplot(x='prolongued_decelerations',y='baseline
value',hue='fetal_health', data=df, alpha=1.0)
plt.title('Prolonged Decelerations and Fetal Heart Rate')
```

 # Checking the relationship between Fetal Heart Rate Distrubution & Fetal Health

```
sns.set_palette(palette='Accent')
sns.displot(data=DF, x='baseline value', hue='Fetal
Health',hue_order=('Normal','Suspicion','At Risk'), kde=True)
plt.title('Fetal Heart Rate Distrubution vs Fetal Health',fontsize=15)
plt.xlabel("Fetal Heart Rate",fontsize=12)
plt.ylabel('Count',fontsize=12)
```

 # Visualization of data correlation using heatmap

```
plt.figure(figsize=(30,30))
```

```
sns.heatmap(df.corr(), annot=True, annot_kws={'size':
15},cmap="viridis")
plt.title('Correlation of the Attributes',fontsize=50)
plt.xticks(fontsize=40)
plt.yticks(fontsize=40)
```

 # Checking the relatinship between Prolongued Decelerations & Fetal Health

```
sns.set_palette(palette='cool')
sns.barplot(x='prolongued_decelerations',y='Fetal Health', data=DF)
y = [0,1,2]
labels = ['Normal', 'Suspicion','At Risk']
plt.yticks(y, labels,fontsize=13)
plt.title('Prolongued Decelerations Vs Fetal Health',fontsize=20)
plt.gcf().set_size_inches(5,5)
plt.ylabel('Fetal Health',fontsize=15)
plt.xlabel('Prolongued Decelerations',fontsize=15)
plt.show()
```

 # Checking the relatinship between Prolongued Decelerations & each Class of Fetal Health

```
sns.set_palette(palette='Dark2')
sns.catplot(x="prolongued_decelerations", col="fetal_health", data=df,
kind="count", height=4, aspect=0.7)
```

 # Creating a scatter plot for sample from each class

```
sns.set_palette(palette="tab10")
plt.figure(figsize=(10,6))
sns.scatterplot(x="mean_value_of_short_term_variability",
y="mean_value_of_long_term_variability", hue='Fetal Health',
hue_order=('Normal','Suspicion','At Risk'), data=DF)
plt.title('Mean Value of Short-Term and Long-Term Variability in FHR')
```

 # Replacing the values of fetal health column for Modelling

```
Y['fetal_health'].replace(['Normal', 'Suspicion', 'At Risk'], [0, 1,
2],inplace=True)
```

 # Model Building:

 # Decision Tree model build

```
dt = DecisionTreeClassifier(max_depth=5)
```

 # Gaussian Naive Bayes Model

```
gnb = GaussianNB()
```

 # XGBoost Classifier Model Build

```
xgb = XGBClassifier(n_estimators = 100)
```

 # Support Vector Classifier Model Build

```
svc = SVC(kernel ='linear', C = 3)
```

 # K-Nearest Neighbours Model Build

```
knn = KNeighborsClassifier(n_neighbors=15)
```

#Save the models in a variable

```
model_list=[dt, gnb, xgb, svc, knn]
```

#use the function model_list to print the without scaling value

```
def get_score(model_list):
for i in model_list:
print(i)
print('Without Scaling')
i.fit(x_train,y_train)
tr=i.score(x_train,y_train)
print('Train Score:',tr)
te=i.score(x_test,y_test)
print('Test Score: ',te)
print('With Standard Scaling')
i.fit(xss_train,yss_train)
trss=i.score(xss_train,yss_train)
print('Train Score:',trss)
tess=i.score(xss_test,yss_test)
print('Test Score: ',tess)
print('With Min Max Scaling')
i.fit(xmms_train,ymms_train)
trmms=i.score(xmms_train,ymms_train)
```

```
print('Train Score:',trmms)

temms=i.score(xmms_test,ymms_test)

print('Test Score: ',temms)

print('With Robust Scaling')

i.fit(x_train,y_train)

trrs=i.score(xrs_train,yrs_train)

print('Train Score:',trrs)

ters=i.score(xrs_test,yrs_test)

print('Test Score: ',ters)get_score(model_list)
```

 # Model Evaluation:

```
def without_s_score(m):

print(m, 'Without Scaling')

m.fit(x_train,y_train)

y_score = m.predict_proba(x_test)

y_pred = m.predict(x_test)

print("Accuracy: ", accuracy_score(y_test, y_pred))

print(f'Classification Report:\n {metrics.classification_report(y_test,

y_pred)}\n')
```

 # Printing the Confusion Matrix of XGBoost Classifier Model

```
plt.figure(figsize=(6,5))

sns.heatmap(metrics.confusion_matrix(y_test,
```

```
y_pred),annot=True,xticklabels = ['Normal', 'Suspicion','At Risk'] ,
yticklabels = ['Normal', 'Suspicion','At Risk'],cmap='RdPu' )
plt.ylabel('True Values',fontsize=15)
plt.xlabel('Predicted Values',fontsize=15)
plt.title('Accuracy of the Test',fontsize=20)
plt.xticks(fontsize=13)
plt.yticks(fontsize=13)
def with_ss_score(m):
print(m, 'With Standard Scaling')
m.fit(xss_train,yss_train)
y_score = m.predict_proba(xss_test)
y_pred = m.predict(xss_test)
print("Accuracy: ", accuracy_score(yss_test, y_pred))
print(f'Classification Report:\n {metrics.classification_report(yss_test,
y_pred)}\n')
```

 # Printing the Confusion Matrix of XGBoost Classifier Model

```
plt.figure(figsize=(6,5))
sns.heatmap(metrics.confusion_matrix(yss_test,
y_pred),annot=True,xticklabels = ['Normal', 'Suspicion','At Risk'] ,
yticklabels = ['Normal', 'Suspicion','At Risk'],cmap='RdPu' )
plt.ylabel('True Values',fontsize=15)
```

```
plt.xlabel('Predicted Values',fontsize=15)

plt.title('Accuracy of the Test',fontsize=20)

plt.xticks(fontsize=13)

plt.yticks(fontsize=13)

def with_mms_score(m):

print(m, 'With Min Max Scaling')

m.fit(xmms_train,ymms_train)

y_score = m.predict_proba(xmms_test)

y_pred = m.predict(xmms_test)

print("Accuracy: ", accuracy_score(ymms_test, y_pred))

print(f'Classification Report:\n {metrics.classification_report(ymms_test,

y_pred)}\n')
```

 # Printing the Confusion Matrix of XGBoost Classifier Model

```
plt.figure(figsize=(6,5))

sns.heatmap(metrics.confusion_matrix(ymms_test,

y_pred),annot=True,xticklabels = ['Normal', 'Suspicion','At Risk'] ,

yticklabels = ['Normal', 'Suspicion','At Risk'],cmap='RdPu' )

plt.ylabel('True Values',fontsize=15)

plt.xlabel('Predicted Values',fontsize=15)

plt.title('Accuracy of the Test',fontsize=20)

plt.xticks(fontsize=13)
```

```
plt.yticks(fontsize=13)

def with_rs_score(m):

print(m, 'With Robust Scaling')

m.fit(xrs_train,yrs_train)

y_score = m.predict_proba(xrs_test)

y_pred = m.predict(xrs_test)

print("Accuracy: ", accuracy_score(yrs_test, y_pred))

print(f'Classification Report:\n {metrics.classification_report(yrs_test,

y_pred)}\n')
```

 # Printing the Confusion Matrix of XGBoost Classifier Model

```
plt.figure(figsize=(6,5))

sns.heatmap(metrics.confusion_matrix(yrs_test,

y_pred),annot=True,xticklabels = ['Normal', 'Suspicion','At Risk'] ,

yticklabels = ['Normal', 'Suspicion','At Risk'],cmap='RdPu' )

plt.ylabel('True Values',fontsize=15)

plt.xlabel('Predicted Values',fontsize=15)

plt.title('Accuracy of the Test',fontsize=20)

plt.xticks(fontsize=13)

plt.yticks(fontsize=13)
```

 # Evaluating XGBoostclassifier Models

```
without_s_score(xgb)
```

with_mms_score(xgb)

 # Model Selection:

xgb.fit(x_train,y_train)

y_score = xgb.predict_proba(x_test)

y_pred = xgb.predict(x_test)

print("Accuracy: ", accuracy_score(y_test, y_pred))

#Saving the Selected Model:

 # Saving the XGBoost Classifier Model

data={"Model": xgb}

with open('saved_steps.pkl', 'wb') as file:

pickle.dump(data, file)

 # @title Default title text

 # Checking the saved model

with open('saved_steps.pkl','rb') as file:

data = pickle.load(file)

model=data["Model"]

 # Storing one entry of the testing dataset

test=x_test.iloc[7:10,:]

print(model.predict(x_test))

 # Storing the data for testing purposes

test.to_csv('test_csv.csv',index=0)

```
class MyClass:

def _init_(self):

self._my_property = None

@property

def my_property(self):

return self._my_property

@my_property.setter

def my_property(self, value):

# Validation logic can be added here

self._my_property = value

@my_property.deleter

def my_property(self):

del self._my_property

obj = MyClass()

obj.my_property = 10 # Calls the setter method

value = obj.my_property # Calls the getter method

# del obj.my_property # Calls the deleter method

#USER MANAGEMENT

# Importing training data

df_dataset=pd.read_csv('/content/fetal_health.csv')

import pandas as pd
```

```
test_first_row = df_dataset

#drop the last column from the dataset

test_first_row.drop(columns=['fetal_health'], inplace=True)

import pandas as pd

import xgboost as xgb

# Get 21 values as input

values = []

ran=test_first_row.columns

print(ran)

for i in ran:

 value = input(f"Enter value {i}: ")

 values.append(value)

# Create DataFrame variable

column_labels = [f'{i}' for i in ran]

df = pd.DataFrame(data=[values], columns=column_labels)

print("DataFrame:")

#convert the predicted output into float

df=df.astype(float)

# print the output

 print(model.predict(df))
```

REFERENCES

[1] Xie,Y.,Xu,L.,Shen,J.,&Luo,X.(2021). Prediction of maternal health risk using machine learning: A systematic review. Journal of medical systems, 45(7), 1-14.

[2] Garimella, R.,Puppala, S.,Seshadri, S.,&Chandran,A.(2021).Machine learning approaches for maternal health risk prediction: A systematic review. Journal of medical systems, 45(5), 1-12.

[3] Zhang,X.,Wang,J.,Yu,Y.,&Huang,Y.(2021).Prediction of maternal health risks using machine learning models: A systematic review and meta-analysis. Journal of clinical medicine, 10(13), 2756.

[4] Kim,J.H.,&Lee,J.S.(2020).Maternal and fetal risk assessment using machine learning: a review. Healthcare informatics research, 26(2), 77-87.

[5] Liu, Y., Lv, Y., Chen, H., & Sun, X. (2021). Risk prediction models of maternal complications during pregnancy using machine learning algorithms: A systematic review. PLoS one, 16(6), e0252842

[6] Karim, M.N., Islam, M.T., Rahman, M.S., Rahman,M.A., & Kadir, M.A (2021). Maternal Health Risk Prediction using Machine Learning: A comparative study. Journal of Healthcare Engineering, 2021, 1-16.

[7] Ali,M.A.,El-Sappagh,S.,Hussain,M.,&Kwak,D.(2021).Amachine learning approach for early prediction of gestational diabetes mellitus. Journal of Ambient Intelligence and Humanized Computing, 12(5), 4915-4926.

[8] Luo, Y., Wang, Y., Lu, C., Xie, M., & Chen, R. (2020). Predicting gestational hypertension using machine learning: A nationwide population-

based study in China. Journal of biomedical informatics, 107, 103464.

[9] Martínez-Pérez,O.,Torralba-Estelles,J.,Garcia-Simó,I.,Rizo-Baeza, M. M., & Martínez-Roig, A. (2020). Machine learning algorithms for predicting maternal and obstetric complications in pregnant women: A systematic review. PLoS one, 15(10), e0240705.

[10] Shah, R. L., Pooley, J. A., Kumar, A., & Pacheco, L. D. (2020). Prediction of postpartum hemorrhage using machine learning algorithms. American journal of obstetrics and gynecology, 222(5), 478-e1.

[11] Mohseni,M.,&Aryankhesal,A.(2021).Predicting the risk of maternal mortality using machine learning techniques: a systematic review. BMC pregnancy and childbirth, 21(1), 1-16.

[12] Naeem,M.,Ali,M.A.,&Khan,A.A.(2021).Maternal and fetal health risk prediction using machine learning: A systematic review. Computer Methods and Programs in Biomedicine, 2018, 106227.

[13] Shah, R. L., Pooley, J. A., Kumar, A., & Pacheco, L. D. (2020). Prediction of postpartum hemorrhage using machine learning algorithms. American journal of obstetrics and gynecology, 222(5), 478-e1..

[14] Luo, Y., Wang, Y., Lu, C., Xie, M., & Chen, R. (2020). Predicting gestational hypertension using machine learning: A nationwide population-based study in China. Journal of biomedical informatics, 107, 103464.

[15] Antwi, E.,Adu, P., & Affum-Osei, E. (2020). Predictive model for the early identification of maternal health complications using machine learning algorithms. BMC pregnancy and childbirth, 20(1), 1-10.