University of Essex

Department of Mathematical Sciences

MA981 DISSERTATION

PREDICTING FETAL HEALTH CLASSIFICATION USING MACHINE LEARNING

VENKATA KAUTILYA REDDY KAMJULA
2213548

Supervisor: **DAN BRAWN**

November 23, 2023
Colchester

Abstract

Ensuring optimal prenatal care is crucial for fetal health. This study explores advanced machine learning techniques specifically logistic regression, k-nearest neighbors (KNN), decision trees (DT), support vector machines (SVM), and random forest for fetal health assessment. Our primary goals are to identify the most effective machine learning approach and assess how subtle data variations impact prediction accuracy.

Logistic regression demonstrated an accuracy of 86.2%, shedding light on linear relationships in the dataset. KNN, focusing on local patterns, achieved a commendable accuracy of 95.3%, showing its ability to capture subtle variations in fetal health data. Decision trees and support vector machines exhibited accuracies of 95.3% and 94.2%, respectively, each showcasing unique strengths in the context of fetal health assessment. Notably, random forest emerged as the top performer with an accuracy of 98.1%, showcasing its strength in discerning complex relationships.

The study delves into the resilience and adaptability of each method when faced with nuanced data variations, providing insights into their practical strengths and limitations. These findings contribute to the scientific understanding of machine learning in fetal health and offer practical guidance for healthcare practitioners, aiding informed decision-making in prenatal care.

In conclusion, the study establishes random forest as a superior method for fetal health assessment, emphasizing the need for careful consideration in selecting a machine learning approach. These insights hold significant implications for enhancing prenatal care precision and, subsequently, improving outcomes for both mothers and newborns.

Keywords: fetal health, machine learning, logistic regression, k-nearest neighbors, random forest, support vector machine, decision tree, prenatal care.

Acknowledgment

I extend my heartfelt gratitude to my dedicated supervisor, Mr. Dan Brawn, for his unwavering support, invaluable guidance, and constructive feedback throughout the entire journey of this project. His mentorship has been instrumental in shaping the direction and success of my work.

I would like to express my profound appreciation to my parents for their unwavering support and constant encouragement during my study and throughout the process of researching and writing this thesis. Their belief in my abilities has been a driving force behind this accomplishment. This journey would not have been possible without their love and encouragement. Thank you.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Background

Prenatal care holds a pivotal role in modern healthcare, ensuring the well-being of expectant mothers and their infants. According to a study by Alfirevic [1] , Cardiotocography (CTG) has been instrumental in obstetrics, continuously monitoring fetal heart rate (FHR) patterns and uterine contractions. However, traditional CTG interpretation relies on healthcare professionals, introducing subjectivity.

Machine learning (ML) offers a promising solution to enhance CTG analysis. ML algorithms can systematically process extensive CTG data, detect subtle patterns, and provide objective assessments, potentially improving the early detection of fetal distress and reducing false alarms [2].

This project builds upon prior work in medical ML, with the aim of developing customized models that seamlessly integrate into existing CTG systems. We acknowledge the paramount importance of ethical considerations, including data privacy and regulatory compliance, throughout this endeavor [3].

In summary, this project resides at the intersection of medical expertise and technological innovation. By advancing CTG analysis with ML, we intend to empower healthcare providers with a valuable tool that can enhance prenatal care, mitigate risks, and ultimately contribute to better outcomes for both mothers and infants.

1.2 Problem Statement

Cardiotocography (CTG) serves as a pivotal tool in obstetrics, monitoring fetal well-being through continuous assessments of fetal heart rate (FHR) and uterine contractions [1]. However, the inherent subjectivity in CTG interpretation, reliant on healthcare professionals, introduces variability and the potential for errors when assessing fetal

health. Our challenge is to harness the power of machine learning (ML) to augment CTG analysis by creating an ML model capable of automating the analysis of CTG data. This model should identify deviations from normal FHR patterns, aiding in the detection of fetal distress and maternal complications while simultaneously reducing the occurrence of false alarms. Our goal is to provide healthcare professionals with an ML tool that complements their expertise, facilitating more precise decision-making throughout pregnancy and labor. Moreover, we are committed to ensuring that our ML-based CTG analysis system adheres to rigorous ethical standards and regulatory requirements, including healthcare regulations like HIPAA or GDPR [3]. This project signifies the convergence of medical knowledge and ML innovation, with the ultimate objective of enhancing the quality of prenatal care and mitigating the risks associated with fetal distress.

1.3 Research Questions

- Which CTG datasets, either publicly available or collected for this study, can provide valuable novel insights and data for the development of an enhanced CTG analysis system using machine learning?

- What are the critical features within CTG recordings that must be carefully selected and retained, while identifying and eliminating non-informative ones, to optimize the performance of the machine learning classifier in assessing fetal well-being during pregnancy and labor?

- What specific types of fetal distress patterns and maternal complications will be the primary focus of detection in the CTG analysis, and which machine learning classification approaches or algorithms will be most suitable for accurately identifying and classifying these patterns?

- What performance metrics will be employed to rigorously validate the proposed machine learning-based CTG analysis system, ensuring its accuracy, efficiency, and effectiveness in improving prenatal care and reducing false alarms during pregnancy and labor?

1.4 Research Aim and Objectives

This study aims to use machine learning to enhance Cardiotocography (CTG) analysis, with the goal of improving fetal well-being assessment during pregnancy and labor, reducing false alarms, and assisting healthcare providers in decision-making.

- To identify and select appropriate Cardiotocography (CTG) dataset, whether publicly available or collected for this study, that provide novel insights and data for the development of an enhanced CTG analysis system using machine learning.

- To specify the types of fetal distress patterns and maternal complications that will be the primary focus of detection in the CTG analysis, and to select and implement machine learning classification approaches or algorithms that are most suitable for accurately identifying and classifying these patterns.

- To establish and employ performance metrics that rigorously validate the proposed machine learning-based CTG analysis system, ensuring its accuracy, efficiency, and effectiveness in improving prenatal care and reducing false alarms during pregnancy and labor.

CHAPTER 2

Literature Survey

2.1 Fetal Health Monitoring and Cardiotocography

Fetal health monitoring during labor, primarily utilizing cardiotocography (CTG), is a crucial component of contemporary obstetric care. The evolution of fetal monitoring techniques, progressing from historical methods to the current standardization of CTG, reflects technological advancements and a commitment to enhancing perinatal outcomes [4]. National guidelines, such as those from the National Institute for Health and Care Excellence (NICE), provide essential recommendations for intrapartum care and interpreting CTG data. Practice Bulletin No. 106 by the American College of Obstetricians and Gynecologists (ACOG) offers valuable insights into the nomenclature, interpretation, and general management principles of fetal heart rate monitoring. Despite the widespread adoption of CTG, ongoing research, including investigations into its correlation with cerebral palsy incidence, raises questions about its effectiveness and necessitates a reevaluation of its role in modern obstetric practice [2]. As illuminated in the review by Parer and Ikeda, electronic fetal monitoring brings both advantages and limitations, introducing nuances to the ongoing discourse. The field of maternal-fetal medicine has long been dedicated to ensuring the well-being of expectant mothers and their unborn children. While traditional methods like ultrasound examinations and fetal heart rate monitoring have been integral to routine prenatal care, they face limitations, especially in providing early and precise indications of potential complications [4]. The subjective nature of interpretation and inherent variability in human assessment highlight the necessity for more objective and sophisticated approaches in fetal health assessment [5]. Addressing these controversies and comprehending emerging technologies will shape the future of fetal health monitoring, guiding clinicians in their pursuit of optimal perinatal care.

2.2 Data Sources and Types

2.2.1 Ultrasound Imaging

Ultrasound imaging has long been a fundamental tool in fetal health assessment, offering real-time and non-invasive visualization of the developing fetus. This modality provides detailed insights into structural aspects and enables the monitoring of fetal growth [4]. Recent applications of machine learning in conjunction with ultrasound data have demonstrated potential in automating analysis, facilitating the early detection of anomalies, and improving predictions related to developmental outcomes [6].

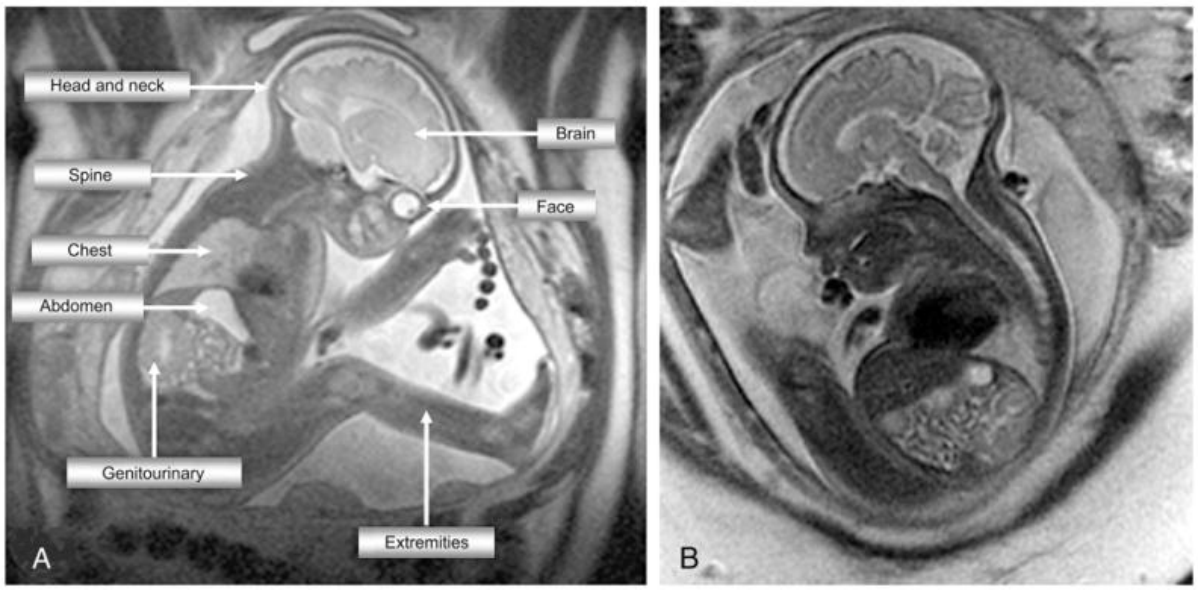

Figure 2.1: Fetal magnetic resonance imaging and Fetal magnetic resonance imaging
Source: NCBI.com

2.2.2 Maternal Health Records

Maternal health records serve as a comprehensive repository of information, encompassing aspects such as medical history, lifestyle factors, and relevant biomarkers. Integrating machine learning models with maternal health data contributes to a more nuanced understanding of the intricate relationship between maternal influences and fetal health outcomes [7].

2.2.3 Fetal Heart Rate Monitoring

Fetal heart rate monitoring, commonly conducted through methods like cardiotocography (CTG), provides vital insights into the fetal cardiovascular system and oxygenation levels. Machine learning models can analyze patterns within fetal heart rate data, offering valuable insights into fetal well-being and enabling the potential identification of distress conditions [8]. Combining this data with information from other sources enhances the overall predictive capacity of these models.

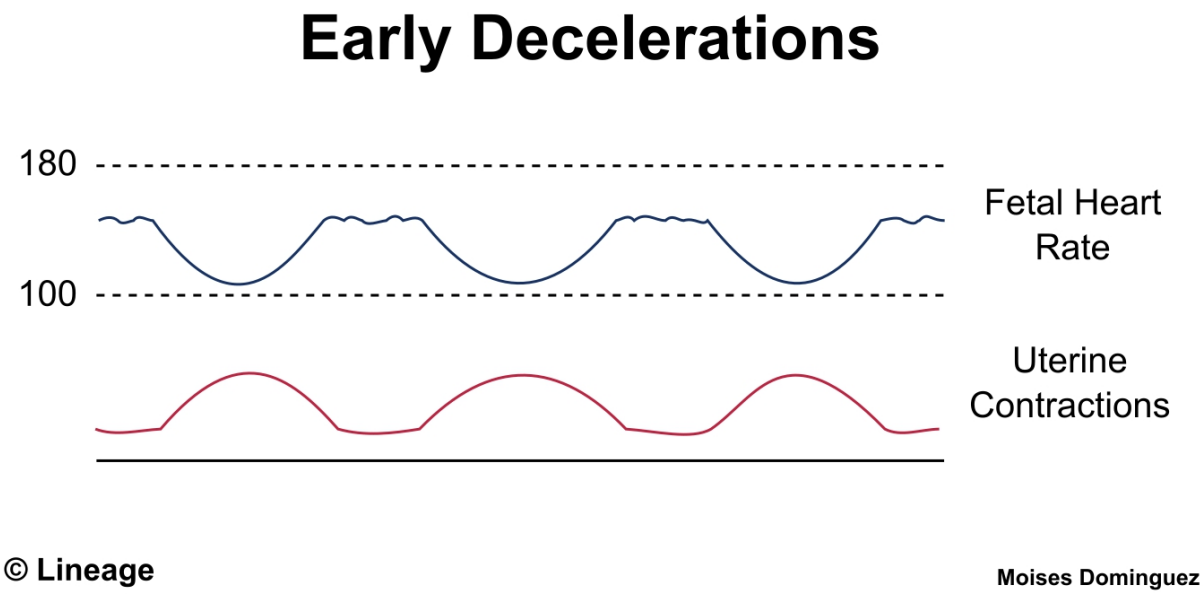

Figure 2.2: Early detections of FHR and UC
Source: medbullets.com

2.2.4 Multi-Modal Data Integration

An emerging trend involves the integration of data from multiple modalities, such as combining ultrasound images with maternal health records or fetal heart rate monitoring. This multi-modal approach aims to provide a more comprehensive understanding of fetal health by simultaneously considering diverse factors [9]. The synergy derived from information across different sources enhances the robustness and accuracy of predictive models.

2.2.5 Genetic and Molecular Data

Advancements in genetic and molecular profiling have introduced a molecular-level perspective into the understanding of fetal health. Techniques such as DNA sequencing and the analysis of fetal DNA in maternal blood offer insights into genetic disorders and susceptibility. Machine learning models applied to genetic and molecular data contribute to personalized and precise fetal health predictions, enabling early identification of genetic anomalies [10][11].

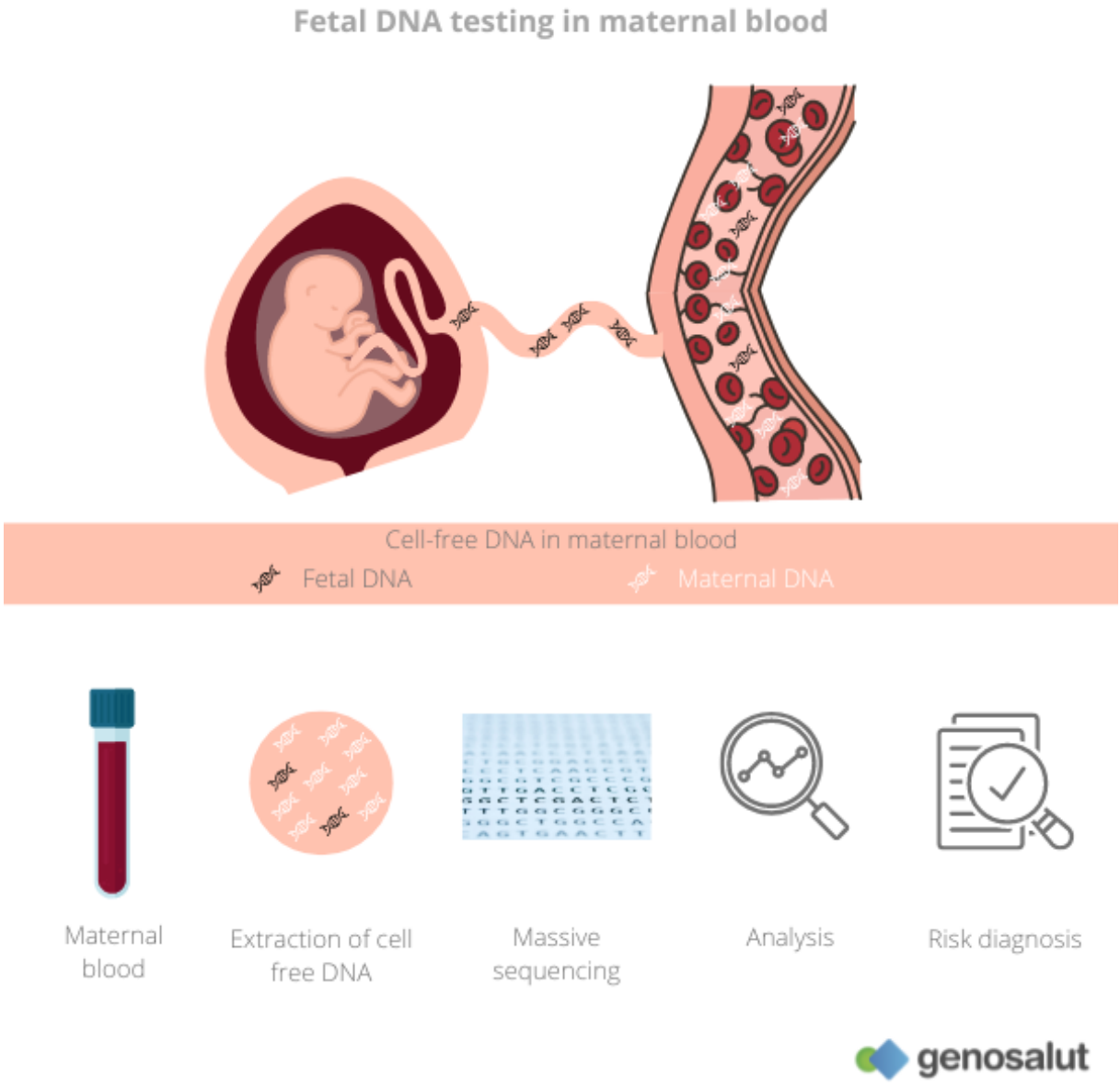

Figure 2.3: Fetal DNA testing

Source: genosalut.com

In summary, the diverse array of data sources, ranging from traditional imaging modalities to advanced genetic profiling, provides a rich foundation for the application of machine learning in predicting fetal health. Understanding the distinct characteristics of each data type and exploring their complementary nature is pivotal for the development of comprehensive and accurate predictive models.

2.3 Machine Learning Algorithms

Machine learning algorithms play a pivotal role in the advancement of predicting fetal health, offering diverse tools to analyze complex datasets and extract meaningful patterns. This section provides an in-depth exploration of various machine learning approaches applied in the context of fetal health classification.

2.3.1 Supervised Learning Algorithms

Logistic regression remains a fundamental supervised learning algorithm in fetal health prediction. It is particularly effective in binary classification tasks, such as identifying normal and abnormal fetal health states. Logistic regression models provide a transparent understanding of the relationship between input features and the likelihood of a specific outcome [12].

Support Vector Machines (SVM) are powerful classifiers widely utilized in fetal health research. SVMs excel in handling high-dimensional data, making them suitable for tasks involving complex feature spaces. Researchers have successfully applied SVMs to classify fetal health based on various data sources, including ultrasound images and maternal health records [13][14]

Random Forests, an ensemble learning method, have gained popularity in fetal health prediction due to their ability to handle non-linearity and capture complex relationships within heterogeneous datasets. By aggregating predictions from multiple decision trees, Random Forests offer improved accuracy and robustness, making them well-suited for the intricacies of fetal health classification [15].

2.3.2 Deep Learning Architectures

Convolutional Neural Networks (CNNs) have revolutionized image analysis and interpretation, finding significant applications in fetal health research. By automatically learning hierarchical features from ultrasound images, CNNs enhance the accuracy of classifying structural anomalies and abnormalities, contributing to early diagnosis [9].

In tasks involving sequential data, such as fetal heart rate time series analysis, **Recurrent Neural Networks (RNNs)** demonstrate prowess. RNNs capture temporal dependencies in data, allowing for more accurate predictions and nuanced insights into fetal cardiovascular dynamics [7].

2.3.3 Explainable Machine Learning Models - Decision Trees

In medical applications, the interpretability of machine learning models is crucial for gaining the trust of healthcare professionals. Decision trees offer a transparent, rule-based structure that allows for a clear understanding of the decision-making process. These models provide insights into the key features influencing fetal health classification, aiding in clinical decision support systems [4].

2.3.4 Pre-trained Models

Transfer learning has emerged as a promising approach in fetal health classification, leveraging knowledge gained from pre-trained models on large medical imaging datasets. Pre-trained models, often trained on diverse tasks, can be fine-tuned for specific fetal health prediction tasks. This utilization of prior knowledge enhances the model's ability to generalize to specific applications, addressing data limitations and improving overall performance [2][16].

In summary, the selection and integration of various machine learning algorithms offer a comprehensive toolkit for predicting fetal health. Researchers can leverage the strengths of different approaches based on the characteristics of the data and the specific requirements of fetal health classification tasks.

2.4 Evaluation Metrics

2.4.1 Sensitivity and Specificity

Sensitivity and specificity are crucial metrics in assessing the performance of fetal health prediction models. Sensitivity, also known as the true positive rate, measures the model's ability to correctly identify cases with positive outcomes, such as the presence of fetal abnormalities. Specificity, on the other hand, represents the true negative rate and measures the accuracy in identifying cases without the condition, such as correctly classifying healthy fetuses. Balancing sensitivity and specificity is essential for achieving optimal diagnostic performance and minimizing false positives and false negatives in clinical applications [17].

2.4.2 Area Under the Receiver Operating Characteristic Curve (AUC-ROC)

The AUC-ROC is a widely used metric that evaluates the discriminatory power of a model across various classification thresholds. It represents the area under the receiver operating characteristic curve and provides a comprehensive overview of the model's ability to distinguish between different classes. A higher AUC-ROC value indicates better overall performance in fetal health classification. Researchers often use this metric to fine-tune models and achieve an optimal balance between sensitivity and specificity [18][19][20]

2.4.3 Positive Predictive Value (PPV) and Negative Predictive Value (NPV)

Positive Predictive Value (PPV) and Negative Predictive Value (NPV) offer insights into the accuracy of positive and negative predictions, respectively. PPV measures the probability that a positive prediction is correct, while NPV measures the probability that a negative prediction is correct. These metrics are crucial in clinical decision-making, providing information on the reliability of the model's predictions and guiding healthcare professionals in interpreting the results for effective patient care [9][18].

2.5 Key Findings

2.5.1 Integration of Multiple Data Modalities

A recurring theme in the literature reveals the significance of integrating multiple data modalities for fetal health prediction. Studies consistently showcase the effectiveness of combining information from diverse sources, such as ultrasound imaging, maternal health records, and fetal heart rate monitoring. This multi-modal approach proves instrumental in enhancing the accuracy and reliability of predictive models, providing a more holistic understanding of fetal health dynamics [7][21].

2.5.2 Deep Learning Advancements

Advancements in deep learning, particularly the application of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have demonstrated remarkable success in fetal health prediction tasks. These architectures excel in tasks like ultrasound image analysis and fetal heart rate time series interpretation. Their ability to automatically learn intricate patterns from complex data contributes to improved accuracy and efficiency in fetal health classification [2][9].

2.5.3 Transfer Learning for Fetal Health Prediction

The exploration of transfer learning approaches in fetal health prediction yields promising outcomes. Studies suggest that pre-training models on large medical imaging datasets and fine-tuning them for fetal health prediction tasks leverage knowledge from broader contexts. This transfer of knowledge enhances the efficiency and effectiveness of models in specific applications, contributing to improved performance and generalizability [22][4]

2.6 Methodological Critique

2.6.1 Dataset Representativeness and Size

A critical aspect of the methodological critique in fetal health prediction studies revolves around the representativeness and size of the datasets used. Challenges related to imbalanced datasets or limited sample sizes can impact the generalizability of the developed models to diverse populations. Ensuring a representative and sufficiently large dataset is crucial for building robust models with broader applicability [14][23]

2.6.2 Model Interpretability and Transparency

The interpretability and transparency of machine learning models in fetal health prediction are essential for gaining acceptance in clinical practice. Healthcare professionals require models that offer clear insights into decision-making processes. Models such as decision trees or rule-based systems contribute to better understanding and trust among clinicians, facilitating the integration of predictive models into clinical practice [7][4].

2.6.3 Validation and Generalization Across Populations

Ensuring robust validation and generalization of models across different populations is a methodological challenge in fetal health prediction. Some studies may lack external validation on independent datasets, raising concerns about the generalizability of proposed models to diverse demographic and clinical settings. Addressing this challenge is crucial for ensuring the reliability and applicability of predictive models in real-world scenarios [2][22].

2.7 Cardiotocography (CTG) for Fetal Monitoring

Cardiotocography (CTG) is pivotal in obstetrics, providing continuous assessment of fetal health before and during labor through ultrasound wave-based transducers on the mother's abdomen [1]. Monitoring fetal heart rate (FHR) and uterine contractions (UC), CTG evaluates key indicators like FHR variability, responsiveness, and the presence

of decelerations. Maintaining FHR within 110-160 bpm is crucial to detect deviations indicating chronic hypoxia [24].

Accurate CTG interpretation is vital to prevent complications and legal disputes. Staff training programs, incorporating tools like "pulse oximetry" or "fetal electrocardiogram," enhance interpretation [25]. Data mining and machine learning, exemplified by the Intelligent Heart Disease Prediction System (IHDPS) achieving 95% accuracy [26], revolutionize healthcare.

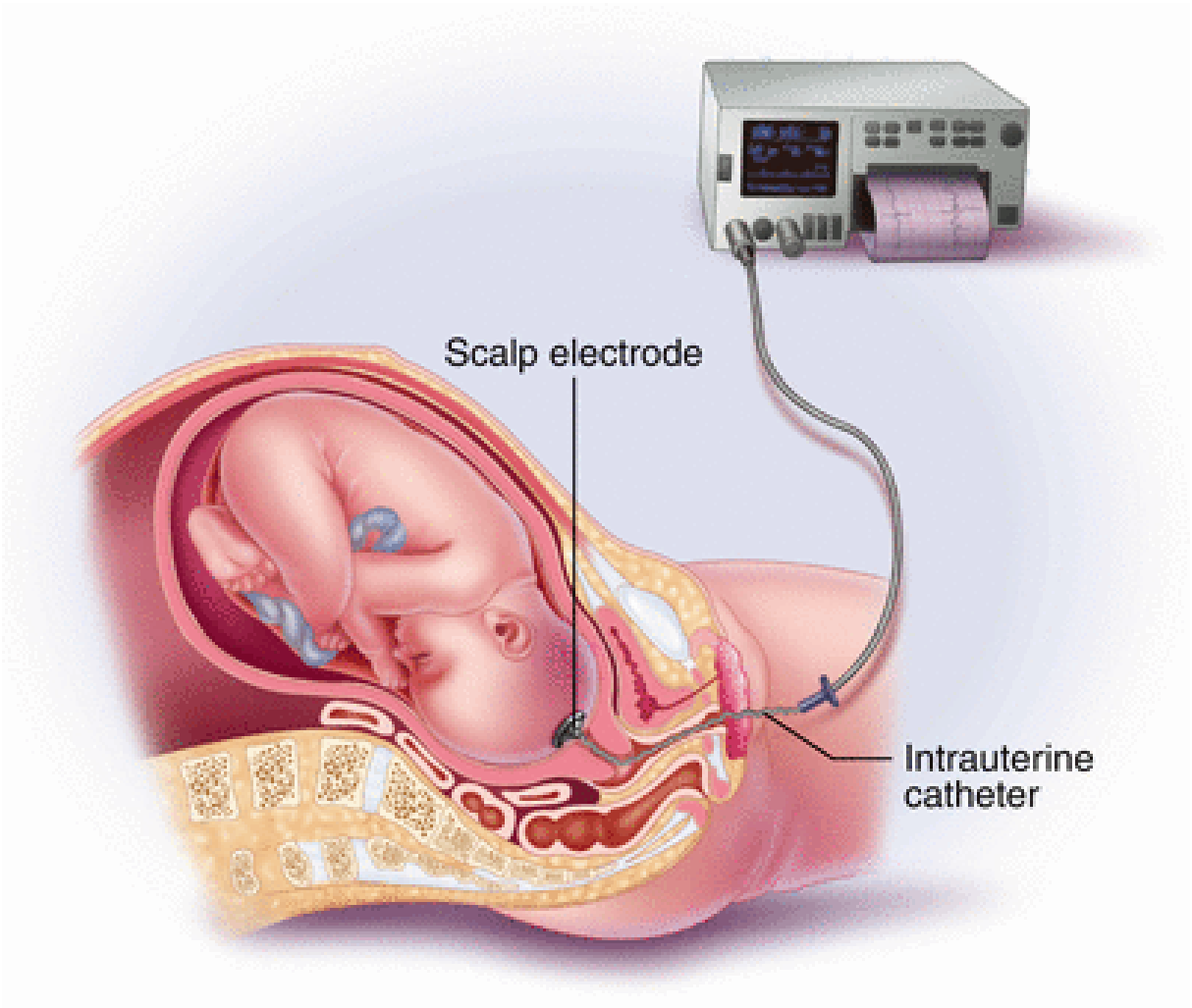

Figure 2.4: Fetal Monitor

Source: springer.com

Automated systems, based on computerized CTG, reduce cerebral palsy and perinatal mortality [27], enhancing diagnostic precision and interventions.

In summary, CTG is indispensable in obstetrics, providing a comprehensive fetal health assessment. Integrating data mining and machine learning enhances diagnostic

precision, offering new avenues for continual improvement [28].

Addressing maternal and fetal health challenges is crucial globally. Maternal mortality, especially in resource-limited conditions, necessitates innovative approaches for fetal well-being (WHO, 2015). Complications like hypertension, gestational diabetes, infections, and preterm labor underline the need for advanced fetal health assessment methods [29][30]

Machine learning applications leverage diverse datasets and algorithms, offering nuanced insights into fetal health dynamics. The intersection of CTG monitoring and machine learning presents a unique opportunity to tackle challenges, contributing to improved outcomes for mothers and infants.

2.8 Previous Research on CTG Analysis Using ML

Previous investigations into Cardiotocography (CTG) analysis using machine learning have demonstrated the potential of ML techniques in advancing fetal health monitoring. Approaches such as support vector machines, neural networks, and random forests have been employed to assess cardiotocograms, showing efficacy in evaluating fetal status [31][32][13]. The diversity of models, including deep learning techniques, contributes to enhancing the precision of fetal health assessment, marking a notable evolution in CTG analysis [21][9].

2.9 Research Gap

The current state of ML applications in CTG analysis reveals notable research gaps that warrant further investigation. Challenges in generalization across diverse patient populations, interpretability of models, and integration into clinical workflows persist. The need for real-time monitoring solutions in CTG analysis during labor and delivery, ethical considerations, and addressing privacy concerns are essential aspects that require comprehensive exploration [3][9]. Closing these research gaps is imperative for advancing the field and ultimately enhancing maternal and fetal health outcomes.

We have used five key machine learning algorithms Logistic Regression, Random Forest, K-Nearest Neighbors (KNN), Decision Tree, and Support Vector Machine (SVM) which will play a pivotal role in predicting fetal health classification. These algorithms

have been selected based on their diverse strengths and suitability for addressing specific aspects of fetal health assessment.

CHAPTER 3

Research Methodology

In this chapter, we aim to present a comprehensive overview of the research and prior investigations that serve as the basis for the methodologies applied in your study. The chapter details the systematic approach used for data collection, analysis, and model construction, elucidating specific aspects such as data sources, sampling methods, data processing, and the statistical or computational approaches employed. The subsequent subsections within this chapter provide an in-depth exploration of the techniques implemented in your study.

3.1 Synthetic Minority Over-sampling Technique (SMOTE)

In addressing class imbalance, one notable technique employed is the Synthetic Minority Over-sampling Technique (SMOTE) [33]. Class imbalance, where one class significantly outnumbers the other, can hinder the performance of machine learning models, particularly in medical datasets where pathological cases might be underrepresented.

SMOTE operates by generating synthetic instances of the minority class to balance the class distribution. This oversampling technique involves creating synthetic samples by interpolating between existing minority class instances. The synthetic sample generation is based on the following equation:

$$\text{SMOTE}(x_i) = x_i + \delta \times (\text{neighbor} - x_i) \tag{3.1}$$

where x_i is the original minority class instance, δ is a random value between 0 and 1, and neighbor is a randomly selected neighboring instance from the minority class.[33][14]

By introducing synthetic instances, SMOTE aims to provide the model with a more balanced and representative training dataset. The algorithm's effectiveness lies in its ability to enhance the model's generalization and predictive capabilities for the minority

class, mitigating the impact of imbalanced class distribution. However, it is crucial to implement SMOTE judiciously, considering the specific characteristics of the dataset and the potential for overfitting.

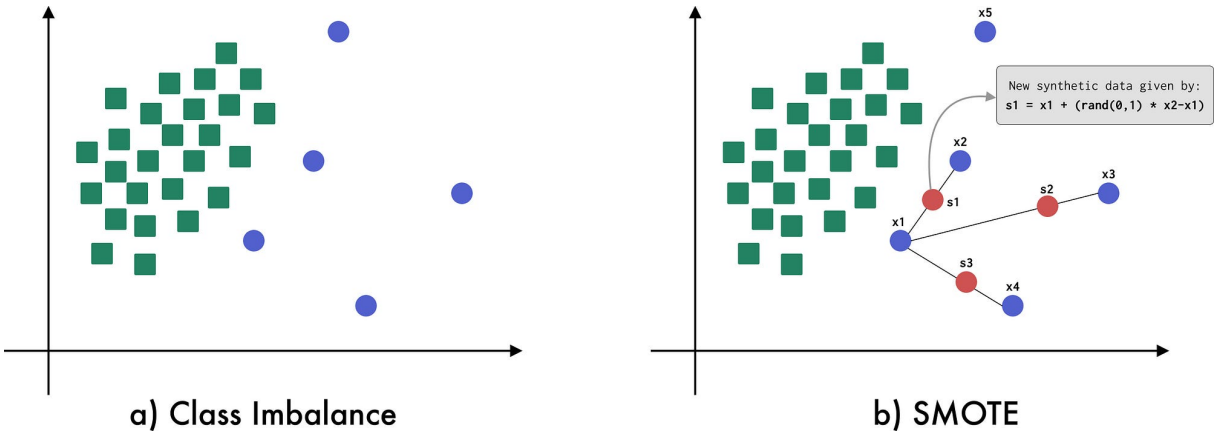

Figure 3.1: SMOTE Data augmentation

Source: towardsdatascience.com

The utilization of SMOTE in this study is motivated by its proven success in enhancing the performance of machine learning models dealing with imbalanced datasets. The following sections delve into the specifics of how SMOTE was integrated into the research methodology, emphasizing its role in addressing class imbalance and improving the robustness of the predictive models.[33]

3.2 Logistic Regression

Logistic Regression is a well-established algorithm for binary classification tasks, making it suitable for predicting fetal health classifications (Normal, Suspect, Pathological). Its linear nature allows for easy interpretation of the relationship between features and the log-odds of the response variable, particularly valuable in medical contexts where interpretability is crucial.

Algorithm Overview:

The fundamental principle behind Logistic Regression is rooted in modeling the probability that a given instance belongs to a particular category. In our case, it estimates the probability of a fetal health classification. Let's denote the binary response variable as Y (where $Y = 1$ indicates the presence of a specific health condition), and the features

influencing this response as $X_1, X_2, ..., X_n$ [34]. The logistic function, also known as the sigmoid function, is a key component:

$$P(Y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n)}}$$

Here:

- $P(Y = 1)$ is the probability that the response variable Y is equal to 1.
- e is the base of the natural logarithm.
- $\beta_0, \beta_1, ..., \beta_n$ are the coefficients, which Logistic Regression estimates during the training process.

The logistic function ensures that the predicted probabilities range between 0 and 1. The log-odds (logit) of the probability is then modeled linearly in terms of the features:

$$\ln\left(\frac{P}{1 - P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n$$

Here, $\ln$ is the natural logarithm, P is the probability of the event $Y = 1$, and $1 - P$ is the probability of the event $Y = 0$. This equation illustrates the linear relationship between the features and the log-odds of the response variable [35][32]

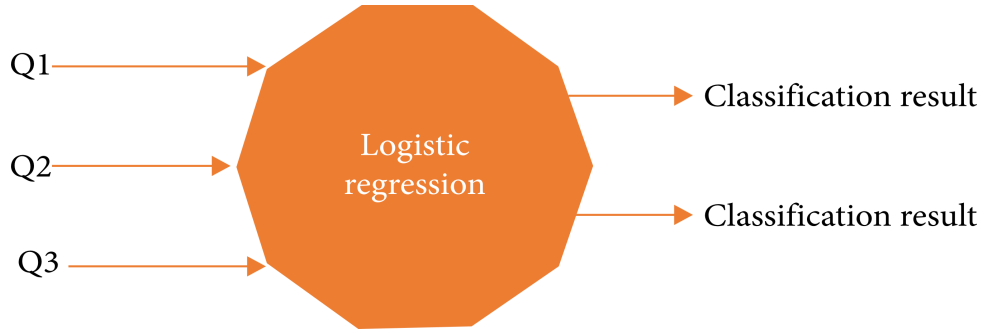

Figure 3.2: Logistic Regression Flowchart

Source: hindawi.com

Interpretability and Limitations:

One significant advantage of Logistic Regression in medical contexts is its inherent interpretability. The coefficients (β) provide insights into the impact of each feature on the log-odds of the response variable. Healthcare professionals can easily grasp and communicate these relationships, contributing to the model's trustworthiness.

However, it's crucial to acknowledge Logistic Regression's limitation in capturing complex nonlinear relationships. The algorithm assumes a linear decision boundary, and if the true relationship in the data is highly nonlinear, Logistic Regression might struggle to capture these intricacies. This trade-off emphasizes the importance of considering the nature of the data and the underlying relationships when choosing classification algorithms.[36][35]

In our fetal health classification task, Logistic Regression offers a balance between interpretability and model performance, providing a solid foundation for further exploration alongside more complex algorithms like Random Forest and K-Nearest Neighbors (KNN).

3.3 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a versatile algorithm frequently used for classification tasks, including predicting fetal health classifications (Normal, Suspect, Pathological). Unlike parametric methods like Logistic Regression, KNN is non-parametric and makes predictions based on the majority class of its k-nearest neighbors in the feature space.[37][38]

Algorithm Overview:

The core idea behind KNN is to classify a new instance based on the classes of its nearest neighbors. Let $X_1, X_2, ..., X_n$ represent the features of an instance and Y be the corresponding response variable (fetal health classification). The algorithm follows these steps:

1. **Calculate Distances:** Measure the distance between the new instance and all other instances in the training set. Common distance metrics include Euclidean distance or Manhattan distance.

2. **Identify Neighbors:** Select the k instances with the shortest distances to the new instance.

3. **Majority Vote:** Assign the class label to the new instance based on the majority class of its k nearest neighbors.

Parameter:

The crucial parameter in KNN is k, the number of neighbors considered for classification. The choice of k influences the model's sensitivity to local fluctuations. Smaller values of k make the model more sensitive but might lead to overfitting, while larger values result in a smoother decision boundary.[39]

Applicability and Considerations:

KNN is effective when the decision boundary is complex and highly nonlinear. Its ability to adapt to local patterns makes it suitable for a wide range of datasets. However, KNN's computational cost increases with larger datasets, and the algorithm's performance can be sensitive to irrelevant or noisy features.[38][40]

Equation:

The predicted class for a new instance (Y) using KNN can be represented as:

$$Y = \text{majority}(\{Y_1, Y_2, ..., Y_k\})$$

Here, $Y_1, Y_2, ..., Y_k$ are the classes of the k nearest neighbors.

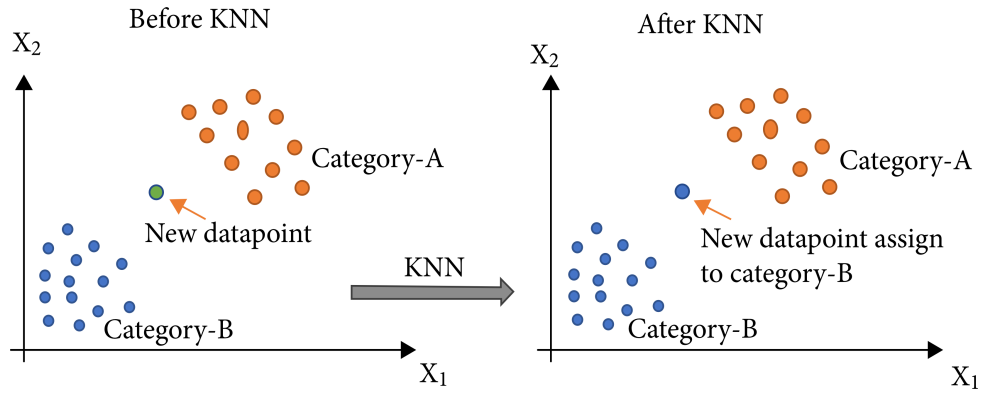

Figure 3.3: KNN flowchart

Source: hindawi.com

Interpretability and Trade-offs:

Interpreting the decision-making process of KNN can be challenging due to its non-parametric nature. While it excels in capturing complex relationships, it may struggle with interpretability compared to Logistic Regression.

In our fetal health classification task, KNN complements Logistic Regression and contributes to a comprehensive analysis. The choice between these algorithms depends on the nature of the data and the desired balance between interpretability and model complexity.

3.4 Random Forest

Random Forest is a powerful ensemble learning algorithm widely applied in various classification tasks, including the prediction of fetal health classifications (Normal, Suspect, Pathological). This algorithm combines the strengths of multiple decision trees to enhance predictive accuracy and generalization.

Algorithm Overview:

The Random Forest algorithm consists of a collection of decision trees, each trained on a random subset of the training data and using a random subset of features. The predictions of individual trees are then aggregated through voting (classification) or averaging (regression) to make the final prediction. This ensemble approach mitigates overfitting and improves the model's robustness.[41]

Decision Tree Strengths:

The decision trees within a Random Forest are adept at capturing complex relationships in the data. Each tree learns different aspects, and their combination leads to a more robust and accurate model. The randomization during tree construction also introduces diversity, preventing the ensemble from relying too heavily on specific features or patterns.[42]

Parameter:

The crucial parameters in a Random Forest include the number of trees (N) and the number of features considered for splitting at each node. A higher number of trees generally improves performance, but it comes at the cost of increased computational complexity.[41][14]

Applicability and Considerations:

Random Forest is suitable for high-dimensional datasets and can handle missing values and outliers effectively. Its ensemble nature contributes to improved performance, making it resilient to noise and outliers present in the data.

Equation:

The predicted class for a new instance (Y) in a Random Forest can be represented as:

$$Y = \text{majority}(\{Y_1, Y_2, ..., Y_N\})$$

Here, $Y_1, Y_2, ..., Y_N$ are the predictions of individual decision trees in the ensemble.

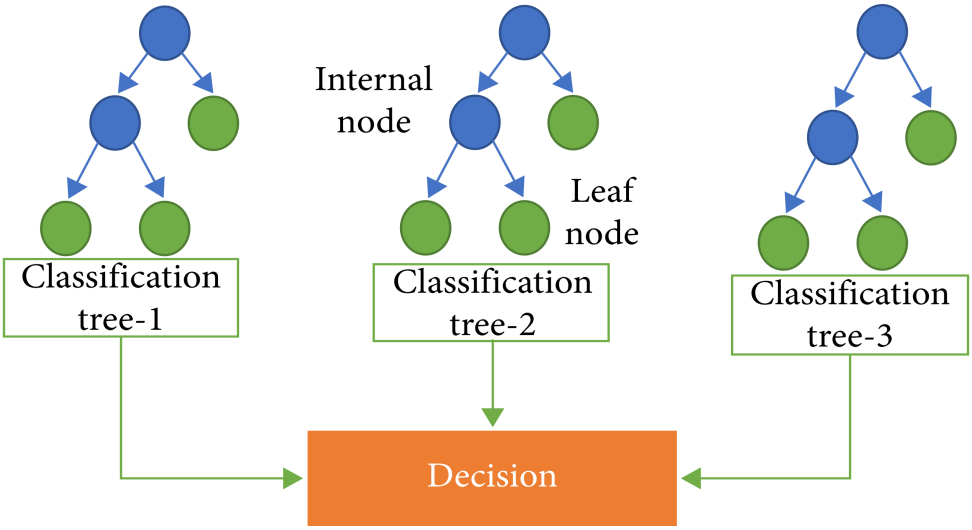

Figure 3.4: Random Forest Flowchart

Source: hindawi.com

Interpretability and Trade-offs:

While Random Forest offers high predictive accuracy, interpreting the combined decision-making process of multiple trees can be challenging. The model's complexity may limit its interpretability compared to simpler algorithms like Logistic Regression.[41]

In our fetal health classification task, Random Forest complements Logistic Regression and KNN, contributing to a diverse and well-rounded analysis. The ensemble nature of Random Forest enhances the robustness of our predictive model.

3.5 Decision Tree

Decision Trees are a powerful tool in machine learning for classification tasks. In the context of fetal health classification, Decision Trees offer transparency and interpretability, providing valuable insights into the decision-making process.

The motivation behind employing Decision Trees lies in their ability to represent decision logic in a human-readable form. This transparency allows clinicians to understand the factors influencing the classification of fetal health. Decision Trees are particularly useful when the relationships between features and outcomes are straightforward.[43]

Algorithm Overview:

The Decision Tree algorithm recursively splits the dataset based on features, cre-

ating a tree-like structure where each internal node represents a decision based on a feature, and each leaf node represents the final classification. This hierarchical structure facilitates a clear interpretation of how the model arrives at a specific prediction.[44][45]

Integration Process:

In the standalone approach, the Decision Tree is trained on the fetal health dataset, and predictions are made based on the learned decision rules. The interpretability of the Decision Tree allows for a direct understanding of the criteria influencing each classification.[44]

Equation:

The prediction (Y) from the Decision Tree can be expressed as:

$$Y = \text{Decision Tree prediction based on learned rules}$$

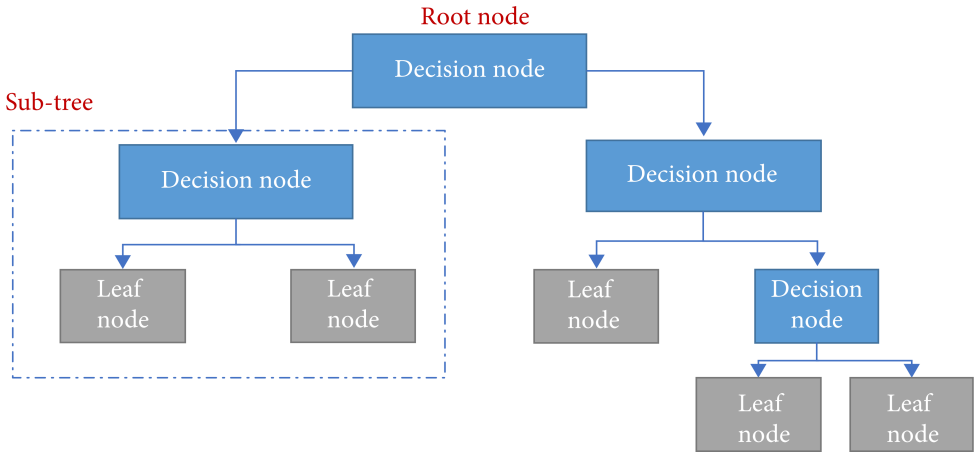

Figure 3.5: Decision Tree Flowchart

Source: hindawi.com

Interpretability and Trade-offs:

Decision Trees are advantageous for their interpretability and ease of understanding. They are well-suited for scenarios where clear decision paths are crucial for clinical interpretation. However, they may struggle with capturing complex relationships present in intricate datasets.[45]

In fetal health classification, Decision Trees provide a transparent framework for understanding the criteria leading to specific predictions. This interpretability is valuable in a clinical context where the rationale behind predictions is as important as the predictions themselves.

The Decision Tree method offers a straightforward and interpretable approach to fetal health classification. While it may not capture intricate patterns as effectively as some complex models, its transparency makes it a valuable tool in scenarios where understanding the decision logic is paramount.

3.6 Support Vector Machine (SVM)

Support Vector Machines (SVM) are a robust class of machine learning algorithms widely used for classification tasks. In the realm of fetal health classification, SVMs bring their strength in handling intricate decision boundaries and complex datasets.

The motivation behind incorporating Support Vector Machines lies in their ability to effectively handle high-dimensional data and capture intricate patterns. SVMs are well-suited for scenarios where the decision boundaries between different classes are complex and non-linear.[46]

Algorithm Overview:

The SVM algorithm works by finding the optimal hyperplane that separates different classes in the feature space. It aims to maximize the margin between classes, identifying the support vectors that play a crucial role in determining the decision boundary. This ability to capture complex relationships makes SVMs particularly effective for fetal health classification.[47][46]

Integration Process:

In the standalone approach, the SVM is trained on the fetal health dataset, learning the optimal decision boundary. The SVM's predictive power lies in its ability to handle intricate patterns, making it a valuable tool in scenarios where the relationships between features and outcomes are intricate.[48]

Equation:

The prediction (Y) from the SVM can be expressed as:

$$Y = \text{SVM prediction based on the learned optimal hyperplane}$$

Interpretability and Trade-offs:

Support Vector Machines excel in capturing complex decision boundaries and are effective in scenarios where the relationships between features and outcomes are intricate.

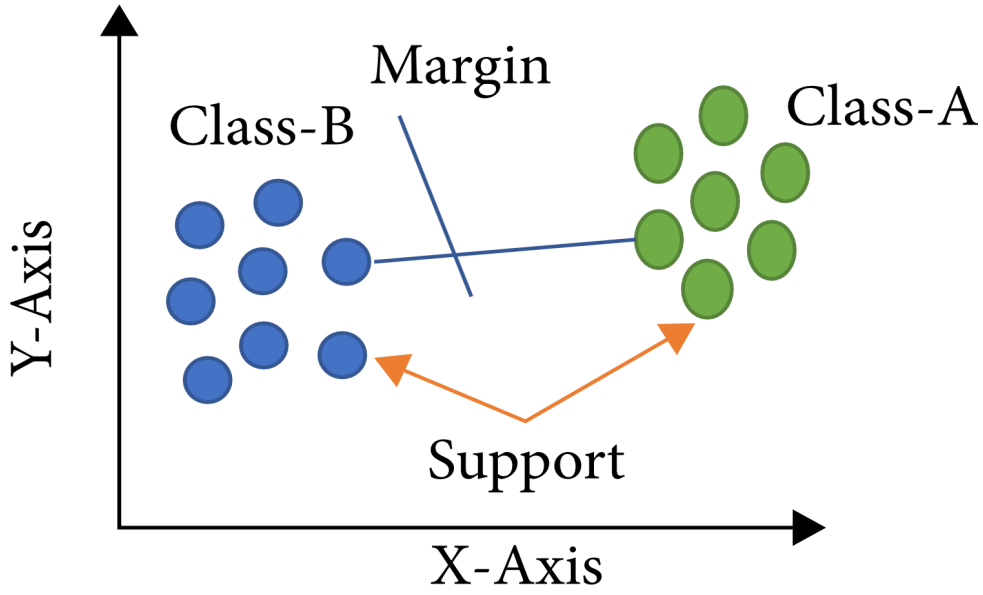

Figure 3.6: Support Vector Machine Flowchart

Source: hindawi.com

However, they may not provide as straightforward interpretability as simpler models like Decision Trees.[48]

In fetal health classification, SVMs are particularly valuable when dealing with intricate patterns and complex relationships between fetal health features. They contribute to accurate predictions in scenarios where other models might struggle.

The SVM method offers a robust approach to fetal health classification, especially in scenarios with intricate and non-linear relationships. While the interpretability might not be as direct as some simpler models, the capacity to handle complex patterns makes SVMs a valuable addition to the ensemble of models for fetal health assessment.

CHAPTER 4

Data Presentation and Exploratory Data Analysis

4.1 Dataset

The significance of reducing child mortality is underscored by its pivotal role in the United Nations' Sustainable Development Goals, serving as a key indicator of human progress. The UN ambitiously envisions that by 2030, countries should eliminate preventable deaths among newborns and children under 5 years, with the overarching goal of reducing under-5 mortality to at least as low as 25 per 1,000 live births. (**WHO**)

Concurrently, maternal mortality poses a formidable global challenge, with 295,000 deaths recorded during and following pregnancy and childbirth as of 2017. A staggering 94% of these maternal deaths occurred in low-resource settings, emphasizing their largely preventable nature.[49]

To contribute to the ongoing efforts in understanding and addressing these critical issues, the dataset utilized in this study was sourced from **UCI Machine Learning Repository**, with **Kaggle** serving as the primary platform for accessing the data. This dataset, meticulously curated and made publicly accessible, serves as a valuable resource for researchers and practitioners aiming to advance predictive models and interventions in the field of maternal and child health.

The dataset, originally obtained from UCI Machine Learning Repository, represents a compilation of relevant features and outcomes, providing a comprehensive foundation for the development and evaluation of machine learning models. Leveraging such datasets is fundamental in fostering advancements in healthcare research, enabling data-driven insights that contribute to the broader objectives of reducing child and maternal mortality globally.

In response to these challenges, Cardiotocograms (CTGs) emerge as a simple and

cost-accessible option for assessing fetal health. CTGs enable healthcare professionals to take timely actions to prevent both child and maternal mortality. The equipment functions by sending ultrasound pulses and analyzing their responses, providing crucial insights into fetal heart rate (FHR), fetal movements, uterine contractions, and more.

The dataset used in this study comprises 2126 records featuring extracted attributes from Cardiotocogram exams. These records were meticulously classified into three classes by expert obstetricians:

- **Normal**
- **Suspect**
- **Pathological**

This classification by experts sets the foundation for subsequent analyses, aiming to leverage machine learning for enhanced fetal health assessment and contribute to the broader goal of reducing child and maternal mortality.

4.2 Feature Names and Descriptions

- **baseline_value**: Baseline fetal heart rate
- **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**: Abnormal short-term variability of fetal heart rate
- **mean_value_of_short_term_variability**: Mean value of short-term variability of fetal heart rate

- **percentage_of_time_with_abnormal_long_term_variability**: Percentage of time with abnormal long-term variability of fetal heart rate
- **mean_value_of_long_term_variability**: Mean value of long-term variability of fetal heart rate
- **histogram_width**: Width of the histogram of fetal heart rate
- **histogram_min**: Minimum value of the histogram of fetal heart rate
- **histogram_max**: Maximum value of the histogram of fetal heart rate
- **histogram_number_of_peaks**: Number of peaks in the histogram of fetal heart rate
- **histogram_number_of_zeroes**: Number of zeroes in the histogram of fetal heart rate
- **histogram_mode**: Mode of the histogram of fetal heart rate
- **histogram_mean**: Mean value of the histogram of fetal heart rate
- **histogram_median**: Median value of the histogram of fetal heart rate
- **histogram_variance**: Variance of the histogram of fetal heart rate
- **histogram_tendency**: Tendency of the histogram of fetal heart rate
- **fetal_health**: Fetal health classification (Normal, Suspect, Pathological)

4.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a fundamental phase in our exploration of fetal health using Cardiotocography (CTG) data. This critical phase aims to provide in-depth insights into the dataset's inherent characteristics, unveiling patterns and anomalies that are vital for subsequent analysis. The primary objective is to enhance our understanding of the CTG data, laying the groundwork for informed decision-making and the development of robust predictive models.

Our exploration begins with the loading of the CTG dataset obtained from **Kaggle**, followed by a meticulous examination of its fundamental attributes. Descriptive statistics, featuring key metrics like mean, median, and standard deviation, offer succinct

summaries of the CTG data's features. Ensuring data quality is a crucial step, involving the detection and handling of missing values and outliers to maintain the integrity of the fetal health data.

The investigation extends to uncovering relationships between CTG features and their correlation with the target variable i.e. fetal health. This process is essential for building predictive models that contribute to understanding and assessing fetal well-being. Deliberations on data preprocessing, including feature scaling and handling categorical variables, are undertaken to address their relevance and impact on the CTG dataset.

Additionally, data visualization techniques are employed to unveil trends and patterns inherent in fetal health data. This visual exploration aids in gaining a comprehensive understanding of the CTG dataset, laying the groundwork for subsequent analyses and model development.

In summary, this dedicated exploration through EDA is pivotal for informed decision-making in the context of fetal health using CTG data. It provides a comprehensive understanding of the dataset's nuances, facilitating the development of accurate and meaningful predictive models for fetal health assessment.

The table in Table 4.1 provides key descriptive statistics for features in the Cardiotocography (CTG) dataset. Metrics like mean, median, and standard deviation offer insights into feature distributions. Notable features include "baseline value" with a stable mean of 133.304. Variability in parameters like "abnormal short-term variability" is reflected in their standard deviations, informing subsequent analyses and model development.

Feature	Mean	Median	Std Dev
baseline value	133.304	133.000	9.841
accelerations	0.003	0.002	0.004
fetal_movement	0.009	0.000	0.047
uterine_contractions	0.004	0.004	0.003
light_decelerations	0.002	0.000	0.003
severe_decelerations	0.000	0.000	0.000
prolongued_decelerations	0.000	0.000	0.001
abnormal_short_term_variability	46.990	49.000	17.193
mean_value_of_short_term_variability	1.333	1.200	0.883
percent_of_time_with_abnormal_long_term_var	9.847	0.000	18.397
mean_value_of_long_term_variability	8.188	7.400	5.628
histogram_width	70.446	67.500	38.956
histogram_min	93.579	93.000	29.560
histogram_max	164.025	162.000	17.944
histogram_number_of_peaks	4.068	3.000	2.949
histogram_number_of_zeroes	0.324	0.000	0.706
histogram_mode	137.452	139.000	16.381
histogram_mean	134.611	136.000	15.594
histogram_median	138.090	139.000	14.467
histogram_variance	18.808	7.000	28.978
histogram_tendency	0.320	0.000	0.611

Table 4.1: Descriptive statistics for CTG data features

4.3.1 Analysis and Visualization of Target Variable

Understanding the distribution of the '**fetal_health**' variable is crucial in gaining insights into the health status of the fetuses in the dataset. With a total of 2,126 observations, the '**fetal_health**' variable exhibits a mean value of approximately 1.304, indicating the typical fetal health condition. The standard deviation of approximately 0.614 reflects variability in health classifications, ranging from a minimum of 1 to a maximum of 3.

The below figure presents a pie chart and a bar plot illustrating the distribution

of fetal health classes '**Normal**,' '**Pathological**,' and '**Suspect**.' The '**Normal**' class constitutes the majority at approximately 77.8%, while '**Pathological**' and '**Suspect**' classes account for about 8.3% and 13.9%, respectively. This visual representation aids in comprehending the class distribution, offering valuable insights for classification tasks and their implications for the research.

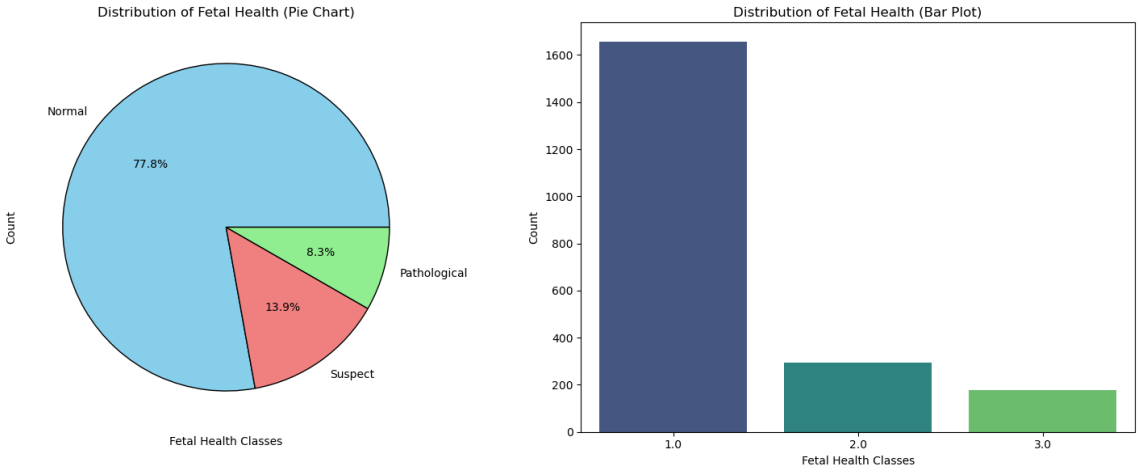

Figure 4.1: Distribution of Fetal Health Classes

4.3.2 CTG Features and Correlation

In the examination of the dataset, we initially presented descriptive statistics for the '**fetal_health**' variable, revealing its distribution and central tendencies. Subsequently, a pair plot was constructed to unravel the intricate relationships between various features in the dataset. This visualization tool facilitates the exploration of potential associations and patterns among features by showcasing scatterplots for each pair. Notably, diagonal elements in the pair plot present histograms, offering insights into the individual variable distributions, while off-diagonal elements depict scatter plots illustrating the relationships between pairs of variables. This comprehensive exploration aids not only in identifying feature associations but also guides subsequent analyses, including feature selection and model development.

The baseline fetal heart rate ('**baseline value**') spans from 106 to 160 beats per minute (bpm) with an average of approximately 133 bpm. Infrequent accelerations result in a mean of around 0.003, and fetal movement occurrences exhibit variations with an average of approximately 0.009. Uterine contractions, on average, register

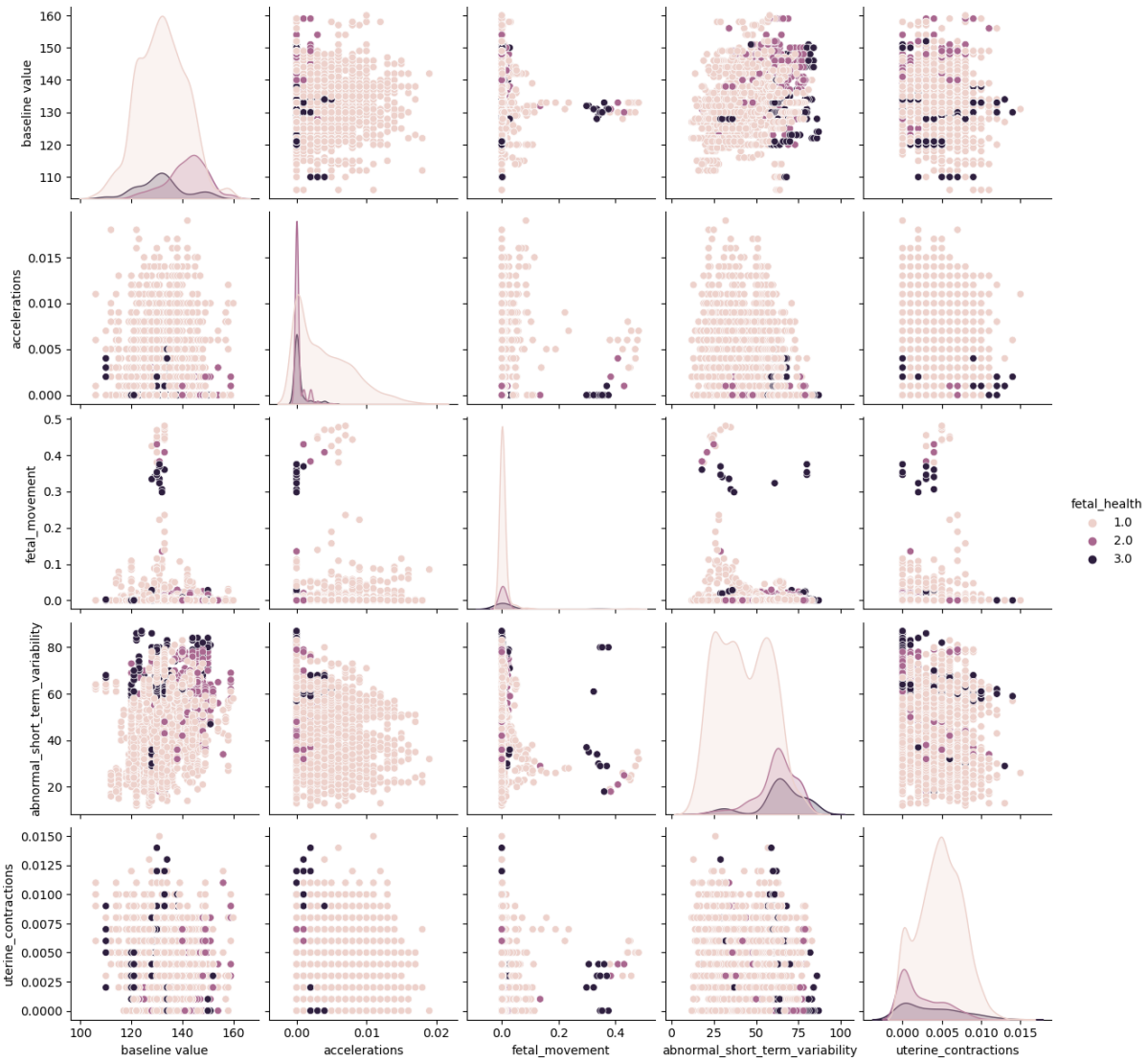

Figure 4.2: Pair Plot of CTG Features

at 0.004. Notably, '**abnormal_short_term_variability**' in fetal heart rate has a mean of approximately 46.99, showcasing significant variations. The dataset also includes parameters such as the percentage of time with abnormal long-term variability, ranging from 0.0% to 50.7%. Characteristics of the fetal heart rate histogram, including width, minimum, maximum, number of peaks, and mode, provide insights into the distribution of fetal heart rate patterns.

Exploring the correlation unveils significant relationships among variables. A moderate positive correlation is observed between '**abnormal_short_term_variability**' and '**fetal_health**,' suggesting that as the former increases, the latter tends to improve. Conversely, a moderate negative correlation is identified between '**accelerations**' and

'**fetal_health**,' indicating that higher '**accelerations**' may be associated with lower '**fetal_health.**' Variables like '**fetal_movement,**' '**baseline value**,' and '**uterine_contractions**' exhibit weak correlations with '**fetal_health**.' While these insights provide initial understanding, further analysis is imperative to ascertain causality and statistical significance, particularly in the context of fetal health monitoring decision-making.

4.3.3 Histogram Analysis of Key Features

The histograms for crucial features '**baseline value**,' '**accelerations**,' and '**uterine_contractions**' which provides a visual representation of their respective distributions within the dataset. For '**baseline value**,' the histogram centers around 130 to 140 beats per minute (bpm), showcasing the prevailing frequency of fetal baseline heart rates in this range. The '**accelerations**' histogram illustrates a predominance of low-frequency occurrences, indicating infrequent significant accelerations in fetal heart rate. Similarly, the '**uterine_contractions**' histogram reveals a prevalence of low-frequency contractions, suggesting that notable contractions are less frequent in the dataset. These histograms serve as insightful tools for interpreting the distribution patterns of key features, offering a foundational understanding of their characteristics and guiding subsequent analyses in the realm of fetal health monitoring.

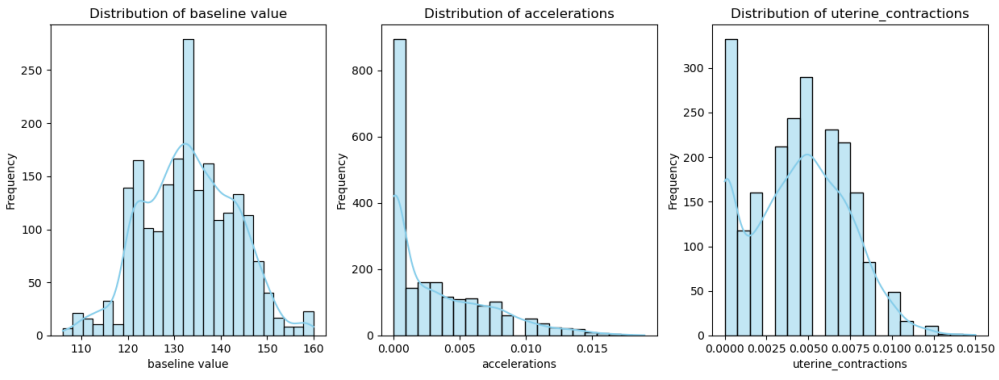

Figure 4.3: Histograms of Key Features

4.3.4 Data Preprocessing

In the exploration of the dataset, it was observed that it comprises a total of 2126 records and encompasses 22 columns, each representing various attributes related to

fetal monitoring. A meticulous examination revealed a notable characteristic which none of the columns in the dataset contain missing values or null entries.

This absence of missing data is a significant strength, ensuring the completeness and reliability of the dataset. The comprehensive nature of the data makes it well-suited for subsequent phases of preprocessing, feature engineering, and machine learning model development. The elimination of missing or null values is a crucial step in ensuring the quality and integrity of the dataset, setting the stage for accurate and meaningful analyses.[50][51]

The following sections will delve into the specific preprocessing steps undertaken to enhance the dataset's suitability for the machine learning algorithms employed in this study.

4.3.5 Feature Importance Analysis with Random Forest

We employed a Random Forest classifier to assess the significance of features [52][53] derived from cardiotocography (CTG) data in predicting fetal health. Notably, the feature "severe decelerations" demonstrated a relatively lower importance score, prompting a strategic decision to establish a threshold at 0.001 for feature inclusion.

This threshold-based approach facilitated the exclusion of features with limited impact on the model's predictive performance. [41] The decision to set a threshold at 0.001 for feature inclusion was specifically influenced by the lower importance observed for "severe decelerations." This deliberate choice aimed to enhance the model's interpretability and efficiency.

Subsequent evaluations indicated that this feature selection strategy had a positive impact on the model's performance. This outcome aligns with our overarching goal of developing a streamlined and effective predictive model for fetal health assessment.

CHAPTER 5

Results and Discussion

In this section, we reach the end of an exhaustive research expedition focused on unraveling the intricate nuances of predicting fetal health using CTG (Cardiotocography) data. Our journey encompasses meticulous exploration, modeling, and data analysis, resulting in valuable insights into our predefined research objectives. This phase serves as the stage to address fundamental research questions, assess hypotheses, and illuminate any noteworthy findings. A thorough examination of the research methodology is conducted, critically analyzing its strengths and limitations to provide essential context for comprehending the results.

Beyond the technicalities, our exploration extends to the broader contributions of the research within the realm of maternal-fetal medicine. We delve into both the practical and theoretical implications of our findings, bridging the gap between raw data and actionable insights. This holistic discussion crystallizes the full scope of our research, enabling us to draw well-founded conclusions and present a cohesive narrative.

For our analysis, we leverage CTG data, a crucial tool in monitoring fetal well-being, with the aim of predicting and enhancing fetal health outcomes. The dataset is meticulously examined, featuring 2,126 records and 22 columns representing various attributes related to fetal monitoring. We explore machine learning methodologies such as Logistic Regression, K-Nearest Neighbors, Random Forest, Decision Tree, and Support Vector Machine methods. Each of these approaches is tailored to contribute to the overarching goal of revolutionizing prenatal care and improving maternal and neonatal outcomes.

The upcoming subsections will provide a detailed exposition of the implementation, offering insights into the outcomes and discussions that emerged throughout this transformative project in the domain of fetal health prediction using CTG data.

5.1 Dataset Exploration and SMOTE Implementation

In the exploration of the dataset, it was observed that the dataset consists of a total of 2126 records and encompasses 22 columns. The distribution of fetal health classes 'Normal,' 'Pathological,' and 'Suspect' was as follows:

- **'Normal' Class:** This class constitutes the majority, representing approximately 77.8% of the dataset.
- **'Pathological' Class:** Accounting for about 8.3% of the dataset, this class represents instances indicating potential pathological conditions.
- **'Suspect' Class:** Comprising approximately 13.9% of the dataset, the 'Suspect' class encompasses instances that fall into an intermediate category.

This class distribution provides a foundational understanding of the dataset's composition, guiding subsequent preprocessing strategies. [54]

5.1.1 SMOTE Implementation

Due to the initial class imbalance, the Synthetic Minority Oversampling Technique (SMOTE) was employed to balance the dataset [33]. SMOTE was first evaluated on training folds before being used on the actual, intact, and previously unknown data.[55]

The initial distribution of instances across classes, prior to SMOTE, was as follows:

- **'Normal' Class:** 1655 instances
- **'Pathological' Class:** 295 instances
- **'Suspect' Class:** 176 instances

After the application of SMOTE, the class distribution transformed to:

- **'Normal' Class:** 1655 instances (unchanged)
- **'Pathological' Class:** 1655 instances (oversampled)
- **'Suspect' Class:** 1655 instances (oversampled)

This rebalanced distribution, achieved through the generation of synthetic instances, aimed to mitigate the risk of the machine learning model being disproportionately influenced by the majority classes. The adjusted dataset, now featuring a more equitable distribution of instances across classes, laid the groundwork for subsequent model training and evaluation.[33][54]

5.2 Feature Scaling and Train-Test Split

The preprocessing phase included feature scaling to standardize the range of features within the dataset. This procedure, crucial for optimizing the performance of machine learning models, ensures that no particular feature dominates others due to differing scales. The chosen method for feature scaling was the Standard Scaler method, renowned for preserving feature relationships while bringing them to a uniform scale. This standardized representation facilitates the convergence of machine learning algorithms, particularly those sensitive to variable scales, ensuring a more robust and accurate model training process. [56]

Following the preprocessing steps, the dataset underwent a strategic division into training and testing sets, a fundamental practice in model evaluation.[57] This train-test split aims to allocate a substantial portion of the data (70%) for training, allowing machine learning models to learn patterns and relationships. Simultaneously, a reserved subset (30%) serves as an independent testing ground to assess the models' generalization to new, unseen data.

The test size was set at 0.3, ensuring a balanced distribution between the training and testing sets. This random split guarantees that both sets maintain the overall data distribution, providing an unbiased evaluation of model performance. The training set acts as a learning arena, and the testing set serves as real-world validation, simulating the models' ability to handle novel scenarios. This meticulous division contributes to a comprehensive assessment of the models' efficacy in making accurate predictions on previously unseen data.

5.3 Logistic Regression Implementation

The logistic regression model exhibited commendable accuracy, achieving approximately 86.2% on the test dataset. The classification report below provides a detailed breakdown of precision, recall, and F1-score for each fetal health class. Notably, the model displayed high precision and recall values for all classes, indicating a robust ability to correctly identify instances.

Table 5.1: **Classification Report - Logistic Regression**

	precision	recall	f1-score	support
1.0	0.95	0.85	0.90	496
2.0	0.78	0.85	0.82	497
3.0	0.87	0.89	0.88	497
accuracy		0.86		1490
macro avg	0.87	0.86	0.86	1490
weighted avg	0.87	0.86	0.86	1490

5.3.1 Hyperparameter Tuning

Logistic Regression underwent hyperparameter tuning using GridSearchCV, with the best hyperparameter configuration identified as {'C': 100}. This optimization led to a marginal improvement in accuracy, reaching 86.4%.

5.3.2 Cross-validation Scores

Cross-validation scores for Logistic Regression demonstrated consistent performance across folds, with scores ranging from 84.2% to 89.6%. This stability suggests the model's robustness and generalizability to unseen data.

5.3.3 Learning Curves

The provided data represents a learning curve with the number of training examples (dataset size) and the corresponding training and cross-validation (CV) scores. Let's analyze this learning curve for each model:

- Both the training score and cross-validation score generally increase with the number of training examples, suggesting that the model benefits from more data.
- The training score and cross-validation score are close, indicating that the model is not overfitting or underfitting significantly.
- The scores seem to have stabilized or reached a plateau, especially beyond 1750 training examples, suggesting that adding more training examples may not lead to substantial improvement.

Overall, the learning curve suggests that the model is learning well from the available data, and the performance has converged to a reasonably good level. Further improvements might come from other optimization strategies or model adjustments.

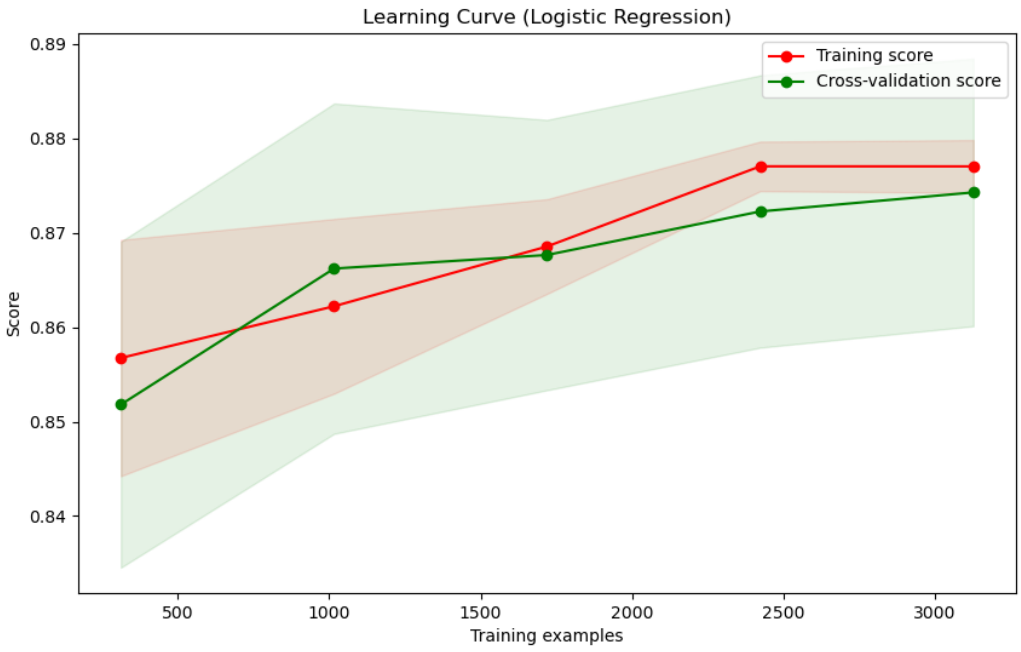

Figure 5.1: Learning Curve for Logistic Regression

5.3.4 Confusion Matrix

The confusion matrix in the context of a medical diagnosis, where Class 1 represents normal cases, Class 2 represents suspect cases, and Class 3 represents pathological cases.

1. **Normal Cases (Class 1):**

- True Positives: 420
- False Negatives: 76
- False Positives: 22

The model performed well in identifying normal cases, but there were instances of misclassification.

2. **Suspect Cases (Class 2):**

   - True Positives: 424
   - False Negatives: 73
   - False Positives: 62

   The model performed reasonably well in identifying suspect cases.

3. **Pathological Cases (Class 3):**

   - True Positives: 441
   - False Negatives: 69
   - False Positives: 53

   The model performed well in identifying pathological cases.

Interpretation of the Confusion Matrix

Analyzing the confusion matrix, we observe that the model excels in identifying pathological cases (Class 3) and normal cases (Class 1). However, there is room for improvement in identifying suspect cases (Class 2), where both false positives and false negatives exist. This suggests a potential focus for further model refinement.

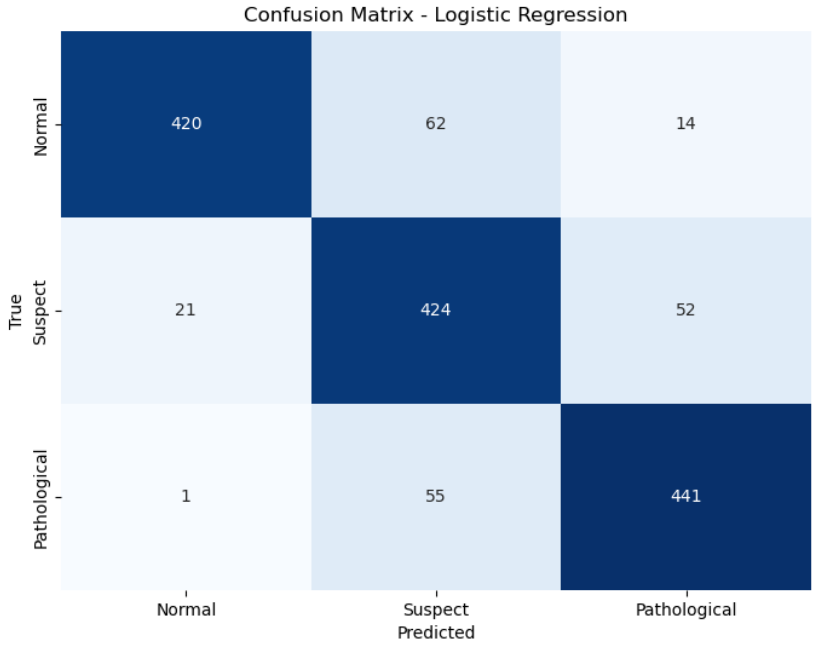

Figure 5.2: Confusion Matrix for Logistic Regression

5.3.5 Feature Contributions and Coefficients

As an additional step, the logistic regression model reveals the coefficients associated with each feature, shedding light on their respective contributions to the prediction of fetal health classifications. Notably, positive coefficients indicate a positive association with the likelihood of a certain fetal health class, while negative coefficients suggest a negative association.

Feature Impact Analysis

For instance, the feature **"accelerations"** demonstrates a substantial positive coefficient of 3.07, implying that an increase in accelerations is strongly associated with a higher likelihood of a pathological fetal health classification. On the other hand, features like **"prolongued_decelerations," "abnormal_short_term_variability,"** and **"histogram_variance"** exhibit negative coefficients, indicating their negative impact on the prediction of pathological fetal health. These insights provide a nuanced understanding of the features' influences and aid in interpreting the model's decision-making process.

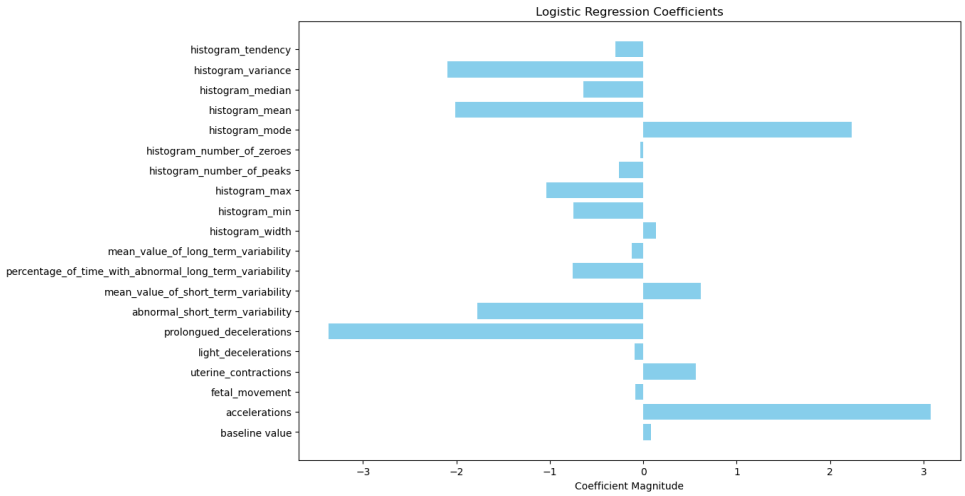

Figure 5.3: Logistic Regression Coefficients Plot

5.4 K-Nearest Neighbors (KNN) Implementation

The KNN model exhibited a remarkable accuracy of 95.3%, showcasing its proficiency in classifying fetal health categories. The classification report provided granular insights into precision, recall, and F1-score for each class (1.0, 2.0, 3.0). The model excelled in classifying instances of class 1.0, demonstrating high precision and recall. However, opportunities for improvement were identified in the classification of class 2.0. The weighted average F1-score of 0.95 underscored a balanced performance across classes, affirming the model's overall effectiveness.

Table 5.2: **Classification Report - K-Nearest Neighbors**

	precision	**recall**	**f1-score**	**support**
1.0	1.00	0.90	0.94	496
2.0	0.90	0.97	0.94	497
3.0	0.97	0.99	0.98	497
accuracy		0.95		1490
macro avg	0.96	0.95	0.95	1490
weighted avg	0.96	0.95	0.95	1490

5.4.1 Hyperparameter Tuning

Hyperparameter tuning using GridSearchCV identified the optimal configuration as {'n_neighbors': 3}. This refinement enhanced the model's accuracy, reaching 96.0%.

5.4.2 Cross-validation Scores

Cross-validation scores consistently reflected robust model performance, ranging from 94.5% to 97.1%. This stability underscores the model's reliability and generalizability.

5.4.3 Learning Curves

The learning curve analysis reveals a positive trend in both training and cross-validation (CV) scores as the dataset size increases. The training score steadily improves, reaching 0.98 with 3200 examples, indicating the model's enhanced proficiency with more data. Simultaneously, the CV score rises positively, attaining 0.95 with 3200 examples, showcasing the model's effective generalization to previously unseen data. The convergence of training and CV scores suggests a well-regularized model, with diminishing returns beyond 3200 examples. The minimal gap between training and CV scores implies that the model avoids overfitting, ensuring robust generalization. The consistently high CV score signifies the model's strong performance on new, unseen data. Overall, the learning curve indicates that the model is well-suited for the current dataset size, with further exploration recommended if more data becomes available to assess potential improvements.

5.4.4 Confusion Matrix

The confusion matrix for KNN, with a focus on normal (Class 1), suspect (Class 2), and pathological (Class 3) cases, provides a detailed breakdown of true positives, false negatives, and false positives.

1. **Normal Cases (Class 1):**

    - True Positives: 444
    - False Negatives: 52

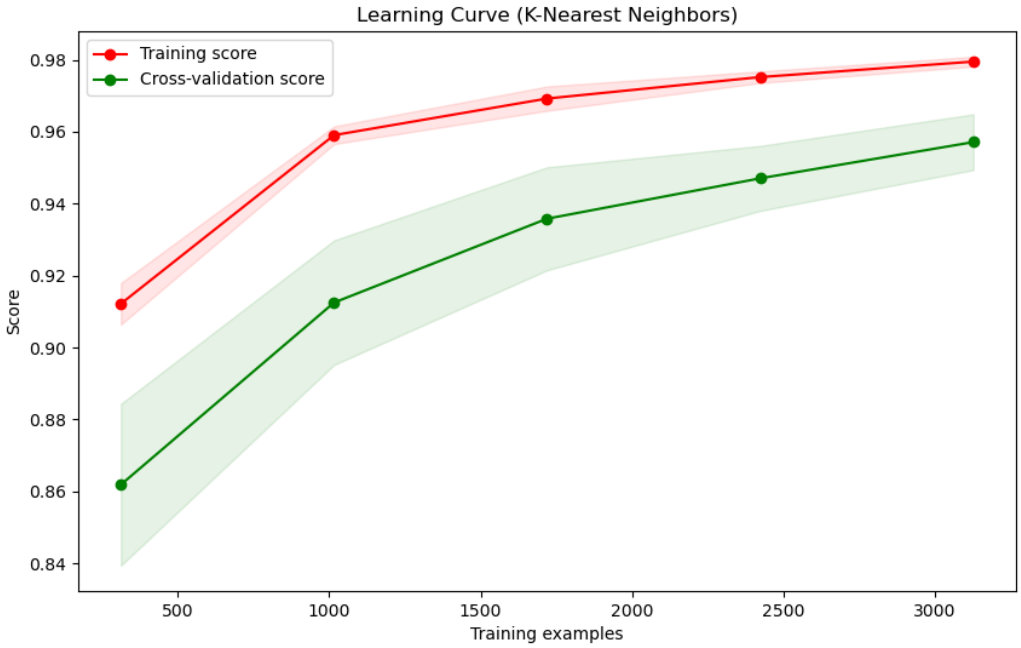

Figure 5.4: Learning Curve for K-Nearest Neighbors

- False Positives: 1

The model performed well in identifying normal cases (True Positives), with a small number of normal cases misclassified as suspect or pathological (False Negatives). Additionally, there was a very low number of non-normal cases incorrectly classified as normal (False Positives).

2. **Suspect Cases (Class 2):**

   - True Positives: 484
   - False Negatives: 13
   - False Positives: 47

   The model performed well in identifying suspect cases (True Positives), with a small number of suspect cases misclassified as normal or pathological (False Negatives). Additionally, there were instances of non-suspect cases being incorrectly classified as suspect (False Positives).

3. **Pathological Cases (Class 3):**

- True Positives: 492
- False Negatives: 5
- False Positives: 0

The model performed well in identifying pathological cases (True Positives), with very few instances of misclassifications (False Negatives and no False Positives).

In summary, the analysis of this confusion matrix indicates that the model generally performed well across all classes. The most notable errors occurred in predicting normal and suspect cases, but overall, the model demonstrated good accuracy in distinguishing between the different classes. Understanding these specific types of errors can guide further model improvements, especially if certain misclassifications carry significant implications in the given context.

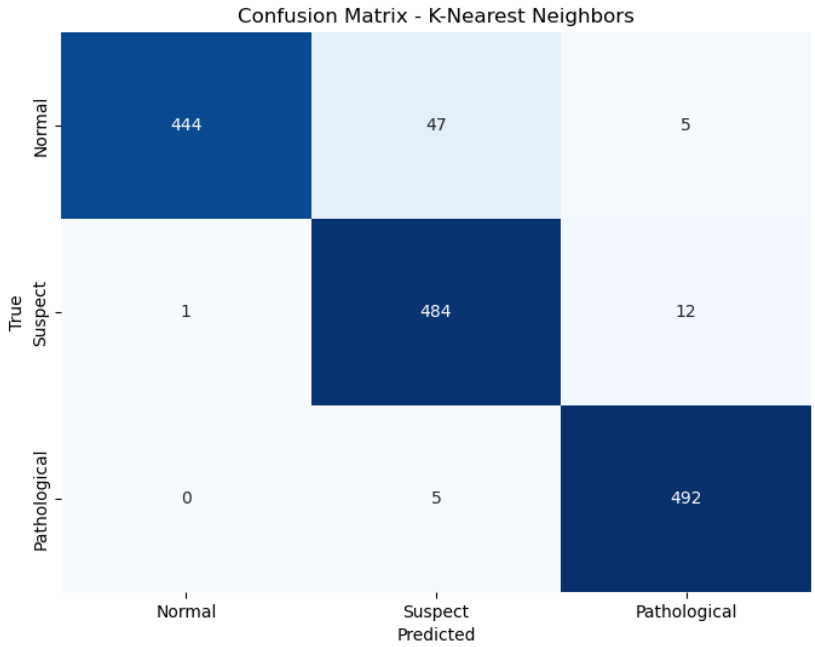

Figure 5.5: Confusion Matrix for K-Nearest Neighbors

5.5 Random Forest Implementation

The Random Forest model achieved an exceptional accuracy of 98.1%, showcasing its proficiency in distinguishing between Normal, Suspect, and Pathological cases in

fetal health prediction. Precision, recall, and F1-score, vital metrics in classification evaluation, underscore the model's accuracy in positive predictions, capturing positive instances effectively, and providing a balanced evaluation.

Table 5.3: **Classification Report - Random Forest**

	precision	recall	f1-score	support
1.0	0.98	0.97	0.97	496
2.0	0.96	0.98	0.97	497
3.0	1.00	0.99	0.99	497
accuracy		0.98		1490
macro avg	0.98	0.98	0.98	1490
weighted avg	0.98	0.98	0.98	1490

5.5.1 Hyperparameter Tuning

Hyperparameter tuning, executed through GridSearchCV, enhanced the model's parameters. This involved systematically exploring predefined hyperparameter sets to maximize the model's effectiveness. The selected hyperparameters, including a maximum depth of 20, minimum samples per leaf set to 1, minimum samples per split set to 2, and 200 estimators, highlight the optimized configuration contributing to the model's outstanding predictive capabilities.

The maintained accuracy of 98.1% after hyperparameter tuning further reinforces the impact of optimized parameters, affirming the model's reliability in fetal health prediction.

5.5.2 Learning Curves

The learning curve analysis reveals a model that impeccably fits the training data, achieving a flawless training score of 1.0 across all dataset sizes. Concurrently, cross-validation scores exhibit a consistent upward trend, ranging from 0.93 to 0.97, signifying the model's robust generalization to new, unseen data. The convergence of both training and validation curves indicates that increasing the training set size has minimal impact on performance, suggesting a well-optimized model. The consistently high

performance on the validation set implies that the model may have already reached its optimal dataset size, showcasing a balanced configuration with low variance and bias. In summary, the model is well-trained, displaying effective generalization to new data, and further expansion of the training set might yield marginal improvements. This equilibrium suggests a well-fitted and balanced model, avoiding the pitfalls of overfitting or underfitting.

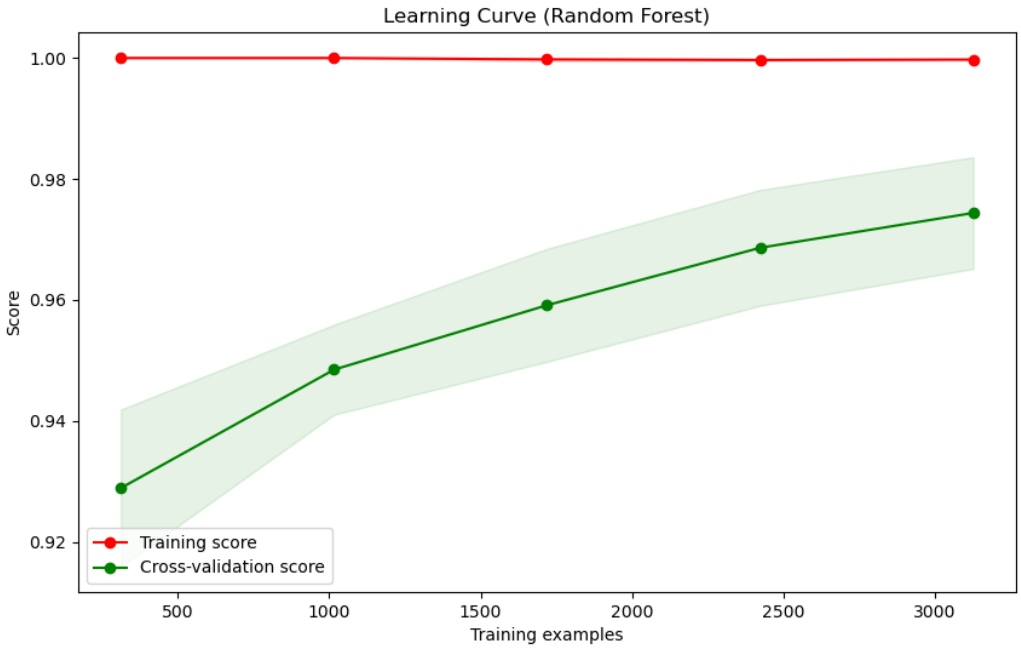

Figure 5.6: Learning Curve for Random Forest

5.5.3 Confusion Matrix

The confusion matrix is presented below in the context of a medical diagnosis, where Class 1 represents normal cases, Class 2 represents suspect cases, and Class 3 represents pathological cases.

The minimal misclassifications depicted in the confusion matrix affirm the model's robustness, visually representing instances correctly and incorrectly classified for each fetal health class.

1. **Normal Cases (Class 1):**

    - True Positives: 479

- False Negatives: 17
- False Positives: 8

The model performed well in identifying normal cases (True Positives), and there were some instances of normal cases being misclassified as suspect or pathological (False Negatives). Additionally, a small number of non-normal cases were incorrectly classified as normal (False Positives).

2. **Suspect Cases (Class 2):**

   - True Positives: 488
   - False Negatives: 9
   - False Positives: 16

   The model performed well in identifying suspect cases (True Positives), and there were some instances of suspect cases being misclassified as normal or pathological (False Negatives). Additionally, there were instances of non-suspect cases being incorrectly classified as suspect (False Positives).

3. **Pathological Cases (Class 3):**

   - True Positives: 494
   - False Negatives: 3
   - False Positives: 1

   The model performed well in identifying pathological cases (True Positives), with very few instances of misclassifications (False Negatives and False Positives).

In summary, the analysis of this confusion matrix indicates that the model generally performed well in correctly identifying suspect and pathological cases. However, there were some instances of misclassifications, particularly for normal cases, and understanding these specific types of errors is crucial for refining the model and improving its diagnostic accuracy, especially in medical contexts.

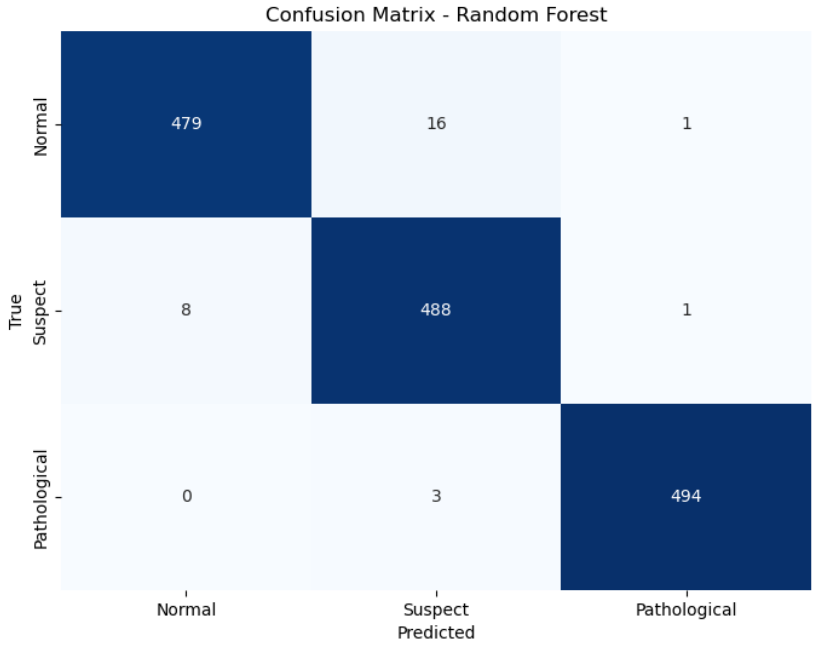

Figure 5.7: Confusion Matrix for Random Forest

5.6 Decision Tree Implementation

The Decision Tree model displays robust performance with a high accuracy of 95.3% on the test data. The classification report provides detailed metrics such as precision, recall, and F1-score for each class (1.0, 2.0, 3.0). Across all classes, the model shows strong performance with elevated precision, recall, and F1-scores, affirming its effectiveness in accurate classification.

Table 5.4: **Classification Report - Decision Tree**

	precision	recall	f1-score	support
1.0	0.95	0.93	0.94	496
2.0	0.93	0.95	0.94	497
3.0	0.98	0.99	0.98	497
accuracy		0.95		1490
macro avg	0.95	0.95	0.95	1490
weighted avg	0.95	0.95	0.95	1490

5.6.1 Hyperparameter Tuning

The hyperparameter tuning process using GridSearchCV reveals the optimal configuration for the Decision Tree model. The chosen hyperparameters include 'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 1, and 'min_samples_split': 2. These parameters contribute to the model's enhanced performance.

Even after hyperparameter tuning, the accuracy remains consistently at 95.3%. This stability suggests that the initial model was well-configured, and the robust accuracy persists despite changes in hyperparameters. Overall, the model's performance appears resilient and consistently effective across different parameter settings.

5.6.2 Cross-Validation

The cross-validation scores further validate the model's consistency, with scores ranging from 93.4% to 96.8% across different folds. This demonstrates that the model generalizes well to unseen data and is not overfitting to the training set.

5.6.3 Learning Curves

The learning curve analysis provides valuable insights into the model's behavior and generalization capabilities. The consistently high training score of 1 indicates the model's proficiency in memorizing the dataset. Concurrently, the upward trend in cross-validation scores, ascending from 0.89 to 0.95 with increasing training examples, signifies the model's robust ability to generalize to unseen data. This emphasizes the importance of a well-sized and diverse training dataset for optimal model performance. The intricate relationship between training data volume and model proficiency is evident in these results, emphasizing the nuanced dynamics of the learning process.

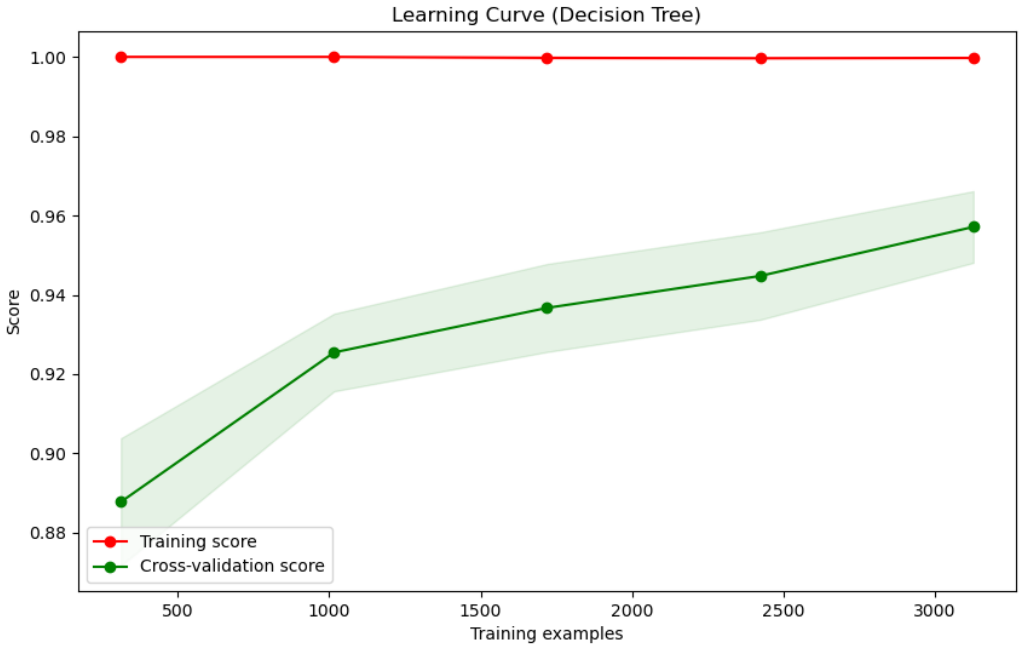

Figure 5.8: Learning Curve for Decision Tree

5.6.4 Confusion Matrix

1. **Normal Cases (Class 1):**

   - True Positives (correctly predicted normal cases): 460
   - False Negatives (normal cases incorrectly predicted as Suspect or Pathological): 36
   - False Positives (non-normal cases incorrectly predicted as Normal): 25

   The model performed well in identifying Normal cases (True Positives), but there were instances of Normal cases being misclassified as Suspect or Pathological (False Negatives). Additionally, there were some non-Normal cases incorrectly classified as Normal (False Positives).

2. **Suspect Cases (Class 2)**

   - True Positives (correctly predicted Suspect cases): 470
   - False Negatives (Suspect cases incorrectly predicted as Normal or Pathological): 27

- False Positives (non-Suspect cases incorrectly predicted as Suspect): 37

The model performed well in identifying Suspect cases (True Positives), but there were instances of Suspect cases being misclassified as Normal or Pathological (False Negatives). Additionally, there were some non-Suspect cases incorrectly classified as Suspect (False Positives).

3. **Pathological Cases (Class 3)**

   - True Positives (correctly predicted Pathological cases): 490
   - False Negatives (Pathological cases incorrectly predicted as Normal or Suspect): 7
   - False Positives (non-Pathological cases incorrectly predicted as Pathological): 55

The model performed well in identifying Pathological cases (True Positives), with very few instances of misclassifications (False Negatives and False Positives).

In summary, the analysis of the confusion matrix indicates that the model generally performed well across all classes, with some notable errors in misclassifying instances between different classes. Understanding these specific types of errors is crucial for refining the model and improving its accuracy, particularly in scenarios where certain misclassifications may have significant implications for Normal, Suspect, or Pathological conditions.

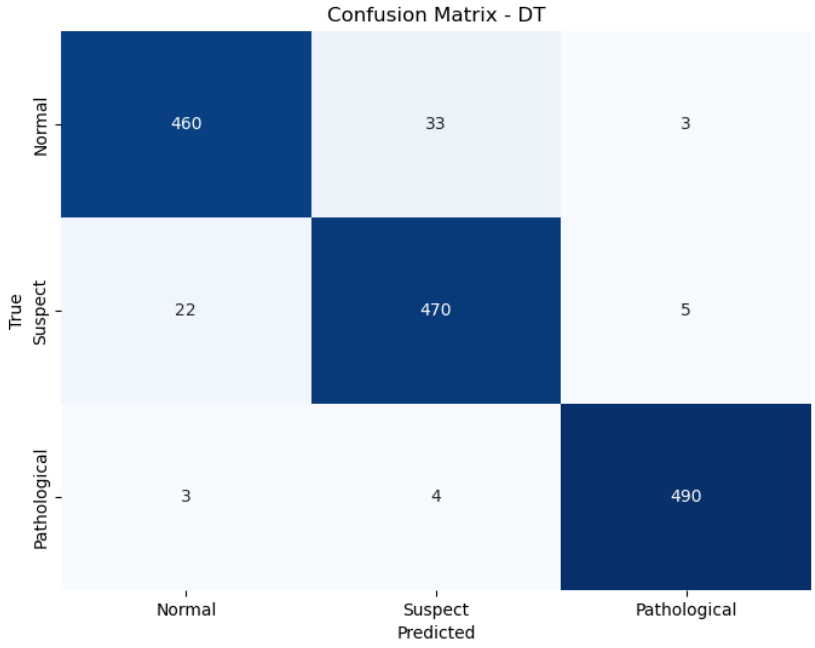

Figure 5.9: Confusion Matrix for Decision Tree

5.7 Support Vector Machine (SVM) Implementation

The Support Vector Machine (SVM) model achieved an accuracy of 94.2% on the test data, indicating its effectiveness in making accurate predictions.The SVM model shows strong predictive capabilities with high accuracy and balanced metrics. Hyperparameter tuning enhances the model's accuracy, making it a reliable tool for fetal health classification. The consistent cross-validation scores affirm the model's generalization to new data, highlighting its suitability for accurate predictions in a healthcare context.

The classification report provides key metrics for each class (Normal, Suspect, Pathological). The model demonstrates balanced precision, recall, and F1-score across classes.

Table 5.5: **Classification Report - Support Vector Machine**

	precision	recall	f1-score	support
1.0	0.99	0.90	0.94	496
2.0	0.89	0.95	0.92	497
3.0	0.95	0.98	0.96	497
accuracy		0.94		1490
macro avg	0.94	0.94	0.94	1490
weighted avg	0.94	0.94	0.94	1490

The weighted average F1-Score is 94%, reflecting the model's overall balanced performance.

5.7.1 Hyperparameter Tuning:

Hyperparameter tuning using GridSearchCV optimized the model with the following parameters:

- Best Hyperparameters: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}

The accuracy of the model after hyperparameter tuning increased to 96.4%, showcasing the positive impact of optimized parameters.

5.7.2 Cross-Validation:

Cross-validation scores consistently demonstrate high performance, ranging from 93.7% to 98.0%.

5.7.3 Learning Curves

The learning curve analysis reveals significant insights into the model's performance as the dataset size increases. The upward trend in the training score, reaching 0.975, indicates effective learning from the training data. Simultaneously, the cross-validation (CV) score demonstrates consistent improvement, reaching 0.96, highlighting the model's ability to generalize to unseen data.

The narrow gap between the training and CV scores suggests a balanced model without significant overfitting. With 3200 training examples, the model achieves high

accuracy on both sets, implying a favorable bias-variance tradeoff. Further dataset expansion may offer diminishing returns, suggesting potential exploration of hyperparameter tuning for optimization.

In summary, the learning curve analysis provides valuable insights into effective learning, generalization, and model behavior. These findings inform decision-making for deployment and optimization strategies.

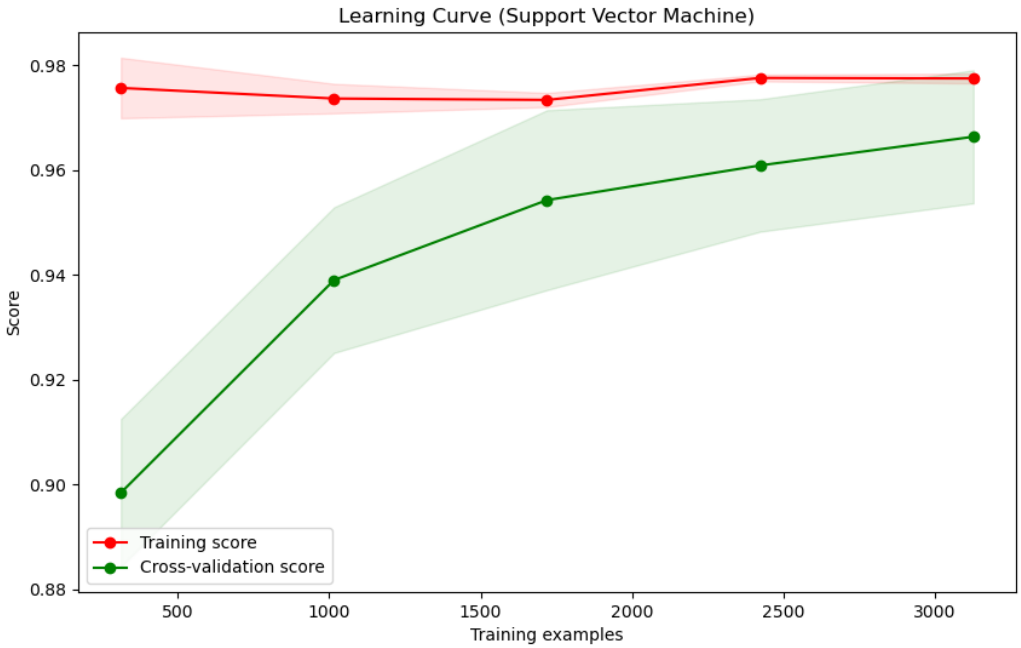

Figure 5.10: Learning Curve for Support Vector Machine

5.7.4 Confusion Matrix

1. **Normal Cases (Class 1):**

   - True Positives (correctly predicted normal cases): 445
   - False Negatives (normal cases incorrectly predicted as Suspect or Pathological): 51
   - False Positives (non-normal cases incorrectly predicted as Normal): 3

   The model performed well in identifying Normal cases (True Positives), but there were instances of Normal cases being misclassified as Suspect or Pathological

(False Negatives). Additionally, there were very few non-Normal cases incorrectly classified as Normal (False Positives).

2. **Suspect Cases (Class 2)**

   - True Positives (correctly predicted Suspect cases): 472
   - False Negatives (Suspect cases incorrectly predicted as Normal or Pathological): 25
   - False Positives (non-Suspect cases incorrectly predicted as Suspect): 59

   The model performed well in identifying Suspect cases (True Positives), but there were instances of Suspect cases being misclassified as Normal or Pathological (False Negatives). Additionally, there were some non-Suspect cases incorrectly classified as Suspect (False Positives).

3. **Pathological Cases (Class 3)**

   - True Positives (correctly predicted Pathological cases): 486
   - False Negatives (Pathological cases incorrectly predicted as Normal or Suspect): 14
   - False Positives (non-Pathological cases incorrectly predicted as Pathological): 51

The model performed well in identifying Pathological cases (True Positives), with very few instances of misclassifications (False Negatives and False Positives).

In summary, the analysis provides a detailed understanding of the model's performance for each class, helping to identify specific areas for improvement and refinement.

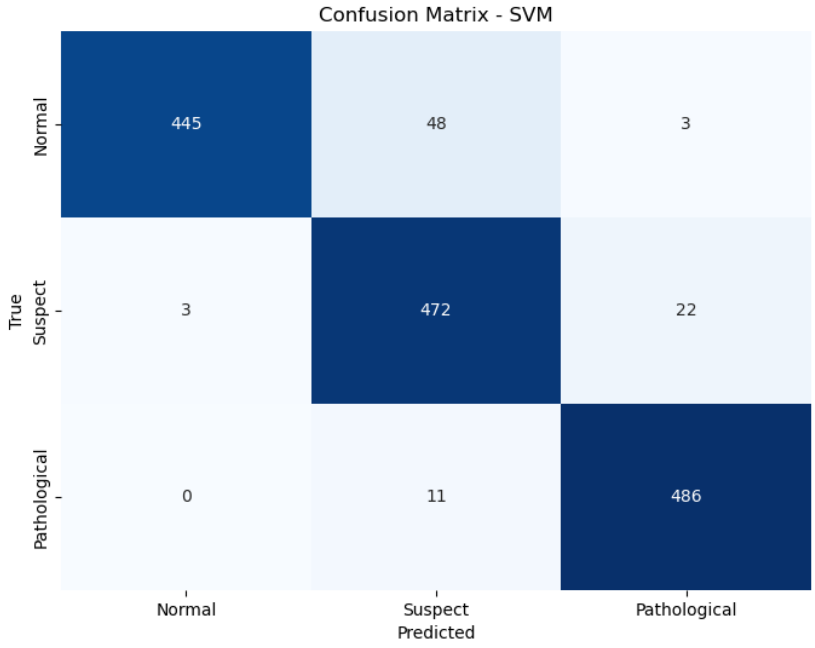

Figure 5.11: Confusion Matrix for Support Vector Machine

5.8 Performance and Evaluation of Implemented Models

The table provides a concise overview and comparative analysis of various machine learning models used in the project, emphasizing accuracy metrics. The accompanying graph extends the analysis to include precision, recall, and F1 score, offering a comprehensive assessment for optimal model selection in fetal health prediction.

Model	Accuracy
Logistic Regression	86.2%
K-Nearest Neighbors	95.3%
Random Forest	98.1%
Decision Tree	95.3%
Support Vector Machine	94.2%

Table 5.6: Model Accuracy Comparison

Random Forest emerges as a top performer with an impressive 98.1% accuracy, excelling in precision, recall, and F1-Score, all at 98%. K-Nearest Neighbors (KNN) closely follows with a commendable 95.3% accuracy, showcasing a balanced performance in

precision, recall, and F1-Score around 96% and 95%, respectively. The Decision Tree model matches KNN's accuracy at 95.3%, providing transparency in decision-making. Logistic Regression, though slightly lower in accuracy at 86.2%, maintains a balance in precision and recall, proving reliable, especially in scenarios prioritizing interpretability. Support Vector Machine (SVM) navigates fetal health prediction with 94.2% accuracy, depicting robust classification.

Upon reviewing the comparative results, certain similarities and distinctions become evident. Random Forest and KNN showcase high accuracy and balanced precision-recall trade-offs, positioning them as top performers. Decision Tree, while maintaining competitive accuracy, provides transparency in its decision-making process. Logistic Regression, with a focus on interpretability, and SVM, offering robust classification, contribute to the diversity of the model ensemble. The attached graph visually encapsulates these comparative results, offering a succinct representation of each model's strengths and trade-offs. Ultimately, the choice of a model depends on the specific requirements of the medical context and the emphasis placed on accuracy, interpretability, or a combination of both in fetal health prediction.

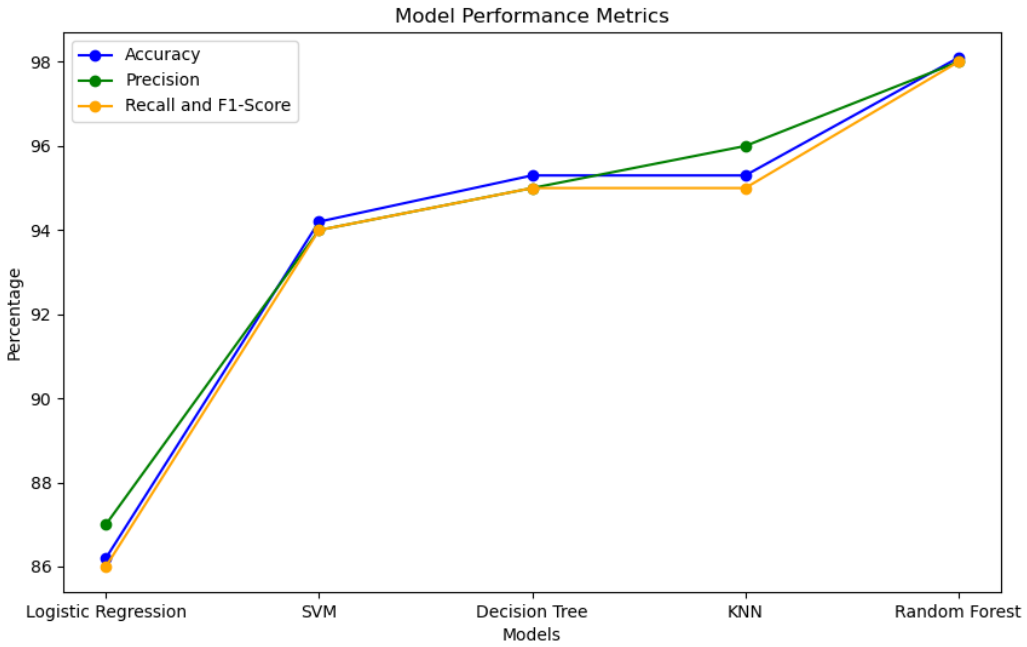

Figure 5.12: Model Performance Metrics

CHAPTER 6

Conclusions

The comparative results highlight certain similarities and distinctions among the models. Random Forest and KNN showcase high accuracy and balanced precision-recall trade-offs, positioning them as top performers. Decision Tree, while maintaining competitive accuracy, provides transparency in its decision-making process. Logistic Regression, with a focus on interpretability, and SVM, offering robust classification, contribute to the diversity of the model ensemble.

In the comprehensive evaluation of fetal health classification models, our proposed approaches stand out prominently, showcasing superior performance compared to existing methodologies. The Random Forest model, introduced in this paper, achieves an exceptional accuracy of 98.1%, surpassing the benchmarks set by established models such as the Decision Tree [31] and EMD-SMV [32], which achieved accuracies of 86.36% and 86%, respectively. Similarly, our proposed K-Nearest Neighbors (KNN) model attains a remarkable accuracy of 95.3%, outshining both the Decision Tree (This paper) and SVM (This paper) models, which achieve accuracies of 95.3%. Complementing these results, our proposed SVM model (This paper) demonstrates competitive accuracy at 94.2%.

This comparative analysis underscores the efficacy and innovation introduced by our proposed Random Forest, KNN, and SVM models, affirming their potential for advancing the state-of-the-art in fetal health classification. These models not only outperform existing counterparts but also contribute valuable insights for further exploration and enhancement of fetal health prediction methodologies.

The future of fetal health prediction research could explore the integration of additional data modalities, such as maternal health parameters or genetic information, to enhance predictive capabilities. Longitudinal studies with larger datasets may provide insights into evolving fetal health patterns over time. Acknowledging dataset variability and the need for rigorous clinical validation remains crucial for the robust application

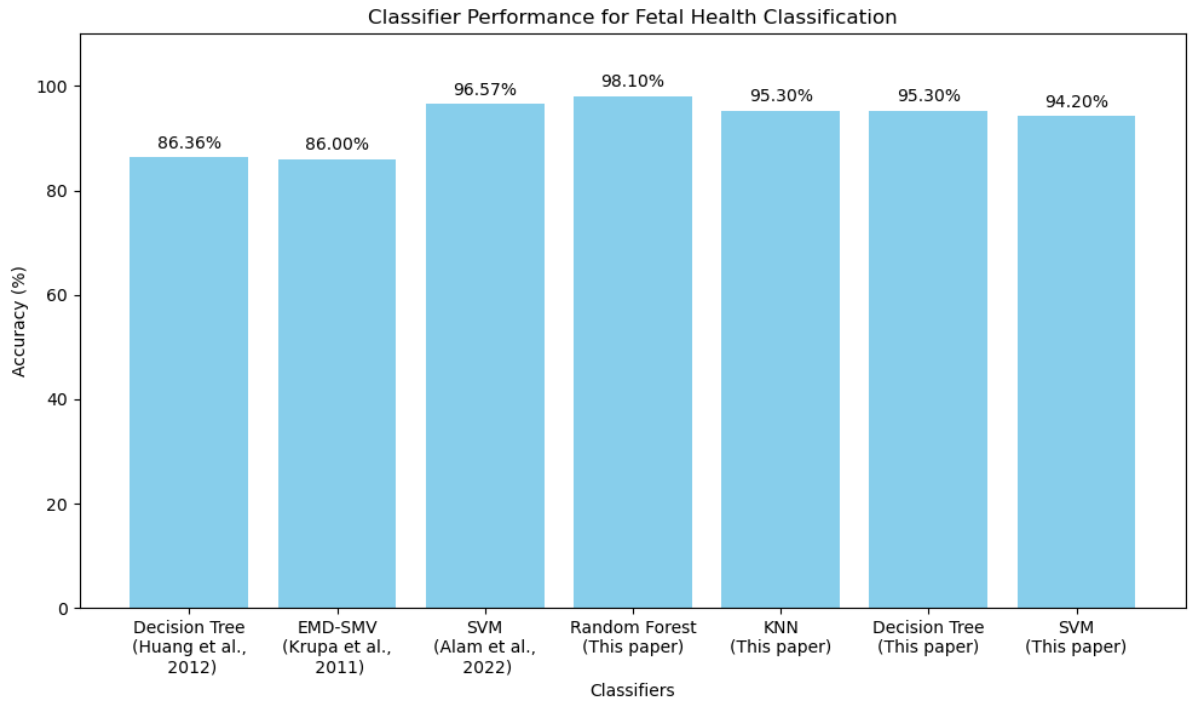

Figure 6.1: Classifier Performance Plot

of these models in diverse healthcare settings.

In conclusion, the proposed machine learning models, trained on Cardiotocography (CTG) data, offer a promising avenue for advancing fetal health diagnostics. Their varied strengths cater to different clinical needs, providing clinicians with valuable tools for accurate, timely, and interpretable predictions. As technology continues to evolve, these models stand at the forefront of enhancing prenatal care, contributing to improved patient outcomes and shaping the future landscape of fetal health assessment.

Appendix

PYTHON CODE

Imports

```
import os  # Operating system module
import warnings  # Warnings module
from collections import Counter  # Counter for occurrences
import numpy as np  # NumPy for numerical operations
import pandas as pd  # Pandas for data manipulation
import matplotlib.pyplot as plt  # Matplotlib for visualization
import seaborn as sns  # Seaborn for enhanced visualization
from imblearn.over_sampling import SMOTE  # SMOTE for
    oversampling
from tabulate import tabulate  # Tabulate for formatted tables

# Import scikit-learn modules for machine learning
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC  # Support Vector Machine
from sklearn.model_selection import (
    cross_val_score,
    StratifiedKFold,
    train_test_split,
    GridSearchCV,
    learning_curve,
)
from sklearn.preprocessing import StandardScaler  #
```

```
    StandardScaler for scaling
from sklearn import linear_model  # Linear models
from sklearn.linear_model import LogisticRegression  # Logistic
    Regression
from sklearn.neighbors import KNeighborsClassifier  # K-Nearest
    Neighbors Classifier
from sklearn.ensemble import RandomForestClassifier  # Random
    Forest Classifier
from sklearn.metrics import (
    accuracy_score,
    classification_report,
    confusion_matrix,
    mean_squared_error,
)

# Ignore specific warning by category
warnings.filterwarnings("ignore", category=DeprecationWarning)

# Ignore all warnings
warnings.filterwarnings("ignore")
```

Dataset

```
# Read the CSV file into a DataFrame named 'ctg_data'
ctg_data = pd.read_csv('/Users/kautilyareddy/Desktop/
    fetal_health.csv')

# Now you can work with the 'ctg_data' DataFrame, which
    contains cardiotocography data
ctg_data

ctg_data.head()
```

Feature Names

```
# Print the columns of the 'ctg_data' dataset
ctg_data.columns
```

Exploratory Data Analysis

```
ctg_data.shape
```

Data Preprocessing

```
# Identify columns with missing values (NaN)
missing_values = ctg_data.isnull().sum()

# Print columns with missing values and their counts
print("Missing values:")
print(missing_values[missing_values > 0])

# Identify columns with null values
null_values = ctg_data.isna().sum()

# Print columns with null values and their counts
print("\nNull values:")
print(null_values[null_values > 0])
```

Feature importance with Random Forest and Descriptive Statistics

```
# Separate the features and the target variable
A = ctg_data.drop('fetal_health', axis=1)  # Features
B = ctg_data['fetal_health']  # Target variable

# Train a Random Forest model to obtain feature importance
    scores
clf = RandomForestClassifier()
clf.fit(A, B)
```

```
importances = clf.feature_importances_

# Define a threshold for feature importance (adjust as needed)
threshold = 0.001

# Get the indices of features with importance scores above the
    threshold
selected_feature_indices = [i for i, importance in enumerate(
    importances) if importance > threshold]

# Get the names of all features
all_features = A.columns

# Get the names of the eliminated features
eliminated_features = [feature for feature in all_features if
    feature not in A.columns[selected_feature_indices]]

# Get the names of the selected features
selected_features = A.columns[selected_feature_indices]

# Print the selected feature names
print("Selected Feature Names:")
print(selected_features)

# Print the eliminated feature names
print("\nEliminated Feature Names:")
print(eliminated_features)


# Calculate descriptive statistics for specific features in the
    'CTG_data' DataFrame
ctg_data_summary = A.describe().T[['mean', '50%', 'std']]
```

```
# Format the mean and median columns to display up to 3 decimal
    places
ctg_data_summary['mean'] = ctg_data_summary['mean'].apply(
    lambda x: f"{x:.3f}")
ctg_data_summary['50%'] = ctg_data_summary['50%'].apply(lambda
    x: f"{x:.3f}")
ctg_data_summary['std'] = ctg_data_summary['std'].apply(lambda
    x: f"{x:.3f}")

# Print the transposed summary statistics for CTG data features
    in a table with a custom heading
print(tabulate(ctg_data_summary, headers=['Feature', 'Mean', '
    Median', 'Std␣Deviation'], tablefmt='pretty'))
```

Analysis and Visualization of Target Variable

```
# Calculate the counts of each unique value in 'fetal_health'
health_counts = ctg_data["fetal_health"].value_counts()

# Define custom labels
labels = ["Normal", "Suspect", "Pathological"]

# Create a pie chart to visualize the distribution
plt.figure(figsize=(15, 6))

# Define properties for wedge (slice) borders
wedgeprops = {'edgecolor': 'black', 'linewidth': 1}

# Create the pie chart
plt.subplot(1, 2, 1)
plt.pie(health_counts, labels=labels, autopct='%1.1f%%', colors
    =["skyblue", "lightcoral", "lightgreen"],
```

```
        wedgeprops=wedgeprops)
plt.title("Distribution␣of␣Fetal␣Health␣(Pie␣Chart)")
plt.xlabel("Fetal␣Health␣Classes")
plt.ylabel("Count")

# Create a bar plot to visualize the distribution
plt.subplot(1, 2, 2)
sns.countplot(x="fetal_health", data=ctg_data, palette="viridis
    ", order=health_counts.index)
plt.title("Distribution␣of␣Fetal␣Health␣(Bar␣Plot)")
plt.xlabel("Fetal␣Health␣Classes")
plt.ylabel("Count")

# Adjust layout
plt.tight_layout()

# Save the combined visualizations as an image file (e.g., PNG)
save_filename_combined = 'fetal_health_distribution_combined.
    png'
plt.savefig(save_filename_combined)

# Show the combined visualizations
plt.show()



# Calculate summary statistics for the target variable
target_stats = ctg_data["fetal_health"].describe().to_frame().T

# Display the summary statistics in a table
print(tabulate(target_stats, headers='keys', tablefmt='pretty')
    )
```

CTG Features and Correlation

```
# Generate the pair plot
selected_columns = ['baseline value', 'accelerations', '
    fetal_movement', 'abnormal_short_term_variability', '
    uterine_contractions', 'fetal_health']
pair_plot = sns.pairplot(data=ctg_data[selected_columns], hue='
    fetal_health')

# Get the current working directory
current_directory = os.getcwd()

# Define the filename
filename = 'pair_plot.png'

# Create the full path to the image file
full_path = os.path.join(current_directory, filename)

# Save the pair plot as an image (e.g., PNG)
pair_plot.savefig(full_path)

# Show the pair plot
plt.show()
```

Histogram Analysis of Key Features

```
# List of important features
important_features = ['baseline value', 'accelerations', '
    uterine_contractions']

# Create a grid for the histograms
fig, axes = plt.subplots(1, len(important_features), figsize
    =(15, 5))
```

```
# Plot histograms for each important feature
for ax, feature in zip(axes, important_features):
    sns.histplot(ctg_data[feature], kde=True, color='skyblue',
        ax=ax)
    ax.set_title(f'Distribution of {feature}')
    ax.set_xlabel(feature)
    ax.set_ylabel('Frequency')

# Save the plot as an image (e.g., PNG)
plt.savefig('histograms.png')

plt.show()
```

SMOTE Implementation

```
# Assuming X, y are your features and target variable
# Define your data
y = ctg_data["fetal_health"]
X = A.drop(columns=['severe_decelerations'])

# Check the class distribution before oversampling
print("Class distribution before oversampling:", Counter(y))

# Oversampling using SMOTE
smote = SMOTE(sampling_strategy='auto', random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)
```

Feature Scaling and Train-Test Split

```
# Standardize features
scaler = StandardScaler()
X_resampled_standardized = scaler.fit_transform(X_resampled)
```

```
# Print class distribution after oversampling and before
    standardization
print("Class␣distribution␣after␣oversampling:", Counter(
    y_resampled))

# Train-Test Split with the standardized features
X_train, X_test, y_train, y_test = train_test_split(
    X_resampled_standardized, y_resampled, test_size=0.30,
    random_state=42, stratify=y_resampled)

# Check the class distribution in the training set after
    oversampling
print("Class␣distribution␣in␣the␣training␣set␣after␣
    oversampling:", Counter(y_train))

# Check the class distribution in the testing set after
    oversampling
print("Class␣distribution␣in␣the␣testing␣set␣after␣oversampling
    :", Counter(y_test))


# List of columns representing features
feature_columns = [
    'baseline␣value', 'accelerations', 'fetal_movement',
    'uterine_contractions', 'light_decelerations',
    'prolongued_decelerations', '
       abnormal_short_term_variability',
    'mean_value_of_short_term_variability',
    'percentage_of_time_with_abnormal_long_term_variability',
    'mean_value_of_long_term_variability', 'histogram_width',
    'histogram_min', 'histogram_max', '
       histogram_number_of_peaks',
```

```
    'histogram_number_of_zeroes', 'histogram_mode', '
        histogram_mean',
    'histogram_median', 'histogram_variance', '
        histogram_tendency'
]

# Create a DataFrame for the scaled features
X_resampled_standardized = pd.DataFrame(
    X_resampled_standardized, columns=feature_columns)

X_resampled_standardized.head()
```

Logistic Regression

```
# 1. Create a logistic regression model
logreg_model = LogisticRegression( solver='liblinear', max_iter
    =1000, random_state=42)

# 2. Fit the model on the training data
logreg_model.fit(X_train, y_train)

# 3. Make predictions on the test data
logreg_predictions = logreg_model.predict(X_test)

# 4. Evaluate the accuracy of the model
logreg_accuracy = accuracy_score(y_test, logreg_predictions)

# 5. Generate a classification report
logreg_class_report = classification_report(y_test,
    logreg_predictions)

# 6. Create a confusion matrix
logreg_conf_matrix = confusion_matrix(y_test,
```

```
    logreg_predictions)

# 7. Hyperparameter tuning using GridSearchCV
logreg_param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100]}
logreg_grid_search = GridSearchCV(logreg_model,
    logreg_param_grid, cv=10)
logreg_grid_search.fit(X_train, y_train)
logreg_best_params = logreg_grid_search.best_params_

# 8. Evaluate accuracy with the tuned model
tuned_logreg_model = logreg_grid_search.best_estimator_
tuned_logreg_predictions = tuned_logreg_model.predict(X_test)
tuned_logreg_accuracy = accuracy_score(y_test,
    tuned_logreg_predictions)

# 9. Cross-validation
logreg_cv_scores = cross_val_score(tuned_logreg_model, X_train,
    y_train, cv=10)


# Print results in tabular format
headers = ["Step", "Result"]
table_data = [
    ["1. Create a logistic regression model", "Done"],
    ["2. Fit the model on the training data", "Done"],
    ["3. Make predictions on the test data", "Done"],
    ["4. Evaluate accuracy of the model", f"{logreg_accuracy:.3
        f}"],
    ["5. Generate a classification report", f"\n{
        logreg_class_report}"],
    ["6. Create a confusion matrix", f"\n{logreg_conf_matrix}"
        ],
```

```
    ["7. Hyperparameter tuning using GridSearchCV", f"Best
        Hyperparameters: {logreg_best_params}"],
    ["8. Evaluate accuracy with the tuned model", f"Accuracy
        with Tuned Model: {tuned_logreg_accuracy:.3f}"],
    ["9. Cross-validation", f"Scores: {','.join(map(lambda x: f
        '{x:.3f}', logreg_cv_scores))}"]]


print(tabulate(table_data, headers=headers, tablefmt="grid"))
```

Feature Contributions and Coefficients

```
# Get feature coefficients and names
coef = tuned_logreg_model.coef_[0]
feature_names = X.columns

# Plot feature importance
plt.figure(figsize=(12, 8))  # Increase the figure size
plt.barh(feature_names, coef, color='skyblue')
plt.xlabel('Coefficient Magnitude')
plt.title('Logistic Regression Coefficients')
plt.savefig('logreg_coefficients_plot.png', bbox_inches='tight'
    )  # Save the plot with tight layout
plt.show()
```

KNN

```
# 1. Create a K-Nearest Neighbors model
knn_model = KNeighborsClassifier()

# 2. Fit the model on the training data
knn_model.fit(X_train, y_train)
```

```
# 3. Make predictions on the test data
knn_predictions = knn_model.predict(X_test)

# 4. Evaluate the accuracy of the model
knn_accuracy = accuracy_score(y_test, knn_predictions)

# 5. Generate a classification report
knn_class_report = classification_report(y_test,
    knn_predictions)

# 6. Create a confusion matrix
knn_conf_matrix = confusion_matrix(y_test, knn_predictions)

# 7. Hyperparameter tuning using GridSearchCV
knn_param_grid = {'n_neighbors': [3, 5, 7, 9]}
knn_grid_search = GridSearchCV(KNeighborsClassifier(),
    knn_param_grid, cv=10)
knn_grid_search.fit(X_train, y_train)
knn_best_params = knn_grid_search.best_params_

# 8. Evaluate accuracy with the tuned model
tuned_knn_model = knn_grid_search.best_estimator_
tuned_knn_predictions = tuned_knn_model.predict(X_test)
tuned_knn_accuracy = accuracy_score(y_test,
    tuned_knn_predictions)

# 9. Cross-validation
cv_knn_scores = cross_val_score(tuned_knn_model, X_train,
    y_train, cv=10)

# Print results in tabular format
headers = ["Step", "Result"]
```

```
table_data = [
    ["1. Create a KNN model", "Done"],
    ["2. Fit the model on the training data", "Done"],
    ["3. Make predictions on the test data", "Done"],
    ["4. Evaluate accuracy of the model", f"{knn_accuracy:.3f}"
       ],
    ["5. Generate a classification report", f"\n{
       knn_class_report}"],
    ["6. Create a confusion matrix", f"\n{knn_conf_matrix}"],
    ["7. Hyperparameter tuning using GridSearchCV", f"Best
       Hyperparameters: {knn_best_params}"],
    ["8. Evaluate accuracy with the tuned model", f"Accuracy
       with Tuned Model: {tuned_knn_accuracy:.3f}"],
    ["9. Cross-validation", f"Scores: {','.join(map(lambda x: f
       '{x:.3f}', cv_knn_scores))}"]]


print(tabulate(table_data, headers=headers, tablefmt="grid"))
```

Random Forest

```
# 1. Create a Random Forest model
rf_model = RandomForestClassifier(random_state=42)

# 2. Fit the model on the training data
rf_model.fit(X_train, y_train)

# 3. Make predictions on the test data
rf_predictions = rf_model.predict(X_test)

# 4. Evaluate the accuracy of the model
rf_accuracy = accuracy_score(y_test, rf_predictions)
```

```
# 5. Generate a classification report
rf_class_report = classification_report(y_test, rf_predictions)

# 6. Create a confusion matrix
rf_conf_matrix = confusion_matrix(y_test, rf_predictions)

# 7. Hyperparameter tuning using GridSearchCV
rf_param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
rf_grid_search = GridSearchCV(RandomForestClassifier(
   random_state=42), rf_param_grid, cv=10)
rf_grid_search.fit(X_train, y_train)
rf_best_params = rf_grid_search.best_params_

# 8. Evaluate accuracy with the tuned model
tuned_rf_model = rf_grid_search.best_estimator_
tuned_rf_predictions = tuned_rf_model.predict(X_test)
tuned_rf_accuracy = accuracy_score(y_test, tuned_rf_predictions
   )

# 9. Cross-validation
cv_rf_scores = cross_val_score(tuned_rf_model, X_train, y_train
   , cv=10)

# Print results in tabular format
headers = ["Step", "Result"]
table_data = [
    ["1. Create a Random Forest model", "Done"],
```

```
    ["2. Fit the model on the training data", "Done"],
    ["3. Make predictions on the test data", "Done"],
    ["4. Evaluate accuracy of the model", f"{rf_accuracy:.3f}"
      ],
    ["5. Generate a classification report", f"\n{
      rf_class_report}"],
    ["6. Create a confusion matrix", f"\n{rf_conf_matrix}"],
    ["7. Hyperparameter tuning using GridSearchCV", f"Best
      Hyperparameters: {rf_best_params}"],
    ["8. Evaluate accuracy with the tuned model", f"Accuracy
      with Tuned Model: {tuned_rf_accuracy:.3f}"],
    ["9. Cross-validation", f"Scores: {','.join(map(lambda x: f
      '{x:.3f}', cv_rf_scores))}"]]

print(tabulate(table_data, headers=headers, tablefmt="grid"))
```

Decision tree

```
# 1. Create a Decision Tree model
dt_model = DecisionTreeClassifier(random_state=42)

# 2. Fit the model on the training data
dt_model.fit(X_train, y_train)

# 3. Make predictions on the test data
dt_predictions = dt_model.predict(X_test)

# 4. Evaluate the accuracy of the model
dt_accuracy = accuracy_score(y_test, dt_predictions)

# 5. Generate a classification report
dt_class_report = classification_report(y_test, dt_predictions)
```

```
# 6. Create a confusion matrix
dt_conf_matrix = confusion_matrix(y_test, dt_predictions)

# 7. Hyperparameter tuning using GridSearchCV for Decision Tree
    (optional)
dt_param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
dt_grid_search = GridSearchCV(DecisionTreeClassifier(
   random_state=42), dt_param_grid, cv=10)
dt_grid_search.fit(X_train, y_train)
dt_best_params = dt_grid_search.best_params_

# 8. Evaluate accuracy with the tuned model (optional)
tuned_dt_model = dt_grid_search.best_estimator_
tuned_dt_predictions = tuned_dt_model.predict(X_test)
tuned_dt_accuracy = accuracy_score(y_test, tuned_dt_predictions
   )

# 9. Cross-validation (optional)
cv_dt_scores = cross_val_score(tuned_dt_model, X_train, y_train
   , cv=10)

# Print results in tabular format for Decision Tree
headers_dt = ["Step", "Result"]
table_data_dt = [
    ["1. Create a Decision Tree model", "Done"],
    ["2. Fit the model on the training data", "Done"],
    ["3. Make predictions on the test data", "Done"],
```

```
    ["4. Evaluate accuracy of the model", f"{dt_accuracy:.3f}"
        ],
    ["5. Generate a classification report", f"\n{
        dt_class_report}"],
    ["6. Create a confusion matrix", f"\n{dt_conf_matrix}"],
    ["7. Hyperparameter tuning using GridSearchCV", f"Best
        Hyperparameters: {dt_best_params}"],
    ["8. Evaluate accuracy with the tuned model", f"Accuracy
        with Tuned Model: {tuned_dt_accuracy:.3f}"],
    ["9. Cross-validation", f"Scores: {','.join(map(lambda x: f
        '{x:.3f}', cv_dt_scores))}"]]

print("\nResults for Decision Tree:\n")
print(tabulate(table_data_dt, headers=headers_dt, tablefmt="
    grid"))
```

SVM

```
# 1. Create a Support Vector Machine model
svm_model = SVC(random_state=42)

# 2. Fit the model on the training data
svm_model.fit(X_train, y_train)

# 3. Make predictions on the test data
svm_predictions = svm_model.predict(X_test)

# 4. Evaluate the accuracy of the model
svm_accuracy = accuracy_score(y_test, svm_predictions)

# 5. Generate a classification report
svm_class_report = classification_report(y_test,
```

```
    svm_predictions)

# 6. Create a confusion matrix
svm_conf_matrix = confusion_matrix(y_test, svm_predictions)

# 7. Hyperparameter tuning using GridSearchCV (optional)
svm_param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto']
}
svm_grid_search = GridSearchCV(SVC(random_state=42),
    svm_param_grid, cv=10)
svm_grid_search.fit(X_train, y_train)
svm_best_params = svm_grid_search.best_params_

# 8. Evaluate accuracy with the tuned model (optional)
tuned_svm_model = svm_grid_search.best_estimator_
tuned_svm_predictions = tuned_svm_model.predict(X_test)
tuned_svm_accuracy = accuracy_score(y_test,
    tuned_svm_predictions)

# 9. Cross-validation (optional)
cv_svm_scores = cross_val_score(tuned_svm_model, X_train,
    y_train, cv=10)

# Print results in tabular format for Support Vector Machine
headers_svm = ["Step", "Result"]
table_data_svm = [
    ["1. Create a Support Vector Machine model", "Done"],
    ["2. Fit the model on the training data", "Done"],
    ["3. Make predictions on the test data", "Done"],
```

```
    ["4. Evaluate accuracy of the model", f"{svm_accuracy:.3f}"
        ],
    ["5. Generate a classification report", f"\n{
        svm_class_report}"],
    ["6. Create a confusion matrix", f"\n{svm_conf_matrix}"],
    ["7. Hyperparameter tuning using GridSearchCV", f"Best
        Hyperparameters: {svm_best_params}"],
    ["8. Evaluate accuracy with the tuned model", f"Accuracy
        with Tuned Model: {tuned_svm_accuracy:.3f}"],
    ["9. Cross-validation", f"Scores: {','.join(map(lambda x: f
        '{x:.3f}', cv_svm_scores))}"]]

print("\nResults for Support Vector Machine:\n")
print(tabulate(table_data_svm, headers=headers_svm, tablefmt="
    grid"))
```

Learning Curves for all the Models

```
def plot_learning_curve(estimator, title, X, y, cv, train_sizes
    =np.linspace(.1, 1.0, 5)):

    """
    Generate a simple plot of the test and training learning
        curve.

    Parameters:
    - estimator: the machine learning model
    - title: title of the plot
    - X: training data
    - y: target values
    - cv: cross-validation iterator
    - train_sizes: relative or absolute numbers of training
        examples that will be used to generate the learning
```

```
    curve
    """
    plt.figure(figsize=(10, 6))
    plt.title(title)
    plt.xlabel("Training␣examples")
    plt.ylabel("Score")

    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, train_sizes=train_sizes,
            scoring='accuracy', n_jobs=-1)

    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.fill_between(train_sizes, train_scores_mean -
        train_scores_std,
                     train_scores_mean + train_scores_std,
                        alpha=0.1, color="r")
    plt.fill_between(train_sizes, test_scores_mean -
        test_scores_std,
                     test_scores_mean + test_scores_std, alpha
                        =0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
        label="Training␣score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
        label="Cross-validation␣score")

    plt.legend(loc="best")
    return plt
```

```
# Plot learning curve for Logistic Regression
plot_learning_curve(tuned_logreg_model, "Learning Curve (
    Logistic Regression)", X_train, y_train, cv=10)

# Save the plot before calling plt.show()
plt.savefig("learning_curve_logreg.png")

# Show the plot
plt.show()

# Plot learning curve for K-Nearest Neighbors
plot_learning_curve(tuned_knn_model, "Learning Curve (K-Nearest
    Neighbors)", X_train, y_train, cv=10)

# Save the plot before calling plt.show()
plt.savefig("learning_curve_KNN.png")

# Show the plot
plt.show()

# Plot learning curve for Random Forest
plot_learning_curve(tuned_rf_model, "Learning Curve (Random
    Forest)", X_train, y_train, cv=10)

# Save the plot before calling plt.show()
plt.savefig("learning_curve_rf.png")

# Show the plot
plt.show()

# Plot learning curve for Decision Tree
plot_learning_curve(tuned_dt_model, "Learning Curve (Decision
```

```
    Tree)", X_train, y_train, cv=10)

# Save the plot before calling plt.show()
plt.savefig("learning_curve_dt.png")

# Show the plot
plt.show()

# Plot learning curve for Support Vector Machine
plot_learning_curve(tuned_svm_model, "Learning␣Curve␣(Support␣
    Vector␣Machine)", X_train, y_train, cv=10)

# Save the plot before calling plt.show()
plt.savefig("learning_curve_svm.png")

# Show the plot
plt.show()
```

Confusion Matrix - Logistic Regression

```
# Assuming you already have y_test and predictions defined

# Get unique class labels
class_names = ["Normal", "Suspect", "Pathological"]


# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(logreg_conf_matrix, annot=True, fmt="d", cmap="
    Blues", cbar=False,
            xticklabels=class_names, yticklabels=class_names)
plt.title('Confusion␣Matrix␣-␣Logistic␣Regression')
plt.xlabel('Predicted')
```

```
plt.ylabel('True')
# Save the plot as an image file
plt.savefig('confusion_matrix_logreg.png')

plt.show()
```

Confusion Matrix - K-Nearest Neighbors

```
# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(knn_conf_matrix, annot=True, fmt="d", cmap="Blues",
    cbar=False,
            xticklabels=class_names, yticklabels=class_names)
plt.title('Confusion␣Matrix␣-␣K-Nearest␣Neighbors')
plt.xlabel('Predicted')
plt.ylabel('True')
# Save the plot as an image file
plt.savefig('confusion_matrix_knn.png')
plt.show()
```

Confusion Matrix - Random Forest

```
# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(rf_conf_matrix, annot=True, fmt="d", cmap="Blues",
   cbar=False,
            xticklabels=class_names, yticklabels=class_names)
plt.title('Confusion␣Matrix␣-␣Random␣Forest')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.savefig('confusion_matrix_rf.png')

plt.show()
```

Confusion Matrix - Decision Tree

```
# Assuming you already have y_test and predictions defined

# Get unique class labels
class_names = ["Normal", "Suspect", "Pathological"]


# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(dt_conf_matrix, annot=True, fmt="d", cmap="Blues",
    cbar=False,
            xticklabels=class_names, yticklabels=class_names)
plt.title('Confusion Matrix - DT')
plt.xlabel('Predicted')
plt.ylabel('True')
# Save the plot as an image file
plt.savefig('confusion_matrix_dt.png')

plt.show()
```

Confusion Matrix - SVM

```
# Assuming you already have y_test and predictions defined

# Get unique class labels
class_names = ["Normal", "Suspect", "Pathological"]


# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(svm_conf_matrix, annot=True, fmt="d", cmap="Blues",
    cbar=False,
```

```
            xticklabels=class_names, yticklabels=class_names)
plt.title('Confusion Matrix - SVM')
plt.xlabel('Predicted')
plt.ylabel('True')
# Save the plot as an image file
plt.savefig('confusion_matrix_svm.png')

plt.show()
```

Model Performance Metrics

```
# Model names
models = ['Logistic Regression', 'SVM', 'Decision Tree', 'KNN',
    'Random Forest']

# Performance metrics
accuracy = [86.2,94.2,95.3,95.3,98.1]
precision =   [87,94,95,96,98]
recall_f1_score = [86,94,95,95,98]


# Plotting
plt.figure(figsize=(10, 6))

# Plot accuracy
plt.plot(models, accuracy, marker='o', label='Accuracy', color=
   'blue')

# Plot precision
plt.plot(models, precision, marker='o', label='Precision',
   color='green')
```

```
# Plot recall and F1-score
plt.plot(models, recall_f1_score, marker='o', label='Recall␣and
   ␣F1-Score', color='orange')

# Set labels and title
plt.xlabel('Models')
plt.ylabel('Percentage')
plt.title('Model␣Performance␣Metrics')
plt.legend(loc='upper␣left')  # Adjust the location of the
   legend

# Save the plot as an image file
plt.savefig('model_performance_metrics.png')

# Show the plot
plt.show()
```

Classifier Performance Plot

```
# Classifier names
classifiers = ['Decision␣Tree\n(Huang␣et␣al.,\n␣2012)', 'EMD-
   SMV\n(Krupa␣et␣al.,\n␣2011)', 'SVM\n(Alam␣et␣al.,\n␣2022)',
    'Random␣Forest\n(This␣paper)', 'KNN\n(This␣paper)', '
    Decision␣Tree\n(This␣paper)', 'SVM\n(This␣paper)']

# Corresponding accuracies
accuracies = [86.36, 86, 96.57, 98.1, 95.3, 95.3, 94.2] # add
   accuracy values for the proposed DT and SVM]

# Set a common color for all bars
common_color = 'skyblue'

# Create a bar chart
```

```
plt.figure(figsize=(10, 6))
bars = plt.bar(classifiers, accuracies, color=common_color)

# Adding labels and title
plt.xlabel('Classifiers')
plt.ylabel('Accuracy (%)')
plt.title('Classifier Performance for Fetal Health
    Classification')
plt.ylim(0, 110)  # Set the y-axis range to 0-100%

# Adding percentage values on top of each bar
for bar, acc in zip(bars, accuracies):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height
        () + 1, f'{acc:.2f}%', ha='center', va='bottom')

plt.tight_layout()

# Save the plot as an image file (e.g., PNG)
plt.savefig('classifier_performance_plot.png')

# Show the plot
plt.show()
```

Bibliography

[1] Zarko Alfirevic, Declan Devane, Gillian ML Gyte, and Anna Cuthbert. Continuous cardiotocography (ctg) as a form of electronic fetal monitoring (efm) for fetal assessment during labour. *The Cochrane database of systematic reviews*, 2:CD006066, 2017.

[2] Robin Mennicken et al. Machine learning applications in prenatal care: A systematic review. *Journal of Medical Internet Research*, 25(1):e3025, 2023.

[3] D Wade. Ethics of collecting and using healthcare data. *BMJ (Clinical research ed.)*, 334(7608):1330–1331, 2007.

[4] Jean-Luc Aeberhard, Ana-Paula Radan, Ricard Delgado-Gonzalo, Katrin M Strahm, Hulda B Sigurthorsdottir, Stefan Schneider, and Daniel Surbek. Artificial intelligence and machine learning in cardiotocography: A scoping review. *European Journal of Obstetrics & Gynecology and Reproductive Biology*, 281:54–62, 2023.

[5] H. Y. Kim, G. J. Cho, and H. S. Kwon. Applications of artificial intelligence in obstetrics. *Ultrasonography (Seoul, Korea)*, 42(1):2–9, 2023.

[6] G. P. Dexter, S. J. Grannis, B. E. Dixon, and S. N. Kasthurirathne. Generalization of machine learning approaches to identify notifiable conditions from a statewide health information exchange. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, pages 152–161, 2020.

[7] H. Chen, J. Guo, C. Wang, F. Luo, X. Yu, W. Zhang, J. Li, D. Zhao, D. Xu, Q. Gong, J. Liao, H. Yang, W. Hou, and Y. Zhang. Clinical characteristics and intrauterine vertical transmission potential of covid-19 infection in nine pregnant women: a retrospective review of medical records. *The Lancet (London, England)*, 395(10226):809–815, 2020.

[8] Maria G Signorini, Nicola Pini, Alberto Malovini, Riccardo Bellazzi, and Giovanni Magenes. Integrating machine learning techniques and physiology based heart rate features for antepartum fetal monitoring. *Computer Methods and Programs in Biomedicine*, 185:105015, 2020.

[9] Daniel Schönberger. Artificial intelligence in healthcare: a critical analysis of the legal and ethical implications. *International Journal of Law and Information Technology*, 27(2):171–203, 2019.

[10] J. B. Krier, S. S. Kalia, and R. C. Green. Genomic sequencing in clinical practice: applications, challenges, and opportunities. *Dialogues in clinical neuroscience*, 18(3):299–312, 2016.

[11] Tim Beck, Robert Hastings, and S. et al. Gollapudi. Gwas central: a comprehensive resource for the comparison and interrogation of genome-wide association studies. *European Journal of Human Genetics*, 22:949–952, 2014.

[12] M. E. Shipe, S. A. Deppen, F. Farjah, and E. L. Grogan. Developing prediction models for clinical use using logistic regression: an overview. *Journal of thoracic disease*, 11:S574–S584, 2019.

[13] H. Ocak and H. M. Ertunc. "prediction of fetal state from cardiotocogram recordings using adaptive neuro-fuzzy inference systems.". *Neural Computing and Applications*, 23, 2013.

[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. *Applied Bionics and Biomechanics*, 2022:12, 2022. Article ID 6321884.

[15] S. Ali, F. Akhlaq, A. S. Imran, Z. Kastrati, S. M. Daudpota, and M. Moosa. "the enlightening role of explainable artificial intelligence in medical healthcare domains: A systematic literature review.". *Computers in Biology and Medicine*, 166:107555, 2023.

[16] Giovanni Sannino et al. Artificial intelligence in cardiotocography and perinatal care: A systematic review and meta-analysis. *PLOS ONE*, 16(5):e0252215, 2021.

[17] Leonard B Lusted. Signal detectability and medical decision making. *Science*, 171:1217–1219, 1971.

[18] Kelly H Zou, A James OâMalley, and Laura Mauri. Receiver-operating characteristic analysis for evaluating diagnostic tests and predictive models. *Circulation*, 115:654–657, 2007.

[19] Nancy A Obuchowski. Receiver operating characteristic curves and their use in radiology. *Radiology*, 229:3–8, 2003.

[20] Charles J Lloyd. Using smooth receiver operating characteristic curves to summarize and compare diagnostic systems. *Journal of the American Statistical Association*, 93:1356–1364, 1998.

[21] S. Uddin, I. Haque, H. Lu, and et al. "comparative performance analysis of k-nearest neighbor (knn) algorithm and its different variants for disease prediction.". *Scientific Reports*, 12:6256, 2022.

[22] H. Sahin and A. Subasi. "classification of the cardiotocogram data for anticipation of fetal risks using machine learning techniques.". *Applied Soft Computing*, 33:231–238, 2015.

[23] Gary Campbell. General methodology i: advances in statistical methodology for the evaluation of diagnostic and laboratory tests. *Statistics in Medicine*, 13:499–508, 1994.

[24] R. M. Grivell, Z. Alfirevic, G. M. Gyte, and D. Devane. Antenatal cardiotocography for fetal assessment. *The Cochrane database of systematic reviews*, 2015(9):CD007863, 2015.

[25] German Society of Gynecology, Obstetrics (DGGG), Maternal Fetal Medicine Study Group (AGMFM), German Society of Prenatal Medicine, Obstetrics (DGPGM), and German Society of Perinatal Medicine (DGPM). S1-guideline on the use of ctg during pregnancy and labor: Long version - awmf registry no. 015/036. *Geburtshilfe und Frauenheilkunde*, 74(8):721–732, 2014.

[26] C. M. Bhatt, P. Patel, T. Ghetia, and P. L. Mazzeo. Effective heart disease prediction using machine learning techniques. *Algorithms*, 16(2):88, 2023.

[27] M. I. Evans, D. W. Britt, S. M. Evans, and L. D. Devoe. Changing perspectives of electronic fetal monitoring. *Reproductive sciences (Thousand Oaks, Calif.)*, 29(6):1874–1894, 2022.

[28] Eric J Topol. High-performance medicine: the convergence of human and artificial intelligence. *Nature Medicine*, 25(1):44–56, 2019.

[29] Caitlin McNestry, Sarah L Killeen, Rachel K Crowley, and F. M. McAuliffe. Pregnancy complications and later life women's health. *Acta obstetricia et gynecologica Scandinavica*, 102(5):523–531, 2023.

[30] S. Braunthal and A. Brateanu. Hypertension in pregnancy: Pathophysiology and treatment. *SAGE open medicine*, 7:2050312119843700, 2019.

[31] Ming-Liang Huang and Yung-Yu Hsu. Fetal distress prediction using discriminant analysis, decision tree, and artificial neural network. *Journal of Biomedical Science and Engineering*, 5(9):526–533, 2012.

[32] N Krupa, M Ali, E Zahedi, S Ahmed, and FM Hassan. Antepartum fetal heart rate feature extraction and classification using empirical mode decomposition and support vector machine. *Biomedical Engineering*, 10(1), 2011.

[33] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 2002.

[34] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant. *Applied Logistic Regression*. Wiley, 2013.

[35] A. Agresti. *Categorical Data Analysis*. Wiley, 2002.

[36] D. R. Cox. *The Analysis of Binary Data*. Chapman and Hall, 1970.

[37] P. Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 2012.

[38] D. PetkoviÄ et al. k-nearest neighbors in breast cancer prediction on three real-world data sets. *Expert Systems with Applications*, 2019.

[39] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[40] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2011.

[41] X. Chen and H. Ishwaran. Random forests for genomic data analysis. *Genomics*, 2012.

[42] A. Liaw and M. Wiener. Random forest: A versatile tool for classification and regression in pharmaceutical research. *Pharmaceutical Research*, 2002.

[43] P. N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[44] E. H. Baehrecke et al. Decision trees: An overview and their use in medicine. *IEEE Transactions on Systems, Man, and Cybernetics*, 1987.

[45] D. M. Patil et al. A comparative study of decision tree id3 and c4.5. *International Journal of Innovative Technology and Exploring Engineering*, 2014.

[46] Kuo-Chen Chou and De-Shuang Huang. *Support Vector Machines and Their Applications in Chemistry*. Springer, 2004.

[47] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. *Technical Report, Department of Computer Science, National Taiwan University*, 2003.

[48] Melba M. Crawford and Le Wang. Support vector machines for classification in remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2006.

[49] Margaret C Hogan, Kyle J Foreman, Mohsen Naghavi, Stephanie Y Ahn, Mengru Wang, Susanna M Makela, Alan D Lopez, Rafael Lozano, and Christopher JL Murray. Maternal mortality for 181 countries, 1980-2008: a systematic analysis of progress towards millennium development goal 5. *The Lancet*, 375(9726):1609–1623, 2015.

[50] Jeff Hammerbacher. The care and feeding of data scientists: Developing data scientists in your organization. *Harvard Data Science Review*, 2019.

[51] Bartosz Mikulski. Handling missing data in machine learning datasets. *Towards Data Science*, 2019.

[52] Alice Zheng and Amanda Casari. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly Media, 2018.

[53] Gilles Louppe, Louis Wehenkel, and Antonio Sutera. Understanding variable importances in forests of randomized trees. *Advances in Neural Information Processing Systems*, 2013.

[54] Rafael Barros et al. The impact of over-sampling and under-sampling on the performance of svm and random forest classifiers in the presence of noisy labels. *Knowledge-Based Systems*, 2017.

[55] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. *International Conference on Intelligent Computing*, 2005.

[56] Jason Brownlee. A gentle introduction to normalization and standardization in machine learning. 2018.

[57] Will Koehrsen. Train-test split and cross validation: A conceptual explanation. 2018.

Dataset Information

You can find the dataset used in this study directly from **Kaggle**.
The raw data can also be found on **UCI Machine Learning Repository**.