A black background with white text

Description automatically generated with low confidence

MSc Data Science Project

7PAM2002-0509-2023

Department of Physics, Astronomy and Mathematics

**Data Science FINAL PROJECT REPORT**

**Project Title:**

Enhancing Maternal Health Risk Prediction: A Comparative Analysis of Original vs. Feature-Engineered Data Using Traditional Machine Learning Models, FNN, and Ensemble Meta-Models

By:

Ukonu Chizoba Maryann- 21089329

Supervisor: Dr. Klaas Wiersema

Date Submitted: 28th August 2024

Word Count: 7350

DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science at the University of Hertfordshire.

I have read the guidance to students on academic integrity, misconduct, and plagiarism information at [Assessment Offences and Academic Misconduct](https://www.herts.ac.uk/__data/assets/pdf_file/0007/237625/AS14-Apx3-Academic-Misconduct-v17.0.pdf) and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project module or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7, and UPR AS/C/5, section 3.6). I have not used chatGPT, or any other generative AI tool, to write the reportor code (other than where declared or referenced).

I did not use human participants or a survey my MSc Project.

I hereby permit the report to be made available on module websites provided the source is acknowledged.

Student Name printed: Ukonu Chizoba Maryann

Student Name Signature: A blue signature on a white background

Description automatically generated

Student SRN number: 21089329

Acknowledgment

I thank God for His infinite grace and mercies, and for the health, strength, and resources to begin and complete this program.

I want to acknowledge the incredible people who played pivotal roles in this journey. To my sister, Chiamaka, who always made everything seem possible—your unwavering support means the world. The Akabuogus, thank you for giving me a soft landing and helping me settle in; I will always be grateful. A special thanks to Aunty Nky, NK, Stanley, Dozie, and Aunty Chinwe—your financial and emotional support was deeply felt and appreciated.

To Aunty Chikoo, who brought laughter to lighten the burdens of this process, I truly appreciate you. Aunty Ify, Anuli, and Onyii, your words of encouragement were priceless.

To Uno and her husband, thank you for being instrumental in this journey.

To my mum, your wisdom and affirmations were exactly what I needed to push forward. May God continue to strengthen you.

I must also recognize my coursemates—great men and women who were always willing to share insights and ideas. I’m grateful our paths crossed, and I thank you all. You’re stars.

A heartfelt thank you to my supervisor, Dr. Klaas Wieresema, for your invaluable corrections, guidance, and constant availability. Your input was crucial to the completion of this project.

To my husband and children, thank you for your sacrifices, patience, and understanding. You bore my absences and supported me despite my shortcomings during this journey. I love you all.

And lastly, to my Dad—I've done it. Though not in your lifetime, I know you're smiling down on me.

Abstract.

Maternal health is the bedrock of a nation’s well-being. Its accurate risk prediction is crucial in the healthcare sector as it enables health practitioners to detect potential issues early and provide timely management, thereby reducing its effect on both mother and child. This study examines the effectiveness of traditional machine learning models and a Feedforward Neural Network (FNN) in predicting maternal health risks using the original and feature-engineered data.

The research compares the predictive performance of individual models and explores the impact of forming an ensemble model by combining the predictive probabilities of these models using original data. Furthermore, a meta-model is trained using the ensembled probabilities as features, and its performance is evaluated to determine if it enhances accuracy and generalization compared to the individual models.

The study also examines the role of feature engineering in improving model performance, particularly focusing on how it affects the predictive accuracy and generalization ability of both the individual models and the meta-model.

The results provide comprehensive comparison between models trained on original data and those trained on the feature-engineered data. This offers insights into the optimal strategy for enhancing maternal health risk prediction.

By identifying the most effective approach to predictive modeling in maternal health, this study contributes to the field, with the goal of enhancing early detection and intervention strategies.

Table of Contents

[CHAPTER 1 1](#_Toc175604986)

[INTRODUCTION 1](#_Toc175604987)

[1.1 Background 1](#_Toc175604988)

[1.2 Problem Statement 4](#_Toc175604989)

[1.3 Research Question 4](#_Toc175604990)

[1.4 Research Objectives 5](#_Toc175604991)

[1.5 Significance of the study 5](#_Toc175604992)

[CHAPTER 2 6](#_Toc175604993)

[REVIEW OF LITERATURE 6](#_Toc175604994)

[2.1.0 Causes of Maternal Health Risks 6](#_Toc175604995)

[2.1.1 Effects of Maternal Health Risks. 8](#_Toc175604996)

[2.2 Traditional Approached to Maternal Health Risk Prediction 8](#_Toc175604997)

[2.3 Machine Learning in Healthcare 9](#_Toc175604998)

[2.4 Related Work 10](#_Toc175604999)

[CHAPTER THREE 12](#_Toc175605000)

[METHODOLOGY 12](#_Toc175605001)

[3.2 Data Overview. 12](#_Toc175605002)

[3.3 Data Analysis. 14](#_Toc175605003)

[3.4 Data preprocessing and cleaning. 15](#_Toc175605004)

[3.5 Model Building. 17](#_Toc175605005)

[3.6 Selected Models Definitions 19](#_Toc175605006)

[CHAPTER 4 20](#_Toc175605007)

[RESULT ANALYSIS 20](#_Toc175605008)

[4.2 Performance metrics. 21](#_Toc175605009)

[4.3 Performance Analysis of Original Data 22](#_Toc175605010)

[4.4 Performance Analysis of Models on Feature-Engineered Data. 25](#_Toc175605011)

[4.5 Performance comparison of All models 26](#_Toc175605012)

[CHAPTER FIVE 30](#_Toc175605013)

[CONCLUSION 30](#_Toc175605014)

[5.1 Comparative Analysis with Existing Literature 30](#_Toc175605015)

[5.2 Limitations of the study 31](#_Toc175605016)

[5.3 Conclusion 31](#_Toc175605017)

[References 32](#_Toc175605018)

[Appendix 34](#_Toc175605019)

# CHAPTER 1

# INTRODUCTION

# 1.1 Background

Maternal health (MH) encompasses the health and well-being of an expectant mother. It is women's physical, mental, and social well-being during **pregnancy**, **childbirth**, and **postnatal**. The term “**Pregnancy**” refers to when an egg fertilizes, implants, and develops into a fetus inside a woman’s uterus over approximately 9 months, culminating in childbirth. **“Childbirth”** is the process of delivering a developed fetus either via the vagina (vaginal delivery) or by surgical intervention (cesarean session). **“Postnatal”** is the care a woman and the child receive after childbirth. It is pertinent to state that each of these three (3) phases should be a good experience, making sure that women and their babies can be as healthy and happy as possible (WHO, 2024). For decades, including the 1980s, plans to improve maternal health in developing nations were almost non-existent in the Global health agenda. It was not until 1985, after an article published by Lancet with the subheading, “Where is the M in MCH?” that public health sectors took note of the troubling numbers (approx. half a million) of women dying per minute, due to avoidable complications from pregnancy and childbirth.(Rosenfield and Maine, 1985). The significance of good maternal health cannot be overemphasized. It not only lowers maternal mortality but also significantly reduces the risk of maternal morbidity. **Maternal mortality** refers to the death of a woman due to pregnancy or pregnancy-related complications. This includes the period from the start of pregnancy up to 42 days after the termination of the pregnancy, whether through delivery or other means. **Maternal Morbidity** highlights the aggravation of health conditions that existed before pregnancy and childbirth, which introduces higher risks to a woman’s health (Firoz et al., 2013).

The World Health Organization (WHO) in its fact sheets, published 26th April 2024, states that about 287000 women died during and following pregnancy and childbirth in 2020. it further listed that though other complications may exist before pregnancy, the following complications account for nearly 75% of all maternal deaths.

* hemorrhage
* Infections usually after birth
* High blood pressure during pregnancy
* Obstetric complications
* Unsafe abortion.

The Safe Motherhood Initiative (SMI), launched in 1987 by the World Bank, WHO, and United Nations Fund for Population Activities (now UNFPA), aimed to reduce maternal mortality in developing countries by 50% in one decade. This marked a pivotal moment in global maternal health policy for the first time. It gained momentum through international efforts in the following years, helping to elevate maternal health on the global agenda. This culminated in including maternal health in the Millennium Development Goals and later in the Sustainable Development Goals, demonstrating its continued importance as a Global health priority (De Brouwere and Van Lerberghe, 2001). The Global initiatives have aimed to address the alarmingly high maternal mortality rates in low- and middle-income countries, with a particular focus on Sub-Saharan Africa and Asia, where most maternal deaths occur.

The global community continues to grapple with the challenges of reducing maternal mortality and improving maternal health outcomes, as evidenced by the shift from Safe Motherhood to more recent efforts like "Ending Preventable Maternal Mortality" (EPMM) in 2014. This evolution reflects a broader understanding of the determinants of maternal health and the need for multipronged actions to address them. This shift in approach recognizes that maternal health is not just a medical issue, but an intricate one, involving economic and socio-political aspects of life.

However, persistent challenges continue to impede progress. Significant disparities in maternal health outcomes exist between and within countries, with women in low-income countries and marginalized communities facing higher risks (UNICEF, 2020; WHO, 2023). In 2020, the lifetime risk of maternal death in low-income countries was 1 to 49, compared to 1 in 5,300 in high-income countries (WHO, 2023). This glaring disparity between low and high-income maternal health outcomes sheds light on the substantial imbalance in maternal health issues on a global scale. Sub-Sahara Africa and Southern Asia account for approximately 86% of global maternal deaths with Sub-Sahara Africa recording 533 deaths per 100,000 live births as compared to developed countries with 10 deaths per 100,000 live births (UNICEF, 2023). In 2020, the WHO reported that skilled health personnel assisted with only 60% of births in low-income countries as against 99% in high-income countries (WHO, 2021). The disparity in maternal healthcare outcomes encompasses several critical components that need to be addressed.

Maternal Health is significantly impacted by a woman’s level of education. Women with no formal education face a 2.7 times higher risk of maternal death/complications compared to women who have completed more than 12 years of schooling. Similarly, women with 1-6 years of education are twice as likely to experience maternal mortality compared to those with higher levels of education (Karlsen et al., 2011). The difference between the maternal health risk of education and uneducated women highlights the crucial role education plays in maternal health outcomes. It enables women to actively participate in making informed decisions regarding their reproductive health.

Improved antenatal care coverage which has helped in identifying and managing potential complications in early pregnancy played a crucial role in the maternal mortality decline (Moller et al., 2019). Furthermore, maternal health concerns have gained international recognition, resulting in targeted interventions and policy efforts. The number of births attended by skilled health personnel rose from 58% in 1990 to 81% in 2019. This progress has partly contributed to the decline in the global maternal mortality ratio by about 34%— a remarkable improvement in maternal survival rates worldwide (WHO, 2024).

While the mortality ratio has experienced substantial declines worldwide, maternal morbidity has not shown the same degree of progress and continues to be a significant worry. For every maternal death, an average of 20-30 women experience acute or chronic morbidity (Firoz et al., 2013). This means that millions of women around the world experience pregnancy-related complications every year. The effect of the various pregnancy-related complications on women’s well-being can persist for an extended period, even after the immediate postpartum period has elapsed. Due to a lack of standardized definitions and measurement tools, maternal morbidity is often underreported and underrecognized (Chou et al., 2016). As with maternal mortality, maternal morbidity also has a more significant effect on women in countries with low and middle incomes, as well as on marginalized populations in high-income countries (Graham et al., 2016).

The issue of maternal health is multifaceted and presents a complex challenge in the healthcare sector. The use of machine learning (ML) in the healthcare sector in recent years has grown exponentially, with the most reviewed articles published in the last five years (Mennickent et al., 2023). The technology has shown great potential with promising results in different areas of healthcare, including but not limited to diagnosis, treatment planning, and patient monitoring (Topol, 2019).

# 1.2 Problem Statement

Traditional methods for predicting maternal health risks often rely on incomprehensive clinical records and static risk models, which may not capture the intricate nature of pregnancy-related complications. Thus, this underscores the need for holistic approaches in risk assessment and care provision, highlighting the potential value of integrating advanced technologies like the Internet of Things (IoT) and machine learning to address these diverse aspects effectively (Marques et al., 2020).

Hence this study. A comprehensive examination of current ML models in selected existing research will be carried out. This will guide to development of a robust predictive model by leveraging both traditional model algorithms and neural networks, and to improve predictive accuracy through an ensemble approach that combines the prediction probabilities of these models.

# 1.3 Research Question

This study intends to answer the below question:

“How does the predictive performance of maternal health risk models compare when using original data versus feature-engineered data, considering individual traditional ML models, a Feedforward Neural Network (FNN), and their ensemble combinations? Specifically, can a meta-model trained on the combined predictions of these models enhance accuracy compared to using the models individually?”

# 1.4 Research Objectives

This study aims to:

1. Use the original data to evaluate and compare the predictive performance of selected traditional models and the FNN model
2. Examine the impact of forming an ensemble model by combining prediction probabilities of the models in (1) and examine if the predictive and generalization improves compared to the individual models
3. Train a meta-model using ensembled probabilities in (2) as features to investigate further which gives a robust performance.
4. Investigate the role of feature engineering in the individual models and meta-models and determine its effect on predictive and generalization ability.
5. Provide a comprehensive comparison between the original data and feature-engineered data in the context of model performance, to identify the optimal strategy for enhancing maternal health risk prediction.

# 1.5 Significance of the study

This comprehensive approach, barring socio-economic factors and underlying health conditions, aims to enhance healthcare outcomes for expectant mothers by:

* Personalized Care Plans: Early detection of individual risk factors will help healthcare providers in offering person-centered care based on individual needs. Thus, each mother receives the most appropriate and effective care based on her unique circumstances, ensuring better health outcomes (Raza et al., 2022)
* Reduction in Maternal Mortality and Morbidity: Early interventions will be employed owing to timely management of severe complications. This will prevent the conditions from escalating to loss of life or life-threatening issues.
* Increase Maternal Education and Commitment: Timely detection will allow healthcare providers to enlighten pregnant women about their health risks and engage them in their care plans. Patients aware of their pregnancy risks are more likely to comply with medical advice and make informed decisions about their health (Afreen et al., 2021)
* Encouragement of Data-Driven Healthcare: The predictive accuracy of this model will help in encouraging data-driven approaches in healthcare decision-making, thereby reducing reliance on subjective judgment.
* Contribution to National Development: Improving maternal health outcomes has far-reaching implications for national development. Healthy mothers are more likely to raise healthy children, participate in the workforce, and contribute to economic growth (WHO, 2019)

# CHAPTER 2

# REVIEW OF LITERATURE

# 2.1 Overview of Maternal Health Risk

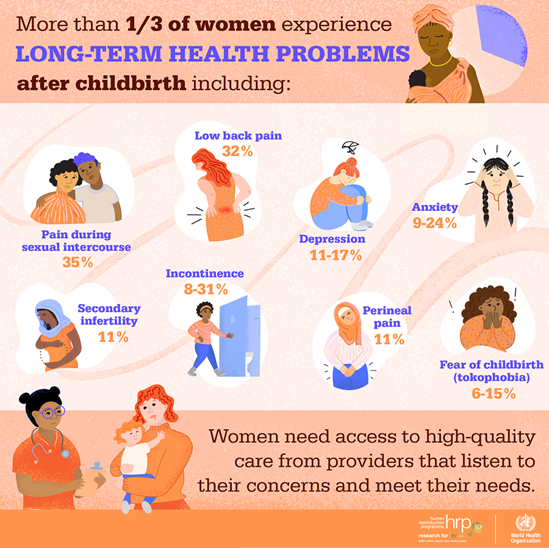
Maternal health risk references factors that may adversely affect a mother and her baby’s health when poorly managed. Sound knowledge of the risk inherent in these stages is crucial in developing and formulating effective interventions that could improve maternal and fetal outcomes.

# 2.1.0 Causes of Maternal Health Risks

Maternal health risks can arise from genetics, physiology, lifestyle, failure in the healthcare system, exposure to environmental dangers, or any combination. Awareness of these risk factors is necessary for evolving ways of addressing and preventing maternal health problems. The below subheadings summarize factors contributing to maternal health risk.

* Biological/Genetic factors: Genetic predisposition plays a crucial role in maternal health risks. Understanding family health history can help combat the possibility of maternal health risk. Genetic factors such as hypertension, diabetes, kidney disease, thyroid disease, etc. can increase the chances of conditions such as gestational diabetes, preeclampsia, and congenital anomalies. For instance, a history of preeclampsia in a family would suggest the same conditions could arise in patients with such a history if preemptive measures are not taken. (Ahmed et al., 2020). Age can crucially affect maternal health. Adolescents and older women (above 35 years) are more likely to develop health conditions related to pregnancy. Data shows that conditions such as gestational diabetes, hypertension, and chromosomal abnormalities like Down syndrome are more likely to develop in older pregnant women, complicating the pregnancies and affecting fetal growth. (Afreen et al., 2021). Similarly, adolescent pregnancies are often associated with higher rates of preeclampsia, anemia, and cephalopelvic disproportion, which can lead to obstructed labor and the need for surgical interventions (Mutlu et al., 2023).
* HealthCare System Factors: In many low-resource settings, women may not have access to regular antenatal visits, essential screenings, or the presence of skilled healthcare providers. These constraints increase the likelihood of undetected complications and poor management of existing conditions.
* Environmental Factors: Women living in areas with inadequate sanitation, lack of clean water, and exposure to environmental pollutants are at risk of complications in pregnancy. They are more susceptible to infections, which can complicate pregnancy and lead to adverse outcomes for both mother and fetus (Marques et al., 2020). Similarly, occupational hazards can pose significant risks to pregnant women. Prolonged standing, heavy lifting, and exposure to harmful chemicals in the workplace can cause complications in pregnancy. Inadequate workplace protections and limited access to maternity leave often compound these occupational hazards, which can aggravate health issues (Ghassemi et al., 2020).

# 2.1.1 Effects of Maternal Health Risks.

The implications of maternal health risks are far-reaching. Its implications extend beyond mother and child, including socioeconomic factors and long-term generational impacts.

**picture 1-effects of maternal health problems(**[**https://www.who.int/health-topics/maternal-health#tab=tab\_1**](https://www.who.int/health-topics/maternal-health#tab=tab_1)**)**

One of the most significant implications of maternal health risk is the likelihood of a surge in its mortality and morbidity rate and this has an exacerbating effect on the nation’s economy.

# 2.2 Traditional Approached to Maternal Health Risk Prediction

Primarily, the prediction of maternal health risk relies on clinical assessment. This involves monitoring various risk factors contributing to adverse maternal and fetal outcomes such as age, medical history like hypertension and diabetes, obstetric history like previous pregnancy outcomes, and or maternal/family lifestyle. Also, healthcare professionals use assessment tools like the Bishop Score to ascertain the risk level and the need for constant checks. The Bishop Score tool is used in assessing the readiness of the cervix for labor giving a guide to the likelihood of a woman going into labor or if induction is needed to facilitate labor. Another traditional approach used in maternal risk prediction is clinical guidelines and protocols. For example, guidelines from the WHO emphasize the importance of regular antenatal visits, screening for gestational diabetes, and monitoring blood pressure to detect and manage preeclampsia (WHO, 2019).

Traditional methods for predicting maternal health risks are well utilized but have limitations. One such limitation is the dependence on fixed risk factors, which may not completely reflect the ever-changing nature of pregnancy. These limitations hinder the accuracy and effectiveness of maternal health risk prediction.

# 2.3 Machine Learning in Healthcare

Recently, leveraging advanced technologies like Machine learning (ML), and the Internet of Things (IoT) and personalized predictions has garnered momentum. Its integration holds promising potential in addressing the limitations of traditional risk prediction/assessment methods, thereby further enhancing the accuracy of risk assessment and its management.

The term “Internet of Things” (IoT) comprises many physical devices equipped with sensors and other technology to enable communication and data exchange. Its concept focuses on seamless device connectivity and data sharing to provide real-time data updates. An example of such a device is the wearable sensor.

Machine learning as an area of expertise in Artificial Intelligence is focused on examining and creating statistical algorithms that can learn from data and extrapolate from the learned data, to interpret and predict other related data. The data fed into it could be structured or unstructured. Structured data are presented in a tabular form with rows and columns in a clear and consistent format. In healthcare, some structured data may include patients’ demographics, medical history, lab results, treatment plans, etc. This info can be organized systematically. For example, electronic health records (EHRs) often contain structured data fields for patient names, dates of birth, medications, and diagnosis codes, allowing healthcare providers to quickly access and analyze patient information (Zeng et al., 2019). Data without a predefined format is unstructured. While they may contain facts and data such as dates and numbers, they are also text-heavy. In healthcare, unstructured data includes clinical notes and reports that contain detailed patient information and observations. Medical imaging reports, patient feedback, and surveys, including open-ended responses from patients on their experiences and symptoms, also form unstructured data. Audio and video recordings from patient consultations that capture verbal and non-verbal cues essential for comprehensive patient care further contribute to unstructured data.

In the healthcare sector, predictive analysis and risk assessment are profound ML applications. In medical image analysis, deep learning algorithms have demonstrated the potential to detect various types of radiological cancers, exceeding human experts' assessment (Ghassemi et al., 2020). In the pharmaceutical industry, ML is transforming drug discovery and development. Its predictive ability, regarding the interactions of different chemical compounds with biological targets, can accelerate the drug discovery process and reduce the time between the discovery and the introduction of the discovered drug into the market (Raza et al., 2022). Natural Language Processing (NLP), an arm of ML, is making giant strides in analyzing electronic health records. NLP can extract important clinical records from unstructured clinical notes, thereby making important patient information more accessible to healthcare providers. This ensures that important information is not overlooked.

The synergy between ML and IoT can enable more proactive and personalized healthcare (Marques et al., 2020).

# 2.4 Related Work

The study by Togunwa et al. presents a deep hybrid model for Maternal Health risk classification during pregnancy. This study combines the strength of an Artificial Neural Network (ANN) and Random Forest classifier (RF) algorithms. The study aims to improve the accuracy of using data generated via IoT in developing countries (Bangladesh) to classify risk in pregnant women. The data used for the task has six (6) features; blood sugar, body temperature, diastolic bp, systolic bp, Heart rate, and Age. The data was split into 75% for training and 25% for testing. The study shows that the RF classifier, known for handling high-dimensionality data while mitigating overfitting recorded an accuracy of 89%. Similarly, the ANN model, known for its ability to capture complex and non-linear relationships achieved an accuracy of 71%. The predictions obtained from these two (2) models were combined using a maximum probability voting system to build a deep hybrid model with an accuracy of 95%. The increase in accuracy of the deep hybrid model could be attributed to the fact that the individual models in the hybrid differ in misclassifications, consequently leveraging on each other’s distinctive behavior to provide more accurate classification. The performance evaluation of the deep hybrid across other metrics like recall, precision, and F1 score was 97%. The consistently high score across all evaluation metrics shows that the deep hybrid model can effectively classify maternal risk.

The authors concluded that their approach can improve health outcomes for pregnant women and their babies. They suggested the direction future research may pursue, including exploring generalizing the model to other populations, incorporating unstructured medical data, and evaluating the feasibility of clinical use. The study underscores the potential of AI-based models in enhancing maternal healthcare by providing timely and accurate risk assessments (Togunwa et al., 2023).

Leila Jamil et al. (2022) in their study “Improving Prediction of Maternal Health Risks using PCA Features and TreeNet” aim to enhance the prediction of maternal health risks by using principal component analysis (PCA) for feature extraction and employing an ensemble learning approach. The study proposed a stacked ensemble voting classifier that combines one ML and one deep learning model. The dataset used for this study consists of 1014 samples collected from maternal healthcare facilities. The dataset features comprise age, systolic bp, diastolic bp, blood glucose level, body temperature, heart rate, and estimated risk density. Different ML models such as Random Forest, Extra Trees, Gradient Boost, AdaBoost, Multilayer Perceptron (MLP), Decision Tree, Logistic Regression, and Convolutional Neural Network (CNN) were deployed for this task. To improve the model performance. PCA was employed to extract significant features from the dataset, thus reducing dimensionality. The models were trained on both the original data and the PCA-selected features. An ensemble model is created using the trained models, which is a combination of Extra Trees and MLP. The result showed that the ensemble model on both the original data and PCA-based features outperformed other models deployed for this study with an accuracy of 80% on the original features and 98.25% on the PCA-based features.

The study highlights that the use of PCA for feature extraction, combined with an ensemble learning approach, significantly improves the accuracy of maternal health risk prediction.

Ali Raza et al. (2022) in their research “Ensemble learning-based feature engineering to analyze maternal health during pregnancy and health risk prediction” developed a robust system for predicting maternal health risks during pregnancy. The dataset utilized for this research had 1218 samples collected from maternal healthcare facilities, hospitals, and community clinics using an IoT-based risk monitoring system. The study focuses on improving prediction accuracy through ensemble learning and feature engineering. The study proposed a novel deep neural network architecture called DT-BiLTCN, which integrates decision trees, bidirectional long-short-term memory networks, and temporal convolutional networks. The Synthetic Minority Oversampling Technique (SMOTE) was used to address class imbalance in the dataset. Feature extraction was done using the proposed DT-BiLTCN model and trained on various ML classifiers, including Support Vector Machines (SVM). SVM showed superior performance compared to other ML classifiers like Decision Tree, Logistic Regression, K-nearest Neighbor, Extra Trees, and Random Forest with a score of 98% across all evaluation metrics- accuracy, precision, recall, and F1 score. The high score by SVM using the DI-BiLTCN predictions as features suggest that the proposed ensemble learning approach with feature extraction is highly effective in predicting and classifying maternal health risks.

# CHAPTER THREE

# METHODOLOGY

# 3.1 Data Overview.

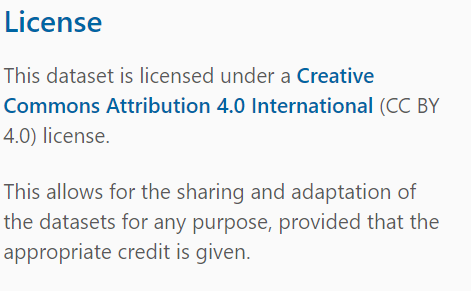
The dataset used in this research was obtained from the University of California Irvine (UCI) ML repository. It was created by Marzia Ahmed (A.Marzia et al, 2020) of Daffodil International University through data collected via IoT used in different hospitals, community clinics, and maternal health care in the rural areas of Bangladesh. The data was generated via wearable sensing technology. The table below is a summary of the features of the dataset.

A screenshot of a medical report

Description automatically generated

***Table 1 Data set features description***

The data is provided in a comma-separated values (CSV) format, widely compatible with data analysis tools. It contains 1014 samples that span over 7 features. The samples in the dataset meet the General Data Protection Regulation (GDPR) requirements on data protection as the data is anonymized and does not contain personally identifiable information. The dataset size is approximately 36KB, making it manageable for analysis without requiring extensive computational resources. Furthermore, the creators of the dataset have made it available to be used by the public for educational and research purposes as shown below.



***Picture 2- Data usage License***

# 3.2 Data Analysis.

A close-up of a chart

Description automatically generated

***Figure 1. Target class distribution***

Figure 1 shows the class distribution of our target variable ‘RiskLevel’. It is seen that there is some degree of imbalance among the classes.

A diagram of a graph

Description automatically generated with medium confidence

**Figure 2. Plot of Age, SystolicBP and RiskLevel Figure 3. Plot of Age, BS and RiskLevel**

Figure 2 and Figure 3 show that risk levels are spread across different ages both younger and older. In both figures, Low risk is color-coded in red, mid-risk orange, and high-risk gray. In Figure 2, the risk level progresses as BS increases. The same pattern is noticed in Figure 3, where the risk level progresses as the systolic BP increases. Women with higher BS and or SystolicBP tend to be associated with higher risk in pregnancy.

A screenshot of a graph

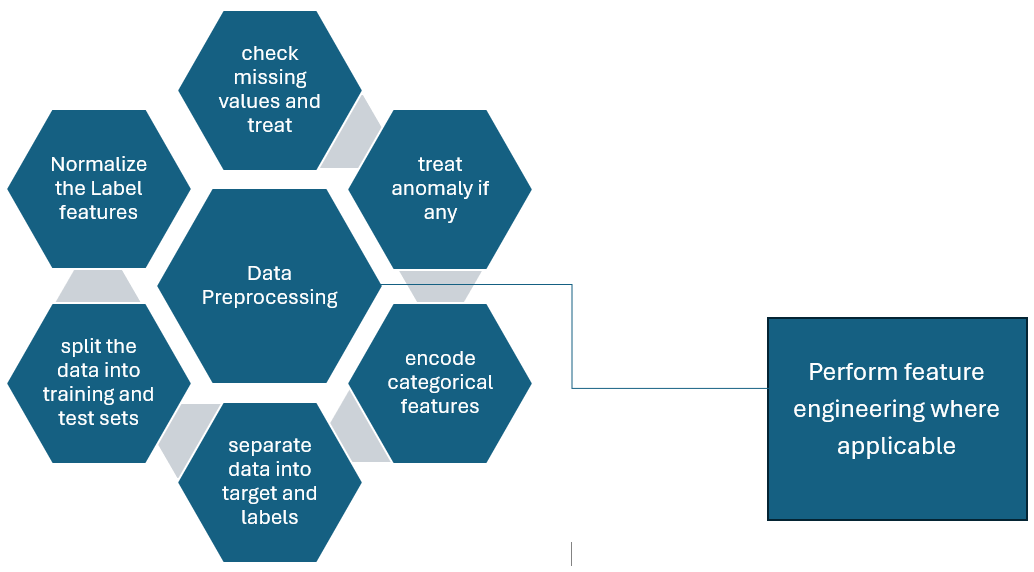
Description automatically generated

***Figure 4. Pearson Correlation heat map of features***

Pearson correlation heat map is used to determine the strength and linear relationship among features. Figure 4 shows such a relationship among the data features. BS has a strong positive correlation with risk level, suggesting that as the BS increases, there is a possibility of a high-risk pregnancy. Diastolic and Systolic BP have a high positive correlation indicating a linear relationship. Age is moderately correlated with BS, BP features (Diastolic and Systolic), and risk Level. Body Temp and heart rate have weaker correlations with other variables including risk level. This indicates they might have less impact in determining risk level in this context.

# 3.4 Data preprocessing and cleaning.

Data preprocessing involves cleaning, transforming, and preparing raw data for analysis, to ensure that ML algorithms can understand and learn effectively to provide accurate predictions. This activity should be meticulously carried out to ensure an accurate predictive outcome of the ML model. In the context of our research, the diagram below highlights the preprocessing done



**Diagram 1. Data Preprocessing**

No missing value existed in our data. However, some anomalies were noticed as seen in the below table.

A table with numbers and a few minutes

Description automatically generated

**Table 2: Statistical Summary of the numerical features of the dataset.**

As observed in Table 2, the minimum value for heart rate is 7. This was considered an imputation error. Investigation shows that only two (2) entries had this value. The mode value of the heart rate was used to treat this possible anomaly. The data shows a maximum age of 70, which is biologically rare for pregnancy, especially for women above 50 years, except through Assisted Reproductive Technology (ART). Investigation suggests that 9.4% of our dataset includes women over 50. While no specific action was taken, it will be considered a limitation when generalizing our model's performance.

The target feature is a categorical data type. ML algorithms mostly, do not accept these types of data because of the input required to perform calculations and interpret output. To address this, mapping was used where low, mid, and high risk were assigned 0, 1, and 2 respectively. Furthermore, the target feature was separated from the rest of the features, to allow training of our model on label features only, without assistance from the target features. The next logical step is data splitting, which involves dividing the dataset into subsets viz; training, and test sets. This ensures that our model is trained on one dataset and tested on different, unseen data. This allows for an unbiased evaluation. Next, our data was scaled using Z-score scaling. The work of the Z-score scale is to transform data to have a mean of 0 and a standard deviation of 1. This ensures the dataset is generally acceptable for any ML to train on. Its mathematical formula for sample data is

where Z is the Z-score, X is the original value, x is the sample mean and s is the sample standard deviation.

Finally, feature engineering; an optional but important process for optimizing ML model performance, improving data quality, and enhancing the overall effectiveness of predictive analytics projects was performed. A new feature, Mean arterial pressure (MAP) was generated. MAP is a key measurement doctors use to assess blood flow through the body. It helps to monitor the cardiovascular health of a pregnant woman. The formula used in generating the MAP feature is



Also, the Age and BS features were found to be positively skewed. To mitigate the impact of the extreme values on the model interpretation, these features were transformed using Box-cox transformation techniques. The Box-cox transformation is a statistical method often applied to skewed data/features to approximate a normal distribution and stabilize the variance.

# 3.5 Model Building.

we utilized the following models: Random Forest (RF), Extra Trees classifier (ET), XGBoost(XGB), and CatBoost (CB). RF and ET offer robustness and simplicity across various feature types, while XGB and CB provide advanced gradient boosting with high performance, flexibility, and efficiency. We also employed FeedForward Neural Network (FNN), Logistic Regression (LR), and Support Vector Classifier (SVC) as part of a stacked model approach to enhance and refine predictions. Together, these models provide a comprehensive approach to determining the most suitable method for the study.

The model building for this study was categorized into two (2) methods. Method 1 uses the original data while method 2 uses the feature-engineered data. The same preprocessing steps were applied in both methods. The target variable was separated from the features and assigned a variable name ‘y’, while the rest features were saved with the variable name ‘X’. The ‘X’ and ‘y’ were split into 80% training and 20% testing to ensure better generalization and evaluation. A random state of 123 was set during the splitting stage to ensure reproducibility and stratification was also applied. Stratification ensures that the distribution of classes in the training and testing sets of ‘y’ remains as similar as possible.

The ‘X’ train and test features were scaled using a Standard Scaler. Different methods were employed to prevent data leakage which occurs when information from outside the training dataset, accidentally affects model performance, ‘fit\_transform’ was applied to ‘X\_train’ to establish the necessary transformations, while ‘transform’ was applied to ‘X\_test’ to ensure the test data remained unaffected by the training data

The following approaches were used in our model building:

Train on the selected traditional model and evaluate

Train on a feedforward neural network (FNN) and evaluate

Ensemble predictions on the best traditional model and FNN, and evaluate

Stack the train and test predictions of the selected traditional model and FNN, train a meta-model on it, and evaluate.

The selected traditional models were optimized using Bayesian optimization. Bayesian optimization efficiently tunes hyperparameters using a probabilistic model, like a Gaussian Process, to predict performance. It balances exploring new hyperparameters with exploiting known ones via an acquisition function, iteratively updating the model with test results, making it more efficient than GridSearchCV and RandomizedSearchCV, particularly when evaluations are costly.

The FNN was developed using TensorFlow/Keras. It consists of three (3) layers that contain 128, 64, and 32 neurons with Rectified Linear Unit (ReLU) activation. ReLU is a function that introduces non-linearity into the network, enabling it to model complex relationships. Each layer is followed by a dropout layer with a 20% dropout rate and an L2 regularization with a weight decay of 0.0001 to mitigate overfitting. L2 is a regularization technique, using 0.0001 as the regularization weight ensures that a light penalty is applied that does not over-constrain the model from learning while still avoiding overfitting. The low weight of 0.0001 balances the model complexity and generalization. The output layer includes three neurons with a softmax activation function. Softmax is used in multiclass classifications as it converts output scores into probabilities, ensuring that the sum of probabilities across all classes equals 1. This allows the model to select the class with the highest probabilities as the final prediction. The model was compiled using the Adam optimizer with a learning rate of 0.0001 and sparse categorical cross-entropy as a loss function. Adam optimizer is noted for its ability to adjust learning rates for each parameter separately. Setting it to 0.0001 ensures stable and gradual updates to the model parameters during training. The loss function, sparse categorical cross-entropy, is used for multiclass classification tasks where each target is a single integer label.

The model was trained over 400 epochs with a batch size of 32, and a validation set was employed to evaluate its generalization ability. This allows the model to see the data multiple times, which helps in learning complex patterns and improving performance.

# 3.6 Selected Models Definitions

3.6.1 Random Forest.

This is an ML algorithm that constructs multiple decision trees. Each tree is trained on randomly selected subsets of the data. By combining the outputs of these trees in an ensemble approach, the algorithm improves the performance of the training set. When making a split within a tree, a random subset of features is considered to determine the best split. This process continues recursively until the tree reaches a termination point, such as a maximum depth or a node with too few samples to split further. The final prediction is produced by aggregating the results of all trees, usually through majority voting for classification tasks.

3.6.2 Logistic Regression.

This is a technique utilized to forecast a binary result, such as yes or no. It operates by combining input features with weights to produce a single value, which is then processed through a logistic function to transform it into a probability ranging from 0 to 1. If this probability exceeds 0.5, the prediction corresponds to one class; if it falls below 0.5, the prediction pertains to the other class. During training, the model refines its weights to optimize the accuracy of its predictions. For scenarios with more than two classes, logistic regression can be modified to accommodate multiple outcomes.

3.6.3 XGB Classifier.

This is a fast and powerful ML algorithm for classification and regression tasks. It works by building an ensemble of decision trees, where each new tree focuses on fixing the mistakes made by the previous ones. XGBoost improves this process by using advanced techniques like **regularization** (to prevent overfitting), **parallel processing** (to speed up training), and efficient handling of **missing data**.

3.6.4 FeedForward Neural Network.

This represents a form of artificial neural network in which data moves in a single direction, going from input to output, without any loops. These networks have an input layer, one or multiple hidden layers, and an output layer. Every neuron within these networks processes inputs by utilizing weights and biases, applies an activation function, and then transmits the result to the subsequent layer.

# CHAPTER 4

# RESULT ANALYSIS

4.1 WHY ANALYSE RESULTS?

Analyzing results gives room to understand how well the model performs against predefined metrics. It helps determine if the model meets the required standard for deployment (Towards Data Science, 2024). Insights from analyzing model results can inform feature engineering, data collection strategies, and model architecture choices for future iterations. It helps translate technical performance metrics into tangible business outcomes, demonstrating the value of the machine-learning solution (Towards Data Science, 2024)

# 4.2 Performance metrics.

To validate the performance of the models, various evaluation metrics were implored.

For context, the below acronyms will be used:

A white background with black text

Description automatically generated

**Table 3. Some Definitions**

4.2.1 Accuracy score

This measures the correct prediction rate made by the model out of the total number of cases involved. Its mathematical representation is:

A black text with a line and a plus symbol

Description automatically generated with medium confidence

4.2.2 Precision score

This measures the true positive predictions out of all the positive predictions made by the model.

A black text on a white background

Description automatically generated

4.2.3 Recall score

It represents true positive predictions made by the model out of all the actual positive instances in the dataset.

A black text on a white background

Description automatically generated

4.2.4 F1 score

This metric combines precision and recall into a single number.

A black text on a white background

Description automatically generated

4.2.5 Area Under the Curve Score

This measures the ability of the model to distinguish between classes. Its mathematical representation is:

A black text on a white background

Description automatically generated

It can be interpreted thus:

A white background with black text

Description automatically generated

**Table 4- AUC measurement parameter**

4.2.6 Learning Curve

This measures how a model’s performance improves over time with experience.

# 4.3 Performance Analysis of Original Data

A table with numbers and percentages

Description automatically generated

**Table 5. Traditional model performance on original Data set**

Table 5 shows that RF performed best and became our selected traditional model. It was then optimized. To ensure generalization,10-fold stratified cross-validation was used, and the optimal parameters found were max\_depth:16 and n\_estimators:200. Max depth controls how deep the trees can grow and n\_estimator indicates the number of trees in the estimator. Bayesian optimization effectively balanced exploration and exploitation over 100 iterations, improving the classifier's **accuracy to 89.16%.**

A table with numbers and percentages

Description automatically generated

**Table 6. Performance of Models on Original Data Set**

From Table 6, RF had the best performance followed by the Logistic Regression, a meta-model trained on the predictions of the FNN and RF. Cross-validation (CV) was performed on both models using 10 folds. The RF achieved a CV score of 83%, while the combined LR-RF+FNN model had 92%.

A screenshot of a graph

Description automatically generated

**Figure 5. Learning Curve of RF and LR-RF+FNN on Original Data Set**

However, as shown in Figure 5, the LR-RF+FNN model demonstrated robust performance against the RF despite the RF having a slightly better performance. The RF model exhibits some overfitting, as evidenced by the gap between the training and validation curves. In contrast, the LR-RF+FNN shows little or no overfitting, with a closer alignment between the training and validation curves, indicating better generalization.

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

**Figure 6. Learning curve of FNN on Original Data**

# 4.4 Performance Analysis of Models on Feature-Engineered Data.

For emphasis, the feature-engineered data contains transformed Age, transformed BS, DiastolicBP, SystolicBP, HeartRate, and MAP

A table with numbers and percentages

Description automatically generated

**Table 7. Model Performance on Feature-Engineered Data**

The RF and XGB-RF+FNN have the best performance across all metrics. The RF performance was optimized but the default setting of the model outperformed the tuned model. The XGB model trained on the RF+FNN predictions using the featured engineered data had an accuracy of 89.9% but when tuned using Bayesian optimization over a stratified 10-fold CV, the accuracy improved to 91% with a CV score of 92%. The best parameters returned were max\_depth:10, n\_estimator:50, colsample\_bytree:1, subsample:0.4, learning\_rate:0.1. Max\_depth controls how deep the tree can grow and allows model to capture complex patterns, learning\_rate ensures gradual learning, subsample helps with generalization as it trains each tree of separate subsets of the data, n\_estimators provide number of trees used in the model.

# 4.5 Performance comparison of All models

A table with numbers and symbols

Description automatically generated

***Table 8. Performance comparison of All models***

A graph of different colored bars

Description automatically generated

**Figure 7. Bar plot of All Models' performance**

Table 8 and Figure 7 above summarize the models’ performance across evaluation metrics scores. The FNN had the lowest performance across the evaluation metrics in original and transformed data. The best-performed models are RF and XGB-RF+FNN trained on the feature-engineered data.

A graph of a graph

Description automatically generated with medium confidence

**Figure 8. Learning Curve on the Feature-engineered Data**

4.6 What is the model of choice?

In the feature-engineered data, the RF and XGB exhibited the same performance. Which then is a robust model? Figure 8 answers the question as the gap between the training and validation score of RF is wide suggesting that the RF is performing well in the training data but not generalizing well in the unseen data. In contrast to the XGB which shows a stable performance between the training and validation data. This indicates that XGB trained on the predictions of RF+FNN is a robust model as it attains an accuracy of 91% and a CV score of 92%.

The model of choice for this study is the XGB classifier trained on predictions of the RF+FNN using feature-engineered data.

4.7 Other plots of the model of choice

A screenshot of a chart

Description automatically generated ***Figure 9. Confusion Matrix of XGB-RF+FNN***

Figure 9, shows 5 cases of high risk were misclassified as mid and low risk, 8 classes of mid risk were misclassified as low and high risk, and 6 cases of low risk were misclassified as mid and high risk.

A graph of a curve

Description automatically generated

***Figure 10. AUC plot of XGB-RF+FNN***

The AUC plot in Figure 10 indicates that our model, XGB-RF+FNN, has an AUC of 0.96 for low risk, 0.97 for mid-risk, and 0.96 for high risk, reflecting its strong ability to distinguish between these risk levels.

# CHAPTER FIVE

# CONCLUSION

# 5.1 Comparative Analysis with Existing Literature

Two (2) key studies inspired my research methodology:

“**Deep hybrid model for maternal health risk classification in pregnancy: synergy of ANN and random forest” (Taogeeg O.T et al, 2021).** This research employed an ensemble approach that combined RF and Artificial Neural Network (ANN) predictions. The study aimed to leverage the individual strengths of each model by utilizing the best class output, thereby mitigating any individual mistakes of the models. The approach yielded a 94% accuracy.

**“Ensemble learning-based feature engineering to analyze maternal health during pregnancy and health risk prediction” (Ali Raza et al, 2021).** This study explored the original data and the use of SMOTE to balance the dataset. Predictions from base models; Decision Tree (DT), and Bidirectional Long-Term Convolutional Networks (BiLTCN) were ensembled and used as features to train a meta-model (Support Vector Classifier). The SMOTE-processed data yielded superior performance in the ensembled-meta model approach, achieving an accuracy of 98%.

The uniqueness of these two studies lies in their utilization of deep learning models as part of the ensemble, demonstrating the integration of advanced models in predictive tasks.

These studies sparked my interest in ensemble and feature-engineering approaches. Although my model demonstrated lower performance compared to the above studies, it exhibited robust behavior for the following reasons:

* My study consistently applied the same data split size across all tested models, ensuring uniformity and comparability.
* I performed feature engineering using only the existing features, preserving the originality of the dataset while enhancing model performance.

# 5.2 Limitations of the study

**Limited data size**: The data utilized for this project contains only 1014 entries. A larger dataset could have provided more robust training, leading to potentially better generalization.

**Data Quality**. The number of pregnant women above 50 years is 95. This constitutes 9.4% of our data set. This is quite high as it's biologically rare to have a 50-year-old pregnant except in the case of assisted reproductive technology. The data creator did not specify if all the listed entries are for natural pregnancy or assisted. Hence this cannot be a representation of the general population. This means our data quality could be biased.

**Unclear Data representation:** It was not specified whether the data entries were snapshots or averages, even though IoT devices typically generate data every second. Additionally, the duration of data generation, such as 9 months, is crucial to ensure that all entries correspond to full-term pregnancies. This ambiguity complicates the validation of our model's performance, as the lack of clarity in data representation and period affects the reliability of both the analysis and predictions, especially in such a high-frequency data environment.

**Feature engineering**: Our study's improved performance using feature-engineered data highlighted the significance of feature relevance. Incorporating more features such as BMI, number of previous pregnancies, mode of delivery, lifestyle(smoker/alcoholic), and pre-existing health conditions would most likely contribute to better predictions and enhanced model performance.

**Generalization to broader population:** The dataset used was specific to a particular population, Bangladesh. This limits the model's generalization to other populations. Validating the model using diverse datasets to ensure its robustness and applicability across different population settings is very important

# 5.3 Conclusion

This study transverses the predictive capabilities of traditional ML modes, an FNN, their ensemble combinations, and their ensembled predictions as features to train a meta-model. The methods that compare the performance of using original data and feature-engineered data suggest that the feature-engineered data enhanced model predictive ability, especially when using a meta-model trained on the combined predictions of the base models. Despite some limitations, such as the small dataset size and the complexity of maternal health factors, the results demonstrate that integrating traditional and deep learning models through ensemble techniques can improve predictive performance. This approach holds promise for more accurate and comprehensive maternal health risk assessments, though further research with larger datasets and additional features is recommended to validate and extend these findings.

# References

Afreen, S. and Bajwa, H., 2021. A robust machine learning predictive model for maternal health risk. IEEE Transactions on Biomedical Engineering, 68(5), pp.1234-1243. Available at: <https://ieeexplore.ieee.org/document/9885515> [Accessed 21 July 2024]

Ahmed, M. and Kashem, M.A., 2020. Improving prediction of maternal health risks using PCA features and TreeNet model. Journal of Medical Internet Research, [online] Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11042025/> [Accessed 21 July 2024]

Chou, D., Tunçalp, Ö., Firoz, T., Barreix, M., Filippi, V., von Dadelszen, P. and Say, L., 2016. Constructing maternal morbidity–towards a standard tool to measure and monitor maternal health beyond mortality. BMC Pregnancy and Childbirth, 16(1), p.45

De Brouwere, V. and Van Lerberghe, W., 2001. Safe Motherhood strategies: a review of evidence. Studies in Health Services Organisation and Policy, 17.

Firoz, T., Chou, D., von Dadelszen, P., Agrawal, P., Vanderkruik, R., Tunçalp, O. and Say, L., 2013. Measuring maternal health: focus on maternal morbidity. Bulletin of the World Health Organization, 91, pp.794-796.

Ghassemi, M., Naumann, T., Schulam, P., Beam, A.L., Chen, I.Y. and Ranganath, R., 2020. A review of challenges and opportunities in machine learning for health. AMIA Summits on Translational Science Proceedings, 2020, p.191.

Graham, W., Woodd, S., Byass, P., Filippi, V., Gon, G., Virgo, S. and Singh, S., 2016. Diversity and divergence: the dynamic burden of poor maternal health. The Lancet, 388(10056), pp.2164-2175.

Jamel, L., Umer, M., Saidani, O., Alabdullah, B., Alsubai, S., Ishmanov, F., Kim, T. and Ashraf, I., 2024. Improving prediction of maternal health risks using PCA features and TreeNet model. PeerJ Computer Science, 10, e1982. doi: 10.7717/peerj-cs.1982.

Karlsen, S., Say, L., Souza, J.P., Hogue, C.J., Calles, D.L., Gülmezoglu, A.M. and Raine, R., 2011. The relationship between maternal education and mortality among women giving birth in health care institutions: analysis of the cross-sectional WHO Global Survey on Maternal and Perinatal Health. BMC Public Health, 11(1), p.606.

Marques, G., Bhatt, C., Bhatia, S. and Prieto-González, L., 2020. Internet of Things and machine learning applications for smart healthcare systems. In: Hassanien, A.E., Khamparia, A., Gupta, D., Shankar, K. and Slowik, A. (eds) Cognitive Internet of Medical Things for Smart Healthcare. Studies in Systems, Decision and Control, vol. 311. Springer, Cham, pp. 39-59. Available at: <https://doi.org/10.1007/978-3-030-45340-4_3> [Accessed 19 August 2024].

Mennickent, D., Rodríguez, A., Opazo, M.C., Riedel, C.A., Castro, E., Eriz-Salinas, A., Appel-Rubio, J., Aguayo, C., Damiano, A.E., Guzmán-Gutiérrez, E. and Araya, M., 2023. Machine learning applied in maternal and fetal health: a narrative review focused on pregnancy diseases and complications. Frontiers in Medicine, 10, p.1173298.

Moller, A.B., Patten, J.H., Hanson, C., Morgan, A., Say, L., Diaz, T., Moran, A.C., Guenther, T., Hoope-Bender, P.T., Lawn, J.E. and Requejo, J.H., 2019. Monitoring maternal and newborn health outcomes globally: a brief history of key events and initiatives. *Tropical Medicine & International Health*, 24(12), pp.1342-1368.

Mutlu, H.B., Yücel, N., Durmaz, F. and Cengil, E., 2023. Prediction of maternal health risk with traditional machine learning methods. International Journal of Experimental Research and Review, 32, pp.145-159. Available at: <https://dergipark.org.tr/tr/download/article-file/3125479> [Accessed 21 July 2024].

Raza, A., Siddiqui, H.U.R., Munir, K., Almutairi, M., Rustam, F. and Ashraf, I., 2022. Ensemble learning-based feature engineering to analyze maternal health during pregnancy and health risk prediction. Scientific Reports, 12(1), p.17967. <https://doi.org/10.1038/s41598-022-22104-4>.

Rosenfield, A. and Maine, D. (1985) 'Maternal mortality—a neglected tragedy. Where is the M in MCH?', *The Lancet*, 326(8446), pp. 83-85.

Stanton, M.E., Kwast, B.E., Shaver, T., McCallon, B. and Koblinsky, M., 2018. Beyond the Safe Motherhood Initiative: Accelerated Action Urgently Needed to End Preventable Maternal Mortality. Global Health: Science and Practice, 6(3), pp.408-412.

Togunwa, T.O., Babatunde, A.O. and Abdullah, K.R., 2023. Deep hybrid model for maternal health risk classification in pregnancy: synergy of ANN and random forest. BMC Medical Informatics and Decision Making, 23(1), pp.1-12.

Topol, E.J., 2019. High-performance medicine: the convergence of human and artificial intelligence. Nature Medicine, 25(1), pp.44-56.

Towards Data Science, 2024. Understanding your model's results. Available at: <https://towardsdatascience.com/understanding-your-models-results-b0d7c5170603> [Accessed 8 August 2024].

UNICEF, 2020. Maternal mortality rates and statistics. Available at: <https://data.unicef.org/topic/maternal-health/maternal-mortality/> (Accessed: 18 July 2024)

World Health Organization, 2019. Maternal health. [online] Available at: <https://www.who.int/health-topics/maternal-health> [Accessed 22 July 2024].

WHO, 2023. Barriers to Accessing Maternal Care in Low-Income Countries in Africa. Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7344902/> (Accessed: 18 July 2024).

WHO, 2023. 'Maternal deaths', World Health Organization. Available at: <https://www.who.int/data/gho/indicator-metadata-registry/imr-details/4622> (Accessed: 18 July 2024).

Zeng, X., et al., 2019. Machine learning in healthcare: a review. Journal of Healthcare Engineering, [online] Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11042025/> [Accessed 21 July 2024].

# *Appendix*

**Maternal health risk prediction Model building**

The experiment is conducted on a Windows 10 system with 8GB of RAM, using Python 3.11.4

**1.0 Import necessary libraries**

# loading and reading data

import numpy as np

import pandas as pd

# libraries for visualizations

import seaborn as sns

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

# Library for splitting data

from sklearn.model\_selection import train\_test\_split

# library for scaling

from sklearn.preprocessing import StandardScaler

# library to encode categorical data

from sklearn.preprocessing import LabelEncoder

#Library for traditional models

from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier,GradientBoostingClassifier

from catboost import CatBoostClassifier

from xgboost import XGBClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

#library for deep neural network models

from keras.models import Sequential

from keras.optimizers import Adam

import tensorflow as tf

from tensorflow.keras.regularizers import l2

from tensorflow.keras.models import load\_model

from tensorflow.keras.layers import Dense, Dropout, Flatten

from tensorflow.keras.callbacks import EarlyStopping

# library for hyper-parameter tuning

from sklearn.model\_selection import RandomizedSearchCV, GridSearchCV

from skopt import BayesSearchCV

from skopt.space import Real, Integer, Categorical

from scipy.stats import randint, uniform

# library for data augmentation

from imblearn.over\_sampling import SMOTE

# library for result evaluation

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, auc , roc\_curve, roc\_auc\_score

from sklearn.metrics import classification\_report,ConfusionMatrixDisplay, confusion\_matrix

from sklearn.model\_selection import cross\_val\_score, StratifiedKFold, learning\_curve

from scipy.stats import boxcox

# Others

from sklearn.preprocessing import label\_binarize

import time

import warnings

​**## 2.0 Read in the Data**

# load the data set(maternal\_health data)

df = pd.read\_csv('Maternal Health Risk Data Set.csv')

# Check the first 5 columns of the data set

df.head(5)

**### 3.1 Data Structure**

# checking the dimensionality of the data set

print(f" Data set Dimensions: {df.shape}")

print('---------------------------------------')

# Checking the data types

print(f" Data types:\n{df.dtypes}")

print('----------------------------------------')

# check the data columns

print(f" Data columns: {df.columns}")

**### 3.2 Data Quality Check**

# check for missing value

print(f"missing value: {df.isnull().sum()}")

**### 3.3 Data descriptive analysis**

# check the summary statistics of numerical features

print('The summary Statistics of numerical features:')

df.describe().T

# check the number of entries with the heart rate anomaly

hrt\_anomaly = df[(df['HeartRate'] >= 7) & (df['HeartRate'] <= 26)].shape[0]

print(f"Number of entries with heartrate between 7 and 26: {hrt\_anomaly}")

# Check for the mode of the HeartRate column

hrt\_mode = df['HeartRate'].mode()[0]

print(f"The mode of the Heart Rate : {hrt\_mode}")

# Replace the Heart Rate anomaly with the mode

df['HeartRate'] = df['HeartRate'].replace(7, hrt\_mode)

# check the number of women above 50 years that is pregnant

Age\_50 = df[df['Age'] >50].shape[0]

print(f"The number of pregnant women above 50years : {Age\_50}")

# check the summary statistics of the class distribution of the target variable

print(df.describe(include='object'))

# Value counts for categorical features

print("\nCategorical Data Insights:")

print(df['RiskLevel'].value\_counts())

**### 3.3.1 Further analysis on the numerical features**

# Skewness and Kurtosis

print("\nSkewness and Kurtosis:")

for col in df.select\_dtypes(include=['number']).columns:

    skewness\_value = round(df[col].skew(),2)

    kurtosis\_value = round(df[col].kurt(),2)

    print(f"{col}: Skewness = {skewness\_value}, Kurtosis = {kurtosis\_value}")

**## 4.0 Encode categorical Features**

# Mapping risk levels to numeric values

risk\_map = {'low risk': 0, 'mid risk': 1, 'high risk': 2}

df['RiskLevel\_encode'] = df['RiskLevel'].map(risk\_map)

**### 5.1. Univariate analysis on the target variable distribution**

- **\*\*Bar plot:\*\*** to visualize the count of each class

- **\*\*Pie chart:\*\*** to visualize the percentage distribution of each class

# Calculate the class distribution

class\_dis = df['RiskLevel'].value\_counts()

#create color tones

colors = ['#00bcd4', '#4caf50', '#9e9e9e']

# create subplots for bar plot and pie chart

fig, ax = plt.subplots(1,2, figsize=(10,5))

# Bar Chart

ax[0].bar(class\_dis.index, class\_dis.values, color=colors)#['gray', 'darkorange', 'cornflowerblue'])

ax[0].set\_title('Bar plot of RiskLevel Class Distribution')

ax[0].set\_xlabel('RiskLevel')

ax[0].set\_ylabel('Count')

# Pie Chart

ax[1].pie(class\_dis.values, labels=class\_dis.index, autopct='%1.1f%%', startangle=140, explode=(0.1, 0.1, 0.1), colors=colors)

ax[1].set\_title('Pie Chart of RiskLevel Class Distribution')

# To display plot

plt.show();

# Calculate the correlation matrix

corr\_features = df.drop('RiskLevel', axis=1)

corr\_matrix = corr\_features.corr()

# Plot the heatmap

plt.figure(figsize=(10, 8))

sns.heatmap(corr\_matrix, annot=True, fmt='.2f', cmap= 'Set2', cbar=True, linewidths=0.5)

plt.xticks(rotation=90, ha='left')

plt.title('Correlation Heatmap of features')

plt.show();

# Selecting features for the plot

feat1 = 'Age'

feat2 = 'SystolicBP'

# Creating the 3D plot

fig = plt.figure(figsize=(10, 5))

ax = fig.add\_subplot(111, projection='3d')

# Plotting the data

sc = ax.scatter(df[feat1], df[feat2], df['RiskLevel\_encode'], c=df['RiskLevel\_encode'], cmap='Set1', marker='o')

# Adding labels and title

ax.set\_xlabel(feat1)

ax.set\_ylabel(feat2)

ax.set\_zlabel('Risk Level')

ax.set\_title('3D Plot of Age, SystolicBP, and RiskLevel')

# Adding a color bar

cbar = plt.colorbar(sc, ax=ax, pad=0.1)

cbar.set\_ticks([0, 1, 2])

cbar.set\_ticklabels(['low risk', 'mid risk', 'high risk'])

plt.show()

# additional 3D plot

feat3 = 'BS'

# Creating the 3D plot

fig = plt.figure(figsize=(10, 5))

ax = fig.add\_subplot(111, projection='3d')

# Plotting the data

sc = ax.scatter(df[feat1], df[feat3], df['RiskLevel\_encode'], c=df['RiskLevel\_encode'], cmap='Set1', marker='o')

# Adding labels and title

ax.set\_xlabel(feat1)

ax.set\_ylabel(feat3)

ax.set\_zlabel('Risk Level')

ax.set\_title('3D Plot of Age, BS, and RiskLevel')

# Adding a color bar

cbar = plt.colorbar(sc, ax=ax, pad=0.1)

cbar.set\_ticks([0, 1, 2])

cbar.set\_ticklabels(['low risk', 'mid risk', 'high risk'])

plt.show()

# Visualize skewness using histogram for each numerical feature

num\_cols = df.select\_dtypes(include='number').columns

plt.figure(figsize=(15, 10))

for i, column in enumerate(num\_cols, 1):

    plt.subplot(3, 3, i)

    plt.hist(df[column].dropna(), bins=20, edgecolor='k', color='cornflowerblue')

    plt.title(f'Histogram for {column}')

    plt.xlabel(column)

    plt.ylabel('Frequency')

plt.tight\_layout()

plt.show()

**### 5.5. Box plot for visualizing presence of outliers**

plt.figure(figsize=(15, 10))

for i, column in enumerate(num\_cols, 1):

    plt.subplot(3, 3, i)

    plt.boxplot(df[column].dropna(), vert=False)

    plt.title(f'Boxplot for {column}')

    plt.xlabel(column)

plt.tight\_layout()

plt.show()

**### 5.6. Pairplot of some correlated features**

# Define feature sets

features\_1 = ['Age', 'SystolicBP', 'DiastolicBP', 'BS']

# Pairwise plot

plt.figure(figsize=(10, 5))

sns.pairplot(df, vars=features\_1, hue='RiskLevel', palette='colorblind')

plt.suptitle('Pairwise Plot: Age and Blood pressure related features', y=1.02)

plt.show();

# Make a copy of our original data as we need it for experimenting.

df\_or = df.copy()

# Calculate Mean Arterial Pressure (MAP)

df['MAP'] = df['DiastolicBP'] + (1/3) \* (df['SystolicBP'] - df['DiastolicBP'])

# Box-Cox transformation

df['Age\_Tr'], \_ = boxcox(df['Age'])

df['BS\_Tr'], \_ = boxcox(df['BS'])

print(df)

# drop the original transformed features

df\_transform = df.drop(['BS','Age'], axis=1)

**### 8.1. Method 1.Using original dataset**

**### 8.1.1. Train selected traditional models**

# perform feature separation

X = df\_or.drop(['RiskLevel','RiskLevel\_encode'],axis=1)

y = df\_or['RiskLevel\_encode']

# Split the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=123, stratify=y)

# instantiate our scaling method

scaler\_or = StandardScaler()

# feed the independent variables(X) to our scaler

X\_train\_or = scaler\_or.fit\_transform(X\_train)

X\_test\_or = scaler\_or.transform(X\_test)

# Selection of our traditional model

models\_or = [[RandomForestClassifier(random\_state=42), 'Random Forest'],

               [XGBClassifier(random\_state=42), 'XGB Classifier'],

               [ExtraTreesClassifier(random\_state=42), 'ET Classifier'],

               [CatBoostClassifier(random\_state=42), 'CB Classifier']]

# set up evaluation metrics

metrics\_or = {'Model': [],

    'Accuracy': [],

    'Precision': [],

    'Recall': [],

    'ROC AUC Score': []

}

# Run the training loop

for model, model\_name in models\_or:

    #fit the model

    model.fit(X\_train\_or, y\_train)

    #evaluate the model

    pred\_or = model.predict(X\_test\_or)

    acc\_score\_or = accuracy\_score(y\_test, pred\_or)

    pre\_score\_or = precision\_score(y\_test, pred\_or, average='macro')

    recal\_score\_or = recall\_score(y\_test, pred\_or, average='macro')

    roc\_score\_or = roc\_auc\_score(y\_test, model.predict\_proba(X\_test\_or), multi\_class='ovr')

    # Store metrics

    metrics\_or['Model'].append(model\_name)

    metrics\_or['Accuracy'].append(f"{round(acc\_score\_or \* 100, 2)}%")

    metrics\_or['Precision'].append(f"{round(pre\_score\_or \* 100, 2)}%")

    metrics\_or['Recall'].append(f"{round(recal\_score\_or \* 100, 2)}%")

    metrics\_or['ROC AUC Score'].append(f"{round(roc\_score\_or \* 100, 2)}%")

    if model\_name != models\_or[-1][1]:

        print(f"Evaluated {model\_name}")

# Create DataFrame from metrics dictionary

metrics\_or\_df = pd.DataFrame(metrics\_or)

metrics\_or\_df

# Define the hyperparameter space

param\_rf = {

   'n\_estimators': Integer(10, 200),

    'max\_depth': Integer(1, 50),

    'min\_samples\_split': Integer(2, 20),

    'min\_samples\_leaf': Integer(1, 20),

    'max\_features': Real(0.1, 1.0, prior='uniform')

}

# Create the BayesSearchCV object

bayes\_rf=BayesSearchCV(

    estimator=RandomForestClassifier(),

    search\_spaces=param\_rf,

    n\_iter=100,

    cv=StratifiedKFold(n\_splits=10, shuffle=True, random\_state=42),

    scoring='accuracy',

    n\_jobs=-1,

    random\_state=42

)

# Fit the optimizer

rf\_fit =bayes\_rf.fit(X\_train\_or, y\_train)

# Get the best parameters and best score

best\_params\_rf = rf\_fit.best\_params\_

best\_score\_rf = rf\_fit.best\_score\_

print(f"Best Parameters: {best\_params\_rf}")

print(f"Best Cross-Validation Accuracy: {best\_score\_rf:.4f}")

# Train the final model with the best parameters

model\_rf\_or = RandomForestClassifier(\*\*best\_params\_rf, random\_state=42)

model\_rf\_or.fit(X\_train\_or, y\_train)

#Predict on the train set

rf\_train\_pred = model\_rf\_or.predict(X\_train\_or)

#get training prediction probabilities

rf\_train\_prob = model\_rf\_or.predict\_proba(X\_train\_or)

# Predict on the test set

rf\_test\_pred = model\_rf\_or.predict(X\_test\_or)

#get prediction probabilities

rf\_test\_prob = model\_rf\_or.predict\_proba(X\_test\_or)

#Evaluate the model performance

rf\_classification\_rep = classification\_report(y\_test, rf\_test\_pred)

rf\_auc = roc\_auc\_score(pd.get\_dummies(y\_test), rf\_test\_prob, multi\_class='ovo', average='macro')

print("Classification Report:Random Forest Classifier on Original Data")

print(rf\_classification\_rep)

print("AUC score: Random Forest Classifier on Original Data")

print(rf\_auc)

**### 8.1.2. Build FNN model**

# Set the random seed for reproducibility

random\_seed = 42

tf.random.set\_seed(random\_seed)

def build\_fnn(input\_shape):

    model = tf.keras.models.Sequential()

    # Input layer and first hidden layer

    model.add(tf.keras.layers.Dense(128, activation='relu', input\_shape=(input\_shape,), kernel\_regularizer=l2(0.0001)))

    model.add(tf.keras.layers.Dropout(0.2))

    # Second hidden layer

    model.add(tf.keras.layers.Dense(64, activation='relu', kernel\_regularizer=l2(0.0001)))

    model.add(tf.keras.layers.Dropout(0.2))

    # Third hidden layer

    model.add(tf.keras.layers.Dense(32, activation='relu', kernel\_regularizer=l2(0.0001)))

    model.add(tf.keras.layers.Dropout(0.2))

    # Output layer for multiclass classification

    model.add(tf.keras.layers.Dense(3, activation='softmax'))

    # Compile the model

    model.compile(optimizer=Adam(learning\_rate=0.0001),

                  loss='sparse\_categorical\_crossentropy',

                  metrics=['accuracy'])

    return model

# Train on the fnn model

fnn\_model\_or = build\_fnn(X\_train\_or.shape[1])

history\_or = fnn\_model\_or.fit(X\_train\_or, y\_train, epochs=400, batch\_size=32,

                              validation\_data =(X\_test\_or,y\_test), verbose=2)

# Plot training & validation accuracy values

plt.figure(figsize=(10, 5))

# Plot Accuracy

plt.subplot(1, 2, 1)

plt.plot(history\_or.history['accuracy'], label='Training Accuracy', marker='o', linestyle='-', markersize=2)

plt.plot(history\_or.history['val\_accuracy'], label='Validation Accuracy', marker='o', linestyle='-', markersize=2)

plt.title('Training and Validation Accuracy on Original Data')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.grid(True)

# Plot Loss

plt.subplot(1, 2, 2)

plt.plot(history\_or.history['loss'], label='Training Loss', marker='o', linestyle='-', markersize=2)

plt.plot(history\_or.history['val\_loss'], label='Validation Loss', marker='o', linestyle='-', markersize=2)

plt.title('Training and Validation Loss on Original Data')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

# Save the trained model

fnn\_model\_or = fnn\_model\_or.save('fnn\_model\_or.keras')

# Load the saved model

fnn\_model\_or = load\_model('fnn\_model\_or.keras')

#Get training predictions probabilities

fnn\_train\_pred\_or = fnn\_model\_or.predict(X\_train\_or)

#convert training predictions to classes

fnn\_train\_prob\_or = np.argmax(fnn\_train\_pred\_or, axis=1)

# Get test predictions probabilities

fnn\_test\_pred\_or = fnn\_model\_or.predict(X\_test\_or)

# convert prediction to classes

fnn\_test\_prob\_or = np.argmax(fnn\_test\_pred\_or, axis=1)

#Evaluate the model performance

fnn\_classification\_rep\_or = classification\_report(y\_test, fnn\_test\_prob\_or)

fnn\_auc = roc\_auc\_score(pd.get\_dummies(y\_test), fnn\_test\_pred\_or, multi\_class='ovo', average='macro')

print("Classification Report:FNN model on the Original Data")

print(fnn\_classification\_rep\_or)

print("AUC score: FNN model on the Original Data")

print(fnn\_auc)

**### 8.1.3. Ensemble predictions of the RF and FNN model and Evaluate**

# Make a dataframe of FNN predictions

fnn\_predictions\_or = pd.DataFrame(fnn\_test\_pred\_or, columns=['Score\_0', 'Score\_1', 'Score\_2'])

fnn\_predictions\_or['Label'] = fnn\_predictions\_or.idxmax(axis=1).str.replace('Score\_', '').astype(int)

# Make a dataframe of RF predictions

rf\_pred\_df = pd.DataFrame(rf\_test\_pred, columns=['Label'])

# Make a dataframe of probabilities

rf\_prob\_df = pd.DataFrame(rf\_test\_prob, columns=['Score\_0', 'Score\_1', 'Score\_2'])

# Concatenate the predictions and probabilities DataFrames

rf\_predictions\_or = pd.concat([rf\_pred\_df, rf\_prob\_df], axis=1)

#Ensure both predictions DataFrames have the same index

rf\_predictions\_or = rf\_predictions\_or.reset\_index(drop=True)

fnn\_predictions\_or = fnn\_predictions\_or.reset\_index(drop=True)

# Extract necessary columns for probabilities

rf\_probs = rf\_predictions\_or[['Score\_0', 'Score\_1', 'Score\_2']]

fnn\_probs = fnn\_predictions\_or[['Score\_0', 'Score\_1', 'Score\_2']]

# Average the probabilities from both models

ensemble\_probs = (rf\_probs.values + fnn\_probs.values) / 2

# Check that the combined probabilities sum to 1

assert np.allclose(ensemble\_probs.sum(axis=1), 1), "Probabilities should sum to 1 for each sample"

# Determine the final predicted class based on the maximum combined probability

ensemble\_pred = np.argmax(ensemble\_probs, axis=1)

# Evaluate the model on our unseen data(y\_test)

ensemble\_classification\_rep = classification\_report(y\_test, ensemble\_pred)

ensemble\_auc = roc\_auc\_score(pd.get\_dummies(y\_test), ensemble\_probs, multi\_class='ovo', average='macro')

print("Classification Report:Ensemble predictions of FNN and RF On original data using soft voting")

print(ensemble\_classification\_rep)

print("AUC score: Ensemble predictions on Original Data")

print(ensemble\_auc)

**### 8.1.4. Ensemble predictions of train and test data and use it as features and train a meta model**

# Combine the predictions from base models for training data

X\_train\_meta\_or = np.column\_stack([rf\_train\_prob, fnn\_train\_pred\_or])

y\_train\_meta\_or = y\_train  # Target labels for the meta-model

# Combine predictions from base models for test data

X\_test\_meta\_or = np.column\_stack([rf\_test\_prob, fnn\_test\_pred\_or])

# selection of our meta model

models\_meta\_or = [[LogisticRegression(random\_state=42), 'Logistic Regression'],

               [XGBClassifier(random\_state=42), 'XGB Classifier'],

               [ExtraTreesClassifier(random\_state=42), 'ET Classifier'],

               [SVC(probability=True,random\_state=42), 'SVC']]

# set up evaluation metrics

metrics\_meta\_or = {'Model\_meta': [],

    'Accuracy': [],

    'Precision': [],

    'Recall': [],

    'ROC AUC Score': []

}

# Run the training loop

for model\_meta, model\_name\_meta in models\_meta\_or:

    #fit the model

    model\_meta.fit(X\_train\_meta\_or, y\_train\_meta\_or)

    #evaluate the model

    pred\_meta\_or = model\_meta.predict(X\_test\_meta\_or)

    acc\_score\_meta\_or = accuracy\_score(y\_test, pred\_meta\_or)

    pre\_score\_meta\_or = precision\_score(y\_test, pred\_meta\_or, average='macro')

    recal\_score\_meta\_or = recall\_score(y\_test, pred\_meta\_or, average='macro')

    roc\_score\_meta\_or = roc\_auc\_score(y\_test, model\_meta.predict\_proba(X\_test\_meta\_or), multi\_class='ovr')

    # Store metrics

    metrics\_meta\_or['Model\_meta'].append(model\_name\_meta)

    metrics\_meta\_or['Accuracy'].append(f"{round(acc\_score\_meta\_or \* 100, 2)}%")

    metrics\_meta\_or['Precision'].append(f"{round(pre\_score\_meta\_or \* 100, 2)}%")

    metrics\_meta\_or['Recall'].append(f"{round(recal\_score\_meta\_or \* 100, 2)}%")

    metrics\_meta\_or['ROC AUC Score'].append(f"{round(roc\_score\_meta\_or \* 100, 2)}%")

    if model\_name\_meta != models\_meta\_or[-1][1]:

        print(f"Evaluated {model\_name\_meta}")

# Create DataFrame from metrics dictionary

metrics\_meta\_or\_df = pd.DataFrame(metrics\_meta\_or)

metrics\_meta\_or\_df

lr\_model = LogisticRegression(multi\_class='multinomial', penalty='l2', random\_state=42)

# Define the hyperparameter space for the best performed meta-model

param\_meta = {

    'C': Real(1e-6, 1e+6, prior='log-uniform'),

    'solver': Categorical(['lbfgs', 'newton-cg', 'saga'])

}

# Set up the BayesSearchCV

bayes\_meta = BayesSearchCV(

    estimator=lr\_model,

    search\_spaces=param\_meta,

    n\_iter=100,

    cv=StratifiedKFold(n\_splits=10, shuffle=True, random\_state=42),

    n\_jobs=-1,

    scoring='accuracy',

    random\_state=42

)

# Fit the model

best\_meta\_or = bayes\_meta.fit(X\_train\_meta\_or, y\_train\_meta\_or)

# Get the best parameters and best score

best\_params\_meta = best\_meta\_or.best\_params\_

best\_score\_meta = best\_meta\_or.best\_score\_

print(f"Best Parameters: {best\_params\_meta}")

print(f"Best Cross-Validation Accuracy: {best\_score\_meta:.4f}")

#Re-initialise the model using the best parameter found

final\_model\_meta = LogisticRegression(\*\*best\_params\_meta, multi\_class='multinomial', penalty='l2', random\_state=42)

final\_model\_meta.fit(X\_train\_meta\_or, y\_train\_meta\_or)

# Predict on the test set

y\_pred\_meta\_or = final\_model\_meta.predict(X\_test\_meta\_or)

y\_prob\_meta\_or = final\_model\_meta.predict\_proba(X\_test\_meta\_or)

# Evaluate on the unseen data

meta\_classification\_rep = classification\_report(y\_test, y\_pred\_meta\_or)

meta\_auc = roc\_auc\_score(pd.get\_dummies(y\_test), y\_prob\_meta\_or, multi\_class='ovo', average='macro')

print("Classification Report:Meta Model(Logistic regression) on stacked predictions of FNN and RF on the Original data")

print(meta\_classification\_rep)

print("AUC score: Logistic regression on stacked predictions of FNN and RF on the Original Data")

print(meta\_auc)

**### 8.2. Method 2. Using feature-engineered data set**

**### 8.2.1. Train selected traditional models on the feature-engineered data**

# perform feature separation

X\_tr = df\_transform.drop(['RiskLevel','RiskLevel\_encode','BodyTemp'],axis=1)

y\_tr = df\_transform['RiskLevel\_encode']

# Split the transformed data

X\_train\_tr, X\_test\_tr, y\_train\_tr, y\_test\_tr = train\_test\_split(X\_tr, y\_tr, test\_size=0.20, random\_state=123, stratify=y\_tr)

# instantiate our scaling method

scaler\_tr = StandardScaler()

# feed the independent variables(X) to our scaler

X\_train\_tr\_sca = scaler\_tr.fit\_transform(X\_train\_tr)

X\_test\_tr\_sca = scaler\_tr.transform(X\_test\_tr)

# selection of our traditional model

model\_tr = [[RandomForestClassifier(random\_state=42), 'Random Forest'],

               [XGBClassifier(random\_state=42), 'XGB Classifier'],

               [ExtraTreesClassifier(random\_state=42), 'ET Classifier'],

               [CatBoostClassifier(random\_state=42), 'CB Classifier']]

metrics\_tr = {'Model\_tr': [],

    'Accuracy': [],

    'Precision': [],

    'Recall': [],

    'ROC AUC Score': []

}

# Run the training loop

for models\_tr, model\_name\_tr in model\_tr:

    #fit the model

    models\_tr.fit(X\_train\_tr\_sca, y\_train\_tr)

    #evaluate the model

    pred\_tr = models\_tr.predict(X\_test\_tr\_sca)

    acc\_score\_tr = accuracy\_score(y\_test\_tr, pred\_tr)

    pre\_score\_tr = precision\_score(y\_test\_tr, pred\_tr, average='macro')

    recal\_score\_tr = recall\_score(y\_test\_tr, pred\_tr, average='macro')

    roc\_score\_tr = roc\_auc\_score(y\_test\_tr, models\_tr.predict\_proba(X\_test\_tr\_sca), multi\_class='ovr')

    # Store metrics

    metrics\_tr['Model\_tr'].append(model\_name\_tr)

    metrics\_tr['Accuracy'].append(f"{round(acc\_score\_tr \* 100, 2)}%")

    metrics\_tr['Precision'].append(f"{round(pre\_score\_tr \* 100, 2)}%")

    metrics\_tr['Recall'].append(f"{round(recal\_score\_tr \* 100, 2)}%")

    metrics\_tr['ROC AUC Score'].append(f"{round(roc\_score\_tr \* 100, 2)}%")

    if model\_name\_tr != model\_tr[-1][1]:

        print(f"Evaluated {model\_name\_tr}")

# Create DataFrame from metrics dictionary

metrics\_df\_tr = pd.DataFrame(metrics\_tr)

metrics\_df\_tr

# using optimization to fit our random forest

rf\_fit\_tr = bayes\_rf.fit(X\_train\_tr\_sca, y\_train\_tr)

# Get the best parameters and best score

best\_params\_rf\_tr = rf\_fit\_tr.best\_params\_

best\_score\_rf\_tr = rf\_fit\_tr.best\_score\_

print(f"Best Parameters: {best\_params\_rf\_tr}")

print(f"Best Cross-Validation Accuracy: {best\_score\_rf\_tr:.4f}")

# Train the final model with the best parameters

model\_rf\_tr = RandomForestClassifier(\*\*best\_params\_rf, random\_state=42)

model\_rf\_tr.fit(X\_train\_tr\_sca, y\_train\_tr)

#Predict on the train set

rf\_train\_pred\_tr = model\_rf\_tr.predict(X\_train\_tr\_sca)

#get training prediction probabilities

rf\_train\_prob\_tr = model\_rf\_tr.predict\_proba(X\_train\_tr\_sca)

# Predict on the test set

rf\_test\_pred\_tr = model\_rf\_tr.predict(X\_test\_tr\_sca)

#get prediction probabilities

rf\_test\_prob\_tr = model\_rf\_tr.predict\_proba(X\_test\_tr\_sca)

#Evaluate the model performance

rf\_classification\_rep\_tr = classification\_report(y\_test\_tr, rf\_test\_pred\_tr)

print("Classification Report:")

print(rf\_classification\_rep\_tr)

**\*\****\*Observation\****\*\*** Our default random forest on transformed data performed better than the tuned model, so we will use the default model

# extracting our default random forest from the model list.

rf\_default = next(model for model, name in model\_tr if name == 'Random Forest')

# Fit the default Classifier

rf\_default.fit(X\_train\_tr\_sca, y\_train\_tr)

#Predict on the train set

rf\_train\_pred\_de = rf\_default.predict(X\_train\_tr\_sca)

#get training prediction probabilities

rf\_train\_prob\_de = rf\_default.predict\_proba(X\_train\_tr\_sca)

# Predict on the test set

rf\_test\_pred\_de = rf\_default.predict(X\_test\_tr\_sca)

#get prediction probabilities

rf\_test\_prob\_de = rf\_default.predict\_proba(X\_test\_tr\_sca)

#Evaluate the model performance

rf\_classification\_rep\_de = classification\_report(y\_test\_tr, rf\_test\_pred\_de)

rf\_auc\_de = roc\_auc\_score(pd.get\_dummies(y\_test\_tr), rf\_test\_prob\_de, multi\_class='ovo', average='macro')

print("Classification Report:")

print(rf\_classification\_rep\_de)

print("AUC Score:")

print(rf\_auc\_de)

**### 8.2.2. Train FNN on the feature-engineered data**

# we will use the existing FNN model building to train

# Train on the fnn model

fnn\_model\_tr = build\_fnn(X\_train\_tr\_sca.shape[1])

history\_tr = fnn\_model\_tr.fit(X\_train\_tr\_sca, y\_train\_tr, epochs=400, batch\_size=32, validation\_data =(X\_test\_tr\_sca,y\_test\_tr), verbose=2)

# Plot training & validation accuracy values

plt.figure(figsize=(10, 5))

# Plot Accuracy

plt.subplot(1, 2, 1)

plt.plot(history\_tr.history['accuracy'], label='Training Accuracy', marker='o', linestyle='-', markersize=2)

plt.plot(history\_tr.history['val\_accuracy'], label='Validation Accuracy', marker='o', linestyle='-', markersize=2)

plt.title('Training and Validation Accuracy on transformed data')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.grid(True)

# Plot Loss

plt.subplot(1, 2, 2)

plt.plot(history\_tr.history['loss'], label='Training Loss', marker='o', linestyle='-', markersize=2)

plt.plot(history\_tr.history['val\_loss'], label='Validation Loss', marker='o', linestyle='-', markersize=2)

plt.title('Training and Validation Loss on tranformed data')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

# Save the trained model

fnn\_model\_tr.save('fnn\_model\_tr.keras')

# Load the saved model

fnn\_model\_tr = load\_model('fnn\_model\_tr.keras')

#Get training predictions probabilities

fnn\_train\_pred\_tr = fnn\_model\_tr.predict(X\_train\_tr\_sca)

#convert training predictions to classes

fnn\_train\_prob\_tr = np.argmax(fnn\_train\_pred\_tr, axis=1)

# Get test predictions probabilities

fnn\_test\_pred\_tr = fnn\_model\_tr.predict(X\_test\_tr\_sca)

# convert prediction to classes

fnn\_test\_prob\_tr = np.argmax(fnn\_test\_pred\_tr, axis=1)

#Evaluate the model performance

fnn\_classification\_rep\_tr = classification\_report(y\_test\_tr, fnn\_test\_prob\_tr)

fnn\_auc\_tr = roc\_auc\_score(pd.get\_dummies(y\_test\_tr), fnn\_test\_pred\_tr, multi\_class='ovo', average='macro')

print("Classification Report:")

print(fnn\_classification\_rep\_tr)

print("AUC Score:")

print(fnn\_auc\_tr)

**### 8.2.3. Ensemble predictions of the RF and FNN model trained on the feature-engineered data and Evaluate**

```python

# Make a dataframe of FNN predictions

fnn\_predictions\_tr = pd.DataFrame(fnn\_test\_pred\_tr, columns=['Score\_0', 'Score\_1', 'Score\_2'])

fnn\_predictions\_tr['Label'] = fnn\_predictions\_tr.idxmax(axis=1).str.replace('Score\_', '').astype(int)

```

```python

# Make a dataframe of RF predictions

rf\_pred\_df\_tr = pd.DataFrame(rf\_test\_pred\_de, columns=['Label'])

# Make a dataframe of probabilities

rf\_prob\_df\_tr = pd.DataFrame(rf\_test\_prob\_de, columns=['Score\_0', 'Score\_1', 'Score\_2'])

# Concatenate the predictions and probabilities DataFrames

rf\_predictions\_tr = pd.concat([rf\_pred\_df\_tr, rf\_prob\_df\_tr], axis=1)

#Ensure both predictions DataFrames have the same index

rf\_predictions\_tr = rf\_predictions\_tr.reset\_index(drop=True)

fnn\_predictions\_tr = fnn\_predictions\_tr.reset\_index(drop=True)

# Extract necessary columns for probabilities

rf\_probs\_tr = rf\_predictions\_tr[['Score\_0', 'Score\_1', 'Score\_2']]

fnn\_probs\_tr = fnn\_predictions\_tr[['Score\_0', 'Score\_1', 'Score\_2']]

# Average the probabilities from both models

ensemble\_probs\_tr = (rf\_probs\_tr.values + fnn\_probs\_tr.values) / 2

# Check that the combined probabilities sum to 1

assert np.allclose(ensemble\_probs\_tr.sum(axis=1), 1), "Probabilities should sum to 1 for each sample"

# Determine the final predicted class based on the maximum combined probability

ensemble\_pred\_tr = np.argmax(ensemble\_probs\_tr, axis=1)

# Evaluate the model on our unseen data(y\_test)

ensemble\_classification\_rep\_tr = classification\_report(y\_test\_tr, ensemble\_pred\_tr)

ensemble\_auc\_tr = roc\_auc\_score(pd.get\_dummies(y\_test\_tr), ensemble\_probs\_tr, multi\_class='ovo', average='macro')

print("Classification Report:")

print(ensemble\_classification\_rep\_tr)

print("AUC Score:")

print(ensemble\_auc\_tr)

**### 8.2.4. Ensemble predictions of train and test data on the feature-engineered data and use it as features and train a meta model**

# Combine the predictions from base models for training data

X\_train\_meta\_tr = np.column\_stack([rf\_train\_prob\_de, fnn\_train\_pred\_tr])

# Target for the meta model

y\_train\_meta\_tr = y\_train\_tr

# Combine predictions from base models for test data

X\_test\_meta\_tr = np.column\_stack([rf\_test\_prob\_de, fnn\_test\_pred\_tr])

# selection of our meta model

model\_meta\_tr = [[LogisticRegression(random\_state=42), 'Logistic Regression'],

               [XGBClassifier(random\_state=42), 'XGB Classifier'],

               [ExtraTreesClassifier(random\_state=42), 'ET Classifier'],

               [SVC(probability=True,random\_state=42), 'SVC']]

metrics\_meta\_tr = {'Model\_meta\_tr': [],

    'Accuracy': [],

    'Precision': [],

    'Recall': [],

    'ROC AUC Score': []

}

# Run the training loop

for models\_meta\_tr, model\_name\_meta\_tr in model\_meta\_tr:

    #fit the model

    models\_meta\_tr.fit(X\_train\_meta\_tr, y\_train\_meta\_tr)

    #evaluate the model

    pred\_meta\_tr = models\_meta\_tr.predict(X\_test\_meta\_tr)

    acc\_score\_meta\_tr = accuracy\_score(y\_test\_tr, pred\_meta\_tr)

    pre\_score\_meta\_tr = precision\_score(y\_test\_tr, pred\_meta\_tr, average='macro')

    recal\_score\_meta\_tr = recall\_score(y\_test\_tr, pred\_meta\_tr, average='macro')

    roc\_score\_meta\_tr = roc\_auc\_score(y\_test\_tr, models\_meta\_tr.predict\_proba(X\_test\_meta\_tr), multi\_class='ovr')

    # Store metrics

    metrics\_meta\_tr['Model\_meta\_tr'].append(model\_name\_meta\_tr)

    metrics\_meta\_tr['Accuracy'].append(f"{round(acc\_score\_meta\_tr \* 100, 2)}%")

    metrics\_meta\_tr['Precision'].append(f"{round(pre\_score\_meta\_tr \* 100, 2)}%")

    metrics\_meta\_tr['Recall'].append(f"{round(recal\_score\_meta\_tr \* 100, 2)}%")

    metrics\_meta\_tr['ROC AUC Score'].append(f"{round(roc\_score\_meta\_tr \* 100, 2)}%")

    if model\_name\_meta\_tr != model\_meta\_tr[-1][1]:

        print(f"Evaluated {model\_name\_meta\_tr}")

# Create DataFrame from metrics dictionary

metrics\_meta\_tr\_df = pd.DataFrame(metrics\_meta\_tr)

metrics\_meta\_tr\_df

# initiate the XGB model

xgb\_model = XGBClassifier(objective='multi:softprob', eval\_metric='mlogloss', random\_state=42)

# Define the hyperparameter search space

param\_xgb = {

    'n\_estimators': Integer(50, 1000),

    'max\_depth': Integer(3, 20),

    'learning\_rate': Real(0.01, 0.3, prior='log-uniform'),

    'colsample\_bytree': Real(0.3, 1.0),

    'subsample': Real(0.4, 1.0),

    'min\_child\_weight': Integer(1, 10),

    'gamma': Real(1e-8, 1.0, prior='log-uniform'),

    'reg\_alpha': Real(1e-8, 1.0, prior='log-uniform'),

    'reg\_lambda': Real(1e-8, 1.0, prior='log-uniform')

}

# Set up the BayesSearchCV

bayes\_xgb = BayesSearchCV(

    estimator=xgb\_model,

    search\_spaces=param\_xgb,

    n\_iter=50,

    cv=StratifiedKFold(n\_splits=10, shuffle=True, random\_state=42),

    n\_jobs=-1,

    scoring='accuracy',

    random\_state=42

)

# Run the optimization

xgb\_fit\_tr = bayes\_xgb.fit(X\_train\_meta\_tr, y\_train\_meta\_tr)

# Get the best parameters and best score

best\_params\_xgb\_tr = xgb\_fit\_tr.best\_params\_

best\_score\_xgb\_tr = xgb\_fit\_tr.best\_score\_

# Print the best parameters and the best score

print(f"Best parameters found: {best\_params\_xgb\_tr}")

print(f"Best accuracy found: {best\_score\_xgb\_tr: .4f}")

# Train the final model with the best parameters

final\_model\_meta\_tr = XGBClassifier(\*\*best\_params\_xgb\_tr, objective='multi:softprob', eval\_metric='mlogloss',

                                use\_label\_encoder=False, random\_state=42)

final\_model\_meta\_tr.fit(X\_train\_meta\_tr, y\_train\_meta\_tr)

# Predict on the test set

y\_pred\_meta\_tr = final\_model\_meta\_tr.predict(X\_test\_meta\_tr)

# Get prediction probabilities

y\_prob\_meta\_tr = final\_model\_meta\_tr.predict\_proba(X\_test\_meta\_tr)

#Evaluate the model performance

meta\_classification\_rep\_tr = classification\_report(y\_test\_tr, y\_pred\_meta\_tr)

meta\_auc\_tr = roc\_auc\_score(pd.get\_dummies(y\_test\_tr), y\_prob\_meta\_tr, multi\_class='ovo', average='macro')

print("Classification Report:XGB RESULT ON PREDICTIONS OF FNN AND RF USING TRANSFORMED DATA")

print(meta\_classification\_rep\_tr)

print("AUC score:")

print(meta\_auc\_tr)

**## 9.0. Result Evaluation**

# Calculate metrics

metrics\_all = {

    'Model': ['RF\_OR', 'FNN\_OR', 'RF+FNN\_OR', 'LR\_RF+FNN\_OR', 'RF\_TR', 'FNN\_TR', 'RF+FNN\_TR', 'XGB\_RF+FNN\_TR'],

    'Accuracy': [accuracy\_score(y\_test, rf\_test\_pred), accuracy\_score(y\_test, fnn\_test\_prob\_or),

                 accuracy\_score(y\_test, ensemble\_pred), accuracy\_score(y\_test, y\_pred\_meta\_or),

                 accuracy\_score(y\_test\_tr, rf\_test\_pred\_de), accuracy\_score(y\_test\_tr, fnn\_test\_prob\_tr),

                 accuracy\_score(y\_test\_tr, ensemble\_pred\_tr), accuracy\_score(y\_test\_tr, y\_pred\_meta\_tr)],

    'Precision': [precision\_score(y\_test, rf\_test\_pred, average='macro'),

                  precision\_score(y\_test, fnn\_test\_prob\_or, average='macro'),

                  precision\_score(y\_test, ensemble\_pred, average='macro'),

                  precision\_score(y\_test, y\_pred\_meta\_or, average='macro'),

                  precision\_score(y\_test\_tr, rf\_test\_pred\_de, average='macro'),

                  precision\_score(y\_test\_tr, fnn\_test\_prob\_tr, average='macro'),

                  precision\_score(y\_test\_tr, ensemble\_pred\_tr, average='macro'),

                  precision\_score(y\_test\_tr, y\_pred\_meta\_tr, average='macro')],

    'Recall': [recall\_score(y\_test, rf\_test\_pred, average='macro'),

               recall\_score(y\_test, fnn\_test\_prob\_or, average='macro'),

               recall\_score(y\_test, ensemble\_pred, average='macro'),

               recall\_score(y\_test, y\_pred\_meta\_or, average='macro'),

               recall\_score(y\_test\_tr, rf\_test\_pred\_de, average='macro'),

               recall\_score(y\_test\_tr, fnn\_test\_prob\_tr, average='macro'),

               recall\_score(y\_test\_tr, ensemble\_pred\_tr, average='macro'),

               recall\_score(y\_test\_tr, y\_pred\_meta\_tr, average='macro')],

    'F1 Score': [f1\_score(y\_test, rf\_test\_pred, average='macro'),

                 f1\_score(y\_test, fnn\_test\_prob\_or, average='macro'),

                 f1\_score(y\_test, ensemble\_pred, average='macro'),

                 f1\_score(y\_test, y\_pred\_meta\_or, average='macro'),

                 f1\_score(y\_test\_tr, rf\_test\_pred\_de, average='macro'),

                 f1\_score(y\_test\_tr, fnn\_test\_prob\_tr, average='macro'),

                 f1\_score(y\_test\_tr, ensemble\_pred\_tr, average='macro'),

                 f1\_score(y\_test\_tr, y\_pred\_meta\_tr, average='macro')],

    'AUC Score': [roc\_auc\_score(pd.get\_dummies(y\_test), rf\_test\_prob, multi\_class='ovo', average='macro'),

                  roc\_auc\_score(pd.get\_dummies(y\_test), fnn\_test\_pred\_or, multi\_class='ovo', average='macro'),

                  roc\_auc\_score(pd.get\_dummies(y\_test), ensemble\_probs, multi\_class='ovo', average='macro'),

                  roc\_auc\_score(pd.get\_dummies(y\_test), y\_prob\_meta\_or, multi\_class='ovo', average='macro'),

                  roc\_auc\_score(pd.get\_dummies(y\_test\_tr), rf\_test\_prob\_de, multi\_class='ovo', average='macro'),

                  roc\_auc\_score(pd.get\_dummies(y\_test\_tr), fnn\_test\_pred\_tr, multi\_class='ovo', average='macro'),

                  roc\_auc\_score(pd.get\_dummies(y\_test\_tr), ensemble\_probs\_tr, multi\_class='ovo', average='macro'),

                  roc\_auc\_score(pd.get\_dummies(y\_test\_tr), y\_prob\_meta\_tr, multi\_class='ovo', average='macro')]

}

# Create DataFrame

df\_metrics\_all = pd.DataFrame(metrics\_all)

# Set index for easier plotting

df\_metrics\_all.set\_index('Model', inplace=True)

#set the colors

colors = ['#88CCEE', '#CC6677', '#DDCC77', '#117733', '#332288']

# set the fig size

#plt.subplots(figsize=(10, 6))

# Plotting

ax = df\_metrics\_all.plot(kind='bar', figsize=(10,6),color=colors)

#set labels and titles

plt.xlabel('Model')

plt.ylabel('Score')

plt.title('Model Performance Comparison')

# rotate x-axis

plt.xticks(rotation=0)

#Add gridlines to y-axis

plt.grid(axis='y')

# Adjust layout for better spacing

plt.tight\_layout()

#Move the legend below the x-axis

plt.legend(title='Metrics', loc='upper center', bbox\_to\_anchor=(0.5, -0.15), ncol=len(df\_metrics\_all.columns))

#show plot

plt.show();

print(df\_metrics\_all)

**## 9.1 Further evaluation**

```python

# plot learning curve to understand how the models performed across different folds in our transformed data

# Generate learning curves for Random Forest

train\_sizes\_rf\_or, train\_scores\_rf\_or, val\_scores\_rf\_or = learning\_curve(model\_rf\_or, X\_train\_or, y\_train,

                                                                cv=StratifiedKFold(n\_splits=10, shuffle=True, random\_state=42),

                                                                n\_jobs=-1,

                                                                train\_sizes=np.linspace(0.1, 1.0, 10), random\_state=42)

# Generate learning curves for Feedforward Neural Network

train\_sizes\_lr, train\_scores\_lr, val\_scores\_lr = learning\_curve(final\_model\_meta, X\_train\_meta\_or,

                                                                   y\_train\_meta\_or,

                                                                   cv=StratifiedKFold(n\_splits=10,

                                                                                      shuffle=True, random\_state=42),

                                                                   n\_jobs=-1,

                                                                   train\_sizes=np.linspace(0.1, 1.0, 10), random\_state=42)

# Calculate mean and standard deviation for training and validation scores

train\_mean\_rf\_or = np.mean(train\_scores\_rf\_or, axis=1)

train\_std\_rf\_or = np.std(train\_scores\_rf\_or, axis=1)

val\_mean\_rf\_or = np.mean(val\_scores\_rf\_or, axis=1)

val\_std\_rf\_or = np.std(val\_scores\_rf\_or, axis=1)

train\_mean\_lr = np.mean(train\_scores\_lr, axis=1)

train\_std\_lr = np.std(train\_scores\_lr, axis=1)

val\_mean\_lr = np.mean(val\_scores\_lr, axis=1)

val\_std\_lr = np.std(val\_scores\_lr, axis=1)

# Plotting the learning curves in subplots

plt.figure(figsize=(14, 6))

# Random Forest Learning Curve

plt.subplot(1, 2, 1)

plt.plot(train\_sizes\_rf\_or, train\_mean\_rf\_or, label='Training Score', color='red')

plt.plot(train\_sizes\_rf\_or, val\_mean\_rf\_or, label='Validation Score', color='gray')

plt.fill\_between(train\_sizes\_rf\_or, train\_mean\_rf\_or - train\_std\_rf\_or, train\_mean\_rf\_or + train\_std\_rf\_or, color='blue',

                 alpha=0.2)

plt.fill\_between(train\_sizes\_rf\_or, val\_mean\_rf\_or - val\_std\_rf\_or, val\_mean\_rf\_or + val\_std\_rf\_or, color='orange', alpha=0.2)

plt.title('Random Forest On Original Data Learning Curve')

plt.xlabel('Training Set Size')

plt.ylabel('Score')

plt.legend(loc='best')

plt.grid()

# LR on RF+FNN Predictions Learning Curve

plt.subplot(1, 2, 2)

plt.plot(train\_sizes\_lr, train\_mean\_lr, label='Training Score', color='red')

plt.plot(train\_sizes\_lr, val\_mean\_lr, label='Validation Score', color='gray')

plt.fill\_between(train\_sizes\_lr, train\_mean\_lr - train\_std\_lr, train\_mean\_lr + train\_std\_lr, color='blue', alpha=0.2)

plt.fill\_between(train\_sizes\_lr, val\_mean\_lr - val\_std\_lr, val\_mean\_lr + val\_std\_lr, color='orange', alpha=0.2)

plt.title('LR on RF+FNN predictions on Original Data Learning Curve')

plt.xlabel('Training Set Size')

plt.ylabel('Score')

plt.legend(loc='best')

plt.grid()

# Adjust layout and show plot

plt.tight\_layout()

plt.show()

```

# plot learning curve to understand how the models performed across different folds in our transformed data

# Generate learning curves for Random Forest

train\_sizes\_rf, train\_scores\_rf, val\_scores\_rf = learning\_curve(rf\_default, X\_train\_tr\_sca, y\_train\_tr,

                                                                cv=StratifiedKFold(n\_splits=10, shuffle=True, random\_state=42),

                                                                n\_jobs=-1,

                                                                train\_sizes=np.linspace(0.1, 1.0, 10), random\_state=42)

# Generate learning curves for Feedforward Neural Network

train\_sizes\_xgb, train\_scores\_xgb, val\_scores\_xgb = learning\_curve(final\_model\_meta\_tr, X\_train\_meta\_tr,

                                                                   y\_train\_meta\_tr,

                                                                   cv=StratifiedKFold(n\_splits=10,

                                                                                      shuffle=True, random\_state=42),

                                                                   n\_jobs=-1,

                                                                   train\_sizes=np.linspace(0.1, 1.0, 10), random\_state=42)

# Calculate mean and standard deviation for training and validation scores

train\_mean\_rf = np.mean(train\_scores\_rf, axis=1)

train\_std\_rf = np.std(train\_scores\_rf, axis=1)

val\_mean\_rf = np.mean(val\_scores\_rf, axis=1)

val\_std\_rf = np.std(val\_scores\_rf, axis=1)

train\_mean\_xgb = np.mean(train\_scores\_xgb, axis=1)

train\_std\_xgb = np.std(train\_scores\_xgb, axis=1)

val\_mean\_xgb = np.mean(val\_scores\_xgb, axis=1)

val\_std\_xgb = np.std(val\_scores\_xgb, axis=1)

# Plotting the learning curves in subplots

plt.figure(figsize=(14, 6))

# Random Forest Learning Curve

plt.subplot(1, 2, 1)

plt.plot(train\_sizes\_rf, train\_mean\_rf, label='Training Score', color='blue')

plt.plot(train\_sizes\_rf, val\_mean\_rf, label='Validation Score', color='orange')

plt.fill\_between(train\_sizes\_rf, train\_mean\_rf - train\_std\_rf, train\_mean\_rf + train\_std\_rf, color='blue', alpha=0.2)

plt.fill\_between(train\_sizes\_rf, val\_mean\_rf - val\_std\_rf, val\_mean\_rf + val\_std\_rf, color='orange', alpha=0.2)

plt.title('Random Forest On Transformed Data Learning Curve')

plt.xlabel('Training Set Size')

plt.ylabel('Score')

plt.legend(loc='best')

plt.grid()

# XGB on RF+FNN Predictions Learning Curve

plt.subplot(1, 2, 2)

plt.plot(train\_sizes\_xgb, train\_mean\_xgb, label='Training Score', color='blue')

plt.plot(train\_sizes\_xgb, val\_mean\_xgb, label='Validation Score', color='orange')

plt.fill\_between(train\_sizes\_xgb, train\_mean\_xgb - train\_std\_xgb, train\_mean\_xgb + train\_std\_xgb, color='blue', alpha=0.2)

plt.fill\_between(train\_sizes\_xgb, val\_mean\_xgb - val\_std\_xgb, val\_mean\_xgb + val\_std\_xgb, color='orange', alpha=0.2)

plt.title('XGB Classifier on RF+FNN predictions on Transformed Data Learning Curve')

plt.xlabel('Training Set Size')

plt.ylabel('Score')

plt.legend(loc='best')

plt.grid()

# Adjust layout and show plot

plt.tight\_layout()

plt.show()

```

# Plot the confusion matrix of the final selected model

cm\_xgb = confusion\_matrix(y\_test\_tr, y\_pred\_meta\_tr)

cc = ['#DDCC77', '#117733', '#332288']

# Plot the confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(cm\_xgb, annot=True, fmt='d', color=cc, cbar=False, xticklabels=risk\_map.keys(), yticklabels=risk\_map.keys())

plt.title('Confusion Matrix for XGB using ensembled predictions of FNN and RF on the transformed data')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.show()

```

```python

from itertools import cycle

# plot the ROC\_AUC score

# Select probabilities for the positive class

xgb\_prob = final\_model\_meta\_tr.predict\_proba(X\_test\_meta\_tr)

# Binarize the output labels

y\_test\_tr\_bin = label\_binarize(y\_test\_tr, classes=np.unique(y\_test\_tr))

n\_classes = y\_test\_tr\_bin.shape[1]

# Compute ROC curve and ROC area for each class

fpr = dict()

tpr = dict()

roc\_auc = dict()

for i in range(n\_classes):

    fpr[i], tpr[i], \_ = roc\_curve(y\_test\_tr\_bin[:, i], xgb\_prob[:, i])

    roc\_auc[i] = auc(fpr[i], tpr[i])

# Plot all ROC curves

plt.figure(figsize=(10, 8))

colors = cycle(['aqua', 'darkorange', 'cornflowerblue', 'red', 'green', 'purple'])

for i, color in zip(range(n\_classes), colors):

    plt.plot(fpr[i], tpr[i], color=color, lw=2,

             label=f'Class {i} (AUC = {roc\_auc[i]:.2f})')

plt.plot([0, 1], [0, 1], 'k--', lw=2)  # Diagonal line for random guess

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('XGB ROC Curve')

plt.legend(loc='lower right')

plt.grid()

plt.show()

```

```python

# plot feature importance of the RF used on the transformed data

# Get feature importances from the RF model

feature\_imp = rf\_default.feature\_importances\_

# Create a DataFrame for easier plotting

features\_df = pd.DataFrame({

    'Feature': X\_tr.columns,

    'Importance': feature\_imp

})

# Sort the DataFrame by importance

features\_df = features\_df.sort\_values(by='Importance', ascending=False)

# Plotting the feature importance

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

sns.barplot(x='Importance', y='Feature', data=features\_df, palette='cubehelix')

plt.title('Feature Importance - Random Forest on Feature-engineered data')

plt.xlabel('Importance')

plt.ylabel('Feature')

plt.show()

```

```python

# plot feature importance of the RF used on the transformed data

# Get feature importances from the RF model

feature\_imp\_or = model\_rf\_or.feature\_importances\_

# Create a DataFrame for easier plotting

features\_df\_or = pd.DataFrame({

    'Feature': X.columns,

    'Importance': feature\_imp\_or

})

# Sort the DataFrame by importance

features\_df\_or = features\_df\_or.sort\_values(by='Importance', ascending=False)

# Plotting the feature importance

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

sns.barplot(x='Importance', y='Feature', data=features\_df\_or, palette='cubehelix')

plt.title('Feature Importance - Random Forest on Original Data')

plt.xlabel('Importance')

plt.ylabel('Feature')

plt.show()