Vital-sign based Real-time Sepsis Diagnosis Using an Interpretable Machine Learning Model



By

Fahim Mahmud

ID: 18MEE008P

A thesis submitted in partial fulfilment of
the requirements for the degree of MASTER of SCIENCE in
Electrical and Electronic Engineering

Department of Electrical and Electronic Engineering
Chittagong University of Engineering and Technology

August, 2025

Copyright Declaration

I hereby declare that I am the sole author of the thesis titled ”Vital-sign based Real-time Sepsis Diagnosis Using an Interpretable Machine Learning Model”. Along with this, submitted a softcopy is true copy of my final hardcopy thesis including any required final revisions, as accepted by the board of examination. I hereby release my copyright to Chittagong University of Engineering and Technology (CUET) for uploading the electronic version of the same to the ’Digital Repository (Institutional Repository)’ of CUET after one year from the day of submission. Apart from this, the existing or to be approved Open Education Resources (OER) policies and procedures of Chittagong University of Engineering and Technology (CUET) shall be applicable to this research work. I also acknowledge my sincere delivering of copyright to Chittagong University of engineering and Technology (CUET) to reproduce this thesis by reprography or by other means, in total or by part at the necessity of individuals for the purpose of scholarly research.

Signature and Date

Fahim Mahmud
Student ID: 18MEE008P
Department of Electrical and Electronic Engineering
Chittagong University of Engineering & Technology (CUET)

Dedication

To my beloved wife,
whose unwavering love, patience, and support
made this journey possible.

List of Publications

Conference Papers

1. F. Mahmud, N. S. Pathan and M. Quamruzzaman, ”Early detection of Sepsis in critical patients using Random Forest Classifier,” 2020 IEEE Region 10 Symposium (TENSYMP), Dhaka, Bangladesh, 2020, pp. 130-133, doi: 10.1109/TENSYMP50017.2020.9231011.

2. F. Mahmud, N. S. Pathan and M. Quamruzzaman, ”Early Detection of Sepsis in ICU Patients Using Logistic Regression,” 2019 3rd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE), Rajshahi, Bangladesh, 2019, pp. 173-176, doi: 10.1109/ICECTE48615.2019.9303513.

Journal Papers

1. F. Mahmud, M. Quamruzzaman, AI Sanka, RCC Cheung, MH Chowdhury, ”Interpretable Machine Learning-Based Real-time Sepsis Diagnosis” (Submitted to Scientific Reports, under review)

Approval by the Supervisor

This is to certify that Fahim Mahmud has carried out this work under my supervision, and that he has fulfilled relevant Academic Ordinance of the Chittagong University of Engineering and Technology, so that he is qualified to submit the following thesis in application for the degree of MASTER of SCIENCE in Electrical and Electronic Engineering.

Signature and Date

Dr. Muhammad Quamruzzaman
Professor
Department of Electrical and Electronic Engineering
Chittagong University of Engineering & Technology (CUET)

Acknowledgement

First of all, I am grateful to Allah for his blessings throughout this journey. Then, I would like to express my deepest gratitude to my supervisor and co-supervisor, for their invaluable guidance, support, and encouragement throughout my research. Their expertise and insightful feedback have been instrumental in shaping this thesis.

I am grateful to the Department of Electrical and Electronic Engineering at Chittagong University of Engineering and Technology (CUET) for providing me with the resources and environment necessary to conduct my research.

I extend my heartfelt appreciation to my family, for their unwavering support, patience, and understanding during this journey. Their encouragement kept me motivated through the toughest times.

Lastly, I would like to thank my colleagues and friends, for their collaboration and technical assistance. Their help and camaraderie made this journey enriching and fulfilling.

Abstract

Sepsis, defined by Sepsis-3 as life-threatening organ dysfunction caused by a dysregulated host response to infection, imposes profound physiological and socio-economic burdens. Globally, it is associated with roughly 50 million cases and 11 million deaths per year, prolonged ICU stays, and substantial healthcare costs and productivity loss. Because organ failure can evolve rapidly, timely detection is critical yet remains challenging in practice due to label noise, missingness, and severe class imbalance in clinical data. This work investigates lightweight machine-learning models for early sepsis detection using seven non-invasive vital signs (heart rate, temperature, systolic/diastolic/mean arterial pressure, oxygen saturation (SpO_2), and respiratory rate). We took hourly data, addressed missingness and imbalance with a simple preprocessing pipeline, and used SHAP and LIME to provide model insight. We developed a low-cost point-of-care prototype (Raspberry Pi and commodity sensors with a Python/Django backend) that demonstrates feasibility for real-time prediction as well as telemedicine monitoring. On the PhysioNet Challenge dataset, the system achieved a utility score of $\approx$ 46%, which is an improvement of approximately 10% from the state-of-the art works. Our model also achieved an accuracy of 86.49%, and an AUROC of 0.94. For external validation on the MIMIC-IV demo cohort, our best model (XGBoost) attained 40% recall for the sepsis class, substantially outperforming a partial qSOFA baseline at 6% recall. These results indicate that simple vital-sign models can materially improve sensitivity over rule-based screening while remaining deployable on affordable hardware; future work should pursue validation on full-scale cohorts and prospective evaluation in real-world ICU workflows.

বিমূর্ত

সেপসিস বলতে একটি জীবনঘাতী অঙ্গ অকার্যকারিতাকে বোঝানো হয় যা মূলত সংক্রমণ প্রতিরোধ করার সময় শরীরের অনিয়ন্ত্রিত প্রতিক্রিয়া থেকে সৃষ্ট। সেপসিসের শারীরবৃত্তীয় এবং আর্থসামাজিক প্রভাব সুদূরপ্রসারী। সারা পৃথিবীজুড়ে প্রতিবছর আনুমানিক ৫ কোটি মানুষ সেপসিসে আক্রান্ত হয় এবং ১ কোটির মত মারা যায়। এছাড়াও সেপসিসের কারণে দীর্ঘ সময় আইসিইউতে থাকার প্রয়োজন পড়ে, যার ফলে স্বাস্থ্যখাতে খরচ উল্লেখযোগ্য মাত্রায় বেড়ে যায় এবং উৎপাদনশীলতা ব্যাহত হয়। সেপসিসের কারণে শরীরের গুরুত্বপূর্ণ অঙ্গগুলির কার্যক্ষমতা হারানোর প্রক্রিয়াটি যেহেতু ত্বরিতগতিতে ঘটে, তাই সময়মত এটি শনাক্ত করা অত্যন্ত জরুরী। তবে শনাক্তকরণের মাপকাঠির অসামঞ্জস্যতা, অসম্পূর্ণ তথ্য এবং সুস্থ মানুষের তুলনায় সেপসিস রোগীর হার অনেক কম হওয়ার কারণে রোগনির্ণয়ের বিষয়টি বেশ কষ্টসাধ্য। আমাদের গবেষণাতে আমরা কম গাণিতিক জটিলতা বিশিষ্ট তথা সরল মেশিন লার্নিং মডেলের সাহায্যে কয়েকটি সহজপ্রাপ্য পরিমাপের (হৃদস্পন্দন, শরীরের তাপমাত্রা, রক্তচাপ, রক্তে অক্সিজেনের পরিমাণ, শ্বাস-প্রশ্বাসের হার) উপর ভিত্তি করে সময়মত রোগনির্ণয়ের চেষ্টা করেছি। তথ্য প্রক্রিয়াজাতকরণের একটি সরল মডেলের সাহায্যে ঘণ্টাভিত্তিক তথ্যের ভিত্তিতে আমরা অসম্পূর্ণ এবং ভারসাম্যহীন উপাত্তগুলোকে সমন্বয় করেছি এবং শাপ এবং লাইম নামক দুটি গাণিতিক সমীকরণ ব্যবহার করে মডেলটি কিভাবে কাজ করে সে বিষয়ে আলোকপাত করার চেষ্টা করেছি। আমাদের প্রস্তাবিত স্বল্পমূল্যের পরীক্ষামূলক নমুনাটি রোগীর তথ্যের ভিত্তিতে তাৎক্ষণিকভাবে সেপসিস হওয়ার সম্ভাবনা নির্ণয়ে উপযোগী এবং একইসাথে এর সাহায্যে ইন্টারনেটভিত্তিক অভিজ্ঞ চিকিৎসা সেবা প্রদান করাও সম্ভব। ফিজিওনেট ডাটাসেটে আমাদের মডেলটি ৪৬% ইউটিলিটি স্কোর অর্জন করেছে যা এই ডাটাবেজের উপর করা উল্লেখযোগ্য কাজের চেয়ে আনুমানিক ১০% বেশী। উপরন্তু, আমাদের মডেলটি ৮৬.৪৯% নির্ভুলতা এবং ০.৯৪ এইউআরওসি অর্জন করেছে। অধিকতর যাচাইয়ের জন্য আমরা মিমিক-৪ নমুনা ডাটাসেট ব্যবহার করেছি যেখানে আমাদের সবচেয়ে ভাল মডেলটি সেপসিস রোগীর ক্ষেত্রে ৪০% সংবেদনশীলতা দেখিয়েছে যা দ্রুত সেপসিস নির্ণয়ের বহুল প্রচলিত পরিমাপক আংশিক কিউসোফা এর মাধ্যমে প্রাপ্ত ৬% সংবেদনশীলতা থেকে উল্লেখযোগ্য মাত্রায় বেশী। প্রাপ্ত ফলাফলের ভিত্তিতে বলা যায় যে সহজপ্রাপ্য শারীরিক পরিমাপভিত্তিক মেশিন লার্নিং মডেল প্রচলিত পদ্ধতির চেয়ে নির্ভুল ফলাফল দিয়ে থাকে এবং একইসাথে কম খরচে বাস্তবায়নযোগ্য। ভবিষ্যতে বেশী পরিমাণে আইসিইউ তথ্যের সাহায্যে মডেলটিকে অধিকতর যাচাই করা যেতে পারে।

Table of Contents

Copyright Declaration i

Dedication ii

List of Publications iii

Approval by the Supervisor iv

Acknowledgement v

Abstract vi

বিমূর্ত vii

Table of Contents viii

List of Figures xii

List of Tables xvi

Nomenclature xvii

1 Introduction 1

1.1 Definition of Sepsis 1

1.2 Global Health Burden 1

1.3 Economic Impact 2

1.4 Problem Statement 3

1.5 Objectives 4

1.6 Thesis Organization 4

2 Literature Review **6**

2.1 Categorization of Existing Works 6

2.2 Review of Individual Studies 7

2.2.1 Classical Machine Learning Approaches 9

2.2.2 Deep-Learning Based Approaches 18

2.2.3 Hybrid Approaches 23

2.2.4 Hardware-Oriented Approaches 25

2.3 Comparative Summary 25

3 Methodology **27**

3.1 Overall workflow 27

3.2 Dataset Description 27

3.2.1 Main Dataset 28

3.2.2 Extrenal Public Dataset 29

3.3 Explorator Data Analysis (EDA) 29

3.4 Feature Engineering 29

3.5 Feature Selection 33

3.6 Missing Data Handling 35

3.7 Class Imbalance Handling 38

3.8 Model Description and Architecture 42

3.8.1 Random Forest 42

3.8.2 Gradient Boost 45

3.8.3 Extreme Gradient Boost 49

3.8.4 Convolutional Neural Network 50

3.8.5 Recurrent Neural Network 57

3.9 Federated Learning 60

3.10 Shapley Values 62

3.11 Locally Interpretable Model-agnostic Explanation 63

4 Result Analysis **65**

4.1 Evaluation Metrics . . . 65

4.1.1 Confusion Matrix . . . 65

4.1.2 AUROC (Area Under Receiver Operating Characteristics) . . . 67

4.1.3 Utility Score . . . 67

4.1.4 Confidence Metrics . . . 69

4.2 Comparison Between Models Based on Class Imbalance . . . 69

4.2.1 Random Undersampling . . . 69

4.2.2 Cost-Sensitive Learning . . . 70

4.2.3 Synthetic Minority Oversampling Technique (SMOTE) . . . 71

4.2.4 Adaptive Synthetic Sampling (ADASYN) . . . 73

4.2.5 Non-Overlapping Subset Ensemble . . . 74

4.3 Comparison Between Models Based on Missing Value Handling . . . 75

4.3.1 Last Observation Carried Forward (LOCF) . . . 75

4.3.2 Constant Imputation . . . 76

4.3.3 Multiple Imputations by Chained Equations . . . 78

4.3.4 No Imputation . . . 78

4.4 Comparison Between Models Based on Feature Selection . . . 79

4.4.1 Variance Threshold . . . 80

4.4.2 Analysis of Variance . . . 81

4.4.3 Recursive Feature Elimination (RFE) . . . 82

4.5 Performance with CNN . . . 83

4.6 Performance with RNN and LSTM . . . 84

4.7 Performance with Federated Learning . . . 86

4.8 Reliability of Probability Estimates . . . 87

4.9 Model Size . . . 91

4.10 Model Performance on MIMIC-IV Demo . . . 91

5 Model Interpretability **94**

5.1 Beeswarm Plot . . . 94

5.2 Bar Plot 95
5.3 Waterfall Plot 96
5.4 LIME Plot 96
5.5 Model-Based Feature Importance 99

6 Edge Device Prototype **101**
6.1 Edge Device Integration with Raspberry Pi and Arduino 101
6.2 Telemedicine-Based Sepsis Monitoring 104
6.3 Performance on Edge Device 105

7 Conclusion and Future Work **107**
7.1 General Summary 107
7.2 Key Contributions 108
7.3 Limitations 108
7.4 Future Work 109

Bibliography **110**

A Python Source Code **119**
A.1 Python Script for Model Training 119
A.2 Python Script for Real-Time Prediction 121

B Key Django Modules **125**
B.1 Project Layout 125
B.2 Model 126

List of Figures

1.1 Organization of the thesis 5

2.1 Distribution of data sources to develop sepsis detection models 8

2.2 Types of input features used across studies on sepsis diagnosis 9

2.3 Top-ranked 20 input features among studies on sepsis detection 9

3.1 Overall workflow for sepsis detection, interpretation, edge device integration, and telemedicine-based sepsis monitoring 28

3.2 Sepsis prevalence across datasets 30

3.3 Per-feature missing percentages 30

3.4 Processing patient data as images for CNN training 32

3.5 Creation of non-septic subset with equal proportion to septic data 41

3.6 Training of multiple models on the created subsets 41

3.7 Prediction with ensemble model via voting 42

3.8 Visualization of a sample decision tree (depth 2) from the trained random forest model, showing feature-based splits at each node 43

3.9 Another sample decision tree from the random forest model, highlighting variation in splitting criteria among ensemble members 44

3.10 A depth-2 tree from the gradient boosting model showing early-stage feature splits 46

3.11 Another gradient boosting tree highlighting how shallow splits refine predictions 47

3.12 A 3×3 kernel slides over a 5×5 input image producing a 3×3 output feature map 52

3.13 A 3×3 kernel with a stride of 2 slides over a 5×5 input image producing a 2×2 output feature map 53
3.14 Step function 54
3.15 Sigmoid function 55
3.16 Hyperbolic tangent 55
3.17 Rectified linear unit 56
3.18 Leaky rectified linear unit 56
3.19 Exponential linear unit 57
3.20 Architecture of a many-to-one RNN 59
3.21 LSTM cell 59
3.22 A local surrogate model to approximate the complex decision boundary . 64

4.1 Confusion matrix 66
4.2 Utility score diagram for sepsis patients 67
4.3 Utility score diagram for non-sepsis patients 68
4.4 AUROC of models with undersample technique 70
4.5 Performance comparison of models with undersampling method 70
4.6 AUROC of models with cost-sensitive learning technique 71
4.7 Performance comparison of models with cost-sensitive learning method . 71
4.8 AUROC of models with SMOTE 72
4.9 Performance comparison of models with SMOTE method 72
4.10 AUROC of models with ADASYN 73
4.11 Performance comparison of models with ADASYN method 73
4.12 AUROC of models with NOSE 74
4.13 Performance comparison of models with Non-Overlapping Subset Ensemble 75
4.14 AUROC of models with LOCF 76
4.15 Performance comparison of models with LOCF 76
4.16 AUROC of models with constant imputation 77
4.17 Performance comparison of models with constant imputation 77
4.18 AUROC of models with MICE 78

4.19 Performance comparison of models with MICE 79
4.20 AUROC of extreme gradient boost model with no imputation 79
4.21 Performance of XGB with no imputation 80
4.22 AUROC of different models with variance threshold feature selection . . 80
4.23 Performance of different models with variance threshold feature selection 81
4.24 AUROC of models with ANOVA 81
4.25 Performance comparison of models with ANOVA feature selection 82
4.26 AUROC of models with RFE 82
4.27 Performance comparison of models with RFE 83
4.28 Training curve for modified MiniVGG architecture with 3-hour window . 84
4.29 Training curve for modified MiniVGG architecture with 6-hour window . 84
4.30 Performance comparison of modified MiniVGGNet with varying window lengths 85
4.31 MiniVGGNet on undersampled data with timestep of 6 hours 85
4.32 MiniVGGNet on undersampled data with timestep of 3 hours 85
4.33 Training curve for RNN with 3-hour time window 86
4.34 Training curve for LSTM with 3-hour time window 86
4.35 RNN with a 3-hour time window 87
4.36 LSTM with a 3-hour time window 87
4.37 Training curve for RNN with 6-hour time window 87
4.38 Training curve for LSTM with 6-hour time window 88
4.39 RNN with a 6-hour time window 88
4.40 LSTM with a 6-hour time window 88
4.41 Peformance comparison of RNN and LSTM on varying window length . 89
4.42 Performance comparison among individual model and aggregated model . 89
4.43 Probability calibration curve for random forest 90
4.44 Probability calibration curve for gradient boost 90
4.45 Probability calibration curve for extreme gradient boost 90
4.46 Model size comparison on log scale 91
4.47 Lollipop plot of sepsis-class recall in MIMIC-IV demo database 91

5.1 Beeswarm plot for random forest model with 50 features 95
5.2 Bar plot for random forest model with 50 features 95
5.3 Waterfall plot comparison for four instances 97
5.4 LIME plot comparison for four instances 98
5.5 Gain-based feature importance 99
5.6 Weight-based feature importance 99

6.1 Proof-of-concept implementation of the edge-integrated sepsis detection system 102
6.2 Experimental setup of real-time prediction using a raspberry pi and sensor, with telemedicine integration via a live monitoring interface on the laptop 102
6.3 Real-time sensor reading and prediction with raspberry pi 103
6.4 Telemedicine based sepsis monitoring 104
6.5 Live dashboard for sepsis monitoring 104
6.6 Confusion matrix for deployed model 105

List of Tables

3.1 qSOFA criteria 31

3.2 National Early Warning Score (NEWS) criteria 31

3.3 Patient data with missing values 35

3.4 Patient data with initial imputation 36

3.5 Patient data with final imputation 36

3.6 Patient data with missing blood pressure values 37

3.7 Patient data with initial imputation for blood pressure values 37

3.8 Patient Data with final imputation for Blood Pressure Values 38

3.9 CNN model summary 51

4.1 Brier score on the test set 89

4.2 Summary of major works in sepsis detection using machine learning . . . 93

5.1 Comparison of actual and surrogate model prediction 96

6.1 Specifications of components 103

6.2 Performance metrics of temperature-based gradient boosting model . . . 105

Nomenclature

AdaBoost	—	Adaptive Boosting
ADASYN	—	Adaptive Synthetic Sampling
AE	—	Autoencoder
ANOVA	—	Analysis of Variance
AUROC	—	Area Under the Receiver Operating Characteristic Curve
CNN	—	Convolutional Neural Network
CRF	—	Conditional Random Field
EHR	—	Electronic Health Record
ESN	—	Echo State Network
FCNN	—	Fully Connected Neural Network
GBM	—	Gradient Boosting Machine
GRU	—	Gated Recurrent Unit
HEA	—	Heterogeneous Event Aggregation
HMM	—	Hidden Markov Model
ICU	—	Intensive Care Unit
LightGBM	—	Light Gradient Boosting Machine
LIME	—	Local Interpretable Model-Agnostic Explanations
LOCF	—	Last Observation Carried Forward
LSTM	—	Long Short-Term Memory
NaN	—	Not a Number

NEWS	—	National Early Warning Score
NOU	—	Normalized Observed Utility
PCA	—	Principal Component Analysis
PPG	—	Photoplethysmography
PRESEP	—	Prehospital Early Sepsis Detection
PSV	—	Pipe Separated Value
qSOFA	—	Quick Sequential Organ Failure Assessment
RF	—	Random Forest
RNN	—	Recurrent Neural Network
RUSBoost	—	Random Under Sampling Boosting
SHAP	—	SHapley Additive exPlanations
SIRS	—	Systemic Inflammatory Response Syndrome
SMOTE	—	Synthetic Minority Oversampling Technique
SOFA	—	Sequential Organ Failure Assessment
SVM	—	Support Vector Machine
TAE	—	Temporal Autoencoder
TCN	—	Temporal Convolutional Network
TTE	—	Time To Event
XGBoost	—	Extreme Gradient Boosting

Chapter 1

Introduction

Sepsis is a life-threatening condition resulting from the body's dysregulated response to infection, often leading to organ failure and death if not diagnosed and treated promptly. In this chapter, we introduce the critical importance of early sepsis detection and establish the motivation for developing a real-time, interpretable machine learning-based diagnostic system using vital signs. We begin by defining sepsis and highlighting its global health burden and economic impact. This is followed by a clear problem statement, a summary of the key objectives of this research, and an outline of the overall thesis structure.

1.1 Definition of Sepsis

Sepsis is defined as a life-threatening medical condition that arises when the human body attempts to eliminate the bacterial infection within the body. Any sort of infection can develop to sepsis whether it is bacterial, viral or fungal. However, major organs such as kidneys, urinary tract system, wounds or burns are more prone to developing this physiological disorder. In doing so, the body's immune response sometimes reacts in a highly aggressive manner. This acute response leads to the damage of vital organs within the body such as the heart, the lungs and so on. The physiological damage rendered to these organs is irreversible in most cases. As a result, the patient's chance of survival is very low, making sepsis one of the major causes of death all over the world.

1.2 Global Health Burden

Many survivors experience a significant decline in physical and cognitive function following recovery, a condition commonly referred to as post-sepsis syndrome. Sepsis related mortality is on the rise in many countries. In USA, nearly 270000 people die

from sepsis each year which is a higher figure than fatality rate of lung cancer (around 150000 deaths per year) and breast cancer (roughly 41000 deaths per annum) [1]. People with sepsis are 2-3 times more likely to be readmitted to hospital [2]. Number of hospital re-admissions in septic cases per year was approximately 150000 whereas those due to heart failure and pneumonia were estimated to be 80000 and 60000 per annum.

1.3 Economic Impact

Sepsis has multifaceted impacts on society, affecting individuals, healthcare systems, and the economy. The financial burden sepsis puts on the society as a whole is significant. The economic burden of sepsis extends beyond the initial diagnosis and treatment phases, encompassing a wide range of direct and indirect costs. These include the expenses incurred for diagnostic procedures, therapeutic interventions during the acute phase, prolonged hospitalization, and long-term care. Additionally, significant financial strain may arise from the management of post-sepsis complications, rehabilitation, and the sustained reduction in quality of life experienced by survivors, often necessitating continued medical support and social assistance. A study [3] conducted on the hospital-related costs around the world assessed the cost of patients' stay in countries from different income classes. The highest sepsis-related cost per patient per stay was observed in Italy with a figure of approximately 92000 euro [4]. On the other hand, the lowest estimated overall cost per stay for a single patient was the lowest in Indonesia with a cost of around 1100 euro only [5]. However, according to this study, the nationwide cost of sepsis was the highest in the USA, ranging roughly between 20 to 50 billion euros per year [6]. Brazil also endures a substantial economic burden, with annual losses estimated at approximately 35 billion euros per year [7]. Moreover, Brazil, among the other studied countries, spends the highest portion of its allocated healthcare budget (16% of the total allocated budget) to the diagnosis and treatment of sepsis. This proportion is the lowest for the Greece which spends only 0.29% of its total healthcare budget to sepsis-related medical intervention [8]. From a recent report [9] on sepsis-related healthcare costs in Australia, one of the continents, it is found that the direct cost of hospital admission and associated treatment was estimated to be 700 million Australian Dollar (AUD) per annum. Direct cost of sepsis in the age group under 45 stands roughly at 200 million AUD whereas the figure for age group above 45 is more than twice that of the lower age group. In addition to the direct cost burden, sepsis also has a long-term effect on the Australian economy according to this report. It mentioned that the indirect cost of sepsis due to lost productivity and premature death owing to sepsis accounts for roughly 4 billion AUD per year. Based on the findings of the report [9], direct hospital expenditures represent only 14% of the total societal burden of sepsis in Australia, highlighting the substantial impact of indirect costs

such as productivity loss and premature mortality. It emphasizes the need to diagnose and treat sepsis in its earliest stage before it can have any substantial impact on the human body. A report [10] on the costs associated with sepsis afflicted patients illustrated that the expenses are directly proportional to the severity of sepsis. The total hospital expenditure for the septic patients without any major organ dysfunction was around 16000 dollar whereas those of the patients with severe sepsis and septic shock were close to 25000 dollar and 39000 dollar, respectively. Another research article [11] on the societal costs of sepsis in Netherlands showed that sepsis severely affects life quality even after survival with an annual burden of approximately 57000 QALY (Quality Adjusted Life Years). These lost QALYs are valued at around 3.4 billion euros per annum. Moreover, the indirect cost associated with the inability to work and lost productivity is close to 100 million euros annually using the friction method and roughly 2.8 billion euros according to the human capital method. These are two methods to estimate the economic burden of a disease. The friction method focuses on the time it takes for a person to be replaced in the workforce after their premature death or disability. For example, if a person earns $50,000 per year, and it takes 2 years to replace their role, the friction method would estimate that the loss is $100,000. The human capital method estimates the economic loss based on the future earnings that would have been generated by an individual over their lifetime if they had not been affected by the disease or premature death. For instance, if a person is expected to earn $50,000 per year until they turn 65, and they die at 40, the human capital method would calculate the lost income as the sum of the future 25 years of earnings. That's why the estimate is much higher in the latter of the two methods. The estimated indirect costs of sepsis in Austria, Switzerland, Germany are 484-686 million euro, 331-805 million euro and 2622 million euro per year, respectively. [11].

1.4 Problem Statement

The main challenges associated with sepsis are twofold in nature. The first one is the ambiguity of sepsis definition. At present, a number of definitions related to sepsis, septic shock and organ dysfunction exist which lead to discrepancies in sepsis identification, proper and consistent documentation and treatment. Consequently, a task force of 19 members redefined the definition of sepsis in an attempt to exclude the previous limitations regarding systemic inflammatory response syndrome (SIRS) criteria and focused on the dysregulated host response to infection [12]. The other aspect of sepsis diagnosis challenge is the requirement of continuous measurements of a large number of parameters such as heart rate, blood pressure etc. and laboratory values such as creatinine, glucose, pH level etc. in blood for proper and accurate prediction of this medical condition. This mandates Intensive Care Unit (ICU) admission for continuous monitoring and

life-support interventions. However, ICU support is not wide, specially in low-income and underdeveloped countries. Nevertheless, it is suspected that the number of sepsis cases in those countries is even higher than the developed ones but goes unnoticed due to lack of expert healthcare professionals and poor documentation. So, there is a critical need for a system capable of detecting sepsis with reliable accuracy, without relying on complex or high-end infrastructure. Considering this scenario, we have proposed a simple, computationally lightweight model that can detect sepsis in such resource-constrained environment with reasonable accuracy.

1.5 Objectives

The major objectives of our work are outlined below:

1. To develop a Machine Learning (ML) workflow that is capable of detecting sepsis using only non-invasive vital signs which are easily obtainable.
2. To gain insight into the working of the model and thus, bridging the communication gap between healthcare professionals and blackbox ML models and promote trust in the ML models.
3. To assess the feasibility of deploying the model on edge device as a proof of concept for easier integration into real life.
4. To develop a skeletal architecture of telemedicine-based sepsis monitoring using Python-Django web framework.

1.6 Thesis Organization

Our thesis is outlined as follows: The first chapter introduces the problem of sepsis, its clinical relevance, and the motivation for pursuing an accessible, affordable and accurate detection system. Chapter 2 reviews existing literature on sepsis diagnosis, emphasizing the limitations of current methods, especially in resource-limited settings. Chapter 3 details the proposed methodology, including system design, data preprocessing, feature selection, and the machine learning models employed. In Chapter 4, we present the experimental results, showcasing the performance of our approach and comparing it with existing techniques. This chapter also provides a critical discussion of the findings, their implications, and the challenges encountered. Chapter 5 depicts the reasoning behind our model's predictions via SHAP and LIME plots. It also explores the model-based feature importances for further insights. Chapter 6 illustrates our edge device prototype. Finally,

Chapter 7 concludes the thesis with a summary of contributions, limitations in our work and outlines potential directions for future research. Figure 1.1 gives the visual of our work.

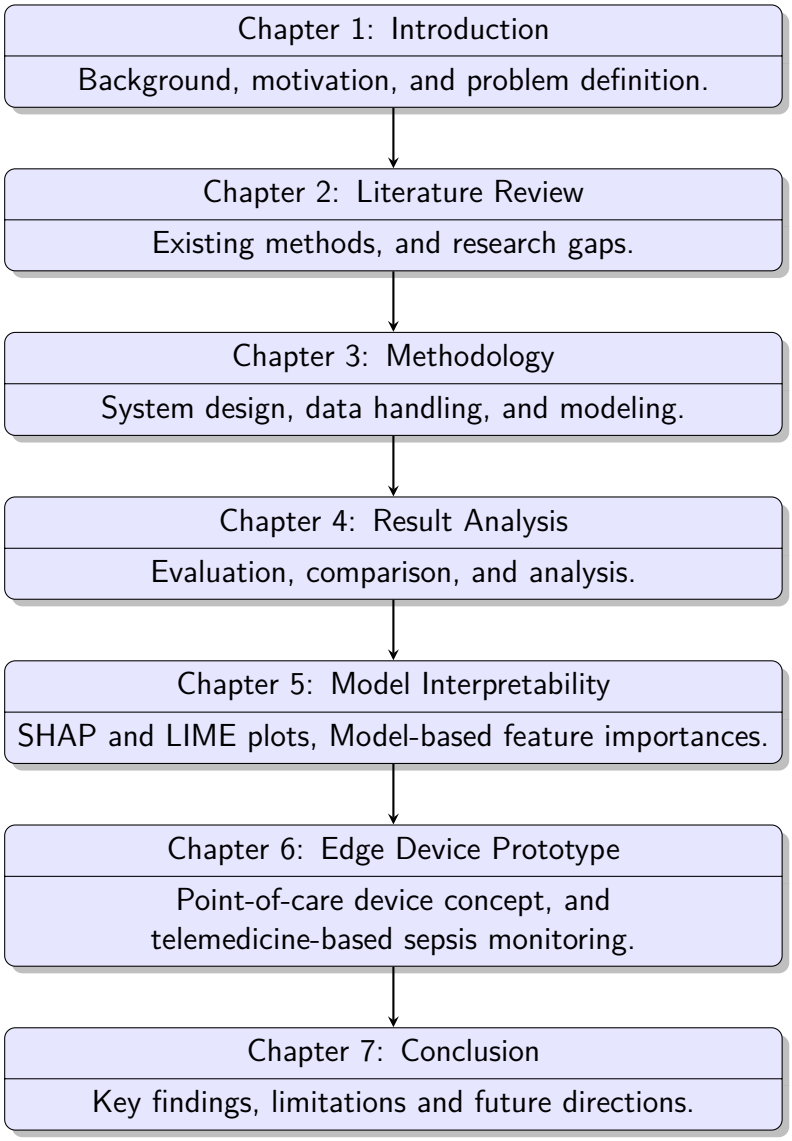


Figure 1.1: Organization of the thesis

Chapter 2

Literature Review

Sepsis remains a major global health concern due to its high mortality rate, diagnostic complexity, and substantial financial burden. Over the past decades, numerous studies have focused on developing reliable diagnostic tools and predictive models to enable timely intervention. This chapter critically examines the existing body of literature on sepsis diagnosis, with particular emphasis on machine learning-based approaches, data challenges, and the feasibility of implementing such solutions in resource-constrained settings with an aim to identify the research gaps.

2.1 Categorization of Existing Works

To systematically review prior works on sepsis detection, we categorize the literature across nine critical dimensions:

1. Definition of sepsis
2. Dataset used
3. Input features
4. Approaches to handle missing data
5. Approaches to handle class imbalance
6. Types of machine learning models used
7. Model feasibility on edge devices
8. Interpretability
9. Telemedicine-based deployment

This categorization not only aids comparison but also identifies the gaps and opportunities that our proposed system aims to address.

Majority of the previous works use the latest definition of sepsis (known as sepsis-3) as proposed by the task force of 19 members from the Society of Critical Care Medicine and the European Society of Intensive Care Medicine. However, some of the works also used the previous sepsis-2 definition. The datasets used in research works on sepsis diagnosis using machine learning are also sourced from mainly three databases, namely, Medical Information Mart for Intensive Care (MIMIC) II and III and Physionet-2019 challenge data [13]. Figure 2.1 shows the different data sources that were used in the existing works. As can be seen clearly, MIMIC-III database accounts for more than 30% of the current works while the Physionet-2019 challenge data holds the second position with a figure of around 12%.

Most of the existing studies so far rely heavily on vital signs and laboratory values along with demographic information like sex, age etc. as the input features for sepsis classifiers [14]. As shown in Figure 2.2, vital signs are employed in over 70% of the reviewed literature, whereas laboratory data and demographic variables are used in approximately 55–65% of the studies. Although imaging information has been used extensively for a large number of disease diagnosis, its use is very narrow in the premise of sepsis diagnosis, which is evident from Figure 2.2 where the percentage stands roughly at 2%. This also shows a research gap which can be explored further to gain insights into how sepsis condition affects image data such as X-ray, MRI and CT scans. Domain knowledge-based features such as Sequential Organ Failure Assessment (SOFA), quick SOFA (qSOFA), SIRS were used in some works as well [15].

Figure 2.3 shows the top-ranked 20 features used among the studies. It clearly shows that heart rate, body temperature and blood pressure (SBP, DBP, MAP) are used among the vital signs whereas most of the other features are laboratory values. Age and gender are also used which are demographic features.

2.2 Review of Individual Studies

This section presents a review of individual studies related to sepsis diagnosis using machine learning techniques. The selected works are categorized based on their methodology, including classical machine learning, deep learning, hybrid models, and hardware-oriented implementations. Each study is summarized in terms of its key contributions, methodologies, datasets, and relevance to interpretable sepsis prediction.

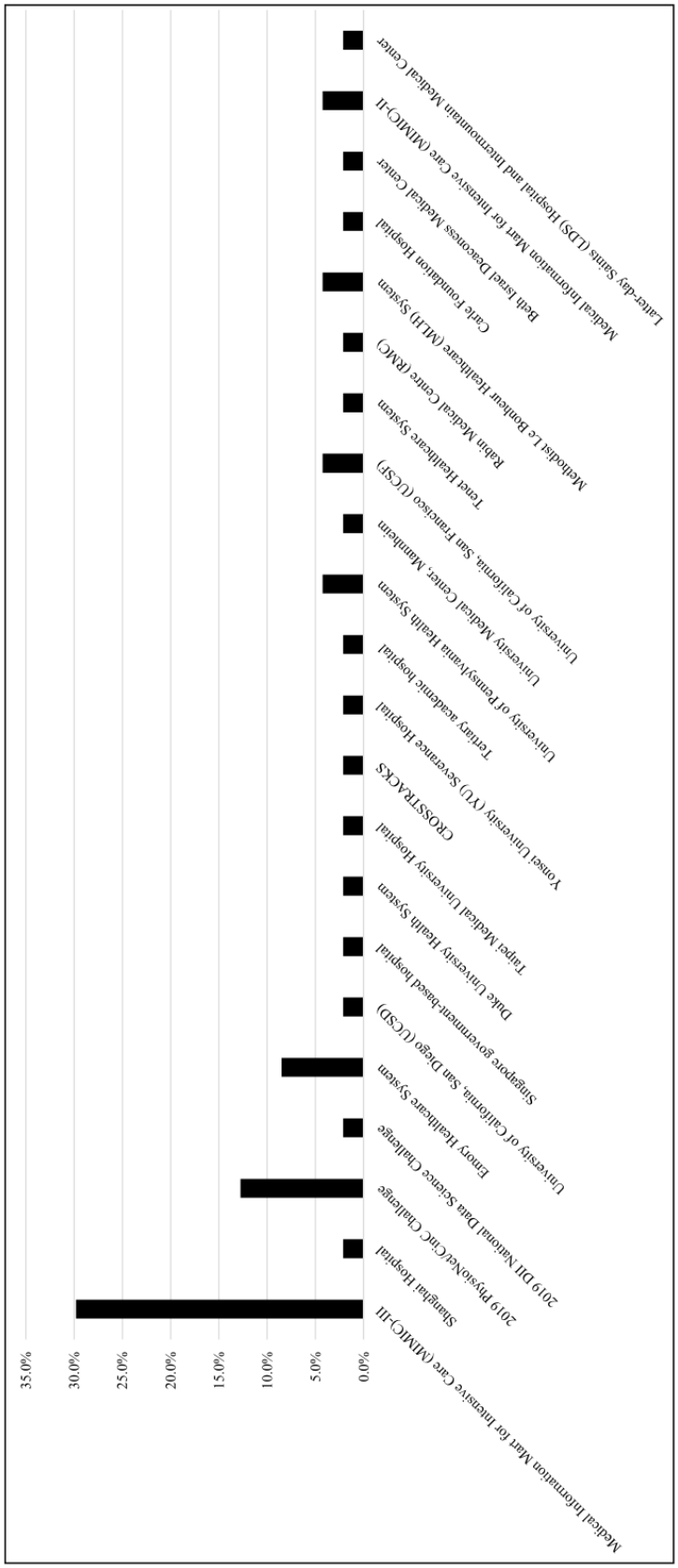


Figure 2.1: Distribution of data sources to develop sepsis detection models

[14]

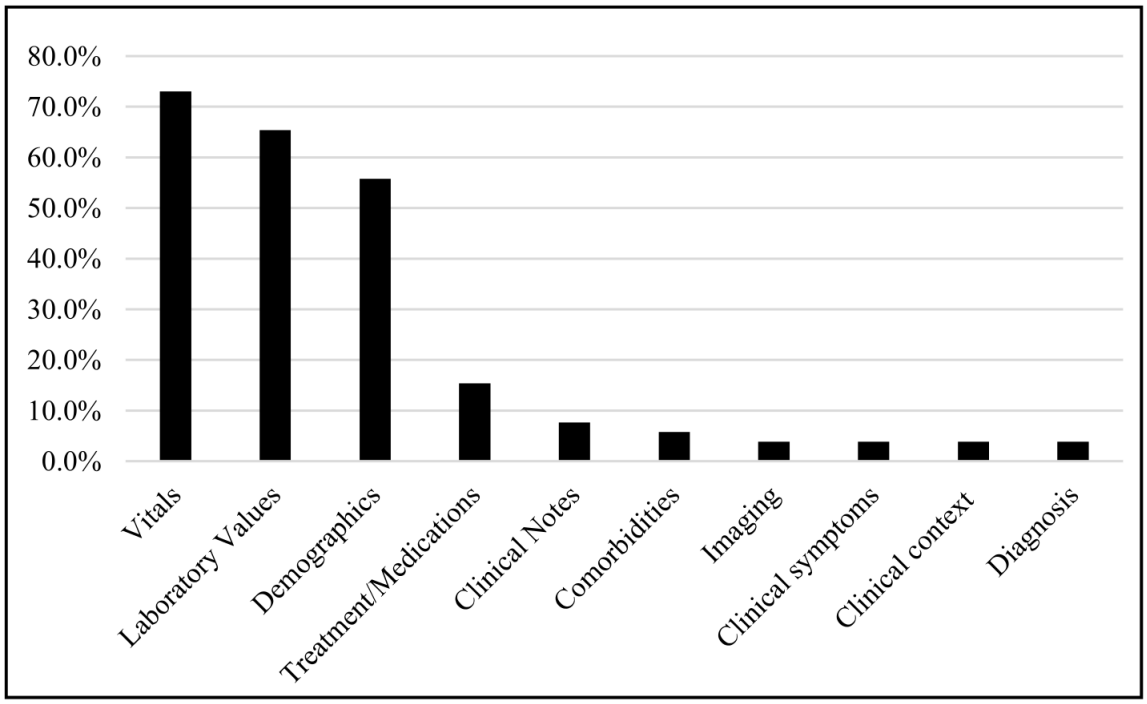


Figure 2.2: Types of input features used across studies on sepsis diagnosis

[14]

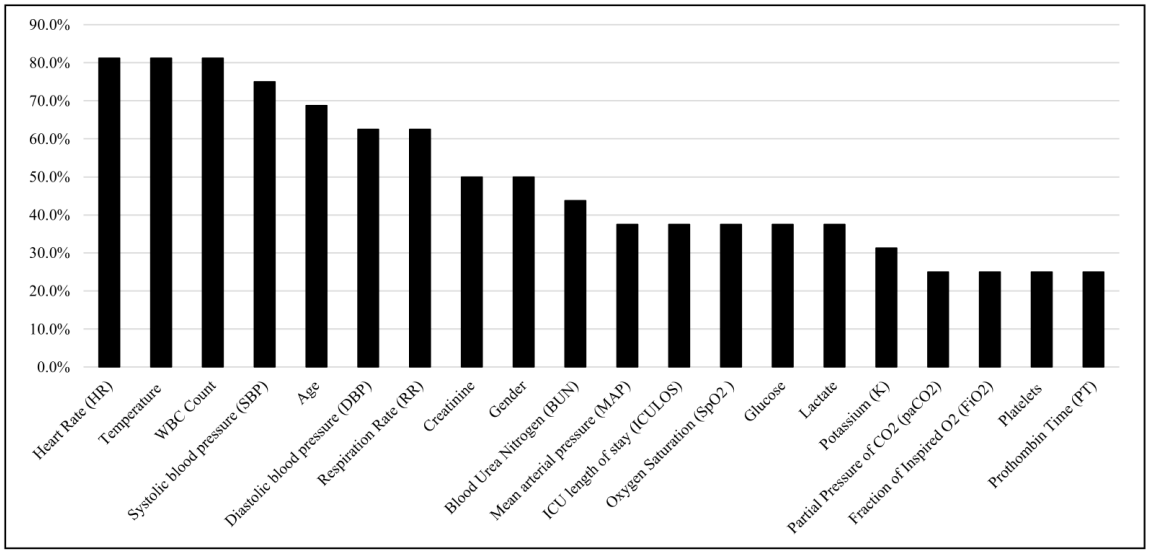


Figure 2.3: Top-ranked 20 input features among studies on sepsis detection

[14]

2.2.1 Classical Machine Learning Approaches

The study conducted by He et al. [15] on sepsis detection made use of the dataset provided by Physionet challenge 2019. All the features in the dataset were used; vital signs which included heart rate, body temperature, different blood pressures (Systolic and diastolic blood pressure, mean arterial pressure), oxygen saturation level etc., laboratory values

such as blood glucose, creatinine, pH level etc. and demographic information such as patient age and gender. SIRS and SOFA scores were also calculated from the raw feature values. The missing values were imputed via forward filling which is a logical approach in electronic health records. To tackle the imbalance problem, the authors randomly downsampled the majority non-septic class to keep the instances in both classes the same. Since, this is a time-series forecasting type problem, Long Short Term Memory (LSTM) based features were additionally used as input to their Extreme Gradient Boost (XGBoost) model. The proposed algorithm achieved a normalized utility score of 0.313 on the full hidden test set.

The work by Liu et al. [16] used tree-based classifiers (gradient boosting and random forest) and LSTM. All the 40 features from the Physionet challenge 2019 were used and the target label was shifted ahead by 6 hours with a target to predict sepsis 12 hours early. The missing values were filled via Last Observation Carried Forward (LOCF). However, it was seen that during testing, the missing value of a feature was imputed by its mean value from the training set. This will likely introduce noise in the data and adversely affect the model performance. The heavy class imbalance was tackled by the authors via mutual information based downsampling and Synthetic Minority Oversampling Technique (SMOTE). The F1-score and Area Under Receiver Operating Characteristics (AUROC) were around 0.36 and 0.52, respectively.

Morrill et al. [17] utilized the novel signature method to identify early case of sepsis onset on the Physionet 2019 challenge database. The authors treated the missing value by forward filling them. They also engineered domain-based features such as partial qSOFA. The significant contribution of this work was the use of path signature as an indicator to when sepsis might occur. The path signature uniquely characterizes a path via a series of summary statistics and is able to differentiate between two sequential events based on which feature change occurs first. The Light Gradient Boosting Machine (LightGBM) was the final classifier which made decisions based on the signature paths and other features. Their proposed algorithm achieved a utility score of 0.36 on the final test set, securing first position in the competition.

Patidar in his work [18] performed extensive feature engineering and focused on ratio-based features. First of all, to ensure good performance, he preprocessed the data to fill out the missing value via linear interpolation of two adjacent non-missing values. Next, out of the 40 features, he used 38 features to calculate the ratios. Afterwards, a genetic algorithm was applied to find the optimal features suitable for classification. In this process, 55 features were selected. As the classifier, he used RusBoost model to identify the septic cases. The proposed algorithm achieved a utility score of 0.309, securing 19th position in the Physionet 2019 challenge.

Singh et al. in his 2019 paper on detecting early onset of sepsis in ICU patients [19]

leveraged the power of informative missingness in electronic health record system. With an aim to diagnose sepsis 6 hours early before the actual onset to maximize the utility score as was prescribed by the Physionet 2019 challenge, the authors analyzed the septic and non-septic cases and plotted the probability of observing a feature. This led to notable distinction in a number of features between the septic and non-septic cases, specially for a window of 6 hours. The data was preprocessed as sliding window to be fed into the XGBoost model. Also, demographic variables were binned for better performance and national early warning score (NEWS) was calculated as a domain-knowledge based feature. qSOFA and SOFA could not be calculated due to the lack of required features, such as Glasgow coma scale. Missing values were not imputed and kept as not a number (NaN) because XGBoost has inherent capability of handling missing values. The final model achieved a utility score of 0.337 on the full test set.

The work by Vollmer et al. [20] devised time-specific metalearner features as predictors for sepsis in ICU patients. The authors screened the data based on ICU length-of-stay and found out that the sepsis cases increased with prolonged stay greater than 60 hours. They also categorized the vital signs based on their observation frequency, whether the measurements were taken hourly, after one hour, after two hour and so on. In the feature engineering stage, rolling window statistics along with a binary mask for missingness were calculated. Additionally, a numerical variable representing the up-to-dateness of a given value was also introduced; 0 for newly measured value, 2 for value measured 2 hours ago and so on. Other clinical scores such as SIRS, qSOFA, NEWS etc. were also calculated based on the available features. A number of models such as XGBoost, Gradient Boosting Machine (GBM) and deep learning architecture were used as stacked ensemble to perform the classification task. A normalized utility score of 0.394 on the held-out set was obtained along with an AUROC of 0.823.

Yang et al. in their proposed workflow [21] suggested a multi-feature fusion based XGBoost model which was later optimized using Bayesian optimization method. The authors first preprocessed the data to overcome the missing value problem. They discarded the features with more than 99% missing value. Then, the remaining feature values were forward filled. Extensive feature engineering was performed in the next stage where in addition to the original features, informative missingness were calculated. Here, the frequency of measurement as well as the time interval between two valid measurements were recorded. Additionally, qSOFA score was calculated and window-based rolling statistics such as mean, median, maximum, minimum value etc. were quantified. Finally, an XGBoost model was used as the classifier with Bayesian optimizer used for hyperparameter tuning. The proposed ensemble model achieved a utility score of 0.364 on the final test set.

The work by Yu et al. suggested an ensemble model of random forest (RF), XGBoost

and LightGBM for timely diagnosis of sepsis [22]. Their work was based on the dataset of Physionet challenge 2019. This work touched all the major aspects of this dataset including handling missing values, class imbalance, feature engineering and selection and deciding on a machine learning algorithm. The authors imputed the missing value by a combination of forward filling and mean value substitution. The class imbalance was addressed via easy ensemble algorithm where the majority low-risk class was split into multiple clusters and combined with the minority high-risk class so that the instances in both classes for each cluster were equal. Extensive feature engineering was also performed whereby multiple time-based features such as differential, integral, window-based statistics were evaluated. Finally, the devised features were fed to an ensemble of three tree-based models and evaluated on the final test set, resulting in a utility score of 0.282.

A comparative study among a number of machine learning models for the early diagnosis of sepsis in ICU patients was done Hsu et al. in their 2019 paper [23]. The irregularly sampled data was imputed via a spline-based algorithm. Data was preprocessed via logarithmic transformation and z-score normalization. Feature selection was performed by calculating various statistical metrics such as mean, median, kurtosis and skewness. After taking care of the missing values and feature selection, the data was fed to a variety of algorithms such as naive Bayes, Support Vector Machine (SVM), Hidden Markov Model (HMM), random forest (RF) and GBM. A novel combination of K-Nearest Neighbor (KNN) and Global Alignment Kernel (GAK) was also proposed by the authors which is known to be good for sequential data classification task. Finally, Convolutional Neural Network (CNN) and LSTM ensemble was also evaluated on the processed data to determine which model yields better predictive accuracy in terms of the proposed utility score by the Physionet 2019 challenge from where the dataset was sourced. The KNN-GAK performed poorly in terms of accuracy while the CNN-LSTM was significantly better with an accuracy of 95% and a final utility score of 0.076.

Pimentel et al. proposed an uncertainty-aware model for estimating the probability of sepsis onset in hospitalized patients based on Physionet 2019 challenge data [24]. The authors suggested a two-step framework where the model in the first step simply detects the probability of developing sepsis whereas the second model in the pipeline predicts the uncertainty of the prediction of the first model by calculating a negative log-likelihood. A number of features such as difference between the two last recorded values, maximum and minimum value of each vital sign in the preceding 12 and 24 hours were extracted from the 40 original features. For the second model, elapsed time between the current observation and the last valid observation was treated as an additional feature which reflected the measurement frequency of a feature and thus helps in determining the uncertainty of the first model which relied on these features. Two XGBoost models were chosen for the

two tasks and their hyperparameter was tuned via Bayesian optimization. The two-model approach was able to secure a utility score of 0.412 on a held-out 15% data from the full training dataset.

An ensemble of XGBoost models was used by Zabihi et al. to diagnose sepsis in critically ill patients in ICU based on the dataset of Physionet 2019 challenge [25]. The authors engineered a wide range of feature set and trained multiple models on different subsets to improve generalizability and prediction accuracy. In the feature engineering phase, multiple window-based features such as mean, median, minimum, maximum, different quantiles, Shannon entropy, energy in the last 5 and 11 hours were calculated. To handle the missing value problem, different approaches were taken based on the rate of missingness and observation length. If all feature values were missing, mean was subtituted. For less than 70% missing rate, linear interpolation was used for imputation. Random undersampling was done on the majority class to balance the two classes. A wrapper feature selection based on XGBoost was used for the best-performing features selection and hyperparameter tuning. The ensemble of the XGBoost models was able to obtain a utility score of 0.339 on the final test set.

Sweely et al. leveraged the effectiveness of the tree-based classifier to build a boosted random forest classifier for early prediction of sepsis in hospital settings [26]. To handle the severe missing rate in the original database, the authors first attempted to find the most relevant features based on medical findings and selected 24 features. However, the reduced feature set resulted in poorer performance compared to the full set of original features. Last observation was carried forward (LOCF) for missing values and in case of all feature values missing, zero was imputed. The high class imbalance was accounted for via creating subsets from the original datasets which were balanced. One of the major contributions according to the authors is the incorporation of 8 hours of previous data along with the current timestamp and zero-padding if not enough rows are available. Based on the literature review, the authors inspected bagging ensemble, Adaptive Boost (AdaBoost) and Random Under Sampling Boost (RUSBoost) classifiers and chose the final one based on the algorithm nature and utility score. Hyperparameter tuning was done via grid search. The proposed pipeline managed to receive a utility score of 0.314 on the final set.

Unlike other existing research on using machine learning methods to predict the risk of sepsis in hospitalized patients, the method proposed by Murugesan et al. mainly focused on the reasoning behind the working of their model [27]. They used a XGBoost model for the classification task and interpreted it using Tree Shapley Additive Explanations (TreeSHAP) algorithm which is good at explaining the predictions. The beeswarm plot produced by the model depicted magnesium level in blood and heart rate as the two topmost features which showed a positive correlation with the outcome, higher value indicates higher risk of developing sepsis. However, other than the explainability part,

there was no significant contribution of the work. Moreover, the insights provided are dependent on the model used.

Another study by Doggart et al. [28] used RUSBoost algorithm for timely diagnosis of sepsis in ICU patients. Their work relied on data preprocessing by forward filling of missing values, class imbalance handling via random undersampling of the majority class and finally, employing a boosted trees for the final classification task. The model achieved a normalized utility score of 0.267 in the unofficial phase of the Physionet 2019 challenge. This approach, however, did not include feature engineering which is essential for such disease classification scenario in sparse Electronic Health Record (EHR).

The authors of [29] used a LightGBM model to assess the risk of developing sepsis in ICU patients. This work relied heavily on data preprocessing to reveal the underlying pattern of this complex time series data with lots of missing records. The missing values were replaced by the mean of two consecutive readings. The remaining missing values were first backward filled and then forward filled. However, the idea of backward filling beats the purpose of predicting sepsis in future in a casual manner, using only past values. Sliding window-based mean and median were calculated as well in an attempt to reveal the short-term and long-term dependencies. Data were binned for some feature with an aim to reduce feature variance. In the classification phase, two approaches were adopted. One was Weibull Time-To-Event (TTE) which tried to estimate the underlying assumed Weibull distribution. However, this method showed poor performance for reasons unknown. The second method was LightGBM. The boosting algorithm obtained a utility score of 0.005 on the final test set.

The effectiveness of the ensemble method was demonstrated again in the work of Du et al. [30] where gradient boosted decision trees were used to estimate the probability of sepsis occurring. Log transform was performed on the original features. Additionally, missing value flags to indicate whether a feature value is present or not, difference value of vital signs with the previous step and variance of vital signs over the last 6 hours were calculated to include in the feature vector. Data was normalized and then forward filled for missing values and zero imputed for the initial missing values. Finally, an XGBoost classifier was used for prediction and it obtained a utility score of 0.345 on the final test set.

Hammoud et al. in their 2019 work on early prediction of sepsis in ICU patients based on the Physionet 2019 challenge data [31] used an ensemble of gradient boosting decision trees along with optimal sample weighting. They performed forward filling on the missing values and proved a theorem that states that maximizing a general utility function is equivalent to minimizing a weighted 0-1 loss. Based on this statement, they put forward a customized loss function for model training. The gradient boosting trees were trained on a window of 20 hours of measurement. The trained model got a utility score of 0.332

on the full test set. Despite the good performance of this approach, there was no mention of class imbalance handling. Moreover, the 20-hour window requirement is a downside of this method because the patient condition might deteriorate during this long period.

A combination of sepsis screening tools followed by a KNN was suggested by Biglarbeigi et al. in their 2019 work [32]. The authors utilized prehospital early sepsis detection (PRESEP), SIRS and SOFA to screen variables for efficient sepsis prediction. The missing values were handled by first categorizing them into two classes and imputing the last observed value for one class and mean for the other class. Undersampling was the method of choice in this work to solve the problem of class imbalance. Finally, a KNN was used to predict sepsis and achieved a utility score of 0.07 on a held-out subset of the training data.

In their 2019 paper, Noorzadeh et al. [33] applied HMM in conjunction with logistic regression (LR) for early diagnosis of septic patients. However, to tackle the problem of independence assumption of features, they also used conditional random field (CRF) and compared the findings. To reduce the computational complexity, they only selected 17 out of the total 40 features based on a p-value threshold of 0.1. Their CRF model outperformed the HMM in terms of utility score of 0.19.

Similar to a lot of existing works, Nakhashi et al. [34] also used a RF classifier, which is an ensemble technique, to assess the probability of developing sepsis in ICU patients. Given the sparsity of the dataset of Physionet challenge 2019, the authors substituted mean values calculated from the dataset for the missing features for better outcome. However, given the age, gender and patient condition, the mean value imputation might not be the best option and can introduce noise. Outlier removal was the next step in the data cleaning to avoid erroneous results. Based on the recording length, two separate RF classifiers were used for the final prediction. The suggested workflow was able to secure a utility score of 0.142 on the full test set.

A framework was put forth by Krissaane et al. [35] to predict sepsis in a hospital settings that consisted of a combination of feature selection by tree-based method and anomaly detection by semi-supervised method. In the first phase of the pipeline, features with more than 80% missing value were discarded. However, this step might result in loss of information because although these features have high sparsity, they contain valuable information as to the onset of sepsis. Some of the features such as systolic and diastolic blood pressure were highly correlated with the mean arterial pressure and hence, dropped from the input feature vector. The missing values were forward filled. A RF classifier was used to find the Gini index based feature importance for selecting relevant features. In addition to that, an autoencoder (AE) was used to efficiently represent the input feature in lower dimensions to reduce overfitting. Finally, the selected features and the lower dimension representation of the AE was used as input to a LR model which was the final

classifier. The LR got a utility score of 0.177 on the held-out test set.

Lyra et al. proposed an optimized RF to identify the onset of sepsis in clinical patients provide by Physionet 2019 challenge data [36]. All 40 features along with the mean and standard deviation with the previous timestamp was used as input to the model, resulting in a total of 120 features. Feature importance of these features were evaluated by first replacing each feature by NaN calculating the Normalized Observed Utility (NOU). To overcome the huge class imbalance problem, 12 hours before and 3 hours after the actual onset of sepsis were marked as positive because the utility score of the challenge rewards model within this time range. In addition to this, trees of the RFs were trained only on the positive classes and a subset of equal number of negative instances to keep the training balanced. The RFs were trimmed using a penalization parameter to reduce overfitting and improve the generalizability of the model. The final model achieved a utility score of 0.296 on the hidden test set.

Gao et al. used a variety of ML models for sepsis diagnosis in MIMIC-IV database [37] out of which, RF was the most effective. Features were selected based on literature review and clinical expertise. Null values were dropped and class imbalance was handled via SMOTE. The RF achieved an AUROC of 0.94 on the test set along with an accuracy of 86.31% and sensitivity of 83.71%. The work also explored SHAP to interpret the prediction which showed that coma score and average urine output were the top two features for prediction.

The work by Firoozabadi et al. on detecting sepsis early in ICU patients [38] was based on an ensemble model of bagged decision trees for predicting sepsis timely in ICU patients. Their work relied extensively on data preprocessing where a plausibility filter was applied first which discarded outliers based on the actual statistical distribution and domain knowledge. Afterwards, missing data were forward filled and still remaining missing values were substituted with the corresponding mean. In the next step, numerical feature values were standardized. An ensemble of 100 bagged decision trees was utilized in the next step for hyperparameter tuning and optimal feature selection which resulted in 15 selected features. These features were then used for the utility score calculation on the final test set which resulted in a utility score of 0.24. This work emphasizes the simple but logical pipeline of data preprocessing with the strength of the ensemble of decision trees.

Deogire suggested a low-dimensional algorithm for detecting sepsis in clinical patients based on Physionet 2019 challenge data [39]. This work does not include any specific method for handling the missing data and the class imbalance. It rather assigns a weight to specific parameters selected based on sepsis-2 and sepsis-3 definitions. Also, the criteria for assigning weights were not mentioned clearly. The authors simply put associated higher weights to vital organs and lower weights to the features that they claimed to be routine measurements such as blood pressure and respiratory rate. Based on a simple

if-else condition where the total weight exceeds a certain threshold, the author predicted sepsis. This method achieved a utility score of 0.013 on the final test set. Despite its minimality, this work shows that even the simplest model with some domain knowledge can predict sepsis correctly to some extent.

A classification framework was developed by Pawar et al. in their 2019 work [40] where comprehensive feature extraction was performed prior to feeding them to a LR model for the final prediction of sepsis in ICU patients. The authors worked on the 40 features dataset of Physionet 2019 challenge and generated features of clinical interest such as ratio between different features. The authors also transformed the variables using z-transformation and dichotomized the candidate variables. The engineered time-based features included first derivative of the features as well as change from the baseline observation. The missingness problem of the dataset was handled by forward filling and substituting mean value for the still missing values. P-value was the feature selection technique where features with p-value lower than the threshold of 0.01 were chosen because they were considered significant for predicting the patient outcome. A LR model was trained and tested on the cleaned and selected variables which was able to secure a utility score of 0.249 on the full test set. This approach shows the effectiveness of proper data cleaning and data selection whereby a simple statistical regression model was able to achieve a decent utility score.

In a study by Abromavicius et al. [41], the authors developed tree-based sepsis-prediction models based on vital signs and their related features. They developed three separate models according to the length of ICU stay of patients. Additionally, the feature engineering process was also different for the three models. For short-term or short-duration model, mean values, median values along with raw features were used. For the medium and long-duration models, mean value, Shannon entropy, kurtosis and standard error were calculated. The class imbalance problem was tackled by using the ADASYN algorithm to upsample the minority class. The final models obtained a utility score of 0.014 on the full hidden test set.

Karthiga et al. [42] built an XGBoost model that flags sepsis early and report a standout accuracy of 95.01 % using a set of time-stamped vital-sign and laboratory variables. They concluded that such a carefully engineered boosted-tree pipeline already provided a credible early-warning tool for clinicians, and suggested that further feature selection or deep-learning refinements could push performance even higher in future iterations.

The authors of [43] proposed a novel rule-based model called RE-MOBDE to predict sepsis early while ensuring the model is interpretable for clinicians. Unlike black-box models, RE-MOBDE extracts human-readable rules using a multi-objective differential evolution algorithm that balances accuracy with simplicity. It was tested on the large-scale 2019 PhysioNet/CinC Challenge dataset and achieved AUC scores up to 0.82, with fewer

than 90 rules of average length under 2, making it suitable for real clinical use. The model outperformed several traditional scoring systems and machine learning baselines in both accuracy and interpretability. Overall, the study demonstrates that interpretable models can provide clinicians with trustworthy, accurate decision support for sepsis diagnosis.

Smitha et al. [44] explored the use of machine learning models to predict sepsis early using data from the PhysioNet 2019 Challenge, which includes real ICU patient records. The authors tested various algorithms—Random Forest, Logistic Regression, and XGBoost—to identify septic and non-septic cases based on time-series vitals and clinical variables. Among them, XGBoost outperformed others, achieving the highest AUROC and accuracy scores, indicating its strength in handling complex, nonlinear relationships. They emphasized the importance of feature selection, noting that heart rate, temperature, and MAP were among the most significant predictors. The paper concludes that machine learning, particularly ensemble methods, can enhance early diagnosis and clinical decision-making in sepsis management.

Ewig et al. [45] proposed a multi-subset machine learning approach to improve early detection of sepsis using temporal trends in patient data. The dataset was divided into distinct subsets based on septic stages, allowing the model to learn unique progression patterns at different time points. Machine learning algorithms such as decision trees and logistic regression were trained separately on these subsets and then integrated. The system achieved promising results, highlighting how segmenting the data temporally enhanced predictive accuracy compared to a single-model approach. The authors concluded that incorporating temporal change trends and patient heterogeneity can significantly improve early sepsis prediction in real-world clinical settings.

2.2.2 Deep-Learning Based Approaches

Liu et al. [46], in their work on early diagnosis of sepsis also used all the raw 40 features of the Physionet challenge 2019 database. The authors didn't mention any technique as to how they handled class imbalance. In the data preprocessing stage, a fix-length sliding window of 1-hour length was used and previous values were zero filled. To effectively capture the interaction among different events, a heterogeneous event aggregation (HEA) module was designed. Since the dimensionality required to capture all the possible interactions is vast, this work made use of attentional multihead aggregation. LSTM was used as the final classifier which achieved an overall utility score of 0.321 on the final test set.

Macias et al. [47] proposed a novel imputing method in their work to avoid the problem of missing records in the dataset of Physionet challenge 2019. Their method imputes the vital signs using second-order interpolation and imputes the more frequently missing lab

values via averaging over a window of N hours. They also used the AE representations of the raw values as additional input to their LSTM model. The authors didn't mention any approach to handle the class imbalance. The model achieved an overall utility score of 0.241 on the full test set and obtained an AUROC of 0.788.

In the study conducted by Nejedly et al. [48], the authors tried to predict sepsis using LSTM where the hyperparameter tuning was performed using a genetic algorithm. As was seen in the previously mentioned works above, this work also made use of all the 40 features. They treated the missing values in two ways: converting them to -1 and forward filling. The forward filling method worked well for their case. Their proposed LSTM architecture consisted of 100 neurons in the hidden layer, followed by a fully connected layer and dropout layer for generalizability. The utility score on the final full test set was 0.278.

The study conducted by Schellenberger et al. [49] demonstrated the distinguishing capability of extensive feature engineering. The authors of this work took 38 out of the total 40 features of the Physionet 2019 challenge database and engineered those features. They kept the raw values of 38 features and in addition to that, calculated the difference value with the most recent non-missing feature value in the past, giving rise to another 38 difference features. Moreover, another 38 features were chosen as the last observed reliable, that is non-missing, value. Partial qSOFA score was calculated because all the features necessary for calculation of qSOFA were not available. Difference of partial qSOFA score in a 24-hour window was calculated as well. The missing values were filled with zero. After this exhaustive feature engineering, they fed them into a custom LSTM architecture. They used different thresholds for 5-fold cross validation to achieve the best one for utility score and the final utility score on the test set 0.29.

In the study conducted in 2019 by Vicar et al. [50] on early prediction of sepsis in ICU patients based on Physionet 2019 challenge database, the authors addressed the challenge of missing values by feature normalization into fixed range along with replacement from outside the normalized range. This allowed the model to include missing values in the learning process. To overcome the issue of extreme class imbalance, the authors used the generalized dice loss function which can adjust the weights of the two classes automatically. A custom LSTM architecture was used as the classifier and the threshold was adjusted via grid search with a focus on maximizing the utility score on the validation set. The final model was able to achieve a utility score of 0.281 on the full test set.

Tran et al. in their work [51] proposed a combination of LSTM autoencoder and Fully Connected Neural Network (FCNN) for early diagnosis of sepsis on the Physionet 2019 challenge dataset. LSTM AE was used to reduce the dimensionality of the data while extracting meaningful features while the FCNN was used as the actual classifier which took the extracted features by the AE and output probability of observing sepsis. The

authors further strengthened the performance of their devised model via ensemble with gradient boosting decision trees and random forest. The mean value for each feature was substituted for the corresponding missing values. The final model achieved a utility score of 0.284 on the test set.

Li et al. suggested an ensemble of CNN and LSTM for early prediction of sepsis in ICU patients [52]. The authors performed upsampling so that the ratio of the positive and negative classes was unity and thus, they handled the class imbalance. They also performed feature selection based on feature importance of a random forest trained on a subset of the training data and selected 11 features out of the total 40 features. Data was also normalized using the global minimum and maximum value of a specific feature. Their proposed CNN architecture took as input 11 hours of data, each hour consisting of 11 features, resulting in a 11-by-11 2-dimensional matrix. The LSTM took as input 12 hours of sequential data, each hour consisting of all the 40 features. The bagging ensemble of these two models achieved a utility score of 0.237 on the full test set.

Roussel et al. proposed a novel architecture of LSTM for early prediction of sepsis in hospitalized patients [53]. Their approach consisted of the main task of sepsis prediction and an auxiliary task of vital sign prediction at a timestamp of 6 hours in the future from current timestamp. The auxiliary task was incorporated to achieve good performance in terms of the utility score which was reflected in a reasonably high score of 0.309 in the full hidden set. Data was preprocessed before feeding them to the classifier by normalizing the values between 0 and 1 and by replacing the missing values by -1. Binary mask was also used as indicator of missingness of a specific feature in this work.

The study by Narayanaswamy et al. [54] on sepsis prediction in ICU patients suggested a framework consisting of feature engineering, processing and predictive classifier. In the feature engineering phase, a synthetic risk feature was created which tracked the presence or absence of a feature to indicate whether a patient is at risk of developing sepsis. However, this method suffers from the fact that it has to depend on the recording of other features for detecting sepsis rather than taking into account the actual feature values themselves. After the feature engineering step, the features were processed via min-max normalization by defining a local baseline as minimum and maximum and a threshold value for lab data. Finally, an LSTM was used as the final classifier which achieved a utility score of 0.179 on the full hidden test set.

Lee et al. suggested a workflow in their 2019 paper [55] to diagnose sepsis that consisted of missing value imputation and transformation, regularization and training process manipulation. The authors resorted to a number of missing value imputation techniques such as forward filling, binary masking, difference feature, duration of missing and decay to default value methods. A cross-entropy function along with a reconstruction loss was used in the training process, the latter was intended to prevent overfitting.

Minority class oversampling together with population-based resampling was used to handle class imbalance. The population-based sampling was based on the observation that certain portion of the training data deteriorated performance, thus, necessitated the parallel training on multiple subsets of the training data. A stacked ensemble of transformer and LSTM model was used for the final prediction task that obtained a utility score of 0.291 on the final test set.

Unlike other works, the authors of [56] relied on an unsupervised framework for predicting the onset of sepsis. In the preprocessing stage, the authors forward filled the feature values where the rate of missing was more than 90%. A spatial AE was used to capture the spatial information and then, a temporal autoencoder (TAE) was stacked on top of it. The TAE consisted of LSTM celles and is good at learning the temporal dependency of the input in a lower dimension. Although the model achieved a negative utility score on the full test set, it explored the possibility of using unsupervised methods in a supervised learning problem to gain insights that might aid the actual prediction task.

A comparative analysis of a number of machine learning models was performed by Sarafrazi et al. in their 2019 paper on sepsis detection in critically ill patients in clinic [57]. The authors solved the issue of missing data by forward filling and afterwards, filling the still missing value with typical value for healthy people. Class imbalance was dealt with by anomaly detection technique where a LSTM AE was used for the purpose. Finally, CNN, a combination of CNN and LSTM, a combination of neural network and XGBoost was used to evaluate the performance on the processed data and it obtained a utility score of 0.22.

Echo State Network (ESN), a variation of the RNN family was employed by Alfaras et al. in their 2019 work on dianosing sepsis in critically ill patients admitted into ICU [58]. Considering the high percentage of missing value in the Physionet 2019 dataset, the authors mainly focused on imputing the missing values using a number of techniques such as constant, mean, random imputation assuming normal distribution and by interpolation. The ESN used by the authors of this work basically leverages the reservoir computing properties of mapping input data into a higher dimensional space and focusing on training only the output layer weights. Comparison between the proposed model with the conventional classifiers were shown and also, their different combinations were tested. However, the tree-based classifiers alone performed better than the suggested model. The ESN model put forth by the authors achieved a utility score of 0.188 on the final test set.

Nonaka et al. used stacked Gated Recurrent Unit (GRU) for early diagnosis of sepsis in ICU patients [59]. The authors preprocessed the data in a number of steps. They first removed the outliers and clamped features with very large variation from the average. After that, they calculated logarithm for a number of features and then, standardized all the

features. Missing values were zero filled and a binary mask variable was created to keep track of whether a feature is missing or not at a timestamp. One of the main contributions of this paper was the initialization of the GRU using demographic information. Additionally, they also proposed a custom loss function based on the provided utility score function. Finally, a stacked model was used as the final classifier which achieved a utility score of 0.323 on the final test set.

Wickramaratne et al. proposed a bidirectional GRU for early assessment of sepsis progression in their 2019 work [60]. The authors preprocessed the data by padding the sequences up to the longest available sequence. Their model made use of all the available features including vital signs, laboratory measurements and demographic information such as age and gender to make prediction about the onset of sepsis. The proposed algorithm achieved an AUROC of 0.97 with an accuracy of 99.8%. However, no mention of the obtained utility score or the recall rate were mentioned which are critical for such disease diagnosis scenario where reducing false negative is one of the topmost priorities.

In another study by Chang et al. in 2019 on the Physionet 2019 challenge [61], the authors put forward a solution to address the problem of timely diagnosis of sepsis that leverages multi-task imputation followed by a temporal convolutional neural network (TCN). The imputation technique consisted of a history-based estimation and feature-based estimation. This history-based approximation predicts missing values based on past values of a feature and the feature-based approximation predicts missing values based on the values of other features at that timestamp. No mention of handling class imbalance is found in this work which is a limitation of the work. After imputing missing values, the features were fed to a TCN for prediction. The TCN was trained with an objective of optimizing an approximation of the utility function provided in the challenge. The final model achieved a utility score of 0.328 on the final test set.

Gilbertson et al. proposed an artificial neural network for detecting sepsis in ICU patients in their 2019 work [62]. The authors used forward filling method to get rid of the missing values. Still remaining missing values were then zero imputed. To reduce the dimension of the input features and improve genralizability, principal component analysis (PCA) was used and 10 axes were selected. In addition to that, modified qSOFA score was calculated based on the systolic blood pressure and respiratory rate. It was called modified qSOFA score because Glasgow coma scale, which was also required for full qSOFA computation, was not available in the dataset. A neural network with two hidden layers was used as the classifier which was able to secure a utility score of 0.022 on the full test set.

The work by Aguet et al. [63] extensively explored the domain of exploiting the information of missing pattern in the data to predict the onset of sepsis. In the first step of the data preprocessing, relevant features were selected based on sepsis-3 definition and

by discarding features which have a missing rate of more than 90%. Next, they used these selected 18 features for sepsis prediction in two methods. In the first method, the missing values were simply forward filled and in the second method, typical normal value was substituted for a missing value. Also, zero imputation was used in yet another approach. A binary mask feature was created which represented whether a specific feature is available or not. All these features were then fed to a RNN consisting of two stacked GRU and achieved a utility score of 0 on the final test dataset.

The study [64] presents DEWS, a real-time sepsis warning system developed for pediatric intensive care units using machine learning. By continuously analyzing vital signs like heart rate, blood pressure, and temperature, DEWS could detect sepsis an average of 5.8 hours earlier than traditional clinical recognition. The system was trained on over 11,000 ICU admissions and achieved an AUROC of 0.851, showing strong predictive performance. Importantly, DEWS operates in real time and is interpretable, making it more suitable for clinical use compared to black-box models. The authors emphasized its potential to support timely intervention and reduce sepsis-related mortality among critically ill children.

2.2.3 Hybrid Approaches

Pou-Prom et al. utilized a CNN, a RF and combined these using a LR model to create an ensemble that was used for timely diagnosis of sepsis in hospitalized patients [65]. The authors first applied preprocessing to the data to account for the missing value and significant class imbalance. The missing values were forward filled and remaining missing features were imputed with -1. A binary mask feature for each of the existing feature was also created. This mask feature was set to zero when the corresponding value was not observed or missing and was set to one when the feature value exists. After that, a rolling window of 24 hours was created to feed the data to CNN later on in the pipeline. The data imbalance was tackled via random sampling and cluster-based sampling after projecting the data into lower dimensions via principal component analysis. Finally, an ensemble of CNN, RF and LR was used on the preprocessed data which received a utility score of 0.266 on the final test set. Despite the good performance based on the utility score, the limitation of this work was that the model requires 24-hour data to predict sepsis which is a long timeframe.

Li et al. proposed a series of models based on ICU length of stay in their work on Physionet 2019 challenge data [66]. The main contribution of this work was the introduction of a time-phased model that utilizes different sub-models based on the ICULOS feature of the database. Based on exploratory data analysis, the authors used different data imputation methods to handle the missing data. In each of the sub-models,

feature engineering was performed to construct window-based feature, ratio features, domain knowledge based features such as qSOFA. In the late stage of ICU stay, a RNN was leveraged to capture the long-term dependency of the input features for better sepsis prediction. The ensemble of the two tree-based methods for early and middle stages and the RNN for the late stage achieved a utility score of 0.337 on the full test set.

Scherpf et al. implemented a combination of RNN and LightGBM in their 2019 work for detecting the onset of sepsis in ICU patients [67]. The idea behind their work was to derive the time-based features from RNN output and feed them with some additional hand-crafted window features such as mean, minimum, maximum, standard deviation and so on to the subsequent LightGBM. Separate preprocessing steps were performed for the two classifiers. For the RNN, three classes were derived from two classes namely sepsis, non-sepsis hour of septic patients and non-sepsis hour of non-septic patients. Z-normalization was performed for all input features and missing values were replaced by LOCF method. The RNN was trained on a 20-hour window and hence suffered when sepsis occurred early in the dataset. Therefore, to improve on this point, a tree-based classifier was stacked on the RNN output which could potentially emphasize short-term feature in the final prediction in addition to the long-term dependencies provided by the RNN outputs. This algorithm achieved a utility score of 0.114 on the full hidden test set.

LSTM and XGBoost models were used by Wang et al. in their 2019 paper on early detection of sepsis in ICU-admitted patients [68]. In order to handle the large proportion of missing data in the database of Physionet challenge 2019, the authors used LOCF method to fill in the missing values. After that, the remaining missing values were replaced with 0. However, due to the capability of XGBoost model to deal with missing data, the zero-filling data preprocessing step was skipped for the XGBoost model. With an objective to predict sepsis 6 hours early before the actual onset of the disease, this work devised new difference features by subtracting the 6th and 7th hour past values for all the features. Additionally, qSOFA score and the associated individual scores were also calculated. This gave rise to a total of 82 features, all of which were used to train the models. The XGBoost model achieved a higher utility score of 0.313 on the final test set.

The study [69] introduced a novel framework that combines contrastive learning and prototype-based classification to detect sepsis early from patient time-series data. It uses self-supervised learning to generate useful patient representations without requiring labeled data, which is often limited in healthcare. The model achieved better performance than previous supervised approaches, particularly in distinguishing septic and non-septic cases during early ICU hours. By learning with both positive and negative pairs and using prototypes to define class boundaries, the system became more robust and generalizable. This approach shows promise for real-world deployment, where data scarcity and early diagnosis are major challenges.

2.2.4 Hardware-Oriented Approaches

A new approach was suggested by Athanasiou et al. to predict sepsis in ICU-admitted patients in their 2019 work [70]. The authors devised a memristor-based model where the feature values were converted into voltage. The sequence of voltages over time is reflected into the resistance value of the memristor, which stands for memory resistor. The simple Pershin Di Ventra memristor model was used in this work. Many inference units were used to take advantage of the heterogeneity of the structure in order to improve prediction accuracy. The final model achieved a utility score of 0.20 on the full test set.

Giordano et al. [71] developed a low-power sepsis alert system leveraging signals from a number of wearable biomarkers which include Photoplethysmography (PPG) measurements, body temperature, heart rate, oxygen saturation level etc. This work focused on reducing the latency and energy consumption while maintaining the sensitivity and specificity of the model. The achieved sensitivity and specificity of the proposed model were 0.60 and 0.48, respectively, on the HiRID dataset (consisting of 34000 patients). However, the authors did not validate their work using the utility score (provided by Physionet challenge 2019) and AUROC, both of which are important performance indicators for effective sepsis diagnosis. Moreover, this work lacks the underlying interpretability of the deployed model.

2.3 Comparative Summary

The heavy class imbalance was treated in a number of ways by the authors of these papers. Random undersampling was observed in the works done by Patidar [18] and Schellenberger et al. [49]. A well-known oversampling technique called SMOTE was used by the authors of the work [16]. The authors of [50] introduced a specialized dice loss function to deal with the class imbalance. The Physionet database for sepsis posed another challenge regarding missing data. This missingness has to be treated properly for the downstream model to perform satisfactorily. The missingness barrier was treated in a number of ways such as forward filling [15, 16, 21, 48, 68, 17], mean imputation [72], linear interpolation [33].

The previous studies discussed so far used a wide variety of input features. The existing works employed both classical machine learning models and deep learning models. GRU was used as the classifier in the works [60, 59]. A significant portion of the works [15, 46, 48, 49, 50] used LSTM considering the time-series nature of the problem. Works by E Macias et al. [47] utilized AE outputs and Sven et al. [68] used difference features in addition to the original features for input to the downstream ML model. Gradient boost model and its variants were used in a number of works

[72, 68, 22, 73, 74]. Some of these researches also relied on the raw features along with the domain knowledge-based features (SOFA, qSOFA, SIRS, NEWS), time-dependent features, moving window statistics, difference features.

Model interpretability is almost an unexplored topic in the field of sepsis detection with machine learning models. Yang et al. [75] attempted to develop an interpretable XGBoost model for sepsis screening. They used beeswarm plot of SHAP values to gain insight into the model's working and to a limited extent. Hu et al. [76], Nesaragi et al. [77] and Jiang et al. [78] also resorted to SHAP values and associated plots to explain how their models work. However, these works lack a thorough interpretation at global as well local level. Regarding edge device implementation, we found only two previous works which are notable. Min et al. [79] created a point-of-care platform for rapid sepsis detection. Their kit consists of a disposable kit which is used for processing blood. Their focus was on detecting the level of interleukin-3 (IL-3) in blood as an indicator of sepsis. That said, a single feature is barely enough for accurate diagnosis of sepsis. Additionally, this method requires invasive investigation which might be uncomfortable for some users. Giordano et al. [71] proposed an edge device based upon ARM cortex-M processor that relies on a number of easily available biomarkers. Their solution also provides low-power advantage which is crucial for point-of-care application. However, due to the lower processing capability, such solution might not always produce the best results. Their approach also lack interpretability and access to expert care via remote healthcare solution.

Chapter 3

Methodology

This chapter outlines the methodology adopted for developing a real-time, interpretable sepsis diagnosis system based on vital signs. We begin by presenting the overall workflow of the system through illustrative flowcharts. This is followed by a detailed description of the dataset used, along with the feature engineering and selection strategies applied. Specific techniques for handling missing data and class imbalance—where we introduce a novel approach—are also discussed. Subsequently, we describe the chosen machine learning model and its architecture, followed by the integration of federated learning to enhance data privacy. Finally, to ensure interpretability, we employ SHAP and Locally Interpretable Model-agnostic Explanation (LIME) to explain the model's predictions.

3.1 Overall workflow

Figure 3.1 illustrates the complete workflow of our thesis. The data was preprocessed first which involved handling missing values using methods such as LOCF, constant imputation and no imputation for the XGBoost model. Class imbalance was also addressed via random undersampling, cost-sensitive learning, ADASYN and SMOTE. Outliners were dropped to avoid adverse effects. In the next step, differrent features such as qSOFA, NEWS along with window-based statistics were calculated. The data was then split into train and test set. Hyperparameter tuning and cross validation was performed on the train set and upon satisfactory result, the model was finally evaluated on the test set.

3.2 Dataset Description

This section details the two publicly available clinical datasets utilized for this research. The first is the PhysioNet Challenge 2019 database, which was designed for a sepsis

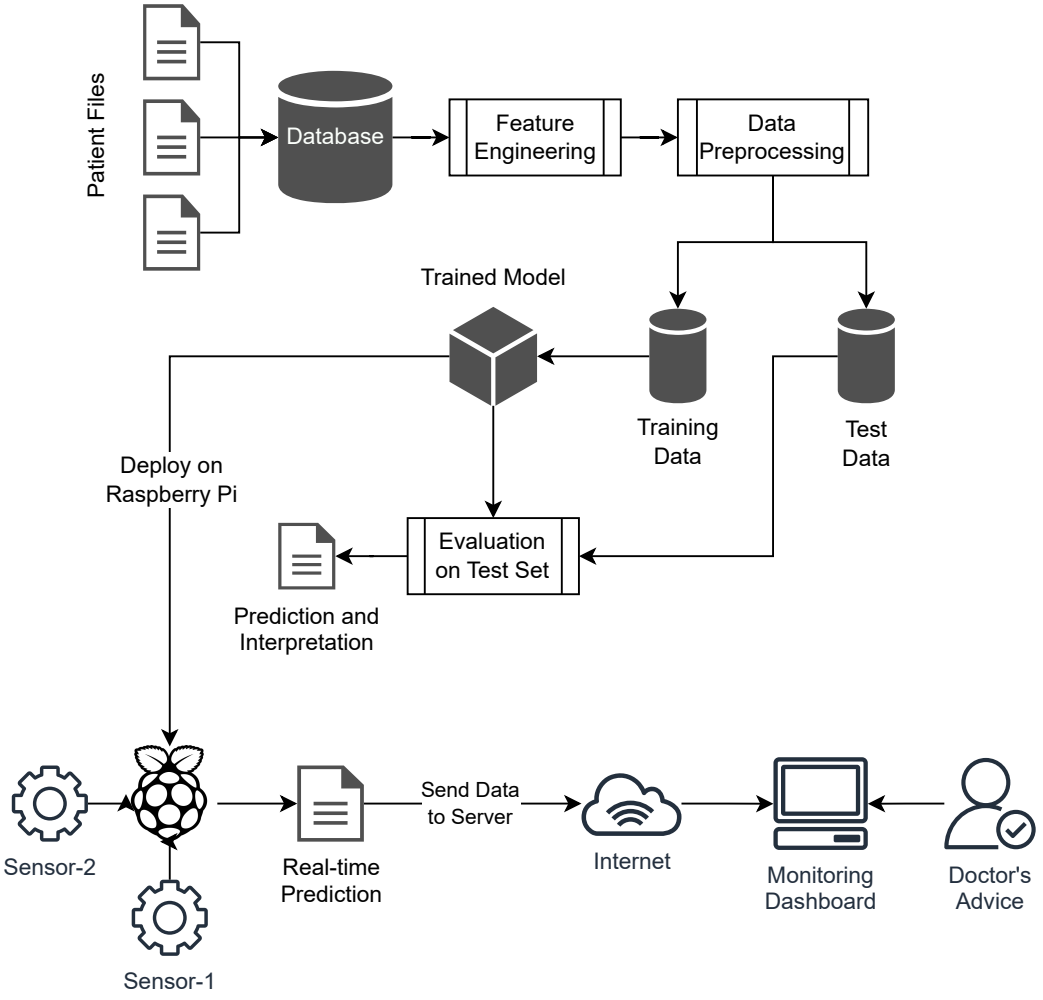


Figure 3.1: Overall workflow for sepsis detection, interpretation, edge device integration, and telemedicine-based sepsis monitoring

prediction task. The second is the MIMIC-IV Clinical Database Demo, a representative sample from a large-scale, single-center intensive care unit (ICU) database.

3.2.1 Main Dataset

Data used in this paper was taken from the Physionet 2019 challenge [13]. They were sourced from ICU patients in two separate hospital systems. Each patient file was in pipe separated value (PSV) format with a header containing the feature name. Each row in the PSV files represent one-hour worth of data. The available features consist of vital signs such as heart rate, systolic and diastolic blood pressure, body temperature and so on. Laboratory values are also found in the feature columns which include glucose, creatinine level, pH and so on. Demographic information about the patient such as age and gender

can also be obtained from the files. The dataset was heavily imbalanced. Moreover, a huge number of features, especially in the laboratory value category, were missing. The vital signs have relatively fewer number of missing values. For this paper, vital signs and their derived features were mainly utilized to diagnose sepsis correctly as early as possible.

3.2.2 Extrenal Public Dataset

We evaluated external generalizability on the publicly available MIMIC-IV clinical database (v2.2). This demo subset is intended for code development and education; therefore, it is smaller and less representative than the full MIMIC-IV cohort. We used it here to test whether our vitals-only model generalizes beyond the development data. We included adult encounters with at least one recorded vital sign among: heart rate, respiratory rate, temperature, oxygen saturation, systolic and diastolic blood pressure, and mean arterial pressure. For each encounter, time-stamped measurements were aligned to an hourly grid. If multiple values occurred within the same hour, we retained the last value in that hour. Missing values were forward filled within each ICU stay and the remaining missing values were imputed with base values for each feature. Sepsis was defined following Sepsis-3: suspected infection plus acute organ dysfunction. Suspected infection was derived using antibiotic administration and culture ordering within pre-specified time windows. Organ dysfunction was assessed using available variables and score components.

3.3 Explorator Data Analysis (EDA)

We performed a brief exploratory analysis on our dataset to characterize the data used for model development. First, class prevalence shows substantial imbalance (Sepsis ≈ [7.5%] in MIMIC and [1.8%] in PhysioNet; Figure 3.2. Second, in Figure 3.3 MIMIC shows consistently higher missingness—especially temperature (98.0% vs 65.9%) and blood-pressure fields (SBP/DBP/MAP ≈57% vs 10–30% in PhysioNet). Heart rate and respiratory rate are comparatively more complete (MIMIC 28.4% vs PhysioNet 8–14%). These gaps motivate dataset-specific imputation and caution when interpreting temperature/BP-derived features.

3.4 Feature Engineering

With a focus on model simplicity and ease of implementation at point of care, only the vital signs and demographic features (age and gender) were chosen for our model. However,

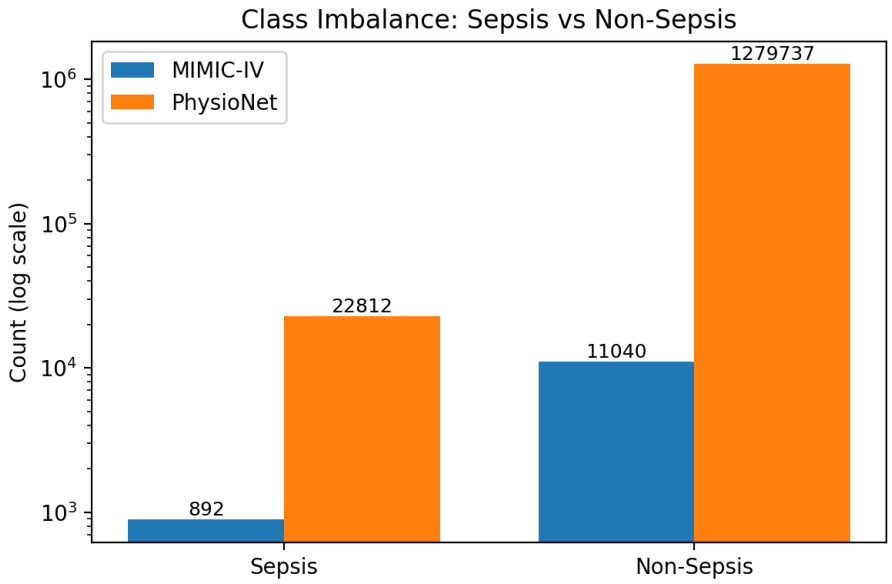


Figure 3.2: Sepsis prevalence across datasets

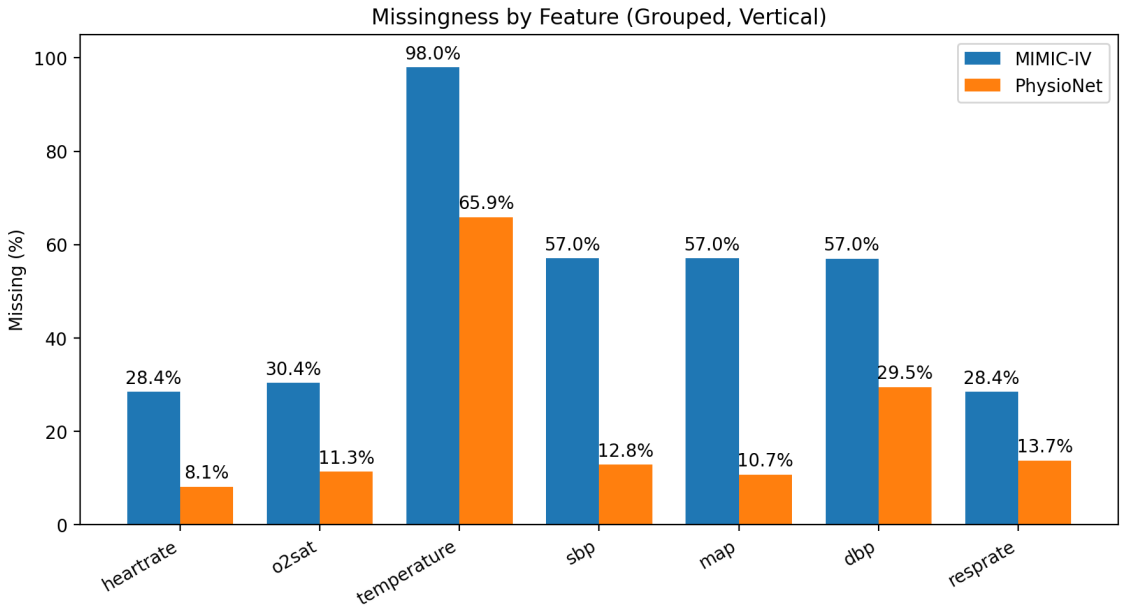


Figure 3.3: Per-feature missing percentages

considering the complexity of the task at hand with the high cost of incorrect or late prediction, these handful features were barely enough. Thus, we had to resort to heavy feature engineering. We used the following methods to devise features from the main existing vital sign features.

1. Domain knowledge-based features (qSOFA and NEWS)
2. Difference features
3. Ratio features

4. Window-based features

5. Binary mask as informative missingness

We first created some domain knowledge-based features such as partial qSOFA score and NEWS score.

Table 3.1 shows the criteria for calculating qSOFA score. If the conditions stated are true, it gives a point of 1, otherwise, 0. A higher qSOFA score is associated with higher risk of developing sepsis. The Glasgow coma scale feature was not available in our database. So, our feature was named partial qSOFA.

Table 3.1: qSOFA criteria

Systolic Blood Pressure <100mmHg	True = 1, False = 0
Respiratory rate >= 20/min	True = 1, False = 0
Glasgow coma scale <= 14	True = 1, False = 0

NEWS defines the degree of illness of a patient using six physiological findings and one observation. Our dataset had only 5 of those observations, namely, respiration rate, heart rate, oxygen saturation level, body temperature and systolic blood pressure.

Table 3.2 demonstrates the criteria for calculating the NEWS score. The total score is calculated based on the column where each feature falls into and summing them. Out of the features considered, the consciousness was missing in our dataset.

Table 3.2: National Early Warning Score (NEWS) criteria

Physiological parameter	**3**	**2**	**1**	**0**	**1**	**2**	**3**
Respiration rate (per minute)	≤8		9-11	12-20		21-24	≥25
Oxygen saturation	≤91	92-93	94-95	≥96			
Systolic blood pressure	≤90	91-100	101-110	111-219			≥220
Heart rate	≤40		41-50	51-90	91-110	111-130	≥131
Consciousness				Alert			CVPU (Confusion, Voice, Pain, Unresponsive)
Temperature (degree Celsius)	≤35.0		35.1-36.0	36.1-38.0	38.1-39.0	≥39.1	

Additionally, differences between current timestamp and previous timestamp of these vital signs were calculated. The previous timestamp variables are known as lagged variables. It was done under the assumption that sometimes, the feature value themselves

at a specific point of time may not be good indicator of the target but their difference with lag features may unfold insights and useful patterns. We also calculated the difference of qSOFA and NEWS score because sometimes, the change of scores rather than the scores themselves prove to be more useful in the long run. On a similar chain of thought, ratio between current and previous timestamp were also measured. In addition to this, ratio between features were calculated in an attempt to reveal any underlying pattern.

Furthermore, we also calculated moving average, minimum and maximum value of varying window width and different window types such as Blackman, Gaussian and so on. These moving window statistics smooth out the noises and thus enable the model to focus more on the underlying trends. Another feature that we incorporated was the expanding window based minimum, sum and average feature.

In addition to the above, we created a binary mask for end-tidal carbon dioxide feature where a value of zero means missing and one means not missing. This informative missingness could also provide useful information regarding timely detection of sepsis.

For training the CNN, we had to process the combined patient dataset as illustrated in Figure 3.4.

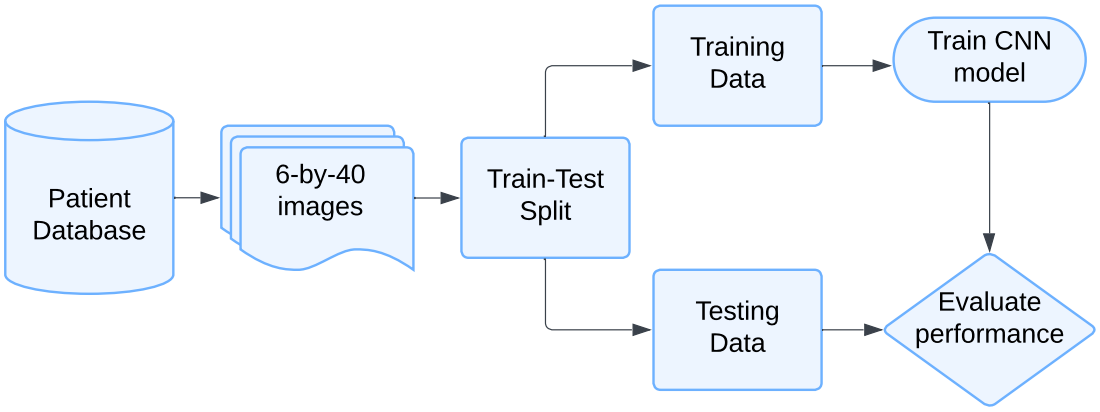


Figure 3.4: Processing patient data as images for CNN training

Here, 6 hours of continous data were extracted and the label of the final hour was taken as the label of that image-like data. Each hour had 40 features. So, it resulted in a dataset of the shape (number of samples, window size, number of features). As a consequence of the method employed, it reduced the number of positive classes to some extent. However, we still had to randomly downsample the data to handle class imbalance. Finally, a custom CNN was trained and evaluated on the testing dataset.

3.5 Feature Selection

A number of feature selection methods are available at our disposal and are implemented as Python packages. The feature selection methods employed in our work can be classified into the following categories:

1. Filter methods
2. Wrapper methods
3. Embedded methods

Firstly, we employed the simple variance threshold method for feature selection which falls under the filter methods category. It calculates the variance of the input features and removes the ones with low variance. The formula for variance is as follows:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{3.1}$$

Here, σ^2 is the population variance.

N = Total number of data points

x_i = i-th feature value of feature x

μ = Population mean

Before calculating the variance, the features are scaled to ensure that the calculated variances are comparable. The feature selection based on variance threshold is done on the basis that features with low variance may contain little information as to serve as good predictor for the target variable. So, they are removed. The variance threshold method didn't provide satisfactory result.

The next method that we leveraged in our work was done via the analysis of variance (ANOVA) method. It also comes under the hood of filter methods. It tests whether a feature gives significantly different results for the septic and non-septic classes. To do this, the overall mean (the $\bar{x}$) is first calculated for a feature among both positive and negative target classes. Then, mean is calculated for each of the classes($\bar{x}_{\text{positive class}}$ and $\bar{x}_{\text{negative class}}$). Here, n is the number of samples in the positive class and m is that of the negative class. The F-score is calculated as the ratio to distance between classes to compactness of classes. The numerator is the distance between the means of a particular feature for the target classes. It is calculated as follows:

$$num = n(\bar{x}_{\text{positive class}} - \bar{x})^2 + m(\bar{x}_{\text{negative class}} - \bar{x})^2 \tag{3.2}$$

The denominator is somewhat similar to sample variances with slight difference in its denominator. It's calculated as follows:

$$den = \frac{1}{(n-1)+(m-1)}[\sum_{i=1}^{n}(x_i - \bar{x}_{\text{positive class}})^2 + \sum_{j=1}^{m}(x_i - \bar{x}_{\text{negative class}})^2] \tag{3.3}$$

$$\text{F-score} = \frac{num}{den} \tag{3.4}$$

After the above calculations, 50 features with the highest F-scores were selected for our model. These 50 selected features proved to be the best ones for our dataset.

The next feature selection method employed in our work was the recursive feature elimination which falls under the class of wrapper methods. This is a simple model-based method yet still a powerful one. The idea behind this approach is to train a model on all the features. Then, the features are ranked based on their importance. We used a XGBoost model which is a tree-based model. The features were ranked based on their importance during splitting and training of the trees. After that, the least important features are removed. The number of features removed at each iteration is determined by the user. Then the model is re-trained on the remaining features and the process is repeated until the predefined number of features has been selected. The downside of this method is that it is computationally expensive and depends heavily on the chosen model.

Another method employed was tree-based feature importance method where the input features were ranked according to their relevance to the final prediction task. This technique belongs to embedded methods. We used XGBoost model which provides feature importance in terms of gain, weight and cover. Gain compares different features based on improvement in prediction when these features are used to split a node. Weights, on the other hand, refers to the number of times a feature appears in a tree across all trees in the overall model. Finally, cover is a measure of the relative number of samples that are affected by a feature when used for splitting trees.

3.6 Missing Data Handling

A major setback regarding working with medical data is the amount of missing data. Often times, the doctors or nurses don't record some features at some points of time probably because they are normal. Other times, some data go missing due to the inattention of the assigned medical personnel.

The performance of machine learning models are adversely affected by missing data. So, we need to take care of them. We leveraged the following strategies to handle the missingness in our dataset:

1. Last Observation Carried Forward (LOCF)
2. Constant imputation
3. MICE (Multiple Imputation by Chained Equations)

The most logical approach for missing data in medical records is last observation carried forward method, also known as forward filling. This was the method used in our work. It is justified on the basis that if a data is missing, it's most probably because its value didn't change much from the last observation. This method is also compatible for real-time diagnosis. Constant value imputation such as mean, mode, median or zero imputation is not suitable because it introduces noise into the data. Moreover, the mean, mode and median varies from person to person and therefore, we cannot find a single value that fits all scenarios. However, we may still encounter some missing value at the very beginning of recording data and in that case, forward filling method fails. In such cases, we imputed them with normal value based on age and gender. The following simple example will clarify the process.

Say, we have a dataset with patient cholesterol level, blood pressure and body temperature recorded each hour. Table 3.3 shows these values for a single patient for a span of 5 hours. Here, NaN means Not a Number which is a common practice to represent missing values.

Table 3.3: Patient data with missing values

Temperature (degree Farenheit)	Cholesterol Level (mg/dL)	Blood Pressure (mmHg)
NaN	NaN	120
NaN	240	99
99.4	NaN	NaN
NaN	190	100.5
100.2	185	125

Imputing initial missing values with typical values, such as 98.4 degree Farenheit for body temperature and 180 mg/dL for cholesterol level, results in Table 3.4:

Table 3.4: Patient data with initial imputation

Temperature (degree Farenheit)	Cholesterol Level (mg/dL)	Blood Pressure (mmHg)
98.4	180	120
98.4	240	99
99.4	NaN	NaN
NaN	190	100.5
100.2	185	125

Now, we forward fill the missing values for each feature until next valid observation is encountered. So, the final dataset with all the NaN values imputed looks as in Table 3.5:

Table 3.5: Patient data with final imputation

Temperature (degree Farenheit)	Cholesterol Level (mg/dL)	Blood Pressure (mmHg)
98.4	180	120
98.4	240	99
99.4	240	99
99.4	190	100.5
100.2	185	125

The constant imputation, although prone to noise, provides a simple way to handle missing values in a dataset. In our work, we also imputed missing data for each patient with typical values for a human being based on the patient's age and gender. The results obtained was slightly worse than the LOCF method.

MICE is an advanced statistical technique used to handle missing data by generating multiple imputations (or estimations) for missing values. The method is designed to provide more robust estimates compared to single imputation techniques by considering the uncertainty inherent in missing data. This is achieved by iteratively imputing missing values based on relationships between observed data. The core idea behind MICE is to treat the missing values as additional parameters that need to be estimated, rather than filling them with a single estimate (e.g., mean imputation). The MICE algorithm works as follows:

1. It first makes an initial assumption about the missing value of any feature.
2. In the second step, it predicts the missing value of that feature based on the values of other features as well as the initial guess of that feature.

3. The underlying predictive model keeps making guess on the missing values until the predefined number of iterations is complete or the imputed value converges.

4. Same is done for the other features with missing values.

The following example will help explain the concept of MICE imputation strategy.

Let's say that we have a small dataset of 5 patients (Table 3.6) with missing data in the "Blood Pressure" column, and we want to impute the missing values using MICE. We also have data for Age and Cholesterol Level.

Table 3.6: Patient data with missing blood pressure values

Patient ID	Age (years)	Cholesterol Level (mg/dL)	Blood Pressure (mmHg)
1	45	220	120
2	50	240	NaN
3	60	200	130
4	55	230	NaN
5	40	210	125

Where NaN denotes missing data in the Blood Pressure column.

We start by imputing the missing values using simple imputation (e.g., using the mean of the observed values for Blood Pressure). For simplicity, let's calculate the mean of the available Blood Pressure values:

Observed values: 120, 130, 125 Mean $= (120 + 130 + 125)/3 = 125$

We now replace the missing values with this initial imputation as shown in Table 3.7:

Table 3.7: Patient data with initial imputation for blood pressure values

Patient ID	Age (years)	Cholesterol Level (mg/dL)	Blood Pressure (mmHg)
1	45	220	120
2	50	240	125
3	60	200	130
4	55	230	125
5	40	210	125

Next, we perform iterative imputations, where we use the existing values to predict the missing ones.

In the second pass, we start with Patient 2 (who has a missing Blood Pressure) and predict the missing value using the other variables (Age and Cholesterol Level) via a regression model.

For simplicity, let's assume the following regression equation for Blood Pressure:

$$\text{Blood Pressure} = \alpha + \beta_1(\text{Age}) + \beta_2(\text{Cholesterol Level})$$

Let's say, based on the observed data, we estimate the regression coefficients as:

$$\alpha = 50, \quad \beta_1 = 0.5, \quad \beta_2 = 0.1$$

So for Patient 2 (Age = 50, Cholesterol Level = 240), we predict Blood Pressure as:

$$\text{Blood Pressure} = 50 + 0.5 \times 50 + 0.1 \times 240 = 50 + 25 + 24 = 99$$

We replace the NaN for Patient 2 with the imputed value 99.

Now, we repeat this process for the remaining missing value (for Patient 4).

In the next iteration, the missing value for Patient 4 will be predicted similarly, using the available values for Age and Cholesterol Level and the updated data from the first imputation.

For Patient 4 (Age = 55, Cholesterol Level = 230), we again use the regression equation:

$$\text{Blood Pressure} = 50 + 0.5 \times 55 + 0.1 \times 230 = 50 + 27.5 + 23 = 100.5$$

So, we update the missing value for Patient 4 to 100.5.

The process of imputing and updating continues iteratively for several cycles. After a sufficient number of iterations (let's say 5 iterations), we'll have multiple imputed datasets with different sets of imputed values reflecting the uncertainty inherent in the missing data.

After several cycles of imputations, the final imputed values might look something like the one shown in Table 3.8 with slight variations due to multiple imputations:

Table 3.8: Patient Data with final imputation for Blood Pressure Values

Patient ID	Age (years)	Cholesterol Level (mg/dL)	Blood Pressure (mmHg)
1	45	220	120
2	50	240	99
3	60	200	130
4	55	230	100.5
5	40	210	125

3.7 Class Imbalance Handling

Class imbalance is a major problem in medical diagnosis scenario. It's because majority of the subjects will not have the anomaly that we are looking for. Same is the case with

the dataset used in this work. It had only about 8% of the total test subjects containing the disease. Other people didn't develop this condition. Training a model on this imbalanced dataset would naturally bias the model's prediction towards negative (non-septic). To tackle this, we took a number of approaches as follows:

1. Random selection of equal number of positive and negative class
2. SMOTE (Synthetic Minority Oversampling Technique)
3. ADASYN (Adaptive Synthetic Sampling)
4. Cost-sensitive learning
5. Non-Overlapping Subset Ensemble (NOSE)

In the first approach, we selected all the timestamps with positive prediction. Then, we randomly selected equal number of non-septic hours. So, our dataset had equal number of positive and negative classes. However, in doing so, we lost some information available in the large number of non-septic hours.

In SMOTE technique, we perturb data points of the minority class with the help of k-nearest neighbors and thus, create synthetic samples. It works as follows:

1. Let A be the minority class, for each $x \in A$, the k-nearest neighbors of x are obtained by calculating the Euclidean distance between x and every other sample in set A.
2. The sampling rate N is set according to the imbalanced proportion. For each $x \epsilon A$, N examples are randomly selected from its k-nearest neighbors, and they construct the set A_1.
3. For each sample $x_k \ \epsilon A_1 \ (k = 1,2,3,...N)$, the following formula is used to generate a new example: $x' = x + rand(0,1) * |x - x_k|$
   Here, rand(0,1) represents a random number between 0 and 1.
4. All synthetic data are labelled as minority class.

ADASYN is almost similar to SMOTE with a minor difference. Here, minority samples which are harder to identify are given higher priority for generating synthetic samples. This is done via assigning weights to the minority samples. The samples with higher weights will be more frequently used to generate new fake samples. ADASYN first calculates the density distribution of the minority class samples by finding the ratio of the majority class samples to minority class samples among the k-nearest neighbors of each

minority sample. For example, if we have two minority points A and B, let's consider the following case:

Point A has 2 minority samples and 5 majority samples among its k-nearest neighbors (here, k = 7). So, the ratio is 5/2 = 2.5

Point B has 1 minority and 6 majority samples among its 7 nearest neighbors. So, the ratio is 6/1 = 6.

Then, weights are calculated by normalizing these ratio values.

For point A, $weight = \frac{2.5}{6 + 2.5} = 0.3$

For point B, $weight = \frac{6}{6 + 2.5} = 0.7$

So, point B is given more importance while generating new data points. It seems logical because point B is located in sparser region, with more majority sample points surrounding it.

The cost-sensitive learning method works by assigning more weight to the minority class according to the class frequencies using the following formula:

$$majorityClass_{weight} = \frac{n_{samples}}{n_{classes} \times n_{samples_in_majority_class}} \tag{3.5}$$

$$minorityClass_{weight} = \frac{n_{samples}}{n_{classes} \times n_{samples_in_minority_class}} \tag{3.6}$$

The Non-Overlapping Subset Ensemble method is a new class imbalance handling technique proposed in our work. It attempts to create multiple non-overlapping subsets of the majority, negative class and train multiple models on the subsets. The following figures show the workflow of this approach.

Figure 3.5 illustrates the primary step of our proposed class imbalance handling technique. Here, the patient database is first divided into septic and non-septic hours of data. Afterwards, multiple subsets of non-septic hours of data were created. It was ensured that these subsets were non-overlapping to ensure that the models are able to capture the diverse patterns in the dataset. Each subset contained same hours of data as the septic hour dataset to keep each subset balanced.

The multiple model training phase from the subsets created in the previous step is depicted in Figure 3.6. In this phase, septic and non-septic hour subset are concatenated and a model is trained on this subset. This process is repeated for each non-septic subset in turn. The models can be of same type or varied. The ability to vary the model selection

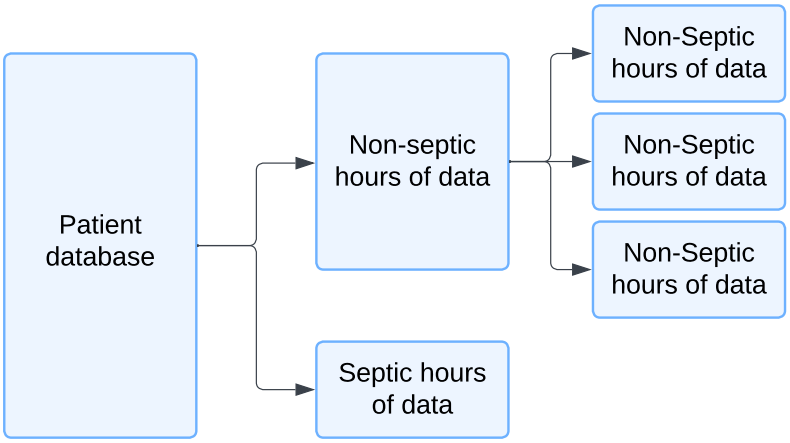


Figure 3.5: Creation of non-septic subset with equal proportion to septic data

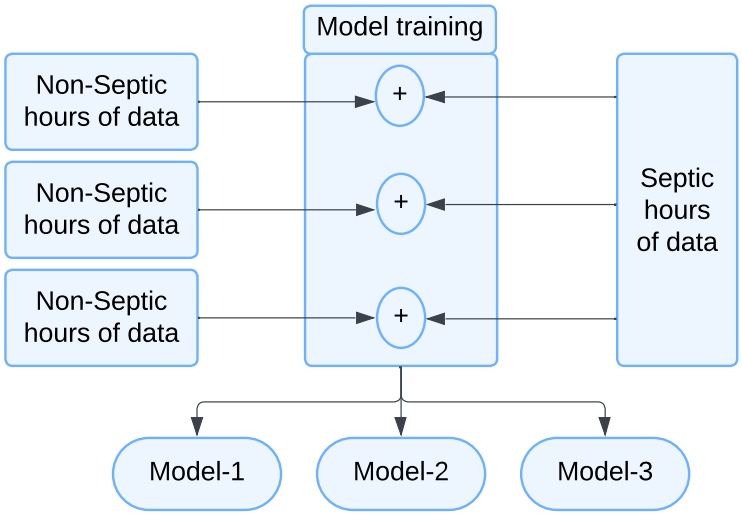


Figure 3.6: Training of multiple models on the created subsets

allows to incorporate models of different architecture such as tree-based method, neural network and so on. This enables us to use the strength of the models to capture subtle patterns in our data.

The final stage of this proposed class imbalance handling method is outlined in Figure 3.7. Here, each of the trained model make predictions on the test dataset. The predictions are combined via voting. The voting can be soft-voting or hard-voting. In the soft-voting, the predictions from each model is averaged and the averaged probability gives us the final prediction based on the set threshold. On the contrary, in the hard-voting, the majority vote

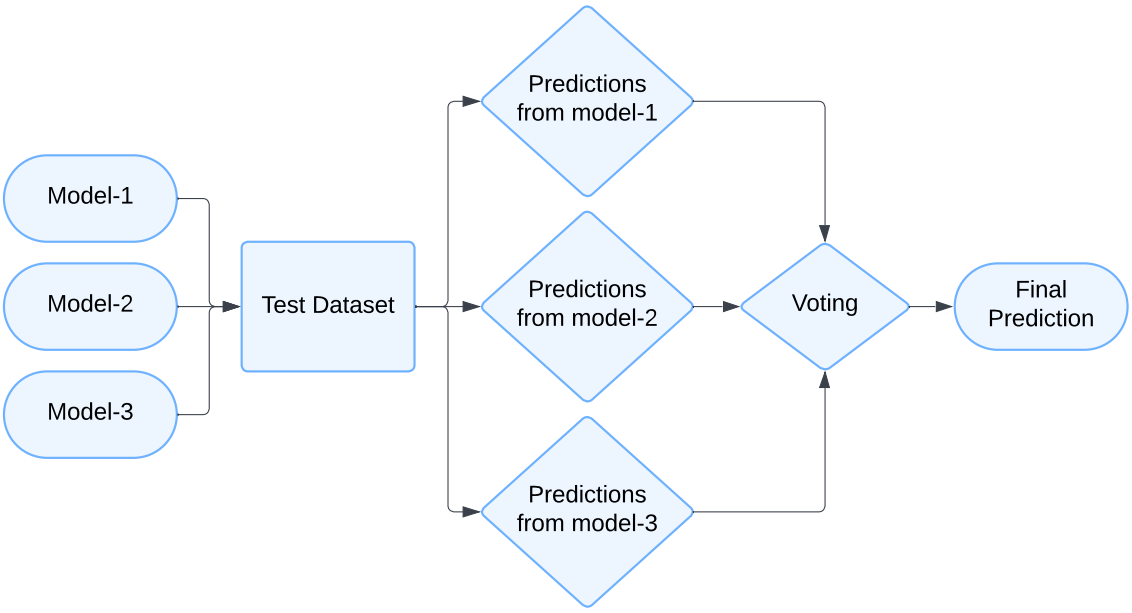


Figure 3.7: Prediction with ensemble model via voting

is taken as the final predicted class of the patient data.

3.8 Model Description and Architecture

A decision tree is an algorithm that tries to classify data based on forming some questions about the input features. Starting from the root node and based on the answers at each node, we arrive at a leaf node which gives the class of that instance. The question at the root node and the choice of threshold can be selected in a number of ways among which, gini impurity index is most commonly used.

3.8.1 Random Forest

A random forest builds a collection of decision trees. The number of trees to build is specified by the user. The trees are independent of one another and are of varying depth. Moreover, the root or top node, the threshold values for different features at different branches also vary widely. This diversity gives random forest its power of capturing the underlying pattern of the input data without overfitting [80]. However, too many trees or very deep trees can incline the model towards overfitting, therefore, resulting in high variance.

In our work, a random forest was trained using 100 decision trees. Maximum depth of

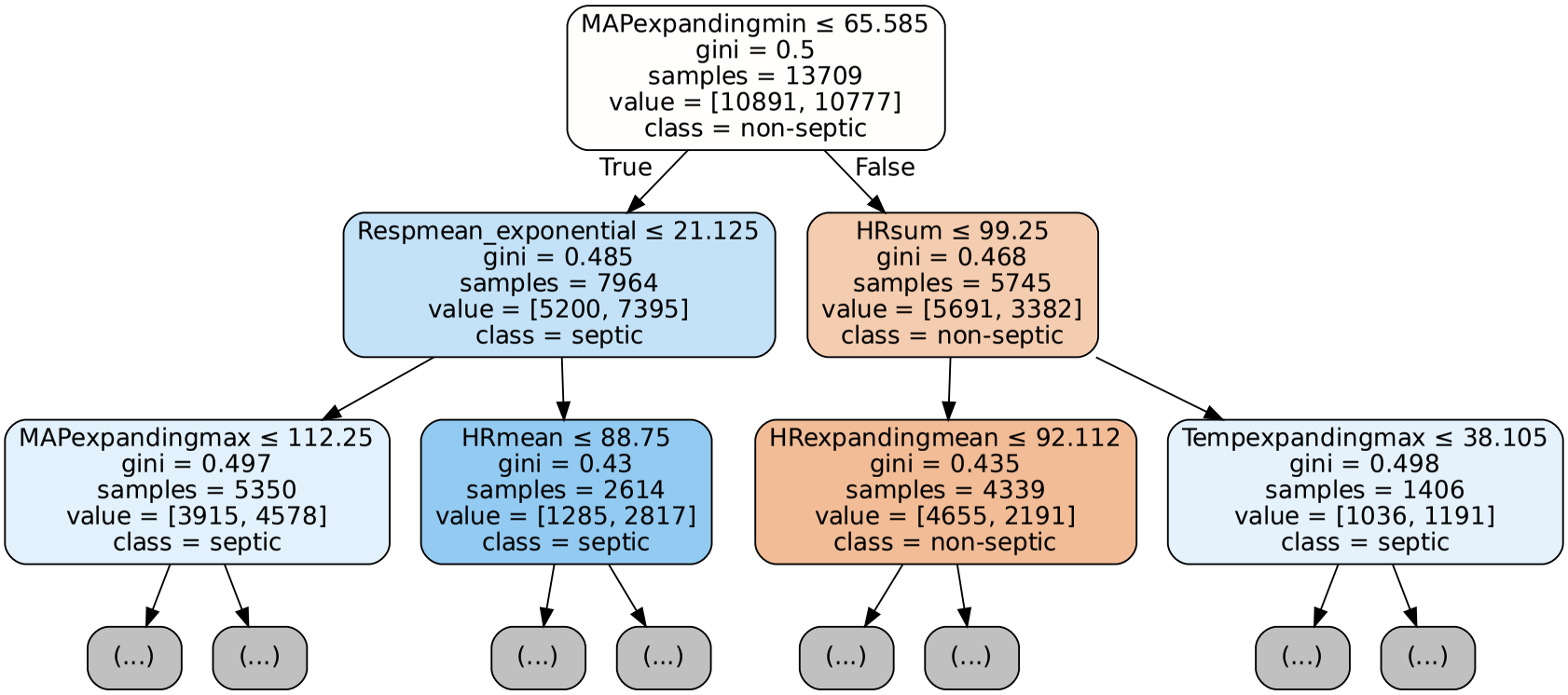


Figure 3.8: Visualization of a sample decision tree (depth 2) from the trained random forest model, showing feature-based splits at each node

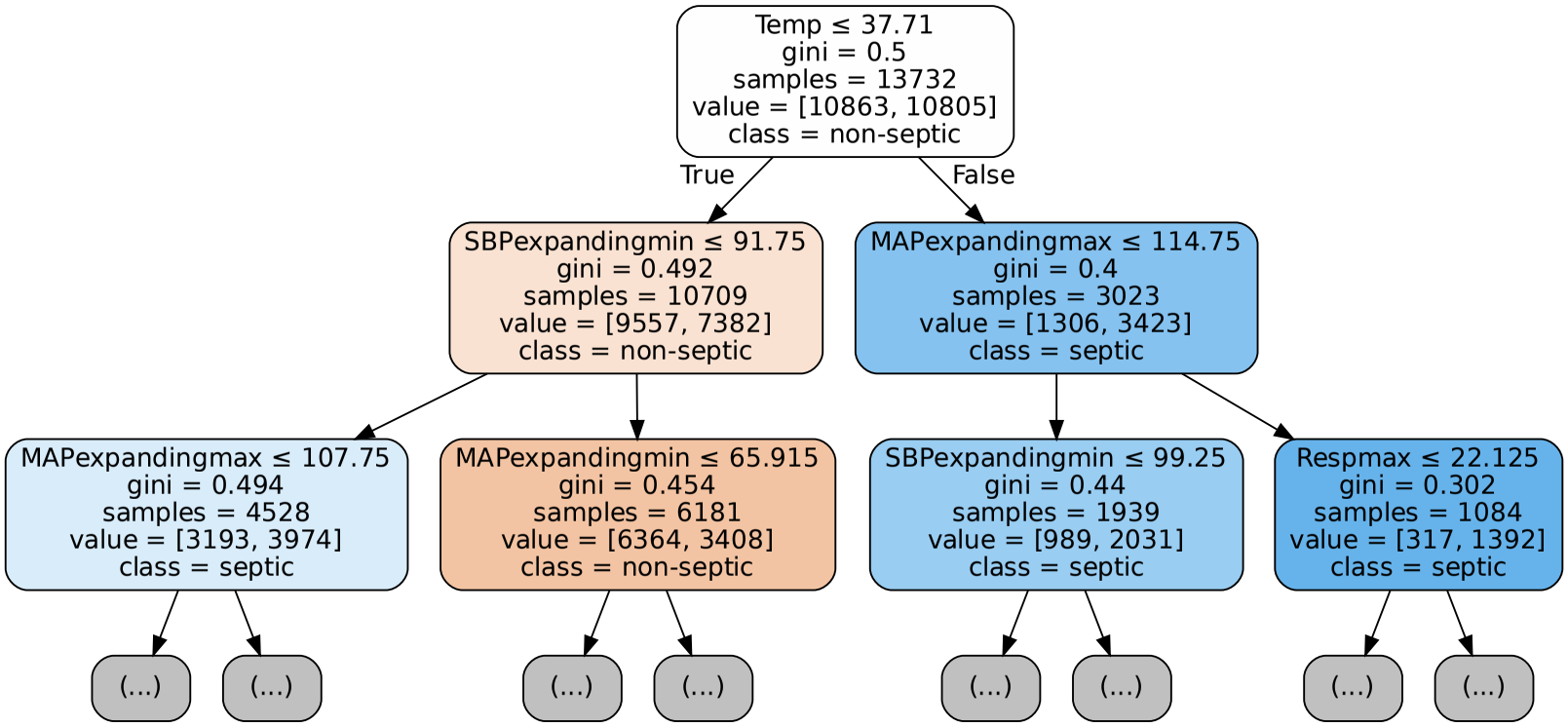


Figure 3.9: Another sample decision tree from the random forest model, highlighting variation in splitting criteria among ensemble members

each tree was restrained to 50 in an attempt to prevent overfitting. Minimum number of required samples for split at a leaf was set to 2 and gini impurity index was selected as the splitting criterion. After dropping the missing value feature columns, all the remaining 133 features were used to build the forest of trees.

Figures 3.8 and 3.9 show two random trees selected from one of the random forest model we trained with selected 50 features on LOCF imputed dataset. Depth of upto 2 levels were shown to keep the figures manageable, otherwise, a random forest trained on 50 features with a depth of 128 is quite difficult to accomodate on a single page. These two trees show the variance of feature selected at nodes and the difference in thresholds as mentioned previously.

3.8.2 Gradient Boost

Gradient boost method is built on the basis of adaboost algorithm. In adaboost, unlike random forest, we use single-leaf trees, known as stumps, instead of full-grown trees. Also, in adaboost, the effect of different stumps on the final prediction vary and the order of the creation of the stumps is also important. Similarly, order is important in gradient boost algorithm too. On the contrary, gradient boost method does not confine its estimators to be only stumps. Rather, the estimators can be trees with limited size, typically having 8 to 32 leaves. Additionally, this algorithm scales all the trees by the same amount. At first, the gradient boost method creates a single leaf as the common prediction for all the samples in the dataset and it's the log of odds. After calculating the residuals from the predicted probabilities, a tree is formed to predict the residuals. This tree as well as the subsequent ones are scaled by a learning rate to control their impact on the final outcome. The prediction from all the estimators are combined to give the final prediction. The visuals in Figure 3.10 and Figure 3.11 illustrate two randomly selected trees from one of our gradient boosting models. To keep the figures manageable, the maximum depth was fixed to only second level.

The mathematical background, according to the original work by Friedman [81], has been presented here in simple terms. To begin, the input to the gradient boosting algorithm is the data $(x_i, y_i)^n$ and a differentiable loss function $L(y_i, F(x))$.

Here, x_i are the features and y_i are the target labels. The total number of rows or samples is indicated by n. We use the negative of the log of likelihood as the loss function.

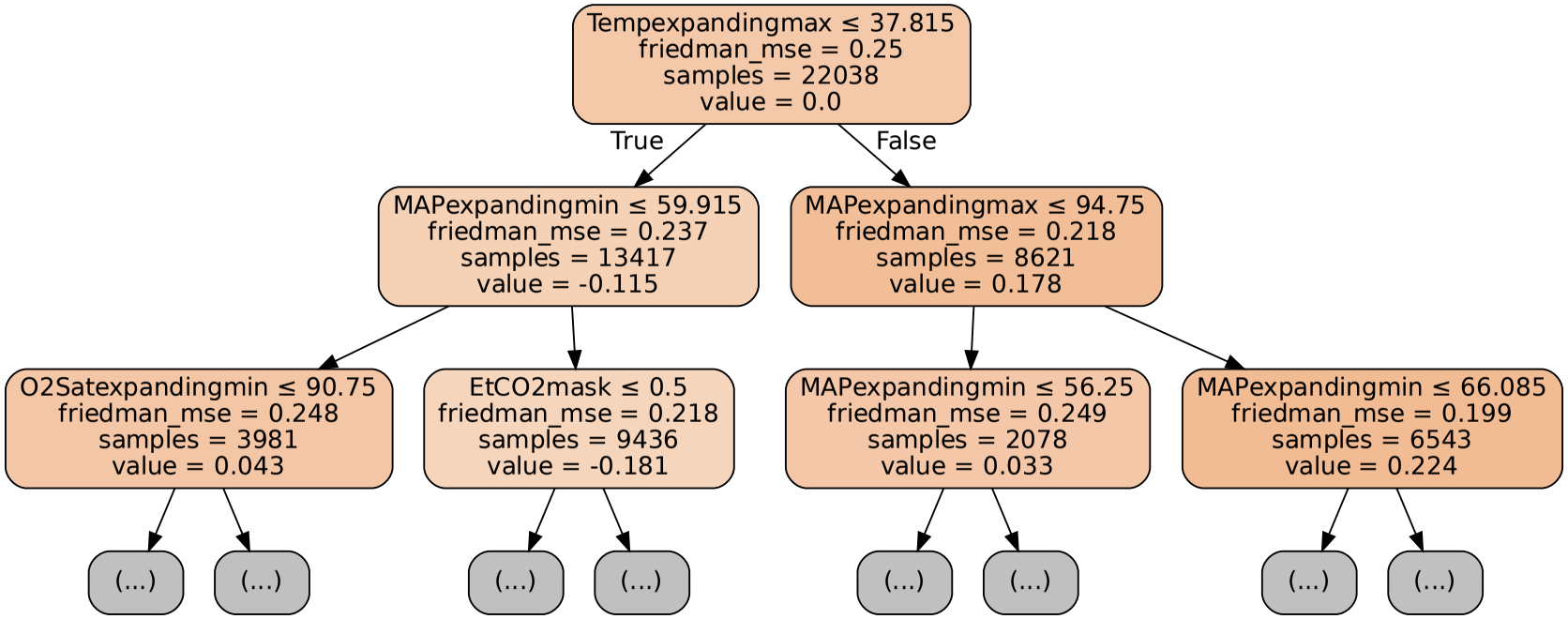


Figure 3.10: A depth-2 tree from the gradient boosting model showing early-stage feature splits

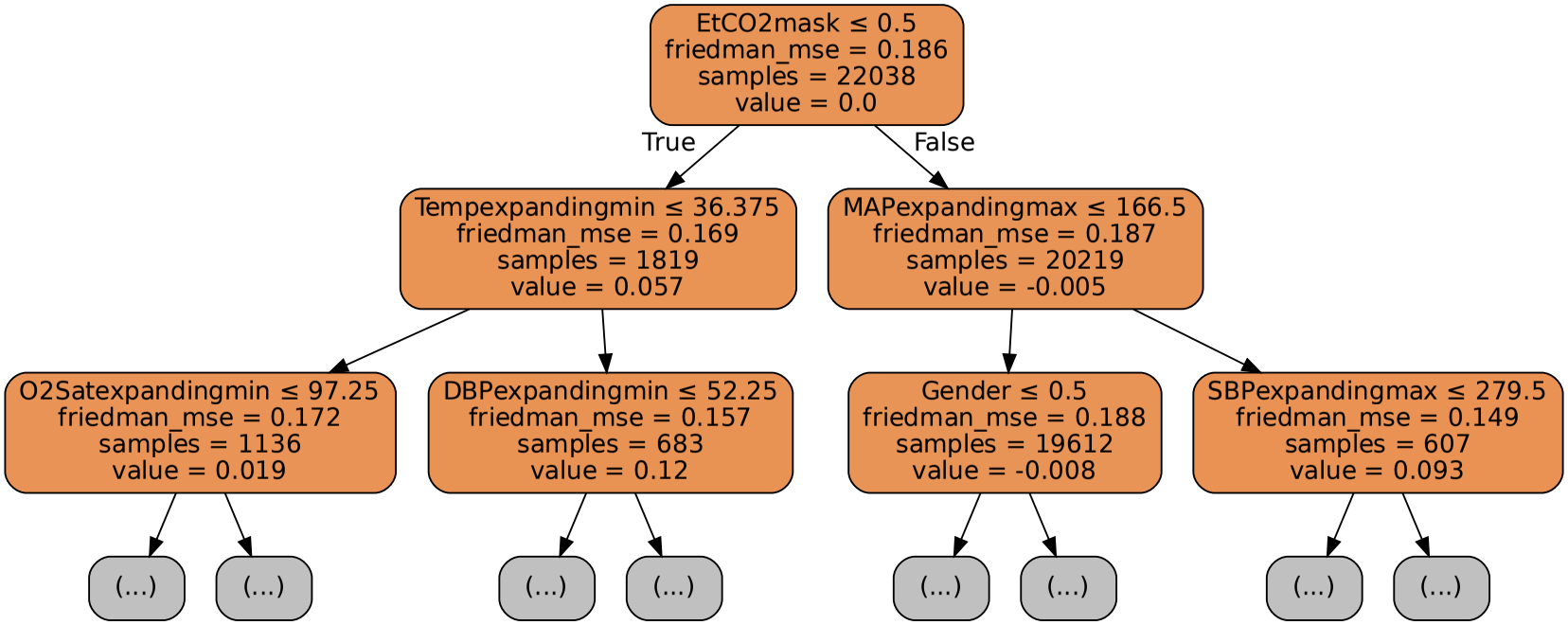


Figure 3.11: Another gradient boosting tree highlighting how shallow splits refine predictions

$$
\begin{aligned}
L &= -\sum_{i=1}^{N} \left(y_i \log(p) + (1 - y_i) \log(1 - p)\right) \\
&= -\text{observed_value} \times \log(p) + \\
&\quad (1 - \text{observed_value}) \times \log(1 - p) \\
&= -\text{observed_value} \times \log(\text{odds}) + \log(1 + e^{\log(\text{odds})})
\end{aligned} \tag{3.7}
$$

Equation 3.7 is our loss function. Here, p is the predicted probability. Taking the derivative of our loss function, we get:

$$
\begin{aligned}
\frac{dL}{d\log(\text{odds})} &= -\text{observed} + \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}} \\
&= -\text{observed value} + p
\end{aligned} \tag{3.8}
$$

We first initialize our model with a single leaf having a constant value as:

$$
F_o(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma) \tag{3.9}
$$

Here, the constant value of the initial leaf node is the value of γ that minimizes the above sum. We can find that the value of the initial leaf node is the log of odds of positive prediction. At the next step, for $m = 1$ to M (total trees), we first compute the pseudo residuals as follows:

$$
r_{im} = -\left[\frac{\partial L(y_i, F(x))}{\partial F(x_i)}\right]_{F(x)=F_{(m-1)}(x)} \tag{3.10}
$$

Then, we fit a regression tree to the pseudo residuals and create terminal regions, i.e. leaves, R_{jm} for j $= 1, J_m$ (total number of leaves). After that, for $j = 1, .J_m$, we compute the output values for each leaf as follows:

$$
\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{ij}} L(y_i, F_{(m-1)}(x_i) + \gamma) \tag{3.11}
$$

The loss function of the equation 3.11 can be simplified as follows using the Taylor series expansion upto the second order term.

$$L(y_1, F_{(m-1)}(x_i) + \gamma) \approx L(y_1, F_{(m-1)}(x_1)) + \frac{d}{dF} L(y_1, F_{(m-1)}(x_1))\gamma + \frac{1}{2}\frac{d^2}{dF^2} L(y_1, F_{(m-1)}(x_1))\gamma^2 \tag{3.12}$$

Finally, we update the output function accordingly and make new predictions for each instance as given below:

$$F_m(x) = F_{(m-1)}(x) + \vartheta \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm}) \tag{3.13}$$

Here, ϑ is the learning parameter which is a value between 0 and 1.

3.8.3 Extreme Gradient Boost

The extreme variant of the gradient boosting model, known as the extreme gradient boost was developed by Chen et al. [82]. We used it in our work for its better performance, built-in regularization and better handling of the missing data. It combines the outcomes from multiple weak learners to formulate the final prediction [83]. XGBoost builds trees by finding optimal values for output that minimizes the following expression.

$$[\sum_{i=1}^{n} L(y_i, p_i^0 + O_{value})] + \frac{1}{2}\lambda O_{value}^2 \tag{3.14}$$

$$\begin{aligned} y_i &= \text{true label for the i-th instance} \\ p_i^o &= \text{initial prediction from the initial leaf} \\ p_i &= \text{current prediction for the i-th instance} \\ O_{value} &= \text{output from the last tree} \\ \lambda &= \text{regularization parameter} \end{aligned}$$

Since we are dealing with classification task in our work, we have used the following loss function for XGBoost.

$$L = -[y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \tag{3.15}$$

Using 2nd order Taylor series expansion to simplify the calculation, we end up with the following expression for the optimal value for output.

$$O_{value} = \frac{\text{Sum of residuals}}{[\sum_i \text{Prev. prob}_i(1 - \text{Prev. prob}_i)] + \lambda} \tag{3.16}$$

Then, we calculate the similarity score using the optimal value for output calculated above and it is found to be as follows:

$$\text{Similarity score} = \frac{1}{2} \times \frac{\left(\sum_{i=1}^{N} g_i\right)^2}{\sum_{i=1}^{N} h_i + \lambda} \tag{3.17}$$

$g_i =$ first derivative, which is related to gradient
$h_i =$ second derivative, which is related to Hessian

The $\frac{1}{2}$ may be omitted because it's only a scaling factor.

Finally, subsequent trees are built by calculating gain based on similarity score. The gain changes depending on the threshold or yes no question we choose for a leaf. The tree building process stops when predefined number of trees is built or no further improvement is seen.

3.8.4 Convolutional Neural Network

We also used convolutional neural network, commonly known as CNN for our task. For this purpose, we had to preprocess our data as a 2-dimensional matrix so that it was suitable for input to the CNN. This process was described in the methodology section.

Table 3.9 illustrates the summary of our implemented CNN, which is a smaller version of the well-known VGGNet architecture.

The convolutional neural network is able to capture spatial pattern in the data by its convolution layers. In simple terms, the convolution layer performs element-wise

Table 3.9: CNN model summary

Layer (Type)	Output Shape	Param #
InputLayer (Input)	(None, 3, 50, 1)	0
Conv2D	(None, 3, 50, 32)	320
Activation	(None, 3, 50, 32)	0
BatchNormalization	(None, 3, 50, 32)	128
Conv2D	(None, 3, 50, 32)	9,248
Activation	(None, 3, 50, 32)	0
BatchNormalization	(None, 3, 50, 32)	128
MaxPooling2D	(None, 1, 25, 32)	0
Dropout	(None, 1, 25, 32)	0
Conv2D	(None, 1, 25, 64)	18,496
Activation	(None, 1, 25, 64)	0
BatchNormalization	(None, 1, 25, 64)	256
Conv2D	(None, 1, 25, 64)	36,928
Activation	(None, 1, 25, 64)	0
BatchNormalization	(None, 1, 25, 64)	256
Dropout	(None, 1, 12, 64)	0
Flatten	(None, 768)	0
Dense	(None, 512)	819,712
Activation	(None, 512)	0
BatchNormalization	(None, 512)	2,048
Dropout	(None, 512)	0
Dense	(None, 1)	513
Activation	(None, 1)	0

multiplication between two matrices. Then, it performs the sum of all the elements and replaces the center pixel with that sum. The convolution layer or filter is also known as kernel. By applying different types of kernels, we can identify different shapes such as vertical edges, horizontal edges in an image. In many scenario, these kernels are hand-defined. However, in a CNN, these filters are automatically learnt via backpropagation so that it is able to detect specific patterns in the 2-dimensional input image. Figure 3.12 illustrates the convolution process.

In Figure 3.12, the small kernel is of 3-by-3 size and the bigger matrix is the image. The kernel slides across the image from left to right and top to bottom. At each step, the overlapping pixels are multiplied and their sum is stored at the location of the central pixel of the kernel. As a natural consequence, the output of this convolution operation will be a matrix of smaller size than the original input image. Sometimes, this decrease in size by convolution layers in larger CNN architecture leads to loss of information. So, padding is added to the input image around the border to make sure that the input and output of a convolution operation are of the same dimension. Padding can be of different types such as zero padding, replicating pixels and wrrapping around. However, zero padding is the

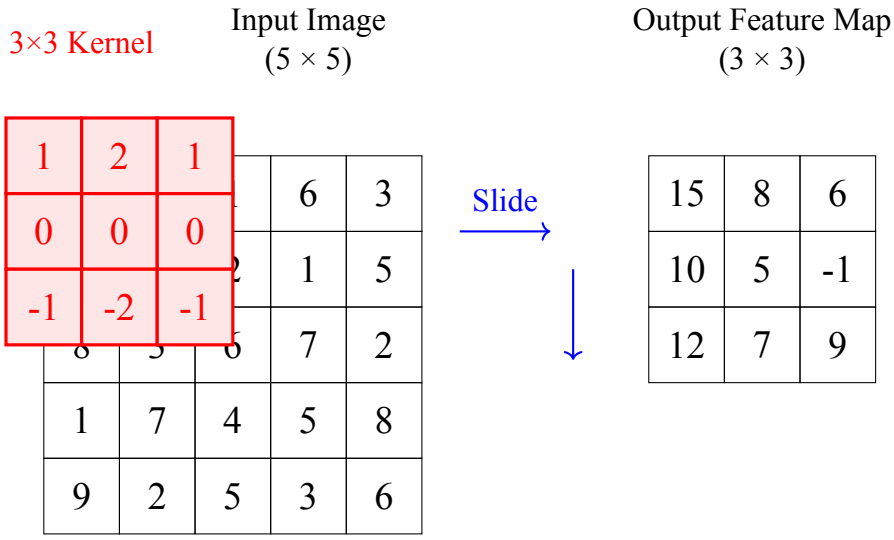


Figure 3.12: A 3×3 kernel slides over a 5×5 input image producing a 3×3 output feature map

most commonly used because it is simple (border of zeros are added to the original matrix) and computationally less expensive.

A CNN consists of the following layers:

1. Convolutional layer
2. Activation layer
3. Pooling layer
4. Batch normalization layer
5. Dropout layer
6. Fully connected layer

The convolutional layers are the core building block of a CNN. This layer consists of a set of K learnable filters where each filter, also known as kernel, has a specific height and width. These filters are small in terms of spatial dimension but extend throughtout the full depth. After applying all the filters in a convolution layer to its input, a set of K 2-dimensional activation maps are obtained. These activation maps are stacked and serve as input to the subsequent layers.

In convolutional neural networks (CNNs), each output value is produced by a neuron that focuses on just a small portion of the input. This design helps the network learn to detect specific features that appear in certain areas of the input image. Rather than connecting every neuron to all previous layer outputs—which would require an overwhelming number of connections and make training difficult for large images—CNNs

limit each neuron to a smaller, local region. This region is known as the neuron's receptive field. The idea of sliding a small filter over a larger image plays a crucial role here, enabling local connectivity and efficient learning of spatial patterns.

For our dataset, we reshaped 6 hours of data, each consisting of 50 features, as a 2-dimensional image. When the first convolution layer consisting of 32 kernels of 3-by-3 size was convolved with the input image, we got 32 activation maps, each of size 6-by-50. Since our 2-dimensional data has only single channel, each kernel connects to a region of 3-by-3.

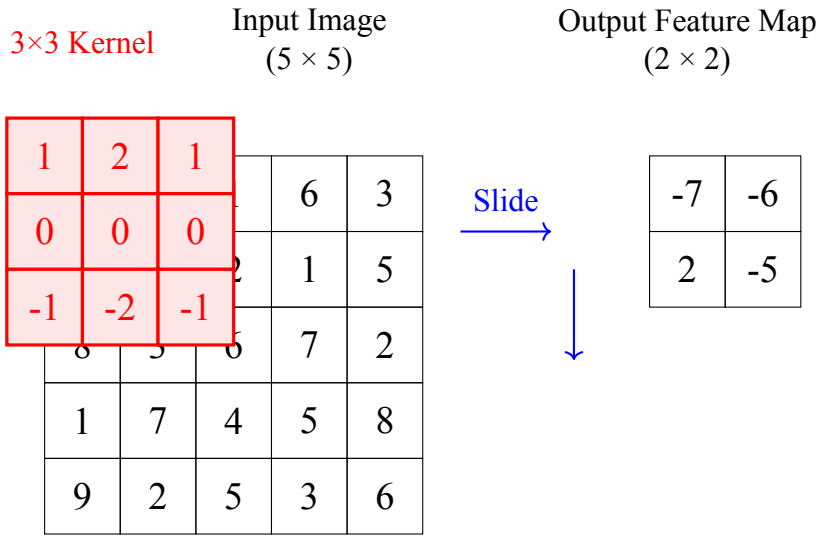


Figure 3.13: A 3×3 kernel with a stride of 2 slides over a 5×5 input image producing a 2×2 output feature map

The stride of the kernel during convolution operation is defined as the step size. After each convolution operation, the kernel can be slided by 1 or 2 pixels, or even more as required. Smaller stride results in overlapping receptive fileds and thus, larger output volumes. On the contrary, larger stride leads to non-overlapping receptive fields and hence, smaller output volumes. Figures 3.13 shows the result of a stride of 2 of the previous kernel in Figure 3.12 with the same input matrix. The result is a smaller output feature map of size 2 by 2.

After each convolution layer, we apply a nonlinear activation function. Activation functions take the output of the convolutional layer and outputs some value according to the nature of the underlying function. These functions introduce non-linearity in the network, thus, enabling it to learn complex pattern from the data. Without nonlinear activation functions, a neural network, like CNN would behave like a simple linear regression model despite the number of hidden layers in it. Some of the common activation functions are as follows:

1. Step function

2. Sigmoid
3. tanh (Hyperbolic tangent)
4. ReLU (Rectified Linear Unit)
5. Leaky ReLU
6. ELU (Exponential Linear Unit)

Figure 3.14 to Figure 3.19 show the visuals of these activation functions.

The step function is defined as follows:

$$f(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{3.18}$$

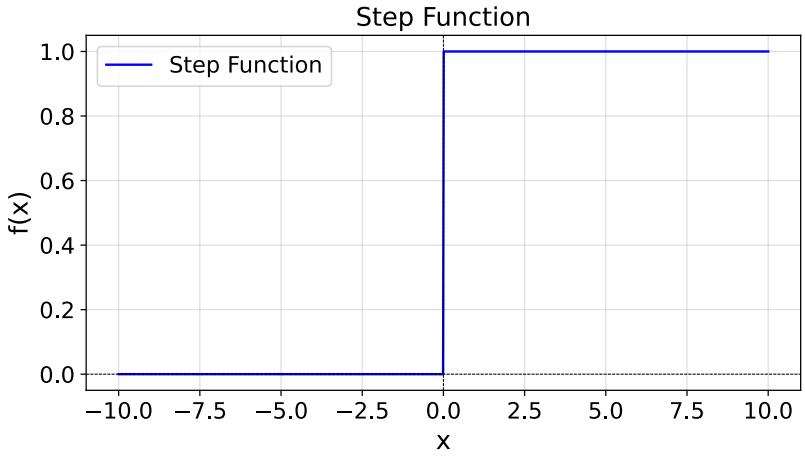


Figure 3.14: Step function

Although it is a nonlinear function, it is not used often since it is not differentiable. Additionally, it outputs only 1 and 0. Hence, not much information about the output is captured. It also suffers from the vanishing gradient problem.

A more commonly used activation function is the sigmoid function which is defined as follows:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.19}$$

This function, unlike the step function, is smooth, differentiable and continuous. It outputs a value between 0 and 1 and is often used for binary classification task. Based on a threshold, the output of sigmoid function is converted into binary label of 1 or 0.

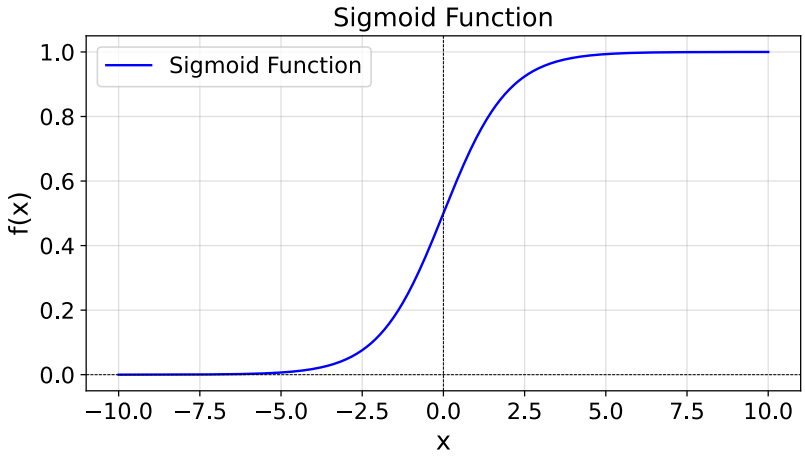


Figure 3.15: Sigmoid function

The hyperbolic tangent function is zero centered, unlike the sigmoid function. It is expressed in the following equation:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.20}$$

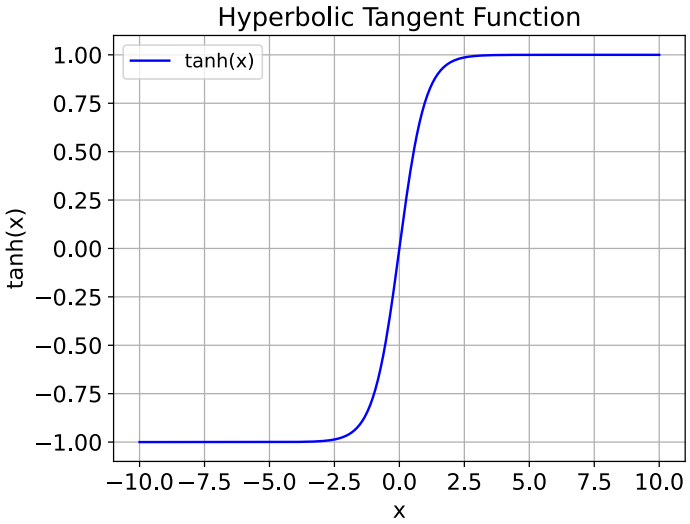


Figure 3.16: Hyperbolic tangent

Rectified Linear Unit (ReLU) is one of the most commonly used activation functions in CNN. This function is described as:

$$f(x) = max(0, x) \tag{3.21}$$

It outputs the maximum of the two inputs, zero or x. Consequently, it is zero for negative values of x and looks like a ramp for positive x values. Unlike some of the previously discussed activation functions, this one is not saturable and is also

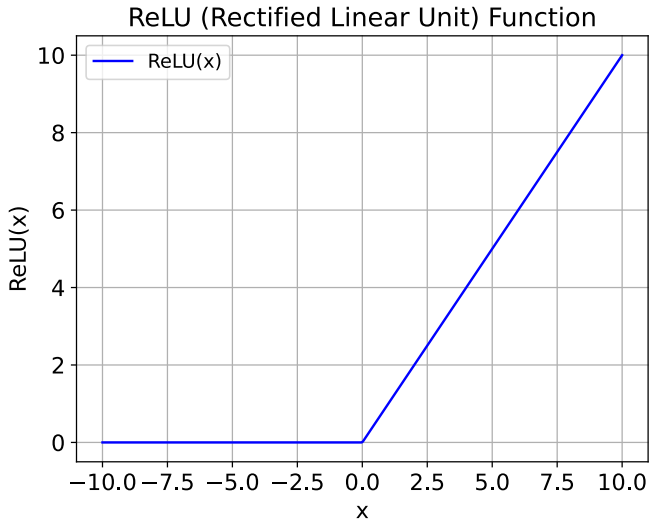


Figure 3.17: Rectified linear unit

computationally efficient. However, when the value of x is negative, the gradient cannot be taken.

A variant of ReLU is called Leaku ReLU which, in mathematical notation, can be described as follows:

$$f(x) = max(\alpha x, x) \tag{3.22}$$

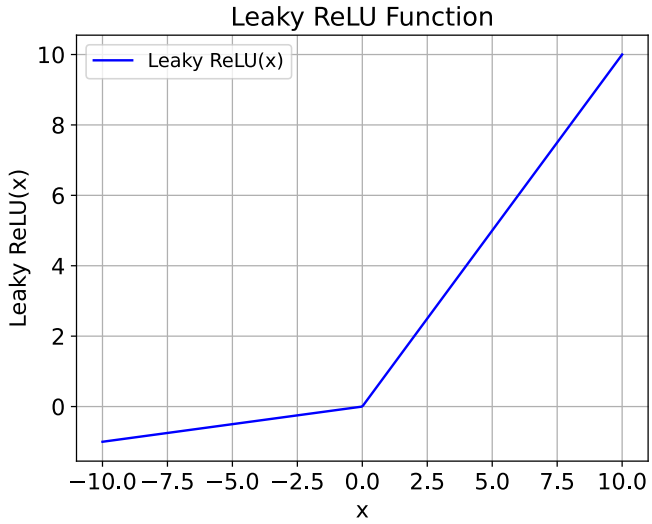


Figure 3.18: Leaky rectified linear unit

Where α is a small value of, for example, 0.01. This expression allows for small gradient for negative values of x, that is, when the unit is inactive.

ELU aims to fix some problems with ReLU and Leaky ReLU by allowing negative values for inputs, which helps to push the network's activations to zero mean and alleviate

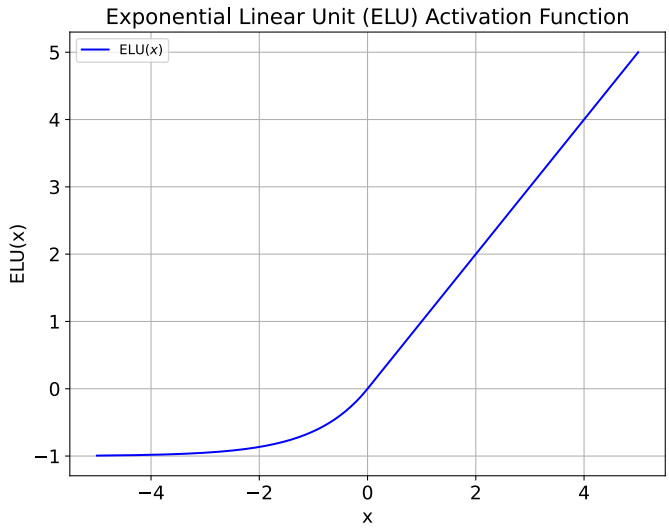


Figure 3.19: Exponential linear unit

the vanishing gradient problem.

$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0, \\ \alpha(\exp(x) - 1) & \text{if } x \leq 0, \end{cases} \tag{3.23}$$

3.8.5 Recurrent Neural Network

Recurrent Neural Network is a category of neural network that is specialized to work on sequential data, also known as time series data. Typical neural network fails to capture the time dependence nature of data, such as patient heart rate, body temperature changing from one timestep to the next. However, RNN, due to its architecture, is capable of capturing this time-varying pattern. The main strength of RNN lies in its hidden state, which works as a memory and is used to use past information in combination with current input to predict current output.

In a standard RNN, the hidden state at current timestep is updated based on the current input and previous hidden state. The equation for hidden state update is given as:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b_h) \tag{3.24}$$

h_t : hidden state at time t

x_t : input at time t

W_x : weight matrix for input x

W_h : weight matrix for hidden state h

b_h : bias term for hidden state

σ : activation function, tanh is most commonly used

The output at timestep t is typically a function of the current hidden state and is expressed as a linear combination of a weight matrix and a bias term as follows:

$$y_t = W_y h_t + b_y \tag{3.25}$$

w_t : weight matrix that decides how strongly
each hidden state feature influences each output unit

b_y : bias vector that lets the network shift
the output up or down independently of the hidden state

Depending on the task at hand, the output is passed through an activation function to get the final prediction.

To train the RNN, the gradients of the loss function with respect to the weight matrices are calculated using chain rule during backpropagation. Then, the new weights are found by adding to the previous weight the gradients multiplied by a learning rate. This learning rate is a hyperparameter.

However, as gradients are calculated back through timesteps, basic RNN suffers from a serious drawback known as vanishing gradient or exploding gradient. When the gradient is greater than unity, as it is calculated back through timesteps, it becomes huge. This is called the exploding gradient problem and may result in instability during the learning phase. It causes overshoot around the minima and the model fails to converge at a low loss region. On the other hand, if the gradient is smaller than unity, as it is backpropagated through timesteps, it becomes very small. This is called the vanishing gradient problem. The model fails to converge in this case due to the steps being very small during the weight updates. So, it becomes difficult to learn long-term dependencies. In our task, the RNN

was many-to-one where multiple inputs at different timesteps were used to predict the output label at current timestep. Figure 3.20 shows a RNN neuron along with a typical architecture for a many-to-many RNN where the final hidden state at time t is used for prediction.

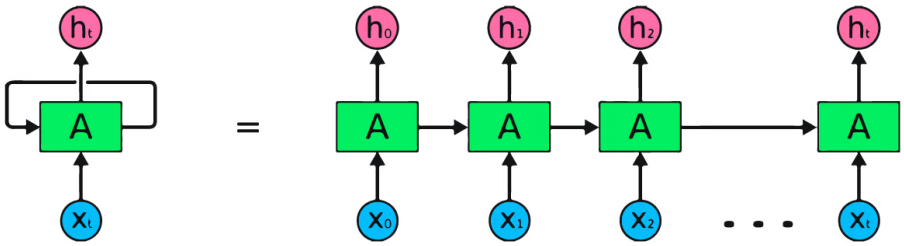


Figure 3.20: Architecture of a many-to-one RNN

To solve the vanishing or exploding gradient problem,a new architecture is used. This is called Long Short Term Memory (LSTM). The LSTM uses different paths for long and short term memories. Figure 3.21 shows a LSTM cell.

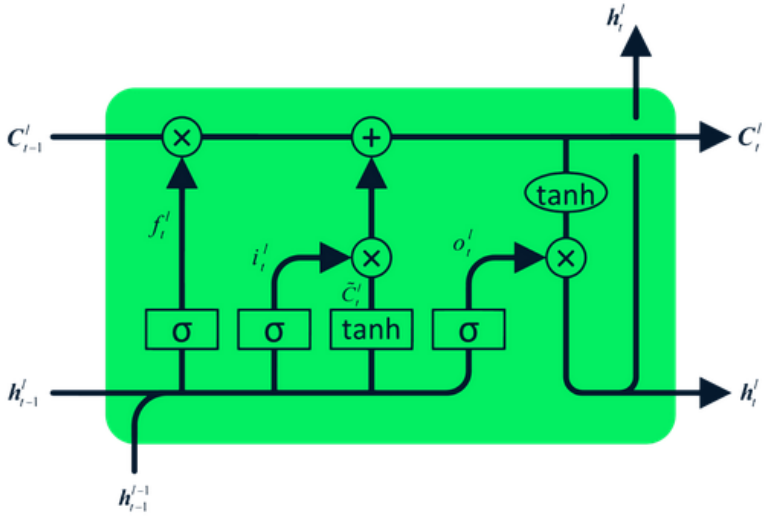


Figure 3.21: LSTM cell

As mentioned, we can see two different paths for the long and short term memories. The upper line represents the long-term memory and is called the cell state. The lower line contains the hidden state and at the bottom, we can see the connection from input. The leftmost gate in the LSTM cell consists of a connection from the previous hidden state and the input. Both of these are multiplied by their respective weight matrices, summed up and passed through a sigmoid activation function. This activation converts the output into a value between zero and one. The output from this sigmoid function, when multiplied by the original long-term memory, determines what portion of the long-term memory is remembered and what portion is forgotten. Thus, it is called the forget gate.

The next part of the LSTM cell contains a similar structure like the forget gate in conjunction to another identical structure. However, its activation function is the hyperbolic tangent function and it creates a potential long-term memory. The forget gate part in this section decides what portion of the newly created potential long-term memory is remembered. This is added with the preserved original long-term memory to create the new long-term memory. This part is called the input gate.

Finally, the rightmost part contains a hyperbolic tangent activation function. It takes the new long-term memory as the input. Another forget gate in this part dictates what portion of this long-term memory is preserved to create the new short-term memory. This new short-term memory is also the output of the LSTM cell. Therefore, this portion is called the output gate. Even though LSTM excels at handling long-term sequences, it is more complex, requires longer to train and is prone to overfitting.

3.9 Federated Learning

Federated learning is a machine learning approach that enables multiple devices (often referred to as ”clients”) to collaboratively train a shared model without transferring sensitive data to a central server. Instead of sending the raw data, each device performs local training on its data and shares only the model updates (such as weights or gradients) with the central server. The server then aggregates these updates to improve the global model. This method ensures data privacy while still leveraging the collective knowledge from multiple sources.

Federated learning is particularly beneficial in scenarios where data is distributed across multiple devices, and privacy is a concern. Some key reasons for its need are:

1. **Data Privacy:** In industries like healthcare or finance, sensitive data cannot be sent to a central server due to privacy regulations. Federated learning allows models to be trained on local data without exposing it.
2. **Efficient Use of Resources:** Federated learning leverages the computational power of devices like smartphones or Raspberry Pis, allowing for decentralized training and avoiding the need for massive data transfer.
3. **Scalability:** With federated learning, a large number of devices can participate in training the model, enabling the use of vast amounts of distributed data without overloading central servers.

Although federated learning offers numerous advantages, it also comes with several challenges:

1. **Data Heterogeneity:** The data available on each device may be diverse in nature, leading to difficulties in aggregating updates that work well across all clients.

2. **Communication Efficiency:** Sending model updates from multiple devices to the central server can be costly in terms of bandwidth. Optimizing the communication of these updates is a critical challenge.

3. **Model Convergence:** Ensuring that the model converges to a useful global solution despite the decentralized training process can be difficult. Sometimes, updates may conflict or slow down the training process.

4. **Security and Trust:** Even though raw data isn't transferred, adversarial attacks could manipulate model updates. Secure aggregation techniques must be used to safeguard the learning process.

In our work, we also extended our approach towards Raspberry Pi, leveraging its lightweight nature, portability and processing power to train model locally. This ensures data privacy of patients. The following points outline the method.

1. **Local Model Training:** Each Raspberry Pi collects data from sensors or other local sources and trains a model on that data.

2. **Sending Model Updates:** Once the local model is trained, the Raspberry Pi sends the model updates (like the weights of a neural network) to a central server over Wi-Fi or a local network. This avoids the need to share sensitive data directly.

3. **Model Aggregation:** The central server aggregates the updates from all devices. Techniques like Federated Averaging can be used to combine the updates into a global model.

4. **Retraining and Synchronization:** The global model is then sent back to the Raspberry Pis, where the devices continue training the model on their local data. This process repeats as new data is collected, continuously improving the model.

To simulate the above scenario, we saved different small portions of our original dataset on the Raspberry Pi and trained separate random forest models on the different datasets. This mimics the settings where each smaller datasets represent data from different hospitals and the different models are independent of one another, not knowing the data from the other hospitals. Also, to keep things simple and ensure lower resource utilization, we restricted the number of estimators in each of the random forest models to five.

Later on, the estimators from the models were aggregated to give the new model better performance and generalizability. Separate models and the aggregated models were all tested on a third dataset and performance metrics were compared.

In another approach, instead of simply aggregating estimators from random forests, we evaluated their cross validation performance for the recall performance metric, emphasizing the need to reduce false negative cases. In this method, we selected more trees from the forest which exhibited better recall. And finally, the selected trees from each forest were merged to create a new forest. The recall of the merged forest was better than the individual forests, indicating the usefulness of federated learning technique.

3.10 Shapley Values

Explainability of machine learning model has become important to solve the trust issue. We tried to explore the working of our model to make it more trustworthy using the SHAP values and LIME plots. SHAP values were also used in the work [84].

SHAP is more commonly used because it satisfies the four axioms of fairness; axioms of efficiency, symmetry, nullity, and linearity. SHAP also provides global as well as local explanation for the predictions. The SHAP package in python provides some useful plots which we used to gain insights into the model. The Shapley value of a feature is the average marginal contribution of that feature across all possible coalitions and is given by the following equation:

$$
\begin{aligned}
&\text{Shapley value of feature } i \\
&= \text{expected marginal contribution of feature } i \\
&= \text{weighted average of contribution of feature } i \\
&= \sum_{S \subseteq \{1,\ldots,P\} \setminus \{i\}} \frac{|S|!(P-|S|-1)!}{P!} [f(S \cup \{i\}) - f(S)]
\end{aligned} \tag{3.26}
$$

$$
\begin{aligned}
P &= \text{total number of players (features)} \\
P! &= \text{number of ways to form a sequential coalition of } P \text{ players} \\
S &= \text{number of players in the coalition where player } i \text{ will join} \\
S! &= \text{number of ways } S \text{ players can join the coalition before player } i \\
(P - |S| - 1)! &= \text{number of ways players can join the coalition after player } i \text{ joins} \\
f(S \cup \{i\}) &= \text{model's prediction with the feature } i \\
f(S) &= \text{model's prediction without feature } i
\end{aligned}
$$

Shapley value is superior to correlation coefficients because the later is applicable to only linear models and only indicates global relationship whereas the Shapley values are equally applicable to both linear and non-linear relationships and is able to explain the individual predictions. However, the downside of Shapley values is that they are computationally expensive.

3.11 Locally Interpretable Model-agnostic Explanation

Local surrogate models are interpretable models that aim to explain the prediction of a complex, black-box machine learning model for a specific instance. These models offer a faithful approximation of the original model's behaviour in the local neighborhood of that instance, without requiring access to the internal workings of the model. The underlying intuition is that while a machine learning model may be highly non-linear and complex globally, its decision boundary can often be well-approximated by a simpler model (e.g., a sparse linear model) in a small region around a specific input.

To generate such explanations, we adopt a perturbation-based strategy. Specifically, the instance to be explained is slightly modified multiple times (perturbed) to generate a set of synthetic data points in its local vicinity. The black-box model is then queried to obtain predictions for each of these perturbed samples. Using these samples and their corresponding predictions, a simple interpretable model — typically a sparse linear regression — is trained. This local model serves as the explanation, highlighting which features contributed most to the prediction of the black-box model in that region.

This approach is model-agnostic, making it applicable to a wide range of machine learning models including neural networks, ensemble methods, and support vector machines. The focus on local fidelity, rather than global accuracy, ensures that the explanation is relevant and specific to the individual prediction under consideration.

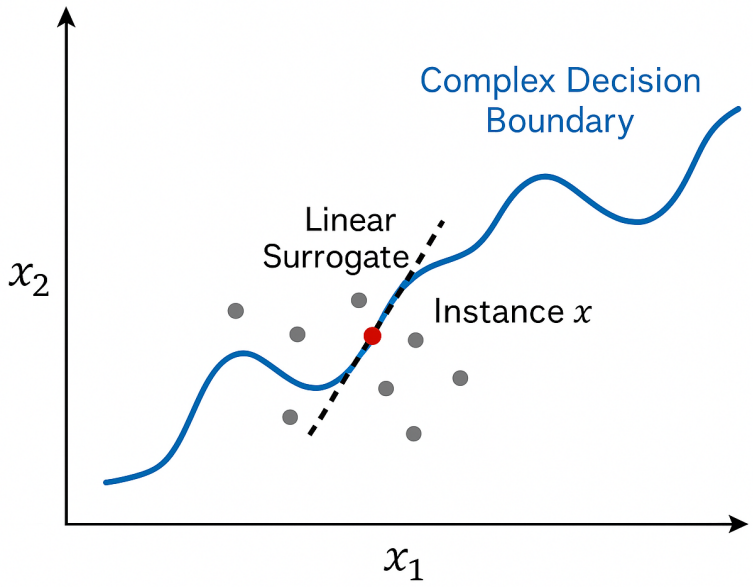


Figure 3.22: A local surrogate model to approximate the complex decision boundary

Figure 3.22 illustrates the concept of fitting a linear surrogate model to a non-linear decision boundary in the neighborhood of a specific instance.

Chapter 4

Result Analysis

This chapter evaluates the performance of the proposed sepsis diagnosis system using various quantitative metrics and comparative analyses. We begin by introducing standard evaluation metrics, including the confusion matrix, AUROC, and utility score. The performance of different models is then compared based on class imbalance handling, missing data strategies, and feature selection techniques. Additionally, we analyze the effectiveness of deep learning models such as RNN and CNN, and assess the impact of incorporating federated learning. Finally, we compare model sizes and evaluate their performance when deployed on edge devices to ensure practical feasibility.

4.1 Evaluation Metrics

In order to evaluate the propsed model for sepsis diagnosis, we used metrics derived from the confusion matrix such as accuracy, recall, F1 score and so on. Moreover, we utilized the AUROC which is a good measure of the model's ability to distinguish between the two classes. In addition to the above mentioned metrics, we also used the utility score proposed by the organizer of the challenge dataset. The utility score is a decent measure to assess the model's capability to diagnose sepsis early with a focus on reducing false negative as well as false positive. Finally, we harness a number of SHAP plots such as beeswarm plot, bar plot, waterfall plot and so on to gain insight into the working of our model. These visuals attempt to solve the trust issues revolving the black-box model and bridge the communication gap between human and AI models.

4.1.1 Confusion Matrix

Confusion matrix is a performance metric to evaluate a machine learning model for classification with two or more categories. Fig. 4.1 shows a typical confusion matrix.

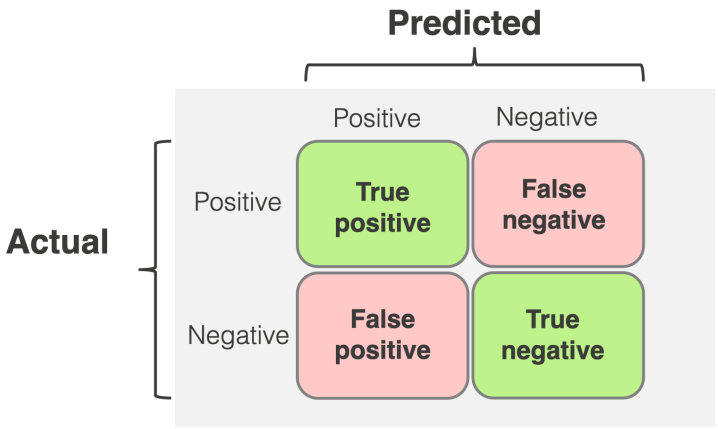


Figure 4.1: Confusion matrix

From the sample confusion matrix shown in Fig. 4.1, we would want to maximize the total number of true positives and true negatives. The false positive and false negative cases have different impacts in different applications. For our case, the focus is to reduce false negative because a false negative would result in high mortality. False positive, although less problematic, leads to unnecessary consumption of limited hospital resources. We can measure different metrics from the confusion matrix that addresses specific aspect of the task at hand.

$$\text{Accuracy} = \frac{TN + TP}{TN + TP + FN + FP} \tag{4.1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

$$\text{Recall (Sensitivity, True Positive Rate)} = \frac{TP}{TP + FN}$$

$$\begin{aligned} \text{False Positive Rate (FPR)} &= 1 - \text{Specificity} \\ &= 1 - \frac{TN}{TN + FP} \end{aligned} \tag{4.3}$$

$$\text{F1 score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{4.4}$$

From the above equations of sensitivity and specificity, we can see that as the threshold increases, positive prediction decreases, negative prediction increases. As a result sensitivity decreases. On the other hand, with decreasing threshold, positive

prediction increases, negative prediction decreases. As a result, specificity goes up. As with any other scenario, we want higher true positives and lower false positives for our model. However, this is challenging due to the heavy imbalance that exists in our database.

4.1.2 AUROC (Area Under Receiver Operating Characteristics)

AUROC is a performance measurement for classification problem at various threshold settings. Receiver Operating Characteristics (ROC) is a probability curve and Area Under Curve (AUC) represents the degree or measure of separability. It tells us how much the model is capable of distinguishing between different classes.

4.1.3 Utility Score

A utility score was defined in the Physionet challenge from where the database in our work was taken. A score $U(s,t)$ was defined for each prediction, that is, for each patient s and for each time interval t. This utility function can take on any of the following four values for the four possible cases, that is false positive and negative, true positive and negative.

$$\begin{aligned} U_{TP}(s,t) &= \text{positive prediction at time} t \text{ for a sepsis patient } s \\ U_{FP}(s,t) &= \text{positive prediction at time } t \text{ for a non-sepsis patient } s \\ U_{FN}(s,t) &= \text{negative prediction at time } t \text{ for a sepsis patient } s \\ U_{TN}(s,t) &= \text{negative prediction at time } t \text{ for a non-sepsis patient } s \end{aligned} \tag{4.5}$$

Fig. 4.2 and Fig. 4.3 illustrate the utility score for a sepsis patient with t_{sepsis} assumed to be 48 as an example.

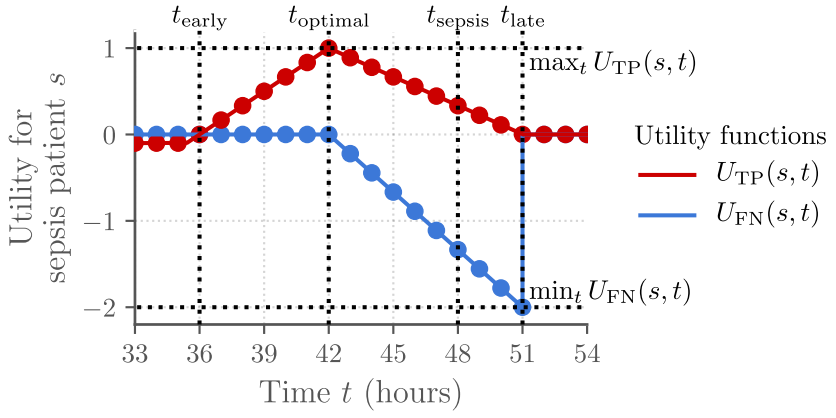


Figure 4.2: Utility score diagram for sepsis patients

[13]

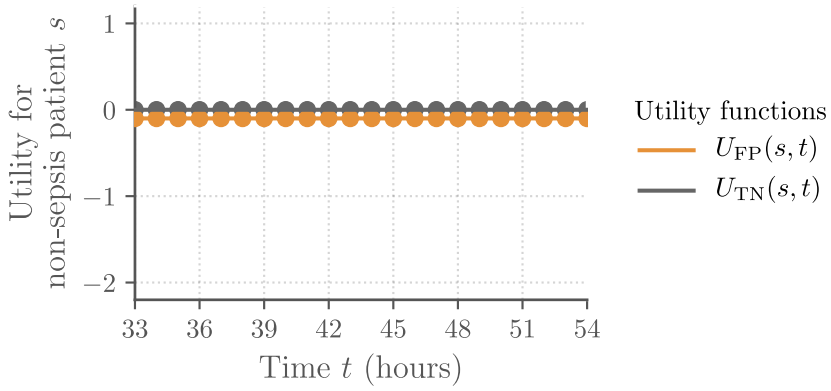


Figure 4.3: Utility score diagram for non-sepsis patients

[13]

The t_{sepsis} is assumed to be the minimum of the two times, t_{sofa} and $t_{suspicion}$ as long as t_{sofa} is between 24 hours before and 12 hours after $t_{suspicion}$. $t_{suspicion}$ is the time when sepsis was suspected by the doctor whereas t_{sofa} is the time indicated by a 2 points deterioration in sofa score. The classifier is rewarded for positive prediction of septic patient between 12 hours before and 3 hours after t_{sepsis} . The maximum reward is given for prediction at an optimum time of 6 hours before t_{sepsis} . Classifiers that do not predict sepsis or predict sepsis too late for septic patient are penalized where the penalty for early detection is -0.05 and the maximum penalty for late detection is -2.0. For non-septic patient, classifiers that do not predict sepsis don't get any reward. However, classifiers that predict sepsis for the non-septic patients are penalized by an amount of -0.05. Afterwards, a score is calculated after summing over predictions from all rows from all patients as follows:

$$U_{\text{total}} = \sum_{s \in S} \sum_{t \in T(s)} U(s,t) \tag{4.6}$$

Then, a normalized utility score is calculated to improve readability and compare performance across classifiers:

$$U_{\text{normalized}} = \frac{U_{\text{total}} - U_{\text{no predictions}}}{U_{\text{optimal}} - U_{\text{no predictions}}} \tag{4.7}$$

4.1.4 Confidence Metrics

Predicted probability (confidence score): For every patient i the calibrated random–forest outputs a probability

$$\hat{p}_i \;=\; \Pr\big(\text{sepsis} \mid \mathbf{x}_i\big) \in [0,1],$$

which we refer to as the *confidence score*. After post-hoc calibration, these values can be interpreted literally: e.g. $\hat{p}_i = 0.83$ means that about 83% of patients with similar features are expected to develop sepsis.

Brier score (overall probabilistic accuracy): The Brier score measures the mean-squared gap between predicted probabilities and actual outcomes:

$$\text{BS} \;=\; \frac{1}{N}\sum_{i=1}^{N}\big(y_i - \hat{p}_i\big)^2, \qquad y_i \in \{0,1\}. \tag{4.8}$$

A perfect forecaster has BS $=$ 0. Lower values therefore indicate more reliable and discriminative probability estimates.

4.2 Comparison Between Models Based on Class Imbalance

Following sections compare the performance of different models with different approaches for handling class imbalance. In these cases, missing values were handled via forward filling and 50 features were selected with analysis of variance method.

4.2.1 Random Undersampling

Figure 4.4 illustrates the area under receiver operating characteristics for our tree-based models where class imbalance was handled by random downlsampling of the majority class. It shows that the random forest model achieved the highest area under curve score of 0.94 whereas the other two models (gradient boost and its extreme version) both received a score of 0.89. Figure 4.5 illustrates the comparison of the models in terms of accuracy, precision, recall and F1-score.

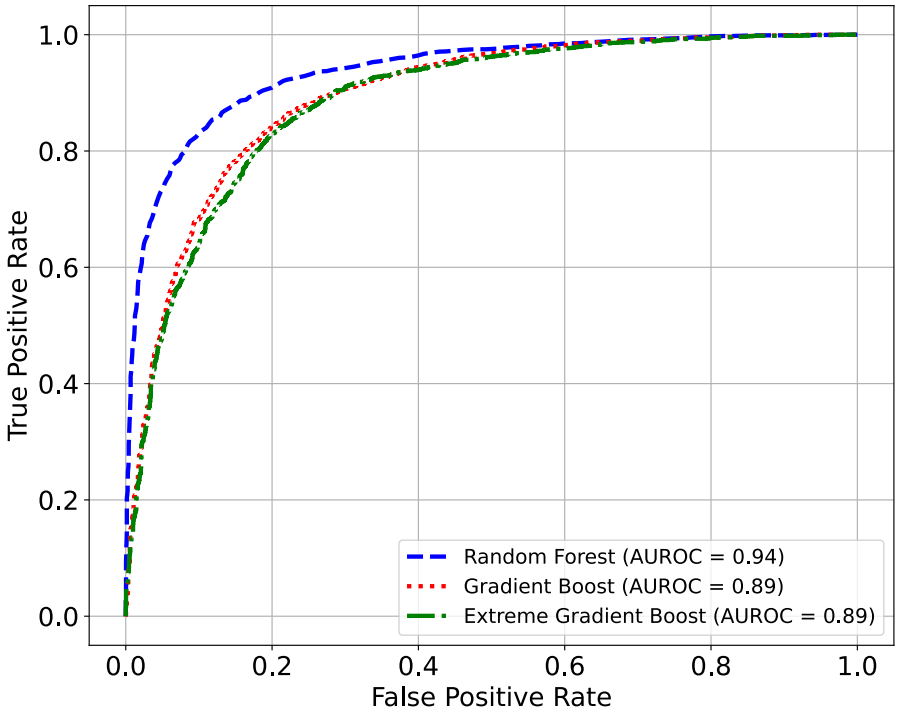


Figure 4.4: AUROC of models with undersample technique

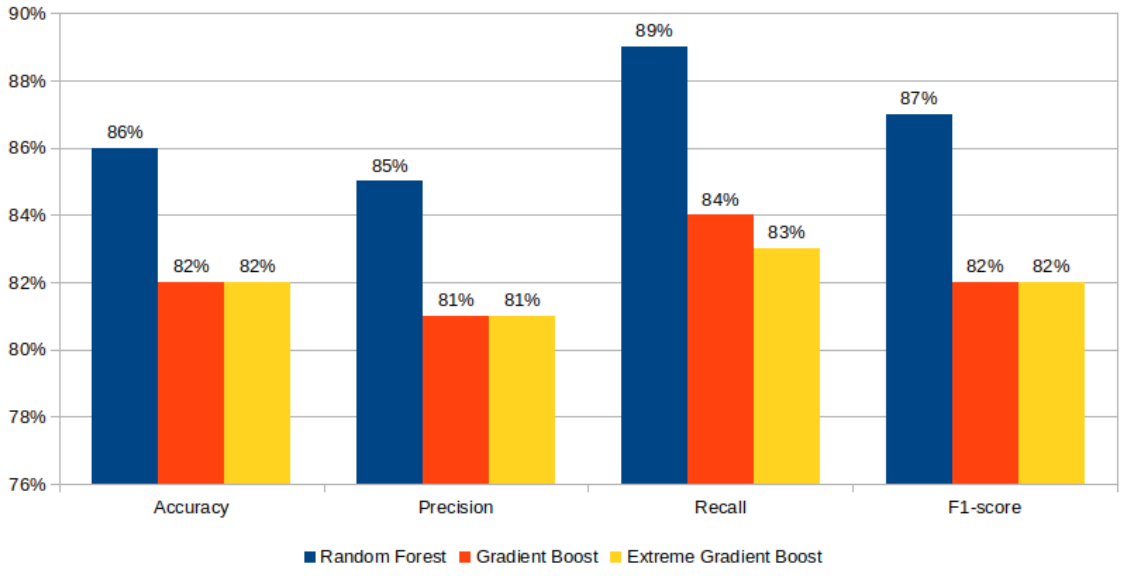


Figure 4.5: Performance comparison of models with undersampling method

4.2.2 Cost-Sensitive Learning

Figure 4.6 depicts the AUROC for our models where class imbalance was handled by cost-sensitive learning. This works by providing higher weights to the instances of the minority class. It demonstrates that the RF model and XGBoost both achieved an AUC score of 0.98 On the other hand, standard gradient boost received a score of 0.91. However, the superior performance of XGBoost model is illustrated in Figure 4.7 by its highest F1-score of 0.73.

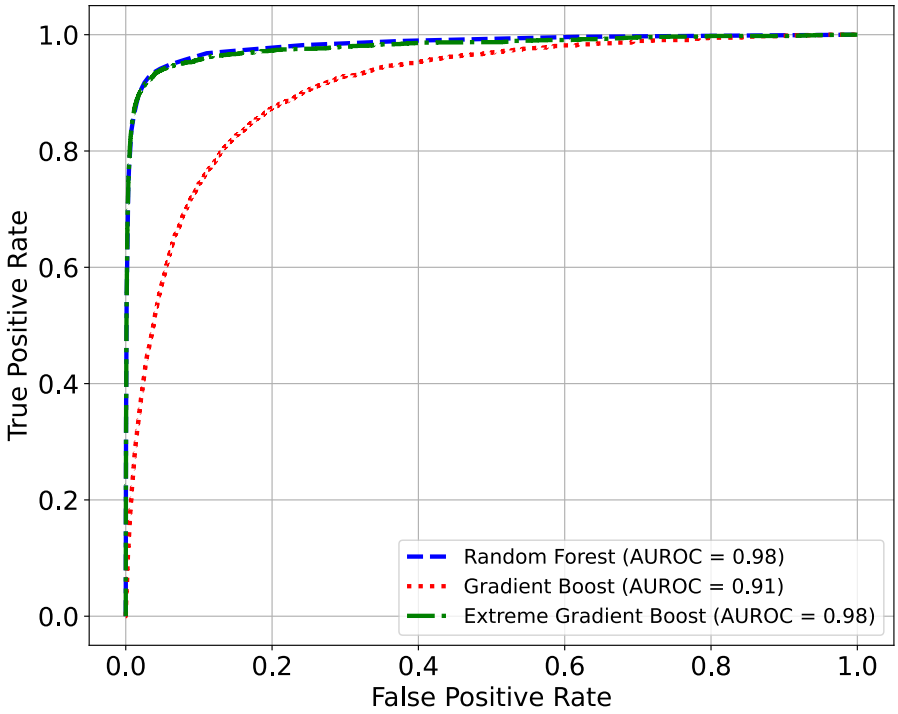


Figure 4.6: AUROC of models with cost-sensitive learning technique

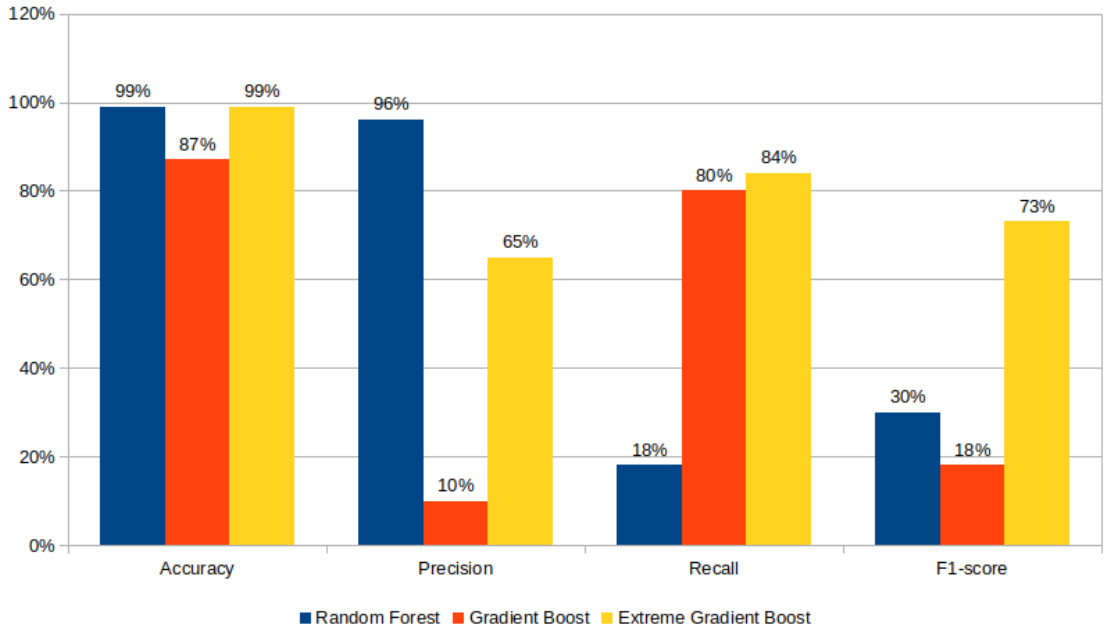


Figure 4.7: Performance comparison of models with cost-sensitive learning method

4.2.3 Synthetic Minority Oversampling Technique (SMOTE)

Figure 4.8 depicts the AUROC for models where class imbalance was handled by SMOTE. Here, it is seen that all the models achieved perfect AUC of 1, demonstrating near perfect capability of distinguishing power between target classes. This can be attributed to the fact that the synthetic samples are somewhat similar to the actual samples which enables

the models to achieve perfect AUC. Real-world data shows variation which will lead to lower AUC score. Figure 4.9 presents a comparative analysis of the three models based on accuracy, precision, recall and F1-score.

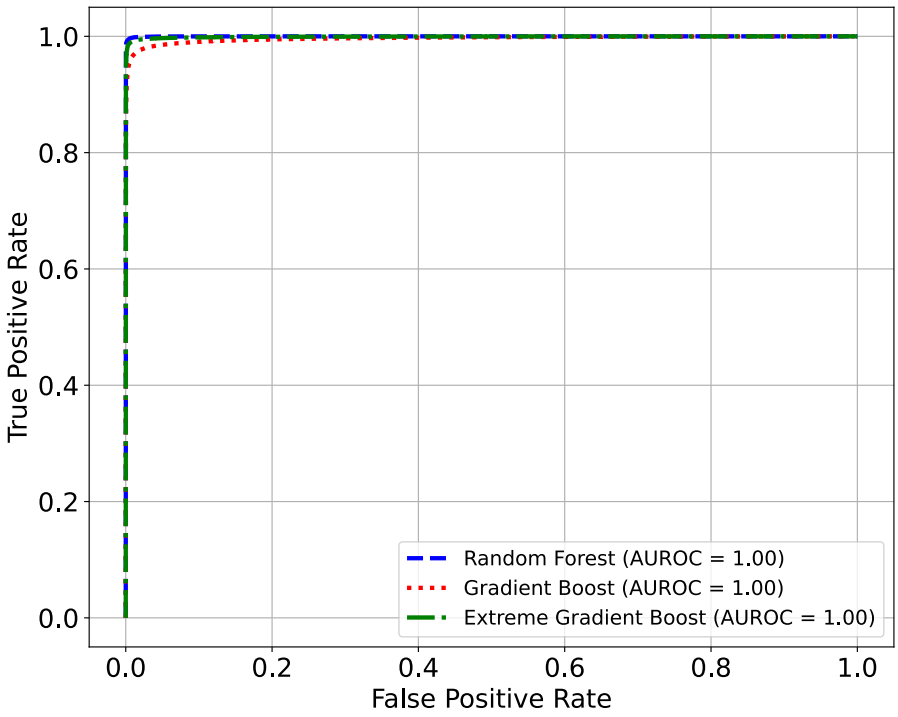


Figure 4.8: AUROC of models with SMOTE

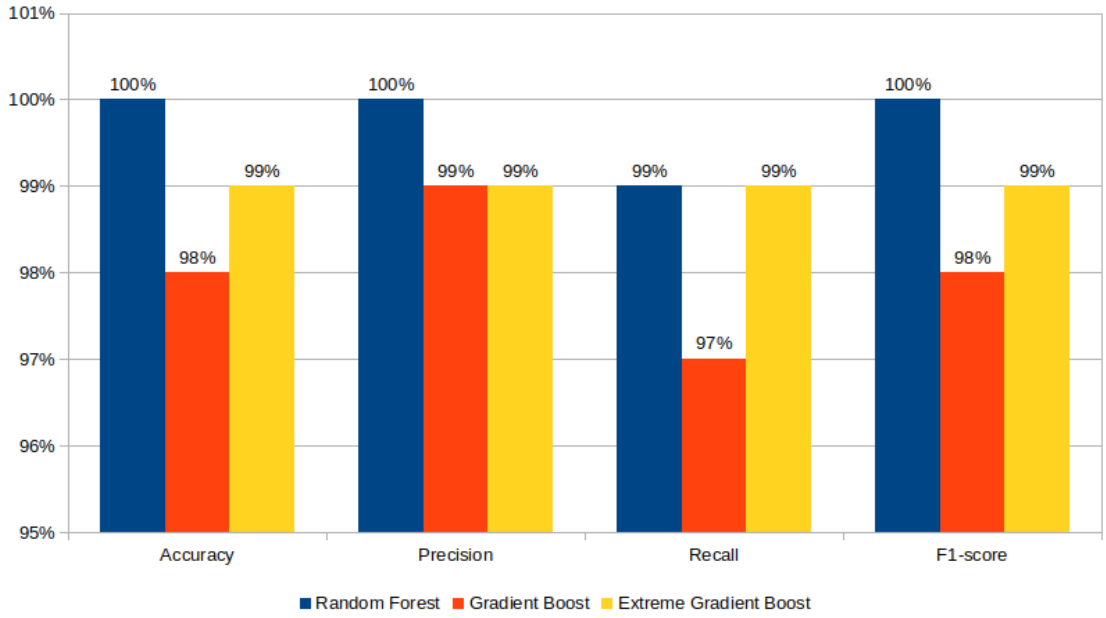


Figure 4.9: Performance comparison of models with SMOTE method

4.2.4 Adaptive Synthetic Sampling (ADASYN)

The discriminating power of the models with ADASYN technique is depicted via the AUROC as shown in Figure 4.10. As was seen in Figure 4.8, the models again show perfect score of 1. This fact can be attributed to the similarity of the artificially generated samples.

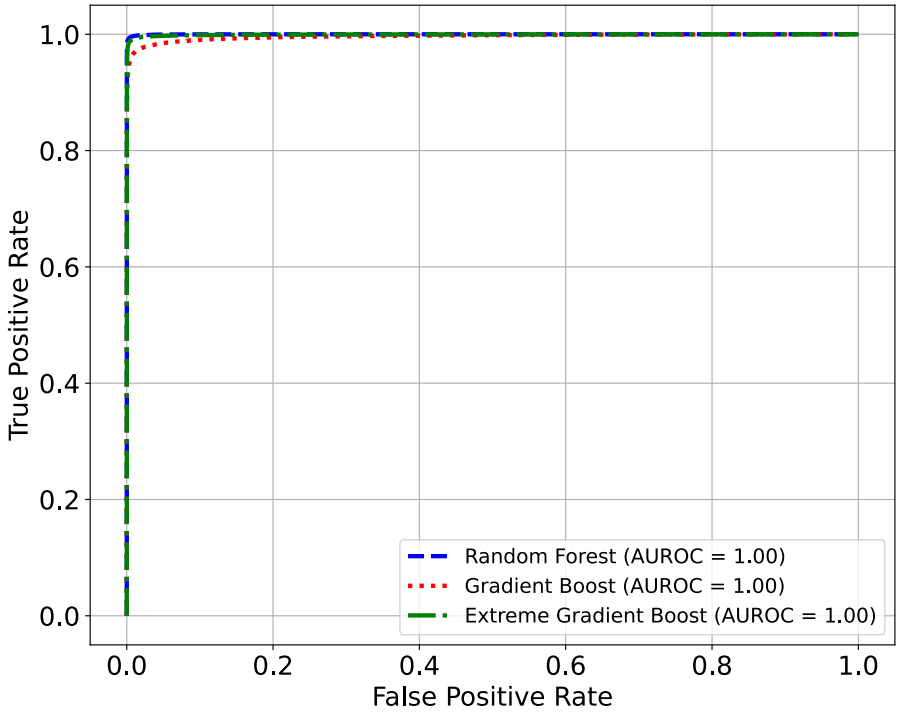


Figure 4.10: AUROC of models with ADASYN

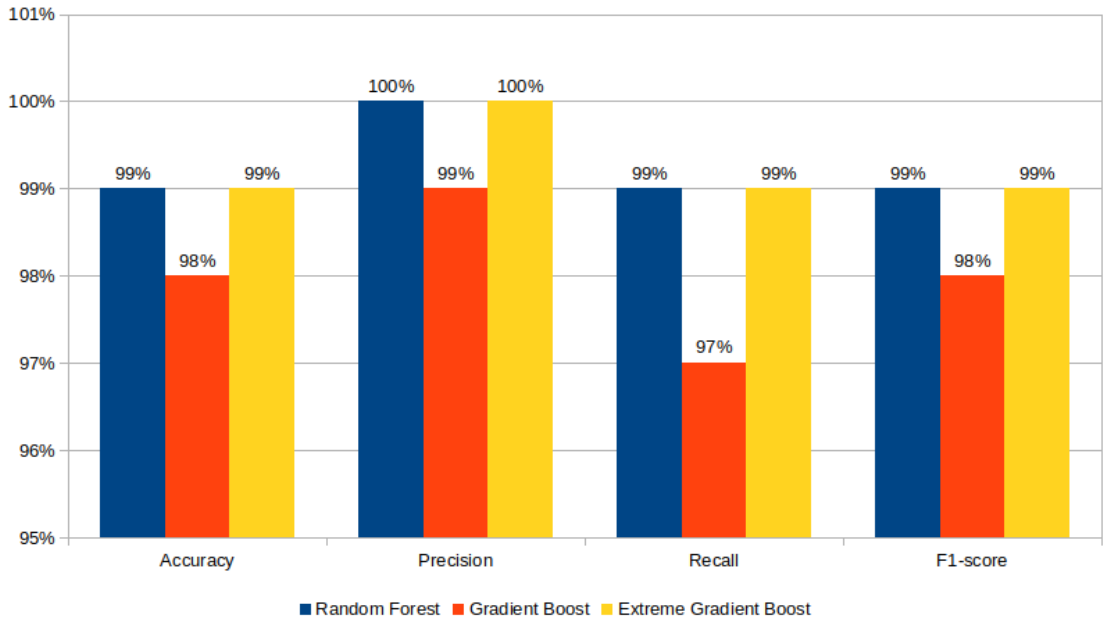


Figure 4.11: Performance comparison of models with ADASYN method

Figure 4.11 summarizes the evaluation results for the three models across four key metrics: accuracy, precision, recall, and F1-score. As was evident from the AUROC of these models, the bar chart also shows near perfect score. Such high scores are potentially due to the synthetic samples being highly representative of the original dataset.

4.2.5 Non-Overlapping Subset Ensemble

Figure 4.12 shows the performance of our models with our proposed non-oversampling subset ensemble technique. Here, the gradient boost model has a lower score of 0.94 than the other two models whose AUC score is 1. However, the perfect scores achieved by the Random Forest and Gradient Boost models may be attributed to the repeated exposure to positive samples during training, as the models attempted to learn patterns from distinct subsets of the negative class.

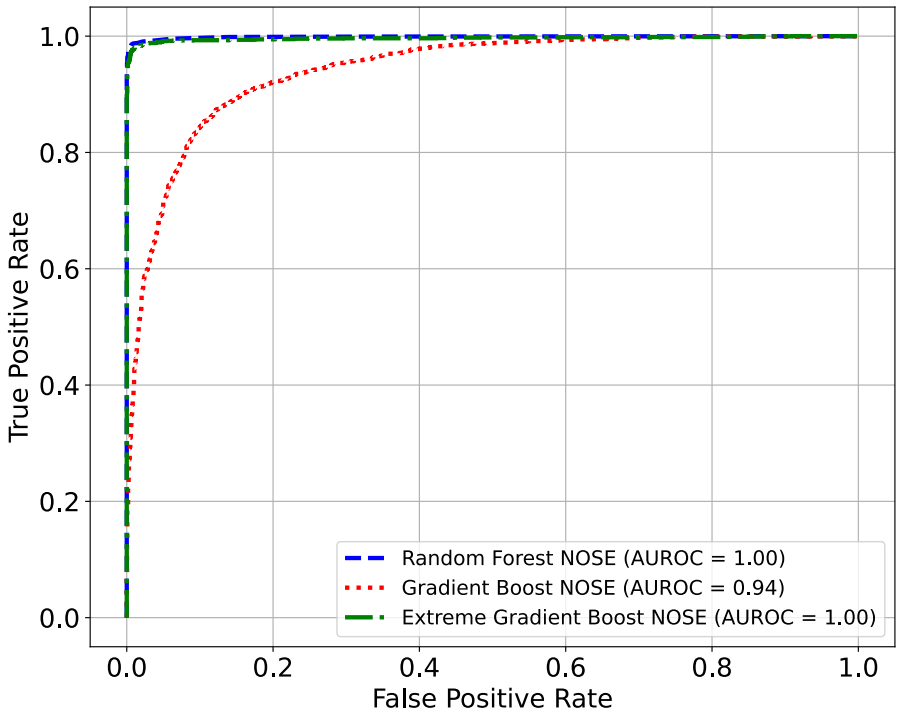


Figure 4.12: AUROC of models with NOSE

The bar chart in Figure 4.13 illustrates the comparative performance of three machine learning algorithms—Random Forest, Gradient Boost, and Extreme Gradient Boost (XGBoost)—across four key evaluation metrics: accuracy, precision, recall, and F1-score. Among the three models, Random Forest consistently demonstrates superior performance. It achieves the highest accuracy at 90%, compared to 81% for Gradient Boost and 83% for XGBoost. In terms of precision, Random Forest attains 85%, whereas both Gradient Boost and XGBoost yield 80%. Notably, Random Forest exhibits a recall of 97%, significantly outperforming Gradient Boost (84%) and XGBoost (87%). Similarly, for the F1-score,

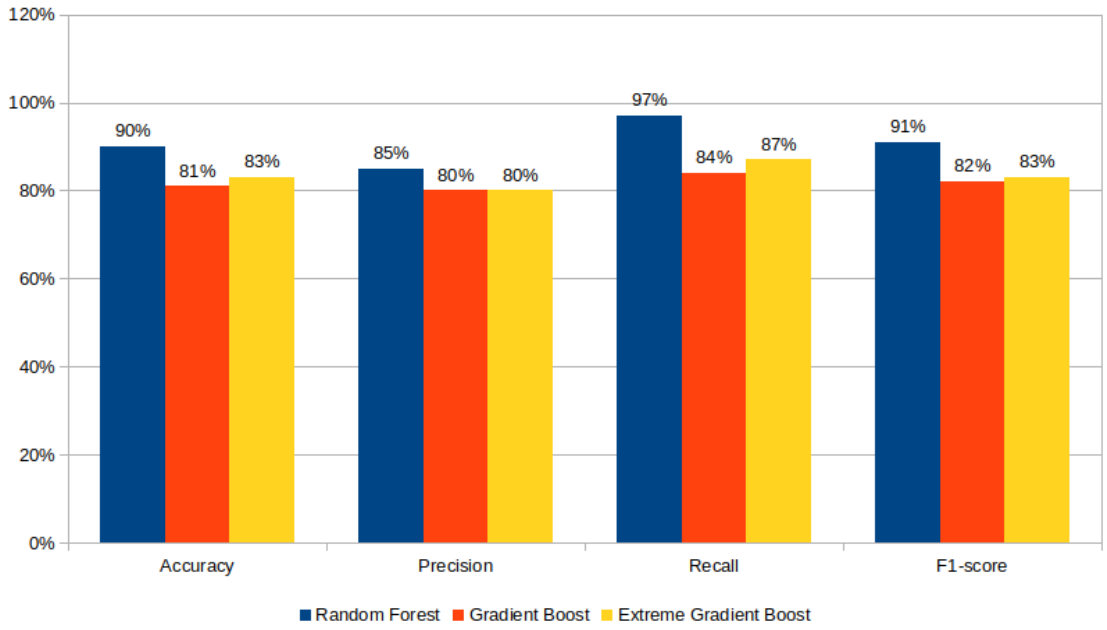


Figure 4.13: Performance comparison of models with Non-Overlapping Subset Ensemble

Random Forest scores 91%, while Gradient Boost and XGBoost reach 82% and 83%, respectively. These results suggest that Random Forest offers more robust and reliable classification performance across all evaluated metrics, making it a preferable choice in applications where both accuracy and generalization are essential.

4.3 Comparison Between Models Based on Missing Value Handling

Following sections illustrate the performance of our models where a number of approaches relevant to the dataset in this work were undertaken. In these scenario, class imbalance was handled via random downsampling. Random downsampling was chosen over other methods because it was computationally less expensive while achieving decent score. Feature selection was done via analysis of variance technique where the 50 most releveant features were chosen.

4.3.1 Last Observation Carried Forward (LOCF)

As shown in the ROC curve of Figure 4.14, the Random Forest model demonstrates the highest classification performance with an AUROC of 0.94, outperforming both Gradient Boost and XGBoost each of which scored 0.89. This indicates that Random Forest is more effective at distinguishing between positive and negative cases. These results align with earlier evaluation metrics and further support the model's robustness for this classification

task. The Bar chart in Figure 4.15 summarizes the performance of the models across the four key evaluation metrics, namely, accuracy, precision, recall and F1-score. The random forest exhibits superior performance to other models in all aspects.

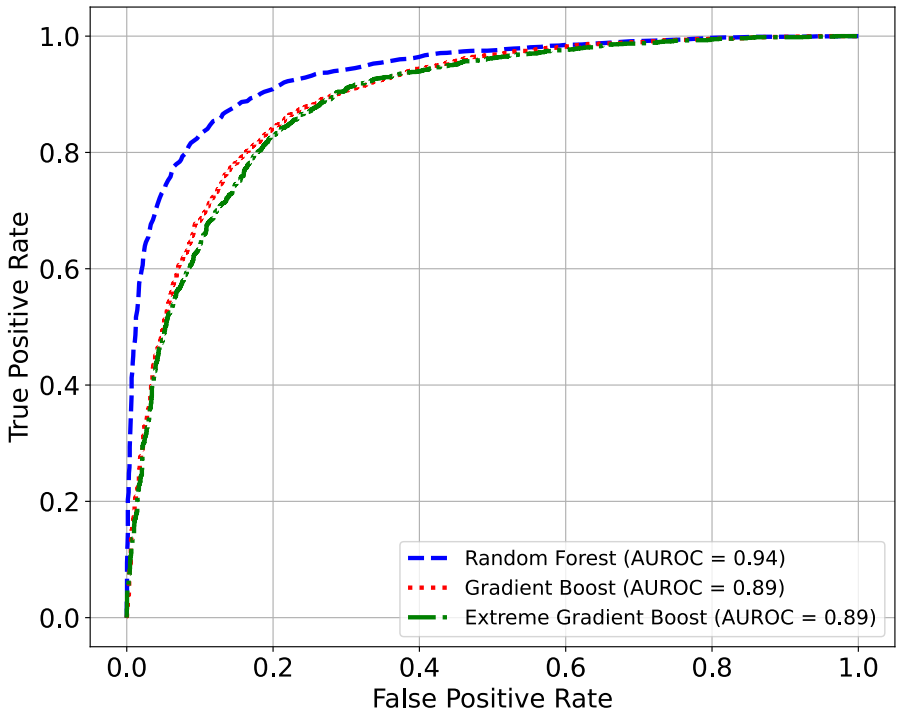


Figure 4.14: AUROC of models with LOCF

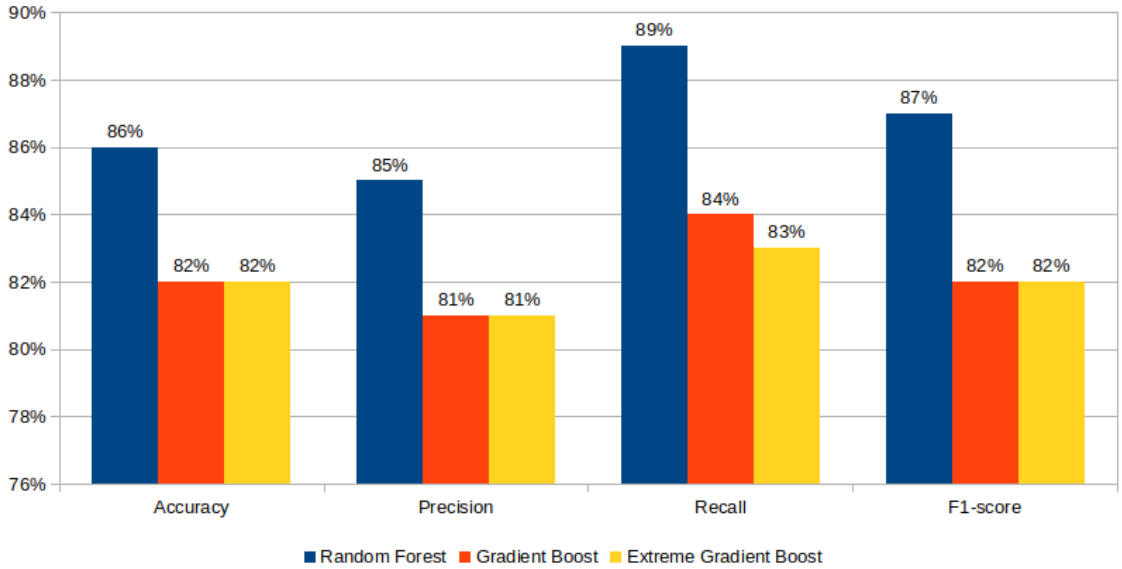


Figure 4.15: Performance comparison of models with LOCF

4.3.2 Constant Imputation

In this method, the missing values of each feature were filled by a constant value based on their age and gender. The AUC scores of our three tree-based models from Figure 4.16

show a slight decline in the performance, indicating that constant imputation method adds noise rather than useful information even if it is based upon age and gender. The random forest retains the topmost position with a score of 0.93. An overview of the models' performance is shown in Figure 4.17. From this figure, we can see that the two variants of the gradient boosting model performed very close to each other whereas the random forest model obtained the highest score in all four metrics: accuracy, precision, recall and F1-score.

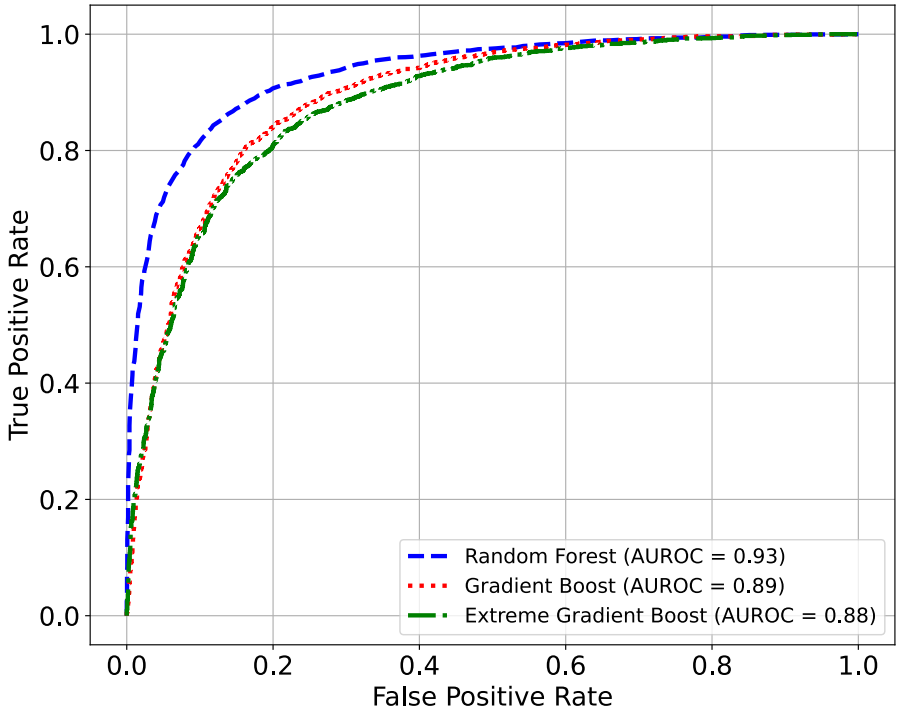


Figure 4.16: AUROC of models with constant imputation

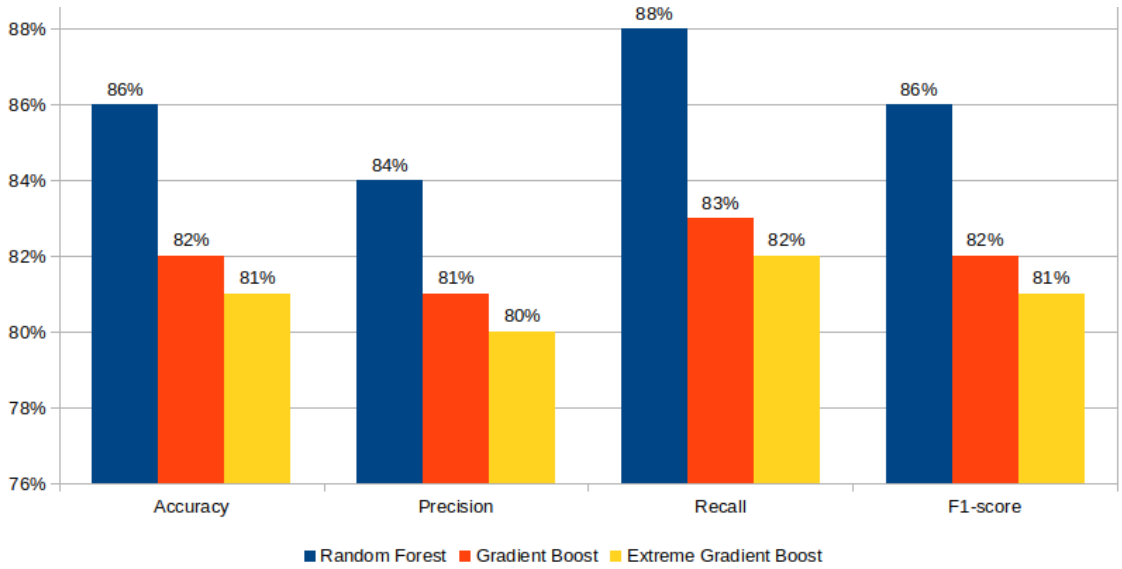


Figure 4.17: Performance comparison of models with constant imputation

4.3.3 Multiple Imputations by Chained Equations

The subsequent figures (Figure 4.18 and 4.19) show the AUROC and performance comparison of our ensemble models, namely, random forest, gradient boost and its extreme variant with missing value addressed via MICE method. Here, random forest outperformed the other two algorithms in terms of discriminating capability and all the performance criteria. XGBoost stands second and the standard gradient boost came out last. However, random forest requires longer prediction time and its storage requirement is larger compared to the other models, making it a difficult choice for deploying on resource-constrained setup.

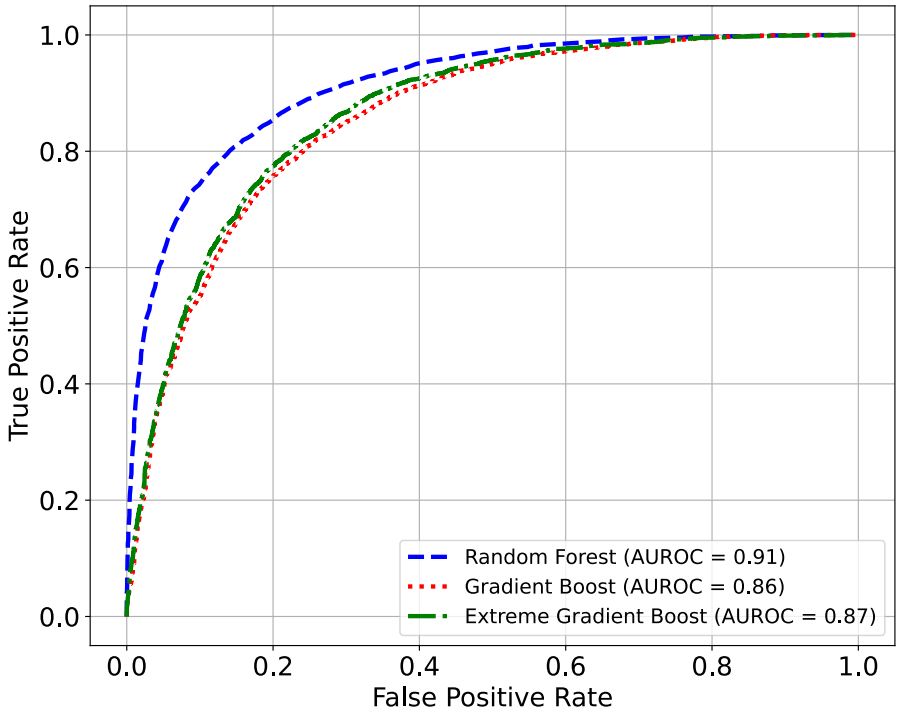


Figure 4.18: AUROC of models with MICE

4.3.4 No Imputation

Figure 4.20 shows the performance of extreme gradient boost with no imputation. It can be seen that this model still performed very good even with missing values. This inherent capability of the XGBoost model makes it robust to missingness in data which is common in real world. Moreover, this ability gives this model edge over other existing algorithms because it reduces the computational complexity and makes it more suitable for deploying on edge devices. The bar chart in Figure 4.21 details the accuracy, precision, recall and F1-score of the XGBoost model. The high recall of 89% along with a precision of 82% of the model indicates its capability to reduce both false positives and false negatives while avoiding the overhead of missing data imputation.

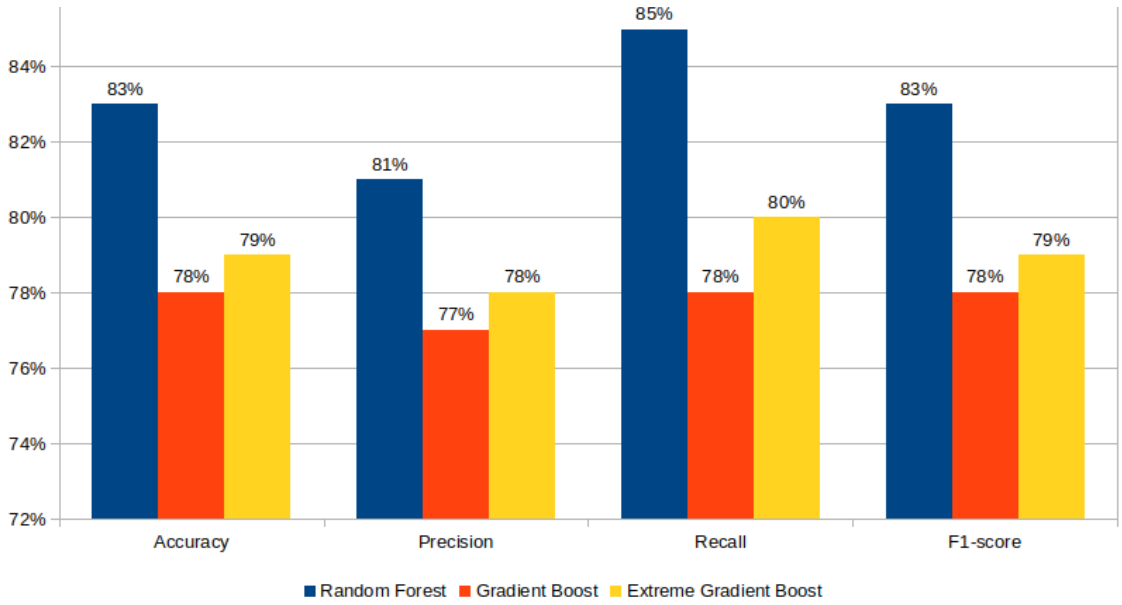


Figure 4.19: Performance comparison of models with MICE

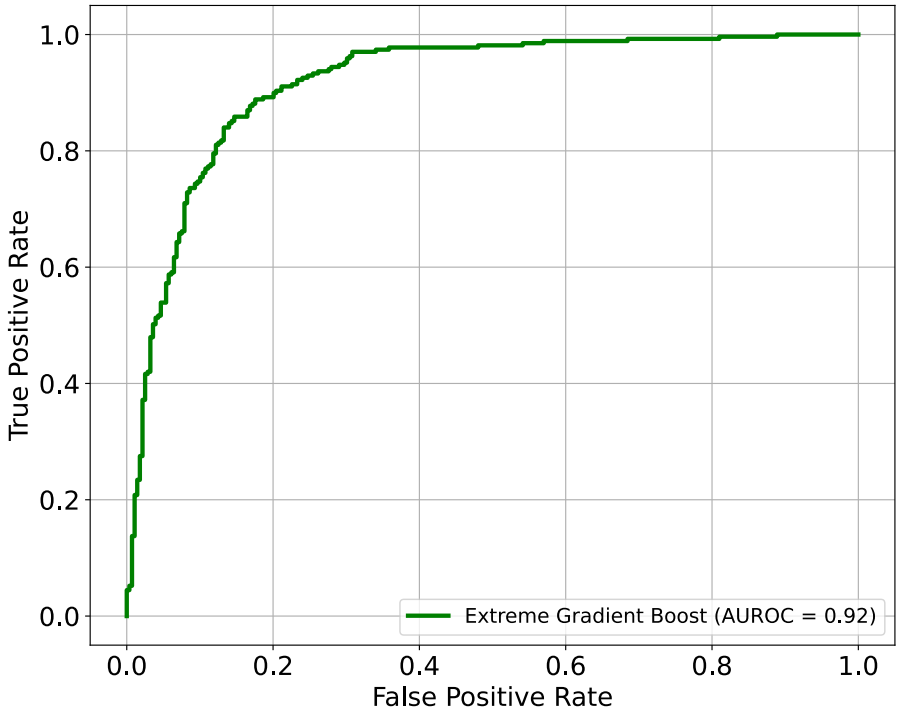


Figure 4.20: AUROC of extreme gradient boost model with no imputation

4.4 Comparison Between Models Based on Feature Selection

While performing feature selection, we used random downsampling to tackle class imbalance issue and forward filling to impute missing values.

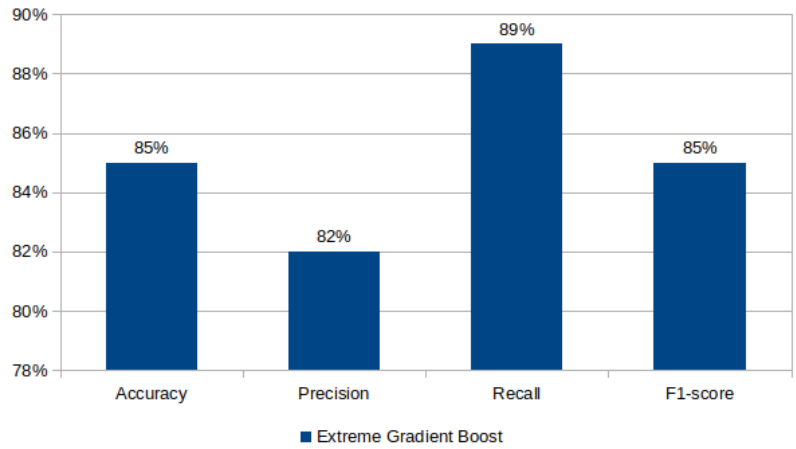


Figure 4.21: Performance of XGB with no imputation

4.4.1 Variance Threshold

Figure 4.22 shows the performance of our models trained on features selected via variance threshold method, where we chose the typical threshold of 0.5. It shows that random forest outperformed the others. However, extreme gradient boost still shows decent performance with low disk space requirement.

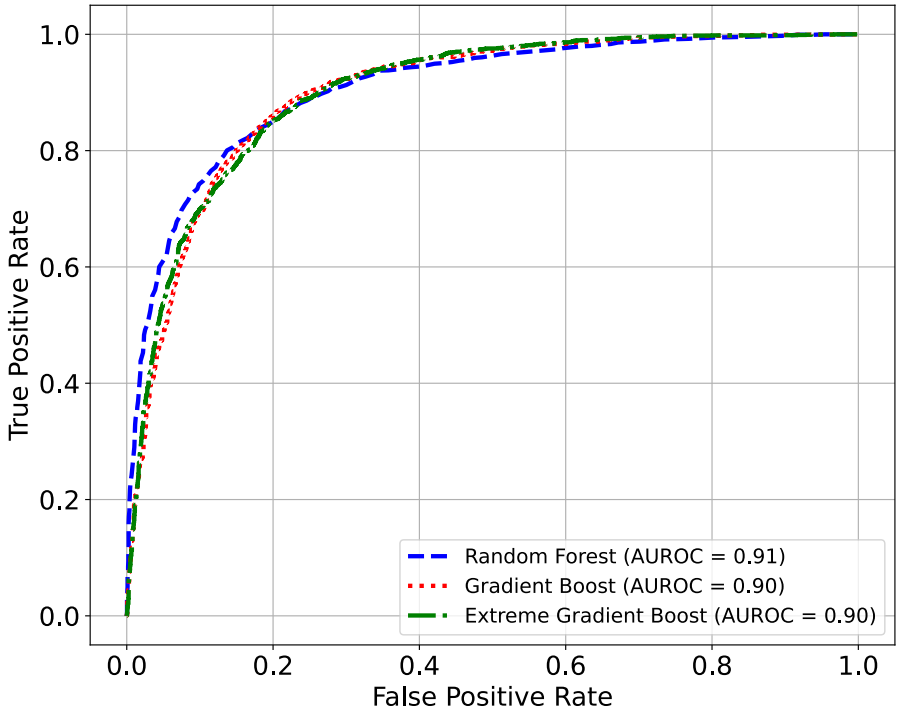


Figure 4.22: AUROC of different models with variance threshold feature selection

The bar chart in Figure 4.23 shows the better performance of the XGBoost model, specially in terms of recall which is very important for our sepsis diagnosis case.

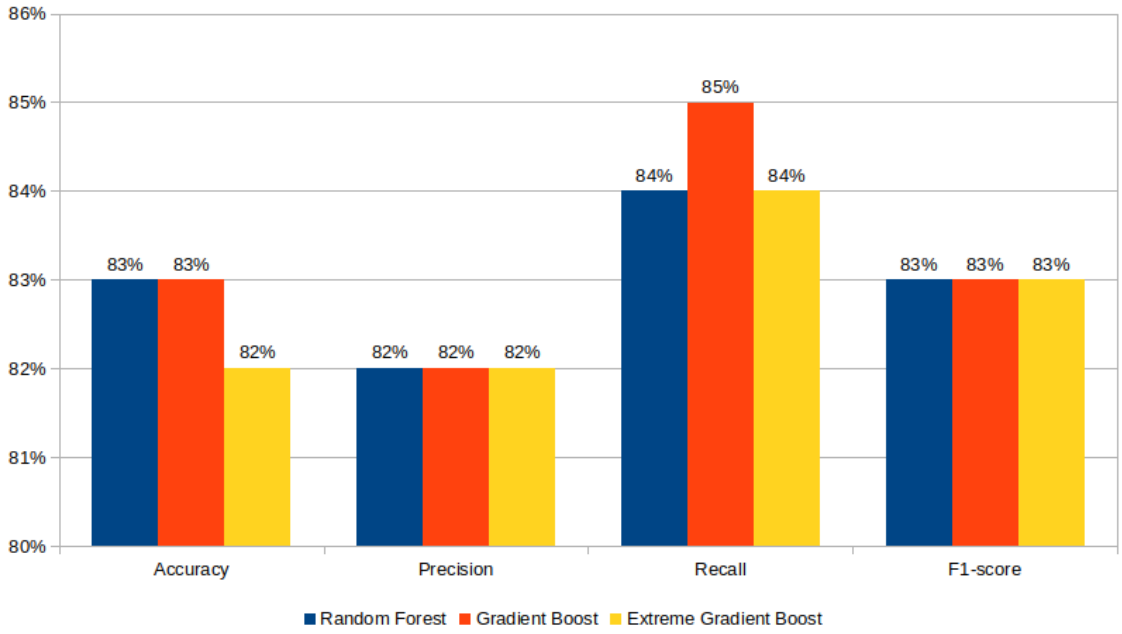


Figure 4.23: Performance of different models with variance threshold feature selection

4.4.2 Analysis of Variance

Figure 4.24 illustrates the performance of the models with ANOVA F-value based feature selection method. Here, we selected 50 features from the total 181 features. All models demonstrated satisfactory performance, with Random Forest attaining the highest AUC value of 0.94.

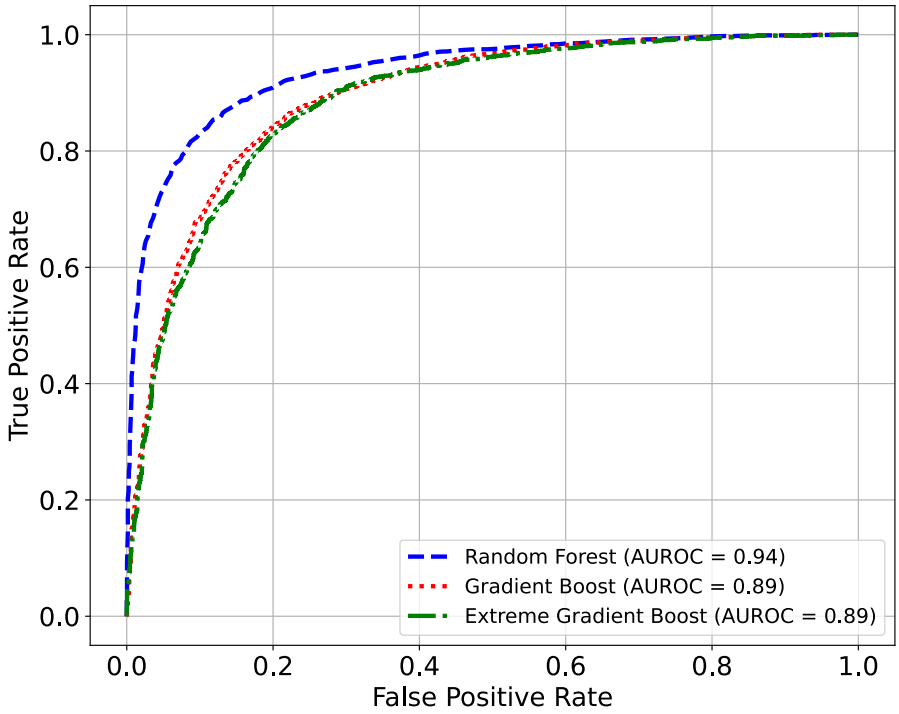


Figure 4.24: AUROC of models with ANOVA

Figure 4.25 shows the performance comparison of the models with ANOVA F-value based feature selection. Here, random forest shows superior performance to other two models in all the performance metrics.

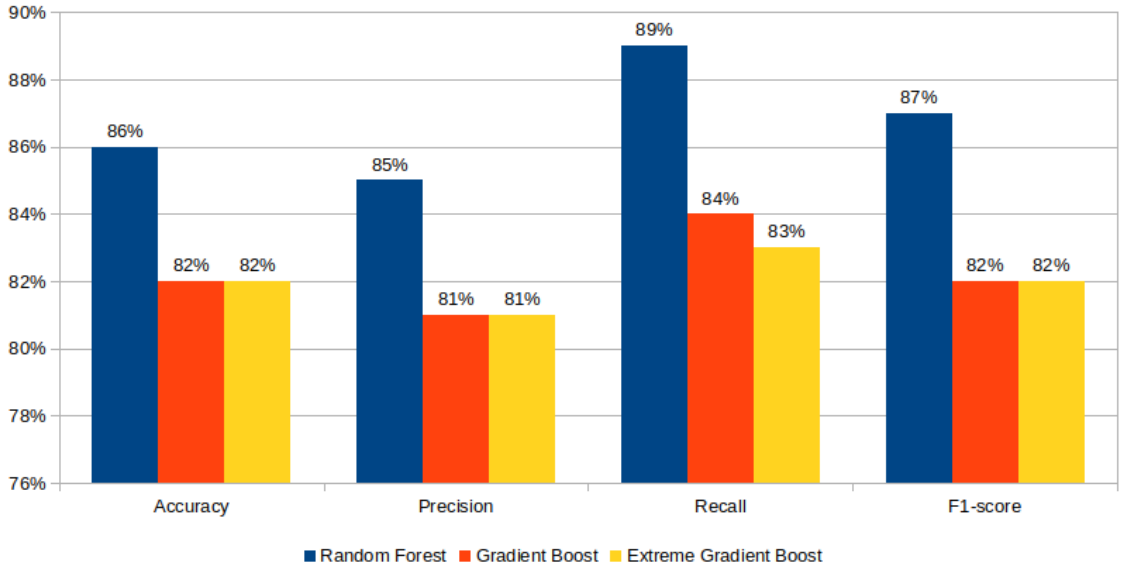


Figure 4.25: Performance comparison of models with ANOVA feature selection

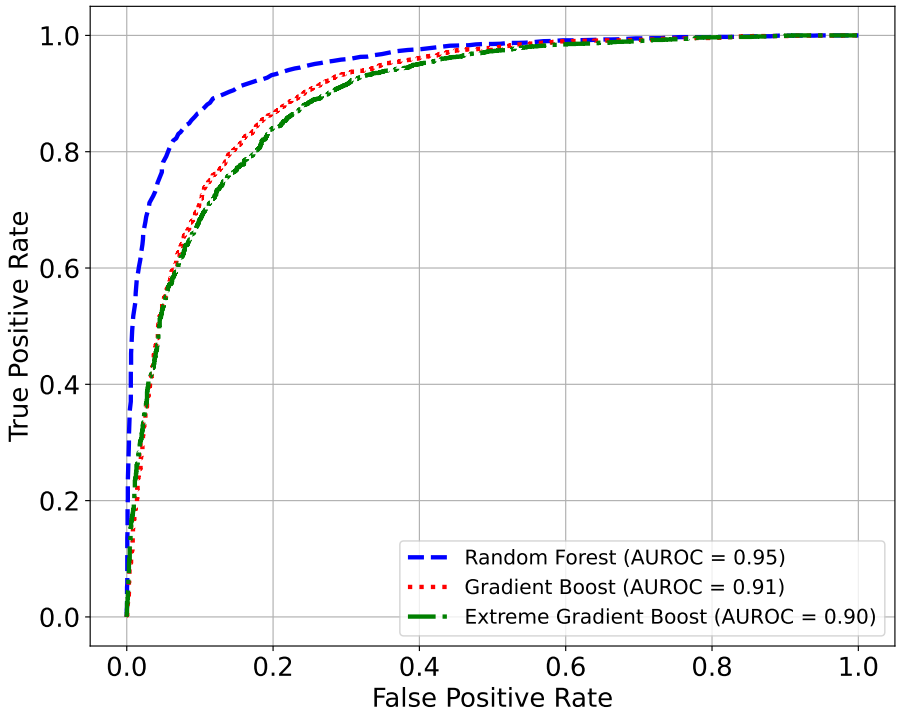


Figure 4.26: AUROC of models with RFE

4.4.3 Recursive Feature Elimination (RFE)

Figure 4.26 shows the AUROC for the three models trained on features selected via recursive feature elimination. 50 features were selected and 10 features were eliminated

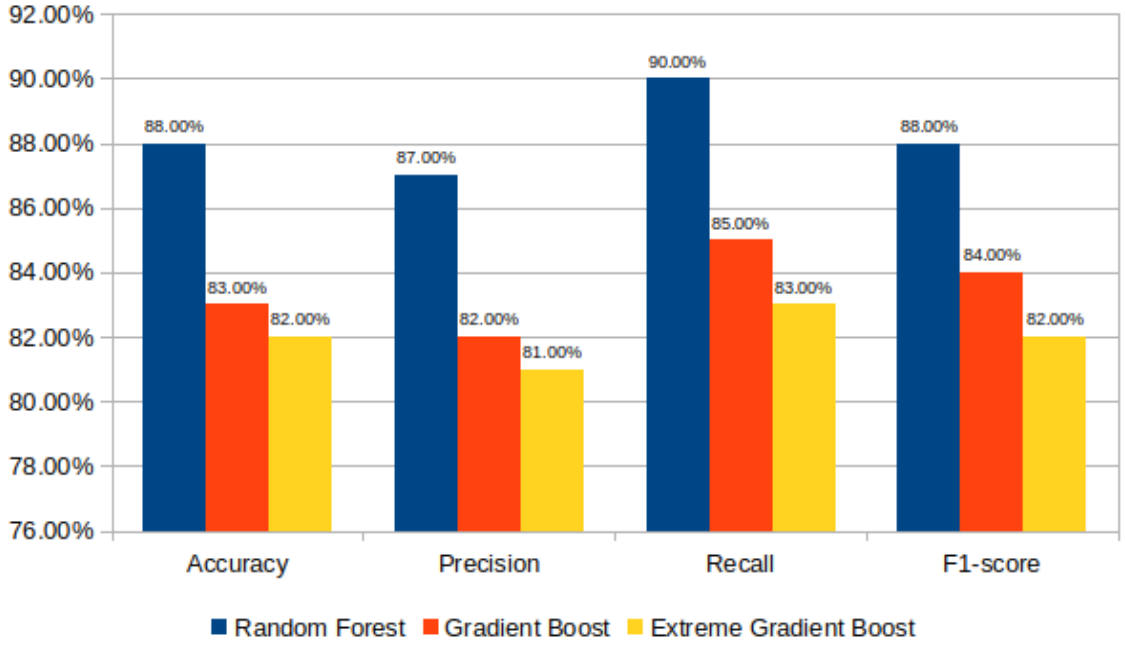


Figure 4.27: Performance comparison of models with RFE

at each step to keep the computational overhead low. Random forest performed best with an AUC score of 0.95.

The bar chart in Figure 4.27 depicts the performance of the models with RFE. Unlike the previous feature selection method, XGBoost performed worse than the other two models in this case. Nevertheless, the performance is still satisfactory.

4.5 Performance with CNN

Figure 4.28 and 4.29 depict the learning curves for our modified MiniVGGNet architecture for a 3-hour and 6-hour time window, respectively. Both of them show overfitting past 20 epochs.

Figure 4.30 shows the performance of the MiniVGGNet architecture on our downsampled and processed 2-dimensional patient data. The model trained and evaluated on 3-hour window shows overall better performance in all the performance metrics except precision by a small margin. The AUROC of the Figures 4.31 and 4.32 illustrate the distinguishing capacity of these two models. We can see that the MiniVGGNet trained on images created from a 3-hour timestep performed better with an AUC score of approximately 0.76 whereas that of the model trained on 6-hour 2-dimensional data was 0.74. However, due to the lower dimension of the data in the smaller time window, the CNN architecture had to be slightly modified by dropping the final max pooling layer. It was done to ensure that the dimensions of the subsequent layers properly match and no negative dimension arises in calculation.



Figure 4.28: Training curve for modified MiniVGG architecture with 3-hour window



Figure 4.29: Training curve for modified MiniVGG architecture with 6-hour window

4.6 Performance with RNN and LSTM

Figures 4.33 and 4.34 illustrate the training curve for accuracy and loss metrics for a simple, custom RNN and LSTM architecture. Unlike the RNN architecture, which overfit quite heavily past 15 epochs, LSTM was substantially resilient to overfitting up until the complete 30 epochs. This shows its ability to learn long term dependencies of the features. Figures 4.35 and 4.36 show the ROC curve for the RNN and LSTM models. The area under curve (AUC) for these models are inferior to the previous models in terms of their distinguishing power between the positive and negative classes.

Figures 4.37 and 4.38 show the training curve for these models utilizing a 6-hour time window. They also show signs of overfitting, with the RNN being greatly affected.

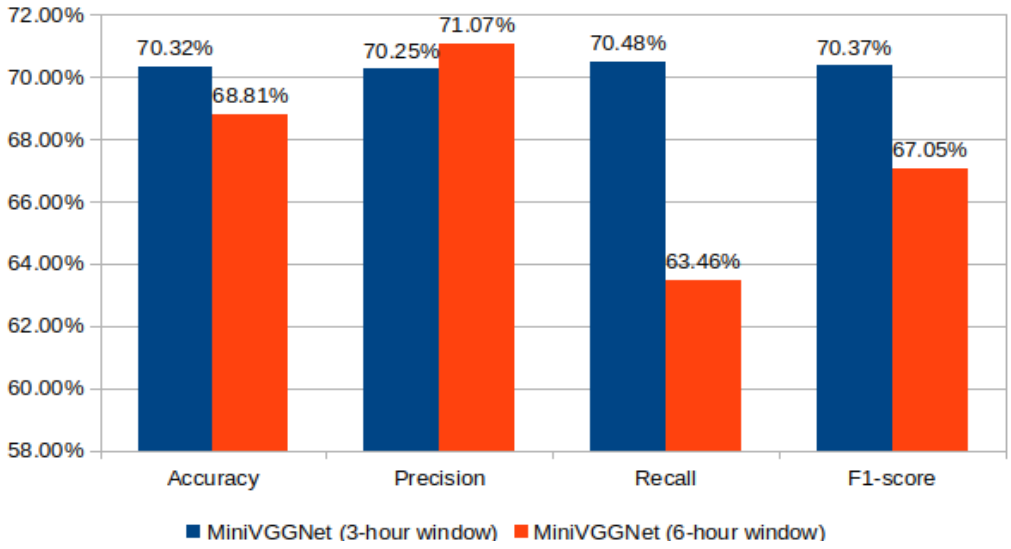


Figure 4.30: Performance comparison of modified MiniVGGNet with varying window lengths

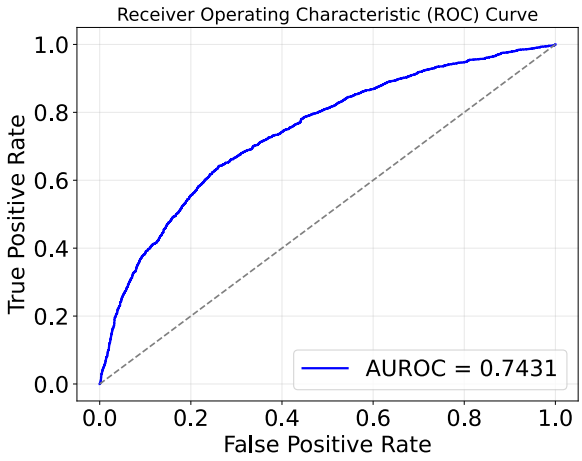


Figure 4.31: MiniVGGNet on undersampled data with timestep of 6 hours

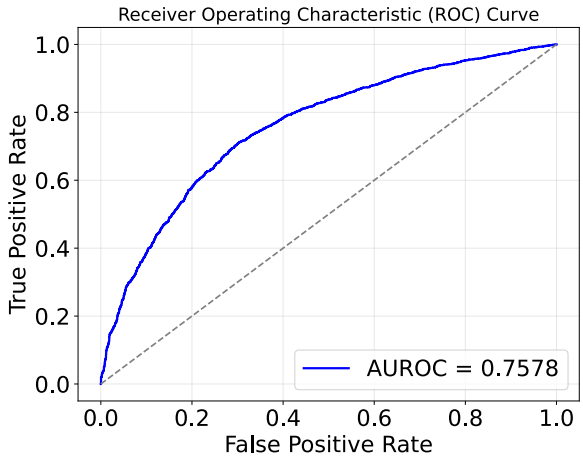


Figure 4.32: MiniVGGNet on undersampled data with timestep of 3 hours

However, from their AUROC curves (Figure 4.39 and 4.40), we can see that they have better differentiating capabilities between target classes.

Figure 4.41 compares the predictive performance of RNN and LSTM models using 3-hour and 6-hour input windows across four key metrics. Both architectures benefit from the longer, 6-hour window, with accuracy rising from 70.38% to 70.82% for RNN with 3-hour window and from 68.29% to 72.30% for LSTM with 3-hour window. Precision, however, declines from 72.64% to 70.21% for RNN on 3-hour window but rises from 69.11% to 71.52% for LSTM. Recall increases markedly, particularly for LSTM (from 66.14% to 74.10%). Overall, the LSTM with a 6-hour window attains the highest F1-score (72.79%), outperforming both the shorter-window LSTM (67.59%) and the corresponding RNN configurations (68.82% and 71.25%), indicating that extended temporal context contribute to more robust classification.



Figure 4.33: Training curve for RNN with 3-hour time window

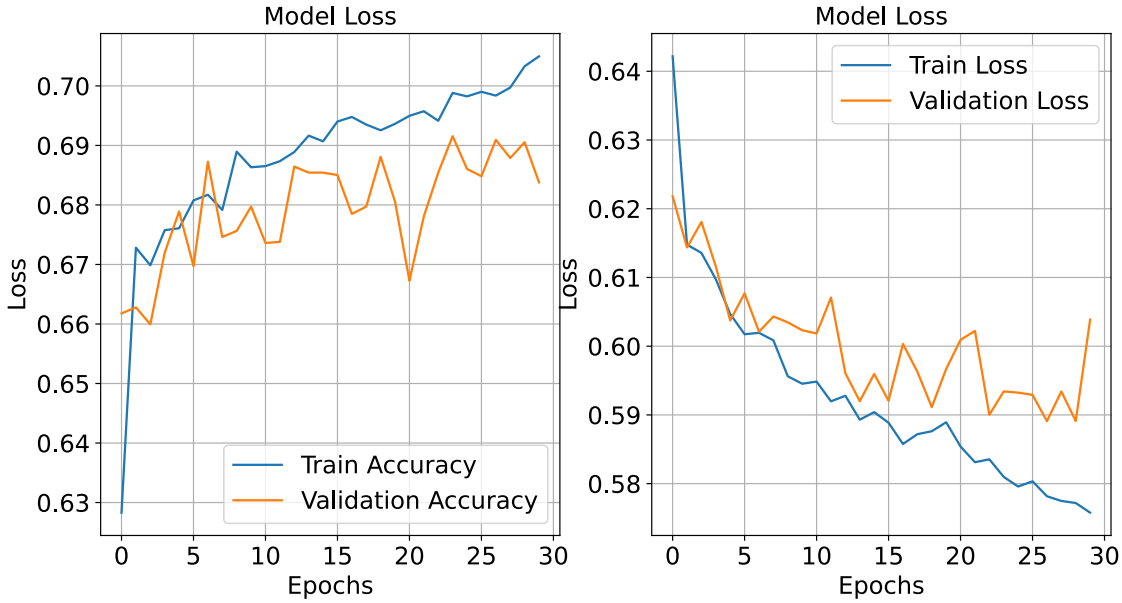


Figure 4.34: Training curve for LSTM with 3-hour time window

4.7 Performance with Federated Learning

Figure 4.42 shows the performance comparison in terms of different metrics among the individual models and the aggregated models.

The visual clearly depicts the superior performance of the aggregated model in accuracy and precision. However, the weighted aggregated model, where weight was determined based on its performance on cross validation recall result, demonstrates a higher recall on the unseen dataset. This is reflected in its highest F1-score among all

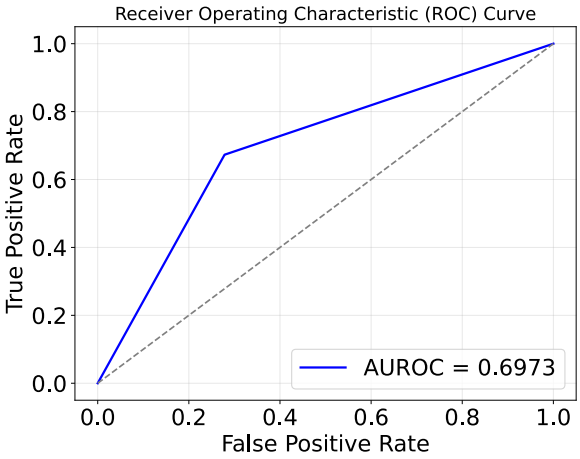


Figure 4.35: RNN with a 3-hour time window

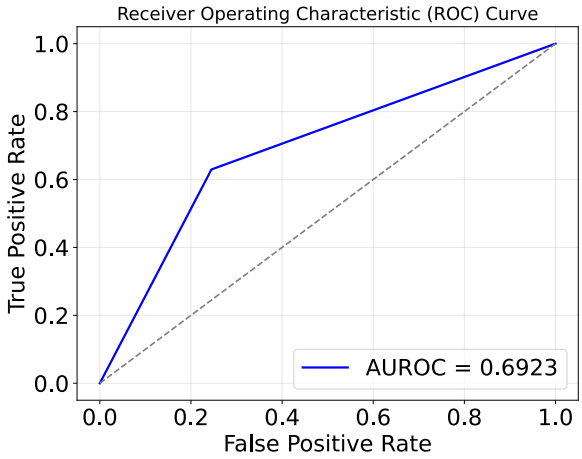


Figure 4.36: LSTM with a 3-hour time window

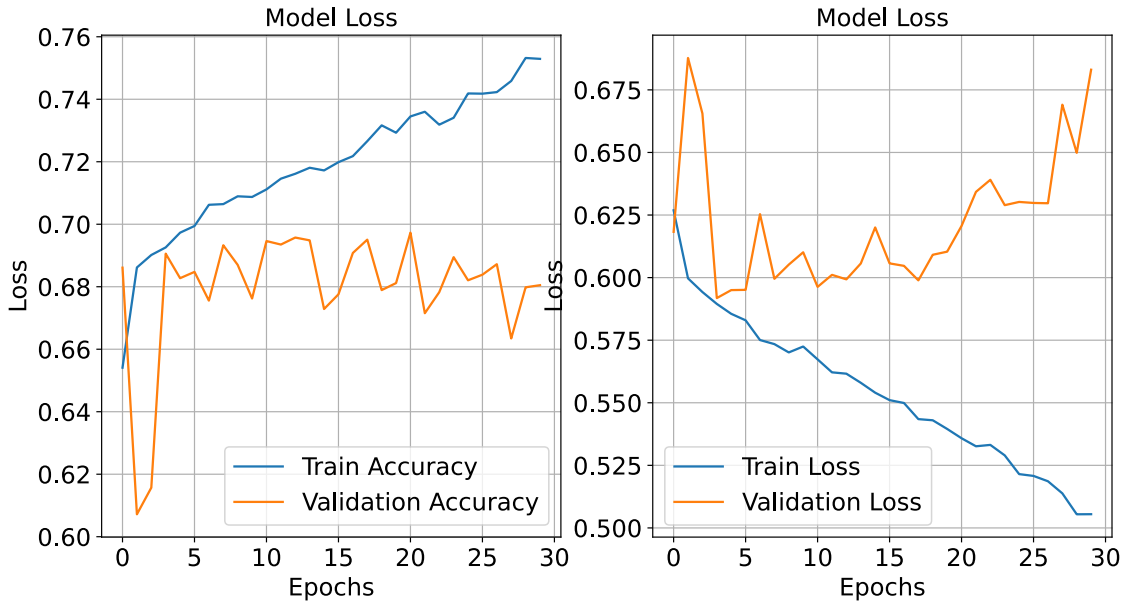


Figure 4.37: Training curve for RNN with 6-hour time window

the models. This picture shows the effectiveness of federated learning by simple merging of trees based on simple rules while maintaining confidentiality.

4.8 Reliability of Probability Estimates

Figures 4.43 to 4.45 illustrate the probability calibration curves obtained for our tree-based models.

Across all three reliability diagrams, the dashed 45° line denotes perfect calibration, where predicted probabilities equal observed event rates. The random forest exhibits a pronounced S-shape: it is over-confident at low predicted risks and under-confident in the



Figure 4.38: Training curve for LSTM with 6-hour time window

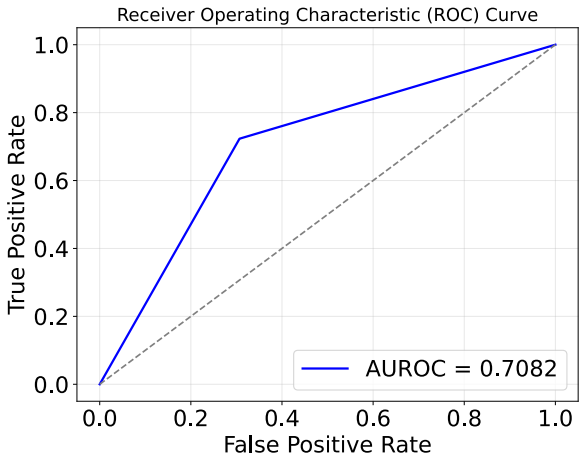


Figure 4.39: RNN with a 6-hour time window

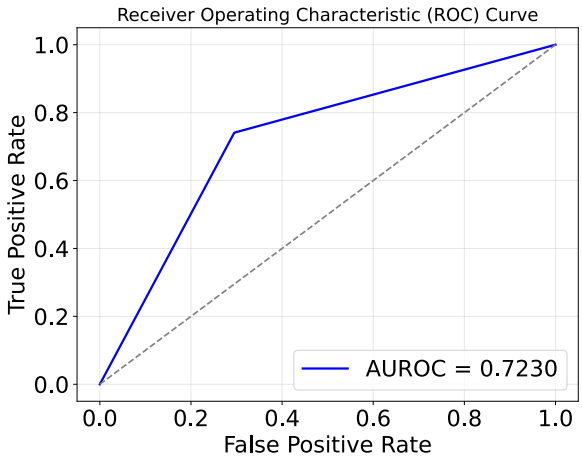


Figure 4.40: LSTM with a 6-hour time window

mid–high range. Both calibration methods correct this behaviour; isotonic gives slightly better alignment near the extremes, while sigmoid yields a smoother, globally consistent mapping. The gradient boosting model is already close to the diagonal in its raw form, and post-hoc calibration produces only marginal adjustments, indicating little room for improvement. The XGBoost model shows mild miscalibration (slight over-confidence at low probabilities and under-confidence around 0.6–0.8); calibration shifts the curve toward the diagonal with modest but consistent gains. Consistent with these visuals, we expect Brier score to decrease meaningfully.

Table 4.1 shows the Brier scores obtained for the calibrated and non-calibratd tree based models. It can be seen that calibration reduces Brier error for all models, most notably for RF (0.118 to 0.105, around 11% improvement), indicating its raw probabilities

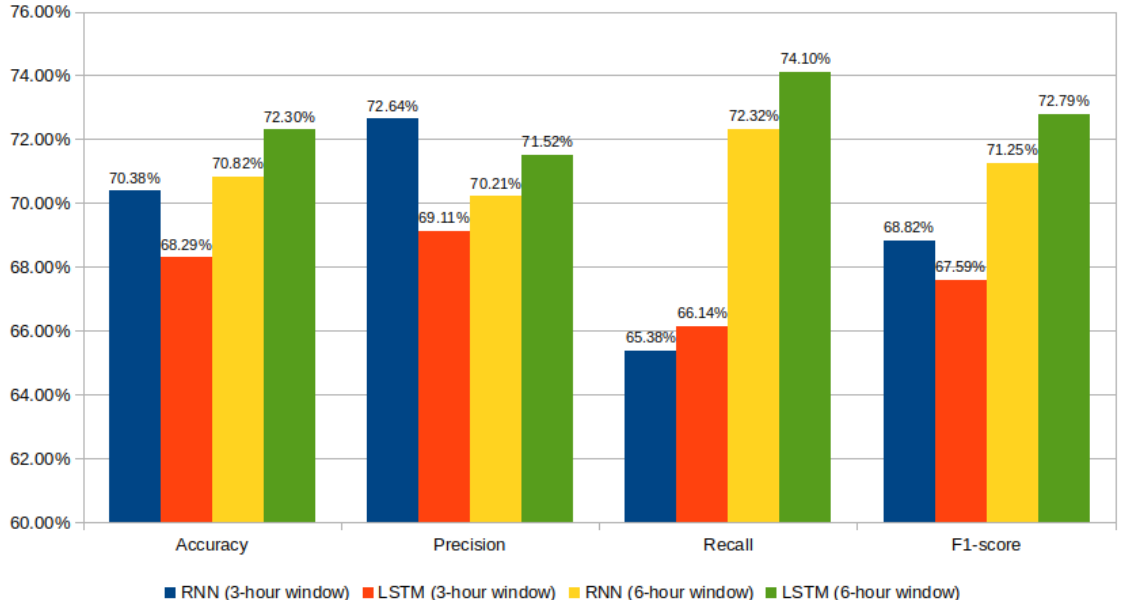


Figure 4.41: Peformance comparison of RNN and LSTM on varying window length

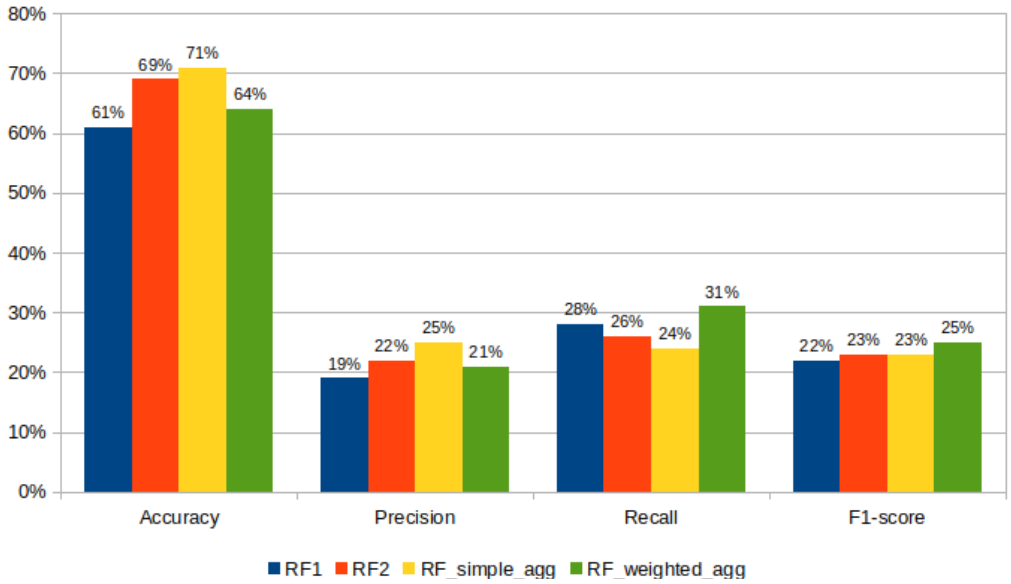


Figure 4.42: Performance comparison among individual model and aggregated model

were miscalibrated. GB and XGB improve only slightly (approximately 3%), consistent with already decent calibration. Sigmoid and isotonic perform identically here. Overall, calibrated probabilities are more reliable for decision making.

Table 4.1: Brier score on the test set

Model	Raw	+ Sigmoid	+ Isotonic	Best	Δ vs Raw (%)
RF	0.118	**0.105**	**0.105**	**0.105**	−0.013 (−11.0%)
GB	0.130	**0.126**	**0.126**	**0.126**	−0.004 (−3.1%)
XGB	0.156	**0.152**	**0.152**	**0.152**	−0.004 (−2.6%)

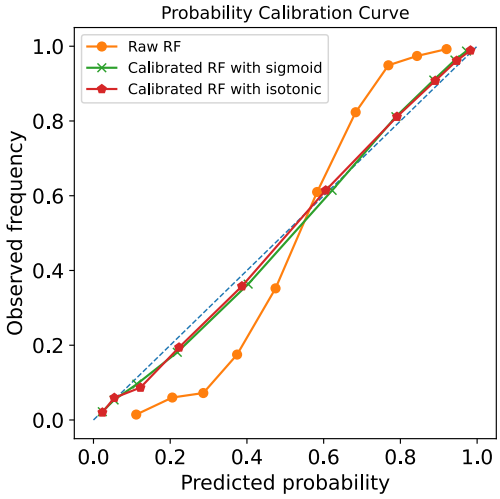


Figure 4.43: Probability calibration curve for random forest

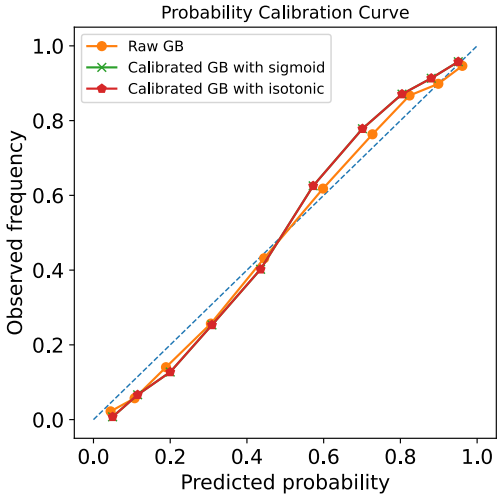


Figure 4.44: Probability calibration curve for gradient boost

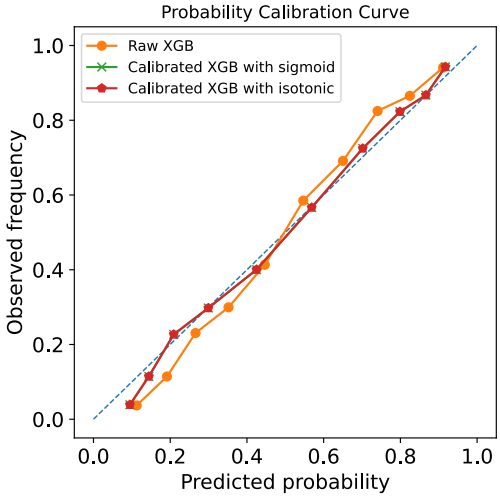


Figure 4.45: Probability calibration curve for extreme gradient boost

4.9 Model Size

Figure 4.46 shows the size of the models. The horizontal axis is in logarithmic scale. It clearly shows that despite the better performance of the random forest model, its disk space requirement is substantially large compared to other two, making it difficult to employ on edge devices.

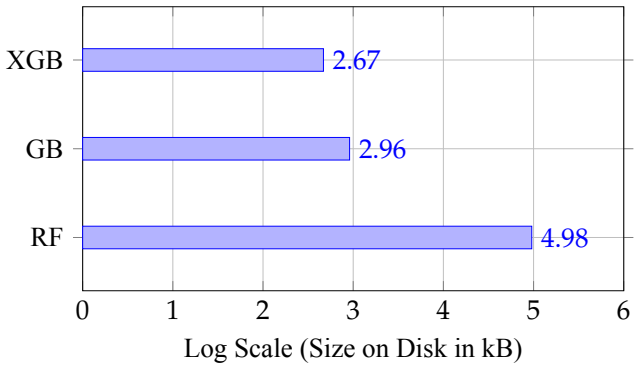


Figure 4.46: Model size comparison on log scale

4.10 Model Performance on MIMIC-IV Demo

Figure 4.47 summarizes sepsis-class recall across models using a lollipop chart.

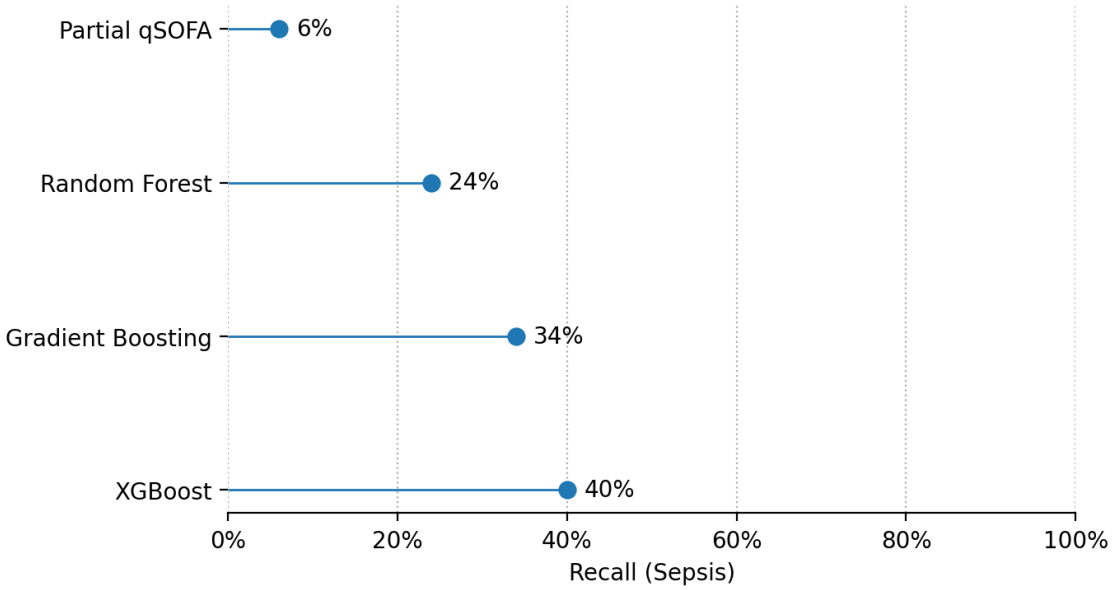


Figure 4.47: Lollipop plot of sepsis-class recall in MIMIC-IV demo database

XGBoost achieved the highest recall (40%), followed by Gradient Boosting (34%) and Random Forest (24%), while the rule-based partial qSOFA baseline detected only 6% of

sepsis cases. The absolute gains over the baseline are +34 pp (XGBoost), +28 pp (GB), and +18 pp (RF), indicating a substantial improvement in sensitivity to sepsis.

Table 4.2 shows an overall comparison of our work with some of the notable previous works. These works used input features ranging from non-invasive vital signs to critical laboratory values which require sophisticated instruments. Only the one based on digital biomarkers relied upon simple vital signs. However, it didn't include any performance metrics which makes it unsuitable for real-world use. Additionally, none of these works explored explainability which is important for ML models used for disease diagnosis. Our work tried to address all of these points while keeping the system prototype minimal.

Table 4.2: Summary of major works in sepsis detection using machine learning

Study	Input Features	Missing Data Handling	Imbalance Handling	Classifier	Performance	Edge Device	Interpretability
[71]	Vital signs	No mention	Undersampling of majority class	TCN	Latency = 143ms Energy = 2.68mJ	ARM Cortex M-33 core	**X**
[17]	Original features, qSOFA, signature of summary statistics	Forward filling	No mention	LightGBM	Utility = 0.321	**X**	**X**
[59]	Vital signs, lab values, demographic info, binary masks	Zero filling	No mention	RNN with DIGRU cell	Utility = 0.323	**X**	**X**
[18]	Original features, ratio and power-based features	Linear interpolation	Random downsampling by classifier	RUSBoost	Utility = 0.309	**X**	**X**
[21]	Original features, measurement frequency, qSOFA, window stats	Forward filling	Undersampling of majority class	XGBoost	Utility = 0.364	**X**	**X**
[66]	Original features, window stats, ratio features, qSOFA	Forward filling	No mention	LightGBM, RNN	Utility = 0.337	**X**	**X**
[37]	Vital signs, lab values, demographic info	Dropped	SMOTE	RF	AUROC = 0.94 Accuracy = 86.31% Recall = 83.71%	**X**	SHAP
Our work	Vital signs	Forward filling	Random downsampling	RF	Utility = 0.466 AUROC = 0.94 Recall = 86.49%	Raspberry Pi	SHAP, LIME

Chapter 5

Model Interpretability

This chapter focuses on explaining the predictions made by the machine learning model to ensure transparency and trust in the sepsis diagnosis system. We employ various interpretability techniques, including SHAP-based beeswarm plots, bar plots, and waterfall plots, to visualize the contribution of individual features to the model's output. Additionally, LIME plots are used to provide local explanations for specific predictions. Model-based feature importance is also analyzed to identify the most influential vital signs, supporting clinical relevance and decision-making.

5.1 Beeswarm Plot

The Beeswarm plot has the input features ordered according to the mean absolute shap value. Each instance is represented by non-overlapping dot based on the x-axis, which contains the SHAP values.

As can be seen from the plot in Fig. 5.1 for the random forest model trained on 50 features, the maximum temperature of an expanding window (ranging from current time stamp to the first observation) has the highest impact on the output. Since the higher values of this feature direct towards positive SHAP values, it indicates that a higher value for this feature is indicative of positive outcome. Similarly, we can see that the model attributes higher values of body temperature, maximum MAP of an extending window etc. to positive outcome. However, it can be seen that the trained model attributes lower values of minimum Mean Arterial Pressure, Systolic Blood Pressure etc. of extending window to positive outcome. The trained model emphasizes the expanding window based features although many other features such as difference features, ratio, qSOFA etc. were available. This suggests some sort of memory dependent prediction of the model.

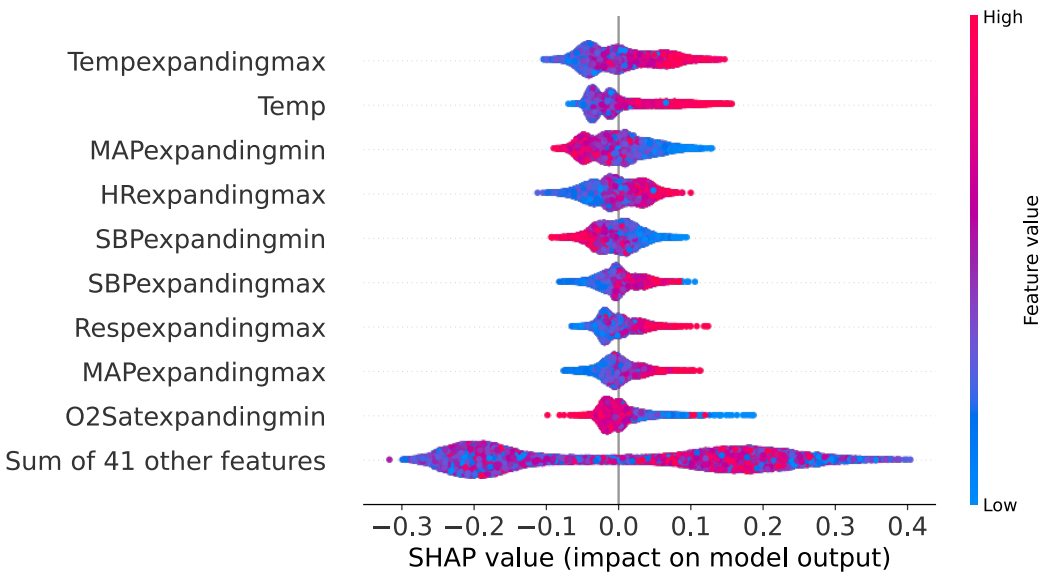


Figure 5.1: Beeswarm plot for random forest model with 50 features

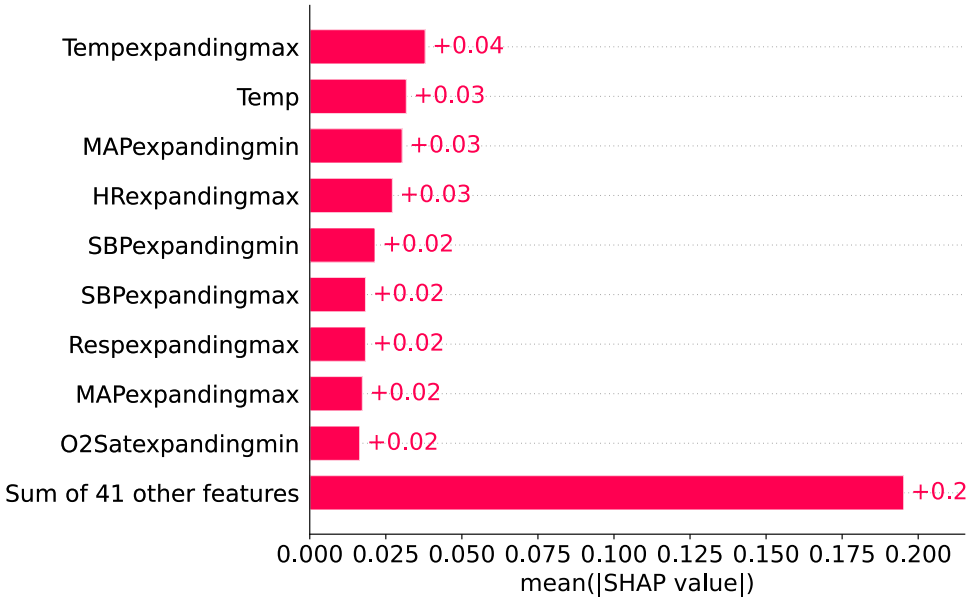


Figure 5.2: Bar plot for random forest model with 50 features

5.2 Bar Plot

The bar plot creates a global feature importance plot, where the global importance of each feature is taken to be the mean absolute SHAP value for that feature over all the given samples. From the bar plot in Fig. 5.2 of our random forest model, we can see that maximum temperature of expanding window feature has the highest mean SHAP value. The expanding window based features also hold the top positions. This is similar to what we saw in the beeswarm plot.

5.3 Waterfall Plot

Waterfall plots help explain how individual predictions are made by a model. They begin with a baseline value, then show how each feature pushes the prediction up or down. Features that increase the prediction are shown as bars extending upward, while those that decrease it are shown pulling it downward. In SHAP waterfall plots, both the direction and size of these bars reveal how strongly each feature influences the outcome. To demonstrate this, we selected the same four examples across our random forest model—covering one true positive, one true negative, one false positive, and one false negative.

Figure 5.3 shows that our model is highly confident in its prediction for the two true positive and negative instances. The predicted probabilities are 0.95 and 0.165, close to 1 and 0, respectively, which are ideal cases.

On the contrary, the model outputs probabilities close to the midpoint which is 0.5 for the false cases as is also depicted in Figure 5.3. This shows that although the model gives wrong predictions, the predictions are not too far from the true value, which is a good indicator of the model's robust performance.

5.4 LIME Plot

The idea behind LIME is to create a local, simpler version of the original model and train it around the point of interest whose prediction we wish to explain. The challenge regarding LIME is the proper selection of kernel width and the number of features.

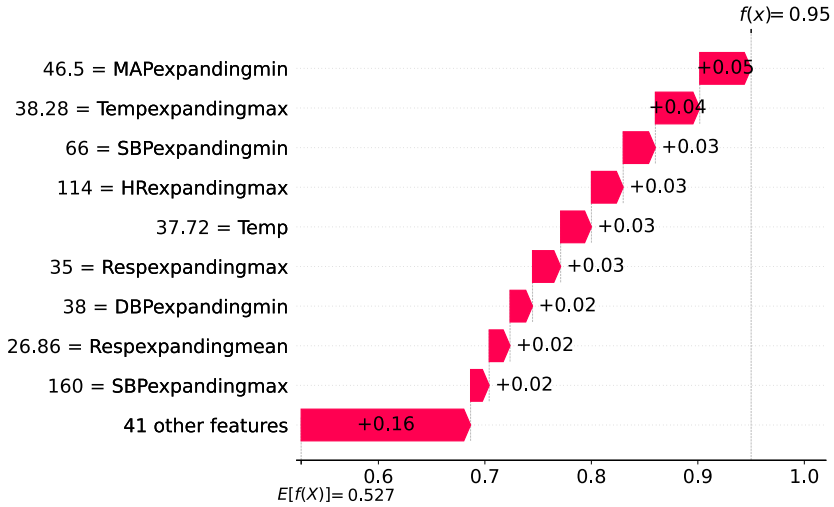
However, for our models, we went with a kernel width of 10. Moreover, we selected 15 out of 133 features to keep the local surrogate model simple. We selected the same previous 4 instances to be explained locally.

Table 5.1: Comparison of actual and surrogate model prediction

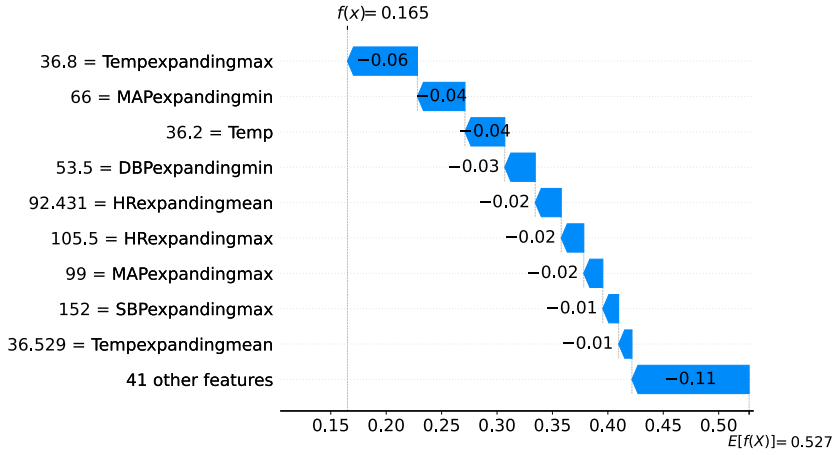
Instance type	Local prediction	Actual prediction
True Positive	0.602	0.950
True Negative	0.452	0.165
False Positive	0.479	0.540
False Negative	0.479	0.465

Table 5.1 illustrates the comparison between the actual prediction and local prediction for the selected instances and Fig. 5.4 depicts the outputs obtained.

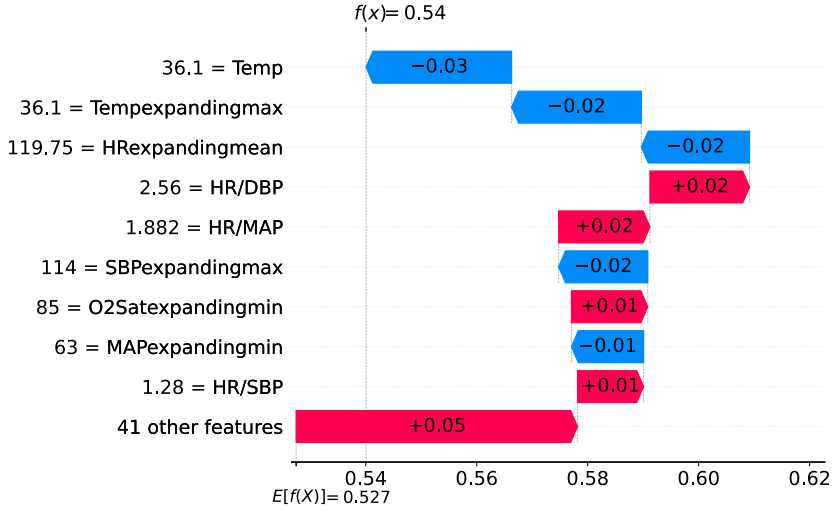
From the results obtained by the local ridge regression model trained by LIME package, we can see that the temperature and blood pressures along with their derived features are considered among the most important features by the local surrogate model.



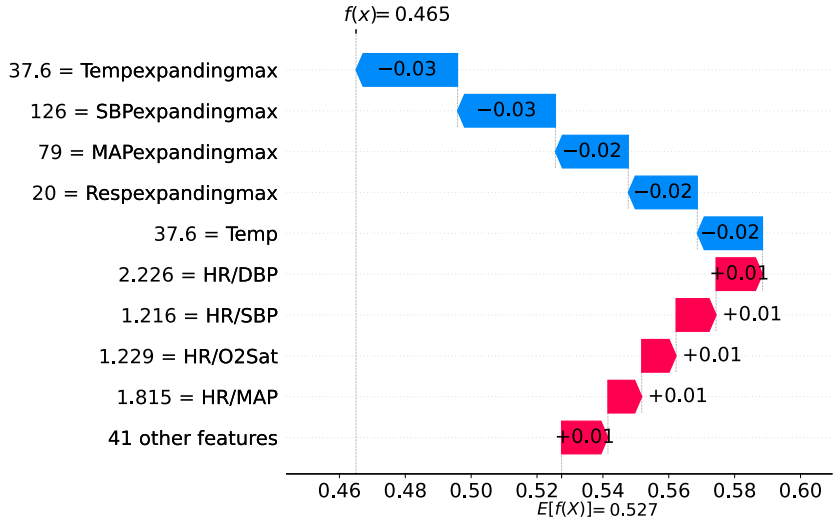
(a) True Positive Instance



(b) True Negative Instance

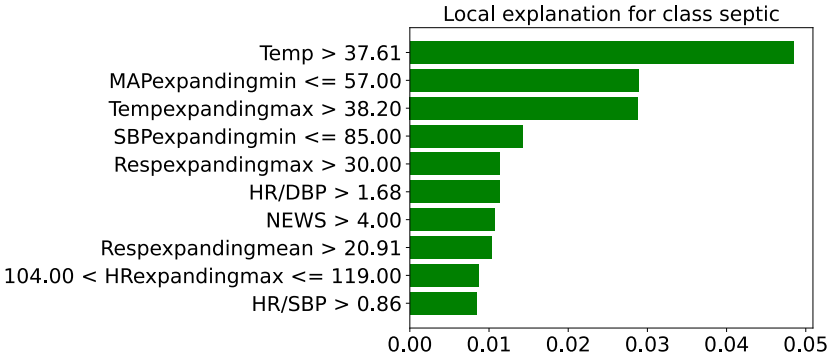


(c) False Positive Instance

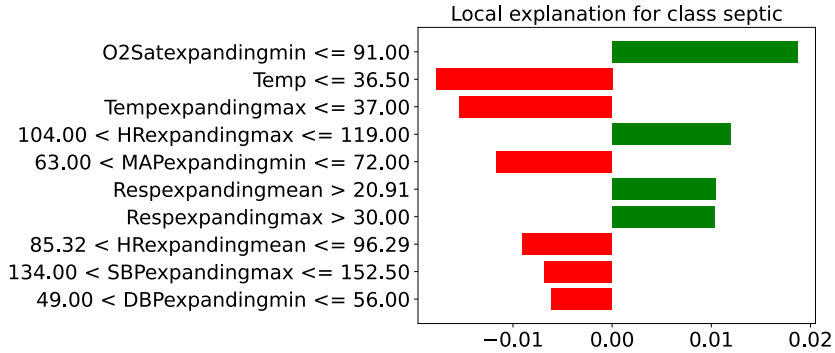


(d) False Negative Instance

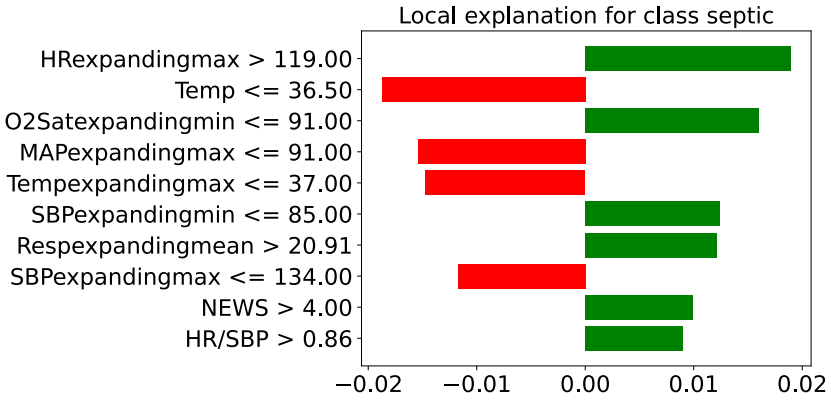
Figure 5.3: Waterfall plot comparison for four instances



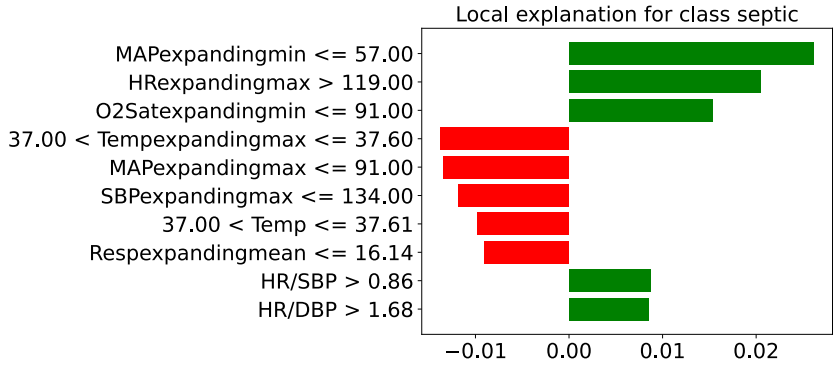
(a) LIME explanations for True Positive instance



(b) LIME explanations for True Negative Instance



(c) LIME explanations for False Positive Instance



(d) LIME explanations for False Negative Instance

Figure 5.4: LIME plot comparison for four instances

Together with the plots acquired from the SHAP python package and the LIME python package, we can conclude that body temperature, mean arterial pressure, heart rate and their derived features play a key role in influencing the outcome of our models.

5.5 Model-Based Feature Importance

Gain-based feature importance measures the average improvement in model performance brought by a feature when it is used in a decision split. Specifically, it quantifies how much each feature contributes to reducing the loss function across all trees in the ensemble. In the context of XGBoost, gain is a direct reflection of how useful a feature is in making accurate predictions. Features with higher gain are considered more influential, as they lead to greater reductions in impurity or error when selected for splitting.

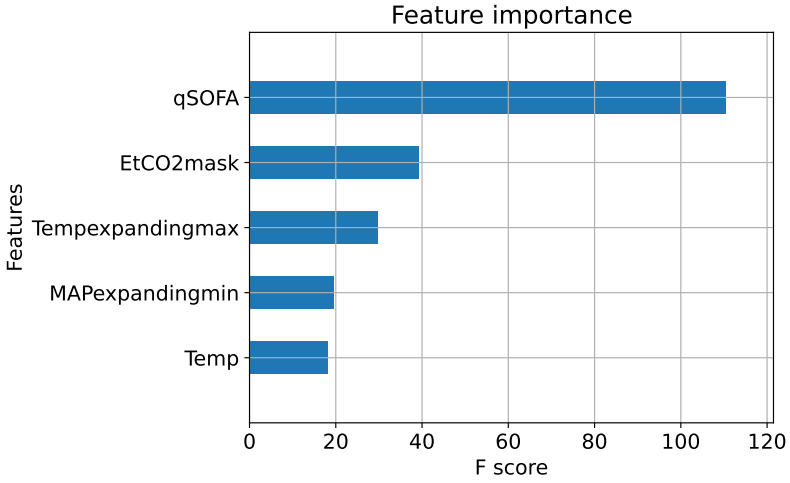


Figure 5.5: Gain-based feature importance

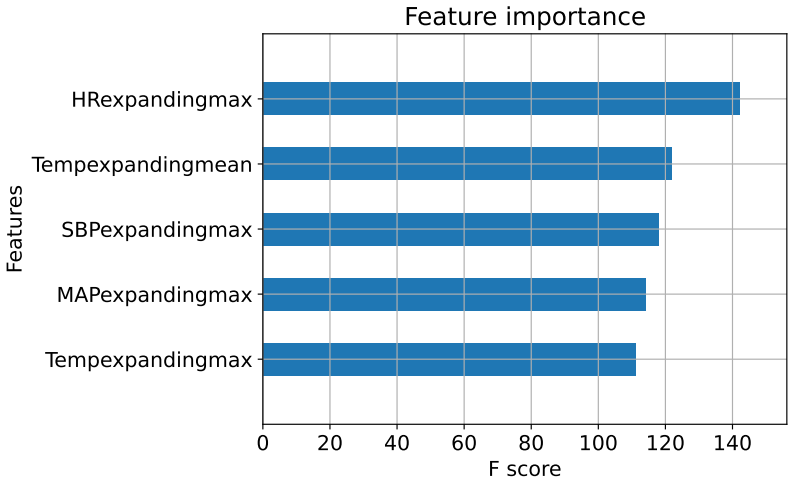


Figure 5.6: Weight-based feature importance

Weight-based feature importance, also known as frequency-based importance, counts the number of times a feature is used to split the data across all trees in the XGBoost model. It provides a simple indication of how often each feature is selected, regardless of the quality or magnitude of its contribution to model performance. While easy to interpret, weight-based importance can sometimes overemphasize features used in shallow splits or those appearing frequently but contributing little to predictive accuracy. Figure 5.5 and Figure 5.6 show the feature importances obtained from the XGBoost. These plots suggest the importance of qSOFA and also show that expanding window based features play key role in model's prediction, suggesting that long-time history of a patient is a good indicator of sepsis diagnosis.

Chapter 6

Edge Device Prototype

This chapter presents the development of a prototype system for real-time sepsis monitoring using edge devices. We describe the integration of Raspberry Pi and Arduino platforms to collect, process, and transmit vital signs efficiently. Furthermore, we demonstrate how the system supports telemedicine-based sepsis monitoring, enabling remote diagnosis and continuous patient observation in resource-constrained settings.

6.1 Edge Device Integration with Raspberry Pi and Arduino

Up to this point, our proposed machine learning workflow used the vital signs available in the Physionet-2019 database. Based on our findings, we devised a proof-of-concept setup using a subset of the vital signs for seamless sepsis diagnosis without the need for costly ICU setup. We used a Raspberry Pi 3 model and an IR temperature sensor for developing our prototype as. We developed a simple gradient boost model leveraging the body temperature,and its extracted features. Then we trained the model on the computer and saved it. After that, we implemented a simple interface with the MLX90614 infrared sensor to detect body temperature. This setup senses the data at a regular interval, sends it to the Raspberry Pi. The pre-saved model on the Pi then makes prediction based on the received sensor data in real-time and outputs the result to the screen. Figure 6.1 shows the block diagram of our setup.

Figure 6.2 illustrates an experimental setup consisting of a Raspberry Pi-3 interfaced with the temperature sensor, which is reading data and sending to the Pi. The Pi is making real-time prediction of sepsis onset probability and displaying on the monitor. Additionally, the sensor data is being sent over WiFi to the Django server running on the laptop where a dashboard with live data is being shown.

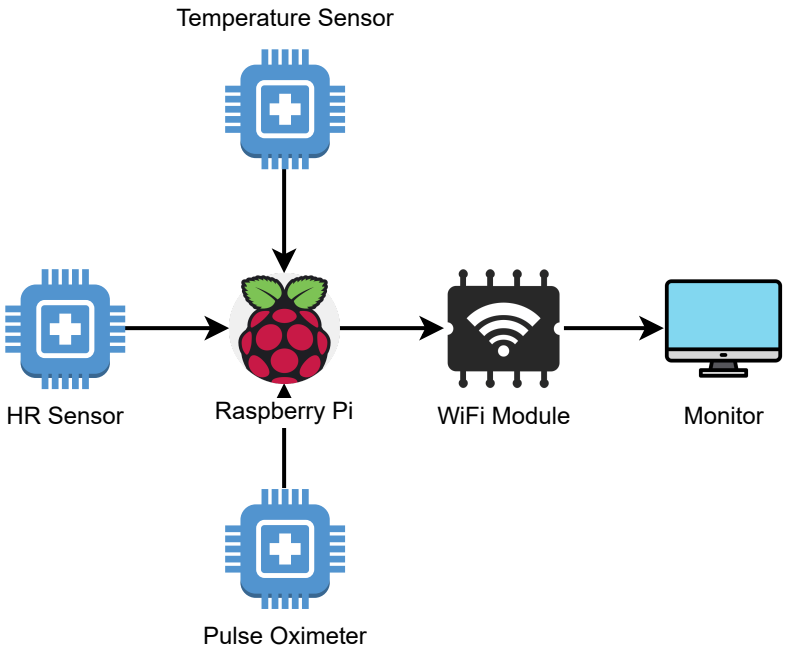


Figure 6.1: Proof-of-concept implementation of the edge-integrated sepsis detection system

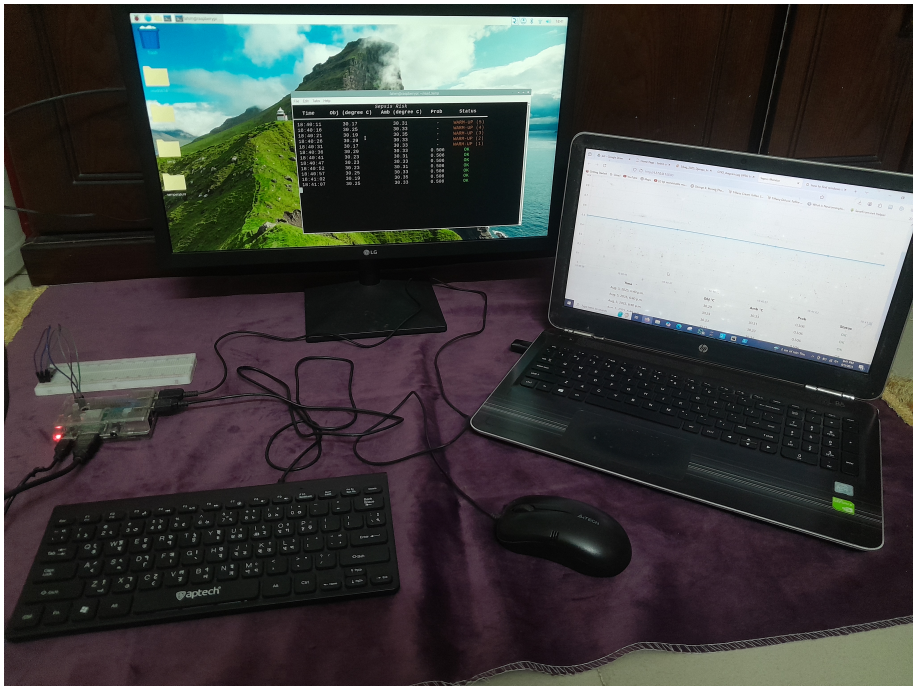


Figure 6.2: Experimental setup of real-time prediction using a raspberry pi and sensor, with telemedicine integration via a live monitoring interface on the laptop

Figure 6.3 depicts the live sensor reading from the interfaced IR temperature sensor with Pi. Along with the patient (object) temperature, ambient temperature is also shown.

The key specifications along with price are mentioned in Table 6.1. This shows that

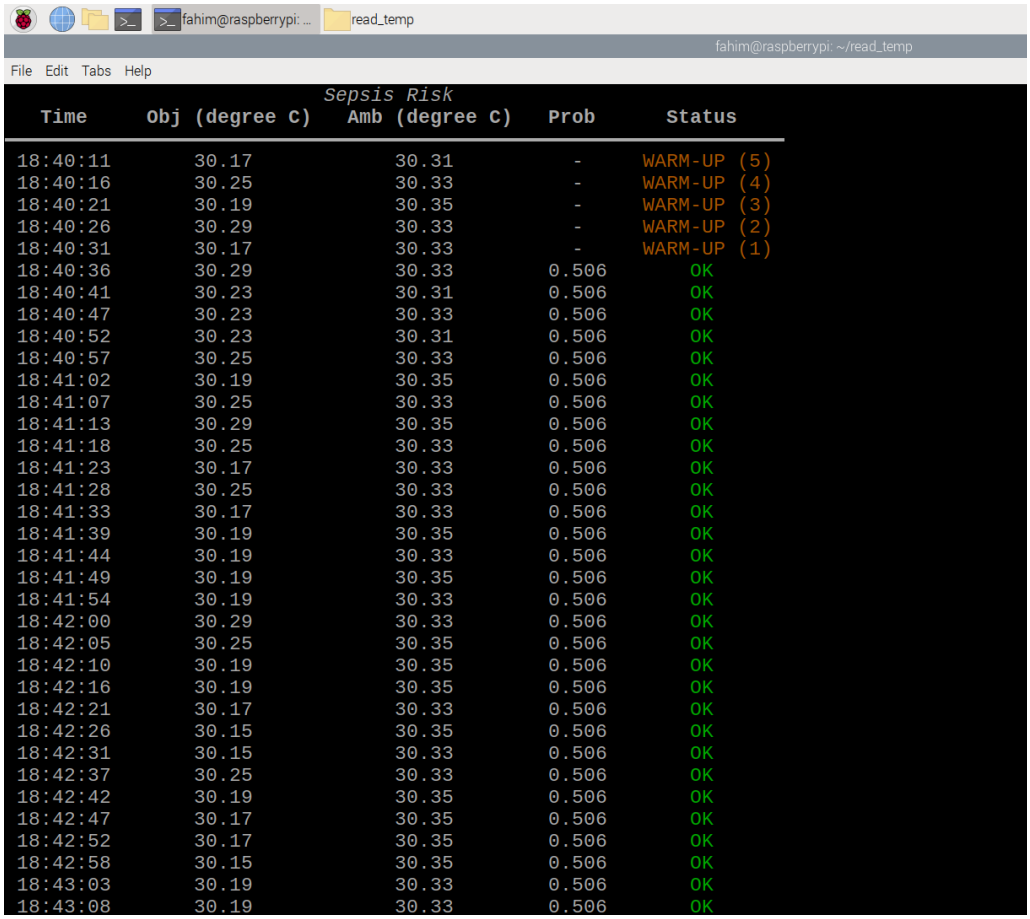


Figure 6.3: Real-time sensor reading and prediction with raspberry pi

our proposed sepsis detection setup can be implemented for under $150.

Table 6.1: Specifications of components

Component Name	**Raspberry Pi**	**MLX90614**
Type	Microprocessor	Sensor
Key Specifications	CPU: 1.2 GHz RAM: 1 GB Quad-core Wi-Fi: Built-in	Range: -70 to +350°C
Power Requirements	5V, 2.5A	3-5V, 2mA
Communication Protocol	I2C, SPI, UART	I2C
Price (USD)	$35.00	$23.00

6.2 Telemedicine-Based Sepsis Monitoring

Access to expert healthcare in remote areas is difficult, which makes it harder to diagnose sepsis timely. Thus, we have proposed a telemedicine approach towards sepsis diagnosis by setting up a web framework using Django to send patient data in real-time to advanced medical centres and hospitals. The patient data will be available to the doctors via a dashboard where they can monitor and identify anomaly to predict sepsis.

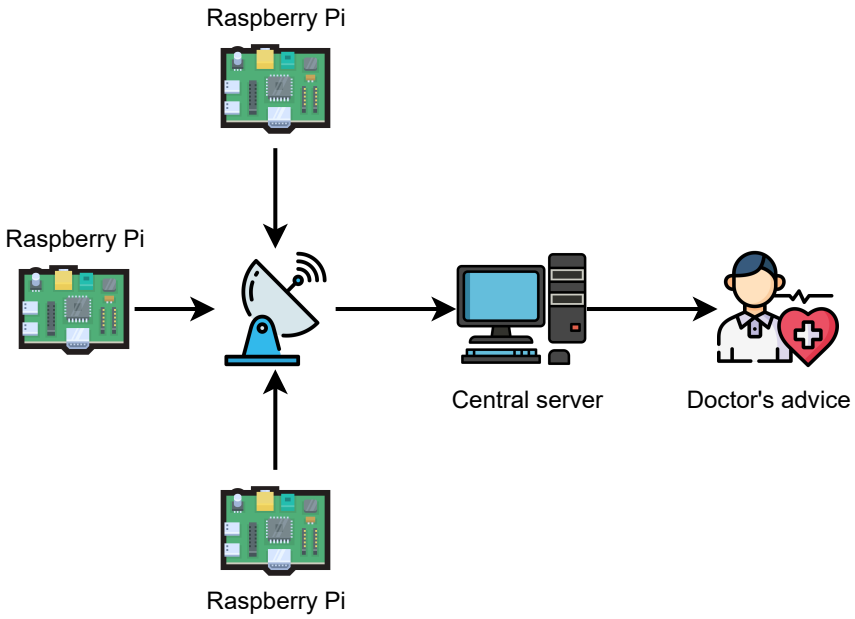
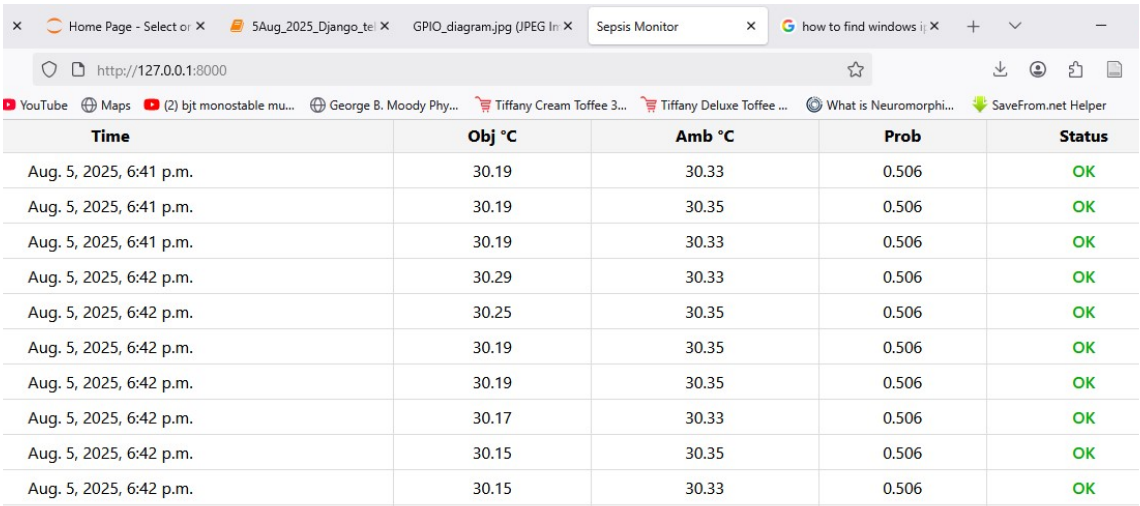


Figure 6.4: Telemedicine based sepsis monitoring



Time	Obj °C	Amb °C	Prob	Status
Aug. 5, 2025, 6:41 p.m.	30.19	30.33	0.506	OK
Aug. 5, 2025, 6:41 p.m.	30.19	30.35	0.506	OK
Aug. 5, 2025, 6:41 p.m.	30.19	30.33	0.506	OK
Aug. 5, 2025, 6:42 p.m.	30.29	30.33	0.506	OK
Aug. 5, 2025, 6:42 p.m.	30.25	30.35	0.506	OK
Aug. 5, 2025, 6:42 p.m.	30.19	30.35	0.506	OK
Aug. 5, 2025, 6:42 p.m.	30.19	30.35	0.506	OK
Aug. 5, 2025, 6:42 p.m.	30.17	30.33	0.506	OK
Aug. 5, 2025, 6:42 p.m.	30.15	30.35	0.506	OK
Aug. 5, 2025, 6:42 p.m.	30.15	30.33	0.506	OK

Figure 6.5: Live dashboard for sepsis monitoring

Figure 6.4 represents our idea. Here, the remote workstation in distant areas run

models that receive patient data from sensors and make predicitions in real time. It sends the timestamped data along with its predicted sepsis onset probability to the central workstation at modern city. There, an expert doctor will view the data on his dashboard and suggest advices on medications and further measures.

Figure 6.5 presents the real-time web dashboard that our Raspberry Pi transmits to the Django server. Each row logs a timestamped sample containing the object (skin-contact) temperature, ambient temperature, the model-predicted sepsis probability, and a status flag. In the sample session shown, object and ambient temperatures remain stable, the Random-Forest model outputs a probability of 0.506, and the status column stays in the "OK" range (green). This interface allows clinicians to monitor live trends, scroll through recent history, and quickly identify when probabilities cross the alert threshold.

6.3 Performance on Edge Device

We evaluated our model with confusion matrix, accuracy, AUROC and utility score. Although it's a simple model, the outcome is satisfactory. Fig. 6.6 illustrates the confusion matrix obtained from our temperature-based model.

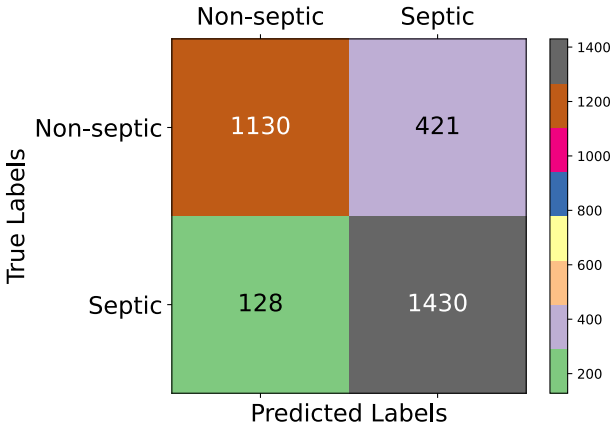


Figure 6.6: Confusion matrix for deployed model

Table 6.2: Performance metrics of temperature-based gradient boosting model

Accuracy	82.34%
Precision	77.26%
Sensitivity/Recall	91.78%
Specificity	72.86%
False positive rate	27.14%
F1 score	83.90%
AUROC	0.92

From Table 6.2, we can see that our simplistic model performs satisfactorily well

in terms of difference criteria, specifically the recall and AUROC despite using fewer features.The model attained an accuracy of 82.34%, which reflects a reliable level of predictive performance. Moreover, its precision of 77.26% and sensitivity of 91.78% reflect the fact that it gave few false positives and fea false negatives. Lower false positives avoid unnecessary utilization of hospital resources which is highly relevant in our work which primarily focuses on sepsis diagnosis in resource-constrained settings. Additionally, lower false negatives mean less chance of disease going untreated, thereby, saving more lives.

Chapter 7

Conclusion and Future Work

This chapter briefly outlines our contributions, limitations and future direction for the task at hand. In this work, we tried to address the problem of diagnosing sepsis early using machine learning approach with a focus on implementation in resource-constrained setup, remote healthcare advice and interpretability. We performed extensive feature engineering on a few non-invasive biomarkers along with different class imbalance handling and missing value imputation methods.

7.1 General Summary

This thesis addressed the critical challenge of early sepsis detection by developing a machine learning workflow with a strong emphasis on interpretability, remote healthcare applications, and implementation in resource-constrained environments. The developed system utilizes a minimal set of non-invasive vital signs, incorporating robust feature engineering and techniques to handle class imbalance and missing data.

The final model, a Random Forest classifier, demonstrated strong performance, achieving a utility score of 0.466 and an AUROC of 86.49% on the PhysioNet 2019 Challenge dataset. Crucially, the model's predictions were made transparent using interpretability tools like SHAP and LIME to foster clinical trust.

To demonstrate real-world applicability, the research includes the development of a proof-of-concept prototype: a standalone diagnostic tool deployed on a Raspberry Pi for edge computing, supported by a backend telemedicine architecture using Django for remote patient monitoring. Furthermore, the proposed workflow was validated against the MIMIC-IV demo dataset, where it outperformed the partial qSOFA score in sensitivity, a vital metric for minimizing missed diagnoses. This chapter summarizes these key contributions, acknowledges the study's limitations, and proposes promising directions

for future research.

7.2 Key Contributions

This thesis makes the following contributions to early sepsis detection from ICU data:

- Developed a machine learning workflow that attempts to diagnose sepsis based only on a number of non-invasive vital sign biomarkers. The random forest model was the best performing model that achieved a utility score (set by Physionet, the database source of our work) of 0.466 and an AUROC score of 86.49%.
- Explained the predictions made by the model using SHAP and LIME. Additionally, model-based feature importances were also explored to gain further insights. This will help bridge the communication gap between a blackbox ML model and doctors by explaining why a model made a specific prediction and what features the model thinks are important for sepsis diagnosis.
- Deployed a simpler version of the developed model on Rasperry Pi as a proof-of-concept. The prototype was able to collect sensor data and made real-time prediction. This facilitates sepsis detection in resource-constrained settings.
- Developed a basic telemedicine-based sepsis monitoring architecture using Django framework. The proof-of-concept prototype, in addition to making real-time prediction, also sent the sensor data over internet to a remote server where the doctors can view the time-logged patient data on a dashboard. The interface also shows the predicted probability of sepsis. This setup thus enables the patients in distant areas to avail expert healthcare without being physically present.
- Evaluated the performance of the proposed ML workflow on MIMIC-IV clinical database demo and compared with simple bedside sepsis diagnosis tool, partial qSOFA. It was seen that our proposed methodology outperformed the partial qSOFA in sensitivity, which is crucial for reducing false negative cases.

7.3 Limitations

The limitations of this work are outlined below:

- The study is primarily based on dataset obtained from the Physionet-2019 challenge since other similar sources for sepsis diagnosis are limited and credentialed access.

Therefore, this might affect the generalizability of our proposed workflow to some extent.

- The significant percentage of missing data even in the vital signs can lead to inaccurate prediction by the model because none of the existing imputation strategies can truly discover the missing data. Moreover, since the vital signs show somewhat similar patterns in many diseases, diagnosing sepsis based solely on vital signs is a challenge.
- In the developed prototype, the sensor quality compromised the data quality and hence, the prediction was mediocre.
- The potential of the CNN and RNN-based architecture to their full extent could not be explored because of the limited amount of data and computational capabilities. Both of these deep learning models require substantial processing power and RAM, which were unavailable to the authors.

7.4 Future Work

While this thesis lays the groundwork for interpretable predictions using simple models using minimalist hardware setup, several directions remain open for further investigation as mentioned below:

- By incorporating the full MIMIC-III and MIMIC-IV datasets, we can investigate deeper into the early diagnosis of sepsis using non-invasive vital sign biomarkers.
- By using additional high-quality sensors, the real-time performance of the proposed prototype can be significantly improved.
- By using a development board like ESP32 capable of interfacing with sensor and sending real-time data over WiFi, multiple patients can be connected to our proposed proof-of-concept setup. This will enable more patients to be monitored using a single Pi, thus, improving healthcare quality.
- Additionally, higher computational resources will enable us to explore the capabilities of CNN and RNN in this context, which will open the door of visual interpretability analysis by SHAP, giving the doctors a new perspective to the existing problem.
- Furthermore, utilizing a more capable edge device platform, such as the Jetson Nano with its onboard GPU, enables the possibility of conducting online training, thereby facilitating real-time performance enhancement.

Bibliography

[1] C. Rhee, R. Dantes, L. Epstein, D. J. Murphy, C. W. Seymour, T. J. Iwashyna, S. S. Kadri, D. C. Angus, R. L. Danner, A. E. Fiore, J. A. Jernigan, G. S. Martin, E. Septimus, D. K. Warren, A. Karcz, C. Chan, J. T. Menchaca, R. Wang, S. Gruber, M. Klompas, and the CDC Prevention Epicenter Program, "Incidence and trends of sepsis in US hospitals using clinical vs claims data, 2009–2014," *JAMA*, vol. 318, pp. 1241–1249, Oct. 2017. doi: 10.1001/jama.2017.13836.

[2] F. B. Mayr, V. B. Talisa, V. Balakumar, C.-C. H. Chang, M. Fine, and S. Yende, "Proportion and cost of unplanned 30-day readmissions after sepsis compared with other medical conditions," *JAMA*, vol. 317, pp. 530–531, Feb. 2017. doi: 10.1001/jama.2016.20468.

[3] M. van den Berg, F. E. van Beuningen, J. C. ter Maaten, and H. R. Bouma, "Hospital-related costs of sepsis around the world: A systematic review exploring the economic burden of sepsis," *Journal of Critical Care*, vol. 71, p. 154096, 2022.

[4] P. Berto, C. Ronco, D. Cruz, R. M. Melotti, and M. Antonelli, "Cost-effectiveness analysis of Polymyxin-B immobilized fiber column and conventional medical therapy in the management of abdominal septic shock in Italy," *Contrib Nephrol*, vol. 32, pp. 331–340, Nov. 2011. doi: 10.1159/000333826.

[5] A. K. R. Purba, N. Mariana, G. Aliska, S. H. Wijaya, R. R. Wulandari, U. Hadi, Hamzah, C. W. Nugroho, J. van der Schans, and M. J. Postma, "The burden and costs of sepsis and reimbursement of its treatment in a developing country: An observational study on focal infections in Indonesia," *International Journal of Infectious Diseases*, vol. 96, pp. 211–218, Jul. 2020. doi: 10.1016/j.ijid.2020.04.075.

[6] S. Micek, M. T. Johnson, R. Reichley, and M. H. Kollef, "An institutional perspective on the impact of recent antibiotic exposure on length of stay and hospital costs for patients with gram-negative sepsis," *BMC Infectious Diseases*, vol. 12, no. 1, p. 56, Mar. 2012. doi: 10.1186/1471-2334-12-56.

[7] D. T. Noritomi, O. T. Ranzani, M. B. Monteiro, E. M. Ferreira, S. R. Santos, F. Leibel, and F. R. Machado, "Implementation of a multifaceted sepsis education program in

an emerging country setting: Clinical outcomes and cost-effectiveness in a long-term follow-up study," *Intensive Care Medicine*, vol. 40, pp. 182–191, Feb. 2014. doi: 10.1007/s00134-013-3131-5.

[8] E. J. Giamarellos-Bourboulis, V. Mylona, A. Antonopoulou, I. Tsangaris, I. Koutelidakis, A. Marioli, M. Raftogiannis, P. Kopterides, K. Lymberopoulou, M. Mouktaroudi, C. Papageorgiou, B. Papaziogas, A.-P. Georgopoulou, T. Tsaganos, E. Papadomichelakis, C. Gogos, M. Ladas, A. Savva, A. Pelekanou, F. Baziaka, P. Koutoukas, T. Kanni, A. Spyridaki, N. Maniatis, N. Pelekanos, A. Kotsaki, I. Vaki, E. E. Douzinas, G. Koratzanis, and A. Armaganidis, "Effect of clarithromycin in patients with suspected Gram-negative sepsis: Results of a randomized controlled trial," *The Journal of Antimicrobial Chemotherapy*, vol. 69, pp. 1111–1118, 2014.

[9] The George Institute for Global Health, "Cost of sepsis in Australia," Tech. Rep., 2021. [Online]. Available: `https://www.georgeinstitute.org/sites/default/files/cost-of-sepsis-in-australian-report.pdf` [Accessed: Apr. 25, 2025].

[10] C. J. Paoli, M. A. Reynolds, M. Sinha, M. Gitlin, and E. Crouser, "Epidemiology and costs of sepsis in the United States—An analysis based on timing of diagnosis and severity level," *Critical Care Medicine*, vol. 46, pp. 1889–1897, Dec. 2018. doi: 10.1097/CCM.0000000000003342.

[11] E. C. N. Luijks, E. C. van der Slikke, A. R. H. van Zanten, J. C. Ter Maaten, M. J. Postma, H. B. M. Hilderink, R. H. Henning, and H. R. Bouma, "Societal costs of sepsis in the Netherlands," *Critical Care (London, England)*, vol. 28, p. 29, Jan. 2024. doi: 10.1186/s13054-024-04816-3.

[12] M. Singer, C. S. Deutschman, C. W. Seymour, M. Shankar-Hari, D. Annane, M. Bauer, R. Bellomo, G. R. Bernard, J.-D. Chiche, C. M. Coopersmith, R. S. Hotchkiss, M. M. Levy, J. C. Marshall, G. S. Martin, S. M. Opal, G. D. Rubenfeld, T. van der Poll, J.-L. Vincent, and D. C. Angus, "The Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3)," *JAMA*, vol. 315, pp. 801–810, Feb. 2016. doi: 10.1001/jama.2016.0287.

[13] M. A. Reyna, C. S. Josef, R. Jeter, S. P. Shashikumar, M. B. Westover, S. Nemati, G. D. Clifford, and A. Sharma, "Early prediction of sepsis from clinical data: The PhysioNet/Computing in Cardiology Challenge 2019," *Critical Care Medicine*, vol. 48, no. 2, p. 210, 2020. doi: 10.1097/CCM.0000000000004145.

[14] K. R. Islam, J. Prithula, J. Kumar, T. L. Tan, M. B. I. Reaz, M. S. I. Sumon, and M. E. H. Chowdhury, "Machine learning-based early prediction of sepsis using electronic

health records: A systematic review," *Journal of Clinical Medicine*, vol. 12, p. 5658, Aug. 2023. doi: 10.3390/jcm12175658.

[15] Z. He, X. Chen, Z. Fang, W. Yi, C. Wang, L. Jiang, Z. Tong, Z. Bai, Y. Li, and Y. Pan, "Early sepsis prediction using ensemble learning with features extracted from LSTM recurrent neural network," in *Proc. Computing in Cardiology (CinC)*, 2019, pp. 1–4. doi: 10.22489/CinC.2019.269.

[16] S. Liu, M. L. Ong, K. M. Kar, J. Yao, and M. Motani, "Early prediction of sepsis via SMOTE upsampling and mutual information based downsampling," in *Proc. Computing in Cardiology (CinC)*, 2019, pp. 1–4. doi: 10.22489/CinC.2019.239.

[17] J. H. Morrill, A. Kormilitzin, A. J. Nevado-Holgado, S. Swaminathan, S. D. Howison, and T. J. Lyons, "Utilization of the signature method to identify the early onset of sepsis from multivariate physiological time series in critical care monitoring," *Critical Care Medicine*, no. 10, pp. e976–e981, 2020.

[18] S. Patidar, "Diagnosis of sepsis using ratio based features," in *Proc. Computing in Cardiology (CinC)*, 2019, pp. 1–4. doi: 10.22489/CinC.2019.207.

[19] J. Singh, K. Oshiro, R. Krishnan, M. Sato, T. Ohkuma, and N. Kato, "Utilizing informative missingness for early prediction of sepsis," in *Proc. Computing in Cardiology (CinC)*, 2019. doi: 10.22489/CinC.2019.280.

[20] M. Vollmer, C. F. Luz, P. Sodmann, B. Sinha, and S.-O. Kuhn, "Time-specific metalearners for the early prediction of sepsis," in *Proc. Computing in Cardiology (CinC)*, 2019, pp. 1–4. doi: 10.22489/CinC.2019.029.

[21] M. Yang, X. Wang, H. Gao, Y. Li, X. Liu, J. Li, and C. Liu, "Early prediction of sepsis using multi-feature fusion based XGBoost learning and Bayesian optimization," in *Proc. Computing in Cardiology (CinC)*, Dec. 2019. doi: 10.22489/CinC.2019.020.

[22] Q. Yu, X. Huang, W. Li, C. Wang, Y. Chen, and Y. Ge, "Using features extracted from vital time series for early prediction of sepsis," in *Proc. Computing in Cardiology (CinC)*, 2019, pp. 1–4. doi: 10.22489/CinC.2019.067.

[23] P.-Y. Hsu and C. Holtz, "A comparison of machine learning tools for early prediction of sepsis from ICU data," in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.206.

[24] M. A. F. Pimentel, A. Mahdi, O. Redfern, M. D. Santos, and L. Tarassenko, "Uncertainty-aware model for reliable prediction of sepsis in the ICU," in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.209.

[25] M. Zabihi, S. Kiranyaz, and M. Gabbouj, “Sepsis prediction in intensive care unit using ensemble of XGBoost models,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.238.

[26] B. Sweely, A. Park, L. Winter, L. Liu, and X. Zhao, “Time-padded random forest ensemble to capture changes in physiology leading to sepsis development,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.281.

[27] I. Murugesan, K. Murugesan, L. Balasubramanian, and M. Arumugam, “Interpretation of artificial intelligence algorithms in the prediction of sepsis,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.332.

[28] P. Doggart and M. Rutherford, “Randomly under sampled boosted tree for predicting sepsis from intensive care unit databases,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.362.

[29] S. Chami and K. Tavakolian, “Early prediction of sepsis from clinical data using single Light-GBM model,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.367.

[30] J. A. Du, N. Sadr, and P. de Chazal, “Automated prediction of sepsis onset using gradient boosted decision trees,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.423.

[31] I. Hammoud, I. V. Ramakrishnan, and M. Henry, “Early prediction of sepsis using gradient boosting decision trees with optimal sample weighting,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.459.

[32] P. Biglarbeigi, D. McLaughlin, K. Rjoob, A. Abdullah, N. McCallan, A. Jasinska-Piadlo, R. Bond, D. Finlay, K. Y. Ng, A. Kennedy, and J. McLaughlin, “Early prediction of sepsis considering early warning scoring systems,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.051.

[33] S. Noorzadeh, S. Faghihroohi, and M. Zarei, “A comparative analysis of HMM and CRF for early prediction of sepsis,” in *Proc. Computing in Cardiology (CinC)*, 2019, pp. 1–4. doi: 10.22489/CinC.2019.258.

[34] M. Nakhashi, A. Toffy, P. V. Achuth, L. Palanichamy, and C. M. Vikas, “Early prediction of sepsis: Using state-of-the-art machine learning techniques on vital sign inputs,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.259.

[35] I. Krissaane, K. Hampton, J. Alshenaifi, and R. Wilkinson, “Anomaly detection semi-supervised framework for sepsis treatment,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.174.

[36] S. Lyra, S. Leonhardt, and C. H. Antink, “Early prediction of sepsis using random forest classification for imbalanced clinical data,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.276.

[37] J. Gao, Y. Lu, N. Ashrafi, I. Domingo, K. Alaei, and M. Pishgar, “Prediction of sepsis mortality in ICU patients using machine learning methods,” *BMC Medical Informatics and Decision Making*, vol. 24, no. 1, Aug. 2024. doi: 10.1186/s12911-024-02630-z.

[38] R. Firoozabadi and S. Babaeizadeh, “An ensemble of bagged decision trees for early prediction of sepsis,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.023.

[39] A. Deogire, “A low dimensional algorithm for detection of sepsis from electronic medical record data,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.037.

[40] R. Pawar, J. Bone, J. M. Ansermino, and M. Görges, “An algorithm for early detection of sepsis using traditional statistical regression modeling,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.061.

[41] V. Abromavičius and A. Serackis, “Sepsis prediction model based on vital signs related features,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.095.

[42] K. M. and V. S. Akshaya, “A XGBOOST Based Algorithm for Early Prediction of Human Sepsis,” in *Proc. 4th Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Tirunelveli, India, 2022, pp. 1643–1647, doi: 10.1109/ICSSIT53264.2022.9716564.

[43] M. Chen, J. Huo, and Y. Duan, “An interpretable model for sepsis prediction using multi-objective rule extraction,” *Journal of Intelligent Information Systems*, vol. 62, no. 6, pp. 1403–1429, Apr. 2024, doi: 10.1007/s10844-024-00859-8.

[44] S. N. *et al.*, “Early Prediction of Sepsis using ML Algorithms on Clinical Data,” in *Proc. 14th Int. Conf. Comput. Commun. Netw. Technol. (ICCCNT)*, Delhi, India, 2023, pp. 1–8, doi: 10.1109/ICCCNT56998.2023.10307995.

[45] K. Ewig *et al.*, “Multi-Subset Approach to Early Sepsis Prediction,” in *Proc. Congr. Comput. Sci., Comput. Eng. Appl. Comput. (CSCE)*, Las Vegas, NV, USA, 2023, pp. 1335–1341, doi: 10.1109/CSCE60160.2023.00224.

[46] L. Liu, H. Wu, Z. Wang, Z. Liu, and M. Zhang, “Early prediction of sepsis from clinical data via heterogeneous event aggregation,” in *Proc. Computing in Cardiology (CinC)*, 2019, pp. 1–4. doi: 10.22489/CinC.2019.157.

[47] E. Macias, G. Boquet, J. Serrano, J. L. Vicario, J. Ibeas, and A. Morel, “Novel imputing method and deep learning techniques for early prediction of sepsis in intensive care units,” in *Proc. Computing in Cardiology (CinC)*, 2019, pp. 1–4. doi: 10.22489/CinC.2019.038.

[48] P. Nejedly, F. Plesinger, I. Viscor, J. Halamek, and P. Jurak, “Prediction of sepsis using LSTM neural network with hyperparameter optimization with a genetic algorithm,” in *Proc. Computing in Cardiology (CinC)*, 2019, pp. 1–4. doi: 10.22489/CinC.2019.022.

[49] S. Schellenberger, K. Shi, J. P. Wiedemann, F. Lurz, R. Weigel, and A. Koelpin, “An ensemble LSTM architecture for clinical sepsis detection,” in *Proc. Computing in Cardiology (CinC)*, 2019, pp. 1–4. doi: 10.22489/CinC.2019.297.

[50] T. Vicar, P. Novotna, J. Hejc, M. Ronzhina, and R. Smisek, “Sepsis detection in sparse clinical data using long short-term memory network with dice loss,” in *Proc. Computing in Cardiology (CinC)*, 2019, pp. 1–4. doi: 10.22489/CinC.2019.341.

[51] L. Tran, M. Nguyen, and C. Shahabi, “Representation learning for early sepsis prediction,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.021.

[52] X. Li, G. A. Ng, and F. S. Schlindwein, “Convolutional and recurrent neural networks for early detection of sepsis using hourly physiological data from patients in intensive care unit,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.054.

[53] B. Roussel, J. Behar, and J. Oster, “A recurrent neural network for the prediction of vital sign evolution and sepsis in ICU,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.082.

[54] L. Narayanaswamy, D. Garg, B. Narra, and R. Narayanswamy, “Machine learning algorithmic and system level considerations for early prediction of sepsis,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.161.

[55] B. Lee, K. Cho, O. Kwon, and Y. Lee, “Improving the performance of a neural network for early prediction of sepsis,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.162.

[56] J. Yao, M. L. Ong, K. K. Mun, S. Liu, and M. Motani, "Hybrid feature learning using autoencoders for early prediction of sepsis," in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.243.

[57] S. Sarafrazi, R. S. Choudhari, C. Mehta, H. K. Mehta, O. K. Japalaghi, J. Han, K. A. Mehta, H. Han, and P. A. Francis-Lyon, "Cracking the "Sepsis" code: Assessing time series nature of EHR data, and using deep learning for early sepsis prediction," in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.411.

[58] M. Alfaras, R. Varandas, and H. Gamboa, "Ring-topology echo state networks for ICU sepsis classification," in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.327.

[59] N. Nonaka and J. Seita, "Demographic information initialized stacked gated recurrent unit for an early prediction of sepsis," in *Proc. Computing in Cardiology (CinC)*, 2019, pp. 1–4. doi: 10.22489/CinC.2019.153.

[60] S. D. Wickramaratne and M. D. Shaad Mahmud, "Bi-directional gated recurrent unit based ensemble model for the early detection of sepsis," in *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2020, pp. 70–73.

[61] Y. Chang, J. Rubin, G. Boverman, S. Vij, A. Rahman, A. Natarajan, and S. Parvaneh, "A multi-task imputation and classification neural architecture for early prediction of sepsis from multivariate clinical time series," in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.110.

[62] E. H. Gilbertson, K. M. Jones, A. M. Stroh, and B. M. Whitaker, "Early detection of sepsis using feature selection, feature extraction, and neural network classification," in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.386.

[63] C. Aguet, J. Van Zaen, and M. Lemay, "Sepsis detection using missingness information," in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.392.

[64] T. M., Y. Tae, H. J. Yeo, J. H. Jang, K. Cho, D. Yoo, Y. Lee, S.-H. Ahn, Y. Kim, N. Lee, *et al.*, "Development and validation of deep-learning-based sepsis and septic shock early prediction system (DeepSEPS) using real-world ICU data," *J. Clin. Med.*, vol. 12, no. 22, p. 7156, 2023, doi: 10.3390/jcm12227156.

[65] C. Pou-Prom, Z. Yang, M. Sidhaye, and D. Dai, "Development of an early warning system for sepsis," in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.034.

[66] X. Li, Y. Kang, X. Jia, J. Wang, and G. Xie, “TASP: A time-phased model for sepsis prediction,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.049.

[67] M. Scherpf, M. Goldammer, H. Malberg, and F. Gräßer, “Sepsis onset prediction applying a stacked combination of a recurrent neural network and a gradient boosted machine,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.089.

[68] Y. Wang, B. Xiao, X. Bi, W. Li, J. Zhang, and X. Ma, “Prediction of sepsis from clinical data using long short-term memory and eXtreme gradient boosting,” in *Proc. Computing in Cardiology (CinC)*, 2019, pp. 1–4. doi: 10.22489/CinC.2019.192.

[69] K. Lin, H. Huang, J. Cheng, J. Jiang, J. Zhang, and G. Zhou, “Predicting sepsis using self-supervised representation learning with contrastive learning and prototypes,” in *Proc. 31st ACM Int. Conf. Multimedia (MM)*, Ottawa, ON, Canada, Oct. 2023, pp. 1338–1347, doi: 10.1145/3581783.3612149.

[70] V. Athanasiou and Z. Konkoli, “Memristor models for early detection of sepsis in ICU patients,” in *Proc. Computing in Cardiology (CinC)*, Sep. 2019. doi: 10.22489/CinC.2019.223.

[71] M. Giordano, K. Dheman, and M. Magno, “SepAl: Sepsis alerts on low-power wearables with digital biomarkers and on-device tiny machine learning,” *IEEE Sensors Journal*, vol. 25, pp. 7858–7866, Mar. 2025. doi: 10.1109/JSEN.2024.3424655.

[72] M. Fu, J. Yuan, M. Lu, P. Hong, and M. Zeng, “An ensemble machine learning model for the early detection of sepsis from clinical data,” in *Proc. Computing in Cardiology (CinC)*, 2019, pp. 1–4. doi: 10.22489/CinC.2019.317.

[73] M. Gholamzadeh, H. Abtahi, and R. Safdari, “Comparison of different machine learning algorithms to classify patients suspected of having sepsis infection in the intensive care unit,” *Informatics in Medicine Unlocked*, vol. 38, p. 101236, 2023. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S2352914823000783`.

[74] A. Zhou, R. Beyah, and R. Kamaleswaran, “OnAI-Comp: An online AI experts competing framework for early sepsis detection,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 19, pp. 3595–3603, 2022. doi: 10.1109/TCBB.2021.3122405.

[75] M. Yang, C. Liu, X. Wang, Y. Li, H. Gao, X. Liu, and J. Li, “An explainable artificial intelligence predictor for early detection of sepsis,” *Critical Care Medicine*, vol. 48, pp. e1091–e1096, Nov. 2020. doi: 10.1097/CCM.0000000000004550.

[76] C. Hu, L. Li, W. Huang, T. Wu, Q. Xu, J. Liu, and B. Hu, “Interpretable machine learning for early prediction of prognosis in sepsis: A discovery and validation study,” *Infectious Diseases and Therapy*, vol. 11, pp. 1117–1132, Jun. 2022. doi: 10.1007/s40121-022-00628-6.

[77] N. Nesaragi and S. Patidar, “An explainable machine learning model for early prediction of sepsis using ICU data,” in *Infections and Sepsis Development*, IntechOpen, Jul. 2021. doi: 10.5772/intechopen.98957.

[78] Z. Jiang, L. Bo, Z. Xu, Y. Song, J. Wang, P. Wen, X. Wan, T. Yang, X. Deng, and J. Bian, “An explainable machine learning algorithm for risk factor analysis of in-hospital mortality in sepsis survivors with ICU readmission,” *Computer Methods and Programs in Biomedicine*, vol. 204, p. 106040, Jun. 2021. doi: 10.1016/j.cmpb.2021.106040.

[79] J. Min, M. Nothing, B. Coble, H. Zheng, J. Park, H. Im, G. F. Weber, C. M. Castro, F. K. Swirski, R. Weissleder, and H. Lee, “Integrated biosensor for rapid and point-of-care sepsis diagnosis,” *ACS Nano*, vol. 12, pp. 3378–3384, Apr. 2018. doi: 10.1021/acsnano.7b08965.

[80] S. Buyrukoğlu and S. Savaş, “Stacked-based ensemble machine learning model for positioning footballer,” *Arabian Journal for Science and Engineering*, vol. 48, no. 2, pp. 1371–1383, 2023.

[81] J. H. Friedman, “Stochastic gradient boosting,” *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002. doi: 10.1016/S0167-9473(01)00065-2.

[82] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016. doi: 10.1145/2939672.2939785.

[83] N. Rajendiran and L. S. Kumar, “Pixel level feature extraction and machine learning classification for water body extraction,” *Arabian Journal for Science and Engineering*, vol. 48, no. 8, pp. 9905–9928, 2023.

[84] S. Sousa, S. Paredes, T. Rocha, J. Henriques, J. Sousa, and L. Gonçalves, “Machine learning models’ assessment: trust and performance,” *Medical & Biological Engineering & Computing*, vol. 62, pp. 3397–3410, 2024.

Appendix A

Python Source Code

A.1 Python Script for Model Training

```
# import necessary libraries
import os
import sys
from pathlib import Path
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import joblib
from sklearn.model_selection import (
    GroupShuffleSplit, StratifiedGroupKFold, GroupKFold, cross_validate
)
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    confusion_matrix, ConfusionMatrixDisplay,
    classification_report, roc_auc_score, roc_curve
)
from imblearn.under_sampling import RandomUnderSampler
from pathlib import Path

# ========= CONFIG =========
CSV_PATH     = "train_features.csv"   # your combined file
TARGET       = "SepsisLabel"          # target column name
```

```
GROUP        = "PatientID"            # patient/group column name
TEST_SIZE    = 0.2
N_SPLITS_CV  = 5
RANDOM_STATE = 42

# 1) Load & prep
df        = pd.read_csv(CSV_PATH)
feature_cols = [c for c in df.columns if c not in [TARGET, GROUP]]
X         = df[feature_cols]
y         = df[TARGET].astype(int)
groups    = df[GROUP] if GROUP in df.columns else None

# 2) Hold-out split (-groupaware)
gss = GroupShuffleSplit(n_splits=1, test_size=TEST_SIZE,
                        random_state=RANDOM_STATE)
train_idx, test_idx = next(gss.split(X, y, groups=groups))

X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]
groups_train    = groups.iloc[train_idx]

# 3) Downsample JUST the training set
rus       = RandomUnderSampler(sampling_strategy='majority', random_state=42)
X_train_ds, y_train_ds = rus.fit_resample(X_train, y_train)

print("Before:", y_train.value_counts())
print("After :", pd.Series(y_train_ds).value_counts())

# because X_train is a DataFrame, the returned X_train_ds preserves
# the original row index, so we can align
# groups_train to the down-sampled
# rows by that index:
groups_train_ds = groups_train.loc[X_train_ds.index]

# 4) Pipeline (impute → RF)
pipe = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="median")),
    ("rf", RandomForestClassifier(
        n_estimators=20,
```

```
        max_depth = 16,
        min_samples_leaf = 10,
        class_weight="balanced_subsample",
        n_jobs=1,
        random_state=RANDOM_STATE,
        verbose = 2
    ))
])

# 6) Fit on full TRAIN
pipe.fit(X_train_ds, y_train_ds)

# 7) Evaluate on TEST (once, untouched)
y_pred = pipe.predict(X_test)
y_prob = pipe.predict_proba(X_test)[:, 1]

print("\n===Test Set Performance===")
cm = confusion_matrix(y_test, y_pred, labels=[0,1])
print("Confusion matrix [rows=true, cols=pred] labels(0,1):\n", cm)
print("\n Classification report:\n", classification_report(y_test, y_pred,
                                                          digits=3))
auc = roc_auc_score(y_test, y_prob)
print(f"AUROC: {auc:0.3f}")

# Confusion Matrix (labeled)
joblib.dump(pipe, 'models/rf_pipeline2.joblib')

# after you define feature_cols (same list you used for X)
Path("models").mkdir(exist_ok=True, parents=True)
pd.Series(feature_cols).to_json("models/feature_cols.json", orient="values")
```

A.2 Python Script for Real-Time Prediction

```
# import necessary libraries
import time, json
from collections import deque
from datetime import datetime
from rich.table import Table
from rich import box
```

```
from rich.console import Console

import pandas as pd
import numpy as np
import joblib
import os

import board, busio
import adafruit_mlx90614

#    CONFIG
SAMPLE_PERIOD_SEC = 5
ROLLING_WINDOW_N  = 6          # must match the window used during training
MODEL_PATH        = "models/rf_pipeline2.joblib"
FEATURE_LIST_PATH = "models/feature_cols.json"
THRESHOLD         = 0.50       # adjust via PR-curve on validation data

#    MODEL & FEATURE LIST
pipe = joblib.load(MODEL_PATH)

try:
    with open(FEATURE_LIST_PATH) as f:
        FEATURE_COLS = json.load(f)
except Exception:
    # fallback if the JSON isn't present
    FEATURE_COLS = [
        "Temp", "Temp-Temp1", "Temp/Temp1",
        "Tempmean", "Tempsum", "Tempmax", "Tempmin",
        "Tempexpandingmean", "Tempexpandingmax", "Tempexpandingmin"
    ]

#    SENSOR INIT
i2c = busio.I2C(board.SCL, board.SDA)
mlx = adafruit_mlx90614.MLX90614(i2c)

rolling  = deque(maxlen=ROLLING_WINDOW_N)  # recent N temps
history  = []                               # expanding window
prev     = None                             # previous reading
```

```
console = Console()

table = Table(title='Sepsis Risk', box=box.SIMPLE_HEAVY, show_edge=False)
for col in ["Time", "Obj (degree C)", "Amb (degree C)", "Prob", "Status"]:
    table.add_column(col, justify="center")

#    FEATURE ENGINEERING
def build_features(curr, prev, roll_seq, full_seq):
    """Return 2-D feature array in the order seen by the model."""
    diff  = np.nan if prev is None else curr - prev
    ratio = np.nan if (prev is None or prev == 0) else curr / prev

    r = np.array(roll_seq, dtype=float)
    r_mean = np.nanmean(r) if r.size else np.nan
    r_sum  = np.nansum(r)  if r.size else np.nan
    r_max  = np.nanmax(r)  if r.size else np.nan
    r_min  = np.nanmin(r)  if r.size else np.nan

    f = np.array(full_seq, dtype=float)
    e_mean = np.nanmean(f) if f.size else np.nan
    e_max  = np.nanmax(f)  if f.size else np.nan
    e_min  = np.nanmin(f)  if f.size else np.nan

    feats = {
        "Temp": curr,
        "Temp-Temp1": diff,
        "Temp/Temp1": ratio,
        "Tempmean": r_mean,
        "Tempsum": r_sum,
        "Tempmax": r_max,
        "Tempmin": r_min,
        "Tempexpandingmean": e_mean,
        "Tempexpandingmax": e_max,
        "Tempexpandingmin": e_min,
    }

    # 2-D array because scikit pipelines expect (n_samples, n_features)
    #return np.array([[feats.get(k, np.nan) for k in FEATURE_COLS]],
    # dtype=float)
```

```
    return pd.DataFrame([feats], columns = FEATURE_COLS)

#    MAIN LOOP
print("Press Ctrl+C to stop.")
try:
    while True:
        obj = float(mlx.object_temperature)
        amb = float(mlx.ambient_temperature)

        rolling.append(obj)
        history.append(obj)

        ts = datetime.now().strftime("%H:%M:%S")


        if len(history) >= max(2, ROLLING_WINDOW_N):
            X     = build_features(obj, prev, rolling, history)
            proba = pipe.predict_proba(X)[0]

            # if model ended up one-class, proba has length 1
            prob = float(proba[1]) if len(proba) > 1 else 0.0
            status = "[red]SEPSIS[/red]" if prob >= THRESHOLD
                                         else "[green]OK[/green]"
            table.add_row(ts, f"{obj:.2f}", f"{amb:.2f}", f"{prob:.3f}",
                          status)

        else:
            need = max(2, ROLLING_WINDOW_N) - len(history)
            table.add_row(ts, f"{obj:.2f}", f"{amb:.2f}", "-",
                          f"[yellow]WARM-UP ({need})[/yellow]")

        os.system("clear")
        console.print(table)
        prev = obj
        time.sleep(SAMPLE_PERIOD_SEC)

except KeyboardInterrupt:
    print("\nStopped.")
```

Appendix B

Key Django Modules

B.1 Project Layout

```
django_sepsis_dashboard/
  manage.py
  monitor/
      __init__.py
      admin.py
      apps.py
      models.py
      serializers.py
      tests.py
      views.py
      templates/
          monitor/
              dashboard.html
      migrations/
  sepsisdash/
      __init__.py
      asgi.py
      settings.py
      urls.py
      wsgi.py
```

B.2 Model

Listing B.1: Database schema

```
from django.db import models

class Reading(models.Model):
    ts        = models.DateTimeField()
    obj_c     = models.FloatField()
    amb_c     = models.FloatField()
    prob      = models.FloatField()
    status    = models.CharField(max_length=10)
    created_at= models.DateTimeField(auto_now_add=True)
```

Listing B.2: Django views

```
from rest_framework.decorators import api_view
from rest_framework.response import Response
from rest_framework import status
from django.conf import settings
from django.shortcuts import render
from .models import Reading
from .serializers import ReadingSerializer

API_TOKEN = "supersecrettoken123"  # move to env var in production

@api_view(["POST"])
def readings_create(request):
    # very simple token check
    if request.headers.get("X-API-KEY") != API_TOKEN:
        return Response({"detail": "Unauthorized"},
        status=status.HTTP_401_UNAUTHORIZED)

    ser = ReadingSerializer(data=request.data)
    if ser.is_valid():
        ser.save()
        return Response({"ok": True},
        status=status.HTTP_201_CREATED)
    return Response(ser.errors,
    status=status.HTTP_400_BAD_REQUEST)
```

```
def dashboard(request):
    latest = Reading.objects.order_by('-ts')[:50]  # last 50
    return render(request, "monitor/dashboard.html",
                  {"rows": latest[::-1]})
```