

Visvesvaraya Technological University

Jnana Sangama, Belagavi - 590018



A Project Work Phase-2 (18CSP83)

Report on

“Sepsis Prediction”

*Project Report submitted in partial fulfilment of the requirement for the
award of the degree of*

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Nikhil KH	1KG20CS073
Prithvi Raj	1KG20CS085
Rahul BM	1KG20CS086
Sagar Naidu	1KG20CS095

Under the guidance of
Mrs. Nagaveni B Nimbal
Associate Professor

Department of Computer Science & Engineering
KSSEM, Bengaluru-560109



KSSEM
K. S. SCHOOL OF ENGINEERING AND MANAGEMENT

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
K. S. School of Engineering and Management
#15, Mallasandra, off. Kanakapura Road, Bengaluru – 560109
2023 - 2024



K. S. School of Engineering and Management

#15, Mallasandra, off. Kanakapura Road, Bengaluru - 560109
Department of Computer Science & Engineering

CERTIFICATE

Certified that the Project Work Phase-2 (18CSP83) entitled "SEPSIS PREDICTION" is a bonafide work carried out by: Nikhil KH (1KG20CS073), Prithvi Raj (1KG20CS085), Rahul BM (1KG20CS086), Sagar Naidu N (1KG20CS095) in partial fulfilment for VII semester B.E., Project Work in the branch of Computer Science and Engineering prescribed by **Visvesvaraya Technological University, Belagavi** during the period of February 2024 to May 2024. It is certified that all the corrections and suggestions indicated for internal assessment have been incorporated. The Project Work Phase-2 Report has been approved as it satisfies the academic requirements in report of project work prescribed for the Bachelor of Engineering degree.

Signature of the Guide
Mrs. Nagaveni B Nimbal
Associate Professor
K.S.S.E.M, Bengaluru

Signature of the HOD
Dr. K Venkata Rao
Professor, Head of the Dept.,
K.S.S.E.M, Bengaluru

Signature of the Principal
Dr. K Rama Narasimha
Principal/Director,
K.S.S.E.M, Bengaluru
Dr. K. RAMA NARASIMHA
Principal/Director

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

K S School of Engineering and Management
Bengaluru - 560 109

DECLARATION

We, the undersigned students of 8th semester, Computer Science & Engineering, KSSEM, declare that our Project Work Phase-2 entitled "SEPSIS PREDICTION", is a bonafide work of our's. Our project is neither a copy nor by means a modification of any other engineering project.

We also declare that this project was not entitled for submission to any other university in the past and shall remain the only submission made and will not be submitted by us to any other university in the future.

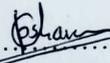
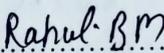
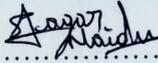
Place: Bangalore

Date :

Name and USN

Signature

Nikhil KH	-1KG20CS073
Prithvi Raj	-1KG20CS085
Rahul BM	-1KG20CS086
Sagar Naidu N	-1KG20CS095

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task will be incomplete without the mention of the individuals, we are greatly indebted to, who through guidance and providing facilities have served as a beacon of light and crowned our efforts with success.

We would like to express our gratitude to our **MANAGEMENT**, K.S. School of Engineering and Management, Bengaluru, for providing a very good infrastructure and all the kindness forwarded to us in carrying out this project work in college.

We would like to express our gratitude to **Dr. K.V.A Balaji**, CEO, K.S. School of Engineering and Management, Bengaluru, for his valuable guidance.

We would like to express our gratitude to **Dr. K. Rama Narasimha**, Principal, K.S. School of Engineering and Management, Bengaluru, for his valuable guidance.

We like to extend our gratitude to **Dr. K Venkata Rao**, Professor and Head, Department of Computer Science & Engineering, for providing a very good facilities and all the support forwarded to us in carrying out this Project Work Phase-I successfully.

We also like to thank our Project Coordinators, **Mrs. M e e n a**, Asst. Professor, **Mrs. Supriya Suresh**, Asst. Professor, Department of Computer Science & Engineering for their help and support provided to carry out the Project Work Phase-II successfully.

Also, we are thankful to **Mrs. Nagaveni B Nimbale**, Associate Professor for being our Project Guide, under whose able guidance this project work has been carried out Project Work Phase-2 successfully.

We are also thankful to the teaching and non-teaching staff of Computer Science & Engineering, KSSEM for helping us in completing the Project Work Phase-2 work.

Nikhil KH

Prithvi Raj

Rahul BM

Sagar Naidu

ABSTRACT

Sepsis is a life-threatening condition that occurs when the body's response to infection causes widespread inflammation, case fatality rates are still unacceptably high and early detection and treatment is vital since it significantly reduces mortality rates. The main goal of this project is to develop a multidimensional approach that integrates traditional clinical parameters, biomarkers, and advanced machine learning techniques. A Long Short-Term Memory (LSTM) network is going to be built to extract the intrinsic relation between different indicators in clinical data. The analysis includes the utilization of electronic health records, multicenter collaborations, and evolving challenges. The aim is to contribute to the development of accurate and timely sepsis prediction tools, ultimately.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
1.	INTRODUCTION	1-4
1.1	Overview	1
1.2	Purpose of the Project	2
1.3	Scope of the Project	2
1.4	Definitions	2
2.	LITERATURE SURVEY	6 -16
2.1	Machine learning for the prediction of sepsis: a systematic review and meta- analysis of diagnostic test accuracy.	6
2.2	Feature selection may aid deep neural networks to provide still better results for the bioinformatics problems.	6
2.3	Learning representations and patterns for the early detection of sepsis using deep neural networks.	7
2.4	Prediction of sepsis in the intensive care unit with minimal electronic health record data: a machine learning approach.	7
2.5	A targeted real-time early warning score (trewscore) for septic shock.	8
2.6	MIMIC-III, a freely accessible critical care database	8
2.7	An attention-based deep learning model of clinical events in the intensive care unit.	9
2.8	Machine learning predicts mortality in septic patients using only routinely available ABG variables: a multi-	9

	centre evaluation.	
2.9	A deep learning approach for sepsis monitoring via severity score estimation.	10
2.10	Evaluate prognostic accuracy of SOFA component score for mortality among adults with sepsis by machine learning method.	10

3.	PROBLEM IDENTIFICATION	17
3.1	Problem Statement	17
3.2	Project Scope	17
4.	GOALS AND OBJECTIVES	18
4.1	Project Goals	18
4.2	Project Objectives	18
5.	SYSTEM REQUIREMENT SPECIFICATION	19
5.1	Software Requirements	19
7.	IMPLEMENTATION	22-45
8.	RESULTS AND SNAPSHOTS	46-49

9.	APPLICATIONS	50
10.	CONTRIBUTION TO SOCIETY AND ENVIRONMENT	51
	REFERENCES	52-58

LIST OF FIGURES

Fig. No.	Figure Name	Page No.
6.1	Data flow diagram LEVEL 0	14
6.1.1	Data flow diagram LEVEL1	14
6.1.2	Data flow diagram LEVEL 2	15
6.1.3	Data flow diagram LEVEL 2.1	15
6.1.4	Data flow diagram LEVEL 2.2	16
8.1	Training the data	22
8.2	Loss curve and Accuracy curve	23
8.3	AUC Curve	23

Chapter 1

INTRODUCTION

1.1 OVERVIEW

Sepsis is a life-threatening condition which occurs when the body's immune system overreacts to an infection, causing damage to its own tissues and organs. Septic shock is a severe form of sepsis, where the body experiences abnormalities in its circulatory system, cells, and metabolism, leading to a high risk of death. Early treatment of sepsis is crucial for improving survival rates. However, traditional scoring systems like APACHE, SAPS, and SOFA, while helpful in assessing the severity of the condition and predicting survival, are not for early detection of sepsis. Recognizing the urgency of early identification and intervention, the (WHO) made sepsis a global health priority in 2017. The WHO adopted a resolution during the 70th World Health Assembly to develop guidelines and support countries in the early diagnosis, prevention, and management of sepsis, highlighting its importance as a priority.

1.2 PURPOSE OF THE PROJECT

Sepsis can progress to severe stages, including severe cases of septic shock, which are associated with higher mortality rates. Early detection provides an opportunity to intervene and prevent the progression to these more severe and life-threatening stages. Aim to reduce mortality rates by addressing sepsis at its initial stages where treatments are more effective.

1.2 SCOPE OF THE PROJECT

The primary goal of using ML for sepsis prediction is to come up with models which can accurately estimate a patient's expected lifespan following thoracic surgery. These models leverage various data sources, including patient information, surgical factors, and medical history, to make predictions. It is essential to acknowledge the importance of ethical considerations when implementing machine learning in healthcare settings, such as data privacy and transparency. These predictive models should be used as supportive tools for healthcare professionals, and clinical judgment should remain a crucial factor in the decision-making process.

1.3 DEFINITIONS

1.3.1 Machine Learning

Machine learning is a branch of AI that allows systems to enhance their capabilities by learning from data and experiences, rather than relying solely on explicit programming. It helps in the development of algorithms that allow machines to identify patterns, make predictions, and improve their performance on specific tasks over time, without the need for manual programming of every scenario.

The ability of machine learning to derive insights and knowledge from data has made it a critical component of numerous technological advancements across various industries. As the volume and complexity of data increases an indispensable tool for solving intricate problems, uncovering hidden patterns, and making accurate predictions based on the available data. Its applications span a wide range of domains, making an powerful and versatile technology with far-reaching implications.

1.3.2 Deep Learning

Deep learning is a specialized field within ML that involves neural networks comprising 3 or more layers. These neural networks aim to mimic the behavior of the human brain with limitations compared to its full capabilities, enabling them to learn from vast datasets. Whereas a neural network with a single layer can provide rough predictions, incorporating extra hidden layers can enhance accuracy by optimizing and refining the learning process.

1.3.3 Neural Network

A neural network, also known as an (ANN) or deep neural network, is a ML model designed to emulate the structure and operation of the human brain. These networks consist of interconnected nodes, or neurons, working together to address complex problems. Neural networks fall under the umbrella of (AI) and are a variety of deep learning technology.

Typically, ANN involve multiple processes operating simultaneously and organized into layers. The initial layer, akin to the optic nerves in human eye processing, receives the raw input data. Each subsequent layer obtain the output from the preceding layer rather than the raw input, similar to how neurons farther from the optic nerve receive signals from closer to it. The final layer produces the system's output. On the other end, a decision tree is depicted upside down, with its root at the top. In a decision tree diagram, the bold black text represents conditions or internal nodes, which determine the tree's branching into edges or branches. At the end of a branch that no longer splits, we find the decision or leaf, such as whether a passenger survived or died, typically depicted with red and green text, respectively.

1.3.3 LSTM

LSTM (Long Short-Term Memory) is a recurrent neural network (RNN) architecture widely used in Deep Learning. It excels at capturing long-term dependencies, making it ideal for sequence prediction tasks. Unlike traditional neural networks, LSTM incorporates feedback connections, allowing it to process entire sequences of data, not just individual data points. This makes it highly effective in understanding and predicting patterns in sequential data like time series , text , and speech. LSTM has become a powerful tool in artificial intelligence and deep learning, enabling breakthroughs in various fields by uncovering valuable insights from sequential data .

1.1.1 Data-Preprocessing

Data preprocessing is a methodology employed to transform raw data into a refined dataset suitable for analysis. When data is sourced from various origins, it is often in a raw format unsuitable for analysis. Hence, a series of procedures are undertaken to condense the data into a more manageable and refined form. Data preprocessing is conducted prior to iterative analysis and encompasses a set of procedures aimed at enhancing data authenticity and usability.

1.1.2 Data Validation

Data validation includes verifying that data has been through a process of data cleansing to guarantee its quality, ensuring it is accurate and valuable. This process employs routines, often referred to as "validation rules," "validation constraints," or "check routines," to assess the correctness, significance, and security of input data. These rules can be implemented through automated features provided by a data dictionary or by incorporating explicit validation logic within the computer system and its applications.

1.1.3 Exploratory Data Analysis

(EDA) is a method used to examine datasets and summarize their primary attributes, typically employing statistical graphics and various data visualization techniques. While a statistical model may or may not be utilized, EDA primarily focuses on uncovering insights from the data beyond formal modeling. As such, it serves as an alternative to traditional hypothesis testing by emphasizing a more exploratory and descriptive approach.

1.1.4 Python

Python stands out as a widely-used programming language, conceived by Guido van Rossum and introduced in 1991. It operates as an interpreted, object-oriented, high-level language with dynamic semantics. Python's appeal lies in its built-in high-level data structures, complemented by dynamic typing and dynamic binding, rendering it exceptionally suitable for swift application development.

Chapter 2

LITERATURE SURVEY

2.1 Machine learning for finding sepsis: a systematic review and meta-analysis of diagnostic test accuracy.

Detecting sepsis in the initial stages can pose a challenge in clinical settings. However, with the advancement of machine learning, promising real-time predictive model of sepsis have emerged. Our search encompassed PubMed, Embase.com, and Scopus databases. Eligible studies included those focusing on sepsis1, sepsis2 and sepsis 3 across various hospital settings. The index test comprised any supervised ML model designed for real-time prediction of these conditions. Quality tests are conducted using the Grading of Recommendations Assessment, Development, and Evaluation (GRADE) methodology, supplemented by a tailored Quality Assessment of Diagnostic Accuracy Studies (QUADAS-2) checklist to gauge bias risk. Meta-analysis of models reporting the AUC of the (AUROC) metric was performed to identify the most impactful contributors to model performance.

2.2 Feature selection may aid DNN to provide still better results for the bioinformatics problems.

DNN algorithms were utilized in predicting various biomedical phenotypes recently, and demonstrated very good prediction performances without selecting features. This study proposed a hypothesis that the DNN models may be further improved by feature selection algorithm.

2.3 Learning representations and patterns for the early detecting of sepsis using deep neural networks.

Sepsis stands as a prominent contributor to mortality among patients in intensive care units. Prompt identification of sepsis assumes critical importance due to the escalation of mortality rates with the progression of sepsis severity. This research endeavors to construct detection models targeting the early stages of sepsis by employing deep learning techniques. Additionally, the study aims to find the feasibility and efficacy of the new deep learning approaches in comparison to conventional regression methods coupled with standard temporal feature extraction. Adherence to the InSight model was observed in selecting the study group. The outcomes of the deep learning-based models were juxtaposed with those of the InSight model for comparison.

2.4 Predicting of sepsis in the ICU with minimal EHR data: a ML approach.

InSight, a ML classification system, which utilizes a mix of easily accessible patient data (including vital signs, peripheral capillary oxygen saturation, Glasgow Coma Score, and age), to predict sepsis. This prediction is based on retrospective (MIMIC)-III dataset, specifically focusing on (ICU) patients aged 15 years or older. Following the Sepsis-3 criteria for defining sepsis, we assess the classification performance of InSight against several scoring systems, including (qSOFA), (MEWS), systemic inflammatory response syndrome (SIRS), simplified acute physiology score (SAPS) II, and (SOFA). This comparison aims to determine the predictive accuracy of these systems in identifying patients who will be developing sepsis within a fixed time period before onset. Additionally, we evaluate robustness of the InSight system by testing its performance under conditions of random deletion of individual input observations.

2.5 A targeted real time early alarming score for septic shock.

Sepsis ranks as a primary cause of mortality in the United States, with the highest death rates seen among patients progressing to septic shock. While existing automated screening tools effectively identify patients currently experiencing severe sepsis or septic shock, none have the capability to predict those at the greatest risk of progressing to shock. In this study, we analyzed readily available physiological and laboratory data from ICU patients to develop "TREWScore," a targeted real-time early alarming score designed to forecast which patients will develop septic shock. TREWScore demonstrated the ability to identify patients before the onset of septic shock with an area under the ROC curve (AUC) of 0.83 (95% CI, 0.81 to 0.85). At a specificity of 0.67, TREWScore achieved a sensitivity of 0.85, identifying patients a median of 28.2 hours (IQR, 10.6 to 94.2) before onset. Notably, two-thirds of the identified patients were flagged before any sepsis-related organ dysfunction occurred. In contrast, the Modified Early Warning Score, commonly used for septic shock prediction, exhibited a lower AUC of 0.73 (95% CI, 0.71 to 0.76). A routine screening protocol based on the presence of 2 SIRS criteria, suspicion of infection, and either hypotension or hyperlactatemia achieved a sensitivity of 0.74 at a comparable specificity of 0.64. By continuously sampling data from EHR and calculating TREWScore, clinicians may be able to find patients at risk for septic 3 and deliver earlier interventions aimed at preventing or mitigating associated morbidity and mortality.

2.6 MIMIC-III, a freely accessible critical care database.

MIMIC-III stands as an extensive repository, encompassing data from patients admitted to critical care units at a prominent tertiary care hospital. This comprehensive dataset comprises a wide array of information, including vital signs, medication records, laboratory results, clinical observations, fluid balance, procedural codes, imaging reports, hospital duration, survival statistics, and additional details. The database serves multiple purposes, facilitating academic and industrial research endeavors, quality enhancement projects, and educational initiatives at various levels of higher education .

2.7 An attention-based deep learning model of clinical events in the ICU.

In this project, (LSTM) (RNNs) were trained, building an inbuilt attention mechanism, to forecast daily occurrences of sepsis, myocardial infarction (MI), and vancomycin antibiotic administration throughout two-week patient ICU stays using the MIMIC-III dataset. These models achieved an impressive predictive performance, with next-day predictive (AUC) values of 0.87 for sepsis, 0.82 for MI, and 0.83 for vancomycin administration. Utilizing attention maps generated by these models, key instances where input variables significantly influenced predictions were highlighted, offering a level of interpretability for clinicians. Notably, these models appeared to focus on variables indicative of clinician decision-making, illustrating a challenge encountered when employing flexible deep learning methods trained with electronic health record (EHR) data for constructing clinical decision support systems. Although further refinement and advancement are warranted, we anticipate that such models could potentially alleviate information load for ICU physicians by furnishing essential clinical decision support across various critical tasks.

2.8 Machine learning predicts death in septic patients using only available ABG variables: a multi centre evaluation.

The study aims in assessing the effectiveness of ML methods, particularly Deep Neural Networks (DNN) models, in predicting death of ICU patients. The objective was to forecast mortality within 96 hours following admission, mirroring the clinical scenario of patient evaluation post-ICU trial, which typically involves 24-48 hours of ICU treatment followed by "re-triage". To enhance real-world applicability, the input variables were intentionally limited to arterial blood gas (ABG) values. We conducted a retrospective evaluation on septic patients using both the multi-center eICU dataset and the single-center MIMIC-III dataset.

2.9 A deep learning approach for sepsis monitoring via criticality score estimation.

Sepsis arises as a response to infection within the body and can escalate to a critical stage. The detection and monitoring of sepsis typically involve a multi-step process, which is time-consuming, expensive, and necessitates trained medical personnel. The (SOFA) score is a key metric for assessing the severity of sepsis, heavily relying on laboratory measurements. In this study, we propose a computational approach for quantitatively monitoring sepsis symptoms and organ system status without the need for laboratory tests. Specifically, we suggest employing a regression-based analysis utilizing seven vital signs readily available at the bedside in the (ICU) to predict the SOFA score of patients prior to the onset of sepsis. To achieve this, we introduce the Deep SOFA-Sepsis Prediction Algorithm (DSPA). This model combines features obtained from (CNN) with the Random Forest (RF) algorithm to predict SOFA scores for sepsis patients. We conduct experiments using a subset of the Medical Information Mart in Intensive Care (MIMIC) III dataset, comprising 5154 samples as input. The experiments involve ten-fold cross-validation tests to evaluate model performance.

2.10 Evaluate prognostic accuracy of SOFA component score for mortality in adults having sepsis by machine learning method.

Patients treated with sepsis upon admission to the ICU were selected from the MIMIC-IV database for retrospective analysis. They were then allocated into a training set and a test set at a ratio of 2:1. This study included six variables, all derived from the scores of six organ systems in the (SOFA) score. ML models were trained using the training set and subsequently evaluated using the validation set. Six machine learning methods were employed, including linear regression analysis, least absolute shrinkage and selection operator (LASSO), logistic regression analysis (LR), Gaussian Naive Bayes (GNB), and (SVM), to develop models predicting the risk of mortality. The accuracy, area under the ROC curve (AUROC), Decision Curve Analysis (DCA), and K-fold cross-validation were utilized to assess the predicting performance of the developed models.

Chapter 3

PROBLEM IDENTIFICATION

3.1 Problem Statement

Sepsis Prediction involves predicting an infection with the support of a Deep Learning model by providing the required data as input , the dataset contains required clinical variables data of patients present in the form of cohorts , which are collected from ICU's and ED's EHR by performing lab tests on individual patient. This model assists clinicians in making informed decisions and providing the proper anti-biotics that lead to overall improvement in patient health.

3.2 Project Scope

Our proposed approach emphasizes harnessing extensive patient data, enabling the models to detect subtle patterns and signs of sepsis that evade human clinicians' notice. Fast detection of sepsis facilitates prompt interventions, like administering appropriate antibiotic treatment and fluid resuscitation, leading to notable reductions in mortality rates. Nevertheless, the successful integration of ML based sepsis predicting models into clinical practice necessitates addressing several challenges.

Chapter 4

GOALS AND OBJECTIVES

4.1 Project Goals

- To improve the accuracy of Sepsis Prediction.
- To identify the proper clinical variables required to predict sepsis.
- To significantly reduce mortality rates.
- To speed up the process of predicting sepsis prior to its onset for saving lives of patients.
- To develop a cost effective model.

4.2 Project Objectives

1. To extract and store datasets.
2. To apply model features.
3. Fill in missing values.
4. Setting up of prediction and observation windows.

Chapter 5

SYSTEM REQUIREMENT SPECIFICATION

4.1 Software Requirements Analysis

The software requirements definition is an abstract description of the services, which the system should be able to provide, and the constraints under which the system must operate. It should only specify only the external spec of system and is not linked with system design characteristics. It is an solution, in a natural language plus diagrams, of what services the system is expected to provide and the constraints under which it must operate.

Software Requirements

- **Operating system** : Windows 7 / 8 / 10
- **Coding Language** : Python,HTML,CSS
- **Software** : Anaconda
- **IDE** : Jupyter Notebook/Google collab

4.2 Hardware Requirements Analysis

Hardware Requirements is to define and analyze a complete set of functional, operational, performance, interface, quality factors, and design, criticality and test requirements.

Hardware Requirements

- **Processor** : Intel 3rd generation or high
- **Hard Disk** : 500 GB SSD
- **Ram** : 8 GB
- Any desktop or Laptop with higher configuration

Chapter 6

METHODOLOGY

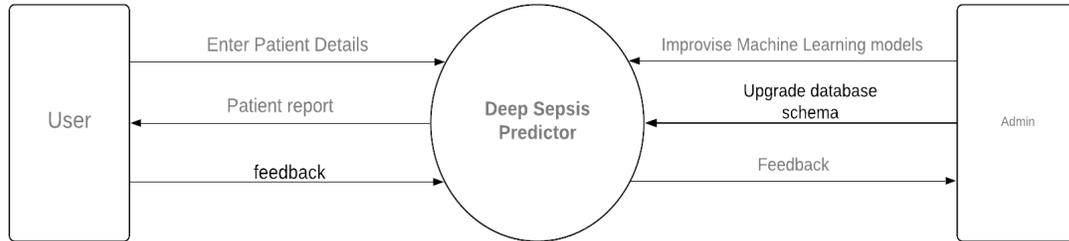


Fig 6.1: Data flow diagram LEVEL 0

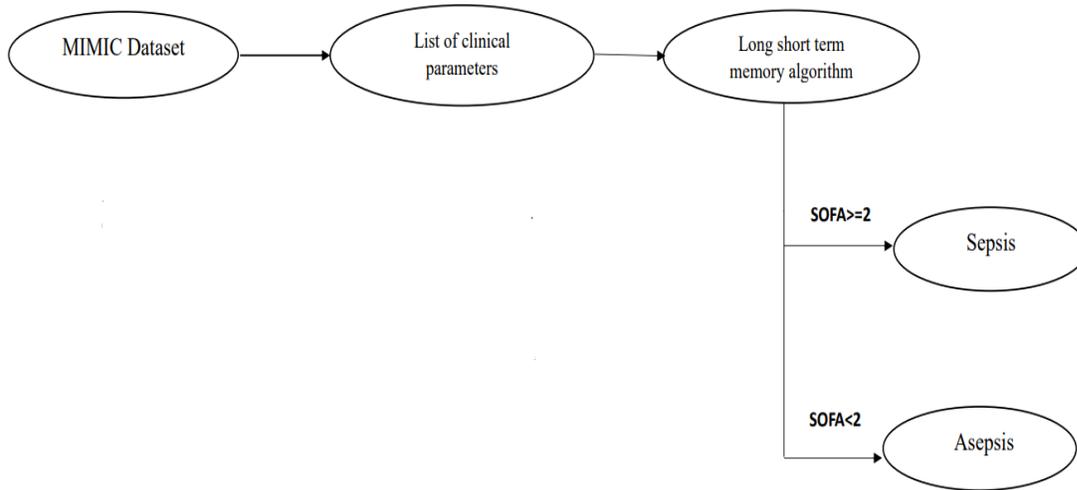


Fig 6.1.1: Data flow diagram LEVEL 1

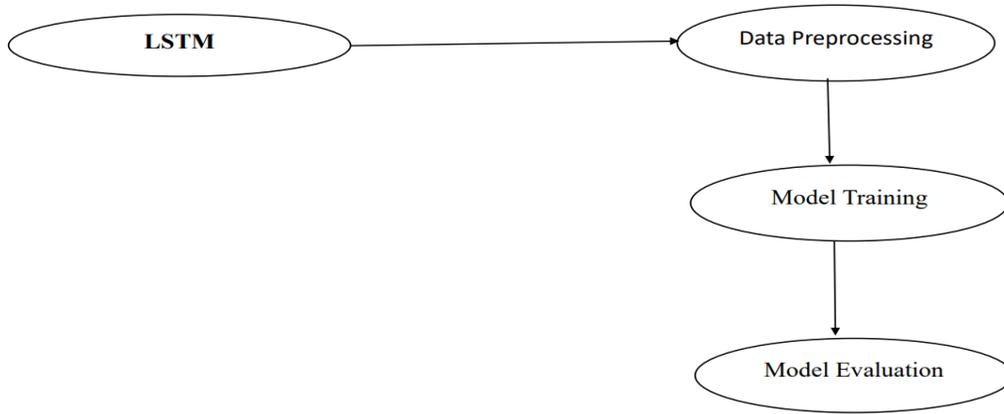


Fig 6.1.2: Data flow diagram LEVEL 2

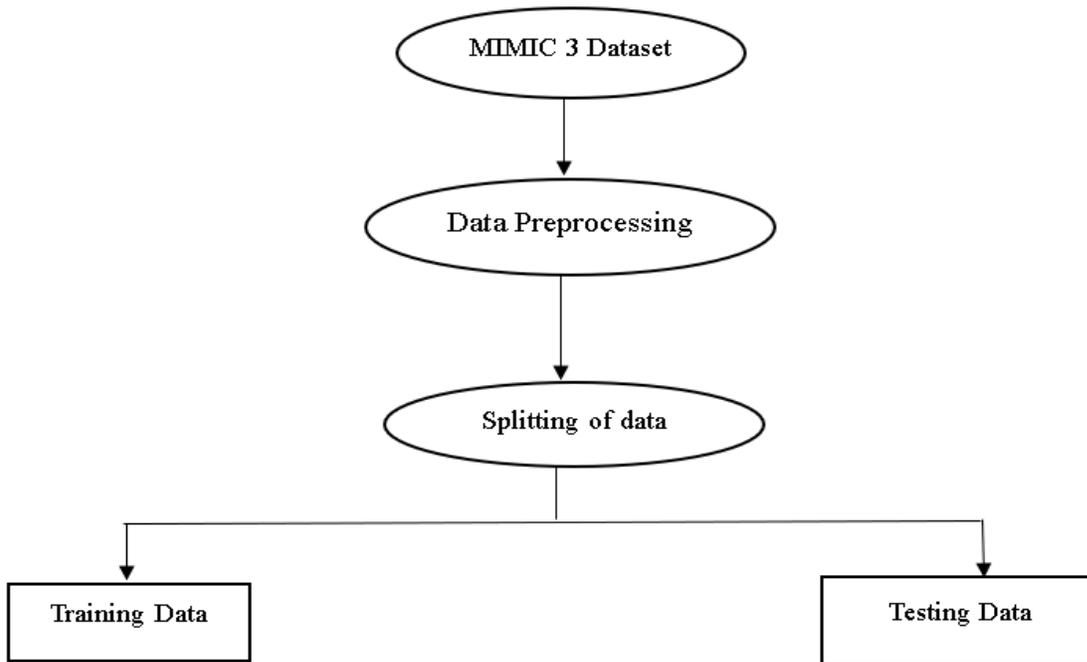


Fig 6.1.3: Data flow diagram LEVEL 2.1

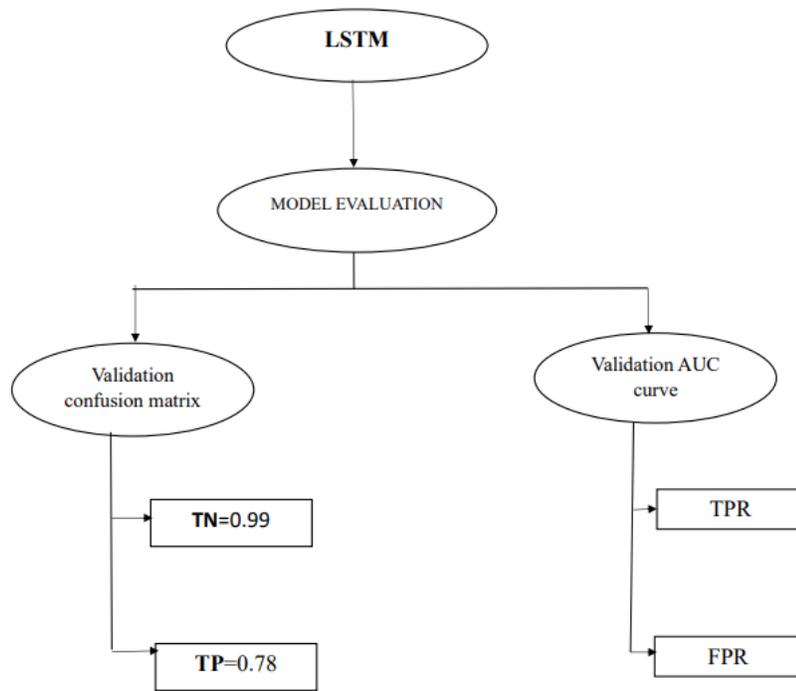


Fig 6.1.4: Data flow diagram LEVEL 2.2

CHAPTER 7

IMPLEMENTATION

7.1 FILES USED

- **Jupyter Notebook:**
 - **project_sepsis.ipynb**
- **Google Collab**

7.2 MODULES AND THEIR ROLES

7.2.1 PYTHON CODE FOR TODDLER DATASET

Import Libraries

```
import os
```

```
from torch.utils.data import Dataset, WeightedRandomSampler
```

```
from torch.nn.utils.rnn import pack_padded_sequence, pad_packed_sequence
```

Extract Transform Load (ETL)

```
def load_labels(labels_fn):
```

```
    labels_df = pd.read_csv(labels_fn, usecols=["icustay_id", "sepsis-3"], index_col=["icustay_id"])
```

```
    labels = labels_df["sepsis-3"]
```

```
"""OneHotEncode categorical variables and drop less useful features"""
```

```
df.fillna(0, inplace=True)
```

```
numerical_features = ['age', 'v_heartrate', 'v_sysbp', 'v_resprate', 'v_shockindex', 'v_fio2', 'v_gcs',  
                      'v_diasbp', 'v_meanbp', 'v_tempc', 'v_spo2', 'v_glucose', 'l_BUN', 'l_WBC',  
                      'l_CREATININE', 'l_BUN_CREATININE_RATIO', 'l_PH', 'l_PO2', 'l_liver']
```

```
df[numerical_features] = MinMaxScaler().fit_transform(df[numerical_features])
```

```
df["gender"] = df["gender"].apply(lambda g: 1 if g=='M' else 0)
```

```
data = df
```

```
data.drop(columns=["subject_id", "ethnicity", "subject_id", "hour", "timestamp", "sofa_24hours",
                  "Sepsis3_start_flg", "sepsis_onset_hr"], inplace=True)

return data

def load_preprocess_features(features_fn):
    df = pd.read_csv(features_fn)
    data = preprocess_data(df)
    return data

def get_sequence(group, id):
    seq_df = group.get_group(id)
    label = seq_df["Sepsis3_diag_flg"].values[-1]
    seq_df = seq_df.drop(columns=["subject_id", "Sepsis3_diag_flg"])
    sequence = seq_df.values.astype('float32')
    return sequence, label

def visit_collate_fn(batch):
    """Sort batch sequences by size and pad zeros"""
    sorted_batch = sorted(batch, key=lambda tup: tup[0].shape[0], reverse=True)
    sequences = [tup[0] for tup in sorted_batch]
    seqs_tensor = torch.nn.utils.rnn.pad_sequence(sequences, batch_first=True)
    lengths = torch.LongTensor([tup[0].shape[0] for tup in sorted_batch])
    labels = torch.LongTensor([tup[1] for tup in sorted_batch])
    ids = torch.LongTensor([tup[2] for tup in sorted_batch])
    return (seqs_tensor, lengths), labels, ids
```

Data Loading

```
class Mimic3InMemoryDataset(Dataset):
    def __init__(self, features_fn, labels_fn):
        self.data = load_preprocess_features(features_fn)
        self.group = self.data.groupby("icustay_id")
        self.ids = self.group.size().index.to_list()
        self.num_features = self.get_num_features()

    def __len__(self):
        return len(self.ids)

    def __getitem__(self, idx):
        id = self.ids[idx]
```

```
sequence, label = self.get_sequence(id)
```

```
return sequence, label, id
```

```
def get_sequence(self,id):
```

```
    """Get sequence and label from the grouped data-frame given icustay_id"""
```

```
    seq_df = self.group.get_group(id)
```

```
    label = seq_df["Sepsis3_diag_flg"].values[-1]
```

```
    seq_df = seq_df.drop(columns=["icustay_id","Sepsis3_diag_flg"])
```

```
    sequence = seq_df.values.astype('float32')
```

```
    sequence = torch.from_numpy(sequence)
```

```
    return sequence, label
```

```
def get_num_features(self):
```

```
    seq,_ = self.get_sequence(self.ids[0])
```

```
    return seq.shape[1]
```

```
class MyLipSep(nn.Module):
```

```
    def __init__(self, dim_input):
```

```
        super(MyLipSep, self).__init__()
```

```
        n_hidden_size = 50
```

```
        n_layers = 4
```

```
        self.rnn = nn.LSTM(input_size=dim_input, hidden_size= n_hidden_size, num_layers=n_layers,
```

```
batch_first=True,dropout=0.2)
```

```
        self.out = nn.Linear(in_features= n_hidden_size, out_features=1)
```

```
    def forward(self, x):
```

```
        isTuple = isinstance(x,tuple)
```

```
        if isTuple:
```

```
            x, lengths = x
```

```
        if isTuple:
```

```
            x = pack_padded_sequence(x, lengths, batch_first = True)
```

```
            x, _ = self.rnn(x)
```

```
        if isTuple:
```

```
            x,_ = pad_packed_sequence(x, batch_first=True)
```

```
            x = self.out(x[:, -1, :])
```

```
            x = torch.sigmoid(x)
```

```
            x = x.view(-1)
```

```
            return x
```

```
class MyLipSepFC(nn.Module):
```

```
def __init__(self, dim_input):
    super(MyLipSepFC, self).__init__()
    n_fc1_out = 1028
    n_fcpost_out = 32
    n_hidden_size = 100
    n_layers = 4
    self.fc1 = nn.Linear(in_features= dim_input, out_features=n_fc1_out)
    self.rnn = nn.LSTM(input_size=n_fc1_out, hidden_size= n_hidden_size, num_layers=n_layers,
        batch_first=True,dropout=0.2)
    self.fcpost = nn.Linear(in_features= n_hidden_size, out_features=n_fcpost_out)
    self.out = nn.Linear(in_features= n_fcpost_out, out_features=1)

def forward(self, x):
    isTuple = isinstance(x,tuple)
    if isTuple:
        x, lengths = x
        x = torch.tanh(self.fc1(x))
    if isTuple:
        x = pack_padded_sequence(x, lengths, batch_first = True)
        x, _ = self.rnn(x)
    if isTuple:

x = torch.relu(self.fcpost(x[:, -1, :]))
    x = self.out(x)
    x = torch.sigmoid(x)
    x = x.view(-1)
    return x
```

Training the Model

```
class AverageMeter(object):
    """Computes and stores the average and current value"""
    def __init__(self):
        self.reset()
    def reset(self):
        self.val = 0
```

```
self.avg = 0
self.sum = 0
self.count = 0
def update(self, val, n=1):
    self.val = val
    self.sum += val * n
    self.count += n
    self.avg = self.sum / self.count

def compute_batch_accuracy(output, target):
    """Computes the accuracy for a batch"""

    with torch.no_grad():
        batch_size = target.size(0)
        pred = output.round()
        correct = pred.eq(target).sum()
        return correct * 100.0 / batch_size

def train(model, device, data_loader, criterion, optimizer, epoch, print_freq=10):
    batch_time = AverageMeter()
    data_time = AverageMeter()

losses = AverageMeter()
    accuracy = AverageMeter()
    model.train()
    end = time.time()

    for i, (input, target, _) in enumerate(data_loader):
        data_time.update(time.time() - end)
        if isinstance(input, tuple):
            input = tuple([input[0].to(device), input[1]])
        else:
            input = input.to(device)
        target = target.to(device)
```

```
optimizer.zero_grad()
output = model(input)
loss = criterion(output, target.float())
assert not np.isnan(loss.item()), 'Model diverged with loss = NaN'
loss.backward()
optimizer.step()
batch_time.update(time.time() - end)

end = time.time()

losses.update(loss.item(), target.size(0))
accuracy.update(compute_batch_accuracy(output, target).item(), target.size(0))
if i % print_freq == 0:
    print ('Epoch: [{0}][{1}/{2}]\t'
          'Time {batch_time.val:.3f} ({batch_time.avg:.3f})\t'
          'Data {data_time.val:.3f} ({data_time.avg:.3f})\t'
          'Loss {loss.val:.4f} ({loss.avg:.4f})\t'
          'Accuracy {acc.val:.3f} ({acc.avg:.3f})'.format(
                epoch, i, len(data_loader), batch_time=batch_time,
                data_time=data_time, loss=losses, acc=accuracy))
```

```
def evaluate(model, device, data_loader, criterion, print_freq=10):
```

```
    batch_time = AverageMeter()
    losses = AverageMeter()
    accuracy = AverageMeter()
    results = []
    model.eval()
    with torch.no_grad():
        end = time.time()
        for i, (input, target, _) in enumerate(data_loader):
            if isinstance(input, tuple):
                input = tuple([input[0].to(device), input[1]])
            else:
```

```
input = input.to(device)
target = target.to(device)
output = model(input)
loss = criterion(output, target.float())
batch_time.update(time.time() - end)
end = time.time()
losses.update(loss.item(), target.size(0))
accuracy.update(compute_batch_accuracy(output, target).item(), target.size(0))
y_true = target.detach().to('cpu').numpy().tolist()
y_prob = output.detach().to('cpu').numpy().tolist()
y_pred = output.detach().to('cpu').round().int().numpy().tolist()
results.extend(list(zip(y_true, y_prob, y_pred)))
if i % print_freq == 0:
    print('Test: [{0}/{1}]\t'
          'Time {batch_time.val:.3f} ({batch_time.avg:.3f})\t'
          'Loss {loss.val:.4f} ({loss.avg:.4f})\t'
          'Accuracy {acc.val:.3f} ({acc.avg:.3f})'.format(
            i, len(data_loader), batch_time=batch_time, loss=losses, acc=accuracy))
return losses.avg, accuracy.avg, results
```

Plots

```
def plot_learning_curves(train_losses, valid_losses, train_accuracies, valid_accuracies):
    """plots learning curves"""
    plt.figure()
    plt.plot(train_losses, label='Train Loss')
    plt.plot(valid_losses, label='Validation Loss')
    plt.legend()
    plt.xlabel('epoch')
    plt.ylabel('Loss')
    plt.title('Loss Curve')
    plt.savefig('loss_curve')
    plt.show()
    plt.figure()
    plt.plot(train_accuracies, label='Train Accuracy')
    plt.plot(valid_accuracies, label='Validation Accuracy')
```

```
plt.legend()
plt.xlabel('epoch')
plt.ylabel('Accuracy')
plt.title('Accuracy Curve')
plt.savefig('accuracy_curve')
plt.show()
```

```
def plot_confusion_matrix(y_true,y_pred, class_names):
    """plots confusion matrix"""
    cm = metrics.confusion_matrix(y_true, y_pred)
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=plt.get_cmap('Blues'))
    plt.title('Normalized Confusion Matrix')
    plt.colorbar()
    tick_marks = np.arange(len(class_names))
    plt.xticks(tick_marks, class_names, rotation=45)

    thresh = cm.max() / 1.5
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, "{:0.4f}".format(cm[i, j]),
                horizontalalignment="center",
                color="white" if cm[i, j] > thresh else "black")
    plt.ylabel('True')
    plt.xlabel('Predicted')
    plt.tight_layout()
    plt.savefig('confusion_matrix')
    plt.show()
```

Inference

```
def get_last_prediction_from_sequence(model, sequence):
    with torch.no_grad():
        if len(sequence.shape) == 2:
```

```
prob = model(sequence.unsqueeze(0)).item()
else:
    prob = model(sequence).item()
return round(prob,4)
def get_timestamp_predictions_from_sequence(model, sequence, step_size=5, start_timestamp=5,
timestamps=None):
    if timestamps is None:

        timestamps = list(range(start_timestamp,sequence.shape[0],step_size))
    predictions = [get_last_prediction_from_sequence(model, sequence[:t]) for t in timestamps]
    return predictions
def get_label_from_probability(probability, threshold=0.5):
    return 1 if probability > threshold else 0
def compile_dataset_predicitons(model, dataset, device, threshold=0.5, step_size=10,
start_timestamp=5, timestamps=None):

df= pd.DataFrame(columns=['dataset_idx','id','label','last_prob','last_pred','timestamps',
                        'probabilities'])
for idx in range(0,len(dataset)):

    sequence, label, id = dataset[idx]
    sequence = sequence.to(device)
    last_prob = get_last_prediction_from_sequence(model, sequence)
    last_pred = get_label_from_probability(last_prob, threshold)
    if timestamps is None:
        timestamps_used = list(range(start_timestamp,sequence.shape[0],step_size))
    else:
        timestamps_used = timestamps
    probabilities = get_timestamp_predictions_from_sequence(model, sequence,
timestamps=timestamps_used)
    df = df.append({
        'dataset_idx': idx,
        'id': id,
        'label': label,
        'last_prob': last_prob,
```

```
'last_pred': last_pred,
'timestamps': timestamps_used,
'probabilities': probabilities
}, ignore_index=True)
return df
def visualize_timeseries_predictions(df, threshold = None):
plt.figure(figsize=(10,6))
min_time = min(df['timestamps'].apply(min))
max_time = max(df['timestamps'].apply(max))
plt.hlines([0,1], min_time, max_time, 'k', '--')
if threshold is not None:
plt.hlines(threshold, min_time, max_time, 'r', '--')
for i in range(0,len(df)):
plt.plot(df['timestamps'][i], df['probabilities'][i],
label=f"Sepsis: {df['label'][i]} ID: {df['id'][i]}")
plt.legend(loc='upper left')
plt.xlabel('timestamps')
```

Main

```
LABELS_PATH = 'C:/Users/LSTM-based-Sepsis-Prediciton-main/Datasets/sepsis3-df.csv'
FEATS_PATH = 'C:/Users/LSTM-based-Sepsis-Prediciton-main/Datasets/features_v1.csv'
DATA_PATH = "./physionet_format"
NUM_EPOCHS = 100
BATCH_SIZE = 10
LEARNING_RATE = 0.0001
USE_CUDA = True # Set 'True' if you want to use GPU
NUM_WORKERS = 2
PATH_OUTPUT = "/output/"
os.makedirs(PATH_OUTPUT, exist_ok=True)
device = torch.device("cuda" if torch.cuda.is_available() and USE_CUDA else "cpu")
if device.type == "cuda":
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False
```

```
print("Device: " + device.type)
dataset = Mimic3InMemoryDataset(FEATS_PATH,LABELS_PATH)
dataset_size = len(dataset)
lengths = [round(dataset_size*0.75), round(dataset_size*0.1), round(dataset_size*0.15)-1]
train_set, valid_set, test_set = torch.utils.data.random_split(dataset, lengths)
dataloader      =      DataLoader(dataset,      batch_size=BATCH_SIZE,      shuffle=False,
collate_fn=visit_collate_fn, num_workers=NUM_WORKERS)
train_loader     =      DataLoader(train_set,     batch_size=BATCH_SIZE,     shuffle=True,
collate_fn=visit_collate_fn, num_workers=NUM_WORKERS)

valid_loader     =      DataLoader(valid_set,     batch_size=BATCH_SIZE,     shuffle=True,
collate_fn=visit_collate_fn, num_workers=NUM_WORKERS)
model = MyLipSepFC(dataset.num_features)
criterion = nn.BCELoss()
optimizer = optim.Adam(model.parameters(),lr = LEARNING_RATE)
model.to(device)
criterion.to(device)

best_val_acc = 0.0
train_losses, train_accuracies = [], []
valid_losses, valid_accuracies = [], []
for epoch in range(NUM_EPOCHS):
    train_loss, train_accuracy = train(model, device, train_loader, criterion, optimizer, epoch)
    valid_loss, valid_accuracy, valid_results = evaluate(model, device, valid_loader, criterion)
    train_losses.append(train_loss)
    valid_losses.append(valid_loss)
    train_accuracies.append(train_accuracy)
    valid_accuracies.append(valid_accuracy)
    is_best = valid_accuracy > best_val_acc
    if is_best:
        best_val_acc = valid_accuracy
        torch.save(model, os.path.join(PATH_OUTPUT, "MyVariableRNN.pth"))
best_model = torch.load(os.path.join(PATH_OUTPUT, "MyVariableRNN.pth"))
plot_learning_curves(train_losses, valid_losses, train_accuracies, valid_accuracies)
```

Validation Dataset Performance

```
y_true = [t[0] for t in valid_results]
y_prob = [t[1] for t in valid_results]
y_pred = [t[2] for t in valid_results]
print("ACC: ", metrics.accuracy_score(y_true,y_pred))
print("AUC: ", metrics.roc_auc_score(y_true,y_prob))
print("Confusion Matirx:")
print(metrics.confusion_matrix(y_true,y_pred))
plot_confusion_matrix(y_true,y_pred, ['No-Sepsis','Sepsis'])
df = compile_dataset_predicitons(best_model,test_set,device)
df.head()
from sklearn import metrics
y_true = df['label'].apply(int).tolist()
y_pred = df['last_pred'].apply(int).tolist()
y_prob = df['last_prob'].tolist()
fpr, tpr, thresholds = metrics.roc_curve(y_true,y_prob)
plt.plot(fpr, tpr, linestyle='--')
plt.title('AUC Curve')

plt.xlabel('FPR')
plt.ylabel('TPR')
plt.show()
```

Chapter 8

RESULTS AND SNAPSHOTS

Training

```
In [32]:
best_val_acc = 0.0
train_losses, train_accuracies = [], []
valid_losses, valid_accuracies = [], []
for epoch in range(NUM_EPOCHS):
    train_loss, train_accuracy = train(model, device, train_loader, criterion, optimizer, epoch)
    valid_loss, valid_accuracy, valid_results = evaluate(model, device, valid_loader, criterion)

    train_losses.append(train_loss)
    valid_losses.append(valid_loss)

    train_accuracies.append(train_accuracy)
    valid_accuracies.append(valid_accuracy)

    is_best = valid_accuracy > best_val_acc # Let's keep the model that has the best accuracy, but you can also use ar
    if is_best:
        best_val_acc = valid_accuracy
        torch.save(model, os.path.join(PATH_OUTPUT, "MyVariableRNN.pth"))

best_model = torch.load(os.path.join(PATH_OUTPUT, "MyVariableRNN.pth"))
```

Streaming output truncated to the last 5000 lines.

Epoch: [46][760/825]	Time 0.009 (0.010)	Data 0.002 (0.002)	Loss 0.4877 (0.2299)	Accuracy 66.667 (91.984)
Epoch: [46][770/825]	Time 0.010 (0.010)	Data 0.002 (0.002)	Loss 0.0581 (0.2287)	Accuracy 100.000 (92.045)
Epoch: [46][780/825]	Time 0.009 (0.010)	Data 0.002 (0.002)	Loss 0.0995 (0.2300)	Accuracy 100.000 (91.976)
Epoch: [46][790/825]	Time 0.010 (0.010)	Data 0.002 (0.002)	Loss 0.0844 (0.2296)	Accuracy 100.000 (91.993)
Epoch: [46][800/825]	Time 0.009 (0.010)	Data 0.002 (0.002)	Loss 0.0634 (0.2323)	Accuracy 100.000 (91.927)
Epoch: [46][810/825]	Time 0.010 (0.010)	Data 0.002 (0.002)	Loss 0.1337 (0.2334)	Accuracy 100.000 (91.862)
Epoch: [46][820/825]	Time 0.011 (0.010)	Data 0.002 (0.002)	Loss 0.1966 (0.2316)	Accuracy 100.000 (91.961)
Test: [0/110]	Time 0.087 (0.087)	Loss 0.0408 (0.0408)	Accuracy 100.000 (100.000)	
Test: [10/110]	Time 0.009 (0.014)	Loss 0.0900 (0.2617)	Accuracy 100.000 (90.909)	
Test: [20/110]	Time 0.010 (0.011)	Loss 0.0663 (0.2475)	Accuracy 100.000 (92.063)	
Test: [30/110]	Time 0.005 (0.009)	Loss 0.5294 (0.2567)	Accuracy 66.667 (91.398)	
Test: [40/110]	Time 0.004 (0.008)	Loss 0.1047 (0.2326)	Accuracy 100.000 (92.683)	
Test: [50/110]	Time 0.006 (0.008)	Loss 0.0325 (0.2339)	Accuracy 100.000 (92.157)	

Fig 8.1: Training the data

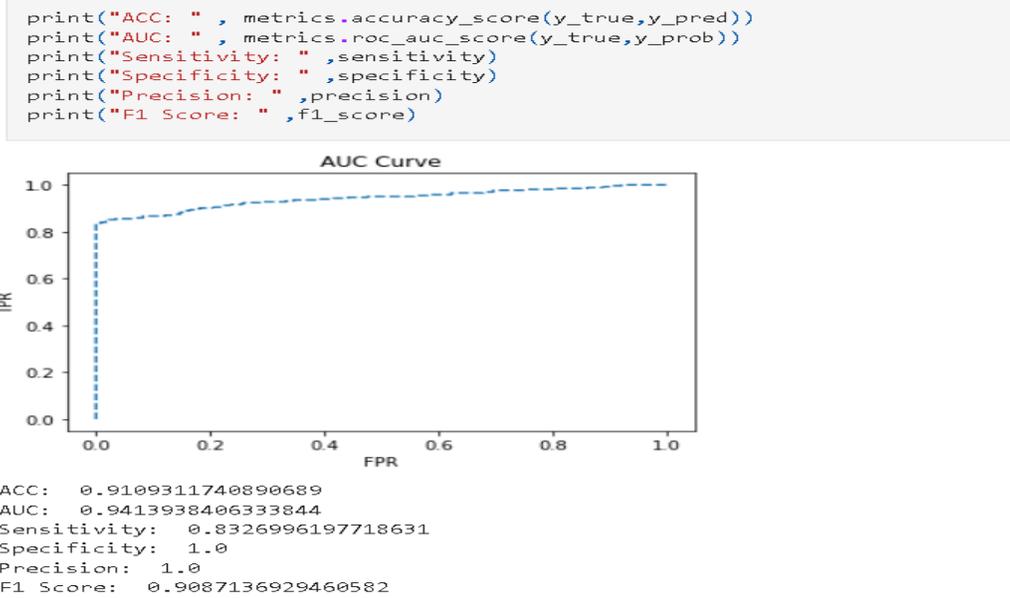


Fig 8.2: AUC Curve

	dataset_idx	id	label	last_prob	last_pred	timestamps	probabilities
0	0	238297	1	0.9997	1	[5]	[0.9997]
1	1	203559	1	0.8192	1	[]	[]
2	2	242284	0	0.0072	0	[5]	[0.9973]
3	3	229791	1	0.9969	1	[]	[]
4	4	274787	0	0.0149	0	[5]	[0.9966]
..
489	489	228387	1	0.9922	1	[]	[]
490	490	280362	1	0.9424	1	[]	[]
491	491	216100	1	0.9098	1	[5]	[0.9998]
492	492	271928	0	0.0025	0	[5]	[0.9989]
493	493	260593	1	0.9524	1	[]	[]

```

[494 rows x 7 columns]
At time 5 hours, the model predicts that patient with ID 238297 is septic.
No timestamps available for patient with ID 203559.
At time 5 hours, the model predicts that patient with ID 242284 is not septic.
No timestamps available for patient with ID 229791.
At time 5 hours, the model predicts that patient with ID 274787 is not septic.
No timestamps available for patient with ID 206547.
At time 5 hours, the model predicts that patient with ID 294299 is not septic.
At time 5 hours, the model predicts that patient with ID 208132 is not septic.
No timestamps available for patient with ID 270229.
At time 5 hours, the model predicts that patient with ID 214169 is not septic.
At time 5 hours, the model predicts that patient with ID 210521 is not septic.
No timestamps available for patient with ID 290622.
At time 5 hours, the model predicts that patient with ID 243222 is not septic.
At time 5 hours, the model predicts that patient with ID 249409 is not septic.
..

```

Fig 8.3: Predicted data

Chapter 9

APPLICATIONS

Sepsis prediction applications are designed to guide healthcare professionals and identify patients who are at the risk of getting affected by sepsis, a potentially life-threatening condition caused by the body's response to an infection. These applications typically utilize advanced algorithms and machine learning techniques to analyze patient data in real-time, allowing for fast detection and intervention. Some common features and sepsis prediction applications are:

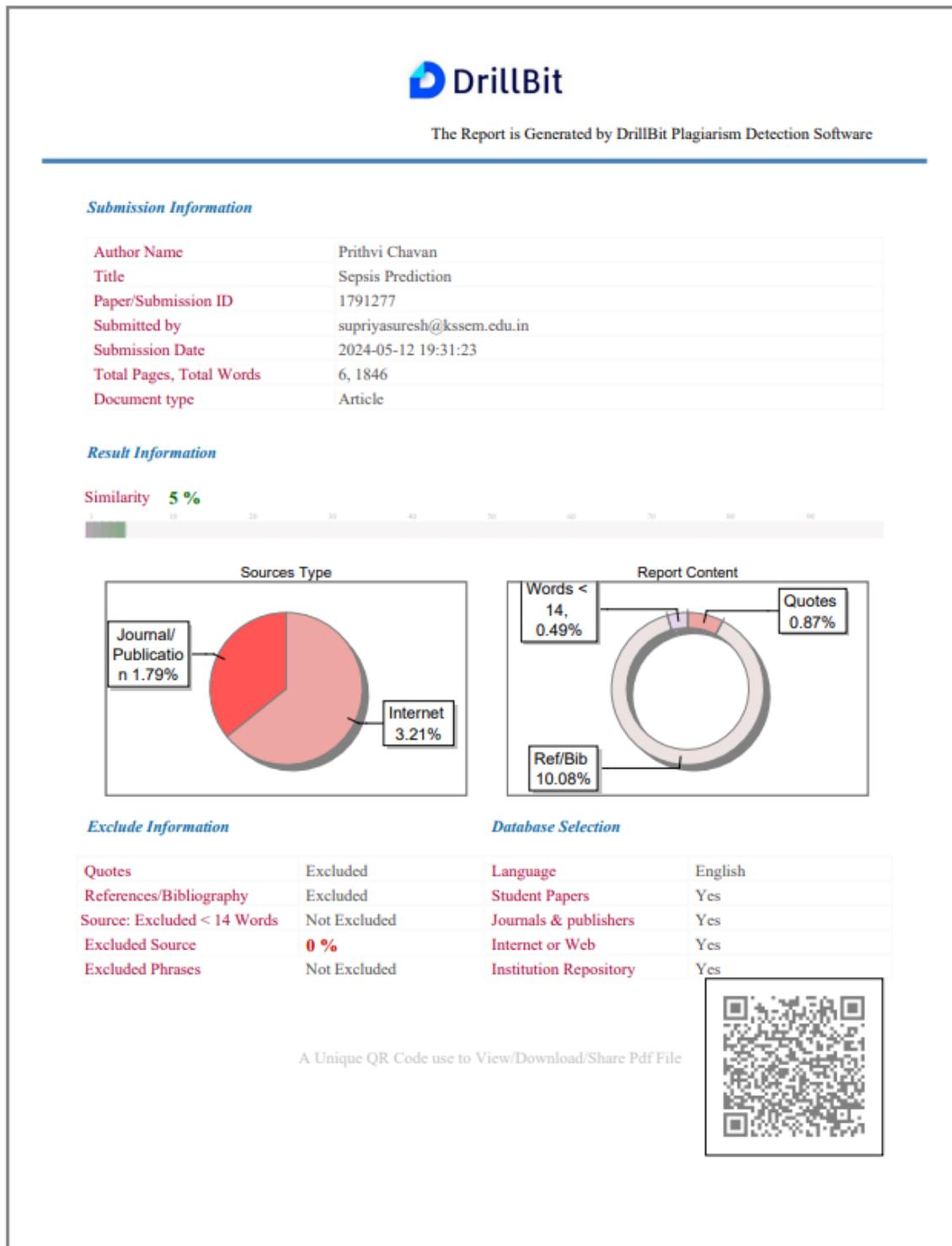
1. **Real-time Data Integration:** These applications integrate with electronic health records (EHR), monitoring devices, and other clinical systems to continuously collect and analyze patient data, vital signs, laboratory results, and clinical notes.
2. **Machine Learning Algorithms:** Advanced ML algorithms are used to identify patterns and trends in the data that indicate the onset of sepsis. These algorithms have capability to learn from historical data and adapt to new information to improve accuracy over time.
3. **Risk Stratification:** Patients are divided into different risk categories based on their likelihood of developing sepsis. This helps prioritize interventions and resources for those at highest risk.
4. **Alerting Mechanisms:** When a patient's data indicates a high risk of sepsis, the application generates alerts to notify healthcare providers. These alerts may be delivered through the EHR, mobile devices, or other communication channels.
5. **Clinical Decision Support:** Along with alerting, sepsis prediction applications may provide clinical decision tools to help guide interventions and treatment decisions. This may include diagnostic tests, fluid resuscitation, antibiotic therapy, and other interventions.
6. **Outcome Tracking:** The application also track outcomes such as mortality rates, length hospital stay, and other clinical parameters to analyze the effectiveness of interventions.

Chapter 10

CONTRIBUTION TO SOCIETY AND ENVIRONMENT

The implementation of DL model for predicting sepsis infection holds the promise of achieving high accuracy and improving patient outcomes, which can significantly enhance clinical decision-making and contribute to the overall advancement of healthcare. By enabling early detection and intervention, such a model can lead to shorter hospital stays and less intensive treatments, resulting in reduced healthcare costs. Avoiding the requirement of intensive care can translate into substantial economic savings for both individuals and healthcare systems. The societal impact of this technology spans beyond individual patient care, extending to broader public health initiatives and research advancements in the domain of sepsis management. Ultimately, sepsis prediction applications have the capability to save lives, alleviate the financial trouble on healthcare systems, and deepen our understanding and approach to managing this critical condition. The far-reaching implications of this technology make significant contributions to society's overall health and well-being, underscoring its importance in the pursuit of improved healthcare outcomes and quality of life for all.

APPENDIX 1





The Report is Generated by DrillBit Plagiarism Detection Software

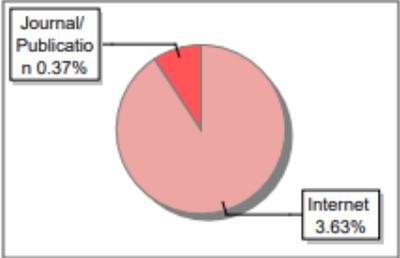
Submission Information

Author Name	Sagar Naidu
Title	Sepsis Prediction
Paper/Submission ID	1687641
Submitted by	supriyasuresh@kssem.edu.in
Submission Date	2024-04-23 18:03:19
Total Pages	36
Document type	Project Work

Result Information

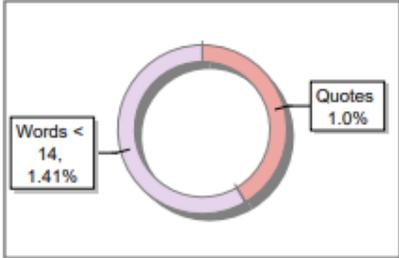
Similarity **4 %**





Journal/
Publication
0.37%

Internet
3.63%



Words <
14,
1.41%

Quotes
1.0%

Exclude Information

Quotes	Excluded
References/Bibliography	Excluded
Sources: Less than 14 Words %	Not Excluded
Excluded Source	0 %
Excluded Phrases	Not Excluded

Database Selection

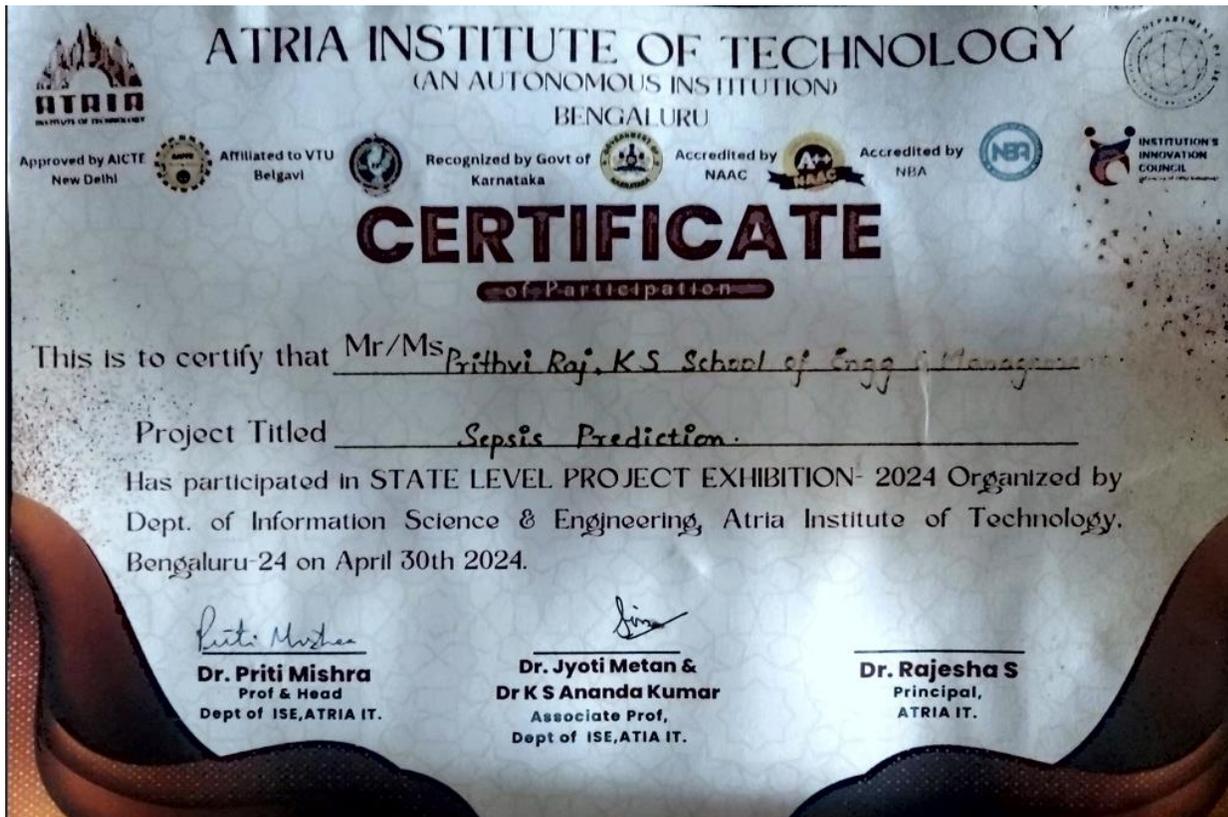
Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File



APPENDIX 2







ATRIA INSTITUTE OF TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)
BENGALURU

Approved by AICTE New Delhi | Affiliated to VTU Belgavi | Recognized by Govt of Karnataka | Accredited by NAAC A++ | Accredited by NBA | INSTITUTION'S INNOVATION COUNCIL

CERTIFICATE

- of Participation -

This is to certify that Mr/Ms *Sagar Anidu, K S School of Engg & Management*

Project Titled *Spsis Prediction.*

Has participated in STATE LEVEL PROJECT EXHIBITION- 2024 Organized by Dept. of Information Science & Engineering, Atria Institute of Technology, Bengaluru-24 on April 30th 2024.

P. Mishra
Dr. Priti Mishra
Prof & Head
Dept of ISE, ATRIA IT.

J. Metan & K. S. Ananda Kumar
Dr. Jyoti Metan & Dr K S Ananda Kumar
Associate Prof,
Dept of ISE, ATIA IT.

R. S.
Dr. Rajesha S
Principal,
ATRIA IT.