EFFECTIVE ARCHITECTURE FOR CLINICAL DECISION SUPPORT SYSTEM

Kalutharage Chamlini Vidyarthi Dayathilake

199315B

Master of Science in Computer Science

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

July 2022

EFFECTIVE ARCHITECTURE FOR CLINICAL DECISION SUPPORT SYSTEM

Kalutharage Chamlini Vidyarthi Dayathilake

199315B

Thesis/Dissertation submitted in partial fulfillment of the requirements for the degree Master of Science in Computer Science.

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

July 2022

DECLARATION

I declare that this is my own work and this thesis/dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

…… …….

K.C.V. Dayathilake Date

The above candidate has carried out research for the Masters thesis/ dissertation under my supervision.

…… …….

Prof. Indika Perera Date

ABSTRACT

The Healthcare domain is a very sensitive domain where the direct stakeholders are patients in the public, which makes a healthcare system directly dealing with patients' lives. There could be heard of several cases in a year where it leads to critical damage or even loss of life, because of misdiagnosis or delays in the diagnosis process or the delay or ignorance of new treatment methods. A clinical decision support system facilitating support for diagnosis and therapeutic decisions could greatly help healthcare professionals when identifying diseases by going through patients' biometrics, life cycle, and symptoms, and when deciding on necessary clinical tests to arrive at confirmation of the diagnosis and decide on treatment methods. The main focus of this study is mapping the real-world diagnosis process to a digitalized system. Senior clinicians use their own experience to derive medical diagnoses accurately. This study proposes an architecture for an evidence-based clinical decision support system, where the system infers knowledge from past knowledge, using machine learning algorithms, and use for future predictions, which could infer and use the medical incidents of the past for future diagnosis, just like experienced doctors. In a practical scenario, diagnosis of disease happens step by step, going through several stages starting from an initial level and digging deeper. To incorporate this behavior, a layered knowledge modeling system is proposed with an ensemble classifier of Random Forest classifier, Support Vector Machine, and Naïve Bayes classifier, and organized into a tree structure based on disease classification hierarchy. Additionally, the proposed system provides feedback and suggestions for clinical tests using feature selection, and a rationale for the diagnosis derived by incorporating explainable machine learning concepts.

Keywords: Clinical decision support system architecture, CDSS architecture, Machine learning, Layered architecture for CDSS, Digital medicine, Disease classification

ACKNOWLEDGEMENTS

My sincere gratitude is extended to Prof. Indika Perera, my supervisor, Head of the Department of Computer Science and Engineering, Faculty of Engineering, University of Moratuwa, for the valuable guidance and support throughout this journey.

I would like to thank my family members and friends for the motivation and continuous support provided throughout this period of my life.

My appreciation goes to all my batch mates in the MSc batch and colleagues at my workplace for the assistance extended in managing work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

The Healthcare domain is a very sensitive domain where the direct stakeholders are patients in the public, which makes a healthcare system directly dealing with patients' lives. There could be heard of several cases in a year where it leads to critical damage or even loss of life, because of misdiagnosis or delays in the diagnosis process or the delay or ignorance of new treatment methods. To avoid these unfortunate incidents and support healthcare providers (Ex: Nurses, Doctors, etc.) in the diagnosis and treatment process, new knowledge and technologies are constantly being added to the healthcare field at different levels of clinical interests. That is where a clinical decision support system comes into the picture, to aid healthcare providers to perform their tasks with good quality, effectiveness, and efficiency.

A clinical decision support system facilitating support for diagnosis and therapeutic decisions could greatly help healthcare professionals when identifying diseases by going through patients' biometrics, life cycle, and symptoms, and when deciding on necessary clinical tests to arrive at confirmation of the diagnosis and decide on treatment methods. While such a decision support system could support this process of diagnosis, the accuracy of the results/ information provided by the system should be at a high level, since they deal with patients' lives. Clinical Decision Support Systems provide healthcare professionals with wisdom that can contribute to the betterment of their patients by integrating mainstream scientific scholarship and up-to-date health information and assessing various items of one's data, such as clinical test reports, medical history, medications, demographics, diagnosis findings, to provide case-specific advice. [6]

Clinical decision support systems have been defined in various statements, varying from very narrow and precise definitions to broad, vague definitions. This study uses a widely accepted definition of clinical decision support: “Providing clinicians, patients or individuals with knowledge and person-specific or population information, intelligently filtered or presented at appropriate times, to foster better health processes, better individual patient care, and better population health.” [3]

Individual clinical competence and experience are used by good doctors with external evidence in combination during the diagnosis and treatment process, neither alone is enough to serve the purpose properly. Without clinical expertise, the availability of excellent evidence does not sufficient to arrive at a good decision. Likewise, even if in the possession of excellent clinical knowledge by standards, without the support of evidence, it only helps a little to boost the doctor’s confidence when arriving at decisions.

The evidence in this sense refers to two main categories: research-based evidence and experience. The reason for the frequent success of an experienced doctor is that he uses his own experience as evidence when handling cases in diagnosis and treatments. Experience that he has gathered as a clinician in his whole working time, from his cases and cases he has heard, witnessed, or read about, surely plays a major role in the diagnosis process when handling cases in the present. Through this knowledge, he shines among the other clinicians. Junior clinicians are eager to gather this knowledge so that they can perform their duty with good quality and efficiency.

This is the point where this research is trying to address, map the real-world diagnosis into a digital system, analyze un-personified medical records on their basic patient information (age and gender), identify symptoms and vital parameters, and use that knowledge to provide constructive suggestions for decision making in diagnosis and treatments.

1.2 Problem

While a clinical decision support system could aid healthcare professionals in their diagnosis and therapeutic decisions, as the healthcare domain is a very sensitive and non-fault tolerant domain, it is very important to produce accurate information within a considerable good response time. As mentioned above, excellent doctors use their own clinical experience as well as the most up-to-date external research in combination during the diagnosis and treatment process, neither alone is enough to serve the purpose properly.
The evidence in this sense refers to two main categories: research-based evidence and experience. There has been research conducted on rule-based clinical decision support systems, only a few were focused on extracting knowledge from past clinical data.

In most clinical decision support systems, each decision support service is developed on its own method, and knowledge is incorporated into a specific application. Because this of tight coupling, it is hard to maintain and has low adaptability and low extendibility. So that it is in high demand for an effective architecture for a CDSS that can be integrated as a plugged-in service with EHR (Electronic Health Record) systems and use the existing data in EHR to infer useful knowledge.

Fact that there are only a few clinical decisio support systems that try to derive knowledge models from past clinical records, a majority of them are focusing on a single specific disease to support. Most of them are focusing on alert generation in critical situations, not on supporting the day-to-day diagnosis process. They are expected to feed all kinds of information to the decision support system, like biometric data, visible symptoms, lifestyle information, test results, and so on, all at once. But that is not how the diagnosis happens in real-world scenarios. First clinicians try to process the visible signs and easily accruable biometric information and come up with a broad diagnosis. And then perform some clinical tests and other medical operations to collect more concrete information, based on the initial diagnosis, to identify the situation more precisely, and come up with a more accurate diagnosis. This process may go on in two, three rounds depending on the situation. This real-world way of

proceeding in diagnosing has not been taken into consideration in any related research or existing systems.

There are a few challenges or characteristics that an effective CDSS should have, could be identified in the literature. CDSS should be able to be integrated with an EHR system as a plugged-in service, which enables the CDSS not to depend on or be tightly coupled with a specific EHR system. CDSS should be evidence adaptive where CDSS should be able to renew its knowledge with the latest available knowledge. CDSS must have a high accuracy rate in automatic decision deriving and provide information to avoid/ reduce distrust towards the system and should be extendable. Since the CDSS should be able to be integrated as a plugin, with a selected EHR system, CDSS interfaces should be standardized. The development of CDSS should consider all these factors to cater to the service effectively.

Therefore, this research is trying to address the said concern above, by proposing an effective architectural design for an evidence-based clinical decision support system, facilitating the usage of knowledge from experience/ case studies, mapping the real-world way of working into a digitalized system, which would actively support the clinical decision-making process in diagnosis and treatment.

1.3 Objectives

Review the literature that is published on architectures proposed and implemented for clinical decision support systems.
Review literature related to knowledge engineering in clinical decision support systems.
Propose an effective architecture for a clinical decision support system.
Design and implement a clinical decision support system, based on the proposed architecture.

1.4 Scope

At the completion of the research, a clinical decision support system architecture would be proposed and implemented as a demonstration project, which would better service clinicians in the diagnosis process by addressing the quality attributes of a good CDSS. The proposed system provides an interface to be connected with external applications which can be an EHR system or a stand-alone application. Users can input the information on symptoms and other necessary information that supports the diagnosis, and the system would derive a diagnosis based on the inputs.

The following section contains a summary of the proposed solution and how it is addressing the quality attributes.

1.5 Proposed solution

This section describes the overview of the proposed architecture for CDSS, the challenges of implementing a CDSS, and how they are addressed in the proposed solution enabling good quality characteristics of the architecture.

This study proposes an effective architecture for an evidence-based clinical decision support system (CDSS). Practical evidence is to be used as the base of knowledge engineering of the CDSS. Analyze the un-personified medical records; on their basic patient information (age and gender), identified symptoms and vital parameters, and use that knowledge to provide constructive suggestions for decision making in diagnosis and treatments.

The clinical decision support system is to be developed as a non-knowledge-based system that employs machine learning algorithms to identify patterns in past medical records, to provide possible diagnostic decisions. The machine learning model is to be trained on decision-making, using an un-personified medical dataset. The clinical

dataset includes the hospital records (un-personified patient information and diagnosis support information).

The focus is on mapping real-world processes into a digital system. The diagnosis based on clinicians' experience is mapped by training a machine learning model based on past clinical data. The practical diagnosis that takes place step by step is matched by deriving diagnosis via a layered knowledge model.

The proposed clinical decision support system consists of two major components: the CDSS interface and the Knowledge engine. The knowledge engine is the most important component of the system, where the decisions/ predictions are generated. The knowledge model is to be implemented in layers, according to the classification of deceases, and organized in a hierarchy. The CDSS service layer exposes the services provided by the CDSS. The other main important component is the CDSS interface which provides the interface to the outside, which exposes the functionalities provided by CDSS.

Most clinical decision support systems are implemented inside a clinical information system (Electronic health record [EHR] system) and tightly coupled with the EHR system. The knowledge is embedded in the system and cannot be used separately.
The study proposes a service architecture that designs clinical decision support systems independent from clinical information systems, and standardized interfaces enable interconnectivity between them. The proposed CDSS will be implemented as a separate service, which can work as a stand-alone system, and as a service that can be plugged into the EHR system. This way, the CDSS can be integrated with any EHR system, as a loosely coupled plugin. This facilitates the interoperability feature of the proposed architecture for CDSS.

Since the CDSS is to be implemented as a separate service and could be used as a loosely coupled plugin; and also, could perform as a stand-alone system as well, usability and maintainability are achieved with this architecture. Even if any change

needs to be done in either CDSS or EHR system, there will be no effect or very less impact on the other system because of this loose coupling.

An effective CDSS should not be merely evidence-based, but an evidence-adaptive system. That means the system should be able to renew the knowledge model with the latest evidence. An iterative training process needs to be in place to achieve this objective. Since the knowledge model is trained using machine learning algorithms, this training process takes some amount of time to update the model with new knowledge. So that the training task could be scheduled to run at a defined time as a background process. This enables adaptability and maintainability in the proposed architecture.

The proposed CDSS, it is providing a separate manual training interface to update the knowledge model. Through this interface, the user can input new features that the knowledge model to be trained on, and to be used for the diagnosis process, and train the knowledge model. This is proposed since the CDSS system should be extendable, which can be trained and used for new deceases as well, and even for upgrading the decision-making process for an already trained disease. Extendibility is achieved through this feature.

Distrust is a major concern in CDSS. Knowing the base for the derived decisions helps in reducing the distrust towards the CDSS. Incorporating an explainable machine learning concept, the system proposes to provide the base that is used for deriving decisions, which should be provided as a rationale, in an easily accessible place where the clinician can refer if needed. This rationale also acts as a summary of the patient and provides the facility to view the patient summary easily, aiding the decision-making process. [32][35]

1.6 Outline

The first chapter gives an introduction to the research, consists of sections on the background, the research problem and motivation, objectives, scope, and a brief of the proposed solution. The second chapter is on the literature review of the research conducted on CDSS architectures, knowledge engineering, and machine learning technics. The third section gives a detailed explanation of the methodology, and the fourth section is on the implementation details. The evaluation chapter includes the details of the evaluations carried out to access the implemented prototype and results. The concluding chapter discusses the contribution of the research, its limitations, and the future work that can be done under this research.

CHAPTER 2

LITERATURE REVIEW

This chapter discusses the related research work conducted and available in the literature regarding clinical decision support systems.

The literature related to different approaches proposed to implement clinical decision support systems and the proposed architecture for them are discussed here.

2. 1 Research carried out under architecture for CDSS (Clinical Decision Support System)

In the literature, there are four types of CDSSs. [2][3]

1. Standalone systems: Independent decision support systems which are not bonded with clinical systems. These systems operate separately from any other system. Any person in need of information has to intentionally find the system out, input the required information, and then understand and interpret the results. Because of this requirement to inputting relevant information, such systems cannot be proactive, which is a limitation. [2][3]

2. Integrated systems: CDSSs are tightly integrated with clinical information systems. Proactive nature is a major advantage and successfully overcome information/ data duplication. [2][3]

3. Standards-based systems: Knowledge representation standards are used to define rules, to enable sharing of information between systems. The complexity of the selected standards can be a limitation. [2][3]

4. Service architectures: CDSS and EHR are independent systems that are communicating via standard interfaces.

In most CDSS, decision support service is developed on their own method, and knowledge is incorporated into a unique application. It is for this reason that current clinical application systems make it difficult to maintain knowledge and add new CDS services. Research [1] focuses on this concern to separate clinical knowledge from the clinical application system. It has suggested sharable common architecture for different CDS services, instead of scattered knowledge, a centralized knowledge repository and a core component for knowledge management are used. The proposed architecture consists of 5 core components; knowledge authoring environment, knowledge repository, knowledge engine, interface server, and interface repository; each component plays an important role in knowledge engineering. [1]

In [2], a new architecture for CDSS called SANDS (Service-oriented Architecture for NHIN Decision Support) is introduced. When designing the architecture, they considered three main points: where the clinical information is stored, where the CDSS would be used, and how to incorporate both ends to meet the goal. This architecture is based upon the learnings from two existing service architectures SAGE and SEBASTIAN. As a solution to some of the problems faced in those systems, standard interfaces for both the clinical system and decision support system are designed, and the CDSS system work as a middle layer to provide decision-making support. In simple, the authors present SANDS as a distributed clinical decision support system [2].

[11], [32], [33], [34], [35], [36] discusses about challenges and the characteristics an effective clinical decision support system must possess. Interoperability between CDSS and various EHR systems, avoiding alert fatigue, adaptability to new knowledge, specificity, and simplicity of alerts, formalization was the identified quality attributes among them. [11] indicates unrelated factors of a decision support system: proactively provide suggestions and warnings into the coupled EHR system to support the decision-making process; information provided digitally; and provide precise recommendations, not just accessed information. As cited by the author in [11], future CDSS should be “evidence adaptive”, meaning the knowledge bases should be

continuously updated to incorporate the latest data and information. This will most likely be one of the criteria used to evaluate CDSS in the future.

These research works are the groundwork that is laid to support the research area on architecture for a clinical decision support system. [35] has identified, by using an iterative mechanism, three main categories of challenges in designing CDSS: "improve the effectiveness of CDS interventions, create new CDS interventions and disseminate existing CDS knowledge and interventions" [35]. The authors believe that if these challenges are addressed, it will lead to implementations that will allow the full potential of CDSSs to be realized as well as improve the overall quality and efficiency of the CDSS.

2.2 Research carried out under knowledge engineering

The objective of Knowledge Engineering is comparable to Software Engineering in that it transforms the art of building Knowledge-based Systems (KBS) into an engineering discipline. This necessitates a thorough examination of the construction and maintenance process, as well as the creation of specialized processes, languages, and tools for developing KBSs [55].

There have been several research works conducted on the knowledge engineering process of a clinical decision support system.

New knowledge emerges at various levels of clinical interest, and the healthcare sector continues to grow. Simultaneously, there is a growing interest in using clinical decision support systems (CDSSs) to improve the quality and efficiency of healthcare. The majority of present CDSSs are not designed to automatically adapt scientific research in a well-established manner. During patient care, clinicians and researchers frequently consult internet resources for unanswered questions. They frequently use a disjointed strategy to find the knowledge they need from the resources of their choice. Furthermore, no established process for integrating important knowledge for future

use exists. [4] proposes a system named "KnowledgeButton" which is an evidence adaptive tool for CDSS and clinical research. This research proposes a well-defined and well-established approach for adapting evidence from online reputable knowledge sources. It saves doctors time that would otherwise be spent searching for research evidence via a disjointed and laborious approach.

A rule-based clinical decision support system for hematology disorder is proposed by Y. Y. Chen, K. N. Goh, and K. Chong, which is an evidence-based practice clinical decision support system. [5] For each sort of produced prediction, the system attempted to give an external evidence list (electronic medical journals). Evidence-Based Medicine (EBM) has grown in popularity as a method for making medical decisions. It's a way of aiding in clinical decision-making by answering clinical queries with the most relevant research information. This technology uses the results of a CBC test to indicate probable diseases and provides online medical material about the management of these ailments. However, EBM is difficult to implement due to a poorly maintained repository and the vast amount of data accessible.

[6] form a basis for future research on clinical decision support systems based on knowledge graphs. It suggests that knowledge graphs be merged into a Clinical Decision Support System to ease the construction of an auxiliary diagnosis schema. As noted by the authors, the important components in developing an auxiliary diagnosis system are standardized representation and structural integration, and reasoning on a knowledge graph unlocks the application's potential. The use of a knowledge graph can aid in the acquisition of useful medical information and the improvement of diagnostic accuracy without the time-consuming process of manually designing and creating domain intellectual. Computer-based systems with this level of awareness are directly related to a finer understanding and management of illness for each patient, and they play a crucial part in achieving precision medicine.

[7] proposes a method for creating evidence-based decision support systems for medical assessment that is built on a routine diagnostic schema. In the visual knowledge definition tool, clinical guidelines are represented using a flowchart. The

flowchart can be transformed into structured knowledge and saved in a knowledge base using mapping rules. A knowledge translation engine parses knowledge and provides the information needed for web-based assisted diagnosis. The knowledge repository is adaptable to varied scenarios and easy to maintain thanks to the mutual conversion of flowcharts and hierarchical information based on mapping principles. [7]

[37] conducted research for a clinical decision support system based on a feature ranking approach. Ranker algorithms are employed to select the most suitable features, and the selected features are fed into the training of a suitable classifier (the Random Forest classifier in this case). As cited by the authors, multiple algorithms have been used in the past to classify medical data, including Pareto-differential evaluation, statistical neural network-based systems, Association Rules (AR) based expert system, hybrid feature selection methods, clustering, fuzzy decision trees, etc. The feature ranking algorithms, as explained by the authors, will evaluate each feature in the dataset and assigns a rank to that feature.

[38] has researched how to use machine learning and other data science-related algorithms that can be used for deriving prognoses based on EHR data. AI approaches enable computers to do a wide range of complicated tasks with surprising precision, thanks to advances in processing power, storage, memory, and the creation of massive amounts of data. On the one hand, informatics is critical for precision medicine because it organizes massive data, generates learning systems, and allows for individual participation. One of the goals of health information technology is to turn electronic health records (EHR) into knowledge that may be used to create a CDSS. Machine learning models can help handle this massive quantity of data by predicting health outcomes and deciphering trends that clinicians may miss. Machine language approaches have been widely utilized to extract information from vast amounts of data, and have been beneficial in improving diagnosis, outcome prediction, and chronic illness management. Learning healthcare systems, which describe ecosystems that

connect science, informatics, economic incentives, and lifestyle instruction for continual improvement and innovation, will rely heavily on machine learning.

One of the most important stages in lowering inpatient morbidity and death is early detection of clinical deterioration. Prediction based on Electronic Health Records (EHRs) is becoming increasingly practical, thanks to the fast-rising area of Artificial Intelligence. EHRs may be analyzed and used as input to machine learning models for constructing Clinical Decision Support Systems (CDSS) to improve hospital workflow since they aggregate information from all patient timelines. [39] has proposed an early warning system for the hospital wards, connecting the CDSS to the EHR stream to generate alerts in critical situations by analyzing patients' history medical records. It has trained machine learning algorithms to derive the alerts incorporating explainable AI.

The black-box aspect of machine learning systems is not a problem in some applications, such as Computer Vision or Sound Processing. That is, if an algorithm works effectively, it is often unimportant why or how it operates. In the field of medicine, inexplicable predictions involving human life are not acceptable, this poses a significant barrier to machine adoption both legally and ethically. Black box classification is not much acceptable for the healthcare domain, since the care providers do not accept a blinded diagnosis that would be produced by some system. An explainable machine learning concept is developed to fill this gap. A basis that has been used for deriving the diagnosis is provided to the user along with the resulting diagnosis decision. [39], [41], [42] [43], [50], and [51] have been researched on how this concept can be integrated with a symptom-disease classifier.

The suggested approach by [40] uses the patient's symptoms to forecast the likelihood of illnesses. It suggests disease-related clinical tests and medication. The data-driven clinical decision support system was proposed using the logistic regression method to forecast probable diseases based on the patient's symptoms as input from care providers.

[45] has proposed a three-layered or three-dimension architecture to derive predictions instead of the general two-layer architecture of disease-symptom combinations. As the three layers, disease-symptom-property/value combinations are used, and a Naïve Bayes classifier is for deriving predictions by using iterative calculations. This will allow the users to easily distinguish between diseases that have similar symptoms. The three-layer model knowledge base can successfully overcome the knowledge interpretation inaccuracy by utilizing more helpful information in inference.

[46] has proposed a system of evidence-based medicine for a CDSS. Experience is combined with the finest external clinical data from systematic research in the practice of evidence-based medicine. This study has used the results of a CBC test to make predictions on possible diseases and then links to medical literature about how to treat these conditions.

[47] presented a rule-based classification system employing machine learning algorithms to aid the clinical diagnosing process for Liver diseases. The objective of the research was to suggest a classification model that can be used to predict liver diseases by implementing data mining techniques. The authors note that with the improvements in medicine and the use of automated tools to anticipate illnesses, a huge number of machine learning algorithms have been included in medical diagnostic equipment such as CT scan machines and ECG machines. These algorithms are used in clinical research to diagnose many sorts of disorders. These approaches are either utilized for illness categorization or prediction, allowing for a more precise assessment of the patient's therapy. The authors also present that data mining techniques like decision tree, Support Vector Machine (SVM), Artificial Neural Network (ANN), rule induction, Naïve Bayes, etc. can be used to classify data, predict the outcome (i.e., the number of patients who suffer from a particular illness) and obtain the desired results. Diagnosing support system for Liver diseases has also been presented by [48], using feature selection classification techniques.

2.3 Machine learning approaches

Naïve Bayes

Naïve Bayes is a classifier based on probabilistic approach. This is a very commonly used algorithm because of its computational simplicity. [24] The underlying idea of this algorithm is computing the probability of a document belonging to a particular class. Modified Naïve Bayes provides ways to improve performance through identifying correlations among attributes. [22] Naïve Bayes is built on the Bayes theorem assuming that each term is independent of the other. Here, every term/ feature in the feature vector contributes to deciding the class for a given document. There are two models are using for Naïve Bayes; Multivariate Bernoulli model and Multinomial model. Multinomial with Naïve Bayes is said to be more effective for classification and when using this algorithm, only a small training dataset is needed. [26]

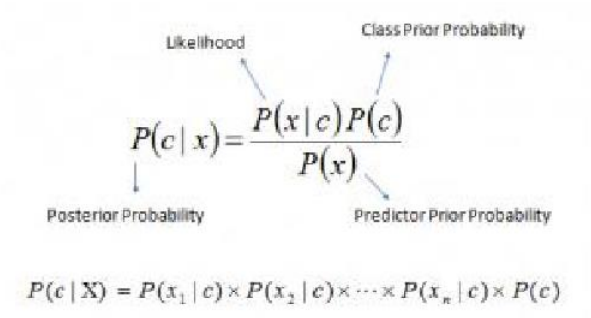


Figure 2.3.1: Equation of posterior probability in Naïve Bayes

Above,

- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.
- P(x) is the prior probability of predictor.

Support Vector Machines (SVM)

SVM performs better when the input vector has high dimensionality. The underlying theory of SVM is identifying the maximum-margin hyperplane between two classes. It has a great generalization capability in the training process a and speedy training process. It stands on the lowest structural risk principle. The vectors with the nearest proximity to the decision hyperplane are called support vectors.

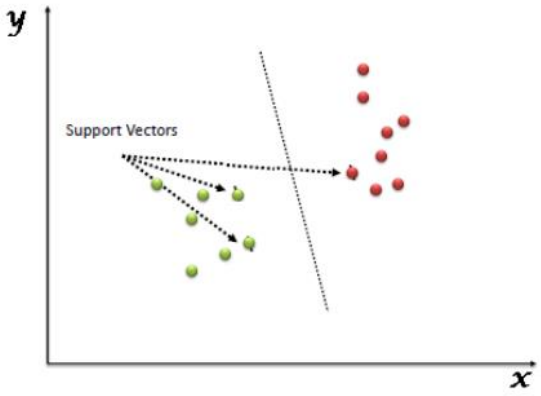


Figure 2.3.2: Support Vector Machine

There are so many researches that have been conducted to increase the performance in classification using SVM. [23] The effectiveness of SVM relies on the kernel parameters and penalty coefficient parameters. Developed a probability-based approach for automatic tuning of these parameters using the Markov Chain Monte Carlo (MCMC) method. [27] SVM performs better for classifying into two classes. Class incremental SVM classification method has been proposed in [28].

Support Vector Machine (SVM) is proven in the literature that the maximum accuracy can be achieved through this technique and it can effectively handle input vectors of high dimensionality. SVM attempts to find the maximum-margin hyperplane between two classes where the classification decision can be drawn.

When the dataset is non-linearly separable, the input dataset will be mapped to a feature space with high dimensionality, which is also called Hilbert space. This will be done using a kernel function which gives the multiplication of two non-linear mapping functions.

To obtain the optimal solution, maximum width gives by; ½ ||w||2 following the constraint of y(w.x + b) -1 >= 0.

Where L is the width of the hyperplane and α is the Lagrange multiplier.

Decision Tree

A decision tree is an easily understandable method even for non-experts. This classifies a given corpus from the root until it arrives at a leaf corresponding to a destination of the classification which is class or a category. [29] In a tree-structured classifier, internal nodes reflect dataset properties, branches reflect decision rules, and each leaf node reflects the conclusion. A decision tree is a supervised learning technique. The forecasts are based on the dataset's features.

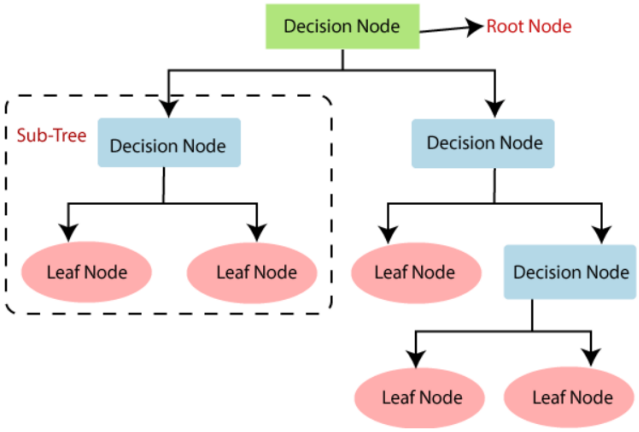


Figure 2.3.3: Decision Tree

Random Forest Classifier

A random forest classifier is made up of many decision trees that are applied to different subsets of a dataset and provide a collective result as an ensemble classifier. This is a much more complex version of the decision tree. Rather than depending on a single decision tree, the random forest collects recommendations from each tree and determines the final output based on the popular election of predictions. Random forests are a set of tree classifiers in which the values of a random vector collected individually and with the same dispersion for all trees in the forest are used to determine the behavior of each tree. [36] It employs bagging and randomization in the construction per tree to create a statistically independent forest of trees that the committee's prediction is more precise than any individual tree's.

Neural Network

Some research has been done using for classification using methods of a neural network like single layer perception, multilayer perception, back-propagation, etc. In these approaches, a term will be assigned a weight calculated and the nodes‟ activation will be propagated towards the output layer, then at the end of the iteration output layer will make the final decision of classification. [22]

CHAPTER 3

METHODOLOGY

This study proposes an effective architecture for an evidence-based clinical decision support system (CDSS), which would actively support the clinical decision-making process in diagnosis. Practical evidence is to be used as the base of knowledge engineering of the CDSS. Analyze the un-personified medical records; on their basic patient information (age and gender), identified symptoms and vital parameters, and use that knowledge to provide constructive suggestions for decision making in diagnosis and treatments.

The focus of this research is on mapping real-world knowledge and process into a digital system. This consists of two main ideas.

Senior clinicians use their experience in the diagnosis process. This knowledge based on experience is modeled in the proposed system by training machine learning algorithms on the past clinical records, to identify the patterns and use that knowledge to derive decisions by classification.

The other one is, this research proposes a layered knowledge engine which is an attempt to map the practical diagnosis process that happens step by step. At the initial level input, the primary data on biometrics, lifestyle information, visible symptoms, and other easily obtainable related information, and based on them come up with an initial diagnosis, which is what happens when a patient meets a doctor. After that step, doctors dig deeper to derive a precise diagnosis by performing some clinical tests or required operations. This process is incorporated into the proposed system via the layered knowledge engine.

3.1 Dataset

This research has proposed a layered knowledge engine employing machine learning technologies to aid clinical the diagnosis process. The implementation of the prototype is scoped into two layers, based on heart diseases.
The first layer, the root layer is containing several diseases as prognosis. The root layer is trained on a dataset that has 132 parameters on which 42 different types of diseases can be predicted. There are nearly 5000 data instances in the dataset.
The second layer is dedicated to heart disease classification.
Cleveland UCI dataset for heart disease has been 13 features have been collected that are relevant to making predictions on heart diseases. Nearly 200 data rows are present.

3.2 Knowledge Engine

The CDSS will be developed as a service architecture which is a system, clinical decision support system and medical information system are independent systems and standard interfaces enables connection between the two systems.

The proposed clinical decision support system consists of two major components: the CDSS interface and the Knowledge engine.

The knowledge engine is the most important component of the system, where the decisions/ predictions are generated. The clinical decision support system is to be developed as a non-knowledge-based system that employs machine learning algorithms to identify patterns in past medical records, provide possible diagnostic decisions, and suggest necessary tests to be conducted to make the confirmation of the diagnosis. The machine learning model is to be trained on decision-making, using a clinical dataset. The clinical dataset includes the medical records (patient information (un-personified), diagnosis, prescribed clinical tests).

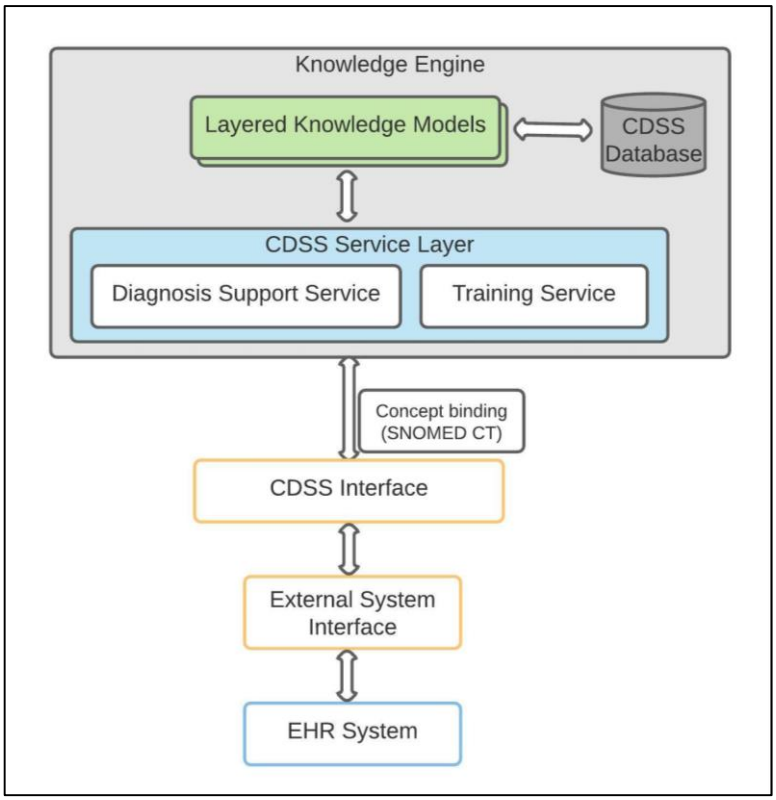


Figure 3.2.1: Service architecture of the CDSS

3.2.1 Layered Knowledge Models

All the organs of the human body are prone to diseases. Thus, there is a huge number of diseases in the world, which can be organized into a big hierarchical tree of diseases. Each of those diseases shows various kinds of symptoms in the human body. And there may need various kinds of vital parameters and further tests to identify the diseases correctly. Collectively that is a huge number of diseases and diagnosis support parameters. A CDSS is a system that is supposed to provide diagnostic support to the clinicians by analyzing the entered or extracted diagnostic information: vital parameters, demographic information, body symptoms, etc. But because of this very diverse and distributed hierarchy of human diseases, it is not very effective to train a single model to derive decisions/ predictions on all those diseases.

To address this challenge, this research proposes a layered knowledge engine. One knowledge model is to be trained for one branch of the disease classification tree. And likewise, train separate knowledge models for each branch of the disease classification tree. And these trained knowledge models are organized according to the order of the disease classification hierarchy.

This layered architecture is proposed for training disease-specific knowledge models which would be more effective in the decision/prediction-making process. It also enables the extendibility of the CDSS, since the layered knowledge model can be expanded as users want to by training desired knowledge models and integrating them with the classification hierarchy.
And as explained earlier, this layered model represents the step-by-step diagnosis process that happens when a patient meets a doctor in a practical scenario.

Since the proposed CDSS should be able to work with any EHR system as a plugin, without having to know about the internal workings or interpretations of the EHR system, the knowledge of the classification of diseases better is standardized.
This research proposes to use the international standard for human diseases classification, presented by the World Health Organization: International Statistical Classification of Diseases and Related Health Problems (ICD-11).
Especially, this study proposes to use ICD-11 for Mortality and Morbidity Statistics (ICD-11 MMS).

ICD-11 is the 11^{th} revision of the international classification of diseases which would be updated annually by WHO. It has a large classification taxonomy with a large number of entities, around the number 85000, which are called nodes or classes. ICD-11 MMS is generally called the ICD-11, where MMS represents Mortality and Morbidity Statistics. It has 3 main elements: chapters, blocks, and categories. A chapter is a top-level entity of a hierarchy. Several related categories are grouped into a block. A category is an end node, which would represent a disease, indicating an end

node of a diagnosis path. A category has a unique code called ICD code, and chapters or blocks do not have ICD codes.

For example, as the scope of this research, when considering heart-related diseases, according to ICD-11, it falls under “Diseases of the circulatory system”. It is the chapter on heart diseases. It has several sub-categories or else blocks: Hypertensive diseases, Heart valve diseases, Diseases of the coronary artery, Pericarditis, Heart failure, Diseases of veins, etc.
Layer 1 knowledge model is to be deciding if the patient has a disease in the circulatory system. Layer 2 knowledge model is to be deciding what is the specific disease category
And the decision-making process continues until it meets a result representing a certain diagnosis: for example, “Mitral valve rupture”.

Diseases of the circulatory system > Heart valve diseases > Mitral valve disease > **BB65** Mitral valve rupture

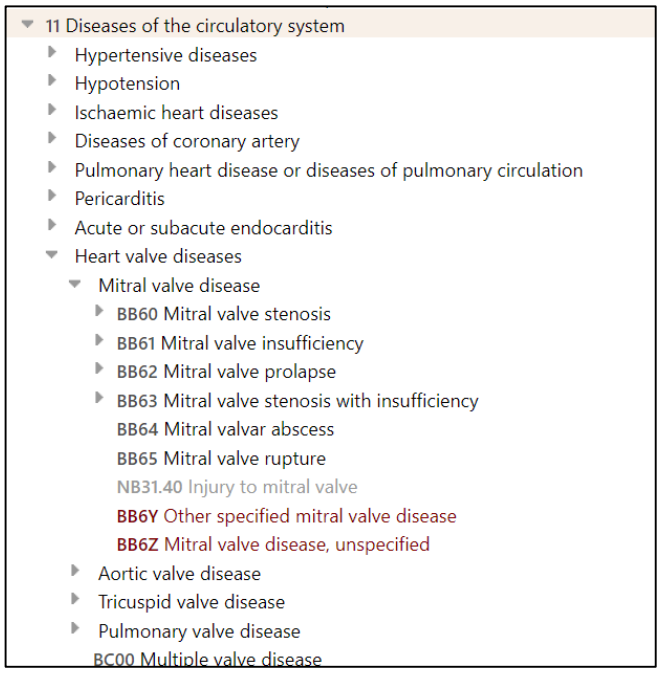


Figure 3.2.1.1: Disease hierarchy of circulatory system according to ICD-11 (not the full hierarchy)

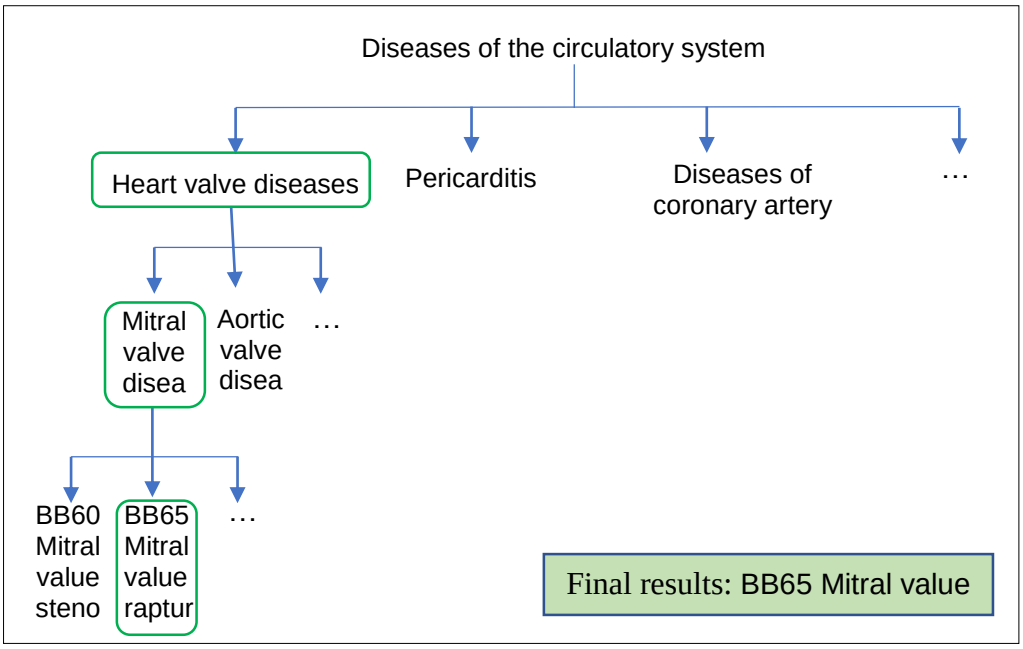


Figure 3.2.1.2: Graphical representation of the disease hierarchy and traversing through the tree structure

The knowledge models are to be organized into layers following a tree structure/ graph structure. When deriving the prediction, the process starts at layer 1 and goes down the layers until a leaf node is met: there the final prediction result is produced. Traversing through the layered knowledge model tree is designed to simulate the human decision-making process. According to the result from layer 1, the next knowledge model to be executed is decided.

When traversing through the layered knowledge model tree, if the current layer is not a leaf node and if the data for the features in the next layer are missing, the CDSS system informs the user that he needs to input these data as well, and terminate the traversing at the point, and produce the intermediate result at the point. This way even if the clinician has missed some tests or clinical measurements to be taken, he will be reminded of them by the system.

And as an additional feature user can select a diagnostic area for the system if the user is sure about the area of disease, then the system will start the classification from there. This feature is facilitated as when the system has grown to a broad disease tree, it will take much processing time that may be unnecessary.

3.2.2 Train knowledge model by machine learning algorithms

The knowledge models are to be trained by machine learning algorithms, using the clinical dataset.

A supervised machine learning process is used for disease classification based on the symptoms. A labeled clinical dataset is provided for the training process.

A clinical dataset that has un-personified clinical data including patient's biometric information, relevant and significant lifestyle and habits, clinical test results, is lined up along with the diagnosed disease. Characteristics of the used datasets are stated in the above section.

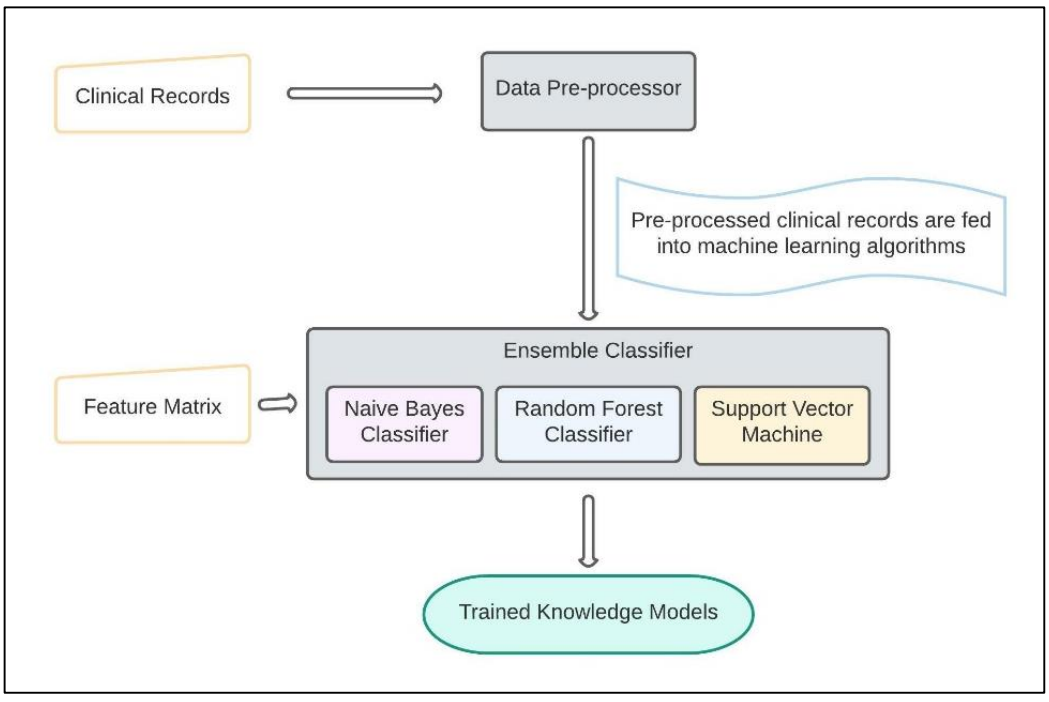


Figure 3.2.2.1: High-level architecture of the knowledge model training

The first step is data pre-processing. Cleaning of the dataset is done here. Empty rows, null or empty values are dealt with and cleaned the dataset preparing for feeding to machine learning algorithms.

Feature matrix should be provided for the training process. When training, the system provides functionality to decide whether to perform feature selection or not and the feature selection technique to be used. If a feature selection technique is to be performed, selected features would be used as the feature matrix; if not the originally provided feature matrix would be used as it is. This study uses correlation as the feature selection technique.

Pre-processed data will be fed into the machine learning algorithm, together with the feature matrix, to train the model. This research is using an ensemble classifier as the machine learning algorithm. This approach is taken to improve the accuracy of the CDSS. The trained knowledge model is the output. This trained model can be used for analyzing new scenarios and deciding on diagnosis and treatment methods.

Random Forest classifier, Naïve Bayes classifier, and Support Vector Machine algorithms were selected for the ensemble classifier. Each classifier is trained with the given dataset separately and stores the trained classifier models to be used for classification tasks later. And evaluation details of each classifier are also saved.

3.2.3 Generate result in the ensemble classifier

As described in the above section, an ensemble classifier is proposed for disease classification.

Ensemble learning is a technique for constructing a new classifier, integrating multiple base classifiers, which would outperform individual base classifiers. Ensemble model boosts machine learning results by collaboratively integrating different models.

This strategy outperforms a singular model in terms of predictive accuracy. The main goal of ensemble approaches is to lower bias and variance.

This research has used the Naïve Bayes classifier, Random Forest classifier, and Support Vector Machines classifier as the base classifiers which are then integrated into the ensemble classifier, which then would be used for making predictions.

Two of the most straightforward ensemble techniques are voting and averaging. They are simple to comprehend and implement. For classification purposes, the voting method is used. Voting has two types based on the technique that is been used to derive the final output.
They are the majority method and weighted voting.
The majority method is also called popularity voting. In this approach, it takes the most voted output as the final result of the ensemble classifier. This is purely based on the arithmetic mode method.
The weighted voting method assigns a weight to individual classifiers. By doing this, this method increases the importance of some classifiers.

Weighted voting is the ensemble technique that has been used in this research.
The proposed solution uses a priority-based mode method to derive the final output of the ensemble classifier.
This mechanism was followed to maximize the accuracy of the ensemble classifier.
Priorities are given to the classifiers using the performance statistics of the individual classifiers.

Each classifier has been evaluated independently for each node and obtained performance statistics. Accuracy performance measurement has been used as the base for this.
Higher the accuracy, the higher the priority that is given to the classifier. The classifier which has the highest accuracy among the trained classifiers is given the highest

priority, likewise follows through the list of classifiers and a priority number is given to all the individual classifiers.

Let's say the results from three classifiers are R-1, R-2, and R-3.

R-1 is the result of the most accurate classifier for the node.
R-2 is the result of the second-most accurate classifier for the node.
R-3 is the result of the third most accurate classifier for the node.

If R-1 and R-2 are the same, the result is R1.
If not, then R1 and R3 are compared. If the same, R1 is the result.
If not R2 and R3 are compared. If the same, R2 is the result.
If not, that means all three results are different from each other. Then R1 is the result.
R1 is the privileged result here.

Privileged Result:

This is the result to be used when all the results are different from each other.

- Normally, when classifiers have different performance measures, the privileged result is the result of the classifier that has the highest accuracy value.
- If the accuracies of all three classifiers are the same, then simply the majority voting technique or the pure mode method is followed to derive the final output of the ensemble classifier.

  If classifiers perform equally and all results are different from each other, privileged result is the result of the Random Forest classifier.

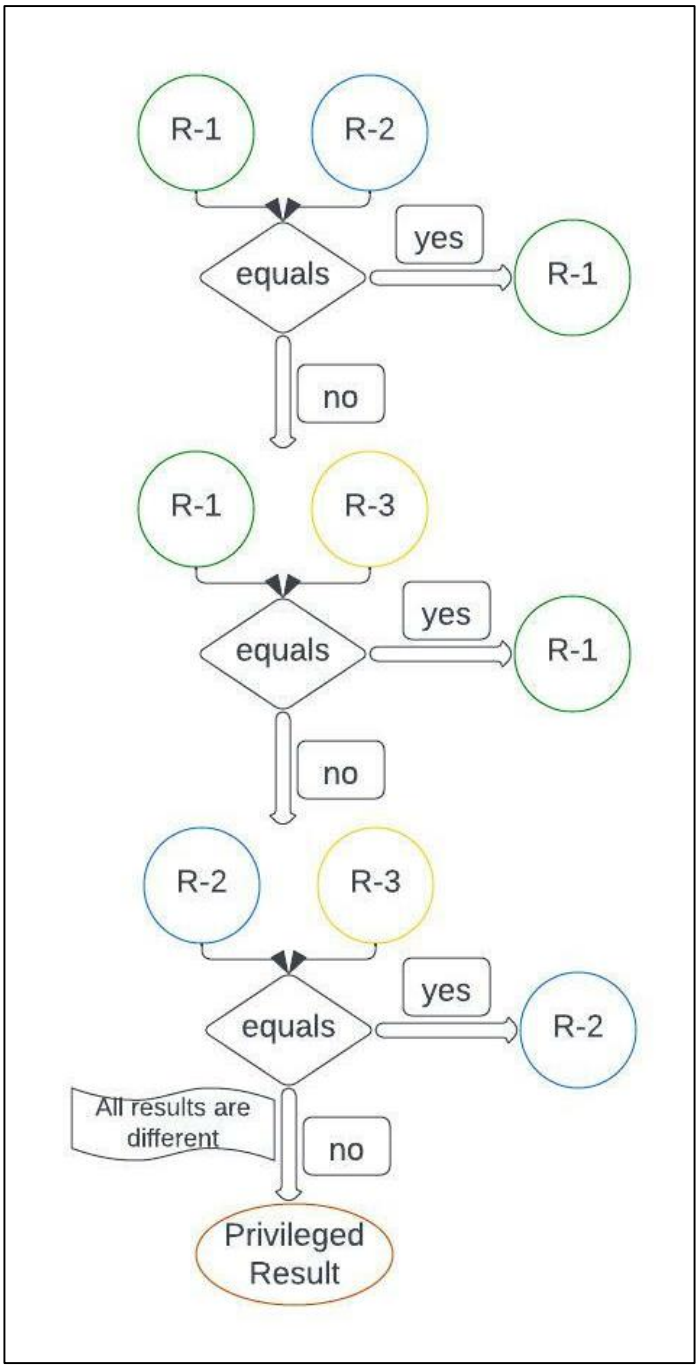


Figure 3.2.3.1: Generate result in the ensemble classifier

3.2.4 Deriving rationale for the decision

Explainable AI is an emerging concept that would enable a very important feature for a clinical decision support system. Just providing the final output of a classifier is not enough for the medical domain, since the users are not blindly accepting a result with no base or explanation.
To reduce the distrust and acceptability of the system, it is important to give users an explanation of how the system is generating the decisions and the basis that has been used for the decision-making process. It generates details or justifications to make its operation plain or simple to comprehend. The advancements in this discipline will eventually lead to a perfect combination of interpretability and performance in machine learning models.

This is another important feature of the proposed system. The rationale is the explanation or the reasoning behind the output result produced by the system. Producing a rationale along with the results improves the trust in the system and more accepting of the decisions derived by clinicians.

The basis for the prediction derived by the CDSS is provided to the user as the rationale, which he can use as a summary or reasoning.
The proposed system has two major parts of the rationale or the explanation or the reasoning behind the final result produced by the system.
They are the propagation array of the layered knowledge tree and the summary of the most important symptom-s that were used as the base for deriving the diagnosis value.

As explained earlier, the diagnosis decision would be derived by traversing through the disease tree based on the user inputs. This traverse/ progressing path would be presented to the user.

For example, if we consider the example that has been used before, that was on heart disease classification, the propagation array would be as follows.

Diseases of the circulatory system >
Heart valve diseases >
Mitral valve disease >
BB65 Mitral valve rupture

The other part is the summary of the patient which consists of the most important symptom-value combinations that have been used as the basis for deriving the diagnosis decisions. This summary is presented to the user as the basis for diagnosis.

For this, feature selection technologies are used to obtain the most important features for each disease. The feature selection technique that the user has chosen for the training process is used to identify the most important features. Correlation has been used as the feature selection technique. The list of the most significant features is obtained for each node by performing feature selection techniques at the training phase of the nodes.
When a user is using the system to classify a clinical instance, the obtained list of the most significant features is provided as the rationale for a list of symptom-value combinations.

As a further improvement for the second part of the rationale which is the list of most important features for each node, importance parameters of the Random Forest algorithm can be used. This built-in feature of the Random Forest algorithm can be utilized for this purpose. Gini importance or mean decrease impurity can be used for this purpose.
The criteria with which to split features on each node of the Random Forest classifier can be chosen by Gini impurity. We can gauge how each feature reduces the split's impurity and measure the average reduction in impurity for each feature. The average of all the trees in the forest serves as a proxy for the significance of a feature.

3.3 CDSS Interfaces

There are two main interfaces in the proposed architecture: the diagnosis support interface and the training interface. These two interfaces would be providing connection points to the CDSS. External application, either an EHR or a stand-alone application can connect and use the functionalities provided by the CDSS.
All the provided functionalities are exposed as REST APIs.

For demonstration and evaluation purposes, a sample stand-alone application is implemented and integrated with the CDSS that was implemented as a prototype.

CDSS interfaces better are standardized using a globally used international standard. SNOMED CT is a good candidate as it is accepted internationally to define medical concepts.

SNOMED CT is the world's most complete, international clinical healthcare terminology. SNOMED is mapped to other international standards as well and is being used by more than eighty countries. Clinical terminology in SNOMED CT is unrivaled in deepness, allowing physicians to record data with greater precision and consistency. The Community of Practice continues to improve SNOMED CT, which is a growing and changing product.

All the inputs which are entered into the CDSS and outputs which are displayed/ presented or published by the CDSS interface must follow this guideline. All the symptoms, diseases & disease categories, clinical tests, treatment methods, or any other medical term should be represented by a standardized concept.

Any of the medical terms that would be used in the system are referred to as keywords. All the keywords must be bound to specific terminology defined in the standard used, in this case, they should be bound to terminology or a concept defined in SNOMED CT. This is called terminology binding or concept mapping.

When this terminology binding is done, a keyword has a specifically defined meaning to it that has been accepted internationally. This facilitates the clearness and clarity of the terms used by the CDSS interfaces and published information by the CDSS. This enables the usage of the CDS system globally without language ambiguities.

CHAPTER 4

IMPLEMENTATION

This chapter is providing a detailed explanation of how the proposed clinical decision support system has been implemented as a prototype.

4.1 Modeling layered knowledge engine

As explained in chapter 3, a layered knowledge engine is constructed representing the disease hierarchy tree/ graph structure.

A disease or a disease category is represented as a node of the tree/ graph structure.

A node consists of several attributes to represent its information and store the hierarchical information of its location in the tree. They are node name, paths to training dataset, classifier model locations, the path of feature matrix file and path of mandatory features file, feature selection technique, the path of the base node directory, parent node, and list of children nodes as hierarchy information.

```java
/**
 * Represents a single node in the disease hierarchy.
 */
public class Node {
    private String name;

    private String pathOfNodeDirectory;

    private Map<String, String> classifierModelLocations;

    private String pathToTrainingDataset;

    private String pathToFeatureMatrix;

    private String pathToMandatoryFeatures;

    private String featureSelectionTechnique;

    private String parentName;

    private List<Node> children = new ArrayList<>();
```

Figure 4.1.1: Code snippet of Node object

The tree structure is stored as a JSON file. The root node is saved into the JSON file which contains the whole node hierarchy.

Figure 4.1.2: Snippet of node hierarchy JSON

Node is the primary building block of the layered knowledge model. Nodes are organized into a disease hierarchical tree to construct the layered knowledge model. Since this is the core element of the architecture of the layered knowledge engine, it possesses the attributes like scalability and maintainability. When the system needs to be extended to support more diseases, new nodes can be trained for the required diseases and map them into the disease hierarchy tree. This way the system can be expanded horizontally as well as vertically. That means the system provides the capability of adding new diseases to increase the depth of the tree and can provide support for new disease categories as well by scaling horizontally.

And for knowledge adaptation and maintaining also has been enabled in the system because of this implementation strategy; which can be done by simply re-training a node.

In the training phase of a node/disease, a directory would be created for the node that is being trained. All the node-related information and files are saved in that directory. Separate classifier models are trained for each node of the tree and stored as files, and would be used when classifying instances for deriving diagnostic decisions.

When classifying single instances for deriving diagnostic decisions, the saved classifier models are loaded to the system for the relevant nodes and do the classification traversing through the disease hierarchical tree.

Further implementation details of how the training process and classification process are done using this node-based tree structure are explained in the rest of this chapter.

4.2 Implementation of Training Process of CDSS

Training dataset, together with feature matrix, node/ disease name, hierarchy information (name of the parent node and the list of names of children nodes), and feature selection related information must be fed into the system as the inputs for training the classifiers of the system.
The primary output is the trained classifier models and evaluation results. And the feature matrix for the node/ disease and mandatory information/ symptoms (selected via feature selection) are stored.

The system provides the facility to train the system for a particular defined node, or bulk training provides a large dataset with a large number of features at once. Two main APIs were defined to facilitate this requirement. One for training a single node, to be used when training a specific node. The other one is for training multiple nodes as a bulk operation, to be used when updating the system with a given set of training data.

If new diseases need to be trained on, then the feature matrices for the new diseases must be provided along with the dataset. Or else, the existing nodes would be trained node by node as explained above in this section. When the API for multiple nodes is invoked, the system creates the training dataset for each node according to the feature matrix that already exists in the system. The training dataset for each node would be generated by filtering out the unwanted features for a node using the feature matrix of the node. This feature needs to be facilitated to train the system using EHR data, where the CDSS has been integrated with an EHR system. To adapt to the new knowledge, the training process can be scheduled to be initiated repeatedly for a period that the user desired. This API can be used when re-training the system for knowledge adoption, mainly for scheduled task service jobs when integrated with an EHR.

```
@PostMapping("/train_classifier_single_node")
public ResponseEntity<TrainingResultDTO> trainClassifierSingleNode(
        @RequestBody TrainingInputDTO inputDTO) {
```

Figure 4.2.1: Rest API for training classifier for a single node

```
@PostMapping("/train_classifier_multiple_nodes")
public ResponseEntity<List<TrainingResultDTO>> trainClassifierMultipleNodes(
        @RequestBody MultipleNodesTrainingInputsDTO inputDTO) {
```

Figure 4.2.2: Rest API for training classifier for multiple nodes

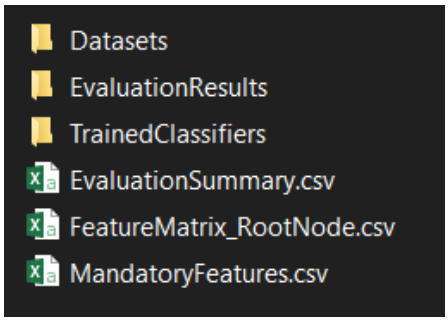
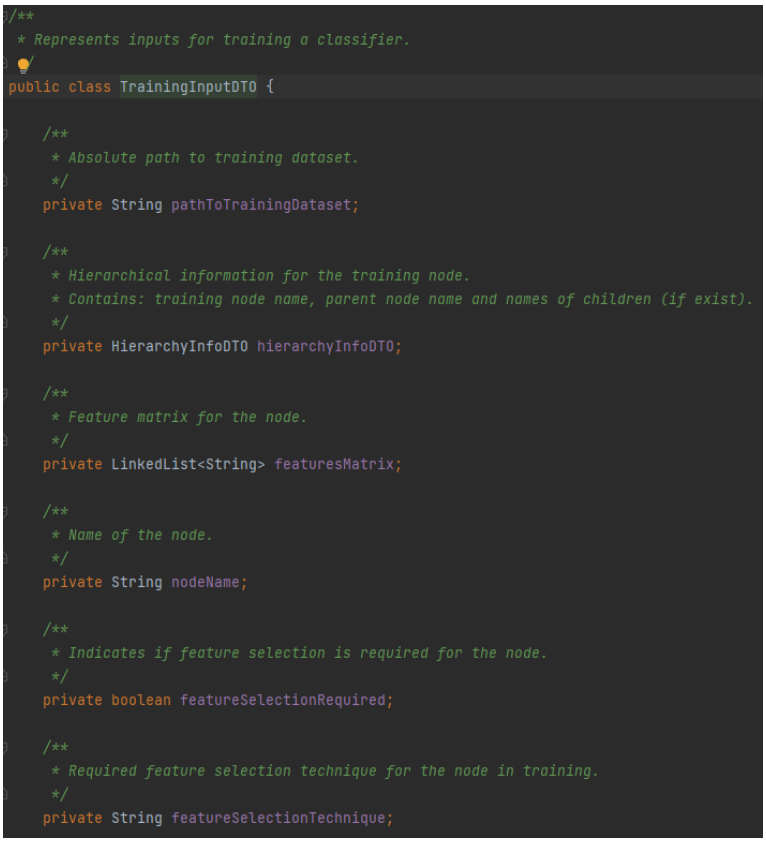
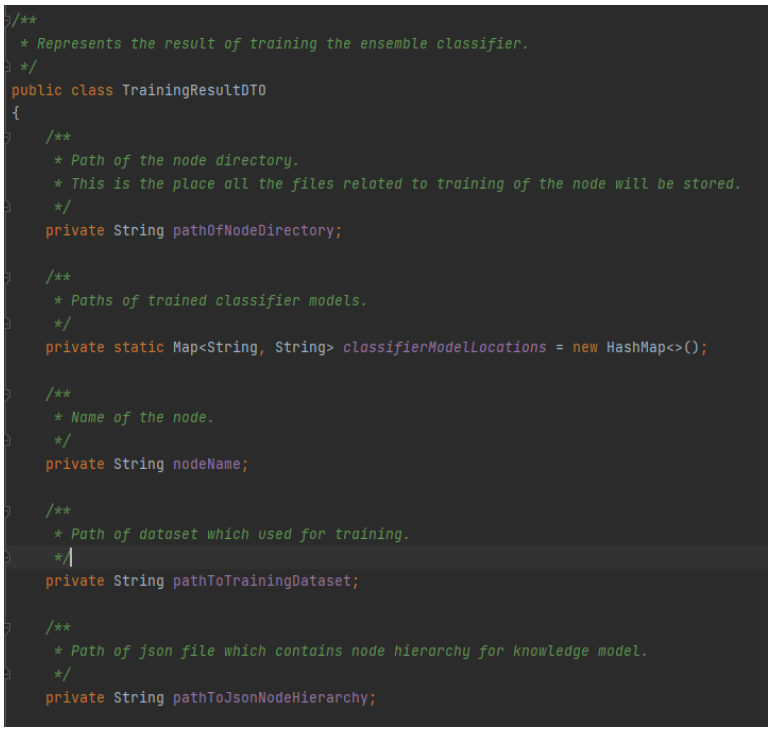


Figure 4.2.3: Base folder for a node



```java
/**
 * Represents inputs for training a classifier.
 */
public class TrainingInputDTO {

    /**
     * Absolute path to training dataset.
     */
    private String pathToTrainingDataset;

    /**
     * Hierarchical information for the training node.
     * Contains: training node name, parent node name and names of children (if exist).
     */
    private HierarchyInfoDTO hierarchyInfoDTO;

    /**
     * Feature matrix for the node.
     */
    private LinkedList<String> featuresMatrix;

    /**
     * Name of the node.
     */
    private String nodeName;

    /**
     * Indicates if feature selection is required for the node.
     */
    private boolean featureSelectionRequired;

    /**
     * Required feature selection technique for the node in training.
     */
    private String featureSelectionTechnique;
```

Figure 4.2.4: TrainingInputDTO



```java
/**
 * Represents the result of training the ensemble classifier.
 */
public class TrainingResultDTO
{
    /**
     * Path of the node directory.
     * This is the place all the files related to training of the node will be stored.
     */
    private String pathOfNodeDirectory;

    /**
     * Paths of trained classifier models.
     */
    private static Map<String, String> classifierModelLocations = new HashMap<>();

    /**
     * Name of the node.
     */
    private String nodeName;

    /**
     * Path of dataset which used for training.
     */
    private String pathToTrainingDataset;

    /**
     * Path of json file which contains node hierarchy for knowledge model.
     */
    private String pathToJsonNodeHierarchy;
```

Figure 4.2.5: TrainingResultDTO

Training process:

The training process is the phase where the system gets prepared to support the clinical diagnosis process by deriving diagnostic decisions by classification. The diseases that are supposed to be supported for diagnosis by the system should be pre-trained into the system. In this phase knowledge models are built for the diseases that the system is being trained for, and save all the information and derived knowledge during the process, to be used later for classifying clinical instances given.

The system could be trained for new diseases as well as update the existing knowledge models to adapt to new knowledge.

First of all, in the training process, the base directory for the node that is being trained is created. This is where all the information and files related to the particular node would be saved.

Before beginning the training process, the system needs to identify the status of the system and the nature of the training process that has been initiated.
If the system is being trained for the first time, in other words, if the system is a brand new project that has not been prepared to support any disease diagnosis, there are no node objects that were trained and no prior information to be loaded into the system. If the JSON file representing node hierarchy is not available, then the system is being trained for the first time.
Training can be done for a new node/ disease or to update an existing node. If the node already exists in the system, then it is an update or a re-training of a node. Then the existing node would be loaded into the system to access information saved in the last training cycle. Mainly the hierarchical information and other saved information need to be accessed during a re-training process.

The feature matrix is the list of symptoms that would be used for deriving diagnostic decisions for a particular disease. A feature matrix is specific to each disease, as the symptoms and other factors to be considered for the diagnosis process are different

from disease to disease. When the system is being trained for a particular disease, a feature matrix must be given as an input parameter to the training API. The training process cannot proceed if the feature matrix is missing, except only when it is an update or a re-training to an existing node/ disease then the feature matrix is not mandatory as an input parameter. The existing feature matrix that has been saved during the last training cycle would be used in this instance. The given feature matrix would be saved in the base directory for the node.

Correlation is used as the feature selection technique to identify the most important features and these selected features are used for training of the algorithms. The user is given the choice to decide if feature selection is needed and what is the technique to be used from the defined list of feature selection techniques. By executing different combinations, most effective technique combination can be preserved for the node.

Node hierarchical information needs to be saved in the system, during the training process. If it is a new node, update the parent and the children nodes (if required) with the new node information to insert the new node/ disease into the disease hierarchical tree into the place where it should belong to.

The training dataset would be saved to the node directory for information purposes and to be used during the next training cycle. If the node already exists in the system, and if the new and old datasets are not the same, then merge the new training dataset with the previous training dataset (which was stored in the node folder during the previous training cycle) to increase the size of the dataset which would help to achieve higher accuracy rates.

Then all these prepared data will be fed to the ensemble classifier for training, which would then invoke training of single classifiers that have been implemented.
Individual classifiers would be trained independently with the given data, and save the trained classifier models in the node directory. These saved classifier models would be used for classifying given instances later.

After the training is done for individual classifiers, evaluation of the trained model would follow immediately. The trained model would be evaluated and the results as well would be saved to the node directory.
In addition to saving the evaluation results of the individual classifiers for the node, a separate entry would be created to store the accuracy of the trained model. This would be done by each classifier that has been implemented in the system.
If the training process is an update or re-training of an existing node, the evaluation results of the new model are compared with the old model. If only the new model outperforms the old one, then replace the old model with the new model. If not keep the old model and discard the new model. This is done to avoid the performance drops when re-training an existing node.

At the end of the training process of individual classifiers, trained classifier models, evaluation results, and the evaluation summary entry would be created for each classifier, for the particular that has been trained. This evaluation summary entries would be used when deriving the final output of the ensemble classifier when classifying given instances.

The feature selection process is initiated to identify the most significant features that were used in the training and evaluation process. The correlation technique has been used for feature selection. Feature selection had been done for each node independently and the collection of them is taken as the set of most significant features for the node. These identified features would be used as the mandatory features that have to be entered into the system to derive a diagnosis. And when traversing through the tree hierarchy, before entering to classification job of a node, always checks if the mandatory features for the node are present with values; if any of them are not provided, then the classification process would be held at that level and provide the intermediate diagnosis result to the user with the information that needs to be provided to proceed with further diagnosis. These missing features would give users guidance on further clinical testing that needs to be performed.
And in a system knowledge update, when a bulk dataset is given, as explained earlier feature matrices are created for each node to be trained. To decide whether to re-train

a particular node, the system validates if the mandatory features for the node are present in the given training dataset.

When the training process of individual classifiers has been completed successfully, the root node which contains the whole node hierarchy is written back to the JSON file. This is the skeleton or the structure of the constructed layered knowledge model of the system.

Data keeping and versioning are very important to the systems that are being used in the medical domain. Any data that has been used in a clinical system cannot be deleted and should be preserved. Some laws see to the fulfillment of this requirement, which are different based on the country that the system is being used, but the underlying concept is all the data should be preserved. To adhere to these laws, the CDSS uses a file versioning system to keep the old data that has been used and to store the derived knowledge. When replacing old files with new files, always the old files are moved to an archive for versioning purposes.

The following figures would display how the explained training process has been implemented.

```java
public TrainingResult trainClassifierForSingleNode(String pathToTrainingDataset, HierarchyInfo hierarchyInfo,
                                                    LinkedList<String> featureMatrix, String nodeName,
                                                    boolean featureSelectionRequired, String featureSelectionTechnique)
        throws TrainClassifierException {
    LOGGER.info("Training has started..");
    try {
        /**
         * Creates base directory for node.
         */
        String pathOfNodeDirectory = ServiceUtil.getInstance().createNodeDirectory(nodeName);
        LOGGER.info("Base directory created for node " + nodeName);

        /**
         * Identify if this is the first time that the system is being trained.
         * If it is, then there would be no existing node hierarchy.
         */
        NodeService nodeService = NodeServiceImpl.getInstance();
        boolean isFirstTimeTraining = nodeService.isFirstTimeTraining();

        String pathToFeatureMatrix;
        String pathToMandatoryFeaturesList;

        Node nodesWithHierarchy;
        Node processingNode;
        Node parentNode = null;
        List<Node> childrenNodes = null;
        boolean isNodeExists = false;
```

```java
        /**
         * If this is the first time on training, then processing node would be a new node object.
         * If not the first time on training, load the existing node hierarchy.
         */
        if (isFirstTimeTraining) {
            /**
             * If feature matrix is absent, cannot proceed with training.
             */
            if (featureMatrix == null || featureMatrix.isEmpty()) {
                LOGGER.error(" Feature matrix for " + nodeName + " is absent.");
                throw new TrainClassifierException(" Feature matrix for " + nodeName + " is absent.");
            }
            String[] featuresArray = featureMatrix.toArray(new String[featureMatrix.size()]);
            /**
             * Save feature matrix for the node, in a csv file.
             */
            pathToFeatureMatrix = saveFeatureMatrix(nodeName, pathOfNodeDirectory, featuresArray);
            pathToMandatoryFeaturesList = getMandatoryFeaturesFilePathForNode(featureSelectionTechnique,
                    pathOfNodeDirectory);

            processingNode = new Node();
            nodesWithHierarchy = processingNode;
            LOGGER.info("This is the first time the system is being trained.");
        } else {
            nodesWithHierarchy = nodeService.loadNodes();

            /**
             * Check if the node to be trained on is already existing in the system.
             */
            isNodeExists = nodeService.isExist(nodeName);
            if (isNodeExists) {
                /**
                 * If node exists, processing node should be the existing node.
                 */
                processingNode = nodeService.getNode(nodeName);
                pathToFeatureMatrix = processingNode.getPathToFeatureMatrix();
                pathToMandatoryFeaturesList = processingNode.getPathToMandatoryFeatures();
                LOGGER.info(nodeName + " node is already exists in system. Loading the node..");
```

```
        } else {
            /**
             * If feature matrix is absent, cannot proceed with training.
             */
            if (featureMatrix == null || featureMatrix.isEmpty()) {
                LOGGER.error(" Feature matrix for " + nodeName + " is absent.");
                throw new TrainClassifierException(" Feature matrix for " + nodeName + " is absent.");
            }
            String[] featuresArray = featureMatrix.toArray(new String[featureMatrix.size()]);
            /**
             * Save feature matrix for the node, in a csv file.
             */
            pathToFeatureMatrix = saveFeatureMatrix(nodeName, pathOfNodeDirectory, featuresArray);
            pathToMandatoryFeaturesList = getMandatoryFeaturesFilePathForNode(
                    featureSelectionTechnique, pathOfNodeDirectory);

            /**
             * If node does not exist, processing node is a new node object
             */
            processingNode = new Node();
            LOGGER.info(nodeName + " node is new to system.");
        }

        parentNode = nodeService.getNode(hierarchyInfo.getParentName());
        childrenNodes = nodeService.getNodes(hierarchyInfo.getChildrenNames());

        /**
         * Update node hierarchy according to the hierarchical information.
         * Update parent and children nodes too.
         */
        updateNodeHierarchy(processingNode, parentNode, childrenNodes);
    }
```

```
        /**
         * If node existed, then merge training datasets, and use the merged dataset for training.
         * If not, then just copy the dataset into the node directory.
         */
        String newPathOfTrainingDataset = updateDataset(isFirstTimeTraining, isNodeExists,
                pathToTrainingDataset, processingNode.getPathToTrainingDataset(), pathOfNodeDirectory);
        LOGGER.info("Copying dataset - Done.");

        /**
         * Extract class column name.
         */
        String classColumnName = getClassColumnName(pathToTrainingDataset);

        /**
         * Train classifiers.
         */
        LOGGER.info("Training classifiers..");
        TrainingResult trainingResult = EnsembleClassifier.getInstance()
                .trainEnsemble(pathToTrainingDataset, nodeName, pathOfNodeDirectory, classColumnName);

        addInformationToProcessingNode(pathToFeatureMatrix, processingNode,
                parentNode, childrenNodes, newPathOfTrainingDataset, trainingResult,
                pathToMandatoryFeaturesList, featureSelectionTechnique);

        /**
         * Write updated node hierarchy to json file. (store root node)
         */
        String base = FileHandlerService.createDirectory(CdssConstants.NODE_HIERARCHY_DIRECTORY);
        String jsonFilePath = FileHandlerService.createFilePath(base,
                CdssConstants.CDSS_HIERARCHY, CdssConstants.JSON_EXTENSION);
        String pathToJsonNodeHierarchy = FileHandlerService.writeToJSON(nodesWithHierarchy, jsonFilePath);
        LOGGER.info("Node hierarchy stored in json file: " + pathToJsonNodeHierarchy);
        trainingResult.setPathToJsonNodeHierarchy(pathToJsonNodeHierarchy);

        LOGGER.info("Training process done!");
        return trainingResult;
    } catch (Exception e) {
        LOGGER.error("Error when training classifiers for " + nodeName, e);
        throw new TrainClassifierException("Error when training classifier.", e);
    }
}
```

Figure 4.2.6: Implementation of training classifier for a single node

```
/**
 * Check if the trained classifier is better.
 * Handle and discard low performing classifier model.
 */
boolean newModelPerformsBetter = ClassifierUtils
        .checkIfNewModelPerformsBetter(evaluationResult, classifierName, nodeDirectoryPath);
if (!newModelPerformsBetter) {
    LOGGER.error("Performance reduction of " + classifierName + " for training data file: " + trainingDataSetPath
            + ". New performance: " + accuracyScore + " for classifier " + classifierName);
    return pathOfSavedModel;
}

/**
 * Copy the trained model to destination location.
 */
String finalModelPath = FileHandlerService.copyOrReplaceFile(pathOfSavedModel, baseModelLocation);

/**
 * Write evaluation results to file.
 */
ClassifierUtils.writeEvaluationResultToFile(evaluationResult, baseEvaluationReportLocation);
ClassifierUtils.writeAccuracyOfClassifierToPerformanceFile(evaluationResult.getAccuracy(),
        nodeDirectoryPath, classifierName);
```

Figure 4.2.7: Implementation of comparing the evaluation summaries of old and new models before saving new model

4.3 Implementation of diagnosis support/ Single instance classification

This section explains how the implementation of the single instance classification is done.

This is the component that supports diagnosis support. The trained classifier models are loaded into the system and derive the final diagnosis decisions for a given set of symptom-value combinations. An ensemble classifier has been implemented to support this decision-making process by classification. The rest of the section will explain the implementation details.

The user needs to enter the factors that are required for the diagnosis of the patient, such as biometric information like age, gender, relevant lifestyle facts and habits, relevant environmental information, occupation details, and the visible symptoms of the patient along with measurement values, and the results of any clinical tests that

have been performed, and any other information that would support the diagnosis process.
The input data values can be numerical or boolean values (present or not).

If the user wishes to, he can input a diagnosis category as the starting point/ starting node of the diagnosis process. This feature has been provided to suffice the need of the users to start the diagnosis at a defined level.

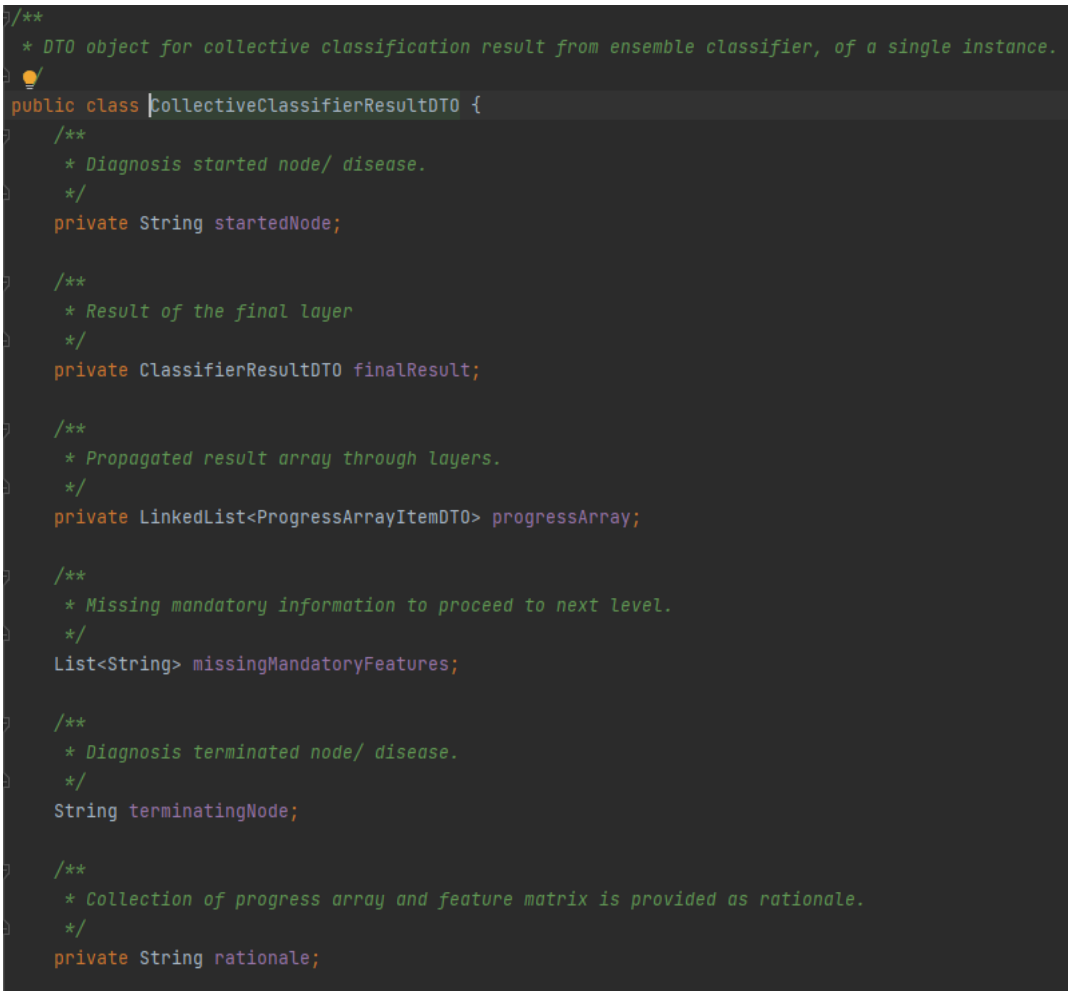
The primary output is the final result of the ensemble classifier and the rationale or explanation for the diagnosis decision that has been made consists of the progress path of tree traversing, and the summary of the patient status.
If there was not enough information to reach a leaf node, the output result would be the list of missing information and the intermediate diagnosis result (terminated node).

```
@PostMapping("/classify_instance")
public ResponseEntity<CollectiveClassifierResultDTO> classifyInstance(@RequestBody ClassifierInputDTO inputDTO)
{
```

Figure 4.3.1: REST API for classifying a single instance

```
/**
 * DTO object for classifier input given for classifying a single instance.
 */
public class ClassifierInputDTO {

    /**
     * User input on what is the area that the disease could be in.
     */
    private String userHighLevelPrediction;

    /**
     * Key: symptom.
     * Value: value for the symptom; can be categorical, numerical or boolean (present or not).
     */
    private Map<String, String> symptoms;
```

Figure 4.3.2: ClassifierInputDTO



```java
/**
 * DTO object for collective classification result from ensemble classifier, of a single instance.
 */
public class CollectiveClassifierResultDTO {
    /**
     * Diagnosis started node/ disease.
     */
    private String startedNode;

    /**
     * Result of the final layer
     */
    private ClassifierResultDTO finalResult;

    /**
     * Propagated result array through layers.
     */
    private LinkedList<ProgressArrayItemDTO> progressArray;

    /**
     * Missing mandatory information to proceed to next level.
     */
    List<String> missingMandatoryFeatures;

    /**
     * Diagnosis terminated node/ disease.
     */
    String terminatingNode;

    /**
     * Collection of progress array and feature matrix is provided as rationale.
     */
    private String rationale;
```

Figure 4.3.3: CollectiveClassifierResultDTO

Single instance classification process:

The first step is loading the starting node. Normally, the root node is the starting node for the instance classification process. The system has provided the facility to enter a starting point for the diagnosis process. If the user is certain that the diagnosis should be done for a particular area/ disease category, this feature can be used. If the user has entered a high-level diagnosis category, then that would be the starting point.

To start the diagnosis, all the mandatory information for the starting node must have been entered into the system by the user. The most significant features for trained

nodes, that have been identified during the training process would be used for this purpose. The same set of the most significant features for the node is taken as the mandatory information for the node. This condition check has been implemented because if the system is to derive an accurate diagnosis, then it needs to be provided with at least the most important information for deriving diagnostic decisions; if not the system output would not be accurate which would lead to misdiagnosis.
If not all the mandatory information is present with values, terminate the diagnosis process and give the intermediate result (if there is any) to the user along with the list of missing information. This information gives a guide to the user on what are the necessary observations that have to be made, the information to be collected, and the required clinical tests and operations that need to be performed.

If all the requirements are fulfilled, the diagnosis process would be initiated. The entered information/ data is forwarded to the ensemble classifier to start classification.

The ensemble classifier would forward the data to the individual classifiers to perform the classification. Each classifier algorithm loads the saved trained model and performs classification and provides the result. The ensemble classifier produces the result by following the priority-based mode mechanism, which was described in chapter 3.

The result from the first layer would be the next node. If a classifier exists for the disease/ node in the system, then repeat the classification process for that node. If not, the classification process ends and results will be published.

```java
public CollectiveClassifierResult classifyInstance(ClassifierInput input) throws ClassifyInstanceException {
    LOGGER.info("Classifying instance started..");
    String nextNodeName = "";
    Node nextNode;
    String pathToInputFile;
    try {
        String directoryPath = new StringBuilder().append(CdssConstants.CLASSIFY_INSTANCES_DIRECTORY)
                .append(LocalDateTime.now().toString()).append("/").toString();
        FileHandlerService.createDirectory(directoryPath);

        CollectiveClassifierResult collectiveClassifierResult = new CollectiveClassifierResult();

        LinkedList<ClassifierResult> progressArrayOfResults = new LinkedList<>();
        EnsembleClassifier ensembleClassifier = EnsembleClassifier.getInstance();
        NodeService nodeService = NodeServiceImpl.getInstance();

        /**
         *  userHighLevelPrediction - classification starting node.
         */
        String userHighLevelPrediction = input.getUserHighLevelPrediction();
        if (userHighLevelPrediction == null && !userHighLevelPrediction.isEmpty()
                && nodeService.isExist(userHighLevelPrediction)) {
            nextNodeName = userHighLevelPrediction;
        }
```

```java
        do {
            /**
             * Load Node object
             */
            if (nextNodeName.isEmpty()) {
                nextNode = nodeService.loadNodes(); // gives the root node of the knowledge tree
            } else {
                nextNode = getStartingNode(userHighLevelPrediction);
            }

            /**
             * Check if all the mandatory features are present.
             * If not, terminate the process.
             */
            List<String> symptoms = new ArrayList<>(input.getSymptoms().keySet());
            InputVerificationResult inputVerificationResult = ServiceUtil.getInstance()
                    .verifyInput(symptoms, nextNode.getName());
            if (!inputVerificationResult.isInputValid()) {
                LOGGER.info("Mandatory information missing for " + nextNodeName + " ..");
                LOGGER.info("Missing mandatory information: " + inputVerificationResult.getMissingMandatoryFeatures());
                collectiveClassifierResult
                        .setMissingMandatoryFeatures(inputVerificationResult.getMissingMandatoryFeatures());
                collectiveClassifierResult.setTerminatingNode(nextNodeName);
                collectiveClassifierResult = createCollectiveClassifierResult(progressArrayOfResults, collectiveClassifierResult);
                return collectiveClassifierResult;
            }

            LOGGER.info("Progressing for " + nextNodeName + " ..");
```

```
            /**
             * Create input file, against feature matrix.
             */
            LinkedList<String> featureMatrix = new LinkedList<>(FileHandlerService
                    .readFromCsv(nextNode.getPathToMandatoryFeatures()).get(0));
            if (featureMatrix == null || featureMatrix.isEmpty()) {
                return null;
            }
            LOGGER.info("Feature matrix for node " + nextNodeName + " : " + featureMatrix);
            pathToInputFile = createInputCsv(featureMatrix, input.getSymptoms(), directoryPath);

            /**
             * Classify instance for the node.
             */
            ClassifierResult result = ensembleClassifier.classifyInstance(pathToInputFile,
                    nextNode.getClassifierModelLocations(), nextNode.getPathOfNodeDirectory());
            result.setNodeName(nextNodeName);
            progressArrayOfResults.add(new ProgressArrayItem(result, featureMatrix,
                    inputVerificationResult.getAllMandatoryFeaturesShouldPresent()));

            /**
             * If resulting node has a classifier for next level,
             * and if the current node has children and next node to be progress is a children of the current node,
             * then progress.
             */
            nextNodeName = result.getResult();
        } while (nodeService.isExist(nextNodeName)
                && nextNode.getChildren() != null && !nextNode.getChildren().isEmpty()
                && nextNode.getChildrenNames().contains(nextNodeName));

        collectiveClassifierResult = createCollectiveClassifierResult(progressArrayOfResults, collectiveClassifierResult);
```

```
        /**
         * Save information of classifying instance for data keeping purposes.
         */
        saveInstanceInfo(input, collectiveClassifierResult, directoryPath);

        LOGGER.info("Classification process done!");
        return collectiveClassifierResult;

    } catch (Exception e) {
        LOGGER.error("Error when classifying single instance for " + nextNodeName, e);
        throw new ClassifyInstanceException("Error when classify instance in ensemble classifier.", e);
    }
}
```

Figure 4.3.4: Implementation of single instance classification

Rationale:

The rationale is the explanation or the reasoning behind the output result produced by the system. Producing a rationale along with the results improves the trust in the system and more accepting of the decisions derived by clinicians.

The basis for the prediction derived by the CDSS is provided to the user as the rationale, which he can use as a summary or reasoning.
The proposed system has two major parts of the rationale or the explanation or the reasoning behind the final result produced by the system.
They are the propagation array of the layered knowledge tree and the summary of the most important symptom-value combinations that were used as the base for deriving the diagnosis value.

The traversed/ progressed path through the disease hierarchy tree structure would be presented to the user as the first part of the rationale. The record of intermediate results is kept in the system temporally when traversing through the tree in a linked list, which would be then published.

The other part is the summary of the patient which consists of the most important symptom-value combinations that have been used as the basis for deriving the diagnosis decisions. This summary is presented to the user as the basis for diagnosis. The set of identified most significant features would be used for this. This is a rather detailed explanation of the basis for each decision made by the system. The rationale for each node would be presented to the user.

Calculating the ensemble classifier result:

An ensemble classifier has been used for decision making, integrating three individual classifier algorithms: Naïve Bayes classifier, Random Forest classifier, and Support Vector Machine classifier.

Following the weighted voting technique, the priority-based mode method has been used for deriving the final result of the ensemble classifier.
Priorities are given to the classifiers using the performance statistics of the individual classifiers.
Each classifier has been evaluated independently for each node and obtained performance statistics. Accuracy performance measurement has been used as the base for this.
Higher the accuracy, the higher the priority that is given to the classifier. The classifier which has the highest accuracy among the trained classifiers is given the highest priority, likewise follows through the list of classifiers and a priority number is given to all the individual classifiers.

If all individual classifiers show the same performance, the ensemble classifier's final output is determined using either the majority voting methodology or the pure mode method.
If all of the outputs differ, the Random Forest classifier's result takes precedence because it has been shown in the literature to be highly accurate, and the classifier is an ensemble classifier.

The following code segments display how the calculation of the ensemble classifier has been done.

```java
private ClassifierResult calculateEnsembleResult(List<ClassifierResult> resultsFromSingleClassifiers, String nodeLocation)
        throws FileHandlerException {
    ClassifierResult ensembleResult = new ClassifierResult();
    ensembleResult.setClassifier(CdssConstants.ENSEMBLE_CLASSIFIER);

    Map<String, String> resultsMap = new HashMap();
    for (ClassifierResult singleResult : resultsFromSingleClassifiers) {
        resultsMap.put(singleResult.getClassifier(), singleResult.getResult());
    }

    String evaluationSummaryFilePath = nodeLocation + CdssConstants.EVALUATION_SUMMERY + CdssConstants.CSV_EXTENSION;
    List<List<String>> classifierPerformance = FileHandlerService.readFromCsv(evaluationSummaryFilePath);

    Map<String, Double> classifierPerformanceMap = new HashMap<>();
    for (List<String> singleClassifier : classifierPerformance) {
        classifierPerformanceMap.put(singleClassifier.get(0), Double.parseDouble(singleClassifier.get(1)));
    }

    /**
     * Order the classifiers according to the performance.
     */
    double accuracyOfNB = classifierPerformanceMap.get(CdssConstants.NAIVE_BAYES);
    double accuracyOfRF = classifierPerformanceMap.get(CdssConstants.RANDOM_FOREST);
    double accuracyOfSVM = classifierPerformanceMap.get(CdssConstants.SVM);

    String r1 = "";
    String r2 = "";
    String r3 = "";
```

```java
    /**
     * This is the results to be used as the final output,
     * if all results are different from each other.
     */
    String privilegedResult = "";

    /**
     * If all classifiers are same, RF is given priority.
     */
    if (accuracyOfNB == accuracyOfRF && accuracyOfRF == accuracyOfSVM) {
        List<String> resultValue = new ArrayList<>(resultsMap.values());
        r1 = resultValue.get(0);
        r2 = resultValue.get(1);
        r3 = resultValue.get(2);

        privilegedResult= resultsMap.get(CdssConstants.RANDOM_FOREST);
    }
```

```
    } else {
        String highestPerformanceClassifier = "";
        String secondHighestPerformanceClassifier = "";
        String thirdHighestPerformanceClassifier = "";

        if (accuracyOfNB >= accuracyOfRF) {
            if (accuracyOfNB >= accuracyOfSVM) {
                highestPerformanceClassifier = CdssConstants.NAIVE_BAYES;
                if (accuracyOfRF >= accuracyOfSVM) {
                    secondHighestPerformanceClassifier = CdssConstants.RANDOM_FOREST;
                    thirdHighestPerformanceClassifier = CdssConstants.SVM;
                } else {
                    secondHighestPerformanceClassifier = CdssConstants.SVM;
                    thirdHighestPerformanceClassifier = CdssConstants.RANDOM_FOREST;
                }
            } else {
                highestPerformanceClassifier = CdssConstants.SVM;
                secondHighestPerformanceClassifier = CdssConstants.NAIVE_BAYES;
                thirdHighestPerformanceClassifier = CdssConstants.RANDOM_FOREST;
            }
        } else {
            if (accuracyOfRF >= accuracyOfSVM) {
                highestPerformanceClassifier = CdssConstants.RANDOM_FOREST;
                if (accuracyOfNB >= accuracyOfSVM) {
                    secondHighestPerformanceClassifier = CdssConstants.NAIVE_BAYES;
                    thirdHighestPerformanceClassifier = CdssConstants.SVM;
                } else {
                    secondHighestPerformanceClassifier = CdssConstants.SVM;
                    thirdHighestPerformanceClassifier = CdssConstants.NAIVE_BAYES;
                }
            } else {
                highestPerformanceClassifier = CdssConstants.SVM;
                secondHighestPerformanceClassifier = CdssConstants.NAIVE_BAYES;
                thirdHighestPerformanceClassifier = CdssConstants.RANDOM_FOREST;
            }
        }
        r1 = resultsMap.get(highestPerformanceClassifier);
        r2 = resultsMap.get(secondHighestPerformanceClassifier);
        r3 = resultsMap.get(thirdHighestPerformanceClassifier);
        privilegedResult = r1;
    }
```

```
        if (r1.equals(r2)) {
            ensembleResult.setResult(r1);
            return ensembleResult;
        } else if (r1.equals(r3)) {
            ensembleResult.setResult(r1);
            return ensembleResult;
        } else if (r2.equals(r3)) {
            ensembleResult.setResult(r2);
            return ensembleResult;
        } else {
            ensembleResult.setResult(privilegedResult);
            return ensembleResult;
        }
}
```

Figure 4.3.5: Implementation of the calculation of ensemble classifier result

4.4 Implemented Prototype – User Interfaces and Sample Scenarios

There are two main user interfaces. Diagnosis support window and the administration window.

Diagnosis support window is the main interface. It provides the interface to the main functionality of the system, disease-symptom prediction, diagnosis support. Clinicians are the targeted user group here.

Administration window provides the admin functionalities. Training support and evaluation support is under this. Training can be done for a single node or for multiple nodes/ system re-training. Evaluation also can be done for a single node or for the whole system.

The following are the UIs of the implemented prototype and few sample scenarios performed.

Figure 4.4.1: CDSS (Prototype implementation) - Home Page

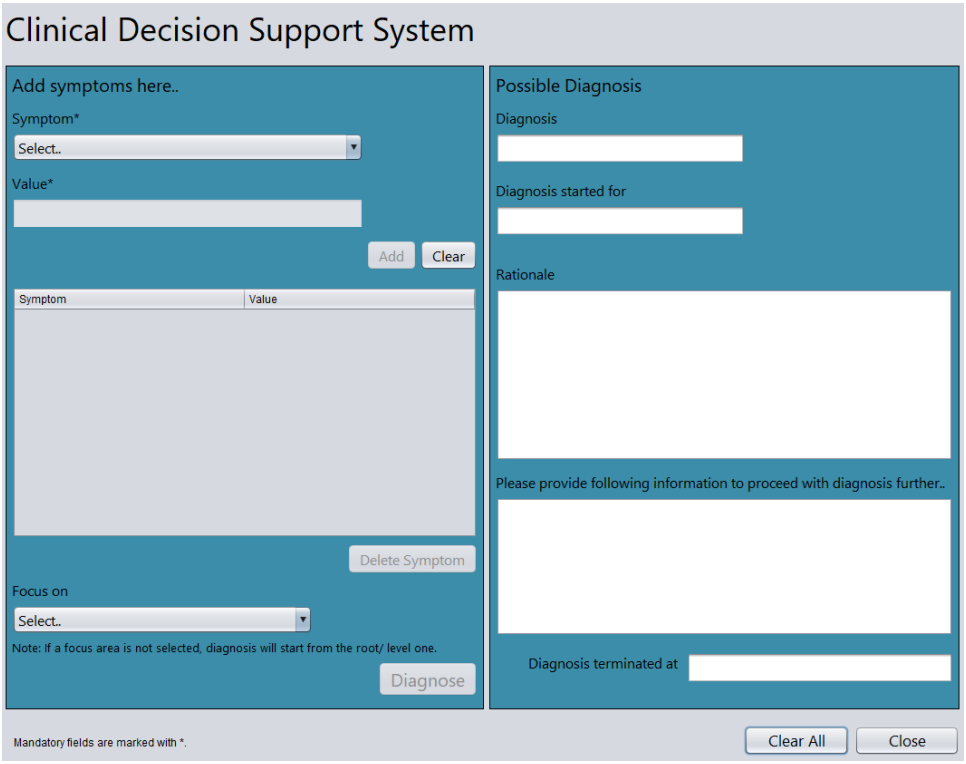


Figure 4.4.2: CDSS (Prototype implementation) – Diagnosis Support

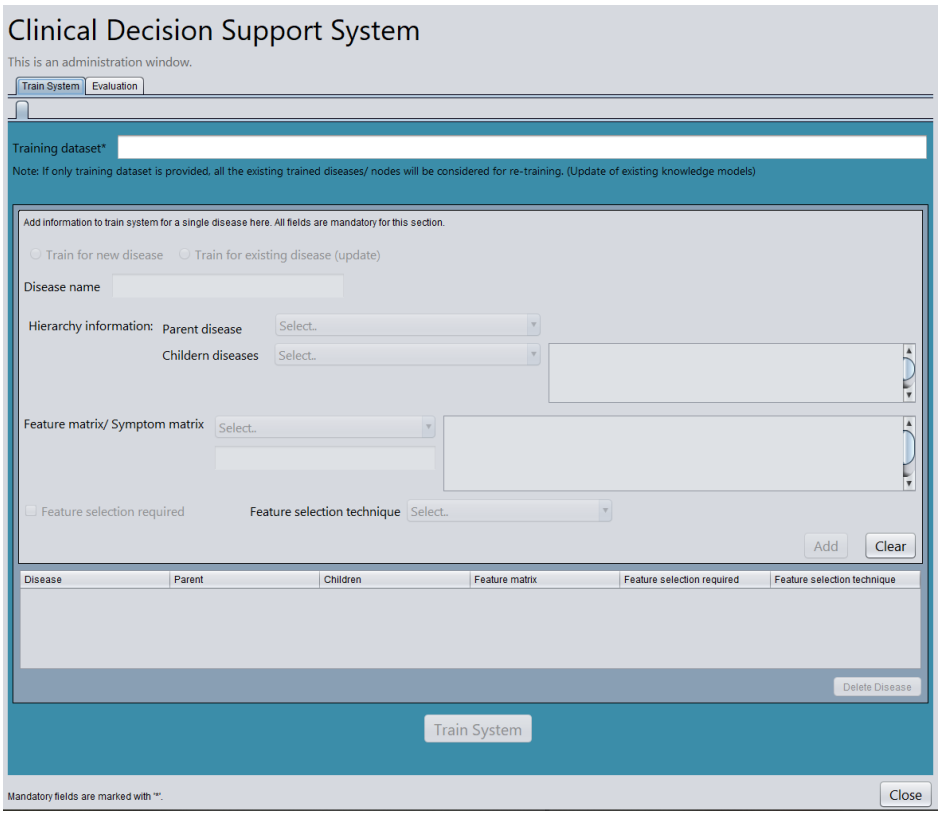


Figure 4.4.3: CDSS (Prototype implementation) – Training Window

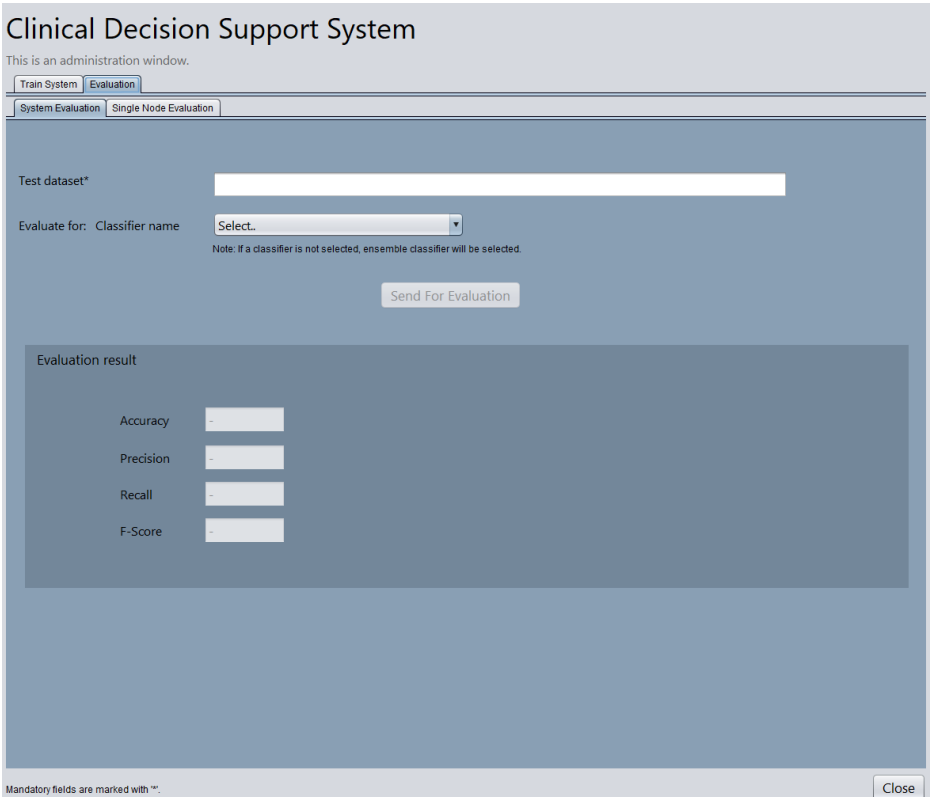


Figure 4.4.4: CDSS (Prototype implementation) – System Evaluation

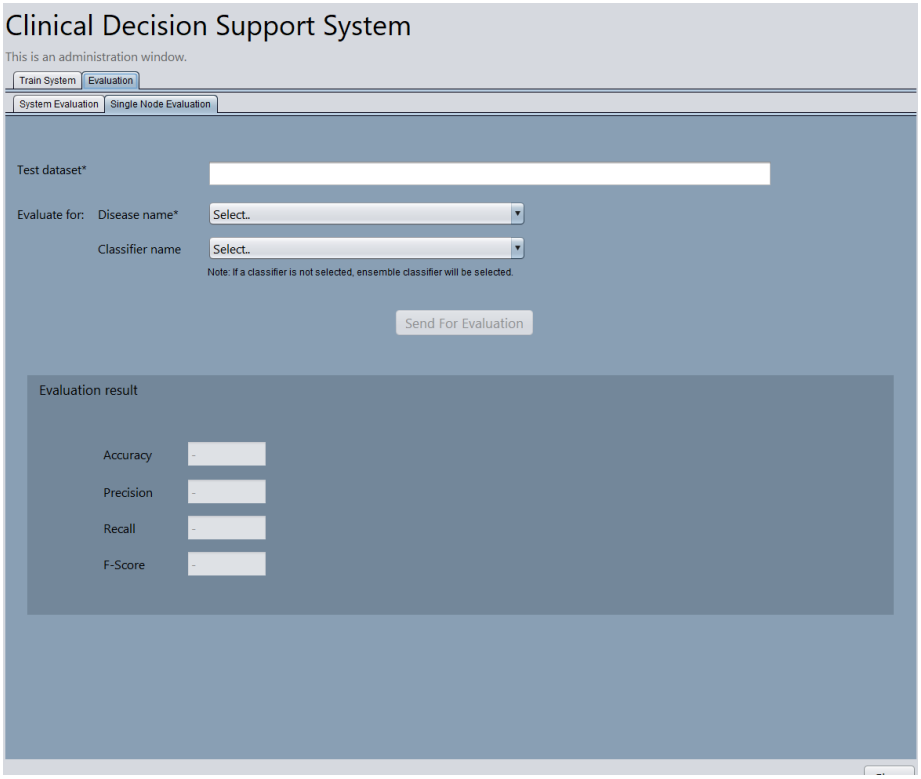


Figure 4.4.5: CDSS (Prototype implementation) – Single Node Evaluation

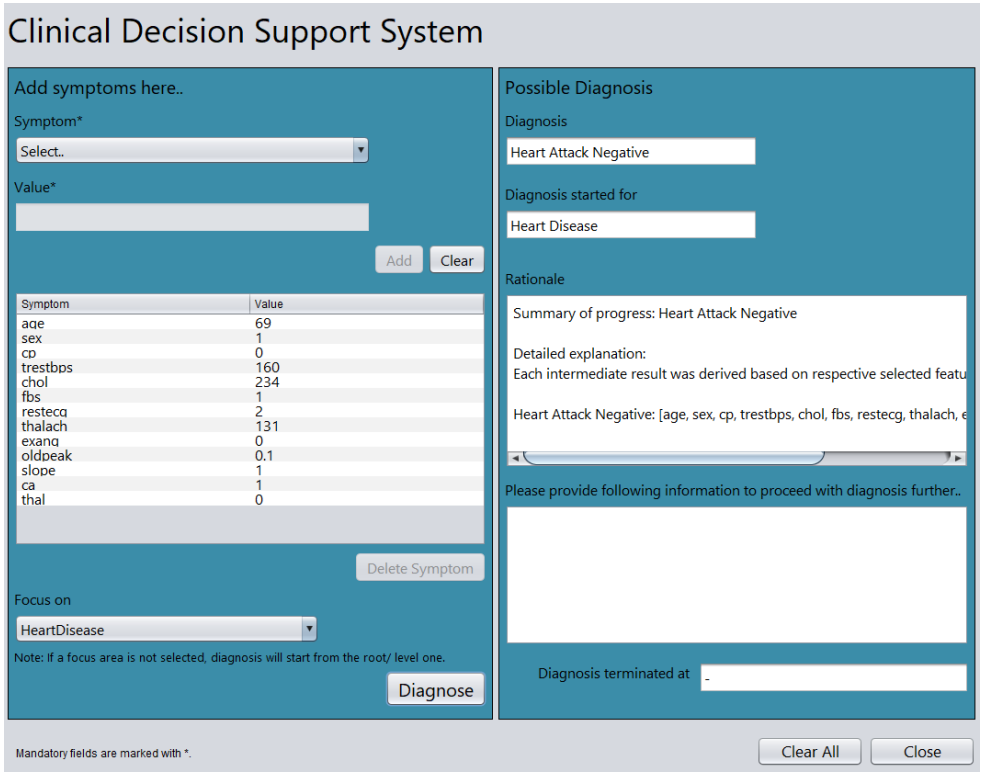


Figure 4.4.6: Sample scenario 1 – Heart Attack Negative

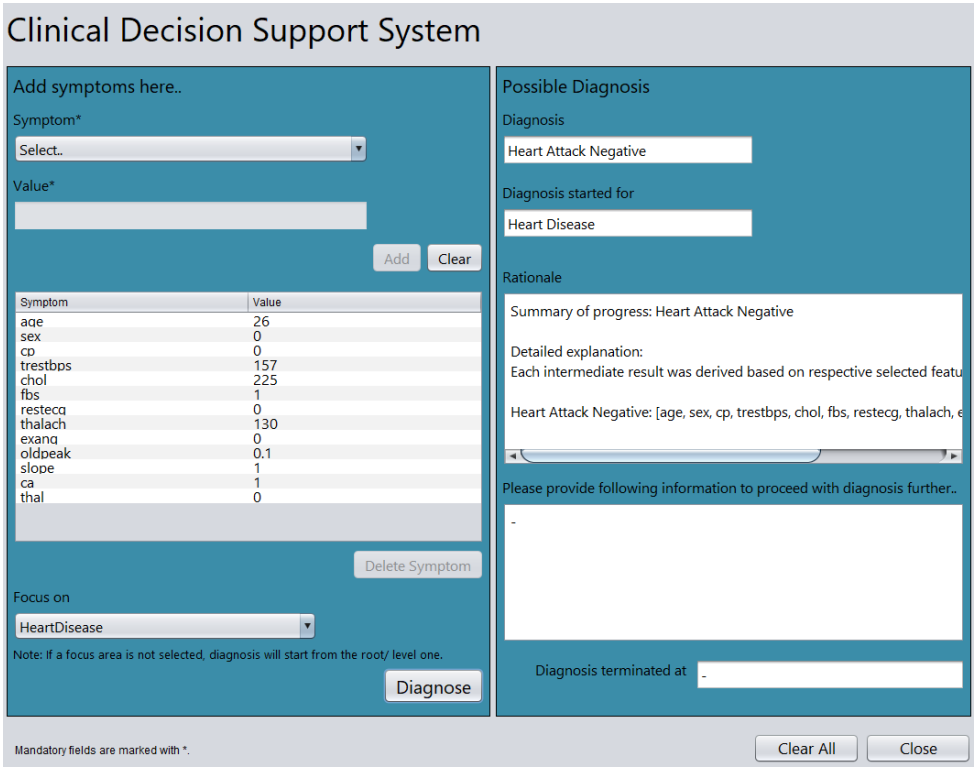


Figure 4.4.7: Sample scenario 2 – Heart Attack Negative

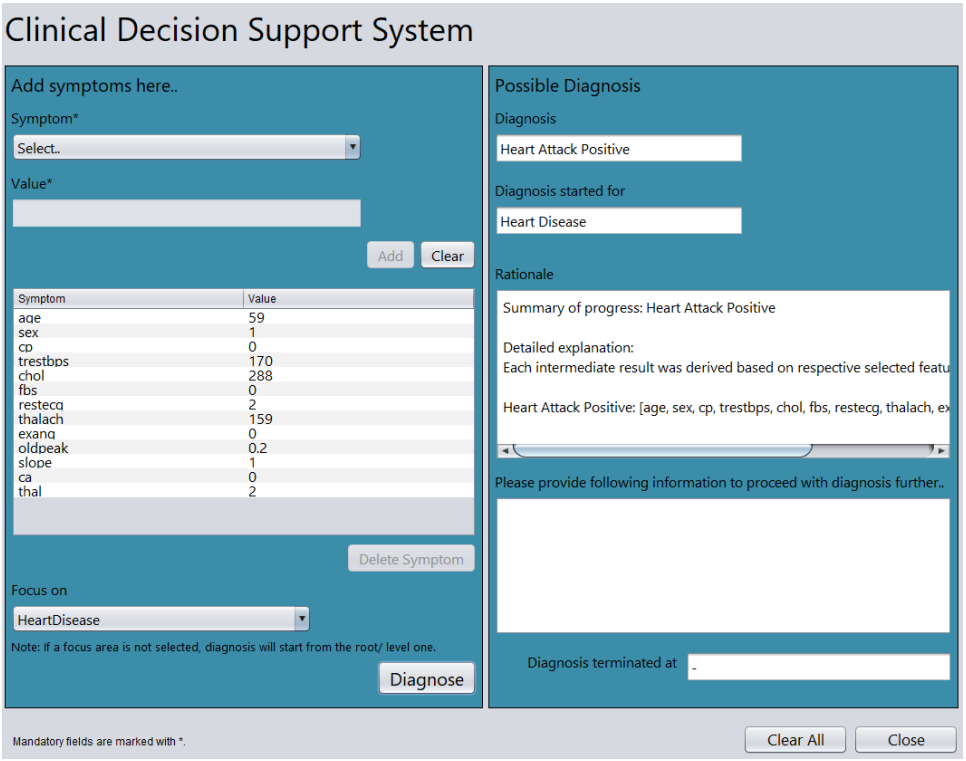


Figure 4.4.8: Sample scenario 3 – Heart Attack Positive

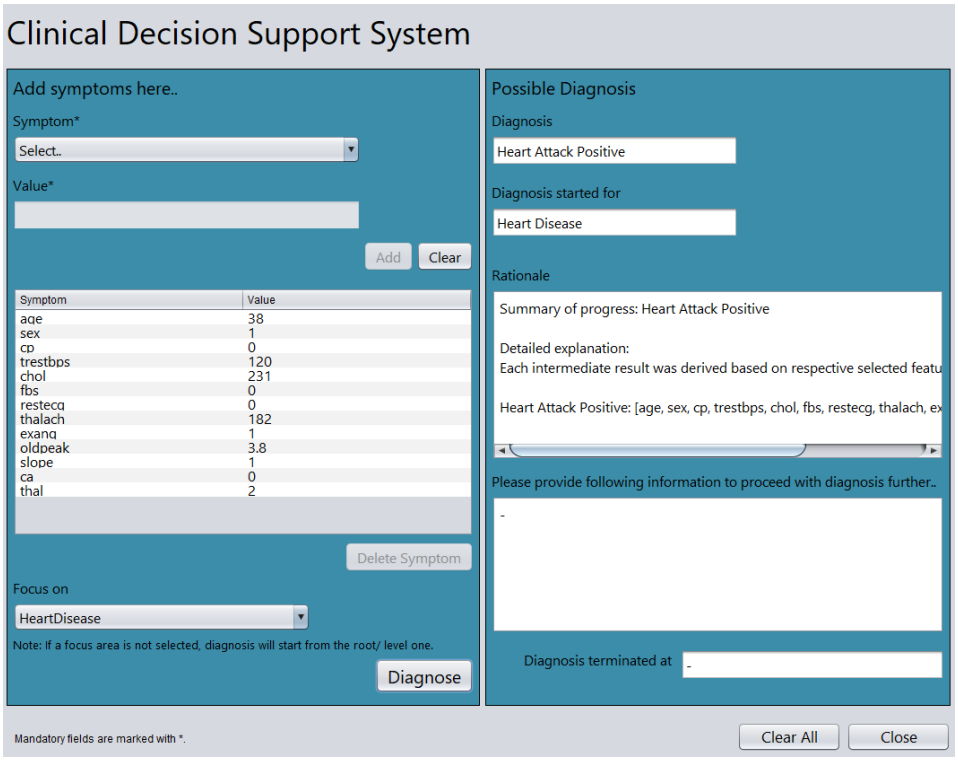


Figure 4.4.9: Sample scenario 4 – Heart Attack Positive

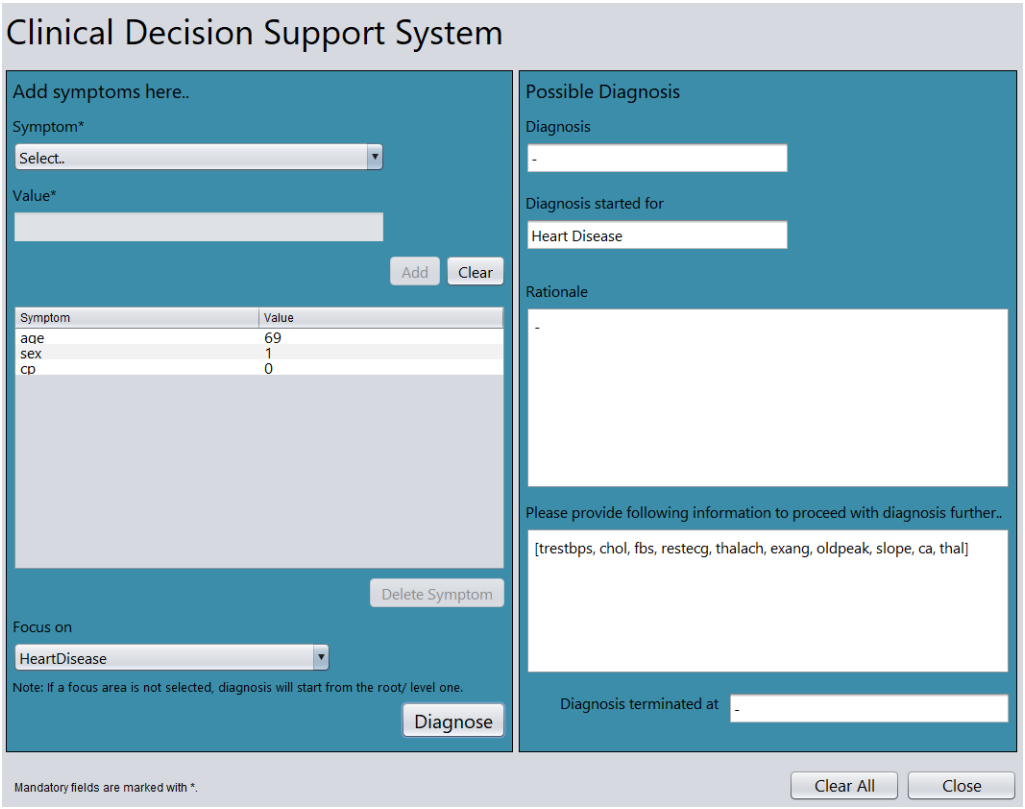


Figure 4.4.10: Sample scenario 5 – Missing mandatory information

CHAPTER 5

EVALUATION

The proposed system has implemented a layered knowledge model for diagnosis support using past medical records. An ensemble classifier has been implemented as the classification technique for deriving the predictions on the diagnosis. As explained in the chapter methodology and implementation, the Naïve Bayes classifier, Random Forest classifier, and Support Vector Machine classifier have been used for the ensemble classification. The proposed system has used a priority-based mode method to derive the final result of the ensemble classifier.

This study has used two main evaluation methods to evaluate the proposed architecture.
They are the clinical-diagnosis-result-based evaluation and the performance-based evaluation.

5.1 Clinical-diagnosis-result-based evaluation

The evaluation aims to access the prediction results produced by the proposed system. The goodness of the ensemble classifier is evaluated by comparing the final result that was obtained by the ensemble classifier, against the final results that were obtained by using each single classifier algorithm individually (Naïve Bayes, Random Forest, and SVM).

To evaluate the results of the system a test dataset that consists of 100 labeled instances was fed into the system and compared the predictions with the actual/ correct diagnosis.

The evaluation of the proposed system has been carried out under several matrices. Accuracy, precision, recall, and F-score have been used as measurements for evaluation.

Definitions

TP – True-Positive – Predicted positive and actually positive

TN – True-Negative – Predicted negative and actually negative

FN – False-Negative – Predicted negative but actually positive

FP – False-Positive – Predicted positive but actually negative

$$\text{Accuracy} \Rightarrow A = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} \Rightarrow P = \frac{TP}{TP + FP}$$

$$\text{Recall} \Rightarrow R = \frac{TP}{TP + FN}$$

$$\text{F-score} \Rightarrow F1 \Rightarrow 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The accuracy value indicates the percentage of the correct predictions made by the system, against all the predictions made. This is a good performance measure for this system as the given dataset is evenly distributed among the classes.

The precision measurement provides a good overview of the system on what proportion of patients that the system has diagnosed as having heart disease, actually had heart disease. This is an important measurement for systems like CDSS which is a clinical application because if a patient that does not actually have the disease was treated for a critical disease situation, it would lead to a life-threatening disaster.

Recall or sensitivity measure indicates what proposition of patients that actually had a heart disease was diagnosed as having a heart disease. This is also an important measurement for a clinical application since if the system fails to identify a patient with a critical disease situation, that would be a huge failure because a patient who is in dire need of emergency care treatments would not be getting any such kind of treatment, just because the system failed to identify the need of it.

F1 score: the total performance of the system was calculated by combining Precision and Recall into a single balance value.

The following table lists the summarized view of the comparison of the results. The system has been evaluated for heart disease classification, employing each classifier individually, and using all the classifiers as an ensemble method as proposed in this study following a priority-based mode method.

Classifier	Accuracy	Precision	Recall	F-Score
Naïve Bayes	91%	0.91	0.91	0.91
Random Forest	85%	0.85	0.85	0.85
Support Vector Machine	75%	0.75	0.75	0.75
Ensemble Classifier	95%	0.91	0.96	0.93

Table 5.1.1: Result comparison of classifiers for heart disease classification

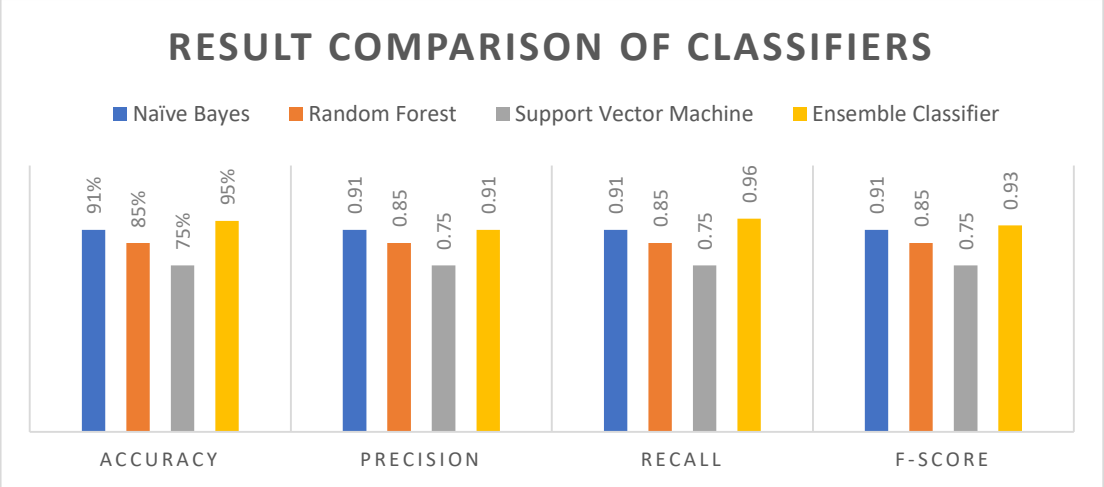


Figure 5.1.1: Graphical visualization of result comparison of classifiers

Using the results obtained from the above evaluation, the result comparison was done for individual classifiers and the ensemble classifier implemented.
The result analysis shows that the ensemble classifier has shown higher values for the measurements considered, for heart disease classification, than the individual classifiers.
Among the individual classifiers, the Naïve Bayes classifier has shown more effective results than the other two classifiers for heart disease classification.

Looking at the results obtained, you can see that precision for Naïve Bayes and ensemble classifiers has the same value. This means, if the system has predicted that a patient has a heart disease, the proportionate of the diagnose being correct that he actually has a heart disease is same for Naïve Bayes and ensemble classifier. This leaves a thinking point in the result analysis when precision values are considered, since according to the performance evaluation (explained in the following section), ensemble classifier has shown much less performance than the Naïve Bayes, yet it has not shown considerable improvement in precision value compared to Naïve Bayes.

The above evaluation is done solely based on the diagnosis results obtained from the classifiers. How much better each classifier has derived the diagnosis has been evaluated statistically. Labeled instances were fed to the system as the test dataset and the actual diagnosis values were compared with the predictions made by the system.

5.2 Performance-based evaluation

In addition to the evaluation done above using the statistical measurements; accuracy, precision, recall, and F1 score, further evaluation has been done to evaluate the proposed architecture on performance matrices; such as response time taken for classifying a given dataset and the server resources consumed by the classification process for each classifier.

This kind of performance evaluation is important for a system like a clinical decision support system since the classification or the diagnosis deriving process is a time-critical task. Once the data is inserted into the system, the classification process should start immediately and produce the final result within a short amount of time without much delay. Because every second that would be spending any other task than on actual treatments, may pose a threat to the patient's life, as much the diagnosis delays, the patient has to wait for a diagnosis without getting any proper treatment and in the meanwhile, the patient's health situation may go bad by each second. Because of this time criticality of diagnosis processes, the system managers or hospital information technology service providers are paying much attention to the performance matrices of the system.

Even though the improvement of the performance matrices is out of the scope of this research, the study has evaluated the implemented prototype for performance. The purpose of this performance evaluation is to get an understanding of the performance of the classifiers employed in the system. How each classifier performs in the training process and in the process of classifying instances were compared against each other, using the performance matrices used in this section. Evaluation is done by employing all the classifiers in the same base system, providing the same set of datasets and other input data, and using the same server configurations and resources. The base for the evaluation has been set by doing it, because any external difference may cause noise and affect the experiment. The values for the performance matrices are evaluated using a comparison-based method, not by judging individual values.

Response time evaluation analyzes how much time it takes a system to complete a certain activity. It's computed as the time it takes the system to handle all requests on average.

$$\text{Average response time} = \frac{\text{Sum of time taken by the system to perform an activity}}{\text{Number of activities}}$$

In the scope of CDSS, the time taken by each classifier to train a classifier model for a given dataset, and the time taken to classify a given instance or in other words, to derive a diagnosis for a given set of inputs, by each classifier is measured and compared against each other. These two are the main two activities performed by the CDSS.

Consumption of server resources comparison has been done using the matrices: CPU usage and memory usage when performing those defined activities above. The server resources consumed by each of the classifiers are compared with each other, to get an understanding of how much server resources would be consumed by each classifier.

Performance Monitor (Perf Mon) was used to analyze the server resources. The analysis has used two performance counters: %processor time which has been used to monitor CPU usage and available memory (Mbytes) to monitor memory usage. When a process is triggered, the processor time graph would show significant spikes, and the available memory graph would show a significant drop since the memory has been used up for the process.

The basic server resource configurations that have been used for this evaluation process are as below.
CPU: Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz 2.60 GHz
RAM: 16.0 GB
System type: 4-bit operating system, x64-based processor
Hard disk capacity: 256GB SSD and 1TB hard drive

Averaged values were used for the evaluation. Each activity is done three times for each classifier and the average value is taken as the final value for the activity for that classifier. The average values obtained are organized into a table as follows for result comparison purposes.

The defined two activities were taken separately for evaluation purposes, and response time, CPU usage, and memory usage matrices' values were compared against each classifier.

Activity 1: Train a classifier model for a given dataset

Response time and server resources consumption for training a classifier model for a particular node/ disease, with a given training dataset is compared for classifiers: Naïve Bayes classifier, Random Forest classifier, Support Vector Machine classifier, and the ensemble classifier

The training was done for two nodes: the root node and the heart disease node.
Training for root node: Training is done for 4920 data instances.
Training for heart disease node: Training is done for 250 data instances.
The results observed are presented below separately for each node.

Classifier	Training of the root node (seconds)	Training of the node for heart diseases (seconds)
Naïve Bayes	10.3	1.4
Random Forest	20.4	2.9
Support Vector Machine	12.2	2.1
Ensemble classifier	37.7	3.5

Table 5.2.1: Response time values for the training of a given node

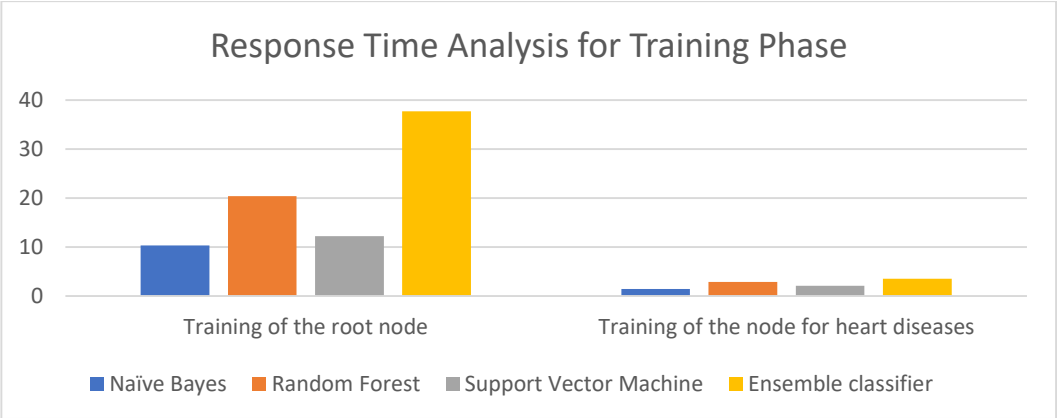


Figure 5.2.1: Graphical representation of response time values for the training of a given node

Figure 5.2.2: Server resource usage by training process by each classifier – Root Node

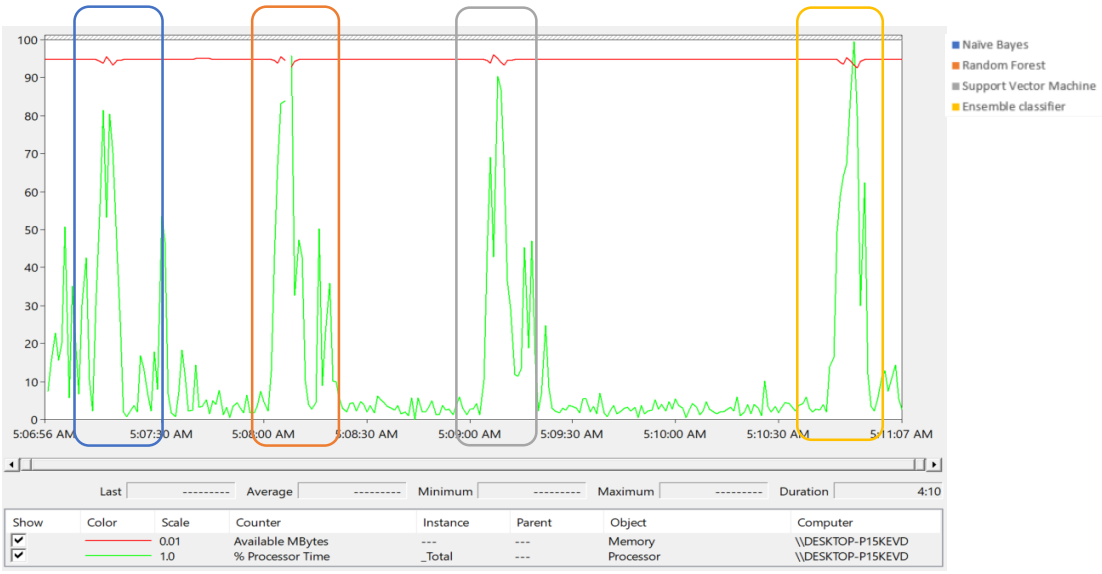


Figure 5.2.3: Server resource usage by training process by each classifier – Node Heart Disease

By analyzing the observations of the evaluation done for the training process for a given node, we can see that the Naïve Bayes classifier shows the highest performance based on the response time matrices. Support Vector Machine classifier is the classifier with the next best response time for the training phase. Random Forest classifier shows a higher response time than both of the other algorithms so that having a low performance compared with the other two algorithms.

The difference in the observed results for the two nodes has been caused by the significant difference between the sizes of the two datasets that have been used as the training datasets. The training dataset that was used for training the root node has a large number of data instances as well as consists of more features/ attributes and target classes/ prognosis. This proves that the size of the dataset directly affects the performance matrices.

The ensemble classifier shows the highest response time for the training phase of both nodes. The reason behind this would be, in the ensemble classifier, it calls the training process of each algorithm, so that the total time for the training of the ensemble classifier includes the response time for training all three individual classifiers.

In the graphs driven for server resource analysis, we can see that even though there are small fluctuations, all the classifiers have used almost the same amount of resources for processing.

Activity 2: Classify a given instance/ derive a diagnosis decision

Response time and server resources consumption for classifying a given instance by the system is compared for classifiers: Naïve Bayes classifier, Random Forest classifier, Support Vector Machine classifier, and the ensemble classifier

Classification is done for 50 instances at a time.
The observed results are presented below.

Classifier	Classify given instances (seconds)
Naïve Bayes	6.2
Random Forest	6.4
Support Vector Machine	6.3
Ensemble classifier	6.8

Table 5.2.2: Response time values for classifying a given instance

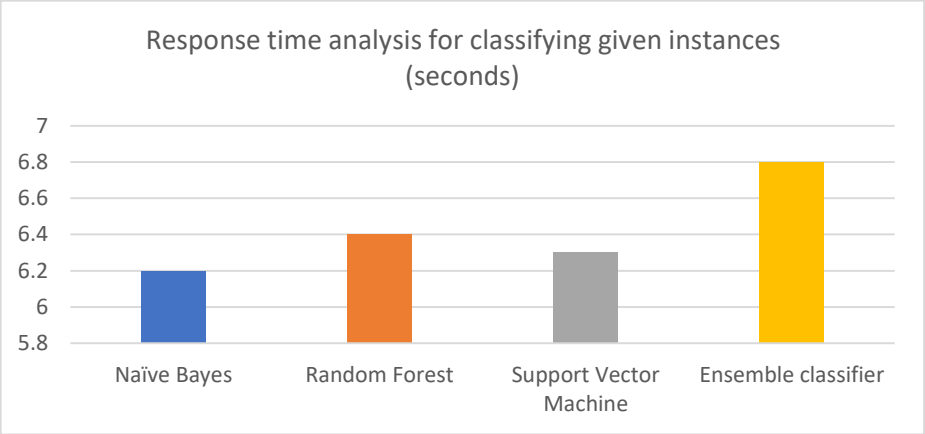


Figure 5.2.4: Graphical representation of response time values for classifying a given instance

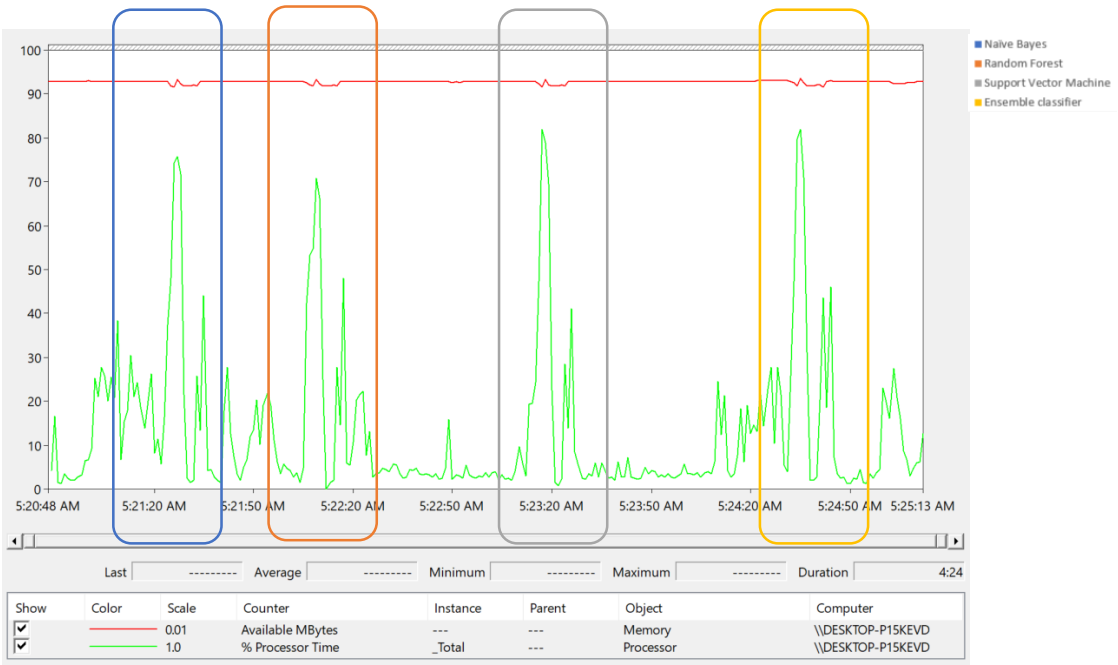


Figure 5.2.5: Server resource usage by the process for classifying instances given by each classifier

The instance classification process follows the same pattern in both response time and server resource consumption, as observed in the training process.

According to the response time analysis, the Naïve Bayes classifier shows the highest performance in the overall system, SVM is the second-best in performance aspects and the Random Forest classifier shows low performance than both other classifiers.

The ensemble classifier shows the lowest performance in the overall system. The direct reason for this is ensemble classifier is an integration of all the classifiers. So that it includes the processing of all integrated algorithms and the calculation on top of that to derive the ensemble output.

When looking at the statistics more closely, we can see that even though the response time of the training process of the ensemble classifier is the highest and nearly the summation of three individual classifiers' response times, it is not the same, but a little lower. This would be because of the additional processing it takes for preparation and concluding the process. Even though the ensemble classifier integrates the classifier processes, it does not have to prepare separately for each classifier.

It is hard to properly demonstrate the advantage of the proposed layered architecture, by implementing only two layers of the disease hierarchy. To better evaluate it, the system should be trained for several layers consisting of several nodes, with a good hierarchy. Then the advantage of the proposed layered architecture can be better demonstrated.
Constructing a better informative dataset and using that dataset to train the system is a future work of this research.

Scalability Analysis

The proposed architecture is facilitating scalability of the system. To support more diseases, relevant nodes should be trained and integrated with the disease hierarchy tree structure. System is horizontally scalable adding more nodes to represent more diseases in a particular level, and by adding more levels in to the hierarchy enables vertical scalability.

To better demonstrate and do a scalability analysis, it is necessary to train the system for more diseases, to increase the levels (depth) and expand the system horizontally by training for more diseases in the same level. Relevant training datasets need to be constructed for this purpose. This is remained as a future work of the research because of unavailability of adequate datasets for expansion of the system at the moment.

The training process of one node does not affect another node or the performance of the training process of another node, since the training for each node is to be performed separately without affecting others. And as observed in above evaluation results, when the size of the dataset increased, the performance of the training process is decreased.

When considering the diagnosis process, when there are more levels in the hierarchy, it will take more time to traverse through the tree, which would result in performance drops. Since the system has been trained for only two nodes as the prototype because of dataset unavailability, this behavior cannot be practically evaluated.

CHAPTER 6

CONCLUSION

Clinical decision support systems facilitate support to clinicians in the diagnosis process by analyzing the provided symptoms and other related information, for better decision-making capacity. Clinical Decision Support Systems provide healthcare professionals with wisdom that can contribute to the betterment of their patients by integrating mainstream scientific scholarship and up-to-date health information and assessing various items of one's data, such as clinical test reports, medical history, medications, demographics, diagnosis findings, to provide case-specific advice.

Senior physicians use their experience as well as expert medical knowledge to derive diagnostic decisions, which is a good knowledge base but specific to each person and hidden inside of their brains. This research has tried to derive this information by identifying the patterns of past clinical data using machine learning techniques so that the knowledge is digitally available outside, making the knowledge accessible to anyone interested to be used in required situations.

6.1 Research contribution

CDSSs are categorized into two types: knowledge-based and non-knowledge-based decision support systems. Knowledge-based decision support systems are implemented as rule-based systems. Rules are created using expert medical knowledge. Non-knowledge-based systems derive decisions using machine learning techniques, artificial intelligence, or using statistical pattern recognition techniques.

This study has proposed an architecture for a non-knowledge-based clinical decision support system employing machine learning techniques, that fulfills quality requirements that a modern CDSS should be adhered to.

It is an evidence-based CDSS, which utilizes machine learning algorithms to derive knowledge from past medical records and identify patterns, then use that knowledge

for the future decision-making process. It is a supervised learning classification mechanism.

The study has proposed a layered knowledge model as the knowledge engine for the clinical decision support system. This is a new way of looking at the CDS systems since all the existing work focuses on a selected disease and derives diagnosis for that selected disease only, which does not provide an effective architecture to cater to the practical situation of handling a large disease hierarchy in the human body.

A prototype has been implemented for the proposed layered knowledge model which is scalable and maintainable. This is an attempt to map the real-world diagnosis process that happens on the side of an actual person, a clinician. Disease hierarchy is organized into a tree structure and a single node of the tree represents a single disease or a disease category. Separate machine learning models are trained for each disease/ node of the tree.

The proposed architecture has succeeded in implementing a clinical decision support system with a service architecture as a system that can be integrated with an EHR system or used as a stand-alone system, supporting quality features such as maintainability, adaptability, and scalability.

An ensemble classifier has been trained for disease classification/ diagnosis support. Naïve Bayes, Random Forest, and Support Vector Machine classifiers are selected for the ensemble classifier because these algorithms have been proven in the literature to produce better results in classification. Priority-based mode method has been used for deriving the final output of the ensemble classifier.

The evaluation was carried out in two main aspects: clinical-diagnosis-result-based evaluation and performance-based evaluation.
For the diagnosis-results-based evaluation, the statistical matrices, accuracy, precision, recall, and F1 score has been calculated based on the classifier results. The ensemble classifier has been compared with the other individual classifiers. Evaluation results

have proven that the proposed ensemble classification mechanism performs better at deriving diagnosis decisions than the individual classifiers. Among the individual classifiers, the Random Forest classifier has shown more effective results than the other two classifiers for heart disease classification.

The performance-based evaluation was done to identify the performance aspects of the system, such as the response time of the system and server resource consumption. Each classifier has been compared against the others for performance and compared with the ensemble classifier. The evaluation results have shown that the Naïve Bayes algorithm performs better than all other algorithms and Support Vector Machines take the second place in performance aspects. The Random Forest classifier has shown low-performance matrices compared to the other two individual classifiers. The ensemble classifier has shown the lowest performance measurements than the individual classifiers. The direct reason for this is, the ensemble classifier is an integration of all the classifiers. So that it includes the processing of all integrated algorithms and the calculation on top of that to derive the ensemble output.
And also the evaluation has proved that the size of the dataset that is being used directly affects the performance matrices.

6.2 Limitations and future work

This study has implemented a prototype of the proposed architecture for evaluation and demonstrating purposes. The implemented prototype had to be limited to two layers represented by two nodes, mainly because of the difficulty of finding acceptable clinical datasets that can be used for the training process. To better evaluate and demonstrate the layered architecture the system should be trained for several layers.

Since this study uses past medical records to derive knowledge, it can really help clinicians in identifying rare diseases, that doesn't come to mind when looking at the symptoms, but witnessed in past. To better aid these situations, it's helpful if the CDSS system is able to provide multiple predictions/ diagnoses along with their probabilities.

Then users can see what are the possible diagnosis for a set of symptoms and the probabilities of them occurring.
The list of probable diagnosis can be used to propagate in the layered knowledge engine architecture, which would result in multiple progress paths and more covered and distributed diagnosis outcomes, but this approach adds much complexity to the system. This is a worthy research area to work on as an extension.

The tree structure that represents the disease hierarchy should be ideally mapped with an international standard like ICD-11, be acceptable and usable internationally, and for better clarity of the structure that has been constructed in the system.

And the features/ keywords representing symptoms and the diseases should be mapped with an international standard like SNOMED CT to standardize the inputs and outputs of the system. This is called concept mapping or terminology binding. All the keywords that would be used in the system are to be mapped with terminologies or concepts, then a particular keyword represents the same meaning internationally. All the input and output data would be represented by specific keywords which have to be mapped with terminology. Input data validation can be done at the interface level when terminology binding has been done in the system.

And the feature matrices used for this study may not be the ideal features to diagnose the selected diseases. A better feature matrix should be constructed for each disease or disease category, representing each node of the tree hierarchy.

Only correlation technique is used as the feature selection mechanisms in the prototype implementation. Implementing more feature selection techniques and integrating them in the training process for better outcome, is remained as a future work or as an extension of this study. Best feature selection technique each node can be defined based on the performance measures.
For feature selection, this study used a simple feature selection technique since it is not the focus of this research. Better feature selection techniques can be researched

further to identify the most important features to provide feedback to the users on necessary clinical tests.

Creating a better clinical dataset is an important task for future work. Using a better informative dataset would be minimized most of the limitations explained above.

Alert generation and prioritization are also nice to have features in a CDSS system, that can be continued under this research. Implementation of a risk factor can be done to prioritize the alerts.

Explainable AI is an emerging concept that would enable a very important feature for a clinical decision support system. Just providing the final output of a classifier is not enough for the medical domain, since the users are not blindly accepting a result with no base or explanation. To reduce the distrust and acceptability of the system, it is important to give users an explanation of how the system is generating the decisions and the basis that has been used for the decision-making process. It generates details or justifications to make its operation plain or simple to comprehend. The advancements in this discipline will eventually lead to a perfect combination of interpretability and performance in machine learning models.
The importance parameters of the Random Forest algorithm can be utilized to further enhance the second part of the rationale provided, which is the list of the most crucial features for each node. This can be achieved by using the Random Forest algorithm's built-in functionality.
This feature has the capacity for further improvement which requires more research. It is a good research area to be continued further.

Security is also a very important quality characteristic of a clinical decision support system, as it is directly dealing with patient information. This is an area that was scoped out from this research, but a good area of research for future work.

REFERENCES

[1] J. A. Kim, I. Cho and Y. Kim, "CDSS (Clinical Decision Support System) Architecture in Korea," 2008 International Conference on Convergence and Hybrid Information Technology, 2008, pp. 700-703, doi: 10.1109/ICHIT.2008.223.

[2] A. Wright and D. Sittig, "SANDS: A service-oriented architecture for clinical decision support in a National Health Information Network", *Journal of Biomedical Informatics*, vol. 41, no. 6, pp. 962-981, 2008. Available: 10.1016/j.jbi.2008.03.001.

[3] A. Wright and D. Sittig, "A four-phase model of the evolution of clinical decision support architectures", *International Journal of Medical Informatics*, vol. 77, no. 10, pp. 641-649, 2008. Available: 10.1016/j.ijmedinf.2008.01.004.

[4] M. Afzal, M. Hussain, W. A. Khan, T. Ali, S. Lee and B. H. Kang, "KnowledgeButton: An evidence adaptive tool for CDSS and clinical research," 2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings, 2014, pp. 273-280, doi: 10.1109/INISTA.2014.6873630.

[5] Y. Y. Chen, K. N. Goh and K. Chong, "Rule based clinical decision support system for hematological disorder," 2013 IEEE 4th International Conference on Software Engineering and Service Science, 2013, pp. 43-48, doi: 10.1109/ICSESS.2013.6615252.

[6] X. Xiang, Z. Wang, Y. Jia and B. Fang, "Knowledge Graph-Based Clinical Decision Support System Reasoning: A Survey," 2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC), 2019, pp. 373-380, doi: 10.1109/DSC.2019.00063.

[7] X. Liu, X. Zhan and L. Xiao, "An approach towards automatic generation of evidence-based decision support systems for clinical diagnosis based on an extensive clinical guideline schema," 2013 IEEE Third International Conference on Information

Science and Technology (ICIST), 2013, pp. 672-676, doi: 10.1109/ICIST.2013.6747635.

[8] J. Wang, "A Service-Oriented Architecture for Integrating Clinical Decision Support in a National E-Health System", 2011.

[9] D. E. Robbins, V. P. Gurupur and J. Tanik, "Information architecture of a clinical decision support system," 2011 Proceedings of IEEE Southeastcon, 2011, pp. 374-378, doi: 10.1109/SECON.2011.5752969.

[10] L. Tabares, J. Hernandez and I. Cabezas, "Architectural Approaches for Implementing Clinical Decision Support Systems in Cloud: A Systematic Review," 2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2016, pp. 42-47, doi: 10.1109/CHASE.2016.55.

[11] G. Purcell, "What makes a good clinical decision support system", *BMJ*, vol. 330, no. 7494, pp. 740-741, 2005. Available: 10.1136/bmj.330.7494.740.

[12] K. Kawamoto, C. Houlihan, E. Balas and D. Lobach, "Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success", *BMJ*, vol. 330, no. 7494, p. 765, 2005. Available: 10.1136/bmj.38398.500764.8f.

[13] S. El-Sappagh and S. El-Masri, "A Proposal of Clinical Decision Support system Architecture for Distributed Electronic Health Records".

[14] Liang Xiao, G. Cousins, T. Fahey, B. D. Dimitrov and L. Hederman, "Developing a rule-driven clinical decision support system with an extensive and adaptative architecture," 2012 IEEE 14th International Conference on e-Health Networking, Applications and Services (Healthcom), 2012, pp. 250-254, doi: 10.1109/HealthCom.2012.6379416.

[15] J. Kannry, L. McCullagh, A. Kushniruk, D. Mann, D. Edonyabo and T. McGinn, "A Framework for Usable and Effective Clinical Decision Support: Experience from

the iCPR Randomized Clinical Trial", *eGEMs (Generating Evidence & Methods to improve patient outcomes)*, vol. 3, no. 2, p. 10, 2015. Available: 10.13063/2327-9214.1150.

[16] R. van de Wetering, "IT-Enabled Clinical Decision Support: An Empirical Study on Antecedents and Mechanisms", *Journal of Healthcare Engineering*, vol. 2018, pp. 1-10, 2018. Available: 10.1155/2018/6945498.

[17] E. Murphy, "Clinical decision support: effectiveness in improving quality processes and clinical outcomes and factors that may influence success", *The Yale journal of biology and medicine*, vol. 872, pp. 187-197, 2014. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4031792/.

[18] D. Arnott and G. Pervan, "A Critical Analysis of Decision Support Systems Research Revisited: The Rise of Design Science", *Journal of Information Technology*, vol. 29, no. 4, pp. 269-293, 2014. Available: 10.1057/jit.2014.16.

[19] Y. Jiang, B. Qiu, C. Xu and C. Li, "The Research of Clinical Decision Support System Based on Three-Layer Knowledge Base Model", *Journal of Healthcare Engineering*, vol. 2017, pp. 1-8, 2017. Available: 10.1155/2017/6535286.

[20] M. Alavi and E. Joachimsthaler, "Revisiting DSS Implementation Research: A Meta-Analysis of the Literature and Suggestions for Researchers", *MIS Quarterly*, vol. 16, no. 1, p. 95, 1992. Available: 10.2307/249703.

[21] T. Kosaka and T. Hirouchi, "An effective architecture for Decision Support Systems", *Information & Management*, vol. 5, no. 1, pp. 7-17, 1982. Available: 10.1016/0378-7206(82)90014-3.

[22] V. Korde, "Text Classification and Classifiers:A Survey", *International Journal of Artificial Intelligence & Applications*, vol. 3, no. 2, pp. 85-99, 2012. Available: 10.5121/ijaia.2012.3208.

[23] S. M. H. Dadgar, M. S. Araghi and M. M. Farahani, "A novel text mining approach based on TF-IDF and Support Vector Machine for news classification," 2016

IEEE International Conference on Engineering and Technology (ICETECH), 2016, pp. 112-116, doi: 10.1109/ICETECH.2016.7569223.

[24] E. Ikonomakis, S. Kotsiantis and V. Tampakas, "Text Classification Using Machine Learning Techniques", *WSEAS Transactions on Computers*, vol. 48, pp. 966-974, 2005.

[25] R. Cretulescu, D. Morariu and L. Vintan, "Ongoing Research in Document Classification at the "Lucian Blaga" University of Sibiu", *5th International Symposium on Intelligent Distributed Computing IDC2011*, vol. 52011, 2011, , , 2011.

[26] M. Gogoi and S. Sarma, "Document Classification of Assamese Text Using Naïve Bayes Approach", *International Journal of Computer Trends and Technology*, vol. 30, no. 4, pp. 182-186, 2015. Available: 10.14445/22312803/ijctt-v30p132.

[27] B. P. C. Lim, M. H. Tsui, V. Charastrakul and D. Shi, "Web Search with Text Categorization Using Probabilistic Framework of SVM," 2006 IEEE International Conference on Systems, Man and Cybernetics, 2006, pp. 2950-2955, doi: 10.1109/ICSMC.2006.384566.

[28] B. Zhang, J. Su and X. Xu, "A Class-Incremental Learning Method for Multi-Class Support Vector Machines in Text Classification," 2006 International Conference on Machine Learning and Cybernetics, 2006, pp. 2581-2585, doi: 10.1109/ICMLC.2006.258853.

[29] F. Harrag, E. El-Qawasmeh and P. Pichappan, "Improving arabic text categorization using decision trees," 2009 First International Conference on Networked Digital Technologies, 2009, pp. 110-115, doi: 10.1109/NDT.2009.5272214.

[30] L. Breiman, "Random Forests", *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001. Available: 10.1023/a:1010933404324.

[31] Wyner et al., "Explaining the Success of AdaBoost and Random Forests as Interpolating Classifiers.", Journal of Machine Learning Research, vol. 18, 2015.

[32] R. CASTILLO and A. KELEMEN, "Considerations for a Successful Clinical Decision Support System", *CIN: Computers, Informatics, Nursing*, vol. 31, no. 7, pp. 319-326, 2013. Available: 10.1097/nxn.0b013e3182997a9c.

[33] R. Sutton, D. Pincock, D. Baumgart, D. Sadowski, R. Fedorak and K. Kroeker, "An overview of clinical decision support systems: benefits, risks, and strategies for success", *npj Digital Medicine*, vol. 3, no. 1, 2020. Available: 10.1038/s41746-020-0221-y.

[34] "Clinical Decision Support Systems: How They Improve Care and Cut Costs", *AltexSoft*, 2022. [Online]. Available: https://www.altexsoft.com/blog/clinical-decision-support-systems/#:~:text=Integrated%20in%20a%20CPOE%20system,cases%20of%20excessive%20medical%20testing.

[35] D. Sittig et al., "Grand challenges in clinical decision support", *Journal of Biomedical Informatics*, vol. 41, no. 2, pp. 387-392, 2008. Available: 10.1016/j.jbi.2007.09.003

[36] A. Garg et al., "Effects of Computerized Clinical Decision Support Systems on Practitioner Performance and Patient Outcomes", *JAMA*, vol. 293, no. 10, p. 1223, 2005. Available: 10.1001/jama.293.10.1223

[37] M. Alam, M. Rahman and M. Rahman, "A Random Forest based predictor for medical data classification using feature ranking", *Informatics in Medicine Unlocked*, vol. 15, p. 100180, 2019. Available: 10.1016/j.imu.2019.100180.

[38] M. Bernardini, "Machine Learning approaches in Predictive Medicine using Electronic Health Records data", *Artificial Intelligence [cs.AI]. Università Politecnica delle Marche*, 2021.

[39] J. Kobylarz et al., "A Machine Learning Early Warning System: Multicenter Validation in Brazilian Hospitals", 2020.

[40] S. Rathi, M. Motwani and M. Ahirwar, "Data-Driven Clinical Decision Support System for Medical Diagnosis and Treatment Recommendation", *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 11, pp. 3660-3668, 2019. Available: 10.35940/ijitee.k1950.0981119.

[41] M. Kamel Boulos and P. Zhang, "Digital Twins: From Personalised Medicine to Precision Public Health", *Journal of Personalized Medicine*, vol. 11, no. 8, p. 745, 2021. Available: 10.3390/jpm11080745.

[42] S. Lundberg and S. Lee, "A unified approach to interpreting model predictions", *Advances in neural information processing systems*, pp. 4765–4774, 2017.

[43] A. Altmann, L. Toloşi, O. Sander and T. Lengauer, "Permutation importance: a corrected feature importance measure", *Bioinformatics*, vol. 26, no. 10, pp. 1340-1347, 2010. Available: 10.1093/bioinformatics/btq134.

[44] S. Ahalt et al., "Clinical Data: Sources and Types, Regulatory Constraints, Applications", *Clinical and Translational Science*, vol. 12, no. 4, pp. 329-333, 2019. Available: 10.1111/cts.12638.

[45] Y. Jiang, B. Qiu, C. Xu and C. Li, "The Research of Clinical Decision Support System Based on Three-Layer Knowledge Base Model", *Journal of Healthcare Engineering*, vol. 2017, pp. 1-8, 2017. Available: 10.1155/2017/6535286.

[46] Y. Y. Chen, K. N. Goh and K. Chong, "Rule based clinical decision support system for hematological disorder," 2013 IEEE 4th International Conference on Software Engineering and Service Science, 2013, pp. 43-48, doi: 10.1109/ICSESS.2013.6615252.

[47] Y. Kumar and G. Sahoo, "Prediction of different types of liver diseases using rule based classification model", *Technology and Health Care*, vol. 21, no. 5, pp. 417-432, 2013. Available: 10.3233/thc-130742.

[48] J. Singh, S. Bagga and R. Kaur, "Software-based Prediction of Liver Disease with Feature Selection and Classification Techniques", *Procedia Computer Science*, vol. 167, pp. 1970-1980, 2020. Available: 10.1016/j.procs.2020.03.226.

[49] E. Capobianco, "Data-driven clinical decision processes: it's time", *Journal of Translational Medicine*, vol. 17, no. 1, 2019. Available: 10.1186/s12967-019-1795-5.

[50] O. Sagi and L. Rokach, "Explainable decision forest: Transforming a decision forest into an interpretable tree", *Information Fusion*, vol. 61, pp. 124-138, 2020. Available: 10.1016/j.inffus.2020.03.013.

[51] D. Petkovic, R. Altman, M. Wong and A. Vigil, "Improving the explainability of Random Forest classifier – user-centered approach", *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, vol. 23, pp. 204–215, 2018.

[52] M. Siraj-Ud-Doulah and M. Alam, "PERFORMANCE EVALUATION OF MACHINE LEARNING ALGORITHMS IN ECOLOGICAL DATASET", *International Journal of Applied Mathematics and Machine Learning*, vol. 10, no. 1, pp. 15-45, 2020. Available: 10.18642/ijamml_7100122032.

[53] M. Bal, M. Amasyali, H. Sever, G. Kose and A. Demirhan, "Performance Evaluation of the Machine Learning Algorithms Used in Inference Mechanism of a Medical Decision Support System", *The Scientific World Journal*, vol. 2014, pp. 1-15, 2014. Available: 10.1155/2014/137896.

[54] V. Belle and I. Papantonis, "Principles and Practice of Explainable Machine Learning", *Frontiers in Big Data*, vol. 4, 2021. Available: 10.3389/fdata.2021.688969.

[55] R. Studer, V. Benjamins and D. Fensel, "Knowledge engineering: Principles and methods", *Data & Knowledge Engineering*, vol. 25, no. 1-2, pp. 161-197, 1998. Available: 10.1016/s0169-023x(97)00056-6.

[56] D. Rubins et al., "Importance of clinical decision support system response time monitoring: a case report", *Journal of the American Medical Informatics Association*, vol. 26, no. 11, pp. 1375-1378, 2019. Available: 10.1093/jamia/ocz133.