Investigation of a Data and Knowledge Driven System for Sequential Diagnosis

by

Diyang Xue

B.S., Shandong University, China, 2006

M.S., Peking University, China, 2009

M.S., University of Pittsburgh, 2015

Submitted to the Graduate Faculty of

the School of Computing and Information in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2022

UNIVERSITY OF PITTSBURGH

SCHOOL OF COMPUTING AND INFORMATION

This dissertation was presented

by

Diyang Xue

It was defended on

November 14, 2022

and approved by

Dr. Daqing He, Professor, Intelligent Systems Program and Department of Informatics and Networked Systems, University of Pittsburgh

Dr. Gregory F. Cooper, Professor, Intelligent Systems Program and Department of Biomedical Informatics, University of Pittsburgh

Dr. Michael M. Wagner, Professor, Intelligent Systems Program and Department of Biomedical Informatics, University of Pittsburgh

Dr. Adam N. Frisch, Assistant Professor, Department of Emergency Medicine, University of Pittsburgh

Dissertation Director: Dr. Daqing He, Professor, Intelligent Systems Program and Department of Informatics and Networked Systems, University of Pittsburgh

Investigation of a Data and Knowledge Driven System for Sequential Diagnosis

Diyang Xue, PhD

University of Pittsburgh, 2022

Medical diagnosis is the process of determining the nature of a disease and distinguishing it from other similar diseases. A diagnostic error happens when a diagnosis is missed, inappropriately delayed, or inaccurate. Diagnostic error accounts for the most severe patient harm, the largest fraction of claims, and highest total penalty payouts. One way to reduce diagnostic error is to use a computer-aided diagnostic (CAD) system to augment doctors' diagnostic abilities. More and more machine learning algorithms have been applied to the medical diagnosis field and achieve good performance. However, because most of the models are very complicated and the diagnostic process is different from physicians' workflow, physicians usually do not trust those models.

My dissertation investigates how to combine electronic health record (EHR) data with medical knowledge to generate a sequential diagnostic system that utilizes clinical alignment, which is when the diagnostic process is in line with physicians' diagnostic process. The new system has two main characteristics: (1) data-driven so that we can use EHR data and machine learning algorithms for developing a multi-label classification system; (2) clinical knowledge-driven so that valuable clinical diagnostic knowledge can be integrated into the system.

I have developed (1) a framework that can integrate pre-defined medical knowledge with disease patterns in EHR data for sequential diagnosis and (2) an algorithm that generates medical diagnostic trees that recommend diagnostic actions by considering clinical workflow, diagnostic accuracy, and misdiagnosis costs. Experiments show that the learned model has better clinical alignment, higher diagnostic accuracy, and lower misdiagnosis costs than baseline models, which were developed using a traditional multi-label classification tree algorithm (ML-C4.5) and a deep reinforcement learning algorithm (deep Q learning), respectively.

Table of Contents

List of Tables

List of Figures

Preface

Each doctoral student is different: they have different family backgrounds, different educational backgrounds, and different levels of maturity. Therefore, each doctoral student's path is unique; Perhaps, finishing a PhD degree is one of the hardest roads in my life, but I think I also have reaped the most rewards. Standing near the finish line and looking back, I am very grateful.

I would like to acknowledge the guidance of my committee professors. I am very lucky to have Professor Daqing He as my advisor. He kindly gave me the precious opportunity to earn a doctoral degree. Under his rigorous guidance on research, I finally completed this doctoral dissertation and achieved a conclusion to my doctoral study. I will never forget his efforts on my behalf. I am very grateful to Professor Gregory F. Cooper for sharing his biomedical informatics research expertise with me. I am also very grateful to Professor Michael M. Wagner, who gave me a lot of good advice on my thesis from a research perspective and an academic writing perspective. I would like to express my appreciation to Professor Adam N. Frisch, who supplied the professional medical knowledge for my research. My dissertation research cannot be finished without the efforts of Professor Randolph A. Miller, who has kindly provided me access to the QMR knowledge base and shared QMR simulation data. Many thanks to Dr. Marek J. Druzdzel and BayesFusion LLC. for providing a copy of the PySMILE package with a free academic license.

In addition, I would like to thank Professor Xia Jiang, who offered me a graduate student researcher position and published several articles with me. I would like to thank Professor Fuchiang (Rich) Tsui. With his efforts and support, I started my study of clinical informatics. I am very grateful to Professor Janyce Wiebe, who taught the AI course and kindly provided a recommendation letter to support my application. Many thanks to Ms. Michele Thomas and Ms. Toni Porterfield for their support, Ms. Cleat Szczepaniak for her encouragement, Ms. Janine Carlock for helpful revision comments. Also, I would like to thank my friends and classmates: Kevin Bui, Binghuang Cai, Lingyun Shi, Fan Mi, Victor Ruiz, Amie Barda, Jose David Posada Aguilar, Rui Meng, Ning Zou, Mengdi Wang, Khushboo Thaker, Fanghui Xiao,

Zhendong Wang, Zhimeng Luo, Yu Chi, Sanqiang Zhao, Danchen Zhang, Yingyi Zhang, Jing Dong, Jingjing Zhang, and Shaobo Liang. We studied together and grew together, which experience has provided me precious memories.

Finally, I would like to acknowledge the unceasing support of my family. My father, Xiaoyong Xue and my mother, Xiaofang Jin, they do not have a high degree of education and don't have a lot of experience in urban life, yet they gave me unconditional support and encouragement throughout this difficult journey. I am also very grateful to my younger brother Diliang Xue, who took good care of my parents when I was not with them. I would like to thank my wife Ye Ye, who has walked with me on this path. She has given me great help – both spiritual and material – and gave birth to our daughter Jane in 2017. I also would like to thank my in-laws, Zhonghua Ye and Yanghua Cheng, who have been taking care of Jane.

1.0 Introduction

More and more machine learning algorithms have been applied into the medical diagnosis field and have gotten good performance. However, because most models for medical diagnosis are complicated and their recommended diagnostic process is largely different from physicians' workflow, physicians usually do not trust them [36]. This dissertation investigates how to combine EHR data with medical knowledge to generate a sequential diagnostic system that is, unlike models currently in use, in clinical alignment. Clinical alignment means the diagnostic process is in line with physicians' diagnostic process. The new system will have two main features: (1) it will be data-driven so that we can use EHR data and machine learning algorithms for developing a multi-label diagnostic system that can handle diagnosis for patients who have multiple diseases; (2) it will be clinical-knowledge-driven so that valuable clinical diagnostic knowledge can be integrated into machine learning algorithm to guide the model building process and make the model clinical alignment. This dissertation research project has three main goals: (1) achieve medical knowledge retrieval, representation, and integration, (2) build a multi-label sequential classification tree system that is in clinical alignment for medical diagnosis, (3) evaluate the proposed system by applying it in real-world heart disease diagnosis tasks. This chapter provides research background, presents the research questions, and defines the scope of this dissertation.

1.1 Overview

Our medical knowledge has increased significantly during the last century. For example, *"The Merck Manual of Diagnosis and Therapy"* is the best-selling medical textbook in the world and the oldest continuously published textbook, demonstrates the growth. Its first edition, published in 1899, only had 192 pages [111], while the current edition (the 20th edition) published in 2018 has 3530 pages [77]. An urgent problem, however, has emerged as a result of this large increase in knowledge: individual physicians cannot hold in their minds

so much knowledge, which makes it difficult at times to make a correct diagnosis.

Medical diagnosis is the process of determining the nature of a disease and distinguishing it from other similar diseases. A diagnostic error happens when a diagnosis is missed, inappropriately delayed, or is wrong [117]. diagnostic errors account for the most severe patient harm, the largest fraction of medical claims and highest total penalty payouts. For example, a recent study estimated that the rate of outpatient diagnostic errors is about 5%, or approximately 12 million US adults every year [92]. Moreover, in a Harvard Medical Practice study [48], researchers found that diagnostic errors accounted for 17% of preventable errors in hospitalized patients. Another 1986-2010 study of malpractice claims assessed that the financial consequences of diagnostic errors were $38.8 billion among all paid claims. Diagnostic error was the leading type of error (28.6%) and accounted for the highest proportion of total payments (35.2%) [96]. Unfortunately, reducing diagnostic errors is not easy.

In general, we have three ways to reduce diagnostic errors: the first way is to improve clinicians' own diagnostic abilities; the second way is to improve laboratory tests for diagnosing diseases more thoroughly and accurately; and the third way is to use a computer-aided diagnosis (CAD) system to augment doctors' diagnostic abilities [19]. The latter has been shown to be capable of reducing instances of delayed diagnosis and of improving diagnostic accuracy [1]. The development of a CAD system depends on many factors, such as the availability of large amounts of clinical data, the existence of rich medical knowledge, and access to new advances in artificial intelligence technology.

CAD systems can be roughly divided into two types: knowledge-driven systems and data-driven systems [70]. In knowledge-driven systems, knowledge is predominant, and such systems require scores and logic rules in reasoning to derive new knowledge. The diagnostic process of these systems is in line with clinical rules. However, these systems are limited in that they are labor-intensive and difficult to adapt to add new diseases.

Data-driven systems, on the other hand, are data dominant. These systems learn diagnostic rules from data directly, so they can be extended to diagnose new diseases very easily. However, there are three main drawbacks to these systems. First, to make final diagnoses, most of these systems must use complicated machine-learned mathematical models, which are difficult for physicians to understand; thus they are perceived to be "black-box" systems.

Physicians usually do not trust the final decisions if the system cannot provide a clinically sound explanation [36]. A second drawback is that even when some mathematical models are easy to understand, they may not be in clinical alignment. A third drawback is that while most data-driven CAD systems provide a list of candidate diagnoses, they cannot provide further suggestions about how to differentiate these diseases and reach a final diagnosis.

All these challenges motivated us to develop a machine learning algorithm to combine patients' medical data and existing medical knowledge to build a white-box computational sequential diagnostic system. This system could help physicians find the most valuable differential diagnosis questions to get a final primary diagnosis, and the order of these questions makes medical sense, being in alignment with the clinical decision-making process.

1.2 Research Background

1.2.1 Computer Application in Disease Diagnosis

When the first digital computers were developed in the 1940s [53], people were told that these new machines could be used for calculation and information retrieval. Later, people found that the computer has several capabilities ideally fitted to medical diagnosis [31]: they can store large quantities of data over a long period of time, recall data exactly as stored, perform complex calculations at very high speed, and display possible diagnoses from the most likely to the least likely. Within 10 years of their first appearance, physicians and other healthcare workers had begun to use computers to solve medical problems [49]. For example, in the late 1950s, a few physicians at the New York and Mt. Sinai Hospital set up a program for differential diagnosis of hematological diseases. They recorded 30 diseases and 98 related symptoms on a magnetic tape. To make diagnosis for a new case, the symptoms of that case were recorded on a tape and compared with the symptoms for the 30 diseases. Based on this comparison, the computer printed out a list of diseases that have the same symptoms with the new case, symptoms that might support the 30 diseases' final diagnosis as well as untested symptoms that are worthy checking [118].

Early CAD systems from the 1970s and 1980s were often referred to as "expert systems in medicine." They used medical knowledge from medical textbooks and flow-chart guidelines [87], statistical pattern-matching theories [83], or probability theories [32] to mimic diagnostic process. Some famous examples of such systems include the MYCIN expert system [90], the INTERNIST-1 expert system [58] and the CADUCEUS expert system [22].

In a typical classic expert system, a user interacts with a knowledge base via the inference engine, and the inference engine can use a forward or backward chaining approach to get a decision/recommendation. The knowledge base is composed of a set of facts and rules. Usually, these rules are in a logical structure of the IF-THEN form.

The advantage of these knowledge-driven systems is that they are easy to understand and apply. They align well with clinicians' reasoning process. The disadvantage of these systems is that the development of the knowledge base is highly labor-intensive, so it is not easy to adapt the system to include new diseases.

As more and more EHR data have become available, various data-driven CAD systems have been developed to help physicians make diagnosis. The architecture of a classic data-driven CAD system can be found in Figure 1. The whole data set is split into training data and test data, where the training data is used to develop models, which are then used on test data to conduct diagnosis. This type of system can be used in different aspects of disease diagnosis, including as heart sound and signal analysis system [64], medical image analysis system [93], and lab test analysis system [55].

The advantages of the data-driven systems are that they are easy to add new diseases. The disadvantages of the data-driven systems are: (1) their models are often complicated (black-box systems), which may not be easily understood by physicians. (2) Most systems only focus on disease diagnosis and do not consider disease diagnosis process. (3) Model generation processes mostly rely on mathematical criterion, so the developed models may not be clinically aligned and clinically actionable. (4) Most systems assume that one patient only has one disease, while in reality one patient usually has multiple diseases at the same time.

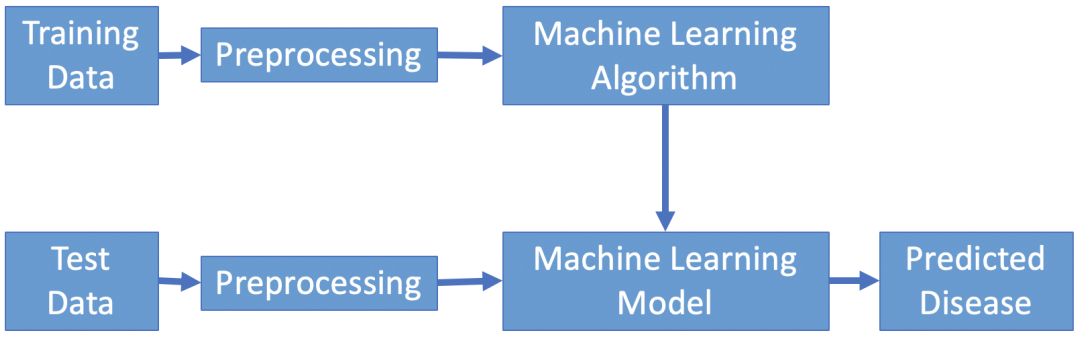

Figure 1: Architecture of a Classic Data-Driven Computational Diagnostic System

1.2.2 Characteristics of Emergency Care

Each year, there are approximately 136 million visits to emergency departments (ED) in the United States [39], accounting for 11% of all outpatient visits [74]. Nearly half of intensive care unit admissions and 25% of surgical intensive care unit admissions are firstly treated in ED [44]. The most surprising thing is that (excluding births) more patients are admitted into hospital through ED than any other routes [61]. This means ED is often the front door of hospital.

Emergency care research is defined as "research that focuses on the discovery and application of time-critical diagnosis, decision-making and treatments that save lives, prevent or reduce disability, and restore human health" [42]. Emergency care has several unique characteristics [10]. For example, ED is the front door of the sickest and worst patients; emergency medical care is the only medical care that emphasizes both immediacy and universality of service; emergency medicine needs to interact with all medical departments in the hospital; emergency physicians must be able to do rapid risk stratification and diagnosis. Briefly, three key factors impacting outcomes of emergency care are: Severity (life-threatening illness) and Time sensitivity (physicians must do diagnosis and treatment in minutes to hours). Lastly, the emergency medicine environment is highly pressurized, immediate, emotional, and often overburdened.

1.2.3 The Demand for a Computational Sequential Diagnostic System in Emergency Care

The common workflow of medical practice is evaluating a patient's signs and symptoms, making diagnosis, and providing treatments. After retrieving a patient's presenting symptoms, medical history, and reviewing results from physical examinations, physicians often create a list of suspected diagnoses (possible conditions or diseases that could be causing the patient's symptoms based on the known information) and order some lab tests to get a final diagnosis. The accuracy of a final diagnosis depends on the physician's experience, the ability to consider the relevant possible diagnoses, and the environment in which the diagnosis is made. And for the same disease, different physicians may choose different diagnostic strategies based on their own experience [95]. Some strategies are good, while some strategies are bad. Physicians in ED must make correct diagnoses in a relatively short time, often with only partial clinical information (e.g., no lab result returns in a short time).

A common situation in ED is that symptoms of patients may be caused by a disorder in any of the several organ systems, which could lead to major irreversible organ injury in a short time if the disorder is undiagnosed or improperly treated. For example, chest pain (one of the most common principal reasons for visiting an ED) [43] may be due to heart disease, pulmonary disease, emotional disease, neural disease, or gastrointestinal disease. Differentiating these diseases and making the correct diagnosis for a patient requires ED physicians to remember medical knowledge related to all organ systems and make the correct diagnosis in a very short time. This situation has created a strong demand for computer-aided diagnostic systems in ED [73].

1.3 Research Motivation and Aims

Since both knowledge-driven systems and data-driven systems have their own advantages, could we combine them together to build a better system that have all the advantages? The goals of this study are to combine the strengths of knowledge-driven systems and data-driven

systems, consider the situation that a patient may have multiple diseases at the same time, use expected misdiagnosis costs as the core criterion, consider both diagnostic accuracy and diagnosis process, and establish a white-box computer-aided diagnostic system.

Two types of algorithms that consider both predictions results and the prediction process are classification trees and reinforcement learning. The developed model of the classification tree algorithm is very straightforward: the leaf nodes can be used to do prediction; and the path from root node to leaf nodes can be used as the prediction process. The reinforcement learning algorithm, on the other hand, has a reward prediction model that is very complicated, but the model can generate a policy for each patient. The policy is similar to the classification tree path. It contains features, the order of features, and the prediction results. Because classification tree model is very straightforward, physicians can understand the model easily. It is a white-box model. Therefore, I choose the classification tree algorithm as the core algorithm in this study. I will compare our classification tree model's performance with the performance of one implementation of reinforcement learning.

There have been very few studies on how to find useful medical domain knowledge, how to transform domain knowledge so that it can be used by machine learning algorithms, or how to add domain knowledge into a pure data-driven computational diagnostic system so that its diagnostic process can make medical sense. Our research aims to fill this void. The goals of this dissertation are (1) to develop a framework that can integrate expert-defined medical knowledge with disease patterns in EHR data for sequential diagnosis; and (2) to develop an algorithm that generates medical classification trees, which recommend diagnostic actions by taking into account clinical workflow, diagnostic accuracy, and misdiagnosis costs simultaneously.

Furthermore, decision processes in the medical field are correct not only because they can arrive at the correct final decisions, but also because the rules that lead to the final decision are in clinical alignment. Therefore, this study evaluated models' performance from two angles: model prediction accuracy and model clinical alignment. The detailed definition of clinical alignment for research Aim 1 can be found in hypothesis 1b in Section 4.2. The detailed definition of clinical alignment for research Aim 2 can be found in hypothesis 2b in Section 4.3.

Research Aim 1. Integrate Expert-Defined Medical Knowledge with EHR Data to Perform Disease Diagnoses

Combining medical knowledge with EHR data is important. The development of an expert-defined medical diagnostic system usually requires a lot of resources. This type of system also may not completely cover all possible complicated medical scenarios and may not accurately capture all uncertain characteristics of medical diagnostic processes. Moreover, retrieving and adding new rules to this type of system can also be tedious. Therefore, the hope is that the knowledge base of an expert-defined diagnostic system can be automatically updated by using routinely collected EHR. On the other hand, while pure machine-developed models (by mining EHR) can get good diagnostic performance, they are limited by the quality and quantity of the training data.

The task of combining medical knowledge with EHR itself is not trivial. Medical knowledge is usually stored in many different formats, such as medical textbooks, peer-reviewed medical articles (e.g., PubMed repository), and medical professional websites (e.g., upToDate). It is not easy to compile all this knowledge and represent them in a coherent way that can be easily integrated with medical data.

I developed a machine learning framework that can integrate expert-defined medical knowledge with EHR data for disease diagnosis. In this dissertation, I only focus on the medical knowledge that has been well represented in an expert-defined diagnostic system. I used the QMR system as an example [57]. I tested the following four hypotheses:

- 1a: When EHR sample size is small, models trained using EHR data only are less accurate than models trained using small sample size EHR data and medical knowledge.

- 1b: When EHR sample size is small, models trained using EHR data only are less clinically aligning than models trained using small sample size EHR data and medical knowledge.

- 1c: When EHR sample size is large, models trained using EHR data only perform similarly to models trained using large sample size EHR data and medical knowledge.

- 1d: When EHR sample size is large, models trained using EHR data only are less clinically aligning than models trained using large sample size EHR data and medical knowledge.

Research Aim 2. Build a Sequential Diagnostic Model that Considers Diagnostic Accuracy, Clinical Workflow, and Misdiagnosis Costs

After obtaining disease diagnostic models, the next step is to use these models to find sequential steps to reach a final diagnosis. When using machine-learned models for clinical decision support, the real-world needs are often multi-dimensional:

1. Because the diagnostic process is actually a sequential decision-making process, a diagnostic system needs to provide users with information about the best next action (e.g., clinical questions to be asked, lab test to be ordered).
2. The suggested diagnosis should have good accuracy.
3. Misdiagnosis costs should be considered in addition to general accuracy. Because not all diseases are equally important, the diseases that have severe clinical outcomes (e.g., death) should be firstly ruled out.
4. The diagnostic action sequence must align with the existing clinical workflow. Because only when physicians think the action sequence makes sense, they can use this system and trust the diagnosis results.

In this dissertation, I developed a machine learning algorithm, Sequential Diagnosis Generating Algorithm (SDG) that have relatively high accuracy and low diagnosis costs. Real-world diagnostic costs involve many complicated scenarios, for example, cost in terms of money, time, emotion, and so on. I ignored these costs in this dissertation and used physician-defined diagnostic costs that focus more on patients' health. I tested following research hypotheses:

- 2a: The sequential diagnostic models generated using the SDG algorithm have a higher diagnostic accuracy than the sequential diagnostic model generated using the ML-C4.5

algorithm.

- 2b: The sequential diagnostic models generated using the SDG algorithm are more clinically aligning than the sequential diagnostic model generated using the ML-C4.5 algorithm.

- 2c: The sequential diagnostic models generated using the SDG algorithm have a higher diagnostic accuracy than the policies developed by one implementation of deep Q learning algorithm.

- 2d: The sequential diagnostic models generated using the SDG algorithm are more clinically aligning than the policies developed by one implementation of deep Q learning algorithm.

- 2e: The sequential diagnostic models generated using the SDG algorithm uses less resources than models generated by ML-C4.5 algorithm and one implementation of deep Q learning algorithm.

1.4 Significance

This dissertation presents a framework that is designed to combine EHR and medical knowledge to make machine learning models that exhibit clinical alignment. Existing literature has demonstrated the importance of a computational diagnostic system in helping physicians reduce diagnostic error; it also shows that pure rule-based and pure machine learning-based approaches each have their own limitations. I use machine learning algorithms to combine EHR and medical knowledge to build a new type of computational diagnostic system. The key innovations of this dissertation research are:

1. A framework that can integrate expert-defined medical knowledge with disease patterns in EHR data for disease diagnosis.
2. An algorithm that generates medical classification trees that recommend diagnostic actions by considering clinical workflow, diagnostic accuracy, and misdiagnosis costs simultaneously.

Although my dissertation focuses on algorithm development and its accuracy and clinical alignment evaluation, if the system I propose works well, it could be applied to the following medical practice scenarios in the future.

Scenario 1: Provide Next-Step Suggestions

When a physician reviews the information that a patient has entered into a computation system, the system will apply the developed algorithm, some suggestions will pop up for the physician that suggest what physical exams may be informative and what lab tests are recommended to be ordered. Though my system may not be available for immediate use, when the ambient healthcare computing is achieved, this issue will have been addressed.

Scenario 2: Conduct a Final Review of Physician Diagnostic Process and Diagnosis

After a physician completes his/her diagnostic process and writes down the diagnosis, the system can compare the physician's diagnostic process with that of the tree model. If the physician forgets to order a lab test that is considered to be important by the sequential diagnostic model, the system can prompt a suggestion about this lab test. If the final diagnosis list of the system and the physician is different, the system can provide a possible diagnosis suggestion.

Scenario 3 : Help with Triage in the Emergency Department Waiting Room

My system can be installed on tablets. When a patient is sitting in the waiting room, a triage nurse can give him/her a tablet. The patient can type chief complaint, medical history in the system. The system will then ask questions in a physician-interview phase prompted by the tree model. In this way, the system can help physicians do the interview job to save

their time. We assume the patient is feeling well enough to enter information and is willing to do so. We also assume the system can elicit information well from the patient.

Scenario 4: Be a Part of Medical Student Training Classroom

Teachers can show the sequential diagnosis tree model to new medical students, and explain how they can use the tree to distinguish similar diseases. Because the sequential diagnostic model is very straightforward, medical students can read and understand rules easily.

1.5 Dissertation Organization

The rest of the dissertation is organized as follows. Chapter 2 provides a literature review of computational diagnostic systems, domain knowledge retrieval and integration. Chapter 3 introduces the research methods. Chapter 4 focuses on experiment design and results. Chapter 5 presents discussions, conclusions, contributions, and future work. The abbreviations used in this dissertation are listed in table 1.

Table 1: Abbreviations and Descriptions

Abbreviation	Description	Usage
C_i	a CCS category	Experiment for research aims (see Section 4.2)
$C(E)$	the expected misdiagnosis costs of a population dataset that includes multiple encounters	SDG algorithm (see Section 3.3)
$C(E_x)$	the expected misdiagnosis costs of one encounter given clinical evidence E_x	SDG algorithm (see Section 3.3)
$C_{d_i}(E_x)$	the expected misdiagnosis costs of one encounter related to d_i given clinical evidence E_x	SDG algorithm (see Section 3.3)
$cost(d_i \rightarrow d_j)$	the misdiagnosis costs of diagnosing an encounter with disease d_i to have disease d_j	Gorry and Barnett's algorithm (see Section 5.1.6)
d_i	a disease	SDG algorithm (see Section 3.3)
D_{ie}	a random sample of EHR training dataset for disease d_i	Experiment for research aims (see Section 4.2)
D_{iq}	all QMR-generated synthetic samples	Experiment for research aims (see Section 4.2)
E	a set of clinical evidence for a population dataset	SDG algorithm (see Section 3.3)

E_x	a set of clinical evidence for one encounter x	SDG algorithm (see Section 3.3)
EDC	expected diagnosis value	Gorry and Barnett's algorithm (see Section 5.1.6)
f_{jx}	a clinical feature, $1 \leq j \leq$ number of candidate features for one encounter x	SDG algorithm (see Section 3.3)
f_j	a clinical feature at the population level	SDG algorithm (see Section 3.3)
$M(\bar{d}_i \mid d_i)$	the cost of mistakenly missing the diagnosis of a disease d_i when a patient has the disease d_i	SDG algorithm (see Section 3.3)
$M(d_i \mid \bar{d}_i)$	the cost of mistakenly diagnosing that a patient has disease d_i when the patient does not have the disease d_i	SDG algorithm (see Section 3.3)
M_{ie}	a Naive Bayes model generated with D_{ie} for disease d_i	Experiment for research aims (see Section 4.2)
M_{iq}	a Naive Bayes model generated with D_{iq} for disease d_i	Experiment for research aims (see Section 4.2)
M_{ieqm}	a knowledge augmented model for disease i using EHR data D_{ie}, and M_{iq} with equivalent sample size m	Experiment for research aims (see Section 4.2)
$P(d_i \mid E_x)$	a posterior probability of having a disease d_i given an encounter's evidence E_x	SDG algorithm (see Section 3.3)

$P(\bar{d_i}\|E_x)$	a posterior probability of not having a disease d_i given an encounter's evidence	SDG algorithm (see Section 3.3)
Q	expected cost of misdiagnosis	Gorry and Barnett's algorithm (see Section 5.1.6)
$R(f_j, E)$	the reduction in the expected misdiagnosis costs given evidence at the population level, when evidence includes E and a new feature f_j.	SDG algorithm (see Section 3.3)
$R(f_{jx}, E_x)$	the reduction in the expected misdiagnosis costs given evidence at a patient level, when evidence includes E_x and a new feature f_j	SDG algorithm (see Section 3.3)
S_{ie}	features of a model M_{ie}	Experiment for research aims (see Section 4.2)
S_{ieqk}	features of a model M_{ieqm}	Experiment for research aims (see Section 4.2)
$Subset_i$	a subset of encounters that belong to the CCS category C_i	Experiment for research aims (see Section 4.3)

2.0 Literature Review

This chapter serves three purposes. Firstly, I compare the development of a computational sequential diagnostic system using a pure knowledge-driven strategy with using a pure data-driven strategy. Secondly, I introduce the techniques of combination of incorporating domain knowledge in machine learning. Thirdly, I present classic classification tree algorithms.

2.1 Computational Sequential Diagnostic System

Researchers have been exploring the potential of using computers to help physicians make diagnosis since computers were first developed. Many studies [49, 98, 106, 69, 52, 56] employ a similar strategy in their investigations of this possibility. A given set of clinical findings (symptoms, signs, and laboratory tests) are obtained for all patients. The results are used as the input of some statistical model; using this input, the model computes the probability distribution for some candidate diseases. Based on the determined distribution, the model suggests a list of final diagnoses. However, this strategy focuses on the inference function of medical diagnosis, which is only one aspect of the disease diagnostic process.

In fact, there is another important aspect in the disease diagnostic process - determining the proper sequence of tests to provide the information most needed to physicians, a process called sequential diagnosis. Physicians refer to the list of possible diagnoses for a patient as the "differential diagnosis", they refer to this disease diagnosis process as "resolving the differential diagnosis".

Existing medical diagnostic systems for sequential diagnosis can be categorized into two categories: clinical-knowledge-driven sequential diagnostic systems and data-driven sequential diagnostic systems.

2.1.1 Clinical-Knowledge-Driven Sequential Diagnostic Systems

Clinical knowledge-driven systems strongly depend on pre-defined clinical knowledge (e.g., the relationship between a disease and clinical findings). These systems often use score or logic in reasoning. The usage of these systems may be restricted by their knowledge base; they can only handle diseases and symptoms in the defined clinical areas. Extending and updating this type of system usually requires a lot of resources. Knowledge-driven sequential diagnostic systems can be divided into two types: rule-based systems and decision theoretic systems.

2.1.1.1 Rule-Based Systems

Flowcharts

A flowchart is the simplest type of rule-based system. A flowchart only contains sequential diagnosis rules, these rules are derived from medical domain knowledge and sometimes consider simple mathematical scores, for example, the Apgar Score [7]. The advantage of flowcharts is that they are easy to understand and use. When compared to statistical models, which are sometimes overly complicated for physicians to use to conduct calculations mentally, flowcharts also have advantages in application and teaching [27]. For example, James Fries et al. [27] constructed a flowchart using 30 questions, with approximately 1,000 separate routes that divides 190 patients in different clinical manifestations into 35 diagnostic categories. Diagnosis using this chart reached an accuracy of 96%. The advantage of this system is its simplicity and the visibility of the logical operations. These make it easy to use when teaching medical knowledge. The disadvantage is that it is not easy to generalize to other diseases.

In general, while a flowchart is very straightforward and easy to use in practice, the application of a flowchart is very limited because any unanticipated finding will render flowchart unusable until it is extended. This fact limits its usefulness in real-world practice, where unanticipated findings are the norm.

Expert Systems

A more complicated type of clinical knowledge-driven system is the expert system. An expert system generally consists of two components: a knowledge base and an inference engine. The knowledge base usually stores many medical terms and rules, and the inference engine uses these terms and rules to make inferences. As an important sub-field of artificial intelligence (AI), the development of expert systems involves the efforts of both AI researchers and expert physicians. The AI researchers first elicit expert physicians' detailed insights into the basic nature of clinical problem-solving, and then translate these insights into an intelligent system. These elicitation and translation processes are often dynamically changing, so it is essential that the team leader knows both AI and medicine well.

Many expert systems have been successfully developed. For example, Adlassnig et al. developed two expert systems, CADIAG-1 [4] and CADIAG-2. [3]. The CADIAG-1 uses symbolic logic to represent the relationships among medical concepts, while the CADIAG-2 uses fuzzy set theory and fuzzy logic to represent these relationships.

Another example, the INTERNIST-1 system [76, 58], is a decision support tool for general internal medicine. It was initially developed at the University of Pittsburgh in the 1970s. In 1985, its knowledge base was incorporated into a diagnostic system named, Quick Medical Reference (QMR). Later, the QMR was converted into as a two-level belief-network (Quick Medical Reference, Decision Theoretic, QMR-DT) [91] to represent the probabilistic dependencies between diseases and clinical findings. It contains 534 adult diseases and 4040 clinical findings, with 40740 arcs pointing from diseases to clinical findings. In QMR-DT, several assumptions were made to reduce the representational and computational complexity of the belief network, including: (1) marginal independence of diseases: there is no arc connecting diseases nodes; (2) conditional independence of clinical findings: there is no arc among finding nodes; (3) binary diseases and findings: each finding or disease has two possible statuses: present or absent; (4) causal independence: the effect of diseases on the finding (present or absent) occurs independently. This final assumption simplifies the calculation of the likelihoods of finding given multiple diseases. Findings are all modeled as manifestations of diseases. The probabilities in the two-level belief network of QMR-DT were defined based on national statistics data and knowledge in QMR. In particular, the

prior probabilities of disease were derived based on the National Center for Health Statistics, which were from approximately 192,000 inpatients discharged from short-stay non federal hospitals in 1984. The conditional probabilities were elicited from experts who originally developed INTERNIST/QMR by asking them to assign a frequency (1, 2, 3, 4, or, 5) to the question "How often do patients with the disease have the finding?" Then, the assignments were mapped to probabilities (1→0.025, 2→0.2, 3→0.5, 4→0.8, 5→0.985). When conducting a diagnosis, simulation algorithms are used, including the likelihood-weighting approach and self-important sampling.

2.1.1.2 Decision Theoretic Systems

The development of decision theoretic systems often follows the following steps: (1) an expert defines the knowledge base, including utilities for individual items; and then (2) some statistical method is used to calculate overall utilities for different settings and then choose the setting with highest overall utility. For example, G. Anthony Gorry built a decision-theoretic system [33, 30, 29, 31] consisting of three components: an information structure, an inference function, and a test selection function.

The information structure contains rich domain knowledge, including the prior probability of each disease, the conditional probability of signs and symptoms for each disease, the cost of the lab test, and the cost of misdiagnosis. The inference function uses Bayes rules and the most recent observation of a patient's clinical features to update the probability distribution of suspected diseases. The test selection function is designed to identify the best test to be done from among a number of candidate tests and also "no test." The test selection function uses the probability distributions of diseases and the cost of possible misdiagnoses to determine the best diagnosis and the expected risk of misdiagnosis. For each of the candidate tests, the test selection function uses the disease probability distribution, the cost of the test, and the likelihood of possible test results to evaluate the potential usefulness of the test.

Alan Rector and Eugene Ackerman [81], upon finding that Gorry's method does not work if none of the candidate tests can change the choice of the best diagnosis, proposed a new

test selection function that chooses tests based on their cumulative effects on misdiagnosis costs. Their experimental results showed that the new function requires little computation and is robust against small errors in the knowledge base.

Mussi [63] proposed a similar but more complicated algorithm, which considers 10 aspects of each test: seriousness, urgency, probability, sensitivity, specificity, risk, cost, discomfort, time required, and remarks about default and exceptional values. This algorithm also considers the interaction among variables. For example, the "urgency" feature affects the importance weight of the "time required" feature. With this domain knowledge, the algorithm uses a Bayesian approach to suggest the next test.

In another study, Castro et al. [16] used an analytic hierarchy process strategy to develop a sequential test selection method for analyzing the underlying cause of abdominal pain. They split the work into three levels. The first level (goal) is to choose the optimal test strategy. The second level consideration of four criteria: minimizing costs, minimizing discomfort, minimizing risk, and maximizing diagnostic ability. The third level involves consideration of the diagnostic capability of four lab tests: abdominal CT, upper GI series, abdominal ultrasound, and endoscopy. The relative weight of the four criteria and probabilities related to the diagnostic accuracy of the four lab tests were estimated by domain experts. The diagnostic capability of each test is calculated using [(test sensitivity)$\times$(disease frequency)+(test specificity)$\times$(1-disease frequency)]. A score is calculated using all criteria and is used for choosing which test should be done next. Once the result of the test is known, the disease probability is updated through a Bayesian method.

In addition, Druzdzel et al. [67, 68, 66] developed a Bayesian network, of which the feature set and structure are defined based on domain knowledge while the parameters (conditional probabilities) are learned from real-world data. They also used a cross-entropy approach to rank the importance of the unobserved features and named the value as the diagnostic value. Based on the diagnostic value of different features, physicians can select the most informative feature to continue their exploration. Druzdzel et al. further implemented this approach in their Bayesian Network software, GeNie.

2.1.2 Data-Driven Sequential Diagnostic Systems

In addition to using domain knowledge to (partially) define computational sequential diagnostic systems, learning from real-world data is another way to approach development. In this section, I will introduce three types of data-driven sequential diagnostic systems.

2.1.2.1 Bayesian Diagnostic Systems

Diagnosis in medicine often involves significant uncertainties. Many diagnostic systems [40, 37, 32, 54] uses Bayesian probability approach to manage these uncertainties. The Bayesian approach often involves a prior probability distribution over the possible diseases and many conditional probability distributions for each clinical feature given each disease. With these probabilities and a patient's clinical manifestations, diagnostic systems can provide posterior probability distribution over the possible diseases that are derived from Bayesian inferences. For example, the Pathfinder [38, 38] is an expert system for the diagnosis of lymph-node disease using over 30 features reflecting clinical, laboratory, immunological, and molecular biological information, where expert knowledge of these features is encoded in a belief network.

If a Bayesian diagnostic system can optimize the order in which it asks about manifestations, then it is called sequential diagnostic system. For example, Gorry [32] developed sequential diagnostic systems for acute renal failure. The Phase I system provides the order of questions so that, at each time point, the expected can be minimized by getting the answers of the identified question at that time, while the Phase II system uses expected utility to guide question asking.

If the probability distributions of a system are learned from data, then it is data-driven. For example, Mcsherry [54] proposed an algorithm to do test selection using a Naive Bayes method. All diseases are assumed to be mutually exclusive and exhaustive, and the test results are assumed to be conditionally independent given each disease. The measure of attribute usefulness is called evidential power. The equation of the evidential power is $\lambda(A, H_t, \zeta) = \sum_{i=1}^{n} p(A = v_i|H_t)p(H_t|A = v_i, \zeta)$, where $v_1, v_2, \ldots, v_n$ are the values of test

A and ζ represents all other evidence provided by previous tests. The evidential power is used to indicate the ability to increase the probability of the target hypothesis (i.e., the probability of the correct diagnosis).

2.1.2.2 The Reinforcement Learning Method

Lastly, reinforcement learning (RL), a sub-field of machine learning, has been explored as a method to solve sequential decision making problems, aiming to optimize sequences of actions to reach the best long-term outcome. The main components of RL are **Agent(s)** and **Environment**. In RL, an agent operates in an environment setting: it observes (possibly partial) **State** of the environment; from a set of candidate actions, the agent chooses an action; after that, the agent takes the chosen **Action**. After the agent takes an action, the state of the environment changes, and the environment gives the agent an immediate **Reward** (or penalty). When operating in the environment, the ultimate goal of the agent is to find a sequence of actions that maximizes a cumulative reward.

Mathematically, a RL problem is often framed as a Markov decision process (MDP) [11]. A MDP includes five components (S, A, P, R, γ) [116]: S is a finite state space consisting of possible states of an environment; A is a finite action space consisting of possible actions that an agent can take; P is a Markovian transition function representing the probabilities of transitioning from one state to another given a (state, action) pair; R is a reward function defining the reward given (state, action) pair; γ is a discount factor related to long-term reward. Additionally, a policy π is a function from S to A. It specifies the recommended actions that the agent should take for every possible state in the environment.

The goal of RL is to find an optimal policy that maximizes cumulative discounted reward. To find this policy, traditional RL algorithms define strong assumptions about transitional probabilities among states. These are called model-based RL, where "model" refers to the model of transitional probabilities. Unfortunately, the difficulty of defining the model of transitional probabilities greatly reduces the ability of these model-based RL algorithms to be applied to many real-world scenarios. However, for complicated scenarios where it is hard to define transitional probabilities, model-free RL algorithms may be useful.

Model-free RL algorithms, unlike their model-based counterparts, do not assume an exact mathematical model of the transitional probabilities. As an example, one classic (transitional probability) model-free RL, **Q-learning algorithm** [107], uses incremental method to estimate action-values. Given a start state s_0 and a policy π, the **value function** provides the expected cumulative discounted reward: $V^{\pi}(s) = E[\sum_{\geq 0}^{t} \gamma r_t | s_0 = s, \pi]$, where t is the time step, γ is the discount factor, and r_t is the reward at time t. This value function indicates how good a state is. Then, what action should an agent take in a state s? The **Q-value function** is the expected cumulative reward from taking action a given a start state s_0 and a policy π: $Q^{\pi}(s, a) = E[\sum_{t \geq 0} \gamma r_t | s_0 = s, a_0 = a, \pi]$. The **optimal Q-value function** Q^* is the maximum Q value achievable: $Q^*(s, a) = max_{\pi} Q^{\pi}(s, a) = max_{\pi} E[\sum_{t \geq 0} \gamma r_t | s_0 = s, a_0 = a, \pi]$. By Bellman's theory, optimal Q-value function also satisfies: $Q^*(s, a) = E_{s' \sim \epsilon}[r + \gamma \times max_{a'} Q^*(s', a') | s, a]$. Therefore, the, Q^* function suggests the **optimal policy** π^*, which takes the best action in any state as specified by Q^*. Briefly, the Q-learning algorithm includes a sequence of episodes. In each episode, the agent observes the current state, selects and performs the action with the highest reward, observes the subsequent state, receives an immediate payoff, and then adjusts the Q values by incorporating the immediate payoff. With these episodes, the Q values will be iteratively updating. The Q-learning algorithm was shown to converge to the optimum action-values (with probability 1) when the action-values are discrete and all actions are repeatedly sampled in all states [108]. The Q-value function learning can involve trial and error to learn about the world (exploration) and explicit selection of the best known action at a state (exploitation).

One popular variant of the Q-learning algorithm is deep Q learning (DQL), firstly developed by Mnih et al. at DeepMind Technologies in 2013 [60]. They used the deep neural network approach to represent and iteratively learn the state-action Q value function. The deep neural network is very powerful in representing distributions - it can represent any non-linear function well. The model the DeepMind team used is a convolutional neural network: the input is raw pixels and the output is a set of estimated future rewards, with one value corresponding to an action. The team applied the DQL based RL to seven Atari 2600 games and demonstrated a high performance on all games.

Researchers are increasingly interested in using DQL for automated medical diagnosis.

Tang et al. [94] applied DQL in their symptom checking system, where the agent makes disease diagnosis by querying information about symptoms. The system needs to make a sequential decision about which symptoms to check, whether a diagnosis should be made and what the diagnosis should be. The system includes a set of deep neural networks. Each deep neural network is used to represent a non-linear state-action Q value function, which estimates a set of Q-values for an input state (a set of clinical findings with one-hot representation of positive, negative, unknown values) and all potential actions (all candidate questions and all suspected diagnoses). The immediate reward after an action is defined as: 1 if the action predicts the correct disease (in training data), -1 if the system asks one question repeatedly, and 0 otherwise. The inquiry system terminates when the action makes a diagnosis (regardless of whether it is correct or not).

Liao et al. [50] extended Tang's work by proposing a hierarchical reinforcement learning approach. This system represents the policy function in a two-level hierarchical way. The high-level policy controls the trigger of a low-level policy, while the low-level policy consists of a disease classifier and many symptom checkers. Each symptom checker controls symptom queries related to a certain group of disease, which is like a specialist. Their collected patient information is then used by the disease classifier, mimicking the scenario of a group of specialists meeting and making a final diagnosis. The strategy of using multiple networks with each representing one medical domain may be rational because of the complication of the relationships between symptoms and actions/diagnosis. For example, a fever related to respiratory disease is much different from a fever related to a digestive disorder.

In addition to the DQL RL approach for medical diagnosis, there is a LEAD system that uses a rule-based RL to suggest proper diagnostic tests. This was developed by Fakih and Das [21]. The performance criterion considered are the cost of testing, morbidity, mortality and time, and the diagnostic ability of the tests. The test candidates and performance criterion are input into the RL algorithm, and the optimized diagnostic strategies are learned. The approach was evaluated on a diagnostic problem of solitary pulmonary nodule, and the results showed that the RL algorithm improved testing strategies in diagnosis compared with several fixed testing strategies. The LEAD system considers the medical diagnostic tests scenarios more completely, but the disadvantage of this type of rule-based system is

that it may need an exponential number of decision rules and thus may not be feasible when the state involves a relatively large size of variables.

2.1.2.3 Other Data-driven Sequential Diagnostic Systems

There are a few other data-driven sequential diagnostic systems. I review two examples here.

Case Similarity System

Rosati et al. [83] built an information system for ischemic heart diseases. For ischemic heart disease patients, the researchers collected a defined set of data, which included history of present illness, medical history, review of forms, physical exams, lab test results, treatment information, and follow-up information. This information system works as follows: When a new patient comes in, his/her information is entered into the system. Based on this information, the system conducts a query on its database, which returns with a specific subgroup of patients that are similar to this new patient. In an example, the off-line report includes three sections: the new patient's information, the clinical descriptions of the subgroup, and the prognosis of the subgroup. Based on this report, physicians can continue the diagnostic process by prescribing further laboratory tests.

Optimal Coordinate System

Kulikowski [46] developed a method of class featuring information compression algorithm to conduct sequential diagnosis for hyperthyroidism. All diagnosis-related features were split into three groups: symptoms, physical exams, and lab tests. Instead of considering features one by one for diagnosis, at each step, a new group of features is combined with the previous ones, and the entire feature set is used to find the best set of diagnostic features. For each class, an optimal coordinate system is found. Most intraclass variation is concentrated along a few of these optimal coordinates; by discarding the less important coordinates, the number of tests is decreased while minimal error is achieved. For a new patient, the patient's information is firstly converted to a test vector. Then, the squared cosine of the angle between the test vector with its projection onto each subclass is calculated to be a measure of the test

vector's closeness to each subclass. By setting the threshold on the difference, the researcher can classify the test case into three categories at each step: 1) he does not have any diseases; 2) he has one of the diseases; 3) more information is needed to get a reliable diagnosis. The sequence of calculation can be summarized as follows:

1) form test case vector

2) calculate the square cosine values between test case and each class

3) obtain the difference, difference = value(class A) - value(class B)

4) decide on classification

4.1)if difference is greater than threshold 1, then test case belongs to class A

4.2)if difference is less than threshold 2, then test case belongs to class B

4.3)if difference is bigger than threshold 2 and less than threshold 1, then move to next stage, where a new variable is added in.

The researcher obtained the sequential order by splitting features into groups, and adding groups into diagnosis by order. The researcher then decreased the number of tests by deleting unimportant coordinates.

2.2 Incorporating Domain Knowledge into Machine Learning

Domain knowledge has not gained enough attention in this era of big data and machine learning, where most modeling processes are data-driven. When developing models, researchers often rely heavily on training data and ignore most of the domain knowledge. The main reasons for this are: (1) researchers who develop machine learning algorithms usually create general purpose algorithms rather than some algorithms in a specific domain; (2) domain knowledge may be represented in many ways; there is no one standard format to represent it; and (3) domain experts usually cannot understand complicated machine learning algorithms, and thus do not have the skills to add their domain knowledge into machine learning.

Unfortunately, when solving real-world problems, only relying on training data can be problematic. Firstly, it is rare that the training data we used can capture all the patterns in

a whole population. Another potential issue is over-fitting. You cannot build a model that fits the training data too well: the developed model will be too specific to the examples in the training data, which means it would work poorly on real life data samples (i.e., lack of generalization).

Combining domain knowledge and training data can be valuable in that machine learning algorithms can discover patterns that are too subtle for humans to detect, and domain knowledge can contain information about a specific domain that is not well represented in the training data. thereby addressing the above issues. There are at least four ways for researchers to incorporate domain knowledge into machine learning: (1) use domain knowledge to prepare training data; (2) use domain knowledge to initiate the hypothesis; (3) use domain knowledge to change the search objective; and (4) use domain knowledge to augment searches. I will review each of these approaches in this section.

2.2.1 Using Domain Knowledge to Prepare Training Data

Using Domain Knowledge to Clean and Transform Training Data

Domain knowledge can provide important information to remove redundant features, two of the main reasons for poor performance in machine learning. If we do not have sufficient domain knowledge, we do not know which features are important for the target variable; this means the model then needs to have access to and then mine a large size of representative training data [100].

Moreover, domain knowledge can be used to transform original features into new features or to find must-link features (features that must be considered at the same time) and cannot-link features (features that must not be considered at the same time). For example, Joao Vieira and Claudia Antunes [103] proposed an ontology-driven classification tree learning algorithm, where each feature in the training data was put into a related domain ontology, and each level of abstraction for each feature was returned and combined into the training data set. These researchers further used domain knowledge to create new features, including abstraction features and feature combinations. With these ontology knowledge-augmented features, machine learning was able to generate smaller and more accurate trees.

Using Domain Knowledge to Generate Virtual Samples

Prior domain knowledge is also useful as it can be used to generate new training samples (i.e., virtual samples), these virtual samples can improve model performance when the sample size is limited, or the training data is unbalanced. Virtual samples can be newly generated [75] or be transformed from original data [65]. For example, Partha Niyogi et al. [65] incorporated prior information into neural networks and used this approach to generate virtual examples, which is mathematically equivalent to adding domain knowledge as a regularizer. Murali Ravuri et al. [80] used an expert system as a generative model to simulate data for modeling. They found that the developed model not only preserved the original properties of the expert system, but also added new properties and addressed some limitations. They also showed that simulated data can be combined with real-world electronic health record data.

The major drawback of adding virtual samples is the increase in computational cost. However, Scholkopf et al. [86] have proposed a virtual support vectors method that can preserve the advantage of virtual samples without increasing the computational cost. In their study, the support vector machine algorithm trained models solely on support vectors, and the training process was split into three steps: (1) using the training data to get the support vector set; (2) generating virtual samples and obtaining virtual support vectors by applying invariance transformations to support vectors; and (3) using the enlarged set of support vectors to obtain another support vector machine model.

Using domain knowledge to generate virtual samples is an indirect way of incorporating the knowledge. This solution can be applied universally for any machine learning method.

2.2.2 Using Domain Knowledge to Restrict the Hypothesis Space

Another way of using domain knowledge is to restrict the hypothesis space so that all candidate hypotheses can align well with the existing domain knowledge. For example, Langseth et al. [47] developed an Object-Oriented Bayesian Network method that defines domain knowledge-based hierarchical relationships. The properties of the super-class are inherited by the subclasses. This facilitates the structural learning of a Bayesian Network

structure.

With domain knowledge, researchers can add some restrictions to the search space or the hypothesis space. For example, Yoon Suk-Chung et al. [115] proposed a semantic query optimization method that transforms a query into an equivalent form, which can be evaluated quickly in the narrowed search space. For example, suppose we have the following domain knowledge: "all ships whose dead weight is greater than 700 tons travel at a speed greater than 60 mph", and "ships whose dead weight are greater than 700 tons are supertankers". According to the domain knowledge, when querying about the ship that fulfills the dead weight and speed requirements (greater than 700 tons and speed greater than 60 mph), only ships that are supertankers should be considered, and the dead weight and speed can be ignored.

Using domain knowledge to pre-define a hypothesis space may provide a better starting point for machine learning, and thus lead to a faster convergence speed and lower computation cost. For example, Shavlik and Towell et al. [88] proposed a Knowledge-Based Artificial Neural Network (KBANN) method. Firstly, they used domain knowledge to construct an initial network. Then, they used training samples to adjust the weights of the initial network using a back propagation algorithm. With a better starting point (the knowledge-defined initial network), the KBANN developed a final model that achieved a better generalization accuracy. In another example, Mirchevska et al. [59] proposed a domain knowledge-based method to identify people who have experienced falls. A domain expert firstly defined an initial classifier that contained some classification rules. Then, the researchers used training data to refine these rules. Their experimental results showed that the refined classifier was more reliable and robust than a machine-learned classifier that was developed using limited samples.

2.2.3 Using Domain Knowledge to Refine the Search Objective or Verify Search Results

Domain knowledge can be used to refine the search objective or verify the search results. Most machine learning algorithms typically transform a learning task into an optimization

problem, which mathematically defines an objective function. The goal of learning is then to find a set of parameters that can minimize the objective function. The machine learning process uses training data to calculate empirical risk and uses algorithms (e.g., gradient descent) to reach the optimal point. In this type of optimization setting, when researchers want to add domain knowledge, they can either add an additional regularizer to the objective function or add a set of additional constraint functions.

Adding domain knowledge in the form of a regularizer or additional constraints is not trivial. What should be added heavily depends on the research design. Overall, there are two types of designs: (1) modifying the objectives or searching scores to fit both the domain knowledge and the training data; and (2) using domain knowledge to verify the search result.

2.2.3.1 Modifying the Objectives or Searching Scores to Fit both the Domain Knowledge and the Training Data

Researchers have explored three main ways to modify objectives or searching scores to fit both the domain knowledge and the training data: setting constraints, assigning weights to samples, and incorporating costs.

Modifying the Objectives or Searching Scores by Setting Constraints

For machine learning algorithms that use objective functions, domain knowledge can be transformed into additional constraints in these functions. For example, Abu-Mostafa [2] defined *Hints* to represent the penalty of violating the domain knowledge, hints included: invariance hint, monotonicity hint, example hint, approximation hint, consistency hint, and catalytic hint. Each hint has its own calculating function. If the training data violate the *Hints*, then the corresponding risk $R_{knowledge}$ will be included in the objective function. In another example, Muralidhar et al. [62] proposed a domain adapted neural networks (DANN) to combine domain knowledge with deep neural networks. They incorporated knowledge about monotonicity constraints and approximation constraints into the loss function of deep neural networks. The results showed that the domain-aware DANN model significantly outperformed the domain-agnostic neural network model in sparse and noisy settings.

Instead of using objective functions, many searching algorithms use scores. For these algorithms, domain knowledge can be added into the searching scores. For example, Iqbal et al. [5] proposed an approach to incorporate a feature importance score into classification tree learning. Importance score is defined as the expected probability of the class variable given the feature. The searching score is a weighted sum of the importance score and the information gain score used in classic classification tree algorithms. Using this new score, their searching algorithm produced better trees than traditional classification tree learning algorithm. Tran et al. [99] provided another example. Their method modified the gain score of a candidate feature of the classification tree C4.5 method by multiplying extra weight based on information from MEDLINE, the National Library of Medicine's bibliographic database containing over 27 million journal articles on the life sciences with a concentration on biomedicine. A feature's external weight was calculated based on the number of references in MEDLINE. The authors hypothesized that the larger the number of documents in MEDLINE, that mentioned the feature, the greater its external weight.

Modifying the Objectives or Searching Scores by Assigning Weights to Samples

Learning with weighted samples is another commonly used method for incorporating domain knowledge into machine learning algorithms. For example, Wu et al. [113] developed a Weighted Margin Support Vector Machine. This algorithm firstly used prior knowledge to generate virtual training data, and then used a hyper-parameter to control the relative importance between virtual training data and other training data. In another example, Tay et al. [15] developed a new SVM algorithm to consider the importance of timeliness in time-series problems, which assigned higher weights to the most recent training samples and lower weights to the older training samples.

Incorporating Costs into Objectives or Scores

Many classification tasks have different costs for misclassification errors, which is an important aspect to consider. A cost-sensitive classification algorithm incorporates the information about costs into machine learning. For example, Vo et al. [105] used deep neural network models to do sentiment analysis in natural language processing. They realized that

the loss functions in traditional approaches did not reflect the degree of errors of sentiment misclassification very well. To solve this problem, they developed a new penalty matrix, which considered the different levels of importance of misclassification errors. This matrix provided a better result, compared to the classical cross entropy loss function.

2.2.3.2 Using Domain Knowledge to Verify Search Results

Finally, domain knowledge can be used to guide the searching process through verifying search results. For example, Russell et al. [84] proposed a knowledge-guided autonomous learning algorithm. This algorithm first generates all candidate hypotheses in terms of primitive language that are then tested for consistency with prior knowledge.

2.2.4 Using Domain Knowledge to Augment a Search

Using domain knowledge to augment a search is similar to using domain knowledge to alter a search objective. The difference is that using domain knowledge to augment a search produces new hypothesis candidates in the process of searching. Pazzani et al. [72] proposed the FOCL algorithm as an extension of the FOIL system. The FOIL system generates hypotheses purely from training data. FOCL adds in the use of domain knowledge to generate additional specifications. A limited amount of research has been done in this category, as it is difficult to adjust both the process of convergence and the hypothesis space at the same time.

2.3 Classification Tree Algorithm

2.3.1 Why Choose a Classification Tree Algorithm for Medical Diagnosis

A classification tree model starts from a root node and partitions data recursively into subgroups. For medical diagnostic problems, we choose classification tree structures because they are:

1. **Interpretable**: In classification tree models, the path from root to leaf node is very similar to reasoning process physicians use to conduct differential diagnosis. With straightforward structures, the tree models can be easily understood by potential users (e.g., clinicians). Blackbox models (e.g., deep neural network models) usually have complicated linear and nonlinear transformations that are much less interpretable. Therefore, we will not consider deep reinforcement learning algorithms in our study.
2. **Actionable**: Because classification tree model is similar to physician's reasoning process and is easy to be understood by physicians, the features learned in classification tree model can be used by physicians directly when diagnosing new patients. This property does not exist in other interpretable models, such as Bayesian network models.

2.3.2 Classification Tree Algorithm Introduction

One of the most popular data-driven approaches is the the classification tree algorithm [8]. A classification tree algorithm generates a sequence of tests, these may can be clinical questions or laboratory tests. For each test, if its value is numeric, then the algorithm finds a cut-off threshold to transform the test value to be categorical. Once the value is categorical, then this test can split patients into two or more subgroups. The classification tree model is trained in this way:

1. Based on the values of some searching score, a feature is chosen as the next node (next test). The first node is the root node.
2. Based on the values of the selected node, the training data is divided into subgroups. Here, there are many variants. If the selected node is numeric, it is transformed into a categorical variable and the threshold is often data-driven. If the selected node is categorical but has a lot of values, these values are grouped so that the number of categories will be much smaller.
3. In each subgroup, step 1 and step 2 are repeated, until the subgroup only contains one category, or reaches some human-defined threshold, such as the maximum tree depth. The leaf nodes of a classification tree can be used to do prediction, the majority category of the node is the predicted value.

For a patient, using a classification tree for diagnosis follows these steps:

1. Starting with the root node, the value of the patient is compared with the node values, and the patient is moved to one of the subgroups.
2. Step 1 is repeated until one of the leaf nodes is reached
3. The majority class in that leaf node is the diagnosis for the patient.

2.3.3 Multi-label Classification Tasks and Solving Strategies

An entity may have multiple labels in real life. For example, a patient may have more than one disease; a gene may have more than one function; a document may cover more than one topic. This creates the need for research methods to cope with multi-label problems.

Multi-label learning aims to predict labels of test cases by learning from training cases that are associated with a set of labels simultaneously. There are three general strategies for multi-label classification tasks: data transformation, ensembles of classifiers, and method adaptation. I first introduce these three general strategies for handling multi-label classification tasks, and then I introduce several multi-label classification tree algorithms.

2.3.3.1 Strategies for Multi-label Classification Tasks

Data Transformation Strategy

The idea of data transformation is to generate datasets that can be processed using binary or multi-class classifiers. Later, the output of those classifiers is back-transformed to obtain the multi-label diagnosis. Some common methods include:

- Applying binarization techniques [28]: this method trains k classifiers, one for each label, taking the instances, in which the labels appear as positive and all others as negative.
- Selecting a single label [12]: when a sample is associated with multiple labels, this method chooses one label randomly or based on some score methods.
- Ignoring multi-label instances [12]: this method dismisses all the samples associated with more than one label. This transformation generates a new dataset that has only one label per instance.

- Unfolding samples with multiple labels [101]: this method unfolds each sample into as many samples as labels it contains, cloning the input features and assigning to each sample one of the labels.
- Using the label set as a class identifier [12]: this method uses each different combination of labels as the identifier of a new class. The new dataset has the same number of samples, but only one class per instance.

Classifier Ensemble Strategy

Classifier ensemble is a widespread technique to improve the performance obtained by individual classifiers. An ensemble is compounded by a set of classifiers, of which the outputs are combined in a weighted or unweighted averaging way. The theory is that a group of weak classifiers that have different biases may perform better than a strong classifier. Ensembles of binary classifiers have been used to solve many multi-class classification tasks, either by one-vs-all or one-vs-one decompositions. Therefore, it is not surprising that ensemble techniques are also applied to many multi-label problems[102, 89, 82].

Method Adaptation Strategy

The method adaptation approach aims to adapt existing classification algorithms so that they can deal with multi-label data, producing several outputs instead of only one. The adaptation can be quite simple or complex, depending on the nature of the original method. I will provide some examples in the next section.

2.3.3.2 Some Multi-label Classification Tree Algorithms

In this section, I introduce some multi-label classification tree algorithms. A few of them used both the data transformation strategy and the classifier ensemble strategy.

1. Multi-label Classification Tree Algorithms Using both Data Transformation and Classifier Ensembles Strategies

Freund and Mason [25] proposed an alternating classification tree algorithm, which is a generalization of classification trees, voted classification trees, and voted decision stumps. It provides an alternative to techniques such as boosting to improve the diagnostic performance of tree-based classifiers.

Francesco De Comite et al. [20] combined the Adaboost method and alternating classification tree ideas to propose an ADTBoost.MH algorithm. The samples are decomposed following the one-vs-all strategy. The samples are re-weighted at each boosting step: samples that were misclassified by the hypothesis in the previous round have a higher weight in the current round. Since ADTBoost.MH trains many models and each of them is an alternating classification tree, it transforms data using the binarization technique. In this way, it combines the data transformation strategy and the classifier ensemble strategy.

Wu [112] proposed an ML-TREE algorithm. This algorithm treats a tree as a hierarchy of data. At each node, it uses many one-vs-all SVM classifiers to recursively partition data into subgroups. For each leaf node, the algorithm defines a predictive label vector to represent the multi-label diagnosis. The obtained model can be considered an ensemble of multiple SVM classifiers.

2. Multi-label Classification Tree Algorithms Using the Method Adaptation Strategy

Clare et al. [17] proposed a multi-label C4.5 (ML-C4.5) algorithm. The new algorithm is founded on the well-known C4.5 algorithm, appropriately modified to deal with multi-label problems. The two key points of the adaptation are:

- The leaves of the tree contain samples that are associated with multiple labels, instead of only one label.
- The original entropy measure is adjusted to take into consideration the possibility that the instance is not a member of a certain label. The original entropy measure of the C4.5 algorithm is $entropy(S) = -\sum_{i=1}^{N} p(c_i)logp(c_i)$. where $p(c_i)$ is the probability (relative frequency) of class c_i in the data set. The updated entropy measure in the ML-C4.5 algorithm is $-\sum_{i=1}^{N}(p(c_i)logp(c_i)+q(c_i)logq(c_i))$. where $p(c_i)$ is the probability (relative frequency) of class c_i; $q(c_i)$=1-$p(c_i)$ is the probability of not being a member of class c_i. If a sample has multiple labels, then it will be counted several times when calculating a new entropy.

AI-Otaibi et al. [6] proposed a LaCova algorithm that uses the adaptation strategy. This

algorithm considers the correlations among labels. The key idea is that the splitting criterion is based on the label covariance matrix at each node, which allows the algorithm to choose between a horizontal split (branching on a feature) and a vertical split (separating the labels).

3.0 Methodology

This chapter includes five sections. The first section describes the medical knowledge that I applied to EHR machine learning in my studies. I used three types of domain knowledge. The first type of knowledge is represented in a computational medical expert system. This system contains a rich set of diseases, their clinical features, and their correlations. I used this system to generate synthetic data that I could use in my studies. The second type of knowledge that I used in my studies is misdiagnosis costs defined by human experts. Using this knowledge, I set up a criterion of automatically identifying the next diagnostic action that can reduce estimated expected diagnosis costs when generating a sequential diagnostic model. The third knowledge that I included in the modeling process is the order of features in a regular clinical workflow. Based on a physician expert's knowledge, I classified the features into four phases: triage nurse phase, physician interview phase, physical exam phase, and lab test phase. Since this order aligns well with the regular clinical workflow, this order was considered by my algorithm when restricting the diagnostic actions (i.e., nodes) in the sequential diagnostic model.

The second section describes how I combine medical knowledge and EHR data to develop Naive Bayes models that can conduct differential diagnosis. The third section introduces how my sequential diagnosis generating algorithm (SDG) generates sequential diagnosis recommendations by considering misdiagnosis costs and clinical workflow. The fourth section introduces how multi-label C4.5 (ML-C4.5) algorithm can be used in sequential diagnostic modeling. This algorithm was used as first approach for comparison. This section then presents a comparison of the performance of a model developed by ML-C4.5 with that of a model learned by the SDG algorithm. Finally, the fifth section introduces how Deep Q learning, one implementation of classic deep reinforcement learning algorithm, can be used find the best diagnostic strategies. This algorithm was used as the second approach for comparison.

3.1 Medical Domain Knowledge Used in this Study

A medical diagnosis problem can be formulated as a sequential decision-making task. After a new patient comes to a hospital, clinicians need to make a sequence of diagnostic actions, including asking the patients many questions, conducting many physical exams, and ordering some lab tests. This sequence of diagnostic actions often directly determines whether a diagnosis will be correctly made, how soon it can be made, and how much medical resource will be utilized.

When I first built a sequential diagnostic model (in the format of a classification tree) using the classic ID3 classification tree algorithm, I encountered three problems. **(1) a multi-label problem**: A patient with multiple heart diseases will have multiple diagnosis labels; the classic ID3 classification tree algorithm cannot handle this type of training data; **(2) limited training data**: when the sample size of training data is very limited, the model's performance is not good; **(3) lack of clinical significance and being clinically unactionable**: the node searching criteria only considers information gain or some mathematical score; thus, it cannot guarantee that the generated tree models are in clinical alignment. For example, a diagnostic tree may suggest conducting a laboratory test prior to a triage nurse question, simply because laboratory test results are more discriminating than some symptoms. Unfortunately, given that this type of order does not align well with regular clinical workflow, it is unlikely that physicians will follow this diagnostic order in their real-world clinical practice.

To solve the first problem, I first built eight Naive Bayes models, one for each disease category, and then combined them into a diagnostic tree using the proposed algorithm. To resolve the last three problems, I combine medical domain knowledge with EHRs for model development.

3.1.1 The QMR System and its Simulation Data

The first knowledge that I used in my study is the medical knowledge stored in the Quick Medical Reference (QMR) [57], which was developed by Dr. Randolph A. Miller

and colleagues at the University of Pittsburgh in 1980s. The QMR helps internists conduct clinical diagnoses for adults by providing access to medical knowledge including diseases and findings about them (e.g., signs, symptoms, lab results, demographic information, and past medical history information), as well as the relationships among them. When there is a relationship between a disease and a finding, then there is a set of representations describing this relationship. Briefly, the strength of the relationship between a disease and a finding is parameterized by two variables: evoking strength (0-5 scale) and frequency (1-5 scale). These values are manually assigned based on physicians' judgement and the medical literature. An evoking strength between finding A and disease B represents the likelihood: "if finding A appears, how likely this patient has disease B." The frequency variable is about "if one patient has disease B, how often finding A appears." In addition, there is another variable "import" for each finding, which is not related to disease. This variable represents the global importance of each finding. For example, "shortness of breath" is more important (higher import score) than the finding "feel cold."

Different from QMR knowledge base, EHR data are in a format of structured variables or free-text notes that contains the majority of clinical information. After using natural language processing to extract clinical features and combining them with structured variables, we often obtain research data as a matrix. In the matrix, each row is one patient encounter; each column is one clinical feature; and the value in each cell is the value of this feature for this record.

Because the format of QMR domain knowledge and EHR data is so different, directly combining the knowledge base in QMR with EHR data is difficult. An alternative is to transform the knowledge provided in the QMR into synthetic records that still contain rich medical knowledge. The approach of simulating cases from the QMR knowledge base has been explored by Parker et al. [71]. Briefly, the steps reported for this approach are as follows: (1) The simulator chooses a disease; (2) For a patient having this disease, the simulator samples the patient's demographics variables with probabilities proportional to the values of associated frequency variables; (3) The simulator then samples clinical findings (predisposing variables) in a decreasing order of the value of the frequency variable. Each clinical finding is randomly chosen to be present or absent. If a finding is chosen to be

present, then findings that are not chosen but are impossible to manifest are removed from the candidate list (e.g., if the patient's gender is male, then this patient cannot be pregnant). If a finding is chosen to be present, its high co-occurrent findings are then prioritized to have a high possibility of being present.(4) The simulation ends when all findings in the knowledge base have been considered. Murali Ravuri et al. [80] have found that QMR simulated data can improve the performance of deep learning models, indicating that QMR simulated data contains valuable quantitative relationships between diseases and clinical findings.

In my study, I used the synthetic data to calculate useful conditional probabilities and then applied a prior equivalent sample size method to combine these probabilities with probabilities learned from EHR data (Section 3.2). My hypothesis is that using the medical domain knowledge found in the QMR system would be able to improve model performance when the sample size is limited.

3.1.2 Misdiagnosis Costs

When using EHR data and a machine learning approach to develop diagnostic models, one fact is often ignored: diseases have different levels of clinical severity. For example, acute myocardial infarction is more severe than chronic high blood pressure. Emergency department physicians usually need to make diagnoses in a short time with limited information, and misdiagnosis errors have various consequences of varying impact. For example, if one patient has a severe acute disease and the physician does not discover this, the consequences can be very serious - valuable time may be wasted and the patient might even die as a result of delay in proper treatment. If a patient has a mild disease and the physician does not discover this, the consequences will be less serious - medical resources may be wasted, but a patient's treatment may not be delayed that much. In a third scenario, if a patient does not have a severe disease but is diagnosed by a doctor as having the disease, the consequences will also be less serious - medical resources may be wasted and patients feel a lot of stress. If a patient does not have a disease but is diagnosed by a physician as having a mild disease, the consequences will be the least serious - not so many medical resources will be wasted and the patient will not have so much stress.

It is important to incorporate different impacts of clinical mistakes into diagnostic modeling. However, defining a misdiagnosis costs matrix in a multi-label diagnosis scenario is difficult. Assuming that a patient can have or not have any of 8 diseases, then there are $2^8 = 256$ types of disease combinations. For each combination, there are 255 types of misdiagnoses, the total number of misdiagnoses is $256 \times 255 = 65280$. Therefore, I assume that expected misdiagnosis costs can be additive as the sum of the expected misdiagnosis costs of each disease category. With this assumption, I only need to consider two types of misdiagnosis costs in each disease category: (1) the misdiagnosis cost when the disease does exist but is ignored; (2) the misdiagnosis cost when the disease doesn't exist and but is mistakenly diagnosed to exist. A human expert only needs to estimate $8 \times 2 = 16$ misdiagnosis costs, then I can calculate any costs for any mistakes with the additive approach.

I elicited information about misdiagnosis costs from an emergency department physician, and these are listed in the Results Chapter. These expected misdiagnosis costs range from 1 to 10, based on the clinician's domain knowledge. By considering misdiagnosis costs, the model development process can give more weight to severe diseases misdiagnosis than to mild diseases misdiagnosis. The information about misdiagnosis costs was integrated with modeling using my algorithm, which is described in Section 3.3.

3.1.3 Knowledge of Clinical Workflow

When physicians perform disease diagnosis, they generally collect information in the following order: first, the triage nurse collects chief complaint information and demographic information; then a physician conducts an interview and asks some more detailed questions; then the physician does a physical exam; and at last, the physician orders some lab tests. Based on the information collected above, the doctor then makes a final diagnosis.

An expert emergency medicine doctor provided information that led us to group the features in our data set into four sources: the triage nurse source (phase 1), the physician interview source (phase 2), the physical exam source (phase 3) and the lab test source (phase 4).

When developing a sequential diagnosis tree, phase 1 features are searched first, followed

by phase 2, phase 3, and finally phase 4. Using this approach, the diagnostic activities (nodes in diagnosis tree) will align well with the existing clinical workflow.

I elicited this knowledge from an emergency department physician who has split all the features into those four phases. If a feature can appear in more than one phase, then it is included in all those phases. The knowledge about clinical workflow can be integrated into modeling using my algorithm that is described in Section 3.3.

3.2 A Bayesian Approach to Combine QMR Knowledge with EHR Data

In this section, I describe how medical knowledge in QMR and EHR data can be combined to develop disease diagnostic models that may perform better than models developed using pure EHR data. I developed a Bayesian approach that can use both medical knowledge and EHR data to develop disease diagnostic models. To provide a complete picture, I first describe the general process of developing a Naive Bayes model for disease diagnosis. Then I point out where and how I add QMR knowledge.

3.2.1 General Process of Developing a Disease Diagnostic Model

In my studies, the disease diagnostic models are in the format of a Bayesian network. Briefly, a Bayesian network is a type of probabilistic graphic model. A directed acyclic graph represents the correlation among variables (i.e., nodes in the graph). When there is an arc pointing from Node A to Node B, we say that Node A is one of the parents of Node B. There are numeric values (ranging from 0 to 1) associated with the nodes in the graph, which represent prob(child value | parents' value). I chose Bayesian network models because they can provide posterior probabilities of a disease given clinical findings and the probability estimation can be done with any set of clinical findings. When some clinical findings are not observed, a Bayesian network model can still make a probability estimation by averaging out all potential values of unknown variables. These important characteristics allow this type of model to deal with the uncertainty in the medical diagnostic process very well. Other

models, such as deep neural networks, cannot operate with missing value scenarios.

The model development process involved both feature selection and model building [114]. To develop a model for a disease A, the main steps include: (1) labeling each training sample as "disease A" or "not disease A." (2) Information gain measures expected entropy reduction and is a common measure in machine learning for measuring a feature's discriminative ability [35]. We sort candidate findings in descending order of the information gain score for disease A. (3) using a greedy forward method to add the feature into the feature set of the model - in each iteration, one feature is added into the feature set, and the corresponding Bayesian networks' average AUROC is calculated in five-fold cross validation experiments. If the average AUROC increases, then the feature is included in the feature set. Otherwise, the feature is not added to the feature set. The greedy forward process stops when no new feature can increase the average AUROC. (4) using the final feature set to develop a Bayesian network model for disease diagnosis.

3.2.2 Using A Bayesian Approach to Add QMR knowledge into EHR Modeling

Firstly, for each disease, I used all provided QMR simulation data and all clinical findings to develop a Naive Bayes model and assumed that the developed models can represent the probability relationships between clinical findings and the disease. I call these models QMR models. Then, I used a Bayesian approach to add the knowledge in these QMR models into the EHR modeling process. Using EHR data to develop disease diagnostic models follows the four steps in the last section. In steps 3 and 4, I added QMR-model knowledge when estimating conditional probabilities for a finding given a disease diagnosis.

Prior Equivalent Sample Size

For a binary finding X and a binary disease Y, this is how we estimate $P(X|Y)$. If we model the prior probability using a beta distribution and model the probability of the data using a binomial likelihood, the estimate we obtain is as follows:

$$\hat{P}(X = present|Y = present) = \frac{n + q \times K}{N + K}$$

where:

- n is the count in the EHR data in which X is present and Y is present.
- N is the count in the EHR data in which Y is present.
- q is the prior probability of $P(X = present|Y = present)$ that we derived from QMR models.
- K is a "prior equivalent sample size", which indicates how strongly we believe the prior; we can think of K as being the sample size of simulated data that are produced by the distribution $P(X = present|Y = present)$ that we derive from QMR.

M is a hyper-parameter not shown in the equation. let $K = M \times P(Y = present)$. then the formula above becomes:

$\hat{P}(X = present|Y = present) = \frac{n+P(X=resent|Y=present)\times M\times P(Y=present)}{N+M\times P(Y=present)}$

Here, $P(X = present|Y = present) \times M \times P(Y = present)$ is the count in the simulation data in which X is present and Y is present. $M \times P(Y = present)$ is the count in the simulation data in which Y is present.

With this Bayesian approach, I was able to add QMR-model knowledge into the EHR modeling process. I conducted many experiments using different sample sizes of EHR training data (N) and different sample sizes of simulation data (M), aiming to find out in what scenarios models learned with EHR data and augmented with QMR knowledge can perform better than models learned with pure EHR data. Experimental results and discussion are provided in the next chapter.

3.3 The Sequential Diagnosis Generating Algorithms (SDG)

An important goal of integrating domain knowledge into a machine learning process is to build a domain-meaningful model, such as a clinical alignment model. If a model is developed to help physicians do diagnosis, the model should be explainable in order to be understood by users (e.g., physicians). The model structure should be similar to physicians' existing workflow so that physicians can use it easily. The model should also make clinical sense so that physicians can trust it.

I chose the classification tree structure to provide sequential diagnosis support to clinicians, because the tree model is easily explainable and is like physicians' diagnostic reasoning process. There are many classic algorithms to develop tree structures. When selecting splitting nodes in the tree development process, most algorithms use some score criteria, such as Information Gain [78], Gain Ratio [79], or Gini index [13]. When a classification tree algorithm is used in medical diagnosis, misdiagnosis costs [30, 31, 51, 24] has also been considered as a selection criterion, because the most important thing for emergency department physicians is to rule out the most likely severe diseases to reduce the misdiagnosis costs. In my study, I also consider misdiagnosis costs in the tree model development process. The goal of using expected misdiagnosis costs as a criterion is to find out important features from the clinical perspective, instead of from an information theory or mathematical theory perspective.

For sequential diagnostic modeling, I developed two algorithms: the SDG-no-phase algorithm considers diagnostic accuracy and misdiagnosis costs, while the SDG-phase algorithm considers those two aspects as well as the knowledge of clinical workflow (as mentioned in Section 3.1.3). Both algorithms use population-level training data to develop a sequential diagnosis tree. In population machine learning, the developed model is optimized to perform well on average on all future individuals, which is different from personalized machine learning that develops models at a patient level [104]. More discussions about population machine learning and personalized machine learning are available in Section 5.4.6.

Next, I will introduce the general flow of SDG-no-phase algorithm, mathematical equations of searching score, and pseudo code of important functions. Then, I will introduce the SDG-phase algorithm, which is different from the SDG-no-phase algorithm in using knowledge of clinical workflow to restrict feature searching processes. Detailed pseudo code of SDG-phase and SDG-no-phase algorithms and related functions are provided at the end of this section. As shown in the pseudo code, the SDG-no-phase algorithm conducts greedy tree growth until at least one of the stopping criteria in every branch has been met (reaches a leaf node). The stopping criterion include: (1) the sample size of encounters in a partition of a leaf node (based on distinct values on the feature) is less than a threshold (minimal number of samples); (2) depth of the tree is greater than the maximum depth.

During the greedy tree growth process, the SDG-no-phase algorithm goes through all candidate features to find out the feature that has the largest score among all candidate feature (highest reduction of expected misdiagnosis costs) and chooses this feature as the root node of the tree. Then, the algorithm repeats the searching process in each sub portion to continuously build the tree model. The score of selecting a splitting node in each growing step of the diagnostic tree is defined as the reduction of expected misdiagnosis costs. To calculate this score, the related concepts and their relationships are as follows:

- d_i: a disease, $1 \leq i \leq$ number of candidate diseases
- E_x: a set of clinical evidence for one encounter x. $1 \leq x \leq$ number of encounters in a population dataset
- $P(d_i|E_x)$: a posterior probability of having a disease d_i given an encounter's evidence E_x
- $P(\bar{d}_i|E_x)$: a posterior probability of not having a disease d_i given an encounter's evidence
- $M(\bar{d}_i|d_i)$: the cost of mistakenly missing the diagnosis of a disease d_i when a patient has the disease d_i
- $M(d_i|\bar{d}_i)$: the cost of mistakenly diagnosing that a patient has disease d_i when the patient does not have the disease d_i
- $C_{d_i}(E_x)$: the expected misdiagnosis costs of one encounter x related to d_i given clinical evidence E_x. In training data, we already know the disease status of each encounter. If an encounter x has disease d_i but it is misdiagnosed as not having the disease, then the expected misdiagnosis cost is $M(\bar{d}_i|d_i)$ and the possibility of making this mistake is $P(\bar{d}_i|E_x)$. On the other hand, if an encounter x does not have disease d_i and is misdiagnosed to have the disease, then the expected misdiagnosis cost is $M(d_i|\bar{d}_i)$ and the possibility of making this mistake is $P(d_i|E_x)$. Using an indicator function, $I(d_{ix} = ture)$, which is 1 when encounter x has the disease d_i, and 0 otherwise. I can use a general expression to represent these two scenarios:
  $C_{d_i}(E_x) = I(d_{ix} = true) \times M(\bar{d}_i|d_i) \times P(\bar{d}_i|E_x) + (1 - I(d_{ix} = true)) \times M(d_i|\bar{d}_i) \times P(d_i|E_x)$
  Now, let me describe a perfect diagnostic model scenario. For each encounter, with sufficient evidence, a perfect probability model will estimate a probability of 1 for diseases that the encounter has, and will estimate a probability of 0 for diseases that the encounter

does not have. That is $P(d_i|E_x) = 1$ and $P(\bar{d_i}|E_x) = 0$ when d_{ix} is true; and $P(d_i|E_x) = 0$ and $P(\bar{d_i}|E_x) = 1$ when d_{ix} is false. In this situation, $C_{d_i}(E_x)$ is 0 for all diseases.

Another extreme situation is the worst diagnostic model scenario, when a probability model estimates a probability of 0 for diseases that the encounter has and estimates a probability of 1 for diseases that the encounter does not have. In this situation, $C_{d_i}(E_x) = I(d_{ix} = ture) \times M(\bar{d_i}|d_i) + (1 - I(d_{ix} = \text{true})) \times M(d_i|\bar{d_i})$

- $C(E_x)$: the expected misdiagnosis costs of one encounter given clinical evidence E_x. I assume that this expected misdiagnosis costs in a multi-label scenario is additive. The cost is the sum of each cost of each disease. $C(E_x) = \sum_{d_i} C_{d_i}(E_x)$
- $C(E)$: the expected misdiagnosis costs of a population dataset that includes multiple encounters. $C(E) = \sum_x C(E_x) = \sum_x \sum_{d_i} C_{d_i}(E_x) = \sum_{d_i} \sum_x C_{d_i}(E_x) = \sum_{d_i} \sum_{E_l} \{n_{d_i,E_l} \times P(\bar{d_i}|E_l) \times M(\bar{d_i}|d_i) + n_{\bar{d_i},E_l} \times P(d_i|E_l) \times M(d_i|\bar{d_i})\}$, where n_{d_i,E_l} is the number of patients who have disease d_i and have the same evidence value E_l, and $n_{\bar{d_i},E_l}$ is the number of patients who have the same evidence value E_l and do not have disease d_i. This was used in the function calcualteCost in pseudo code section.
- f_{jx}: a clinical feature, $1 \leq j \leq$ number of candidate features
- $R(f_{jx},\ E_x)$: the reduction in the expected misdiagnosis costs given evidence in an encounter level, when evidence includes E_x and a new feature f_j.
- f_j: a clinical feature at the population level
- E: a set of clinical evidence for a population dataset. $E = \{E_1, ..., E_n\}$.
- $R(f_j,\ E)$: the reduction in the expected misdiagnosis costs given evidence at the population level, when evidence includes E and a new feature f_j. $R(f_j, E) = C(E) - C(E \cup f_j)$ where $C(E \cup f_j) = \sum_x C(E_x \cup f_{jx})$, $C(E_x \cup f_{jx}) = \sum_{d_i} C_{d_i}(E_x \cup f_{jx})$, and $C_{d_i}(E_x \cup f_{jx}) = I(d_{ix} = ture) \times M(\bar{d_i}|d_i) \times P(\bar{d_i}|E_x \cup f_{jx}) + (1 - I(d_{ix} = \text{true})) \times M(d_i|\bar{d_i}) \times P(d_i|E_x \cup f_{jx})$ From these formulas, we can see that $R(f_j, E) > 0$ if $C(E \cup f_j) < C(E)$. The latter happens if including f_j can reduce the probabilities of making important mistakes in a population where the importance of mistakes is indicated by the expected misdiagnosis cost values: $M(\bar{d_i}|d_i)$ and $M(d_i|\bar{d_i})$.

Same as the SDG-no-phase algorithm, the SDG-phase algorithm uses the score, $R(f_j, E)$, to assess potential feature. The only difference is that SDG-phase algorithm restricts its

search on the best feature in a set of candidate features in phase 1 (the triage nurse phase) first until no feature in phase 1 can reduce the expected misdiagnosis costs. After reaching that point, the SDG-phase algorithm will then start finding the best feature in a set of candidate features in phase 2 (physician interview phase). After finishing all searching in phase 2, then phase 3 (physical exam phase), then phase 4 (lab test phase). In this way, the SDG-phase algorithm conducts feature searching based on the knowledge of different phases in ED (as mentioned in Section 3.1.3) so that a developed sequential tree can align well with existing clinical workflow. As shown in the pseudo code, the SDG-phase algorithm and the SDG-no-phase algorithm are different in the function that finds the best feature.

3.3.1 Pseudocode for SDG-no-phase and SDG-phase Algorithms

In the following pseudo code, main algorithms and functions are bolded. Comments are written in blue color.

ALGORITHM SDG-no-phase(dataset, depth, max-depth, leaf-min-sample-size, knownFeatureList, candidateFeatureList)

This is the SDG-no-phase algorithm. The data structure of a tree is a dictionary of dictionaries, each of which is also a dictionary of dictionaries, ..., and continue until the leaf node.

```
depth = depth + 1
bestFeature = findBestFeature-no-phase(dataset, knownFeatureList,
    candidateFeatureList)
tree={}
if (bestFeature is not NULL):
    tree = {bestFeature:{}}
    knownFeatureList.append(bestFeature)
    candidateFeatureList.remove(bestFeature)
```

On the next line, the splitDataset method splits data based on the values of best feature. Since each NLP-extracted clinical finding has two distinct values, present or absent (missing is categorized as absent), the dataset is split into two sub-datasets. One dataset includes encounters that have the "present" value of the best feature. Another dataset includes en-

counters with "absent."

leftData, rightData = splitDataset(dataset, bestFeature)

minimunSize = minimum(leftData.size, rightData.size)

if (minimunSize > leaf-min-sample-size and depth < max-depth):

Next two lines recursively call the SDG-no-phase algorithm to get a left and a right sub-tree. A deep copy of an object is a copy whose properties do not share the same references as those of the source object from which the copy was made.

tree[(bestFeature)][T] = **SDG-no-phase**(leftData, depth, max-depth,
leaf-min-sample-size, deepcopy(knownFeatureList),
deepcopy(candidateFeatureList))

The tree is a dictionary. The bestFeature is its key. The tree[(bestFeature)] is the value of the "tree" dictionary corresponding to the key "bestFeature". The tree[(bestFeature)] is also a dictionary. The tree[(bestFeature)][T] is the value of the "tree[(bestFeature)]" dictionary corresponding to the key "T." The "T" value denotes the "present" state of an NLP-extracted feature.

tree[(bestFeature)][F] = **SDG-no-phase**(rightData, depth, max-depth,
leaf-min-sample-size, deepcopy(knownFeatureList),
deepcopy(candidateFeatureList))

return tree

END ALGORITHM

FUNCTION findBestFeature-noPhase(dataset, knownFeatureList,
candidateFeatureList):

This function aims to find the best feature and is called by the SDG-no-phase algorithm. MC-parent is the expected misdiagnosis costs of a population dataset, denoted as C(E) in Section 3.3, where E is saved in the knownFeatureList.

MC-parent = **calculateCost**(dataset,knownFeatureList)

bestFeature is NULL

best-MC-reduction = 0

for feature in candidateFeatureList:

tempKnownFeatureList = deepcopy(knownFeatureList)

tempKnownFeatureList.append(feature)

leftData, rightData = splitDataset(dataset, feature)

After a feature f_j is temporally added to knowFeatureList, MC-children is the total expected misdiagnosis costs of two population datasets, $C(E, f_j = present)$ and $C(E, f_j = absent)$.

MC-children = **calculateCost**(leftData,tempKnownFeatureList) +

calculateCost(rightData,tempKnownFeatureList)

The reduction of expected misdiagnosis costs is calculated, which is the $R(f_j, E)$ in Section 3.3.

MC-reduction = MC-parent - MC-children

if MC-reduction > best-MC-reduction:

best-MC-reduction = MC-reduction

bestFeature = feature

print("found best feature: ", bestFeature, reduce ", best-MC-reduction)

return bestFeature

END FUNCTION

FUNCTION calculateCost(dataset,tempKnownFeatureList):

This function aims to calculate the expected misdiagnosis costs of a population dataset given a list of features as evidence.

if dataset.size==0:

return 0

else:

cost = 0

Because the cost is additive for each patient encounter x and also additive for each disease d_i, the function can firstly calculate a population cost for each disease and then sum up these costs. C(E)=$\sum_x C(E_x) = \sum_x \sum_{d_i} C(E_x, d_i) = \sum_{d_i} \sum_x C(E_x, d_i) = \sum_{d_i} \sum_{E_l} \{n_{d_i,E_l} \times P(\bar{d}_i|E_l) \times M(\bar{d}_i|d_i) + n_{\bar{d}_i,E_l} \times P(d_i|E_l) \times M(d_i|\bar{d}_i)\}$. Thus, the following codes have two for-loops, the first goes over disease, and the second goes over data.

for disease in diseaseList:

FeatureList2 = deepcopy(tempKnownFeatureList)

FeatureList2.append(disease)

In the next line, the dataset was grouped based on distinct value combinations of features in the feature list. Encounters with the same values in these features have the same posterior probability of a disease. This grouping slightly reduces running time.

tempDataFrame = dataset.groupby(FeatureList2).size().reset-index(name="count")

BayesianNetwork = BayesianNetworkTable[disease]

for index, record in tempDataFrame.iterrows():

record-true-status = record[disease]

Given the true disease status and a list of features, this calculateMisdiagnosisP method returns the probabilities of making a mistake. If the true disease status of d_i for this group of patients is Yes, then it returns $P(\bar{d}_i|E_l)$. And the getMisCost method looks over a pre-saved dictionary and returns $M(\bar{d}_i|d_i)$. If the true disease status is No, then the calculateMisdiagnosisP function returns $P(d_i|E_l)$ and the getMisCost method returns $M(d_i|\bar{d}_i)$.

misProb = calculateMisdiagnosisP(BayesianNetwork, record,

tempKnownFeatureList, record-true-status)

misCost = getMisCost(disease,record-true-status)

cost = cost + misProb × misCost × record["count"]

return cost

END FUNCTION

ALGORITHM SDG-phase (dataset, depth, max-depth, leaf-min-sample-size, knownFeatureList, candidateFeatureList, phase, phase1List, phase2List, phase3List,

phase4List)

The SDG-phase algorithm is different from SDG-no-phase algorithm in using phase information as restrictions to find the best feature. When firstly calling this algorithm, phase was initiated as 1.

depth = depth + 1

bestFeature, currentPhase = **findBestFeature-phase** (dataset, knownFeatureList,

candidateFeatureList, phase, phase1List, phase2List, phase3List, phase4List)

tree={}

if (bestFeature is not NULL):

tree = {bestFeature:{}}

knownFeatureList.append(bestFeature)

candidateFeatureList.remove(bestFeature)

leftData, rightData = splitDataset(dataset, bestFeature)

minimunSize = minimum(leftData.size, rightData.size)

if (minimunSize > leaf-min-sample-size and depth < max-depth):

Recursively call the SDG-phase algorithm to get a left sub-tree.

tree[(bestFeature)][T] = **SDG-phase**(leftData, depth, max-depth,
leaf-min-sample-size, deepcopy(knownFeatureList),
deepcopy(candidateFeatureList), phase, phase1List,
phase2List, phase3List, phase4List)

tree[(bestFeature)][F] = **SDG-phase**(rightData, depth, max-depth,
leaf-min-sample-size, deepcopy(knownFeatureList),
deepcopy(candidateFeatureList), phase, phase1List,
phase2List, phase3List, phase4List)

return tree

END ALGORITHM

FUNCTION findBestFeature-phase(dataset, knownFeatureList, candidateFeatureList, phase, phase1List, phase2List, phase3List, phase4List):

This function aims to find the best feature among candidate feature list that was an intersection of the candidate feature list and the current phase list. This is the only difference between SDG-phase algorithm and SDG-no-phase algorithm. Other functions and methods, such as calcualteCost and splitDataset are exactly the same.

MC-parent = **calculateCost**(dataset,knownFeatureList)

bestFeature is NULL

best-MC-reduction = 0

currentPhase = phase

if phase==1:

candidateFeatureList-phase = intersection(candidateFeatureList, phase1List)

elif phase==2:

candidateFeatureList-phase = intersection(candidateFeatureList, phase2List)

elif phase==3:

candidateFeatureList-phase = intersection(candidateFeatureList, phase3List)

elif phase==4:

candidateFeatureList-phase = intersection(candidateFeatureList, phase4List)

for feature in candidateFeatureList-phase:

tempKnownFeatureList = deepcopy(knownFeatureList)

tempKnownFeatureList.append(feature)

leftData,rightData = splitDataset(dataset, feature)

MC-children = **calculateCost**(leftData, tempKnownFeatureList) +

calculateCost(rightData, tempKnownFeatureList)

MC-reduction = MC-parent - MC-children

if MC-reduction > best-MC-reduction:

best-MC-reduction = MC-reduction

bestFeature = feature

If there is no best feature found in one phase, then the phase number is added 1 and the search goes to the next phase as shown in the codes below.

if bestFeature is NULL and phase $\leq$3:

currentPhase = phase + 1

bestFeature, currentPhase = **findBestFeature-phase**(dataset, knownFeatureList,

candidateFeatureList, currentPhase, phase1List, phase2List, phase3List, phase4List)

print("found best feature: ", bestFeature, "reduce ", best-MC-reduction)

return bestFeature, currentPhase

END FUNCTION

FUNCTION classifytest(inputTree, testDataSet):

This function can be used to leverage a learned tree (a dictionary of dictionaries) to conduct

patient diagnosis when patient encounters are in testing data and their disease statuses are unknown. This function goes through all encounters in a testing dataset one by one. For each encounter, it firstly adds the demographic features as known features. Then, it calls the classify function to obtain probabilities of diseases for one encounter, which are then saved into a dataframe. Most codes are in python format. pd is the abbreviation of the pandas package.

```
    column-names = diseaseList
    df = pd.DataFrame(columns = column-names)
    for index, testVec in testDataSet.iterrows():
        knowFeatureValueDict = {}
        for demoFeature in demoList:
            knowFeatureValueDict[demoFeature] = testVec[demoFeature]
        probDic = classify(inputTree, testVec, knowFeatureValueDict)
        probValueList = [ ]
        for disease in diseaseList:
            probValueList.append(probDic[disease])
        df.loc[len(df.index)] = probValueList
    return df
END FUNCTION
```

FUNCTION classify(inputTree, testVec, knowFeatureValueDict):

This function returns probabilities of diseases for an individual encounter. It directly calls the calculateProbDic function, unless the diagnosis tree can be continuously traced down (recursively calling the calculateProbDic function).

```
    firstKey = list(inputTree.keys())[0]
    firstFeature = list(inputTree.keys())[0]
    knowFeatureValueDict[firstFeature] = testVec[firstFeature]
    secondDict = inputTree[firstKey]
    probDic = {}
    if len(list(secondDict.keys())) > 0:
```

```
        if testVec[firstFeature] not in secondDict.keys():
            probDic = calculateProbDic(knowFeatureValueDict)
        else:
            for key in secondDict.keys():
                if testVec[firstFeature] == key:
                    if secondDict[key] is not NULL:
                        probDic = classify(secondDict[key], testVec, knowFeatureValueDict)
                    else:
                        probDic = calculateProbDic(knowFeatureValueDict)
    else:
        probDic = calculateProbDic(knowFeatureValueDict)
    return probDic
END FUNCTION
```

FUNCTION calculateProbDic(knowFeatureValueDict):

This function uses pre-learned Bayesian networks to calculate posterior probabilities given clinical evidence.

```
    probDic={}
    for disease in diseaseList:
        BayesianNetwork = BayesianNetworkTable[disease]
        prob-dieaase-true = BayesianNetwork.getProbabilityDiseaseTrue(knowFeatureValueDict)
        probDic[disease] = prob-dieaase-true
    return probDic
END FUNCTION
```

FUNCTION printAllTestPath(inputTree, testDataSet):

This function uses the eight CCS categories scenario as an example to show how to print a diagnosis path and probabilities of diseases for each encounter in a testing dataset. The classifyAndPrintPath function is the one that returns probabilities and a string of a diagnosis path.

```
    column-names = diseaseList
    df = pd.DataFrame(columns = [”path”, ”d721-p”, ”d722-p”, ”d723-p”, ”d724-p”,
        ”d726-p”, ”d727-p”,”d7289-p”, ”d7211-p”])
    for index, testVec in testDataSet.iterrows():
        knowFeatureValueDict = {}
        initialPrint=“”
        for demoFeature in demoList:
            knowFeatureValueDict[demoFeature] = testVec[demoFeature]
            initialPrint=initialPrint+testVec[demoFeature]+”—”
        probDic, toPrint = classifyAndPrintPath(inputTree, testVec,
            knowFeatureValueDict, initialPrint)
        df = df.append('path' : toPrint, 'd721-p' : probDic['d721'], 'd722-p' : probDic['d722'],
            'd723-p' : probDic['d723'],'d724-p' : probDic['d724'],'d726-p' : probDic['d726'],
            'd727-p' : probDic['d727'], 'd7289-p' : probDic['d7289'],'d7211-p' : probDic['d7211'],
            ignore-index = True)
    return df
END FUNCTION
```

FUNCTION classifyAndPrintPath(inputTree, testVec, knowFeatureValueDict, toPrint):
This function returns a diagnosis path and probabilities of diseases for an individual encounter. The probability portion is the same as as the classify function.

```
    firstKey = list(inputTree.keys())[0]
    firstFeature = list(inputTree.keys())[0]
    if firstFeature not in treeFeatureList:
        treeFeatureList.append(firstFeature)
    if firstFeature in nameDic:
        toPrint=toPrint+” -> ” + str(nameDic[firstFeature]) +”(”+testVec[firstFeature]+”)”
    else:
        toPrint=toPrint+”-> ” + firstFeature +”(”+testVec[firstFeature]+”)”
    knowFeatureValueDict[firstFeature] = testVec[firstFeature]
```

```
    secondDict = inputTree[firstKey]
    probDic = {}
    if len(list(secondDict.keys())) > 0:
        if testVec[firstFeature] not in secondDict.keys():
            probDic = calculateProbDic(knowFeatureValueDict)
        else:
            for key in secondDict.keys():
                if testVec[firstFeature] == key:
                    if secondDict[key] is not NULL:
                        probDic = classifyAndPrintPath(secondDict[key], testVec,
                                knowFeatureValueDict, toPrint)
                    else:
                        probDic = calculateProbDic(knowFeatureValueDict)
    else:
        probDic = calculateProbDic(knowFeatureValueDict)
    return probDic
END FUNCTION
```

3.4 Using ML-C4.5 to Generate a Sequential Diagnosis Tree

Clare et al. [17] proposed a C4.5 (ML-C4.5) algorithm to handle multi-label classification problems. I applied the ML-C4.5 algorithm to generate a sequential diagnosis tree. As with a traditional classification tree, the sequential diagnosis tree growing process mainly depends on greedy searching the next best node. The next best node is determined based on which node can reduce the empirical entropy the most. Before splitting data, the empirical entropy is calculated with training data using

$$Entropy(C) = -\sum_{c_i \in C}[P(c_i)log_2P(c_i) + (1 - P(c_i))log_2(1 - P(c_i))]$$

where C is a set of diseases; c_i is each disease, each disease is treated as binary; $P(c_i)$ is the probability (relative frequency) of disease class c_i; and 1-$P(c_i)$ is the probability of not being a member of disease class c_i. When $P(c_i)$ equals 0 or 1, we cannot calculate $P(c_i)log_2P(c_i)$ or $(1-P(c_i))log_2(1-P(c_i))$. In these situations, I define this portion to be 0 for disease class c_i.

After splitting the data using a candidate feature A, the entropy becomes:

$$Entropy(C, A=a) = -\sum_{c_i \in C}[P(c_i,a)log_2P(c_i,a) + (1-P(c_i,a))log_2(1-P(c_i,a))]$$

The reduction of entropy is:

$$Entropy(C) - \sum_{a \in A}\frac{N(a)}{N}Entropy(C,a)$$

After using entropy reduction as score to find the best next node, the algorithm divides the training data into a few subsets according to the values of the chosen node. Then, in each of these subsets, repeat previous step until get a complete classification tree. In the leaf nodes, we use the probability (relative frequency) of each disease as the estimated probabilities for each disease, we use this probability to do prediction.

When using the generated tree for diagnosis, we use a patient's data to follow the path until no more nodes can be filled with clinical evidence and use the probability distributions in the last node as the estimated probabilities for each disease. With these probabilities, we can evaluate the performance of the generated tree using a testing dataset. Moreover, we can ask human experts to review the diagnostic path of a few samples for clinical alignment evaluation.

3.5 Using Deep Q Learning to Generate a Sequential Diagnosis Policy

Reinforcement learning algorithms are designed to solve sequential decision-making problems. In this section, I firstly introduce the main concepts of one popular reinforcement learning algorithm, deep Q learning. Then, I describe how to apply deep Q learning to

suggest a sequence of actions in medical diagnosis. Finally, I briefly present the settings of using deep Q learning in my research experiments. More details about experiment settings and results are provided in the next chapter.

Deep Q learning is one type of Q learning algorithm. In Q learning (as introduced in the reinforcement learning section of the Literature Review Chapter), the Q function is an action-value function that indicates the expected cumulative discounted reward given a state s, action a, and policy π. Corresponding to the best policy, the **optimal** $Q-value$ **function** is $Q^*(s,a) = max_\pi Q^\pi(s,a) = max_\pi E[\sum_{t\geq 0} \gamma r_t | s_0 = s, a_0 = a, \pi]$. According to Bellamn's theory, optimal $Q-value$ function also satisfies: $Q^*(s,a) = E_{s'\sim\epsilon}[r+\gamma \times max_{a'} Q^*(s',a')|s,a]$. Therefore, Q^* function suggests the **optimal policy** π^*, which (in any state s) takes the best action that has the highest value of $Q^*(s,a)$.

To approximate a Q function, deep Q learning uses a deep neural network as the function approximator. After choosing an appropriate Q network architecture, the learning process becomes an optimization process that aims to find the optimal Q-function, a Q-function that satisfies the Bellman Equation. Because learning from a set of consecutive instances is problematic: instances are correlated; current parameters of Q network directly determine next training instances, and this may lead to bad feedback loops, the algorithm uses an experience replay strategy. After initializing a Q network with random parameters, the algorithm goes through multiple episodes. Each episode includes an exploration stage and an optimization stage. In the **exploration stage**, the agent often selects the action that maximizes the estimated Q value in the current state, takes the action, receives the immediate reward, and observes the next state. This generates an instance of transition. Then, the algorithm stores this instance to a replay memory repository. With a small probability, the agent can select a random action instead of the best action. This exploration is important especially at the early episode when the Q network is not close to optimal. Then, in the **optimization stage**, from the replay memory repository, the algorithm randomly selects a set of transitions and use them to update the parameters of the Q network. When conducting the optimization, in the **forward pass**, for a non-terminal state, y_i, the estimated cumulative reward for each case by the Q network, is calculated using the Bellman Equation and the target network (the Q network saved in last episode). In the **backward pass**, using y_i as a

silver standard, the algorithm performs a gradient descent step to update the parameters of the Q network to minimize the square error between y_i and the estimated Q value for each instance in the selected experience replay set. Note that in Mnih et al. deep Q learning paper [60], θ in the formula $r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta)$ corresponds to the parameters in the Q network that have been saved in the previous episode. While in the formula $(y_j - Q(\phi_j, a_j; \theta))^2$, θ corresponds to the parameters that are being updated in a gradient descent step .

Wei et al. [110] developed a deep Q learning-based medical dialogue system that can collect additional symptoms through system-patient conversations before making final diagnosis. In this system, the main components of the Markov decision process mentioned in literature review (S, A, P, R, γ) are the following:

- State S: includes informed symptoms until the current time t (represented in SNOMED CT codes, with three values: positive, negative, not-mentioned), the previous action of the user (i.e., patient), the previous action of the agent (i.e., digital doctor), and the user-agent dialogue information.
- Action A is composed of a dialogue act (e.g., inform, deny) and a slot (i.e., query about a symptom or make a diagnosis from a list of candidate diseases). The inquiry system terminates when the action makes a diagnosis (regardless of whether it is correct or not).
- Reward R is the immediate reward at step t after an action is taken.
- Discount factor γ represents the discount for non-immediate award.
- Transition probability P is not defined here because deep Q learning is a model-free approach.

In my dissertation, I leverage Wei et al. implementation of a dialogue system [109] in order to compare deep Q learning with my algorithm. Details about hyper-parameters and results are provided in the Results Chapter.

4.0 Experimental Design and Results

This research aims to integrate expert-defined medical knowledge with EHRs to build disease diagnostic models, as well as develop and test a sequential diagnosis generating algorithm that considers diagnostic accuracy, misdiagnosis costs, and clinical workflow. The Introduction Chapter briefly described research background, motivation and aims. The Literature Review Chapter presented many strategies that have been used to incorporate domain knowledge into machine learning. The Methodology Chapter described the domain knowledge used in my study and the approaches I used to include domain knowledge into machine learning algorithms. This chapter connects all the dots and shows the research data, experiment design and results of experiments testing multiple hypotheses under two research aims.

4.1 Research Data

In this section, I first introduce the clinical classifications software (CCS), which is used to group many clinically similar ICD codes into one CCS category. This CCS grouping enabled us to have sufficient research samples for model development and evaluation. Then I describe research datasets from two resources and provide summative statistics of the UPMC emergency department data and the QMR-simulation data.

4.1.1 The Clinical Classifications Software (CCS)

Of the over 14,000 diagnosis codes and 3,900 procedure codes in the International Classification of Diseases, 9th Revision, Clinical Modification (ICD-9-CM) system, I used the 239 related to heart diseases to test my algorithms. However, developing a model for each of the 239 codes for the diseases would require training samples of a very large size, which were not available in my research, so I needed to group these 239 codes into several clinically

similar groups. To create these groups, I chose to use the Clinical Classifications Software (CCS) [23] for ICD-9-CM, a diagnosis and procedure categorization scheme that groups more than 4,000 ICD codes into a smaller number of categories. Codes in each CCS category are clinically similar.

Grouping ICD codes into CCS categories allowed me to have sufficient samples for model development and evaluation. Altogether, the 239 heart-disease related ICD-9-CM codes I chose to include could be grouped into 11 CCS categories (Table 2).

Table 2: Heart Disease CCS Categories

CCS category	**CCS definition**
7.2.1	Heart valve disorders
7.2.2	Peri-; endo-; and myocarditis; cardiomyopathy
7.2.3	Acute myocardial infarction
7.2.4	Coronary atherosclerosis and other heart disease
7.2.5	Nonspecific chest pain
7.2.6	Pulmonary heart disease
7.2.7	Other and ill-defined heart disease
7.2.8	Conduction disorders
7.2.9	Cardiac dysrhythmias
7.2.10	Cardiac arrest and ventricular fibrillation
7.2.11	Congestive heart failure; nonhypertensive

4.1.2 UPMC Emergency Encounter Data

Heart Disease Visits

We retrieved data from all heart disease visits from 15 emergency departments at the University of Pittsburgh Medical Center (UPMC) between January 1, 2008 and December

31, 2014. The University of Pittsburgh Institutional Review Board approved this study (Study No.18100069). I identified a visit as a cardiac disease visit if its primary ED diagnosis included one of the 239 ICD-9-CM codes for cardiac diseases. Based on a physician's suggestion, I made the following changes to the categorization list. I deleted category 7.2.5 (nonspecific chest pain) because it is not a specific heart disease; I deleted category 7.2.10 (cardiac arrest and ventricular fibrillation) because it is very easy to do diagnosis and physicians cannot make a mistake with this diagnosis. I merged category 7.2.8 and category 7.2.9 into one category, which I named 7.2.89, because they are very similar. After these modifications to the original list of the 11 CCS categories, eight categories were used for the research data: 7.2.1, 7.2.2, 7.2.3, 7.2.4, 7.2.6, 7.2.7, 7.2.89, and 7.2.11. I used these 8 categories in my research experiments. Detailed information about ICD codes and their associated CCS categories are provided in Appendix A. Table 3 lists counts of encounters in each of eight categories in the EHR data. If one patient has more than one CCS category heart disease, it will be counted in all related CCS categories. For example, one patient has heart disease CCS7.2.1 and CCS7.2.2. When we calculate the count number of category CCS7.2.1, this patient will be included; when we calculate the count number of category CCS7.2.2, this patient will also be included.

Figure 2 shows each step in the data processing workflow. From the 141,660 encounters that have a primary diagnosis code in one of the 239 ICD-9-CM codes, I first excluded CCS 7.2.5, CCS 7.2.10 and encounters without complete demographic information and ED clinical notes. I then divided the remaining encounters into a training dataset for model development (43,910 encounters between 2008 and 2013) and a test dataset for model evaluation (13,444 encounters in 2014).

Table 3: Counts of Encounters in CCS Categories in EHR Data

CCS category	**CCS definition**	**N of Encounters**
7.2.89	Conduction disorders and cardiac dysrhythmias	32,879
7.2.4	Coronary atherosclerosis and other heart disease	26,456
7.2.11	Congestive heart failure; nonhypertensive	9,461
7.2.1	Heart valve disorders	7,147
7.2.2	Peri-; endo-; and myocarditis; cardiomyopathy	3,736
7.2.6	Pulmonary heart disease	3,087
7.2.7	Other and ill-defined heart disease	1,603
7.2.3	Acute myocardial infarction	1,581

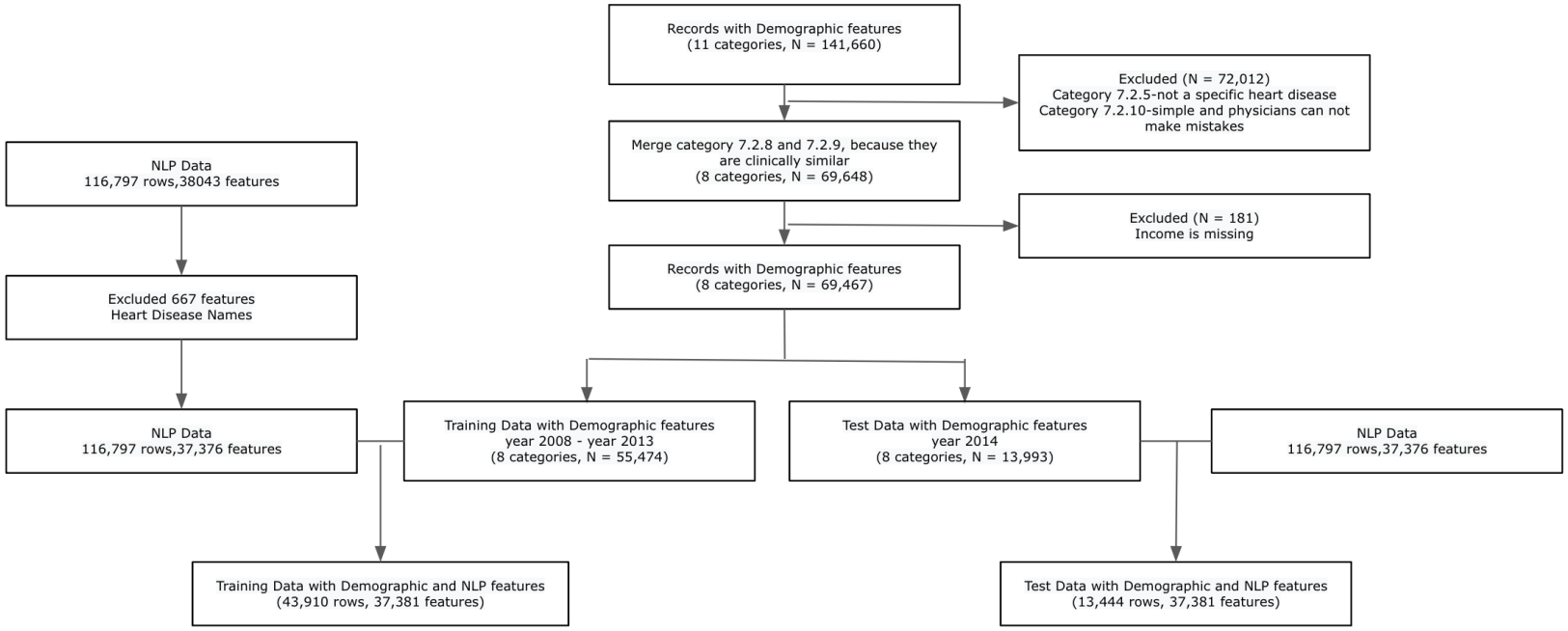

Figure 2: UPMC Data Preprocessing

Demographic Features

Basic structured demographic information from the cohort population includes biological sex (male, female), race (white, black, other), age group (0-17, 18-64, 65+), insurance (commercial, Medicare, other), and income (low, middle, high). We estimated patients' household income based on two variables: home zip code and census data for median income for the area. The income information was then categorized into three groups.

NLP-extracted Features

To include more clinical findings, we collected unstructured ED discharge summaries of retrieved UPMC encounters. From the ED discharge summaries, we extracted medical findings using MedLEE [26], a well-known NLP tool. For each report, MedLEE returned a set of Unified Medical Language System Concept Unique Identifiers (UMLS CUIs), each of which represents a clinical concept. Each concept was associated with a certainty level. For each clinical concept CUI, I assigned "Present" if the certainty value is "yes", "high certainty", "moderate certainty", "positive", or "no negation." I assigned "Absent" if the certainty value is "ignore", "no", "low certainty", "very low certainty", "negative", or "rule out."

If a CUI was not mentioned in a report, we also treated it as "Absent." I made this assumption based on the results of a preliminary study. In that study, I set "Not Mentioned" to be a new value, so each CUI had three values ("Present","Absent", or "Not Mentioned"). I used a classic classification tree algorithm (ID3) and EHR data to generate a classification tree model. The pure data-driven tree had many nodes that combined "Not Mentioned" with "Present" as one group ("Not Mentioned or "Present") and used "Absent" as the other group to divide patient population for disease diagnosis purpose. After reviewing the generated tree, a physician expert commented that most of these machine learned patterns of using "Not Mentioned" and "Present" together as one combination ("Not Mentioned" or "Present") did not make any clinical sense for disease diagnosis. To avoid these patterns, I assumed "Not Mentioned" as "Absent" in this dissertation research. With this assumption, all NLP-extracted clinical features were binary: "Present" or "Absent" in my experiemnts.

From the NLP-retrieved clinical findings, I further removed heart-disease related CUIs

so that these features would not be used for diagnosis. Clinical notes may already contain diagnostic information. Using diagnosis related UMLS codes as candidate features in a computational diagnostic system, its potential performance may be overestimated in real-world clinical practice. On the other hand, historical evidence about heart disease is important information for diagnosis. My research data cannot distinguish whether the disease information in clinical notes is historical evidence or a current diagnosis. To provide a fair evaluation of algorithms, I manually reviewed descriptions of candidate features and removed 667 unique CUIs that were heart disease names and had the semantic type 'disease or symptom' or 'pathologic function' in the UMLS 2019AA version, such as C0085610 (sinus bradycardia).

4.1.3 QMR Simulation Data for Heart Diseases

The simulated heart disease data include 125,659 records, involving 51 primary diagnoses, 100 secondary diagnoses. The secondary diagnoses are diseases that may appear in patients with the primary diagnosis, some of them are heart diseases, some of them are not heart diseases. There are 51 heart diseases in the QMR knowledge base. A collaborator, Dr. Randolph A. Miller, helped to set 51 heart diseases as a primary diagnosis and other possible diseases as a secondary diagnosis to generate simulation data.

Figure 3 shows the steps in the data processing workflow. In the simulation data, diseases and clinical findings are all in free-text description format, such as "acute myocardial infarction", "chest pain". However, the training data that I retrieved from EHR contain diseases annotated in ICD9-CM codes, as well as clinical findings coded in UMLS codes. Therefore, for each simulated record, the definitions of appearing features were concatenated together to create a report. Then, cTakes software [85] was used to parse each report to obtain medical CUI codes. The cTakes is a clinical natural language processing tool. It combines rule-based and machine learning techniques to extract information from clinical tests. In total, 1204 unique CUI codes appeared in the simulation data, of which 864 unique codes also appeared in the UPMC EHR data (37,961 unique CUI codes). Each free-text disease description I mapped to ICD-9-CM codes and then to a CCS category. After these preprocessing steps, the format for the simulation data and UPMC data was the same.

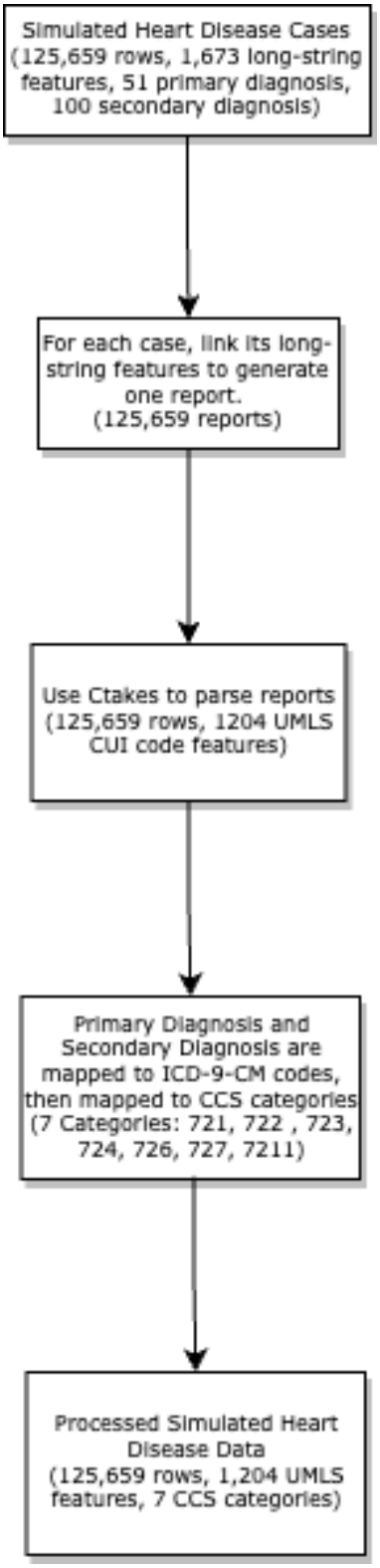

Figure 3: QMR Simulation Data Preprocessing

4.2 Experiment Settings and Results for Research Aim 1

Research Aim 1. Integrate Expert-Defined Medical Knowledge with EHR Data to Perform Disease Diagnoses

The task of combining medical knowledge with EHR was not easy. Medical knowledge is usually published in many different formats, such as medical textbooks, peer-reviewed medical articles (e.g., PubMed repository), and medical commercial websites (e.g., Upto-Date). It is challenging to compile medical knowledge and represent them in a coherent way that can be easily integrated with EHR data, which are often in their own format. In this dissertation, I only focus on medical knowledge that is represented in an expert-defined diagnostic system, the QMR system. I use the QMR system as an example and test four research hypotheses. Since QMR simulation data do not have CCS 7.2.89, experiments in aim 1 focused on 7 CCS categories (CCS 7.2.1, 7.2.2, 7.2.3, 7.2.4, 7.2.6, 7.2.7, 7.2.11).

1a.When EHR sample size is small, models trained using EHR data only are less accurate than models trained using small sample size EHR data and medical knowledge.

In each research hypotheses section, I first introduce the experiment setting, then provide results.

Experiment Setting

For each CCS category, C_i, in (CCS 7.2.1, 7.2.2, 7.2.3, 7.2.4, 7.2.6, 7.2.7, 7.2.11), I developed diagnostic models and evaluated their performance using the following steps:

[1] Generated an EHR training dataset, D_{ie}, by randomly sampling 1 percent of encounters (439 encounters), from the whole training dataset (i.e., UPMC heart-disease related ED encounters between 2008 and 2013). In this way, I simulated a scenario when the EHR sample size is small.

[2] Used the EHR training dataset, D_{ie}, to develop a Naive Bayes model, M_{ie}, through the general process described in Section 3.2.1 ("EHR model").

[3] Used all QMR-generated synthetic samples, D_{iq}, to develop a Naive Bayes model, M_{iq}, through the general process described in Section 3.2.1.

[4] Used the QMR model, M_{iq}, and EHR training dataset, D_{ie}, to develop knowledge augmented models, M_{ieqm}, which differ in the equivalent sample size M. The model development followed the Bayesian approach that is described in Section 3.2.2. ("EHR-knowledge model")

[5] Compared performance of M_{ie} with M_{ieqm}, where performance was measured using the Areas Under the Receive Curve (AUROC) in the whole EHR test dataset (i.e., UPMC heart-disease related ED encounters in 2014). the Delong's test was conducted to calculate statistical significance.

Experimental Results

Table 4 shows the performance for the different models. The first column is sample size of EHR data. The second column is the equivalent sample size. When M equaled 0, final models were developed using EHR data only. When M was greater than 0, final models were developed using EHR data and the QMR model. Other columns list AUROCs of M_{ieqm} models that were developed using EHR data and QMR models. We used the Delong method in the R package "pROC" to compare AUROC values, threshold was 0.05. If a M_{ieqm} model's AUROC is statistically significantly higher than a M_{ie} model's AUROC (the second row in the same column, bolded in the table), then the cell is colored green. If an M_{ieqm} model's AUROC is statistically significantly lower than a M_{ie} model's AUROC, then the cell is colored red. If an M_{ieqm} model's AUROC is not statistically significantly different from the M_{ie} model's AUROC, then the cell is not colored.

Results in Table 4 show that most EHR-knowledge models for CCS 7.2.6 (pulmonary heart disease) and CCS 7.2.11 (congestive heart failure; nonhypertensive) perform statistically significantly better than EHR models. For CCS 7.2.2 (peri-; endo-; and myocarditis; cardiomyopathy), CCS 7.2.3 (acute myocardial infarction), CCS 7.2.4 (coronary atherosclerosis and other heart disease), when assigning a relatively small weight to QMR knowledge, the developed EHR-knowledge models often perform better. When assigning a large weight to QMR knowledge, the performance of developed EHR-knowledge models for CCS 7.2.1 (heart valve disorders) and CCS 7.2.2 (peri-; endo-; and myocarditis; cardiomyopathy) dropped significantly.

Table 4: Performance Comparison of EHR Models and EHR-knowledge Models (EHR Sample Size is Small).

N	M	CCS 7.2.1	CCS 7.2.2	CCS 7.2.3	CCS 7.2.4	CCS 7.2.6	CCS 7.2.7	CCS 7.2.11
439	0	**0.772**	**0.723**	**0.809**	**0.872**	**0.664**	**0.527**	**0.857**
439	44	0.772	0.736	0.838	0.876	0.762	0.529	0.873
439	220	0.77	0.717	0.836	0.87	0.78	0.524	0.864
439	329	0.771	0.709	0.827	0.876	0.764	0.515	0.87
439	439	0.765	0.716	0.824	0.871	0.745	0.526	0.86
439	659	0.759	0.701	0.821	0.872	0.776	0.526	0.867
439	878	0.748	0.704	0.818	0.862	0.774	0.527	0.871
439	1,317	0.744	0.689	0.812	0.874	0.757	0.527	0.865
439	1,756	0.744	0.686	0.793	0.876	0.777	0.532	0.866
439	2,196	0.75	0.702	0.795	0.874	0.766	0.527	0.871
439	2,635	0.749	0.701	0.793	0.878	0.754	0.526	0.868
439	3,074	0.747	0.705	0.792	0.871	0.758	0.526	0.87
439	3,513	0.747	0.691	0.79	0.873	0.758	0.53	0.87
439	3,952	0.746	0.691	0.789	0.867	0.742	0.53	0.87
439	4,391	0.737	0.692	0.788	0.867	0.749	0.53	0.869
439	8,782	0.726	0.717	0.784	0.869	0.767	0.53	0.871
439	17,564	0.728	0.708	0.79	0.86	0.748	0.517	0.872
439	26,346	0.733	0.696	0.786	0.859	0.747	0.518	0.86
439	35,128	0.732	0.737	0.784	0.863	0.771	0.53	0.869
439	43,910	0.728	0.73	0.783	0.868	0.771	0.528	0.87

1b. When EHR sample size is small, models trained using EHR data only are less clinically aligning than models trained using small sample size EHR data and medical knowledge.

Experiment Setting

Clinical alignment is important, but it is not easy to measure. In Aim 1, I mainly used AUROC and number of noisy features as measures of clinical alignment how much a clinician's opinions of features importance agree with the developed models. Specifically, for a CCS category, C_i, I compared M_{ie} with M_{ieqm} ($N = 439, M = 44$) using the following steps:

[1] From a model M_{ie}, got its features, S_{ie}, and their clinical descriptions

[2] From a model M_{ieqm}, got its features, S_{ieqm}, and their clinical descriptions

[3] Got the union of S_{ie} and S_{ieqm} and requested a physician expert to annotate whether each feature was an influential feature for the diagnosis of CCS category, C_i. Labelled 1 if yes; otherwise, labelled 0. This is a single-blind experiment design. For each feature, the physician did not know which model a feature came from.

[4] I used two approaches to handle the situation that an annotated feature did not appear in one model. In the **first** approach, for each feature in the union set, if it did not appear in model M_{ie}, I deleted it; if it appeared in model M_{ie}, I extracted $P(finding = true|disease = true)$ from the model M_{ie}. Then, I used features' $P(finding = true|disease = true)$ and the physician's annotations to calculate an AUROC, which is EHR model performance in Table 5. I used the same method to process features in each EHR-knowledge model to calculate the performance, which is also listed in Table 5. I then compared these two AUROCs using the Delong's method and reported P-values in Table 5. A high AUROC value indicates that the model's perception of the relationships between variables and a disease aligns well with that of a physician.

[5] In the **second** approach, for each feature in the union set, if it appeared in model M_{ie}, I extracted $P(finding = true|disease = true)$ from the model M_{ie}; if it did not appear in model M_{ie}, I assigned $P(finding = true|disease = true) = 0$. Then, I used features' $P(finding = true|disease = true)$ and the physician's annotations to calculate an AUROC, which is EHR model performance in Table 6. With the same approach, I calculated the performance of EHR-knowledge model in Table 6, which also includes p-values for comparisons.

[6] Moreover, I compared number of noisy features as another indication of clinical alignment. I used the two-sample sign test to calculate statistical significance.

Experimental Results

Using these steps, I obtained models' performance and comparisons listed in table 5 and table 6. In table 5, the AUROCs of EHR-knowledge models were higher than the AUROCs of EHR models. However, the differences were not statistically significant. In table 6, the AUROCs of EHR-knowledge models were similar to the AUROC of EHR models, and there was no statistically significant differences.

A high number of noisy features indicates that the model does not align well with physicians' perspective. Table 7 shows that EHR-knowledge models had fewer noisy features than EHR models. The two-sample sign test showed that the number of noisy features in EHR models was statistically greater than the number of noisy features in EHR-knowledge models (p=0.0156).

Table 5: Clinical Meaningfulness of Nodes in EHR Models and EHR-knowledge Models (EHR Sample Size is Small, Delete Missing).

model	CCS 7.2.1	CCS 7.2.2	CCS 7.2.3	CCS 7.2.4	CCS 7.2.6	CCS 7.2.7	CCS 7.2.11
EHR model	0.513	0.607	0.451	0.612	0.538	0.445	0.621
EHR-knowledge model	0.520	0.697	0.619	0.634	0.616	0.5	0.754
P-value	0.6819	0.2518	0.3994	0.6963	0.5072	0.7254	0.06435

Table 6: Clinical Meaningfulness of Nodes in EHR Models and EHR-knowledge Models (EHR Sample Size is Small, Missing as Zero).

model	CCS 7.2.1	CCS 7.2.2	CCS 7.2.3	CCS 7.2.4	CCS 7.2.6	CCS 7.2.7	CCS 7.2.11
EHR model	0.544	0.591	0.520	0.578	0.518	0.449	0.577
EHR-knowledge model	0.488	0.585	0.596	0.504	0.523	0.512	0.569
P-value	0.4275	0.9243	0.7112	0.1499	0.9551	0.538	0.8931

Table 7: Numbers of Noisy Features (Numbers of All Features) in Models When EHR Sample Size is Small.

model	CCS 7.2.1	CCS 7.2.2	CCS 7.2.3	CCS 7.2.4	CCS 7.2.6	CCS 7.2.7	CCS 7.2.11
EHR model	54 (126)	60 (140)	6 (52)	90 (235)	22 (71)	19 (52)	53 (164)
EHR-knowledge model	34 (77)	27 (71)	2 (23)	70 (165)	12 (39)	8 (23)	22 (77)

1c. When EHR sample size is large, models trained using EHR data only perform similarly to models trained using large sample size EHR data and medical knowledge.

Experiment Setting

I tried scenarios with different sample sizes: 43,910 (all training data), 35,128 (randomly selected 80 percent of training data), 26,346 (randomly selected 60 percent of training data), 17,564 (randomly selected 40 percent of training data), and 8,782 (randomly selected 20 percent of training data). Then, following the same steps in the experiment setting in hypothesis 1a, I developed EHR models. With different equivalent sample sizes, I generated many EHR-knowledge models and compared their performance with EHR models.

Experimental Results

Experimental results (Table 8) show that for CCS 7.2.1 and CCS 7.2.4, EHR-knowledge models performed worse than EHR models (bolded values). In other categories, EHR-knowledge models performed similarly to EHR models.

Table 8: Performance Comparison of EHR Models and EHR-knowledge Models (EHR Sample Size is Large).

N	M	CCS 7.2.1	CCS 7.2.2	CCS 7.2.3	CCS 7.2.4	CCS 7.2.6	CCS 7.2.7	CCS 7.2.11
43,910	0	**0.872**	**0.823**	**0.876**	**0.916**	**0.891**	**0.662**	**0.906**
43,910	8,782	0.865	0.824	0.876	0.909	0.89	0.655	0.905
43,910	17,564	0.865	0.824	0.877	0.909	0.89	0.657	0.905
43,910	26,346	0.865	0.825	0.88	0.909	0.889	0.659	0.904
43,910	35,128	0.862	0.826	0.877	0.909	0.89	0.658	0.905
43,910	43,910	0.862	0.827	0.878	0.909	0.892	0.656	0.906
35,128	0	**0.864**	**0.821**	**0.874**	**0.916**	**0.888**	**0.662**	**0.905**
35,128	8,782	0.867	0.821	0.878	0.908	0.89	0.659	0.904
35,128	17,564	0.864	0.821	0.882	0.909	0.891	0.658	0.904
35,128	26,346	0.864	0.821	0.882	0.909	0.891	0.657	0.904
35,128	35,128	0.864	0.821	0.881	0.91	0.887	0.651	0.904
35,128	43,910	0.864	0.821	0.88	0.909	0.888	0.653	0.903
26,346	0	**0.873**	**0.824**	**0.885**	**0.916**	**0.889**	**0.654**	**0.907**
26,346	8,782	0.872	0.826	0.883	0.908	0.887	0.651	0.907
26,346	17,564	0.866	0.824	0.883	0.908	0.887	0.65	0.906
26,346	26,346	0.868	0.825	0.882	0.908	0.888	0.649	0.906
26,346	35,128	0.868	0.823	0.881	0.909	0.889	0.643	0.906
26,346	43,910	0.868	0.824	0.881	0.909	0.889	0.647	0.906
17,564	0	**0.861**	**0.821**	**0.879**	**0.909**	**0.881**	**0.66**	**0.905**
17,564	8,782	0.852	0.818	0.88	0.906	0.883	0.654	0.905
17,564	17,564	0.852	0.82	0.879	0.906	0.883	0.654	0.906
17,564	26,346	v0.851	0.819	0.883	0.906	0.883	0.654	0.906
17,564	35,128	0.85	0.822	0.881	0.906	0.883	0.651	0.906

17,564	43,910	0.853	0.819	0.877	0.906	0.883	0.652	0.905
8,782	0	**0.853**	**0.806**	**0.872**	**0.913**	**0.868**	**0.628**	**0.902**
8,782	8,782	0.844	0.799	0.877	0.905	0.864	0.623	0.904
8,782	17,564	0.842	0.8	0.871	0.905	0.858	0.622	0.903
8,782	26,346	0.843	0.799	0.873	0.905	0.86	0.62	0.902
8,782	35,128	0.846	0.799	0.876	0.905	0.859	0.618	0.901
8,782	43,910	0.843	0.8	0.875	0.905	0.859	0.614	0.901

1d. When EHR sample size is large, models trained using EHR data only are less clinically aligning than models trained using large sample size EHR data and medical knowledge.

Experiment Setting

Similar to hypothesis 1b, for each CCS category, C_i, I compared M_{ie} with M_{ieqm} ($N = 43910, M = 43910$). I also used two ways to handle the situation that a physician annotated feature does not appear in one model. Using the first approach, when one feature did not appear in the model, I deleted it; the AUROCs are listed in table 9. Using the second approach, if one feature did not appear in the model, I assigned its $P(finding = true|disease = true) = 0$; the AUROCs are listed in table 10.

Experimental Results

The results in table 9 show that the AUROCs of EHR models were higher than those of EHR-knowledge models, but their differences were not statistically significant. The results in table 10 show that most AUROCs of EHR models (row 2) and EHR-knowledge models (row 3) were similar, and their differences were not statistically significant.

Comparing the number of noisy features between EHR models and EHR-knowledge models, table 11 shows that and the noisy feature number between two models were not statistically significant different (two-sample sign test, p-value = 0.25). The feature number between two models were also similar. That means that most features in the developed EHR models

and EHR-knowledge models were clinically aligning. Overall, when using all four years of training data (43,910 samples), I obtained diagnostic models that performed well and were clinically aligning. In this situation, my approach to add medical knowledge, although not harmful, was not necessary. This research hypothesis was not supported in my experimental results.

Table 9: Clinical Meaningfulness of Nodes in EHR Models and EHR-knowledge Models (EHR Sample Size is Large, Delete Missing)

model	CCS 7.2.1	CCS 7.2.2	CCS 7.2.3	CCS 7.2.4	CCS 7.2.6	CCS 7.2.7	CCS 7.2.11
EHR model	0.930	0.717	0.453	0.518	0.578	0.578	0.627
EHR-knowledge model	0.744	0.693	0.460	0.503	0.578	0.446	0.685
P-value	0.01366	0.8036	0.951	0.9232	1	0.9168	0.6439

Table 10: Clinical Meaningfulness of Nodes in EHR Models and EHR-knowledge Models (EHR Sample Size is Large, Missing as Zero)

model	CCS 7.2.1	CCS 7.2.2	CCS 7.2.3	CCS 7.2.4	CCS 7.2.6	CCS 7.2.7	CCS 7.2.11
EHR model	0.510	0.668	0.455	0.681	0.513	0.618	0.606
EHR-knowledge model	0.775	0.574	0.467	0.462	0.513	0.632	0.635
P-value	0.1856	0.1707	0.8406	0.1312	1	0.9444	0.7036

Table 11: Numbers of Noisy Features (Numbers of All Features) in Models when EHR Sample Size is Large.

model	CCS 7.2.1	CCS 7.2.2	CCS 7.2.3	CCS 7.2.4	CCS 7.2.6	CCS 7.2.7	CCS 7.2.11
EHR model	1 (44)	14 (79)	16 (85)	6 (70)	3 (50)	3 (18)	12 (91)
EHR-knowledge model	1 (44)	15 (74)	16 (86)	8 (57)	3 (50)	4 (18)	12 (89)

4.3 Experiment Settings and Results for Research Aim 2

Research Aim 2. Build a Sequential Diagnostic Model that Considers Diagnostic Accuracy, Clinical Workflow, and Misdiagnosis Costs

To generate a sequential diagnosis tree model, I used two types of knowledge: (1) expert-defined four-phase workflow to restrict the search of the nodes, (2) expert-defined misclassification costs to calculate searching scores. In this way, the modeling process was able to consider diagnostic accuracy, clinical workflow, and misdiagnosis costs simultaneously. I compared the performance of the classification tree generated by the SDG-phase algorithm and the SDG-no-phase algorithm with the classification trees generated using the multi-label classification tree algorithm ML-C4.5 [17] , as well as policies learned by deep Q learning [110] and tested multiple research hypotheses.

2a: The sequential diagnostic models generated using the SDG algorithm have a higher diagnostic accuracy than the sequential diagnostic model generated using the ML-C4.5 algorithm.

Experiment Setting

Since all three models (SDG-phase model, SDG-no-phase model, and ML-C4.5 model) provided probabilities for each disease category, I used testing data (2014 data) to evaluate their diagnostic performance in AUROCs for each CCS category.

For models with depth 10, I also calculated the testing accuracy for each of the three models. Because these models did not provide diagnostic labels, thresholds were decided using the training data. For each disease category in each model, the threshold value that reached the highest accuracy in the training data was selected as the threshold for this category. In each model (SDG-phase model, SDG-no-phase model, and ML-C4.5 model), each disease category (out of eight categories) had a threshold value for diagnosis. Using these thresholds and the probabilities from the SDG-phase model, SDG-no-phase model, and ML-C4.5 model, I generated diagnostic labels for eight categories. With the generated diagnostic labels, I calculated accuracies for the three models using true diagnosis information

in the testing dataset. If an encounter included multiple diseases, a diagnosis was considered to be correct only if all of the diseases were diagnosed correctly.

Experimental Results

Figure 4 shows the comparison of models' diagnostic accuracies using testing encounters with true diagnosis of category 7.2.1. The horizontal axis is the depth of the sequential tree models, and the vertical axis is the AUROC of models. The SDG-phase model and SDG-no-phase model's performance increased as depth increased. The SDG-no-phase models had higher accuracy than ML-C4.5 models in most of the time. The SDG-phase models had the lowest AUROC. The ML-C4.5 model's performance increased when depth increased. But when the depth number became too large (depth 14 in Figure 4), AUROC decreased, indicating potential over-fitting. Similar patterns can be found in Figure 5 (comparison result of Category 7.2.2), Figure 6 (comparison result of Category 7.2.3), Figure 7 (comparison result of Category 7.2.4), Figure 8 (comparison result of Category 7.2.6), Figure 9 (comparison result of Category 7.2.7), Figure 10 (comparison result of Category 7.2.89), Figure 11 (comparison result of Category 7.2.11).

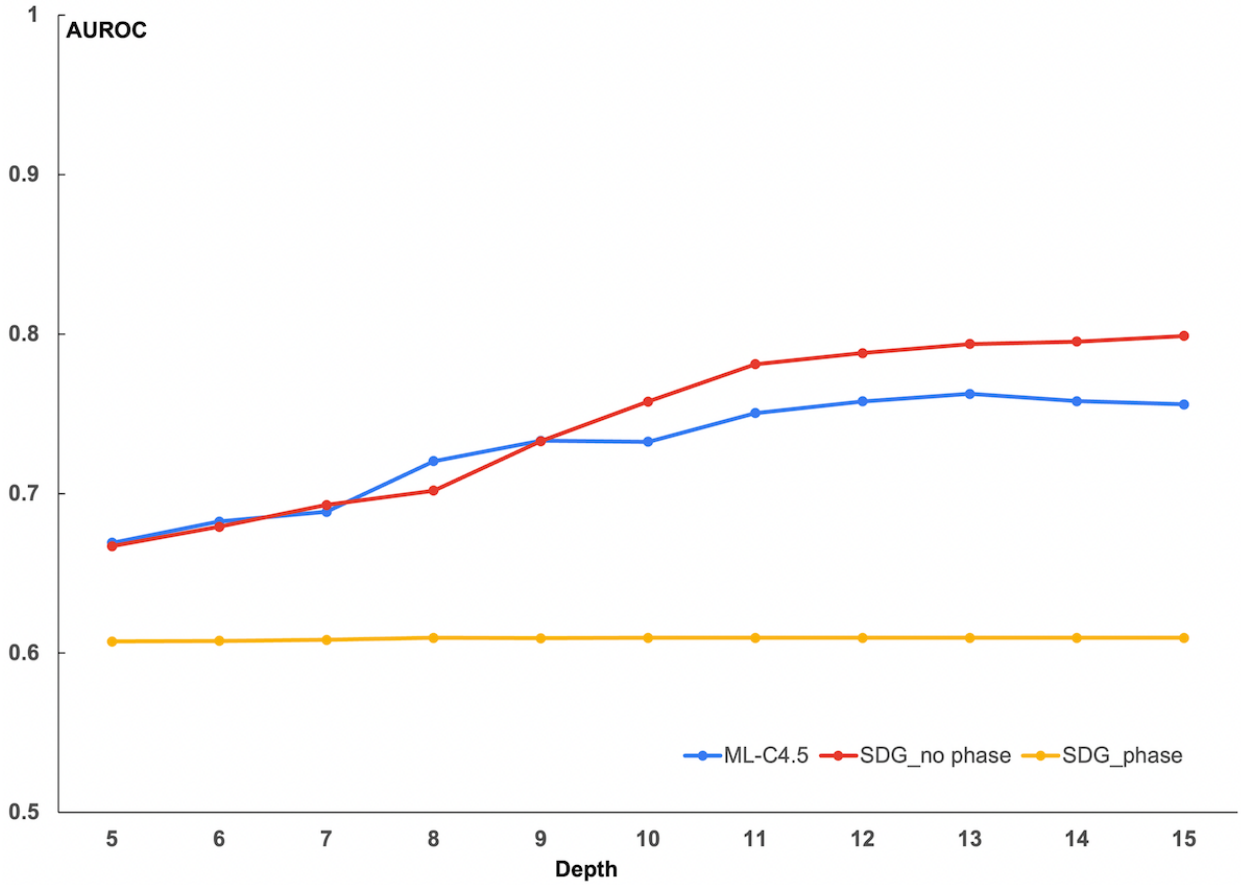

Figure 4: Classification Tree Diagnostic Accuracy Comparison–Category CCS 7.2.1 (Heart valve disorders)

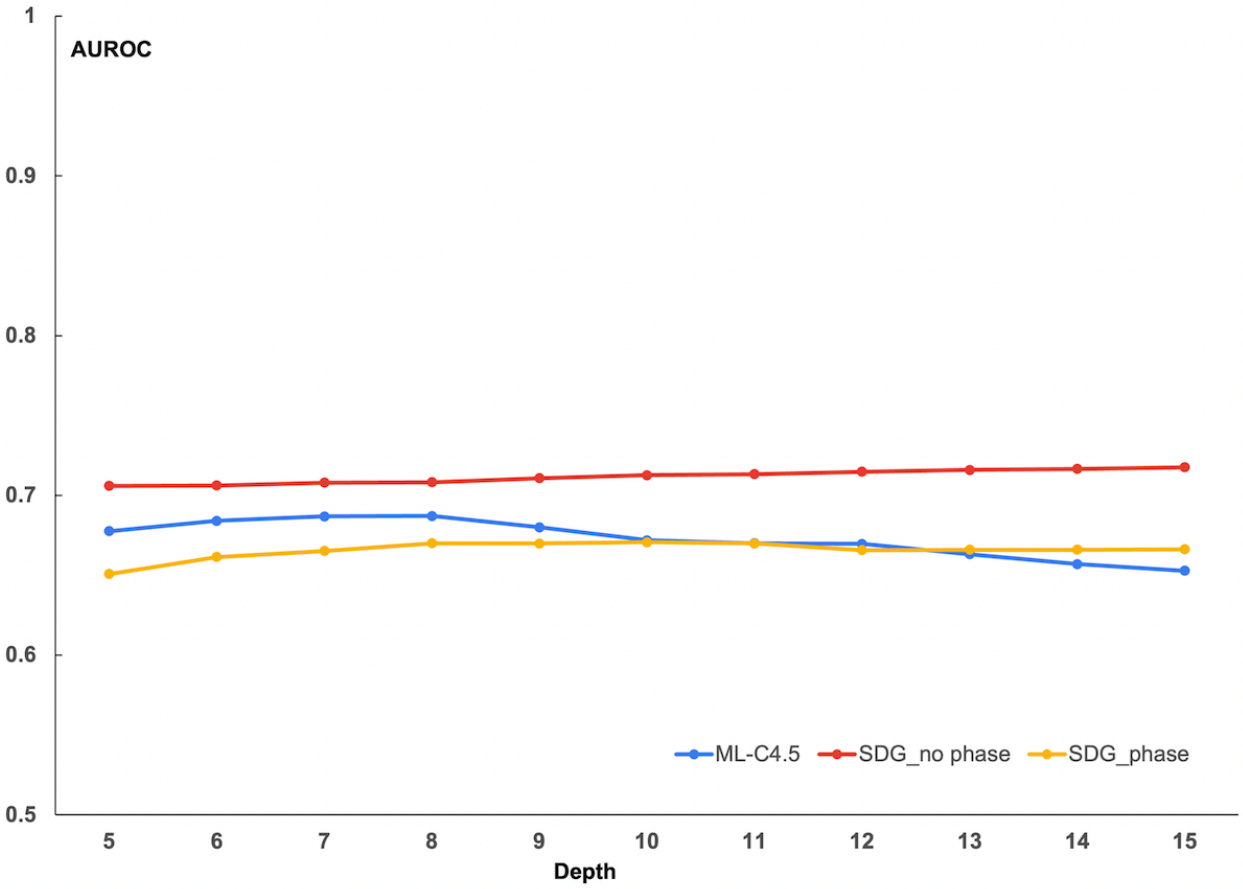

Figure 5: Classification Tree Diagnostic Accuracy Comparison–Category CCS 7.2.2 (Peri-; endo-; and myocarditis; cardiomyopathy)

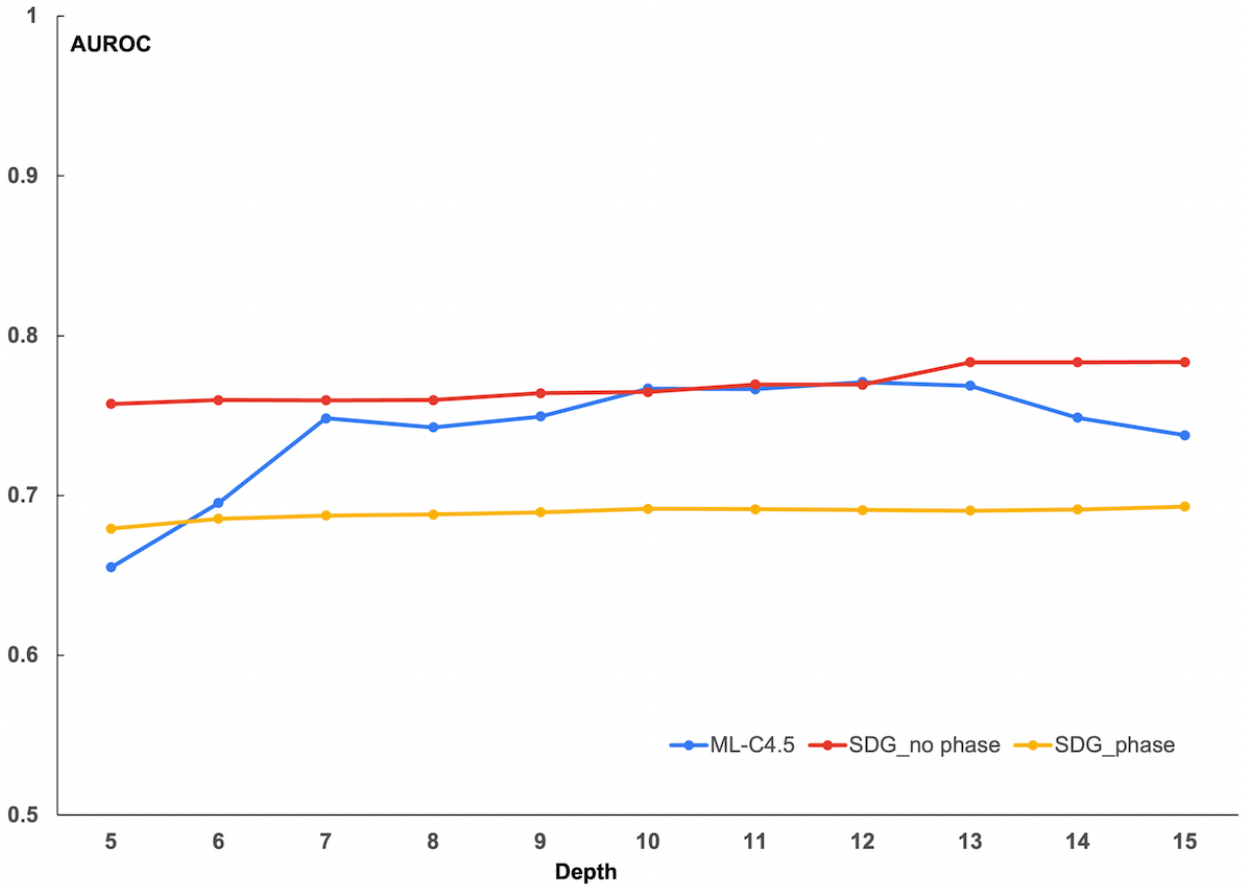

Figure 6: Classification Tree Diagnostic Accuracy Comparison–Category CCS 7.2.3 (Acute myocardial infarction)

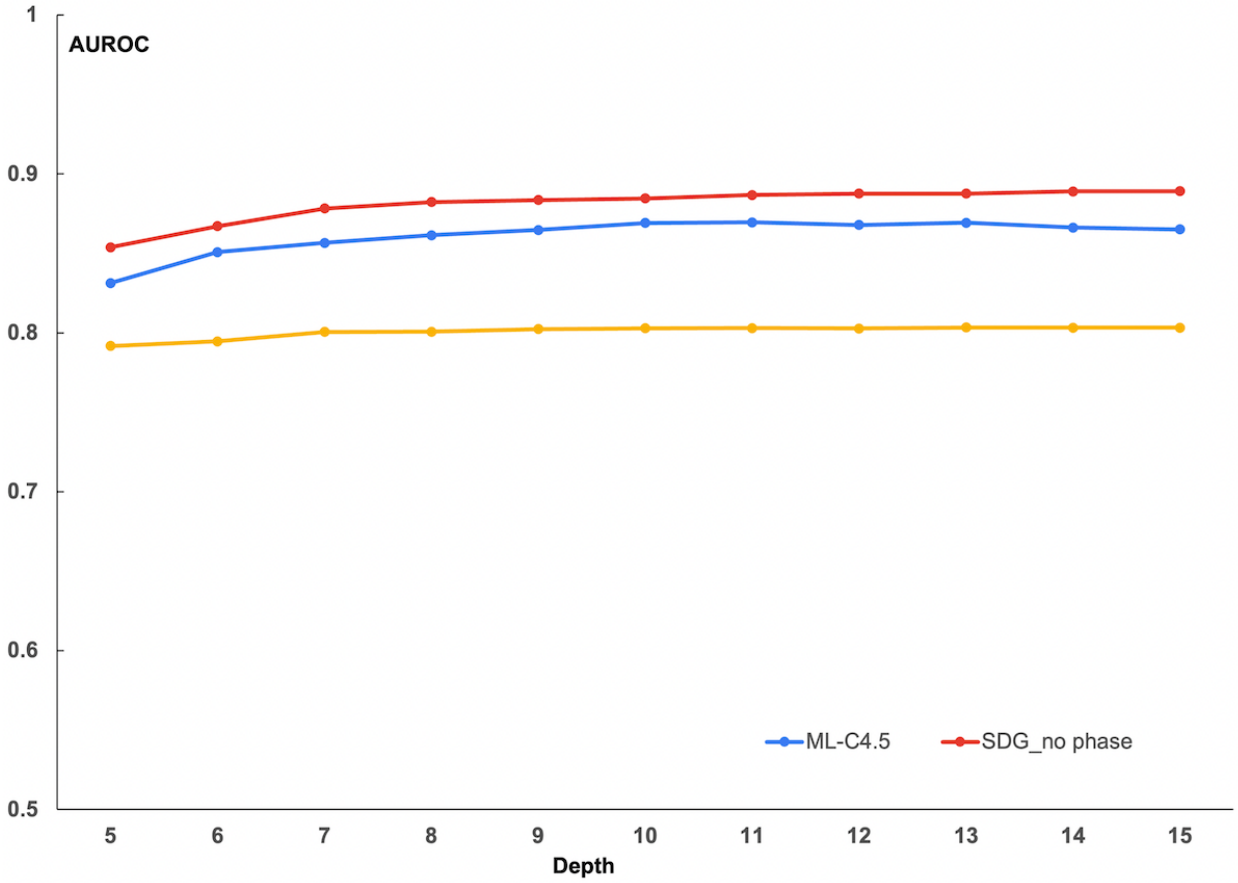

Figure 7: Classification Tree Diagnostic Accuracy Comparison–Category CCS 7.2.4 (Coronary atherosclerosis and other heart disease)

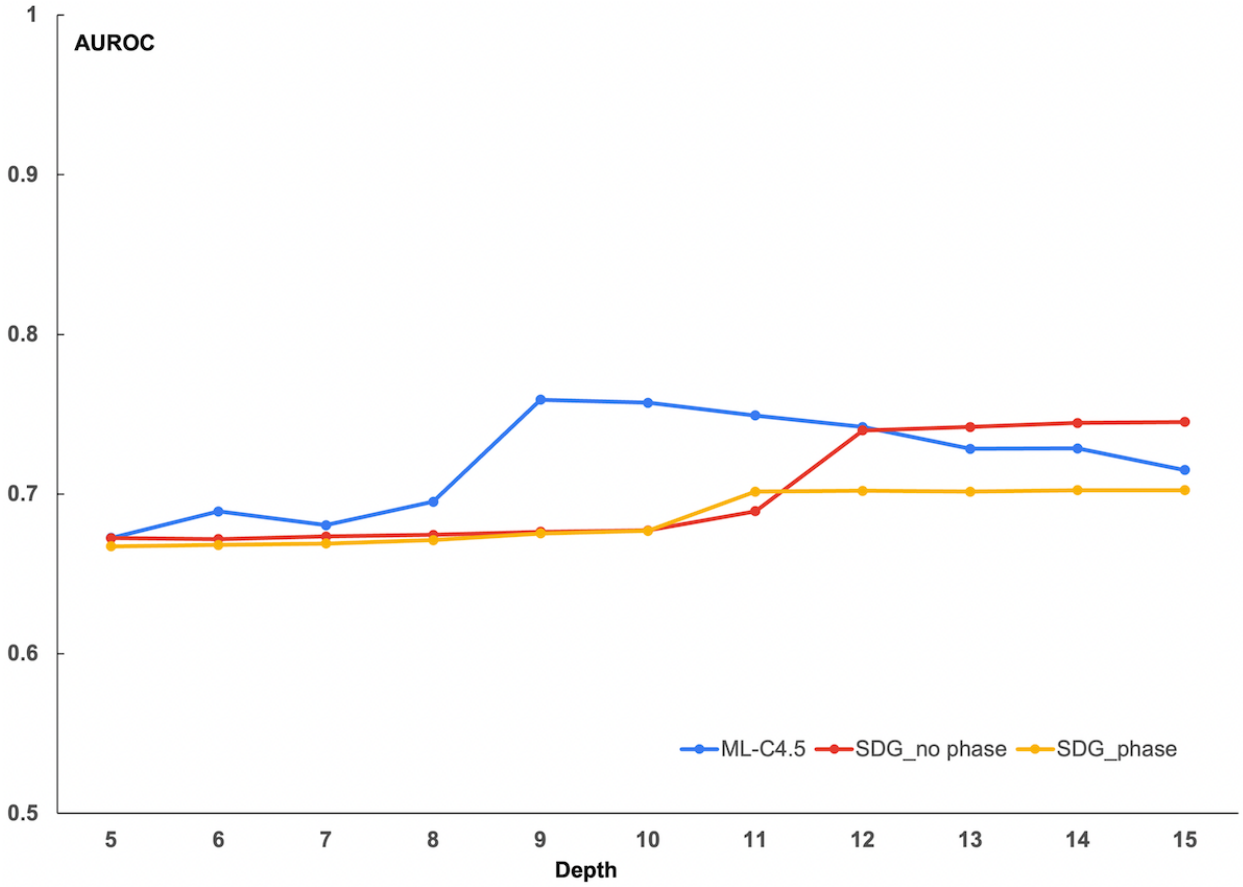

Figure 8: Classification Tree Diagnostic Accuracy Comparison–Category CCS 7.2.6 (Pulmonary heart disease)

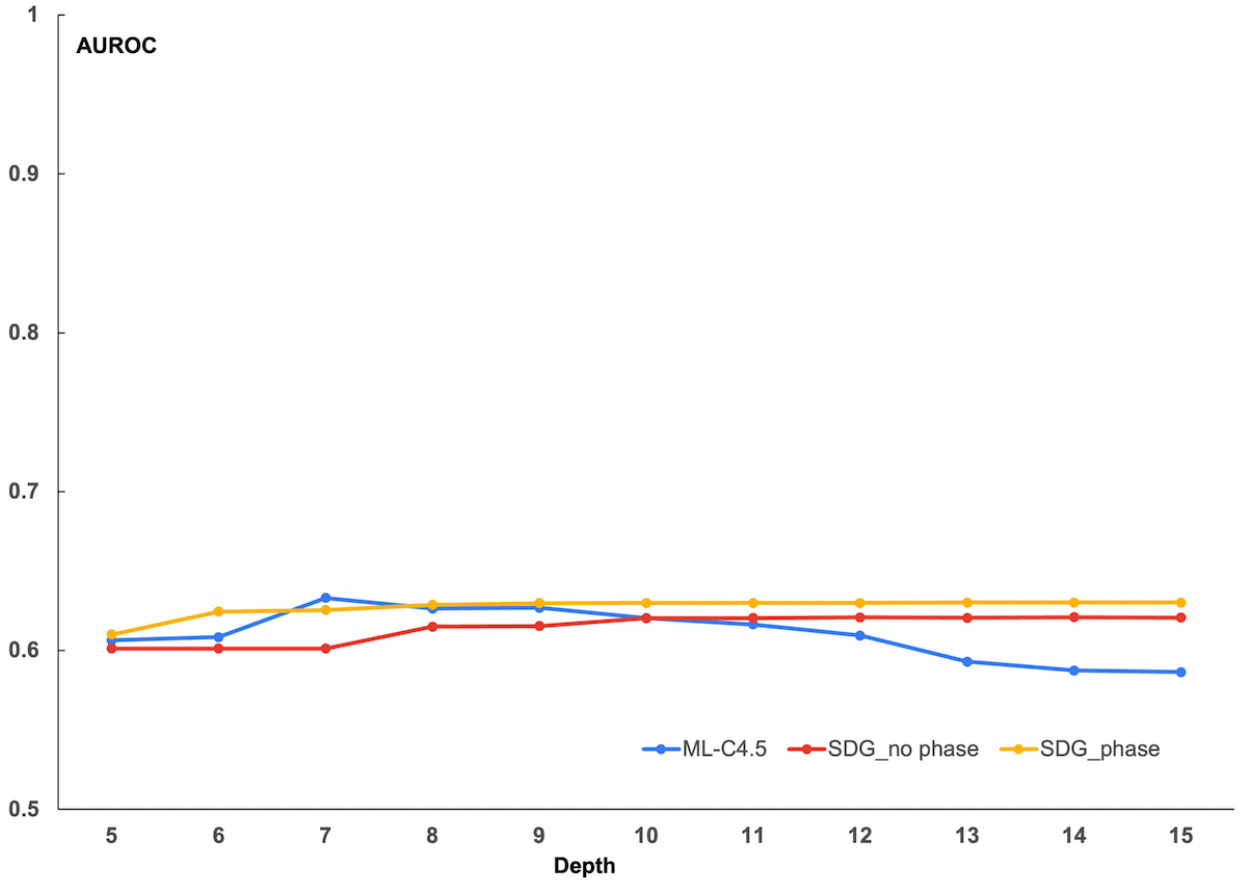

Figure 9: Classification Tree Diagnostic Accuracy Comparison–Category CCS 7.2.7 (Other and ill-defined heart disease)

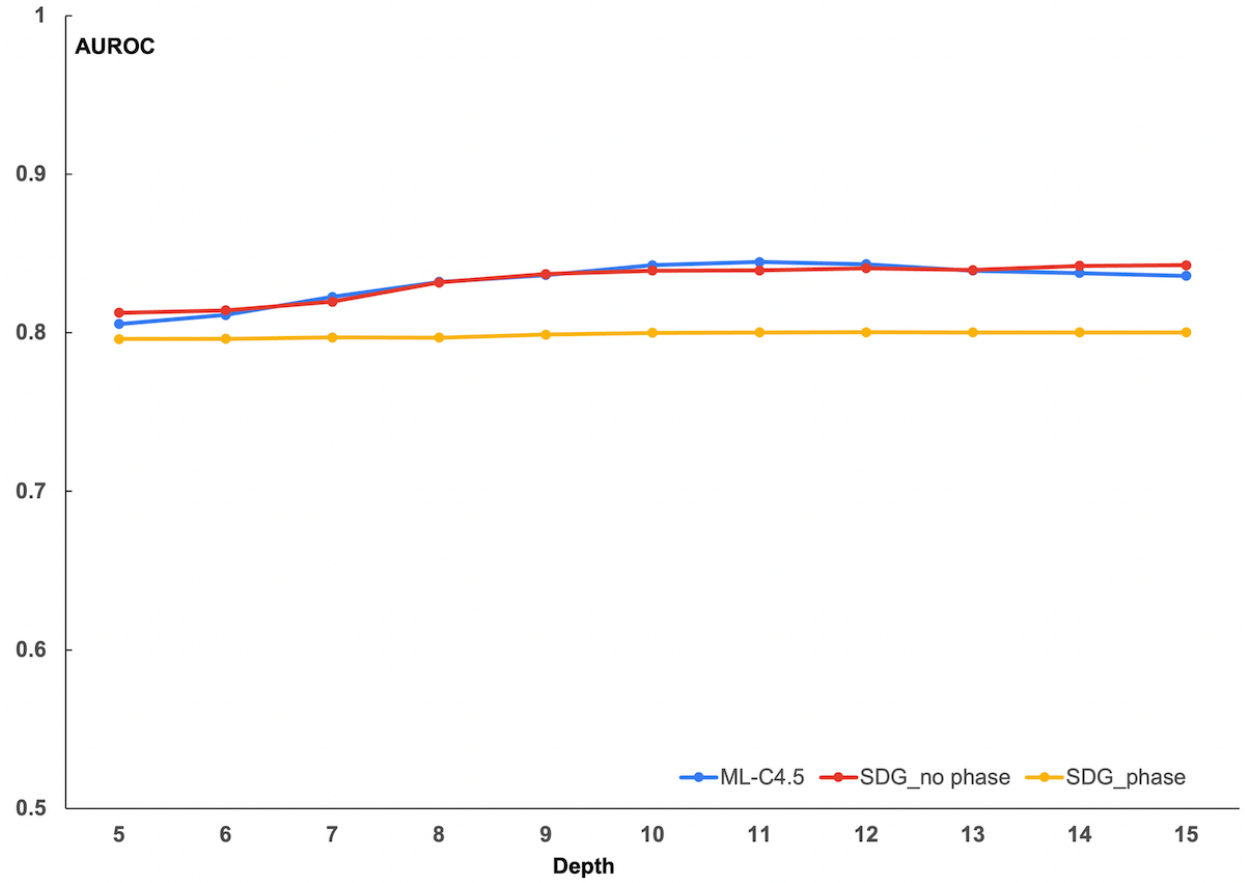

Figure 10: Classification Tree Diagnostic Accuracy Comparison–Category CCS 7.2.89 (Conduction disorders and cardiac dysrhythmias)

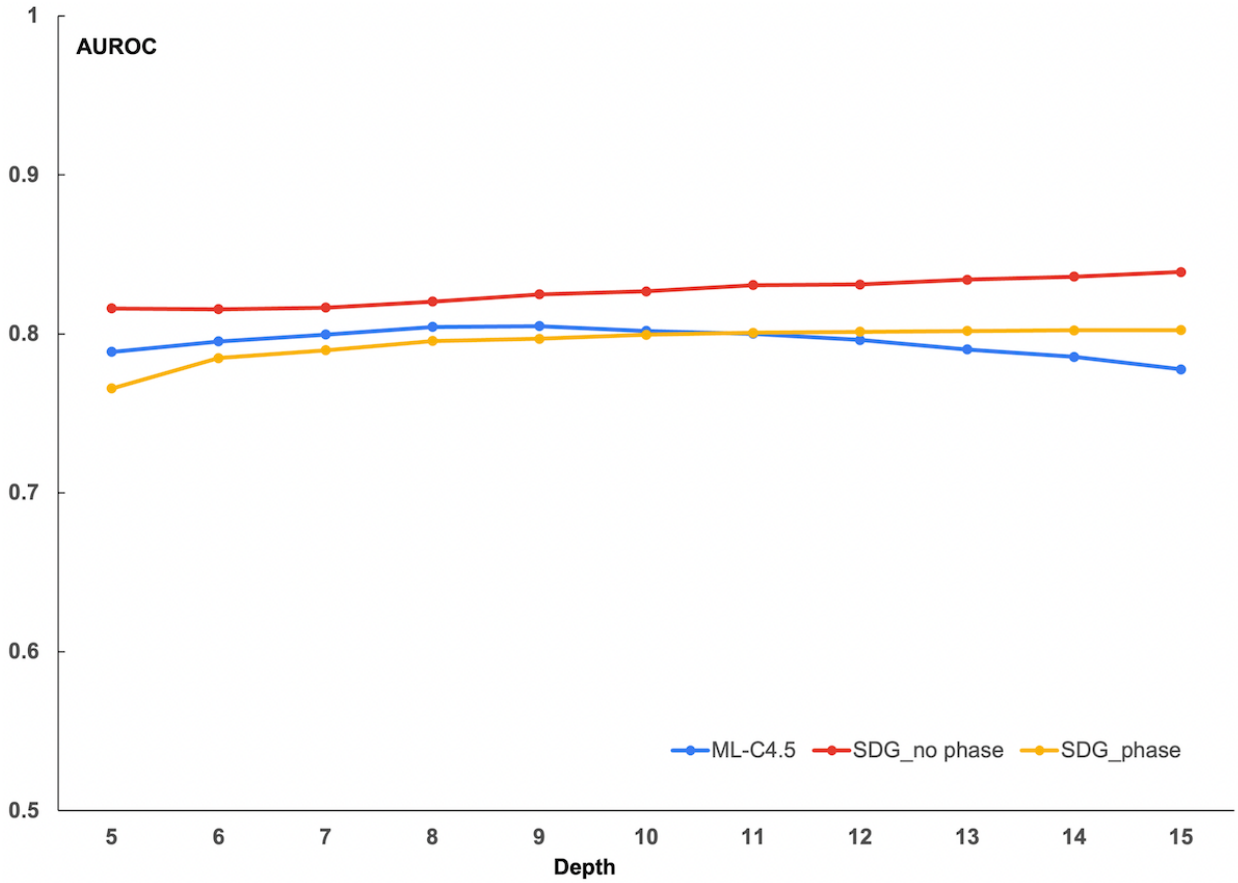

Figure 11: Classification Tree Diagnostic Accuracy Comparison–Category CCS 7.2.11 (Congestive heart failure; nonhypertensive)

Table 12 provides the AUROCs of three sequential tree models for diagnosing different disease categories. In each column, if the highest value(s) are statistically significantly higher than other values, then the number(s) are bolded. These results show that the SDG-no-phase model's performance was significantly better than ML-C4.5 model' in most of the time. The SDG-phase model performed the worst. The SDG-phase model restricted the searching to earlier phases before exploring other phases. With this restriction, the SDG-phase model did not choose the most discriminative features and thus had a worse diagnostic performance.

Table 12: Performance of Sequential Tree Models

Depth	Model	CCS 7.2.1	CCS 7.2.2	CCS 7.2.3	CCS 7.2.4	CCS 7.2.6	CCS 7.2.7	CCS 7.2.8 and CCS 7.2.9	CCS 7.2.11
5	ML-C4.5	**0.6692**	0.6776	0.6551	0.8314	0.6726	0.6065	0.8056	0.7887
5	SDG-no-phase	**0.667**	**0.706**	**0.7574**	**0.8539**	0.6725	0.6013	**0.8126**	**0.8161**
5	SDG-phase	0.6073	0.6508	0.6793	0.7917	0.6673	0.6101	0.7961	0.7657
6	ML-C4.5	**0.6825**	0.6841	0.6954	0.8508	**0.6892**	0.6085	**0.8112**	0.7953
6	SDG-no-phase	**0.6792**	**0.7062**	**0.7598**	**0.8671**	0.6718	0.6013	**0.8142**	**0.8156**
6	SDG-phase	0.6076	0.6615	0.6855	0.7946	0.6682	0.6245	0.7962	0.7848
7	ML-C4.5	**0.6886**	0.6869	**0.7484**	0.8566	0.6805	**0.6331**	**0.8227**	0.7996
7	SDG-no-phase	**0.6928**	**0.708**	**0.7596**	**0.8783**	0.6735	0.6013	**0.8196**	**0.8165**
7	SDG-phase	0.6082	0.6653	0.6875	0.8006	0.669	**0.6255**	0.7971	0.7897
8	ML-C4.5	**0.7203**	0.6871	**0.7427**	0.8615	**0.6953**	0.6264	**0.8321**	0.8044
8	SDG-no-phase	0.7018	**0.7083**	**0.7598**	**0.8823**	0.6745	0.615	**0.8318**	**0.8204**
8	SDG-phase	0.6095	0.6701	0.6882	0.8008	0.6712	0.6288	0.797	0.7956
9	ML-C4.5	**0.7332**	0.6801	**0.7495**	0.8647	**0.7591**	0.6269	**0.8364**	0.8049
9	SDG-no-phase	**0.7329**	**0.7108**	**0.764**	**0.8836**	0.6764	0.6154	**0.8371**	**0.8249**
9	SDG-phase	0.6093	0.6699	0.6896	0.8024	0.6754	0.6298	0.7988	0.7969
10	ML-C4.5	0.7325	0.6719	**0.767**	0.8692	**0.7572**	0.6205	**0.8427**	0.8019
10	SDG-no-phase	**0.7577**	**0.7127**	**0.7648**	**0.8846**	0.6773	0.6203	**0.8392**	**0.8268**
10	SDG-phase	0.6095	0.6707	0.6917	0.8029	0.677	0.6299	0.8	0.7995
11	ML-C4.5	0.7505	0.6701	**0.7666**	0.8695	**0.7493**	0.6164	**0.8446**	0.8001
11	SDG-no-phase	**0.7811**	**0.7133**	**0.7695**	**0.8868**	0.6893	0.6205	0.8394	**0.8307**
11	SDG-phase	0.6095	0.6699	0.6915	0.803	0.7015	0.6299	0.8003	0.8007
12	ML-C4.5	0.7578	0.6697	**0.771**	0.8679	**0.742**	0.6095	**0.8433**	0.7962
12	SDG-no-phase	**0.7881**	**0.7149**	**0.7694**	**0.8876**	**0.7399**	0.6208	**0.8407**	**0.8311**
12	SDG-phase	0.6095	0.6658	0.6909	0.8028	0.702	0.6299	0.8004	0.8014
13	ML-C4.5	0.7625	0.6632	**0.7687**	0.8693	**0.7284**	0.593	**0.8392**	0.7902
13	SDG-no-phase	**0.7938**	**0.716**	**0.7834**	**0.8876**	**0.7421**	0.6206	**0.8396**	**0.8341**
13	SDG-phase	0.6095	0.666	0.6905	0.8034	0.7016	0.6302	0.8003	0.8019
14	ML-C4.5	0.758	0.6571	0.7488	0.8662	**0.7286**	0.5875	**0.8377**	0.7856
14	SDG-no-phase	**0.7953**	**0.7166**	**0.7834**	**0.889**	**0.7446**	0.621	**0.8422**	**0.836**
14	SDG-phase	0.6095	0.666	0.6913	0.8033	0.7025	0.6302	0.8003	0.8022
15	ML-C4.5	0.7559	0.6529	0.7377	0.8651	0.7151	0.5864	0.8359	0.7777
15	SDG-no-phase	**0.7989**	**0.7176**	**0.7835**	**0.8892**	**0.7452**	0.6207	**0.8427**	**0.8389**
15	SDG-phase	0.6095	0.6663	0.6931	0.8033	0.7025	0.6302	0.8003	0.8024

For the model with depth 10, the ML-C4.5 model had the highest accuracy (0.4772). The SDG-no-phase model performed slightly worse (0.4621). The SDG-phase model had the lowest accuracy (0.4035).

2b: The sequential diagnostic models generated using the SDG algorithm are more clinically aligning than the sequential diagnostic model generated using the ML-C4.5 algorithm.

Using three different modeling strategies (SDG algorithm, SDG algorithm without considering phase information, and ML-C4.5 algorithm), I obtained three sequential diagnostic models: an SDG-phase model, SDG-no-phase model, and ML-C4.5 model. To compare their **clinical alignment**, I used two measurements: (1) calculating the expected misdiagnosis costs of each model in a testing dataset. Since an important goal in clinical practice is to reduce severe diagnosis errors and misdiagnosis costs may reflect the consideration of clinical severity and clinical alignment. A low misdiagnosis cost indicates few severe mistakes. (2) retrieving diagnosis paths for many representative real-world records and asking a physician to review them.

Misdiagnosis Costs Calculation

For SDG-phase and SDG-no-phase methods, in the training data, for each category, I used the expected misdiagnosis costs as a criterion to select the best probability threshold, then used this threshold to do prediction in test data. I used the following steps:

[1] For a CCS category, C_i, I used the training data, D_i, to decide the best threshold for disease diagnosis: I sorted disease probability values from highest to lowest, searched the threshold from the highest value to the lowest, and found the threshold that could lead to the lowest total expected misdiagnosis costs in the training data. Then this threshold, p_i , was used as the threshold for disease diagnosis in the CCS category. I used the same approach to find the best thresholds for all CCS categories.

[2] I used p_i and the probabilities of the SDG model for each encounter in the test data to generate a diagnostic label for CCS category, C_i, in the test data. In the same way, I generated a diagnostic label for each of the eight CCS categories.

[3] Using the generated diagnostic labels, I used the expert defined misdiagnosis costs matrix to calculate the expected misdiagnosis costs for all encounters in the test data.

[4] I added up these expected misdiagnosis costs to obtain the total expected misdiagnosis costs for each algorithm.

When calculating the total expected misdiagnosis costs for the ML-C4.5 model in the test data, I used accuracy instead of total expected misdiagnosis costs in the training data as a score to find the best threshold for probabilities in each CCS category. Using these thresholds, I generated diagnostic labels for the ML-C4.5 model's probabilities in the test data. Then, with these diagnostic labels and the expert defined misdiagnosis costs matrix, I calculated the total expected misdiagnosis costs for the ML-C4.5 model in the test data. I used the two-sample sign test to compare misdiagnosis costs.

Sequential Diagnosis Path Review

I selected 20 most representative records for each category from test data, input $20 \times 8 = 160$ paths into each model, obtained the sequential diagnosis path of each record, then asked physicians to annotate whether the diagnosis path makes sense. The larger the number of paths that were agreed by a physician expert, the more clinical alignment the model had. To select the 20 most representative paths for physician review, I used the following steps for each disease category in each model:

[1] For each category, C_i, I assigned a subset of encounters, $Subset_i$, in the testing dataset that had any diagnosis in this category.

[2] I conducted an unsupervised k-means clustering on encounters in S_i. The clustering minimized within-cluster variances; the Euclidean distances between two points were calculated based on the values of the clinical findings of these two encounters.

[3] For each category, I used the k-means clustering approach to find 20 cluster centers. Then for each cluster center, I used the nearest point to represent the center. In this way, I obtained 20 encounters that represented 20 different clinical manifestations of this disease category. The selection of these 20 representative encounters was completely independent of sequential diagnostic models.

[4] I used an SDG-phase model, SDG-no-phase model, and ML-C4.5 model to generate diagnostic paths for these 20 encounters and requested a physician expert to review.

Experimental Results

Table 13 lists the misdiagnosis costs matrix that I elicited from a physician expert.

Misdiagnosis occurs when a doctor diagnoses a patient with a disease but the patient does not have the disease at that time. A missed diagnosis means that the patient has a disease but gets no diagnosis. The estimated costs range from 1 to 10, with 1 be the lowest cost and 10 be the highest cost. For instance, for a patient having a disease in CCS 7.2.1, if a physician missed this disease, then the cost is 2. If a patient does not have a disease in CCS 7.2.1 but a physician mistakenly diagnoses that the patient has the disease, then the cost is 6. The differences in cost values indicate the differences in the severity of the mistakes.

Table 13: Misdiagnosis Costs Defined by an Expert

CCS category	type of error	estimated cost
CCS 7.2.1	missed diagnosis	2
CCS 7.2.1	misdiagnosis	6
CCS 7.2.2	missed diagnosis	3
CCS 7.2.2	misdiagnosis	7
CCS 7.2.3	missed diagnosis	5
CCS 7.2.3	misdiagnosis	10
CCS 7.2.4	missed diagnosis	2
CCS 7.2.4	misdiagnosis	5
CCS 7.2.6	missed diagnosis	2
CCS 7.2.6	misdiagnosis	5
CCS 7.2.7	missed diagnosis	2
CCS 7.2.7	misdiagnosis	6
CCS 7.2.89	missed diagnosis	1
CCS 7.2.89	misdiagnosis	4
CCS 7.2.11	missed diagnosis	3
CCS 7.2.11	misdiagnosis	7

Using the steps mentioned in the experimental setting, I calculated the expected misdiagnosis costs of the SDG models and ML-C4.5 models in the test data. Figure 12 shows that the SDG models had lower misdiagnosis costs than the ML-C4.5 model after the depth was greater than 7. The misdiagnosis costs of ML-C4.5 increased as depth increased, which may result from potential model over-fitting. The SDG-no-phase had the lowest misdiagnosis costs most of the time.

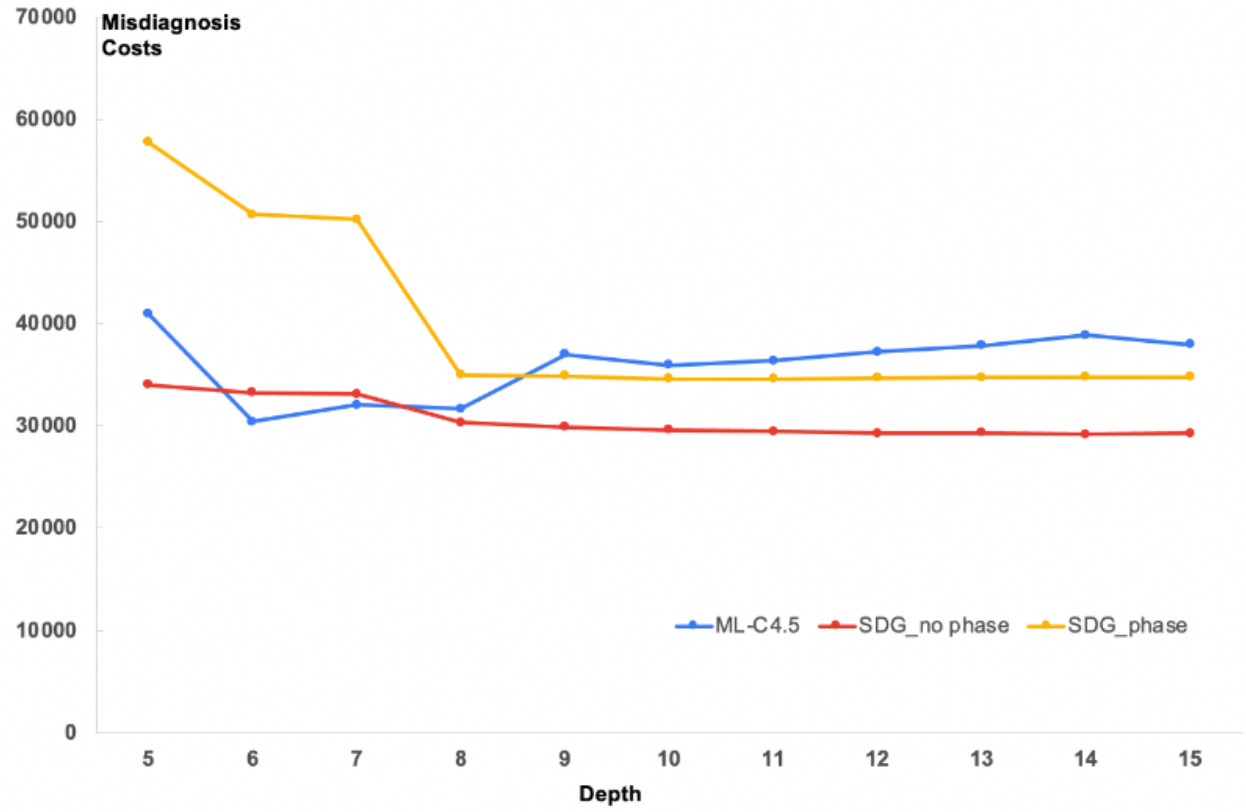

Figure 12: Classification Tree Misdiagnosis Costs Comparison

Now, let us look at the reviewing result of sequential diagnosis paths. In total, I selected $8 \times 20 = 160$ encounters (20 for one category) for physician expert review. Each encounter had three diagnostic paths, each of which was generated by one model (depth=10). A physician expert reviewed these paths and marked 1 or 0 to indicate whether a path was

clinically correct for diagnosing a disease category: 1 indicated correct; 0 indicated wrong. Using the expert's annotation as gold standard, the accuracies of the paths were: SDG-phase model (85 out of 160), SDG-no-phase model (97 out of 160), ML-C4.5 model (81 out of 160). The SDG models were more clinically aligning with the ML-C4.5 model. Table 14 provides frequency counts of the clinically aligned paths in each category. The SDG-no-phase models have more clinically aligned paths than ML-C4.5 model. This difference was not statistically significant (two sample sign test p-value=0.125), due to the small sample size (eight CCS categories mean eight data points). The difference of number of clinically aligned paths between SDG-no-phase model and SDG-phase model was also not statistically significant (two sample sign test p-value=0.6875). Also difference of number of clinically aligned paths between SDG-phase model and ML-C4.5 model was not statistically significant (two sample sign test p-value=1).

Table 14: Frequency of Clinically Aligned Paths

Category	ML-C4.5 model	SDG-no-phase model	SDG-phase model
CCS 7.2.1	5	9	4
CCS 7.2.2	8	9	10
CCS 7.2.3	15	18	16
CCS 7.2.4	13	16	13
CCS 7.2.6	11	7	9
CCS 7.2.7	10	15	10
CCS 7.2.89	11	11	11
CCS 7.2.11	8	12	12
total	81	97	85

The misdiagnosis cost and clinical path comparison results partially proved the research hypothesis: The sequential diagnostic models generated using the SDG algorithm are more clinically aligning than the sequential diagnostic model generated using the ML-C4.5 algorithm. It is surprised that SDG-no-phase models were more clinically aligning than SDG-phase models. Comparisons between the SDG-no-phase algorithm and the SDG-phase algorithm are provided in Section 5.1.4. Discussions on future work of SDG-phase algorithm is provided in Section 5.4.4.

2c: The sequential diagnostic models generated using the SDG algorithm have a higher diagnostic accuracy than the policies developed by one implementation of deep Q learning algorithm.

Experiment Setting

For the deep Q learning, I set demographic features as explicit features, and chose a reward discount 0.95. The learned DQN model provided one final diagnosis for each encounter in the test data. I only used encounters (in the test data) that had one diagnosis (instead of multiple diagnoses) for evaluation and comparison. For each encounter, the SDG-no-phase model and SDG-phase model both provided a probability for each disease category. Each encounter was assigned to the category with the highest probability. Comparing the label generated by models against the true disease category, I obtained a measure of diagnostic accuracy.

Table 15 lists research data that were used in training and testing a deep Q learning model. The research data included 29,607 training samples and 8,532 test samples, with 1799 unique features. The data were saved in pickle files with a total size of 295.4G before using them as input for the deep Q learning implementation.

Table 15: EHR Data for Deep Q Learning

CCS	Description	Cases in Training data	Cases in Test data
7.2.1	Heart valve disorders	1713	513
7.2.2	Peri-; endo-; and myocarditis; cardiomyopathy	628	166
7.2.3	Acute myocardial infarction	570	170
7.2.4	Coronary atherosclerosis and other heart disease	9865	2603
7.2.6	Pulmonary heart disease	795	333
7.2.7	Other and ill-defined heart disease	298	57
7.2.89	Conduction disorders and cardiac dysrhythmias	14742	4421
7.2.11	congestive heart failure; nonhypertensive	996	269

Experimental Results

The training process of a deep Q model finished after about 125 hours on DELL Precision

5820 Tower with Nvidia RTX A5000, 24GB, 4DP. Table 16 shows that the SDG-no-phase model performed the best. The SDG-phase model performed better than the DQN model. In fact, the DQN model assigned the category (CCS 7.2.8 or CCS 7.2.9) to all encounters in testing data. This category had the largest percentage in both training and testing data.

Table 16: Accuracy Comparisons between SDG and Deep Q Learning

Model	Depth=5	Depth=10	Depth=15
SDG-no-phase	0.689	0.737	0.755
SDG-phase	0.682	0.688	0.687
DQL	0.518	0.518	0.518

2d: The sequential diagnostic models generated using the SDG algorithm are more clinically aligning than the policies developed by one implementation of deep Q learning algorithm.

Experiment Setting

Because the deep Q learning model provides only one diagnosis for each encounter, I selected encounters that only had one diagnosis in the testing dataset for evaluation. The SDG-phase model and SDG-no-phase model all provided probabilities for each category. Thus, for each model, the estimated diagnostic label for one encounter would be the category with the highest probability in that encounter. With these estimated diagnostic labels, true diagnostic labels, and the misdiagnosis matrix in Table 13, I calculated expected misdiagnosis costs for encounters with one diagnosis. Since the DQN model directly assigns CCS7.2.89 to all testing encounters without asking any question, there is no diagnosis path for review.

Experimental Results

The SDG models had much lower misdiagnosis costs than the deep Q models in the testing data. The misdiagnosis costs in the testing dataset were: DQN model, 25611; SDG-

no-phase model, 15277; SDG-phase model, 18236.

2e: The sequential diagnostic models generated using the SDG algorithm uses less resources than models generated by ML-C4.5 algorithm and one implementation of deep Q learning algorithm.

Experiment Setting

I compared the resource usage suggested by the developed models. An important consideration for emergency department physicians is to get the final diagnosis as fast as possible - the fewer resources as possible. On the premise of keeping a high diagnostic quality, the fewer the number of the lab tests, the faster physicians can get a final diagnosis.

Experimental Results

Figure 13 shows that ML-C4.5 models used many more features than the SDG-phase model and SDG-no-phase models. The SDG-no-phase models used fewest features. Next, for models with tree depth 10, I compared feature information of the SDG-phase model, SDG-no-phase model, and ML-C4.5 model. The **SDG-phase model** included 32 features (Appendix C), including 12 signs or symptoms, 11 findings, 4 diseases or syndromes, 2 mental or behavioral dysfunctions, 2 pathologic functions, and 1 organism function. When generating the SDG-phase model, the SDG (phase) algorithm restricted its search to findings that had been included in any of eight EHR models. It explored the candidate features in phase 1 first, followed by phase 2, and then phase 3 and phase 4. It only explored features in the next phase after no feature in the previous phase was found to be useful. Because of this restriction, the developed model may align with clinical workflow well. The **SDG-no-phase model** included 76 features (Appendix D), including 16 findings, 6 diseases or syndromes, 15 organic chemical finding, 9 medical devices, 7 therapeutic or preventive procedures, 2 diagnostic procedures, and 1 laboratory procedure. When generating the SDG-no-phase model, the SDG algorithm restricted its search on findings that had been included in any of the eight EHR models while ignoring phase information.

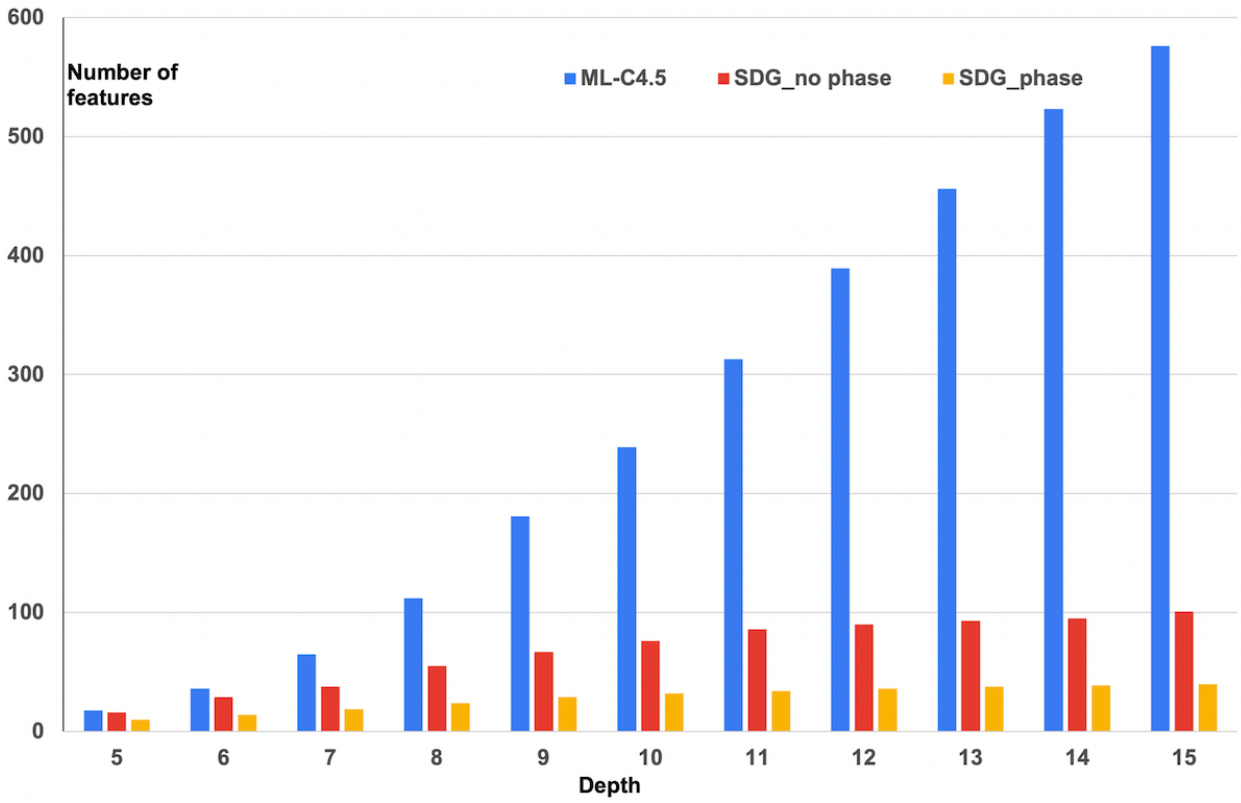

Figure 13: Resource Usage Comparison

The **ML-C4.5 model** included 239 features (Appendix E), including 39 findings, 17 diseases or syndromes, 32 signs or symptoms, 13 laboratory procedures, 12 diagnostic procedures, 14 therapeutic or preventive procedures, and 33 pharmacologic substances. When generating the ML-C4.5 model, the algorithm searched for the best feature among candidate features. The candidate feature set was the union of feature sets, where each feature set contained features whose information gain scores for a disease category (as measured using training data) were greater than 0.0001. The **deep Q network** was designed to provide scores for all candidate actions. Therefore, it involved all features in the candidate feature set. I used the same information gain set (as ML-C4.5) for deep Q learning.

I measured diagnostic quality based on the number of lab tests suggested by the models (depth = 10) in the research data. The SDG-phase model includes 32 features (Appendix C) and does not include any lab test. The SDG-no-phase model has 76 features (Appendix

D), including 2 diagnostic procedures (cardiac catheterization procedures performed, Holter Electrocardiography), 1 laboratory procedure (digoxin blood measurement), and 7 therapeutic or preventive procedures (coronary artery bypass surgery, dialysis procedure, intubation, percutaneous transluminal coronary angioplasty, replacement of aortic valve, and replacement of mitral valve).

The ML-C4.5 model has 239 features (Appendix E), including 13 laboratory procedures, 12 diagnostic procedures, and 14 therapeutic or preventive procedures. **Laboratory procedures** in ML-C4.5 model include blood thyroid stimulating hormone analysis, complete blood count, creatine kinase measurement, folic acid measurement, international normalized ratio, laboratory studies, liver function tests, oxygen saturation measurement, potassium measurement, sodium measurement, total protein measurement, vitamin d measurement, and white blood cell count procedure. **Diagnostic procedures** include auscultation, diagnostic imaging, diagnostic radiologic examination, echocardiography, exercise stress test, holter electrocardiography, pulse oximetry, palpation, plain chest x-ray, plain x-ray, telemetry, and work-up. **Therapeutic or preventive procedures** include bypass, catheterization, coronary artery bypass surgery, dialysis procedure, hysterectomy, intubating, irradiation of neck, knee strapping, organ transplantation, pacemaker placement, pain control, replacement of aortic valve, stabilization, and triage.

5.0 Discussions, Conclusions, Contributions, and Future Work

This chapter will discuss the algorithms and experimental results. Then, it will present conclusions and discuss research contributions and future directions.

5.1 Discussions

5.1.1 Using QMR Knowledge to Enhance Diagnostic Modeling

When UPMC data sample size was small, EHR-knowledge model performance was better than EHR-model performance. The reason is that with very few training samples, it is hard to distinguish useful features and noisy features. The QMR knowledge helps to emphasize useful features that are highly influential and thus help reduce noisy features.

When UPMC data sample size was large, EHR-knowledge model performance was similar to EHR-model performance. The reason is that with very large training samples, it is easy to distinguish useful features and noisy features using EHR data itself. With very large training samples, a model can find all classification patterns in training data. At that point, adding domain knowledge is not very useful.

5.1.2 Comparison of Strategies for Developing a Sequential Diagnostic System

In Chapter 4, I presented experimental results comparing four different sequential diagnostic models - an SDG phase model, an SDG-no phase model, an ML-C4.5 model, and a DQN model - in three different aspects: diagnostic accuracy,clinical alignment (misdiagnosis costs and meaningfulness of diagnostic path), and medical resource usage. Below, I summarize and discuss these results.

Diagnostic Accuracy (SDG-no-phase > ML-C4.5≥ SDG-phase > DQN)

Comparing diagnostic accuracy, results showed that the SDG-no-phase model performed

the best. The SDG-no-phase model leveraged pre-trained Bayesian network models that have been developed using a cross-validation approach to avoid over-fitting. Without control for over-fitting, the ML-C4.5 model began to overfit the data when depth was increased. The SDG-phase model was developed by restricting the feature search process in a small set of annotated candidate features in a particular phase. With this restriction, the diagnostic path can lead to the choice of checking a syndrome over requiring a laboratory test. When we required these models to make diagnosis within 15 steps, the SDG-phase model performed slightly worse than the SDG-no-phase model and the ML-C4.5 model, which do not restrict the search on the features' phase stage. The discussions about DQN are provided in Sections 5.1.5. and 5.4.7.

Misdiagnosis Costs (SDG-no-phase< SDG-phase < ML-C4.5)

For misdiagnosis costs, SDG models had lower costs than the ML-C4.5 model. The SDG models' development used a score, the expected misdiagnosis costs, which took into account the expert-defined misdiagnosis costs matrix (Table 13) that weighted disease categories differently. The ML-C4.5 model's development did not consider the misdiagnosis costs matrix and treated disease categories as equally important. Most features in the SDG-phase model were phase 1 features that had weak discriminative ability when making diagnosis. Many features in the SDG-no-phase model were phases 3 and 4 features that had strong discriminative ability when making diagnosis. Therefore, the SDG-phase model's chance of making mistakes were higher than SDG-no-phase model, and it had a higher misdiagnosis cost.

Clinical Meaningful Diagnostic Path (SDG-no-phase > SDG-phase > ML-C4.5)

Regarding diagnostic paths' clinical meaningfulness, the SDG models' paths were more clinically aligning than the ML-C4.5 model's paths. Because the ML-C4.5 algorithm is purely data-driven, it sometimes induced a few noisy features that may be clinically irrelevant to a target disease. My SDG algorithm, on the other hand, relies on pre-built Bayesian network models, of which the development process involved cross validation that can help reduce noisy features. Because SDG-no-phase search was not restricted to phases to find the best feature, it included many more clinically discriminative features, which enhanced its clinical

meaningfulness. Although the physician's annotation of phase information help SDG-phase model reduce a few noisy features, without many important clinical features, SDG-phase model was less clinically aligning than the SDG-no-phase model.

Resource Usage (SDG-phase < SDG-no-phase < ML-C4.5)

In respect to resource usage, the SDG-phase model used the fewest features, most of which were signs, symptoms, or findings. The ML-C4.5 model used the highest number of features. The ML-C4.5 model (depth 10) includes 39 laboratory procedures, diagnostic procedures, or therapeutic or preventive procedures. The SDG-no-phase model (depth 10) used 10 of these types of procedures, while at the same time, it reached a high diagnostic accuracy, highest clinical alignment, and lowest misdiagnosis costs. Overall, the SDG-no-phase algorithm performed better than the ML-C4.5 algorithm.

5.1.3 Combining EHR Data and Medical Knowledge to Develop Sequential Diagnostic Models

This dissertation explored strategies to integrate medical knowledge into the EHR machine learning process. Use of three types of knowledge was studied: an expert-defined diagnostic system, expert-defined misdiagnosis costs, and knowledge of clinical workflow.

I used the QMR system as the expert-defined diagnostic system. Since the 1970s, the development of the QMR system represents rich medical knowledge (diseases, clinical findings, and their relationships) rigorously defined by physician experts. One of the creators of QMR, Dr. Randolph A. Miller, kindly generated synthetic data of visits having heart diseases and shared them with me. With these synthetic data, I generated Naive Bayes models for diagnosing heart diseases (called QMR models in my dissertation). Naive Bayes models can represent QMR knowledge well. Splitting synthetic data into training and testing data, the developed Naive Bayes models reached very high AUROCs in the testing portion of the synthetic data (CCS 7.2.1: 0.971; CCS 7.2.2: 0.976; CCS 7.2.3: 0.998; CCS 7.2.4: 0.987; CCS 7.2.6: 0.983; CCS 7.2.7: 0.978; CCS 7.2.11: 0.972).

I designed a Bayesian approach to add the QMR models' knowledge into EHR modeling

(Section 3.2.2), where the degree of adding knowledge can be adjusted by setting different values of equivalent sample size. I tested the approach's performance in different scenarios, varying the sample size of EHR data and the value of equivalent sample size. Results showed that adding the QMR model's knowledge can help significantly reduce noisy features and improve model performance when there is a small amount of EHR data; QMR model may weight equally (or slightly less) important as EHR data. When I used a large amount of UPMC EHR data for modeling, adding knowledge did not help much in regard to performance. Further investigation may focus on whether adding knowledge can improve the robustness of a developed model when applying it to a healthcare system over years or to a different healthcare system.

The next domain knowledge I studied is the expert-defined misdiagnosis costs for sequential diagnostic modeling. The misdiagnosis costs could be measured in terms of a patient's physical and/or mental health, and clinical resource usage. The goal of this study was not to provide an economic or utility assessment of misdiagnosis costs. Instead, the misdiagnosis costs were used to develop sequential diagnostic models that penalized more heavily on severe mistakes during the model development process. To evaluate model performance, I requested a physician expert to provide a score (ranged from 1 to 10) to indicate the different degree of costs of different diagnostic errors, and integrated the resulting diagnostic matrix into the search process of the SDG algorithm. The developed SDG models had lower misdiagnosis costs in testing data than the ML-C4.5 model, which did not use the misdiagnosis costs information for model development. Future work can investigate how to create more accurate assessments of costs of various mistakes in the modeling process as well as comprehensive economic analysis or utility analysis on computational sequential diagnosis for clinical decision support in ED.

In addition, to integrate the knowledge of clinical workflow, features were grouped in four phases: the triage nurse phase (1), the physician interview phase (2), the physician exam phase (3), and the lab test phase (4). Other were labelled as 5. The SDG-phase model was developed with the search process restricted to exhaustively choose a useful feature in an earlier phase before going to a next phase, aiming to make the SDG-phase model more in alignment with clinical workflow. Comparing paths of representative encounters, the SDG-

phase model was more clinically aligning than the ML-C4.5 model. However, the SDG-phase model was less clinically aligning than the SDG-no-phase model, which may result from the reduced diagnostic power of the SDG-phase model. On the other hand, integrating the knowledge of clinical workflow made the SDG-phase model efficient - but only when relying on syndromes and findings, and not replying on any laboratory exam.

5.1.4 Comparison between the SDG-phase Algorithm and the SDG-no-phase Algorithm

When developing diagnostic sequence models, both SDG-phase and SDG-no-phase algorithm conducted a greedy search strategy with the same score – the change in expected misdiagnosis costs estimated using diagnostic models, the expert-developed misdiagnosis matrix, and encounters' feature values in the training data. The difference between the two algorithms is that the SDG-phase algorithm firmly restricts its search in one phase until no feature can reduce the expected misdiagnosis costs. The SDG-no-phase algorithm does not have this restriction.

With this restriction, the generated SDG-phase model used fewest procedures. The generated SDG-phase model actually performed less accurate than the SDG-no-phase model. That is because the SDG-phase model included too many phase 1 feature (nurse-triage feature) and almost no features from other phases. And because of low accuracy, the number of meaningful diagnosis paths of SDG-phase model was lower than that of the SDG-no-phase model.

5.1.5 Deep Q Learning Performance

The reason why deep Q learning model did not perform well in sequential diagnosis is that: (1) The action space is too large. Unlike many reinforcement learning tasks, which have few action options, a sequential diagnosis has a much larger number of action options, because there are so many symptoms to check and so many lab test options. (2) There are unclear immediate rewards. In many reinforcement learning tasks, the environment setting often has clear rules about rewards and can provide explicit information about immediate

rewards. In medical diagnosis, the immediate reward is often unclear and hard to define. The silver standard about rewards in different states (estimated using a deep neural network model) could be very inaccurate. This inaccuracy makes the parameter learning of a deep Q network very difficult.

These challenges were also discussed in the paper [97]. This previous study used a knowledge-infused context-driven (KI-CD) hierarchical reinforcement learning algorithm to learn diagnostic policies for 90 diseases, and compared their algorithm with other algorithms, including Unified Dialogue Policy (UDP) algorithm and hierarchical reinforcement learning (HRL) algorithm. The KI-CD algorithm got the best performance, with a success rate (accuracy) around 0.57; success rate of the UDP was 0.342, and the success rate of HRL was 0.504. The authors state the reason for the low performance is: "the underlying huge action space (all symptoms + all diseases + additional actions such as greeting and close), which requires handling by a single policy." Therefore, further exploration of new algorithms for sequential diagnosis may focus on reducing the action space and providing a better estimation of immediate rewards.

5.1.6 Comparison of the SDG-no-phase Algorithm with Gorry and Barnett's Classical Algorithm

Dr. Anthony G. Gorry and Octo G. Barnett are two pioneers in computational medicine. Their work on computer-aided sequential diagnosis [30] is the starting point of this research project.

In this section, I use a simple example to compare the SDG-no-phase algorithm with Gorry and Barnett's algorithm [30], followed by a summary table of the differences of the two algorithms. Because the main difference between the SDG-phase algorithm and the SDG-no-phase algorithm is whether to use phase to restrict the search of features in a smaller space, I only compare the SDG-no-phase algorithm with Gorry and Bernett's algorithm. Since Gorry and Barnett's algorithm can only handle scenarios when patients only have one disease, in the simple example, every patient has one of the three diseases (d1, d2, and d3). For these patients, a physician can order three tests (t_1, t_2, and t_3).

Gorry and Barnett's Algorithm

In Gorry and Barnett's approach [30], for a three-disease scenario, six misdiagnosis costs (called decision losses in the paper) were defined: diagnosing a patient with d_1 to have d_2 as $cost(d_1 \rightarrow d_2)$, and similarly, $cost(d_1 \rightarrow d_3)$, $cost(d_2 \rightarrow d_1)$, $cost(d_2 \rightarrow d_3)$, $cost(d_3 \rightarrow d_1)$, and $cost(d_3 \rightarrow d_2)$. There is also a defined knowledge base about the conditional probabilities of $P(s_j|d_i)$ where s_1, s_2, s_3, s_4 are four attributes that can be observed through tests with some testing costs ($t_1 \rightarrow s_1$, $t_2 \rightarrow s_2$, $t_3 \rightarrow s_3$ and s_4). At each decision point, the system can use Bayes rules to get an estimated distribution of diseases based on current observed evidence. At a particular time point, the probabilities of having d_1, d_2, and d_3 are p_1, p_2, and p_3 respectively. Then, the expected cost of misdiagnosis (denoted by Q in their paper) is the **minimum value** of the three values listed below:

- If making a diagnosis of d_1, then the cost is $p_1 \times 0 + p_2 \times cost(d_2 \rightarrow d_1) + p_3 \times cost(d_3 \rightarrow d_1)$.
- If making a diagnosis of d_2, then the cost is $p_1 \times cost(d_1 \rightarrow d_2) + p_2 \times 0 + p_3 \times cost(d_3 \rightarrow d_2)$.
- If making a diagnosis of d_3, then the cost is $p_1 \times cost(d_1 \rightarrow d_3) + p_2 \times cost(d_2 \rightarrow d_3) + p_3 \times 0$.

If required to make a diagnosis at this time point, then the diagnosis will be the disease that corresponds to the lowest cost among the three values. From these formulas, we can see that Gorry and Barnett's algorithm did not use the true disease statuses of patients in a collected training dataset. It did not need a training dataset and instead used p_i to estimate the probability of having the disease d_i for a patient.

Gorry and Barnett's algorithm chooses the next test for a given patient by comparing the expected diagnosis values (denoted as EDC in their paper) that are associated with each of the three test options. Calculating the EDC value for a test t_1 considers possible results of the attribute s_1 that the test brings with the formula: $ETC_1 = C_1 + \sum_k P\{s_k\} \times Q_k$, where C_1 is the cost of conducting test t_1, and s_k is a possible result of test t_1 and Q_k is the expected cost of an optimal decision after using the value of s_k to update the estimation of the distribution of diseases and choose the lowest value of expected misdiagnosis costs. The test with the lowest EDC was selected as the next test for the patient.

SDG-no-phase Algorithm

For the three-disease example, the misdiagnosis costs matrix used by the SDG algorithm

also includes six values: $M(\bar{d_1}|d_1)$, $M(d_1|\bar{d_1})$, $M(\bar{d_2}|d_2)$, $M(d_2|\bar{d_2})$, $M(\bar{d_3}|d_3)$, $M(d_3|\bar{d_3})$. The meaning of these six values is different from those used in Gorry and Barnett's algorithm. $M(\bar{d_i}|d_i)$ is the cost of ignoring a disease when a patient has the disease d_i, while $M(d_i|\bar{d_i})$ is the cost of diagnosing a disease d_i when a patient does not have the disease.

When developing a sequential diagnostic model, the training dataset includes patients for which the true disease statuses are provided. Let the total number of patients be n; the number of patients with d_1 be n_1; the number of patients with d_2 be n_2; the number of patients with d_3 be n_3. With certain clinical evidence, three Bayesian network models provide three probabilities, $p(d_1|E)$, $p(d_2|E)$, and $p(d_3|E)$. For a patient with a true disease status of d_1, the expected misdiagnosis costs is $(1 - p(d_1|E)) \times M(\bar{d_1}|d_1) + p(d_2|E) \times M(d_2|\bar{d_2}) + p(d_3|E) \times M(d_3|\bar{d_3})$. The rational is that for this patient, three types of diagnosis mistakes can be made: ignoring d_1, concluding that the patient has d_2, and concluding that the patient has d_3. The probability of making the first mistake is $1 - p(d_1|E)$. This assumes we are going to flip a coin with probability $p(d_1|E)$ of being "heads" (i.e., make a diagnosis of d_1). This assumption mimics the scenario that a computer agent makes a diagnosis with this flipping coin approach. When the computer agent works, the probability of not diagnosing d_i is $1 - p(d_1|E)$ for every patient. If a patient has the disease, then the diagnosis of not having d_i is a diagnostic error. Similarly, the probability of making the second mistake is $p(d_2|E)$, and for the third mistake $p(d_3|E)$. Thus, the costs of these three mistakes are weighted and summed up together as the expected misdiagnosis costs for a patient of which the true disease status is d_1. Since there are n_1 patients with d_1, the total costs of this type of patients in the training population is: $n_1 \times [(1-p(d_1|E)) \times M(\bar{d_1}|d_1) + p(d_2|E) \times M(d_2|\bar{d_2}) + p(d_3|E) \times M(d_3|\bar{d_3})]$. Similarly, the total costs of patients with d_2 in the training population is: $n_2 \times [(p(d_1|E)) \times M(d_1|\bar{d_1}) + (1-p(d_2|E)) \times M(\bar{d_2}|d_2) + p(d_3|E) \times M(d_3|\bar{d_3})]$. And the total costs of patients with d_3 in the training population is: $n_3 \times [(p(d_1|E)) \times M(d_1|\bar{d_1}) + p(d_2|E) \times M(d_2|\bar{d_2}) + (1 - p(d_3|E)) \times M(\bar{d_3}|d_3)]$. The total cost for all patients in the population is: $C(E) = n_1 \times [(1 - p(d_1|E)) \times M(\bar{d_1}|d_1) + p(d_2|E) \times M(d_2|\bar{d_2}) + p(d_3|E) \times M(d_3|\bar{d_3})] + n_2 \times [(p(d_1|E)) \times M(d_1|\bar{d_1}) + (1 - p(d_2|E)) \times M(\bar{d_2}|d_2) + p(d_3|E) \times M(d_3|\bar{d_3})] + n_3 \times [(p(d_1|E)) \times M(d_1|\bar{d_1}) + p(d_2|E) \times M(d_2|\bar{d_2}) + (1 - p(d_3|E)) \times M(\bar{d_3}|d_3)]$. The SDG algorithm requires a large number of training data, where the true disease status of each patient is known. And

the $p(d_i|E)$, the probability of making a diagnosis d_i diagnosis decision, is estimated by a pre-trained Bayesian network.

In this simple example setting, four attributes can be observed through three tests ($t_1 \rightarrow s_1$, $t_2 \rightarrow s_2$, $t_3 \rightarrow s_3$ and s_4). When the SDG algorithm needs to choose the next test among three testing options, it compares the reductions of misdiagnosis costs and finds the largest one, $R(E, s_1)$, $R(E, s_2)$, and $R(E, s_3, s_4)$ where $R(E, s_1) = C(E) - C(E, s_1)$, $R(E, s_2) = C(E) - C(E, s_2)$, and $R(E, s_3, s_4) = C(E) - C(E, s_3, s_4)$. Since the three values have the same C(E), the SDG algorithm actually finds the smallest value among $C(E, s_1)$, $C(E, s_2)$, and $C(E, s_3, s_4)$. Recall that the ETC_1 value in Gorry and Barnett's algorithm is calculated using $ETC_1 = C_1 + \sum_k P\{s_k\} \times Q_k$, where C_1 is the cost of conducting test t_1, and s_k is a possible result of test t_1 and Q_k is the expected cost of an optimal decision after using the value of s_k to update the estimation of the distribution of diseases and choose the lowest value of expected misdiagnosis costs. Differently, the SDG algorithm does not consider the cost of conducting test t_1, C_1, and $C(E, s_1) = \sum_k \sum_x C(E, s_k, x) = \sum_k [n\{s_k, E\} \times C(E, s_k)]$ where x refers to individual patient encounter and $n_{s_k,E}$ is the number of encounters having the same value of s_k and E. If dividing the same n (total number of encounter), then $C(E, s_1)/n = \sum_k proportion\{s_k, E\} \times C(E, s_k)$, which is similar to the second part of the ETC_1 formula: $\sum_k P\{s_k\} \times Q_k$. Note that $C(E, s_k)$ is different from Q_k. In SDG-no-phase and SDG-phase algorithm, $C(E, s_k) = n_{1,s_k} \times [(1 - p(d_1|E, s_k)) \times M(\bar{d_1}|d_1) + p(d_2|E, s_k) \times M(d_2|\bar{d_2}) + p(d_3|E, s_k) \times M(d_3|\bar{d_3})] + n_{2,s_k} \times [(p(d_1|E, s_k)) \times M(d_1|\bar{d_1}) + (1 - p(d_2|E, s_k)) \times M(\bar{d_2}|d_2) + p(d_3|E, s_k) \times M(d_3|\bar{d_3})] + n_{3,s_k} \times [(p(d_1|E, s_k)) \times M(d_1|\bar{d_1}) + p(d_2|E, s_k) \times M(d_2|\bar{d_2}) + (1 - p(d_3|E, s_k)) \times M(\bar{d_3}|d_3)]$. While for Gorry and Bernett's algorithm, Q_k=min$\{p_2(E, s_k) \times cost(d_2 \rightarrow d_1) + p_3(E, s_k) \times cost(d_3 \rightarrow d_1)$, $p_1(E, s_k) \times cost(d_1 \rightarrow d_2) + p_3 \times cost(d_3 \rightarrow d_2)$, $p_1(E, s_k) \times cost(d_1 \rightarrow d_3) + p_2(E, s_k) \times cost(d_2 \rightarrow d_3)\}$.

Gorry and Barnett's algorithm was developed in the 1960s, when medical expert systems were manually developed with domain knowledge. The SDG algorithm considers multi-label diagnosis scenarios, reduces the workload of defining the misdiagnosis costs matrix, and leverages a large amount of training label for modeling and probability inference.

In summary, table 17 compares the SDG-no-phase algorithm with Gorry and Barnett's algorithm. Gorry and Barnett's algorithm for sequential diagnosis can only handle the

situation in which a patient only has one disease and a clinician only makes a diagnosis of categorizing the patient into one disease. Even for one disease scenario, the approach of calculating expected misdiagnosis costs is different (as shown in the simple example above). Gorry and Barnett's algorithm provides a sequential diagnosis path for individual patients, while the SDG algorithm generates a sequential diagnosis tree for a population of future patients to be diagnosed. Further discussion of population machine learning and personalized machine learning is provided in Section 5.4.6.

Table 17: Comparisons between Gorry & Barnett's Algorithm and SDG-no-phase Algorithm

Aspect	Gorry and Barnett's algorithm	SDG-no-phase algorithm
Approach	Individual model approach	Population model approach
Disease number	Each patient has one disease	A patient can have no disease, one disease, or multiple diseases
Disease probability distribution given evidence, $P(d_i\|E)$	The probabilities can be added up to 1 so this algorithm may only leverage one underlying model for probability inference. These probabilities are used as an estimation of how likely a patient has the disease.	The SDG algorithm uses multiple pre-trained naive Bayes models to estimate the probabilities. Since the training dataset has the true disease status, these probabilities are used to represent how likely a system makes each diagnosis and thus makes mistakes.
Parameters for Bayes inference	Expert-defined	Learned from EHR data and QMR knowledge

Completeness of mistakes	Does not include a mistake that a patient with d_i was diagnosed as healthy (not having any disease) and does not include a mistake that a healthy patient was diagnosed to as having a disease	Includes all type of mistakes
Training data	Not need	Needed
Calculate misdiagnosis costs of one node	Do not need true disease label of the patient. Directly calculate the misdiagnosis costs for a new patient when identifying the diagnosis path.	Need true disease labels from training data. Once obtained a population model, need not calculate the misdiagnosis costs for a new patient.
The cost of conducting a test	Defined and considered	Not included
Provide optimal diagnosis	Yes, the one with the minimal diagnosis cost is the optimal diagnosis	The developed model provides the probability estimation of all potential diagnoses. In a multi-disease scenario, users can set up thresholds to get a set of diagnoses. If assuming a single disease scenario, users can choose the disease with the highest value of probabilities as the final diagnosis.
Output of the algorithm	Each patient gets one diagnosis path	One diagnostic tree for the population

5.2 Conclusions

The SDG-no-phase algorithm combines EHR data and knowledge about misdiagnosis costs to develop a sequential diagnostic system. In a heart-disease diagnosis task using real-world ED EHR data, the SDG-no-phase model is more clinically aligning than a model learned by a classic data-driven algorithm (i.e., ML-C4.5). Knowledge about misdiagnosis costs help drive the development of a sequential diagnostic system towards more cost-efficient diagnosis paths, which make fewer severe mistakes.

When using a small sample size of EHR data for developing diagnostic models, medical knowledge about the relationships between disease and clinical manifestations emphasizes influential features and reduces noisy features. To maximize the benefits of medical knowledge in the small sample size EHR scenario, balanced weights on medical knowledge and EHR data need to be reached.

When a large amount of EHR data are available locally, medical knowledge about the relationships between disease and clinical manifestations often has few benefits and could be harmful for the local diagnosis task.

For sequential diagnostic modeling, knowledge about different phases in clinical workflow help restrict diagnosis processes to strictly follow the order of phases (e.g., asking questions about symptoms before ordering laboratory tests). Too strict restrictions lead to poor diagnosis performance, indicating that some relaxations are needed.

5.3 Contributions

This study has five main contributions for developing medical sequential diagnostic systems:

1. It found that the prediction performance of a disease diagnostic model could be improved only when the real data and domain knowledge reached a balance point, too little or too much domain knowledge might reduce the performance of the model.

2. It developed SDG-no-phase and SDG-phase algorithm algorithms to learn sequential diagnosis rules from multi-disease data.
3. It introduced a new approach to calculate misdiagnosis probabilities for multi-disease sequential diagnosis. This new method can consider patient-reported information before selecting sequential questions for diagnosis.
4. It introduced a new method to define misdiagnosis costs. It makes it possible to use the misdiagnosis costs as a criterion to process multi-disease data.
5. It defined some new measurements to evaluate clinical alignment. (1) Use AUROC value to evaluate Naive Bayes Model's clinical alignment; (2) insert real patient records into classification tree models, use clinical meaningful paths to evaluate clinical alignment.

Although the SDG algorithms were developed and tested for sequential disease diagnosis, these algorithms may find their applications in non-biomedical domains (e.g., sequential diagnosis for system maintenance and repair) when developing sequential decision making systems for multi-label classification tasks with the consideration of different misclassification costs of different mistakes. The SDG algorithm also has the option to restrict decisions in different phases that can be used in other tasks.

5.4 Limitations and Future Work

In the last portion of this dissertation, I will discuss limitations of this research and the future work that may arise from them.

5.4.1 Limitation 1. Use CCS Category for Disease Modeling

In this study, I merged 238 heart disease ICD-9-CM codes into 11 categories, and used these categories as outcome variables to build models. The reason why I did so is because there are not enough cases to build a model for 238 ICD codes separately.

The pros of this approach is that this rich data set allowed me to develop reliable models and conduct multiple experiments to thoroughly study the effects of my introduced algo-

rithms and the effects of retrieved domain knowledge for sequential diagnosis. However, these designs greatly restrict the usage of the developed sequential diagnostic models. The cons of this approach include: (1) The value of the QMR system's medical knowledge for EHR disease modeling has only been partially explored in this CCS category classification setting. (2) The developed model can only distinguish among CCS categories, which will be insufficient for real-world usage for medical education and clinical decision support purpose. (3) Each CCS category actually includes multiple diseases (ICD codes) with different clinical manifestations, which may increase the difficulty of getting a more accurate model.

5.4.2 Limitation 2. Only Focus on Encounters with Heart Diseases in ED

This study used heart diseases as an example to compare sequential diagnostic modeling strategies. The developed sequential model only focused on heart disease related CCS category classification. This model may not be ready to be directly used in real ED setting. Unlike specialist's office, emergency rooms often have a mixture of patients with different diseases in different systems. Many non-heart diseases also share some clinical manifestations with heart diseases. The developed sequential model can currently only diagnosis CCS category diseases. If all these probabilities are low, then the patient may have other diseases. The possibilities of having other diseases are not provided.

When physicians first meet a patient, they do not know their diagnoses. The diagnosis process often begins with learning the patient's chief complaint, collecting information, and then performing diagnosis. In a preliminary study, I tried to use one chief complaint "chest pain" to select patient encounters, and found that the selected dataset contained a very large number of different diagnoses (many of them were not heart diseases), and most diagnoses did not have sufficient samples for machine learning. So, in this study, I retrieved research data based on their ICD codes and only focused on heart diseases. In this way, the number of diseases is not too large, so that we can study and compare sequential diagnosis algorithms in a tractable and focused manner.

5.4.3 Limitation 3. Assumptions in SDG Algorithms

Both SDG-phase and SDG-no-phase algorithm use the same approach to calculate expected misdiagnosis costs for a population, with three assumptions: (1) The cost of misdiagnosis for individual patients is independent of each other. (2) For each individual patient, the cost related to one disease is independent of the cost related to any other diseases. (3) For each individual patient, the diagnosis of having a disease is independent of the diagnosis of having another disease. These assumptions were made to reduce the size of misdiagnosis matrix and simplify the estimation of misdiagnosis costs so that the costs are all additive.

However, these assumptions, especially the second and the third assumption may not be true in real world clinical practice. For the second assumption, the cost of making two mistakes may be not directly equal to the sum of the cost of each of two mistakes. For example, under the second assumption, when a patient only has a disease d_1, mistakenly diagnosing that the patient only has a disease d_2, the cost is $M(\bar{d_1}|d_1) + M(d_2|\bar{d_1})$. Ignoring a disease d_1 and treating the patient in a wrong direction (d_2) may lead to a larger cost than the sum of just ignoring the disease d_1 or treating a healthy patient for disease d_2. For the third assumption, disease diagnosis may not be independent. Making a diagnosis of having disease d_1 may not be independent of making a diagnosis of not having disease d_2, because having disease d_1 may explain away a patient's clinical manifestations and reduce the chance of making a diagnosis of having disease d_2.

Future study may calculate misdiagnosis costs by obtaining statistical summaries from real administrative data so that the misdiagnosis matrix can cover a large number of scenarios with more accurate estimation of costs in real clinical practice. With a more flexible design of the expected misdiagnosis costs, the diagnostic model can include diseases in one integrated model to consider the relationships among diagnosis of correlated diseases.

5.4.4 Future Work on SDG-phase and SDG-no-phase Algorithms

In this project, the SDG-phase algorithm uses phase information that all features are split into four phases. When the SDG-phase algorithm learns the model, it searches phase-1 features first. It will keep including phase-1 features in the model even if some features only

have very low misdiagnosis cost reduction while many features in phase 2-4 can lead to much larger reductions on misdiagnosis costs.

The disadvantage of this approach is: (1) Because the model contains too much phase-1 features and too few phase-4 (lab test) features, the model performance is poor. The classification capability of phase-4 features is much larger than phase 1 features. (2) Because the number of phase 4 features is limited, the clinical alignment of SDG-phase model is worse than SDG-no-phase model.

In the future, we plan to investigate a new SDG-phase algorithm with the following changes: (1) requiring the reduction of misdiagnosis costs to be greater than a threshold may help the search escape quickly from an earlier phase to explore useful features in next phase; (2) allowing the SDG-phase algorithm to occasionally violate the phase restriction with some heuristic penalty that is counted into the searching cost. For both changes, identifying an optimal threshold and a heuristic penalty may need multiple rounds of experiments with training data.

Moreover, both SDG-phase and SDG-no-phase algorithms use a greedy search strategy, which only focuses on the immediate gain. Therefore, another future improvement of these two algorithms could be to develop a more sophisticated score that can consider cumulative reward and long-term effects.

5.4.5 Future Work on Incorporating Medical Knowledge into EHR Machine Learning

The wide adoption of EHR systems provides us with massive amounts of clinical data, while innovative machine learning algorithms and increased computational power bring us the hope that an era of computational medicine will soon arrive. However, though it seems that most computational models and systems are successful in research lab, it is still questionable whether these models and systems can assist clinicians very well in real-world practice in the near future, let alone they can act alone and completely replace clinicians. One critical reason why physicians do not trust most computational models and systems is that they lack an essential source [36]: medical knowledge accumulated over the course of decades of

practice that guides physicians and their peers in real-world medical practice.

One strategy to enhance the performance of computational medicine that we have presented here is to incorporate medical knowledge into EHR machine learning. This involves four steps: retrieving medical knowledge from multiple sources (e.g., medical books, medical literature, and commercial website such as upToDate), representing knowledge, incorporating knowledge into EHR machine learning, and evaluating performance.

In future studies, these four steps should be integrated into one framework where performance evaluation can provide a feedback loop to the adjustment of knowledge retrieval, representation, and incorporation. This work needs the collaboration of experts in multiple disciplines to bring together use of information retrieval techniques, natural language processing, knowledge representation, and machine learning. Moreover, the whole development process also must involve judgements from the users - clinicians. Beyond accuracy measurement in a research lab, the final system needs to have good interpretability, actionability, and reliability, as well as the potential for discovering new scientific knowledge. Moreover, because medical knowledge doubles every few months [18], the final system should also be able to automatically update using online learning.

5.4.6 Future Work on Personalized Machine Learning for Sequential Diagnosis

The SDG algorithms belong to population machine learning, which derives a single model from a large amount of training data. With this strategy, the developed model is optimized to perform well on average on all future individuals. However, the population machine learning approach [104] may ignore important differences among patients, especially patients with rare conditions.

In the future, to better capture individual differences in machine learning process, I will try the personalized machine learning strategy [104], which aims to tailor model with the individual patient's clinical conditions. The developed model can optimize well for each individual patient [14]. Briefly, for sequential diagnostic modeling, personalized machine learning strategy can be applied when developing probability models and estimating expected misdiagnosis costs so that personalized sequential diagnosis path can be identified for each

individual and the individual misdiagnosis costs can be minimized.

5.4.7 Future Work on Reinforcement Learning for Sequential Diagnosis

In Section 5.1.5, we discussed why the reinforcement learning algorithm did not obtain good performance in sequential diagnosis problems. In this section, I will review a successful example of reinforcement learning for sequential decision making in medicine [45]. I will discuss why this research obtained good performance and the possible implications for our future work.

The authors developed a reinforcement learning agent that learned optimal treatment strategies for sepsis in intensive care. Using rich EHR data from Medical Information Mart for Intensive Care version III (MIMIC III), the agenet extracted implicit knowledge learned optimal treatment by analyzing a group of (mostly suboptimal) treatment decisions. Tested on the eICU Research Institute Database, the AI decisions were shown to be good treatement strategies: mortality was lowest in patients for whom clinicians' actual doses matched the AI decisions. Main steps in developing and testing this AI agent include: (1) They selected 48 features to observe, including demographics, Elixhauser premorbid status, vital signs, laboratory values, fluids and vasopressors received; (2) they included up to 72 hours of measurements taken around the estimated time of onset of sepsis. The data were split into 4-hour time periods; (3) they evaluated the actual treatments ordered by clinicians. When a patient survived, a positive reward was given at the end of the treatment (+100), and a negative reward was given (-100) if the patient died. They calculated the average reward of each treatment option; (4) because management of ICU patients with sepsis is extremely complex, they only focused on medical decisions regarding total volume of intravenous fluids and maximum dose of vasopressors administered over each 4-hour period. The dose of intravenous fluids was split into 5 choices, and the dose of vasopressors was also split into 5 choices. The combination of the two treatments produced 25 possible actions. Those 25 combinations were the action space in their reinforcement learning model; (5) they randomly chose 80% data as training data, and 20% data as validation data. They clustered all time periods from the training data, getting 750 clusters. In each cluster, individuals in that cluster were

similar with respect to those 48 features. Those 750 clusters were the state space in their reinforcement learning model; (6) they used the training data to build a reinforcement learning model (Q-learning method), and used validation data to evaluate model performance. They repeated this step 500 times, getting 500 models. They compared the performance of those 500 models, and chose the best model; and (7) they compared the performance between the reinforcement learning model and the clinician's model using one independent test data. Under their study assumptions, they found the 95% confidence lower bound of the AI policy consistently exceeded the 95% confidence upper bound of the clinicians' policy.

Difference and Implication

In my study, there are 1,799 features in our data set, and each feature is binary. so in theory, the state space is 2^{1799}, and the action space is $1,799$ plus the number of diseases. The action space and state space are much larger than in their study. Future study of sequential diagnosis using deep reinforcement learning should firstly apply some machine learning approaches or domain knowledge to greatly reduce the action space. Moreover, their study used training data that already contained sequential decisions while my study did not leverage this type of data. Future studies should leverage sequential decision data for encounters in the ED to better conduct the reinforcement learning.

Challenges and Future Directions

Deep reinforcement learning is a subfield of machine learning that leverages deep neural networks in reinforcement learning. Deep reinforcement learning has been successfully applied in areas such as video games, robotics, natural language processing, computer vision, as well as clinical domains.

Successfully incorporating deep reinforcement learning into sequential decision modeling for medicine still has many challenges [34]. It is difficult to exam confounding factors' short term and long-term effects. Discovering new treatment approaches through reinforcement learning with observational data is still less reliable than refining existing practices. Other challenges include simplistic reward functions design and potential domain shifting.

Given these challenges, a few future directions include: (1) improving the design of in-

termediate reward and final reward using probabilistic estimation approaches and utility analyses; (2) covering more diseases and multi-labeling diseases and addressing the issue of scalability using ensemble learning and hierarchical learning; (3) exploring causal reinforcement learning that combines causal inference with reinforcement learning, which deal with data in both an interventional and counterfactual manner [41]. Causal reinforcement learning will have the benefit of combining structural invariances of causal inference with the sample efficiency of reinforcement learning [9].

6.0 APPENDIX A. Heart Disease ICD Codes and the CCS Categories

Table 18: Heart Disease CCS Categories

ICD code	Description	CCS category
3940	Mitral stenosis	7.2.1
3941	Rheumatic mitral insufficiency	7.2.1
3942	Mitral stenosis with insufficiency	7.2.1
3949	Other and unspecified mitral valve diseases	7.2.1
3950	Rheumatic aortic stenosis	7.2.1
3951	Rheumatic aortic insufficiency	7.2.1
3952	Rheumatic aortic stenosis with insufficiency	7.2.1
3959	Other and unspecified rheumatic aortic diseases	7.2.1
3960	Mitral valve stenosis and aortic valve stenosis	7.2.1
3961	Mitral valve stenosis and aortic valve insufficiency	7.2.1
3962	Mitral valve insufficiency and aortic valve stenosis	7.2.1
3963	Mitral valve insufficiency and aortic valve insufficiency	7.2.1
3968	Multiple involvement of mitral and aortic valves	7.2.1
3969	Mitral and aortic valve diseases, unspecified	7.2.1
3970	Diseases of tricuspid valve	7.2.1
3971	Rheumatic diseases of pulmonary valve	7.2.1
3979	Rheumatic diseases of endocardium, valve unspecified	7.2.1
4240	Mitral valve disorders	7.2.1
4241	Aortic valve disorders	7.2.1
4242	Tricuspid valve disorders, specified as nonrheumatic	7.2.1
4243	Pulmonary valve disorders	7.2.1
42490	Endocarditis, valve unspecified, unspecified cause	7.2.1
42491	Endocarditis in diseases classified elsewhere	7.2.1
42499	Other endocarditis, valve unspecified	7.2.1

7852	Undiagnosed cardiac murmurs	7.2.1
7853	Other abnormal heart sounds	7.2.1
V422	Heart valve replaced by transplant	7.2.1
V433	Heart valve replaced by other means	7.2.1
4250	Endomyocardial fibrosis	7.2.2
4251	Hypertrophic cardiomyopathy	7.2.2
42511	Hypertrophic obstructive cardiomyopathy	7.2.2
42518	Other hypertrophic cardiomyopathy	7.2.2
4252	Obscure cardiomyopathy of Africa	7.2.2
4253	Endocardial fibroelastosis	7.2.2
4254	Other primary cardiomyopathies	7.2.2
4257	Nutritional and metabolic cardiomyopathy	7.2.2
4258	Cardiomyopathy in other diseases classified elsewhere	7.2.2
4259	Secondary cardiomyopathy, unspecified	7.2.2
03282	Diphtheritic myocarditis	7.2.2
03640	Meningococcal carditis, unspecified	7.2.2
03641	Meningococcal pericarditis	7.2.2
03642	Meningococcal endocarditis	7.2.2
03643	Meningococcal myocarditis	7.2.2
07420	Coxsackie carditis, unspecified	7.2.2
07421	Coxsackie pericarditis	7.2.2
07422	Coxsackie endocarditis	7.2.2
07423	Coxsackie myocarditis	7.2.2
11281	Candidal endocarditis	7.2.2
11503	Infection by Histoplasma capsulatum, pericarditis	7.2.2
11504	Infection by Histoplasma capsulatum, endocarditis	7.2.2
11513	Infection by Histoplasma duboisii, pericarditis	7.2.2
11514	Infection by Histoplasma duboisii, endocarditis	7.2.2

11593	Histoplasmosis, unspecified, pericarditis	7.2.2
11594	Histoplasmosis, unspecified, endocarditis	7.2.2
1303	Myocarditis due to toxoplasmosis	7.2.2
3910	Acute rheumatic pericarditis	7.2.2
3911	Acute rheumatic endocarditis	7.2.2
3912	Acute rheumatic myocarditis	7.2.2
3918	Other acute rheumatic heart disease	7.2.2
3919	Acute rheumatic heart disease, unspecified	7.2.2
3920	Rheumatic chorea with heart involvement	7.2.2
393	Chronic rheumatic pericarditis	7.2.2
3980	Rheumatic myocarditis	7.2.2
39890	Rheumatic heart disease, unspecified	7.2.2
39899	Other rheumatic heart diseases	7.2.2
4200	Acute pericarditis in diseases classified elsewhere	7.2.2
42090	Acute pericarditis, unspecified	7.2.2
42091	Acute idiopathic pericarditis	7.2.2
42099	Other acute pericarditis	7.2.2
4210	Acute and subacute bacterial endocarditis	7.2.2
4211	Acute and subacute infective endocarditis in diseases classified elsewhere	7.2.2
4219	Acute endocarditis, unspecified	7.2.2
4220	Acute myocarditis in diseases classified elsewhere	7.2.2
42290	Acute myocarditis, unspecified	7.2.2
42291	Idiopathic myocarditis	7.2.2
42292	Septic myocarditis	7.2.2
42293	Toxic myocarditis	7.2.2
42299	Other acute myocarditis	7.2.2
4230	Hemopericardium	7.2.2

4231	Adhesive pericarditis	7.2.2
4232	Constrictive pericarditis	7.2.2
4233	Cardiac tamponade	7.2.2
4238	Other specified diseases of pericardium	7.2.2
4239	Unspecified disease of pericardium	7.2.2
4290	Myocarditis, unspecified	7.2.2
4100	Bone marrow transplant, not otherwise specified	7.2.3
41000	Acute myocardial infarction of anterolateral wall, episode of care unspecified	7.2.3
41001	Acute myocardial infarction of anterolateral wall, initial episode of care	7.2.3
41002	Acute myocardial infarction of anterolateral wall, subsequent episode of care	7.2.3
4101	Acute myocardial infarction of other anterior wall	7.2.3
41010	Acute myocardial infarction of other anterior wall, episode of care unspecified	7.2.3
41011	Acute myocardial infarction of other anterior wall, initial episode of care	7.2.3
41012	Acute myocardial infarction of other anterior wall, subsequent episode of care	7.2.3
4102	Acute myocardial infarction of inferolateral wall	7.2.3
41020	Acute myocardial infarction of inferolateral wall, episode of care unspecified	7.2.3
41021	Acute myocardial infarction of inferolateral wall, initial episode of care	7.2.3
41022	Acute myocardial infarction of inferolateral wall, subsequent episode of care	7.2.3
4103	Acute myocardial infarction of inferoposterior wall	7.2.3

41030	Acute myocardial infarction of inferoposterior wall, episode of care unspecified	7.2.3
41031	Acute myocardial infarction of inferoposterior wall, initial episode of care	7.2.3
41032	Acute myocardial infarction of inferoposterior wall, subsequent episode of care	7.2.3
4104	Acute myocardial infarction of other inferior wall	7.2.3
41040	Acute myocardial infarction of other inferior wall, episode of care unspecified	7.2.3
41041	Acute myocardial infarction of other inferior wall, initial episode of care	7.2.3
41042	Acute myocardial infarction of other inferior wall, subsequent episode of care	7.2.3
4105	Acute myocardial infarction of other lateral wall	7.2.3
41050	Acute myocardial infarction of other lateral wall, episode of care unspecified	7.2.3
41051	Acute myocardial infarction of other lateral wall, initial episode of care	7.2.3
41052	Acute myocardial infarction of other lateral wall, subsequent episode of care	7.2.3
4106	True posterior wall infarction	7.2.3
41060	True posterior wall infarction, episode of care unspecified	7.2.3
41061	True posterior wall infarction, initial episode of care	7.2.3
41062	True posterior wall infarction, subsequent episode of care	7.2.3
4107	Subendocardial infarction	7.2.3
41070	Subendocardial infarction, episode of care unspecified	7.2.3
41071	Subendocardial infarction, initial episode of care	7.2.3
41072	Subendocardial infarction, subsequent episode of care	7.2.3

4108	Acute myocardial infarction of other specified sites	7.2.3
41080	Acute myocardial infarction of other specified sites, episode of care unspecified	7.2.3
41081	Acute myocardial infarction of other specified sites, initial episode of care	7.2.3
41082	Acute myocardial infarction of other specified sites, subsequent episode of care	7.2.3
4109	Acute myocardial infarction of unspecified site	7.2.3
41090	Acute myocardial infarction of unspecified site, episode of care unspecified	7.2.3
41091	Acute myocardial infarction of unspecified site, initial episode of care	7.2.3
41092	Acute myocardial infarction of unspecified site, subsequent episode of care	7.2.3
41406	Coronary atherosclerosis of native coronary artery of transplanted heart	7.2.4
4130	Angina decubitus	7.2.4
4131	Prinzmetal angina	7.2.4
4139	Other and unspecified angina pectoris	7.2.4
4111	Intermediate coronary syndrome	7.2.4
4110	Postmyocardial infarction syndrome	7.2.4
4118	Other acute and subacute forms of ischemic heart disease	7.2.4
41181	Acute coronary occlusion without myocardial infarction	7.2.4
41189	Other acute and subacute forms of ischemic heart disease, other	7.2.4
4140	Coronary atherosclerosis	7.2.4
41400	Coronary atherosclerosis of unspecified type of vessel, native or graft	7.2.4

41401	Coronary atherosclerosis of native coronary artery	7.2.4
4142	Chronic total occlusion of coronary artery	7.2.4
4143	Coronary atherosclerosis due to lipid rich plaque	7.2.4
4144	Coronary atherosclerosis due to calcified coronary lesion	7.2.4
V4582	Percutaneous transluminal coronary angioplasty status	7.2.4
412	Old myocardial infarction	7.2.4
4148	Other specified forms of chronic ischemic heart disease	7.2.4
4149	Chronic ischemic heart disease, unspecified	7.2.4
V4581	Aortocoronary bypass status	7.2.4
78650	Chest pain, unspecified	7.2.5
78651	Precordial pain	7.2.5
78659	Other chest pain	7.2.5
4150	Acute cor pulmonale	7.2.6
4151	Pulmonary embolism and infarction	7.2.6
41512	Septic pulmonary embolism	7.2.6
41513	Saddle embolus of pulmonary artery	7.2.6
41519	Other pulmonary embolism and infarction	7.2.6
4160	Primary pulmonary hypertension	7.2.6
4161	Kyphoscoliotic heart disease	7.2.6
4162	Chronic pulmonary embolism	7.2.6
4168	Other chronic pulmonary heart diseases	7.2.6
4169	Chronic pulmonary heart disease, unspecified	7.2.6
4170	Arteriovenous fistula of pulmonary vessels	7.2.6
4171	Aneurysm of pulmonary artery	7.2.6
4178	Other specified diseases of pulmonary circulation	7.2.6
4179	Unspecified disease of pulmonary circulation	7.2.6
V1255	Personal history of pulmonary embolism	7.2.6
41410	Aneurysm of heart (wall)	7.2.7

41411	Aneurysm of coronary vessels	7.2.7
41412	Dissection of coronary artery	7.2.7
41419	Other aneurysm of heart	7.2.7
4291	Myocardial degeneration	7.2.7
4292	Cardiovascular disease, unspecified	7.2.7
4293	Cardiomegaly	7.2.7
4295	Rupture of chordae tendineae	7.2.7
4296	Rupture of papillary muscle	7.2.7
42971	Acquired cardiac septal defect	7.2.7
42979	Certain sequelae of myocardial infarction, not elsewhere classified, other	7.2.7
42981	Other disorders of papillary muscle	7.2.7
42982	Hyperkinetic heart disease	7.2.7
42983	Takotsubo syndrome	7.2.7
42989	Other ill-defined heart diseases	7.2.7
4299	Heart disease, unspecified	7.2.7
4260	Atrioventricular block, complete	7.2.8
42610	Atrioventricular block, unspecified	7.2.8
42611	First degree atrioventricular block	7.2.8
42612	Mobitz (type) II atrioventricular block	7.2.8
42613	Other second degree atrioventricular block	7.2.8
4262	Left bundle branch hemiblock	7.2.8
4263	Other left bundle branch block	7.2.8
4264	Right bundle branch block	7.2.8
42650	Bundle branch block, unspecified	7.2.8
42651	Right bundle branch block and left posterior fascicular block	7.2.8
42652	Right bundle branch block and left anterior fascicular block	7.2.8
42653	Other bilateral bundle branch block	7.2.8

42654	Trifascicular block	7.2.8
4267	Anomalous atrioventricular excitation	7.2.8
4266	Other heart block	7.2.8
42681	Lown-Ganong-Levine syndrome	7.2.8
42682	Long QT syndrome	7.2.8
42689	Other specified conduction disorders	7.2.8
4269	Conduction disorder, unspecified	7.2.8
V450	Cardiac device in situ	7.2.8
V4500	Unspecified cardiac device in situ	7.2.8
V4501	Cardiac pacemaker in situ	7.2.8
V4502	Automatic implantable cardiac defibrillator in situ	7.2.8
V4509	Other specified cardiac device in situ	7.2.8
V533	Fitting and adjustment of cardiac device	7.2.8
V5331	Fitting and adjustment of cardiac pacemaker	7.2.8
V5332	Fitting and adjustment of automatic implantable cardiac defibrillator	7.2.8
V5339	Fitting and adjustment of other cardiac device	7.2.8
4270	Paroxysmal supraventricular tachycardia	7.2.9
4271	Paroxysmal ventricular tachycardia	7.2.9
42731	Atrial fibrillation	7.2.9
42732	Atrial flutter	7.2.9
42760	Premature beats, unspecified	7.2.9
42761	Supraventricular premature beats	7.2.9
42769	Other premature beats	7.2.9
42781	Sinoatrial node dysfunction	7.2.9
4272	Paroxysmal tachycardia, unspecified	7.2.9
42789	Other specified cardiac dysrhythmias	7.2.9
4279	Cardiac dysrhythmia, unspecified	7.2.9

7850	Tachycardia, unspecified	7.2.9
7851	Palpitations	7.2.9
42741	Ventricular fibrillation	7.2.10
42742	Ventricular flutter	7.2.10
4275	Cardiac arrest	7.2.10
42820	Systolic heart failure, unspecified	7.2.11
42821	Acute systolic heart failure	7.2.11
42822	Chronic systolic heart failure	7.2.11
42823	Acute on chronic systolic heart failure	7.2.11
42830	Diastolic heart failure, unspecified	7.2.11
42831	Acute diastolic heart failure	7.2.11
42832	Chronic diastolic heart failure	7.2.11
42833	Acute on chronic diastolic heart failure	7.2.11
42840	Combined systolic and diastolic heart failure, unspecified	7.2.11
42841	Acute combined systolic and diastolic heart failure	7.2.11
42842	Chronic combined systolic and diastolic heart failure	7.2.11
42843	Acute on chronic combined systolic and diastolic heart failure	7.2.11
4280	Congestive heart failure, unspecified	7.2.11
39891	Rheumatic heart failure (congestive)	7.2.11
4281	Left heart failure	7.2.11
4289	Heart failure, unspecified	7.2.11

7.0 APPENDIX B. CCS Category Combination Frequency

Table 19: CCS Category Frequency

Number	CCS Category	Count
1	d7289	19163
2	d724	12468
3	d724d7289	4602
4	d721	2226
5	d724d7289d7211	1825
6	d724d7211	1762
7	d721d7289	1342
8	d7211	1265
9	d726	1128
10	d7289d7211	973
11	d722	794
12	d723	740
13	d721d724	737
14	d721d724d7289	534
15	d722d724d7289d7211	480
16	d722d7289	404
17	d721d724d7289d7211	370
18	d723d724	363
19	d727	355
20	d724d726	329
21	d722d724d7289	319
22	d722d7289d7211	294
23	d722d724	250
24	d726d7289	239

25	d721d724d7211	220
26	d721d7289d7211	203
27	d724d727	198
28	d727d7289	187
29	d722d724d7211	182
30	d722d7211	175
31	d724d727d7289	169
32	d721d7211	155
33	d724d726d7289	144
34	d724d726d7289d7211	141
35	d721d722d724d7289d7211	134
36	d724d726d7211	116
37	d723d724d7289	98
38	d723d7289	92
39	d726d7289d7211	89
40	d721d724d726d7289d7211	88
41	d726d7211	88
42	d721d726	80
43	d724d727d7289d7211	68
44	d721d724d727d7289	63
45	d722d724d726d7289d7211	58
46	d721d722d7289	58
47	d721d726d7289	57
48	d721d722d724d7289	54
49	d721d722	53
50	d721d724d726d7289	52
51	d721d727d7289	52
52	d721d722d7289d7211	46

53	d721d724d727d7289d7211	43
54	d723d724d7211	42
55	d721d727	42
56	d721d724d726	42
57	d723d7211	41
58	d721d724d726d7211	40
59	d721d722d724d7211	39
60	d722d726d7289d7211	38
61	d721d726d7289d7211	36
62	d723d724d7289d7211	35
63	d721d722d724	35
64	d724d727d7211	35
65	d721d724d727	35
66	d727d7211	34
67	d727d7289d7211	32
68	d721d726d7211	30
69	d721d722d724d726d7289d7211	25
70	d722d726	23
71	d721d724d726d727d7289d7211	18
72	d721d723	18
73	d723d7289d7211	17
74	d722d724d726d7289	16
75	d722d727	15
76	d721d722d7211	15
77	d726d727	14
78	d722d724d727d7289d7211	14
79	d722d724d727d7289	14
80	d721d724d727d7211	12

81	d721d722d726d7289d7211	12
82	d721d722d724d727d7289d7211	12
83	d722d727d7289	11
84	d722d726d7211	11
85	d721d727d7289d7211	11
86	d721d723d7289	11
87	d723d724d727	10
88	d722d726d7289	10
89	d721d723d724	10
90	d721d726d727d7289	9
91	d721d723d724d7289	9
92	d722d724d726d7211	9
93	d721d722d726d7211	9
94	d721d722d724d726d7289	9
95	d723d726	8
96	d722d727d7289d7211	8
97	d722d724d727	8
98	d722d724d726	8
99	d724d726d727	7
100	d721d722d724d726d7211	7
101	d722d724d727d7211	7
102	d723d727	6
103	d722d723	6
104	d721d726d727d7289d7211	6
105	d721d722d726	6
106	d726d727d7289d7211	5
107	d724d726d727d7211	5
108	d722d727d7211	5

109	d724d726d727d7289	5
110	d721d724d726d727	5
111	d721d722d726d7289	5
112	d726d727d7289	5
113	d722d723d724d7289d7211	5
114	d724d726d727d7289d7211	5
115	d721d726d727	5
116	d721d723d7211	4
117	d722d723d724d7289	4
118	d721d723d724d7211	4
119	d721d723d724d7289d7211	4
120	d721d726d727d7211	4
121	d721d724d726d727d7211	3
122	d721d727d7211	3
123	d721d722d727d7211	3
124	d726d727d7211	3
125	d723d724d727d7211	3
126	d721d722d724d727	3
127	d721d722d727d7289	3
128	d721d722d727d7289d7211	3
129	d723d724d726	3
130	d723d724d727d7289	3
131	d723d727d7289	3
132	d721d723d724d727d7289	3
133	d723d726d7211	2
134	d722d723d724	2
135	d722d723d7289	2
136	d721d722d723d724d7289d7211	2

137	d721d722d727	2
138	d721d722d724d727d7289	2
139	d722d724d726d727d7211	2
140	d721d724d726d727d7289	2
141	d723d724d727d7289d7211	2
142	d722d724d726d727d7289	2
143	d723d727d7211	2
144	d723d724d726d7289	2
145	d722d723d7211	2
146	d722d723d724d727d7289d7211	2
147	d721d722d724d726	2
148	d721d722d726d727	2
149	d721d722d724d727d7211	2
150	d721d722d724d726d727d7211	1
151	d721d722d723d726d7289d7211	1
152	d721d723d7289d7211	1
153	d721d723d724d726d727d7211	1
154	d722d723d7289d7211	1
155	d722d723d726d7289	1
156	d721d723d726d7289	1
157	d721d723d724d726d7289d7211	1
158	d721d722d723	1
159	d723d724d726d7289d7211	1
160	d721d723d726	1
161	d721d722d723d7289d7211	1
162	d722d723d724d726	1
163	d721d723d724d727d7289d7211	1
164	d723d726d7289d7211	1

165	d723d724d726d727d7289d7211	1
166	d721d722d724d726d727d7289d7211	1
167	d721d722d724d726d727	1
168	d721d722d724d726d727d7289	1
169	d721d723d727d7289	1
170	d723d724d726d7211	1
171	d722d723d724d726d727d7289d7211	1
172	d722d726d727	1
173	d721d723d724d727	1
174	d721d723d726d7289d7211	1
175	d722d723d724d7211	1

8.0 APPENDIX C. Features in one SDG-phase Model

Table 20: Features in the SDG-phase Model

CUI	Description	Semantic type	Phase
C0945826	Ambulation	Organism Function	1,2,3
C0002871	Anemia	Disease or Syndrome	1,2,4
C0002962	Angina Pectoris	Sign or Symptom	1,2
C0858277	angina symptom	Sign or Symptom	1,2
C0428977	Bradycardia	Finding	1,2,3,4
C2006086	caffeine use	Finding	1,2
C0018800	Cardiomegaly	Finding	1,2
C0008031	Chest Pain	Sign or Symptom	1,2
C0009171	Cocaine Abuse	Mental or Behavioral Dysfunction	1,2
C0600427	Cocaine Dependence	Mental or Behavioral Dysfunction	1,2
C0010200	Coughing	Sign or Symptom	1,2,3
C0149871	Deep Vein Thrombosis	Disease or Syndrome	1,2
C0012833	Dizziness	Sign or Symptom	1,2
C0013404	Dyspnea	Sign or Symptom	1,2,3
C0013604	Edema	Pathologic Function	1,2,3
C0743393	EDEMA INCREASING	Finding	1,2
C0020473	Hyperlipidemia	Disease or Syndrome	1,2,4
C1737247	Increased abdominal girth	Finding	1,2,3
C0220870	Lightheadedness	Sign or Symptom	1,2

C0085619	Orthopnea	Finding	1,2
C0030193	Pain	Sign or Symptom	1,2
C0853946	Pain worsened	Sign or Symptom	1,2
C0030252	Palpitations	Finding	1,2,3
C0333243	Pitting edema	Sign or Symptom	1,2,3
C0032285	Pneumonia	Disease or Syndrome	1,2,3,4
C1998297	Recent myocardial infarction	Finding	1,2
C0392680	Shortness of Breath	Sign or Symptom	1,2
C0748648	SHORTNESS OF BREATH PROGRESSIVE	Finding	1,2
C0038990	Sweating	Finding	1,2,3
C0039231	Tachycardia	Finding	1,2,3
C0042487	Venous Thrombosis	Pathologic Function	1,2
C0043144	Wheezing	Sign or Symptom	1,2,3

9.0 APPENDIX D. Features in one SDG-no-phase Model

Table 21: Features in SDG-no-phase Model

CUI	Description	Semantic type	Phase
C0001443	Adenosine	Nucleic Acid	4
C0858277	angina symptom	Sign or Symptom	1,2
C0003483	Aorta	Body Part	2
C0003493	Aortic Diseases	Disease or Syndrome	2
C0004057	Aspirin	Organic Chemical	2
C0972395	Automatic Implantable Cardioverter-Defibrillators	Medical Device	2
C2008326	cardiac catheterization mitral valve prolapse	Finding	2
C2023569	cardiac catheterization procedures performed	Diagnostic Procedure	2
C0018800	Cardiomegaly	Finding	1,2
C2024883	cardiovascular surgery result: angina	Finding	2
C0054836	carvedilol	Organic Chemical	2
C0085590	catheter device	Medical Device	5
C0008031	Chest Pain	Sign or Symptom	1,2
C0024117	Chronic Obstructive Airway Disease	Disease or Syndrome	2
C1947999	Coreg Butoxamine HCl	Organic Chemical	2
C0010055	Coronary Artery Bypass Surgery	Therapeutic or Preventive Procedure	2
C1260596	coronary artery graft device	Medical Device	2
C0687568	Coronary artery stent	Medical Device	2

C0699129	Coumadin	Organic Chemical	2
C0180307	Defibrillators	Medical Device	5
C0011849	Diabetes Mellitus	Disease or Syndrome	2
C0011946	Dialysis procedure	Therapeutic or Preventive Procedure	5
C0012265	Digoxin	Organic Chemical	2
C0428224	Digoxin blood measurement	Laboratory Procedure	4
C0012797	Diuresis	Organ or Tissue Function	2
C0013030	Dopamine	Organic Chemical	2
C0013404	Dyspnea	Sign or Symptom	1,2,3
C0013604	Edema	Pathologic Function	1,2,3
C2700378	Ejection fraction (finding)	Finding	4
C0016860	Furosemide	Organic Chemical	2
C0425583	Heart beat	Organ or Tissue Function	2,3
C0018808	Heart murmur	Finding	3
C0018810	heart rate	Clinical Attribute	3
C0019134	heparin	Organic Chemical	2
C0013801	Holter Electrocardiography	Diagnostic Procedure	4
C0182920	Holter Monitors	Medical Device	4
C0021925	Intubation	Therapeutic or Preventive Procedure	2
C0699992	Lasix	Organic Chemical	2
C0181586	Leads (device)	Manufactured Object	5
C0181598	Left ventricular assist device	Medical Device	2
C0225897	Left ventricular structure	Body Part	3
C0220870	Lightheadedness	Sign or Symptom	1,2
C0593906	Lipitor	Organic Chemical	5
C0700776	Lopressor	Organic Chemical	2
C0024554	Male gender	Organism Attribute	2,3
C0026264	Mitral Valve	Body Part	2,3

C0017887	Nitroglycerin	Organic Chemical	2
C0425452	No obstruction of airway	Finding	2,3
C0232202	Normal sinus rhythm	Finding	4
C0810633	Pacemakers	Medical Device	2
C0030193	Pain	Sign or Symptom	1,2
C0030252	Palpitations	Finding	1,2,3
C2936173	Percutaneous Transluminal Coronary Angioplasty	Therapeutic or Preventive Procedure	2
C1253937	Pericardial effusion body substance	Body Substance	2
C0085096	Peripheral Vascular Diseases	Disease or Syndrome	2
C0333243	Pitting edema	Sign or Symptom	1,2,3
C0522776	Placement of stent	Therapeutic or Preventive Procedure	2
C0633084	Plavix	Organic Chemical	2
C0429068	PR depression	Finding	4
C1847014	PULMONARY DISEASE. CHRONIC OBSTRUCTIVE. SEVERE EARLY-ONSET	Disease or Syndrome	2
C0034065	Pulmonary Embolism	Pathologic Function	2
C0748126	PULMONARY EMBOLUS HIGH PROBABILITY	Finding	4
C0020542	Pulmonary Hypertension	Pathologic Function	2
C0034642	Rales	Finding	3
C0034896	Rectum	Body Part	5
C1565489	Renal Insufficiency	Disease or Syndrome	2,4
C0003506	Replacement of aortic valve (procedure)	Therapeutic or Preventive Procedure	2

C0026268	Replacement of mitral valve (procedure)	Therapeutic or Preventive Procedure	2
C1614029	Revatio	Organic Chemical	2
C0520887	ST segment depression (finding)	Finding	4
C0520886	ST segment elevation (finding)	Finding	4
C0038257	Stent. device	Medical Device	2
C0038990	Sweating	Finding	1,2,3
C0080203	Tachyarrhythmia	Finding	3,4
C0039231	Tachycardia	Finding	1,2,3
C0699226	Tridil	Organic Chemical	2

10.0 APPENDIX E. Features in one ML-C4.5 Model

Table 22: Features in the ML-C45 Model

CUI	Description	Semantic type
C0907402	insulin glargine	Amino Acid. Peptide. or Protein. Hormone. Pharmacologic Substance
C0000726	abdomen	Body Location or Region
C1515974	anatomic site	Body Location or Region
C0004600	back	Body Location or Region
C0817096	chest	Body Location or Region
C0225808	right side of heart	Body Location or Region
C1186983	anatomic valve	Body Part. Organ. or Organ Component
C0003483	aorta	Body Part. Organ. or Organ Component
C0007272	carotid arteries	Body Part. Organ. or Organ Component
C0205076	chest wall structure	Body Part. Organ. or Organ Component
C0205042	coronary artery	Body Part. Organ. or Organ Component
C0010268	cranial nerves	Body Part. Organ. or Organ Component
C0013443	ear structure	Body Part. Organ. or Organ Component
C1305418	entire calf of leg (body structure)	Body Part. Organ. or Organ Component

C1269079	entire lower limb	Body Part. Organ. or Organ Component
C0018787	heart	Body Part. Organ. or Organ Component
C0018792	heart Atrium	Body Part. Organ. or Organ Component
C0022742	knee	Body Part. Organ. or Organ Component
C0230347	left upper arm structure	Body Part. Organ. or Organ Component
C0026264	mitral Valve	Body Part. Organ. or Organ Component
C0026639	oral mucous membrane structure	Body Part. Organ. or Organ Component
C1140618	upper extremity	Body Part. Organ. or Organ Component
C0042449	veins	Body Part. Organ. or Organ Component
C0030471	nasal sinus	Body Space or Junction
C0005889	body fluids	Body Substance
C0026727	mucous body substance	Body Substance
C0038984	sweat	Body Substance
C0042036	urine	Body Substance
C0017189	gastrointestinal tract structure	Body System
C0042066	genitourinary system	Body System
C0024235	lymphatic system	Body System
C1123023	skin	Body System
C0018810	heart rate	Clinical Attribute

C0456165	right atrial pressure	Clinical Attribute
C0518766	vital signs	Clinical Attribute
C0004339	auscultation	Diagnostic Procedure
C0011923	diagnostic imaging	Diagnostic Procedure
C0043299	diagnostic radiologic examination	Diagnostic Procedure
C0013516	echocardiography	Diagnostic Procedure
C0015260	exercise stress test	Diagnostic Procedure
C0013801	holter electrocardiography	Diagnostic Procedure
C0034108	oximetry. Pulse	Diagnostic Procedure
C0030247	palpation	Diagnostic Procedure
C0039985	plain chest X-ray	Diagnostic Procedure
C1306645	plain x-ray	Diagnostic Procedure
C0039451	telemetry	Diagnostic Procedure
C0750430	work-up	Diagnostic Procedure
C0004058	allergy to aspirin	Disease or Syndrome
C0002871	anemia	Disease or Syndrome
C0162871	aortic aneurysm. Abdominal	Disease or Syndrome
C0003493	aortic diseases	Disease or Syndrome
C0003864	arthritis	Disease or Syndrome
C0008149	chlamydia Infections	Disease or Syndrome
C0149871	deep vein thrombosis	Disease or Syndrome
C0011849	diabetes mellitus	Disease or Syndrome
C0020473	hyperlipidemia	Disease or Syndrome
C0600260	lung diseases. Obstructive	Disease or Syndrome
C0024796	marfan syndrome	Disease or Syndrome
C0392525	nephrolithiasis	Disease or Syndrome

C0085096	peripheral vascular diseases	Disease or Syndrome
C0032231	pleurisy	Disease or Syndrome
C0032285	pneumonia	Disease or Syndrome
C2937421	prostatic hyperplasia	Disease or Syndrome
C0040034	thrombocytopenia	Disease or Syndrome
C0426663	abdomen soft	Finding
C2712134	actual positive comfort	Finding
C1835620	alert affect	Finding
C0232693	bowel sounds	Finding
C0428977	bradycardia	Finding
C2008326	cardiac catheterization mitral valve prolapse	Finding
C0578395	chest clear	Finding
C0743393	edma increasing	Finding
C1827170	edema of extremity	Finding
C0239340	edema of lower extremity	Finding
C2700378	ejection fraction (finding)	Finding
C0427692	full blood count normal	Finding
C1445096	has strength	Finding
C0018808	heart murmur	Finding
C0578150	hemodynamically stable	Finding
C0857121	hypertensive (finding)	Finding
C0020649	hypotension	Finding
C0237314	irregular heart beat	Finding
C0425687	jugular venous engorgement	Finding
C0455900	moist oral mucosa	Finding
C1265570	morphology within normal limits	Finding

C0262581	no known drug allergy	Finding
C0278005	normal bowel sounds	Finding
C0232202	normal sinus rhythm	Finding
C0426650	obese abdomen	Finding
C0030252	palpitations	Finding
C0460139	pressure (finding)	Finding
C2143305	pupils equal in size, round, reactive to light	Finding
C0034642	rales	Finding
C0455899	red throat	Finding
C0003813	sinus arrhythmia	Finding
C0429029	ST segment	Finding
C0520886	ST segment elevation (finding)	Finding
C0038990	sweating	Finding
C0038999	swelling	Finding
C0576177	swelling of ankle joint (finding)	Finding
C0080203	tachyarrhythmia	Finding
C0039231	tachycardia	Finding
C2203276	wine consumption	Finding
C0587571	cardiology service (procedure)	Health Care Activity
C0278350	ENT examination	Health Care Activity
C0376405	patient non-compliance	Individual Behavior
C0037369	smoking	individual behavior
C0696098	blood thyroid stimulating hormone analysis	Laboratory Procedure

C0009555	complete blood count	Laboratory Procedure
C0201973	creatine kinase measurement	Laboratory Procedure
C0523631	folic acid measurement	Laboratory Procedure
C0525032	international normalized ratio	Laboratory Procedure
C0681827	laboratory studies	Laboratory Procedure
C0023901	liver Function tests	Laboratory Procedure
C0523807	oxygen saturation measurement	Laboratory Procedure
C0202194	potassium measurement	Laboratory Procedure
C0337443	sodium measurement	Laboratory Procedure
C0555903	total protein measurement	Laboratory Procedure
C0919758	vitamin D measurement	Laboratory Procedure
C0023508	white blood cell count procedure	Laboratory Procedure
C0181586	leads (device)	Manufactured Object
C0030163	artificial cardiac pacemaker	Medical Device
C0972395	automatic implantable cardioverter-Defibrillators	Medical Device
C0085590	catheter device	Medical Device
C1260596	coronary artery graft device	Medical Device
C0027524	nebulizers	Medical Device
C0810633	pacemakers	Medical Device
C0038257	stent. device	Medical Device
C0851406	anxiety disorders and symptoms	Mental or Behavioral Dysfunction

C0009171	cocaine abuse	Mental or Behavioral Dysfunction
C0013146	drug abuse	Mental or Behavioral Dysfunction
C0036341	schizophrenia	Mental or Behavioral dysfunction
C0001443	adenosine	Nucleic Acid. Nucleoside. or Nucleotide.Biologically Active Substance.Pharmacologic Substance
C0036658	esthesia	Organ or Tissue Function
C0232804	renal function	Organ or Tissue Function
C0014806	erythromycin	Organic Chemical.Antibiotic
C0019134	heparin	Organic Chemical.Biologically Active Substance.Pharmacologic Substance
C0965130	advair	Organic Chemical.Pharmacologic Substance
C0004057	aspirin	Organic Chemical.Pharmacologic Substance
C0004147	atenolol	Organic Chemical.Pharmacologic Substance

C0004259	atropine	Organic Chemical.Pharmacologic Substance
C0701009	bumex	Organic Chemical.Pharmacologic Substance
C0700940	cardizem	Organic Chemical.Pharmacologic Substance
C0054836	carvedilol	Organic Chemical.Pharmacologic Substance
C0719509	coreg	Organic Chemical.Pharmacologic Substance
C0699129	coumadin	Organic Chemical.Pharmacologic Substance
C0591301	cozaar	Organic Chemical.Pharmacologic Substance
C0012265	digoxin	Organic Chemical.Pharmacologic Substance
C1692318	docusate	Organic Chemical.Pharmacologic Substance

C0699274	ecotrin	Organic Chemical.Pharmacologic Substance
C0015846	fentanyl	Organic Chemical.Pharmacologic Substance
C0016860	furosemide	Organic Chemical.Pharmacologic Substance
C0699992	lasix	Organic Chemical.Pharmacologic Substance
C0593906	lipitor	Organic Chemical.Pharmacologic Substance
C0049506	mirtazapine	Organic Chemical.Pharmacologic Substance
C0026549	morphine	Organic Chemical.Pharmacologic Substance
C0017887	nitroglycerin	Organic Chemical.Pharmacologic Substance
C0699237	nitrol	Organic Chemical.Pharmacologic Substance

C0162712	norvasc	Organic Chemical.Pharmacologic Substance
C0028978	omeprazole	Organic Chemical.Pharmacologic Substance
C0633084	plavix	Organic Chemical.Pharmacologic Substance
C0700777	prilosec	Organic Chemical.Pharmacologic Substance
C0036656	senna Extract	Organic Chemical.Pharmacologic Substance
C0074554	simvastatin	Organic Chemical.Pharmacologic Substance
C0074722	sodium Bicarbonate	Organic Chemical.Pharmacologic Substance
C0037707	sotalol	Organic Chemical.Pharmacologic Substance
C0037982	spironolactone	Organic Chemical.Pharmacologic Substance

C0040610	tramadol	Organic Chemical.Pharmacologic Substance
C0678181	zocor	Organic Chemical.Pharmacologic Substance
C0015780	female	Organism Attribute
C0024554	male gender	Organism Attribute
C0005823	blood Pressure	Organism Function
C1265876	abnormally opaque structure (morphologic abnormality)	Pathologic Function
C1527304	allergic Reaction	Pathologic Function
C0004144	atelectasis	Pathologic Function
C0013604	edema	Pathologic Function
C0019080	hemorrhage	Pathologic Function
C0231274	intolerant of heat	Pathologic Function
C0034063	pulmonary Edema	Pathologic Function
C0034065	pulmonary Embolism	Pathologic Function
C0020542	pulmonary Hypertension	Pathologic Function
C0232483	reflux	Pathologic Function
C0001645	adrenergic beta-Antagonists	Pharmacologic Substance
C0003280	anticoagulants	Pharmacologic Substance
C0304227	prescription Drugs	Pharmacologic Substance
C0024477	magnesium Oxide	Pharmacologic Substance.Inorganic Chemical
C0035203	respiration	Physiologic Function
C0043100	weight	Quantitative Concept

C0858277	angina symptom	Sign or Symptom
C0003862	arthralgia	Sign or Symptom
C0004093	asthenia	Sign or Symptom
C0262384	atypical chest pain	Sign or Symptom
C0423636	cardiac pain	Sign or Symptom
C0235710	chest discomfort	Sign or Symptom
C0008031	chest pain	Sign or Symptom
C2073320	chest pain occurring suddenly as new onset	Sign or Symptom
C0010200	coughing	Sign or Symptom
C0013404	dyspnea	Sign or Symptom
C0231807	dyspnea on exertion	Sign or Symptom
C2073283	factors initiating chest pain	Sign or Symptom
C2029900	fast heart rate (symptom)	Sign or Symptom
C2242996	has tingling sensation	Sign or Symptom
C0018681	headache	Sign or Symptom
C0476280	musculoskeletal chest pain	Sign or Symptom
C0028643	numbness	Sign or Symptom
C0030193	pain	Sign or Symptom
C0023222	pain in lower limb	Sign or Symptom
C0239376	pain of lower extremities	Sign or Symptom
C1960985	pain radiating to left side of chest	Sign or Symptom
C1960989	pain radiating to neck	Sign or Symptom
C0554990	pale - symptom	Sign or Symptom
C0747199	pancreatitis sign	Sign or Symptom
C0333243	pitting edema	Sign or Symptom
C0008033	pleuritic pain	Sign or Symptom

C0476273	respiratory distress	Sign or Symptom
C0178310	signs and symptoms of ill-defined conditions	Sign or Symptom
C0037763	spasm	Sign or Symptom
C0557875	tired	Sign or Symptom
C0235218	warm skin	Sign or Symptom
C0043144	wheezing	Sign or Symptom
C0741847	bypass	Therapeutic or Preventive Procedure
C0007430	catheterization	Therapeutic or Preventive Procedure
C0010055	coronary artery bypass surgery	Therapeutic or Preventive Procedure
C0011946	dialysis procedure	Therapeutic or Preventive Procedure
C0020699	hysterectomy	Therapeutic or Preventive Procedure
C1998570	intubating	Therapeutic or Preventive Procedure
C0746818	irradiation of neck	Therapeutic or Preventive Procedure
C0188379	knee strapping	Therapeutic or Preventive Procedure
C0029216	organ Transplantation	Therapeutic or Preventive Procedure
C0747130	pacemaker placement	Therapeutic or Preventive Procedure
C1304888	pain control	Therapeutic or Preventive Procedure

C0003506	replacement of aortic valve (procedure)	Therapeutic or Preventive Procedure
C1293130	stabilization	Therapeutic or Preventive Procedure
C0040861	triage	Therapeutic or Preventive Procedure
C0026724	mucous Membrane	Tissue
RACE		
AGE		
INSURANCE		
INCOME		

Bibliography

[1] Julie Abimanyi-Ochom, Shalika Bohingamu Mudiyanselage, Max Catchpool, Marnie Firipis, Sithara Wanni Arachchige Dona, and Jennifer J Watts. Strategies to reduce diagnostic errors: a systematic review. *BMC medical informatics and decision making*, 19(1):174, 2019.

[2] Yaser S Abu-Mostafa. Hints. *Neural computation*, 7(4):639–671, 1995.

[3] Klaus-Peter Adlassnig. Fuzzy set theory in medical diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(2):260–265, 1986.

[4] Klaus-Peter Adlassnig, Gernot Kolarz, Werner Scheithauer, Harald Effenberger, and Georg Grabner. Cadiag: Approaches to computer-assisted medical diagnosis. *Computers in biology and medicine*, 15(5):315–335, 1985.

[5] MD Ridwan Al Iqbal, Saiedur Rahman, Syed Irfan Nabil, and Ijaz Ul Amin Chowdhury. Knowledge based decision tree construction with feature importance domain knowledge. In *2012 7th international conference on electrical and computer engineering*, pages 659–662. IEEE, 2012.

[6] Reem Al-Otaibi, Meelis Kull, and Peter Flach. Lacova: A tree-based multi-label classifier using label covariance as splitting criterion. In *2014 13th International Conference on Machine Learning and Applications*, pages 74–79. IEEE, 2014.

[7] V Apgar. A proposal for a new method of evaluation of the newborn. *Classic Papers in Critical Care*, 32(449):97, 1952.

[8] Ahmad Taher Azar and Shereen M El-Metwally. Decision tree classifiers for automated medical diagnosis. *Neural Computing and Applications*, 23(7-8):2387–2403, 2013.

[9] Elias Bareinboim. Causal Reinforcement Learning. `https://crl.causalai.net/`. Last Accessed: 2022-10-21.

[10] VS Bebarta and CB Cairns. Emergency care research-a primer. *Dallas: American College of Emergency Physicians*, 2012.

[11] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, 6(5):679–684, 1957.

[12] Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.

[13] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[14] Chunhui Cai, Gregory F Cooper, Kevin N Lu, Xiaojun Ma, Shuping Xu, Zhenlong Zhao, Xueer Chen, Yifan Xue, Adrian V Lee, Nathan Clark, et al. Systematic discovery of the functional impact of somatic genome alterations in individual tumors through tumor-specific causal inference. *PLoS computational biology*, 15(7):e1007088, 2019.

[15] Li-Juan Cao and Francis Eng Hock Tay. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on neural networks*, 14(6):1506–1518, 2003.

[16] Frank Castro, Leonard P Caccamo, Kimbroe J Carter, Barbara A Erickson, William Johnson, Edward Kessler, Nathan P Ritchey, and Claudio A Ruiz. Sequential test selection in the analysis of abdominal pain. *Medical Decision Making*, 16(2):178–183, 1996.

[17] Amanda Clare and Ross D King. Knowledge discovery in multi-label phenotype data. In *European conference on principles of data mining and knowledge discovery*, pages 42–53. Springer, 2001.

[18] Breda Corish. Medical knowledge doubles every few months; how can clinicians keep up? *Available online: https://www.elsevier.com/connect/medical-knowledge-doubles-every-few-months-how-can-clinicians-keep-up (accessed on 28 November 2022)*, 2018.

[19] Pat Croskerry. The importance of cognitive errors in diagnosis and strategies to minimize them. *Academic medicine*, 78(8):775–780, 2003.

[20] Francesco De Comité, Rémi Gilleron, and Marc Tommasi. Learning multi-label alternating decision trees from texts and data. In *International workshop on machine learning and data mining in pattern recognition*, pages 35–49. Springer, 2003.

[21] Saif J Fakih and Tapas K Das. Lead: A methodology for learning efficient approaches to medical diagnosis. *IEEE Transactions on Information Technology in Biomedicine*, 10(2):220–228, 2006.

[22] Edward A Feigenbaum and Pamela McCorduck. *The fifth generation*. Pan Books London, 1984.

[23] Agency for Healthcare Research and Quality. Clinical classifications software (ccs) for icd-9-cm. 2015.

[24] Alberto Freitas and Altamiro Costa-Pereira. Learning cost-sensitive decision trees to support medical diagnosis. In *Complex Data Warehousing and Knowledge Discovery for Advanced Retrieval Development: Innovative Methods and Applications*, pages 287–307. IGI Global, 2010.

[25] Yoav Freund and Llew Mason. The alternating decision tree learning algorithm. In *icml*, volume 99, pages 124–133. Citeseer, 1999.

[26] Carol Friedman, Philip O Alderson, John HM Austin, James J Cimino, and Stephen B Johnson. A general natural-language text processor for clinical radiology. *Journal of the American Medical Informatics Association*, 1(2):161–174, 1994.

[27] James F Fries. Experience counting in sequential computer diagnosis. *Archives of internal medicine*, 126(4):647–651, 1970.

[28] Teresa Gonçalves and Paulo Quaresma. A preliminary approach to the multilabel classification problem of portuguese juridical documents. In *Portuguese Conference on Artificial Intelligence*, pages 435–444. Springer, 2003.

[29] G Anthony Gorry. Strategies for computer-aided diagnosis. *Mathematical Biosciences*, 2(3-4):293–318, 1968.

[30] G Anthony Gorry and G Octo Barnett. Experience with a model of sequential diagnosis. *Computers and Biomedical Research*, 1(5):490–507, 1968.

[31] G Anthony Gorry and G Octo Barnett. Sequential diagnosis by computer. *Jama*, 205(12):849–854, 1968.

[32] G Anthony Gorry, Jerome P Kassirer, Alvin Essig, and William B Schwartz. Decision analysis as the basis for computer-aided management of acute renal failure. *The American journal of medicine*, 55(4):473–484, 1973.

[33] George Anthony Gorry. A system for computer-aided diagnosis. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC, 1967.

[34] Omer Gottesman, Fredrik Johansson, Matthieu Komorowski, Aldo Faisal, David Sontag, Finale Doshi-Velez, and Leo Anthony Celi. Guidelines for reinforcement learning in healthcare. *Nature medicine*, 25(1):16–18, 2019.

[35] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[36] Joshua James Hatherley. Limits of trust in medical AI. *Journal of medical ethics*, 46(7):478–481, 2020.

[37] David E Heckerman, Eric J Horvitz, and Bharat N Nathwani. *Update on the Pathfinder Project*. Knowledge Systems Laboratory, Stanford University, 1989.

[38] David E Heckerman, Eric J Horvitz, and Bharat N Nathwani. Toward normative expert systems: The pathfinder project. Technical report, Technical Report KSL-90-08 (Section on Medical Informatics, Stanford ..., 1990.

[39] Esther Hing and Farida A Bhuiya. *Wait time for treatment in hospital emergency departments, 2009*. Number 102. US Department of Health and Human Services, Centers for Disease Control and ..., 2012.

[40] Eric Horvitz and Michael Shwe. Handsfree decision support: Toward a non-invasive human-computer interface. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, page 955. American Medical Informatics Association, 1995.

[41] St John. Causal Reinforcement Learning: A Primer. `https://stjohngrimbly.com/causal-reinforcement-learning/`. Last Accessed: 2022-10-21.

[42] Arthur L Kellermann. Consilience. *Annals of emergency medicine*, 56(5):568–570, 2010.

[43] DC Knockaert, Frank Buntinx, N Stoens, Rudi Bruyninckx, and Herman Delooz. Chest pain in the emergency department: the broad spectrum of causes. *European Journal of Emergency Medicine*, 9(1):25–30, 2002.

[44] Rachel Kohn, Michael O Harhay, Elizabeth Cooney, Dylan S Small, and Scott D Halpern. Do windows or natural views affect outcomes or costs among patients in icus? *Critical care medicine*, 41(7):1645–1655, 2013.

[45] Matthieu Komorowski, Leo A Celi, Omar Badawi, Anthony C Gordon, and A Aldo Faisal. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature medicine*, 24(11):1716–1720, 2018.

[46] Casimir A Kulikowski. Pattern recognition approach to medical diagnosis. *IEEE Transactions on Systems Science and Cybernetics*, 6(3):173–178, 1970.

[47] Helge Langseth and Thomas D Nielsen. Fusion of domain knowledge with data for structural learning in object oriented domains. *Journal of Machine Learning Research*, 4(Jul):339–368, 2003.

[48] Lucian L Leape, Troyen A Brennan, Nan Laird, Ann G Lawthers, A Russell Localio, Benjamin A Barnes, Liesi Hebert, Joseph P Newhouse, Paul C Weiler, and Howard Hiatt. The nature of adverse events in hospitalized patients: results of the harvard medical practice study ii. *New England journal of medicine*, 324(6):377–384, 1991.

[49] RS Ledley and LB Lusted. Reasoning foundation of medical diagnosis: symbolic logic, probability, and value theory aid our understanding of how physicians reason. *Science. v130*, pages 9–21, 1959.

[50] Kangenbei Liao, Qianlong Liu, Zhongyu Wei, Baolin Peng, Qin Chen, Weijian Sun, and Xuanjing Huang. Task-oriented dialogue system for automatic disease diagnosis via hierarchical reinforcement learning. *arXiv preprint arXiv:2004.14254*, 2020.

[51] Charles X Ling, Victor S Sheng, and Qiang Yang. Test strategies for cost-sensitive decision trees. *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1055–1067, 2006.

[52] Gwilym S Lodwick, Cosmo L Haun, Walton E Smith, Roy F Keller, and Eddie D Robertson. Computer diagnosis of primary bone tumors: A preliminary report. *Radiology*, 80(2):273–275, 1963.

[53] Albert Paul Malvino and Jerald A Brown. *Digital computer electronics*. Gregg Division, McGraw-Hill, 1977.

[54] David McSherry. Sequential diagnosis in the independence bayesian framework. *Soft Computing*, 8(2):118–125, 2003.

[55] Kristen E Miller, Muge Capan, Pan Wu, Eric V Jackson, and Ryan C Arnold Jr. Operationalizing sepsis alert design and clinical decision support: developing enhanced visual display models. In *Proceedings of the International Symposium on Human Factors and Ergonomics in Health Care*, volume 4, pages 103–109. SAGE Publications Sage India: New Delhi, India, 2015.

[56] Randolph A Miller and Antoine Geissbuhler. Clinical diagnostic decision support systems—an overview. *Clinical decision support systems*, pages 3–34, 1999.

[57] Randolph A Miller and Fred E Masarie Jr. Quick medical reference (qmr): A microcomputer-based diagnostic decision-support system for general internal medicine. In *Proceedings. Symposium on Computer Applications in Medical Care*, pages 986–988, 1990.

[58] Randolph A Miller, Harry E Pople Jr, and Jack D Myers. Internist-i, an experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, 307(8):468–476, 1982.

[59] Violeta Mirchevska, Mitja Luštrek, and Matjaž Gams. Combining domain knowledge and machine learning for robust fall detection. *Expert Systems*, 31(2):163–175, 2014.

[60] V Mnih, K Kavukcuoglu, D Silver, A Graves, I Antonoglou, and D Wierstra. Riedmiller.: Playing atari with deep reinforcement learning. *Computer Science-Learning*, 2013.

[61] Kristy Gonzalez Morganti, Sebastian Bauhoff, Janice C Blanchard, Mahshid Abir, Neema Iyer, Alexandria Smith, Joseph V Vesely, Edward N Okeke, and Arthur L Kellermann. The evolving role of emergency departments in the united states. *Rand health quarterly*, 3(2), 2013.

[62] Nikhil Muralidhar, Mohammad Raihanul Islam, Manish Marwah, Anuj Karpatne, and Naren Ramakrishnan. Incorporating prior domain knowledge into deep neural networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 36–45. IEEE, 2018.

[63] Silvano Mussi. Diagnostic expert systems: a method for engineering knowledge used in sequential diagnosis. *Expert systems*, 17(4):199–211, 2000.

[64] Mohammed Nabih-Ali, EL-Sayed A El-Dahshan, and Ashraf S Yahia. A review of intelligent systems for heart sound signal analysis. *Journal of medical engineering & technology*, 41(7):553–563, 2017.

[65] Partha Niyogi, Federico Girosi, and Tomaso Poggio. Incorporating prior information in machine learning by creating virtual examples. *Proceedings of the IEEE*, 86(11):2196–2209, 1998.

[66] Agnieszka Onisko, Marek J Druzdzel, and Hanna Wasyluk. A probabilistic causal model for diagnosis of liver disorders. In *Proceedings of the Seventh International Symposium on Intelligent Information Systems (IIS—98)*, page 379, 1998.

[67] Agnieszka Oniśko, Marek J Druzdzel, and Hanna Wasyluk. Extension of the hepar ii model to multiple-disorder diagnosis. In *Intelligent Information Systems*, pages 303–313. Springer, 2000.

[68] Agnieszka Onisko, Marek J Druzdzel, Hanna Wasyluk, et al. A bayesian network model for diagnosis of liver disorders. In *Proceedings of the Eleventh Conference on Biocybernetics and Biomedical Engineering*, volume 2, pages 842–846. Citeseer, 1999.

[69] John E Overall and Clyde M Williams. Conditional probability program for diagnosis of thyroid function. *JAMA*, 183(5):307–313, 1963.

[70] Babita Pandey and RB Mishra. Knowledge and intelligent computing system in medicine. *Computers in biology and medicine*, 39(3):215–230, 2009.

[71] RC Parker and RA Miller. Creation of realistic appearing simulated patient cases using the internist-1/qmr knowledge base and interrelationship properties of manifestations. *Methods of information in medicine*, 28(04):346–351, 1989.

[72] Michael J Pazzani, Clifford A Brunk, and Glenn Silverstein. A knowledge-intensive approach to learning relational concepts. In *Machine Learning Proceedings 1991*, pages 432–436. Elsevier, 1991.

[73] Amirhossein Peyvandi, Babak Majidi, Soodeh Peyvandi, and Jagdish Patra. Computer-aided-diagnosis as a service on decentralized medical cloud for efficient and

rapid emergency response intelligence. *New Generation Computing*, 39(3):677–700, 2021.

[74] Stephen R Pitts, Emily R Carrier, Eugene C Rich, and Arthur L Kellermann. Where americans get acute care: increasingly, it's not at their doctor's office. *Health affairs*, 29(9):1620–1629, 2010.

[75] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

[76] Harry E Pople, Jack D Myers, and Randolph A Miller. Dialog: A model of diagnostic logic for internal medicine. In *IJCAI*, volume 4, pages 848–855. Citeseer, 1975.

[77] Robert S Porter. *The Merck manual of diagnosis and therapy*. Merck Sharp & Dohme Corp., 2018.

[78] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[79] J Ross Quinlan. *C4. 5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[80] Murali Ravuri, Anitha Kannan, Geoffrey J Tso, and Xavier Amatriain. Learning from the experts: From expert systems to machine-learned diagnosis models. *arXiv preprint arXiv:1804.08033*, 2018.

[81] Alan L Rector and Eugene Ackerman. Rules for sequential diagnosis. *Computers and Biomedical Research*, 8(2):143–155, 1975.

[82] Lior Rokach, Alon Schclar, and Ehud Itach. Ensemble methods for multi-label classification. *Expert Systems with Applications*, 41(16):7507–7523, 2014.

[83] Robert A Rosati, J Frederick McNeer, C Frank Starmer, Brant S Mittler, James J Morris, and Andrew G Wallace. A new information system for medical practice. *Archives of Internal Medicine*, 135(8):1017–1024, 1975.

[84] Stuart J Russell. Prior knowledge and autonomous learning. *Robotics and Autonomous Systems*, 8(1-2):145–159, 1991.

[85] Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. Mayo clinical text analysis

and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513, 2010.

[86] Bernhard Schölkopf, Chris Burges, and Vladimir Vapnik. Incorporating invariances in support vector learning machines. In *International Conference on Artificial Neural Networks*, pages 47–52. Springer, 1996.

[87] William B Schwartz. Medicine and the computer: the promise and problems of change. In *Use and Impact of Computers in Clinical Medicine*, pages 321–335. Springer, 1970.

[88] Jude W Shavlik and Geoffrey G Towell. An approach to combining explanation-based and neural learning algorithms. In *Applications Of Learning And Planning Methods*, pages 71–98. World Scientific, 1991.

[89] Chuan Shi, Xiangnan Kong, S Yu Philip, and Bai Wang. Multi-label ensemble learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 223–239. Springer, 2011.

[90] Edward H Shortliffe and Bruce G Buchanan. A model of inexact reasoning in medicine. *Mathematical biosciences*, 23(3-4):351–379, 1975.

[91] Michael A Shwe, Blackford Middleton, David E Heckerman, Max Henrion, Eric J Horvitz, Harold P Lehmann, and Gregory F Cooper. Probabilistic diagnosis using a reformulation of the internist-1/qmr knowledge base. *Methods of information in Medicine*, 30(04):241–255, 1991.

[92] Hardeep Singh, Ashley ND Meyer, and Eric J Thomas. The frequency of diagnostic errors in outpatient care: estimations from three large observational studies involving us adult populations. *BMJ Qual Saf*, 23(9):727–731, 2014.

[93] Aristeidis Sotiras, Christos Davatzikos, and Nikos Paragios. Deformable medical image registration: A survey. *IEEE transactions on medical imaging*, 32(7):1153, 2013.

[94] Kai-Fu Tang, Hao-Cheng Kao, Chun-Nan Chou, and Edward Y Chang. Inquire and diagnose: Neural symptom checking ensemble using deep reinforcement learning. In *NIPS Workshop on Deep Reinforcement Learning*, 2016.

[95] TR Taylor, J Aitchison, and Edward M McGirr. Doctors as decision-makers: a computer-assisted study of diagnosis as a cognitive skill. *Br Med J*, 3(5765):35–40, 1971.

[96] Ali S Saber Tehrani, HeeWon Lee, Simon C Mathews, Andrew Shore, Martin A Makary, Peter J Pronovost, and David E Newman-Toker. 25-year summary of us malpractice claims for diagnostic errors 1986–2010: an analysis from the national practitioner data bank. *BMJ Qual Saf*, 22(8):672–680, 2013.

[97] Abhisek Tiwari, Sriparna Saha, and Pushpak Bhattacharyya. A knowledge infused context driven dialogue agent for disease diagnosis using hierarchical reinforcement learning. *Knowledge-Based Systems*, 242:108292, 2022.

[98] Alan F Toronto, L George Veasy, and Homer R Warner. Evaluation of a computer program for diagnosis of congenital heart disease. *Progress in cardiovascular diseases*, 5(4):362–377, 1963.

[99] TuanNam Tran, Ryutaro Ichise, and Masayuki Numao. Mining hepatitis data set using information gathered from biomedical literature. In *International Workshop on Active Mining (AM-2002)*. Citeseer, 2002.

[100] Edward Tsang, Paul Yung, and Jin Li. Eddie-automation, a decision support tool for financial forecasting. *Decision Support Systems*, 37(4):559–565, 2004.

[101] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2009.

[102] Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *European conference on machine learning*, pages 406–417. Springer, 2007.

[103] João Vieira and Cláudia Antunes. Decision tree learner in the presence of domain knowledge. In *Chinese Semantic Web and Web Science Conference*, pages 42–55. Springer, 2014.

[104] Shyam Visweswaran, Andrew J King, and Gregory F Cooper. Integration of AI for clinical decision support. In *Intelligent Systems in Medicine and Health*, pages 285–308. Springer, 2022.

[105] Khuong Vo, Dang Pham, Mao Nguyen, Trung Mai, and Tho Quan. Combination of domain knowledge and deep learning for sentiment analysis. In *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, pages 162–173. Springer, 2017.

[106] Homer R Warner, Alan F Toronto, and L George Veasy. Experience with baye's theorem for computer diagnosis of congenital heart disease. *Annals of the New York Academy of Sciences*, 115(2):558–567, 1964.

[107] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

[108] CJCH Watkins and P Dayan. Technical note q-learning machine learning. 1992, 8: 279-292. doi: 10.1023. *A: 1022676722315*.

[109] Zhongyu Wei, Qianlong Liu, Baolin Peng, Huaixiao Tou, Ting Chen, Xuanjing Huang, Kam-Fai Wong, and Xiangying Dai. Medical Chatbot Python Implementation. `https://github.com/LiuQL2/MedicalChatbot`. Last Accessed: 2022-09-29.

[110] Zhongyu Wei, Qianlong Liu, Baolin Peng, Huaixiao Tou, Ting Chen, Xuanjing Huang, Kam-Fai Wong, and Xiangying Dai. Task-oriented dialogue system for automatic diagnosis. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 201–207, 2018.

[111] Shelley M White. Merck manuals information online. *Journal of Consumer Health on the Internet*, 13(1):66–76, 2009.

[112] Qingyao Wu, Yunming Ye, Haijun Zhang, Tommy WS Chow, and Shen-Shyang Ho. Ml-tree: A tree-structure-based approach to multilabel learning. *IEEE transactions on neural networks and learning systems*, 26(3):430–443, 2014.

[113] Xiaoyun Wu and Rohini Srihari. Incorporating prior knowledge with weighted margin support vector machines. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 326–333, 2004.

[114] Ye Ye, Michael M Wagner, Gregory F Cooper, Jeffrey P Ferraro, Howard Su, Per H Gesteland, Peter J Haug, Nicholas E Millett, John M Aronis, Andrew J Nowalk, et al. A study of the transferability of influenza case detection systems between two large healthcare systems. *PloS one*, 12(4):e0174970, 2017.

[115] Suk-Chung Yoon, Lawrence J Henschen, EK Park, and Sam Makki. Using domain knowledge in knowledge discovery. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 243–250, 1999.

[116] Chao Yu, Jiming Liu, and Shamim Nemati. Reinforcement learning in healthcare: A survey. *arXiv preprint arXiv:1908.08796*, 2019.

[117] Laura Zwaan and Hardeep Singh. The challenges in defining and measuring diagnostic error. *Diagnosis*, 2(2):97–103, 2015.

[118] Vladimir K Zworykin. Electronic techniques in medicine: The eighth edsel b. ford lecture, 1959. *Henry Ford Hospital Medical Journal*, 8(1):1–18, 1960.