TOWARDS MACHINE LEARNING DRIVEN MEDICAL DISCOVERIES USING HIGH-THROUGHPUT COMPUTING

by

Ross S. Kleiman

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2019

Date of final oral examination: 04/15/2019

The dissertation is approved by the following members of the Final Oral Committee:

C. David Page, Professor, Biostatistics and Medical Informatics, Advisor
Mark Craven, Professor, Biostatistics and Medical Informatics, Thesis Reader
Peggy Peissig, Center Director, BIRC, Marshfield Clinic, Thesis Reader
Jignesh Patel, Professor, Computer Sciences, Committee Member
AnHai Doan, Professor, Computer Sciences, Committee Member

To my wife, Molly Carroll, who always encourages me to 'do good science'.

ACKNOWLEDGMENTS

This thesis, like any other achievement, would have been impossible without the community of mentors, teachers, friends, and family whose support was vital to my success. First, I would like to thank my advisor, David Page, whose guidance throughout my Ph.D. has been critical to my growth as a researcher and as a person. David's door was always open when I needed to think through a tough problem or seek advice. While he always gave excellent input he also respected my autonomy as a student and researcher to make my own decisions. I am grateful to the current and past members of the Page lab group who provided frequent research insights throughout my graduate studies, particularly Aubrey Barnard, Charles Kuang, Finn Kuusisto, Jeremy Weiss, and Wei Zhang. I would also like to thank my committee members, Mark Craven, Peggy Peissig, AnHai Doan, and Jignesh Patel, whose guidance through my Ph.D. has been critical to pursuing valuable research questions. I would like to express additional gratitude to Peggy Peissig who served as a frequent research collaborator and whose guidance greatly benefited my understanding of applying machine learning methods to healthcare data. I would also like to thank Mark Craven for his service as the director of the "Computation and Informatics in Biology and Medicine" training program of which I was a trainee for three years. Finally, I would like to thank Anthony Gitter, who served as an informal mentor throughout graduate school and whose advice I sought when making many important decisions.

Throughout graduate school, I was fortunate to make many friends whose companionship and conversation provided an invaluable foil to my academic studies. Thank you to Matt Bernstein, Chris Magnano, Blake Mason, Ronak Mehta, and Michael O'Neil for the many hours we spent that ranged from relaxation to intense debate. Additionally, I am very thankful for those long-distance friendships that provided support throughout graduate school. Thank you to Josh David, John Falzon, Jon Holt, Amit Jani, Alex Kopp, and Mike Rediker. Each of these friendships both ameliorated those inevitable arduous moments and elevated those moments worth celebrating.

I would finally like to thank my family for their unending support over these last several years. To my parents, Andy and Marcy, who have always embraced and encouraged my passions, I could not have succeeded without you. To my sister, Lindsay, your support through difficult times made them easier to bear. And to my extended family of aunts, uncles, cousins, and my Nana, thank you for your unwavering confidence in me. Finally, and most importantly, to my wife, Molly Carroll, this journey would have lost so much of its meaning and significance without your companionship throughout. You are always the first person I want to share any big news with, whether it is to seek support in challenging times or to celebrate exciting events. You have made the 'good science' worth doing.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The standard multiple outcome prediction task is solved either through multi-label or multi-class machine learning algorithms presented with a single dataset. However, applications exist in which appropriate subsampling of the original dataset, depending on the label, is critical to producing high-quality predictions. We present a solution to this version of the multiple outcome prediction task that we call *high-throughput machine learning*, wherein for each predictive outcome, an appropriate training dataset is first subsampled from the original training data and then a model is learned.

This dissertation considers the application of high-throughput machine learning to the task of predicting all diagnostic outcomes from an electronic health record (EHR) dataset. For each diagnosis, we identify appropriate case and control patients and then train a machine learning classifier to predict that particular diagnosis. We additionally introduce the *dynamic definition refinement* algorithm to automatically identify appropriate pre-requisite control diagnoses on a per-outcome basis. For example, in the task of predicting diabetic retinopathy, we can automatically learn that diabetes should be the control diagnosis.

We consider several applications of high-throughput machine learning and address their associated challenges. The primary contribution of this work provides a performance benchmark for predicting all diagnoses from 1-month to 20-years in advance. We consider cases where data is Incorporated from multiple healthcare systems and provide a cryptographic approach for preserving privacy in this environment. Additionally, we combine a high-throughput inspection of feature importances with a literature-derived knowledge base to identify novel diagnostic uses of laboratory tests. Finally, we consider the multi-class prediction setting and explore a new performance measure for the evaluation of such classifiers. Ultimately, this dissertation aims to show that high-throughput machine learning provides meaningful gains beyond the standard one-off model construction approach.

1 INTRODUCTION

Around the globe, individuals and governments are keenly interested in reducing the costs of healthcare while improving its outcomes. A key component for making such an improvement is for both patients and clinicians to accurately assess patient health risks. Concurrently, the use of electronic health records (EHRs) for electronically capturing patient healthcare encounters has risen dramatically (Hsiao et al., 2014). Consequently, disease prediction tools that can leverage EHR data to assess patient risk have garnered significant attention. Machine learning algorithms have become increasingly prevalent components of such disease prediction tools.

Many of the existing applications of machine learning to healthcare have focused on individual tasks such as predicting cardiovascular disease risk (Dawber et al., 1951) or the appropriate warfarin dosage for a patient (IWPC, 2009). In recent years, healthcare systems have been incorporating these models into their clinical workflows. For example, the use of a machine learning algorithm to predict severe sepsis has been shown to improve patient outcomes (Shimabukuro et al., 2017). However, each of these models must be carefully tuned and evaluated, particularly if they are to be translated to a healthcare system. This approach makes it extremely challenging to construct a broad scale view of patient outcomes, as it would require carefully building thousands of models for each of the possible diagnoses. Yet, there are benefits to constructing such a predictive landscape, as it could be used to identify patterns amongst diseases, or critical health states such as obesity/diabetes that greatly increase risk across many diseases, or assist in the discovery of new medical knowledge by inspection of the learned models.

For construction of such a predictive landscape, we may wish to employ the machine learning approach of multi-label classification which is applicable when there are multiple outcomes we wish to predict and a particular instance, in this case a patient, could have one or more of them. One way of performing multi-label classification is to use a model like a neural network that is capable of predicting more than one label at once. While there are a variety of approaches to performing multi-label classification, they all rely on providing the learning algorithm(s) a

single set of shared training data. This can be problematic for a healthcare setting, as the inclusion of some patients in the training of a model may reduce the generalizability and/or interpretibility of a model. Therefore, we require a new methodology capable of ensuring that each outcome is appropriately modeled without requiring careful hand-tuning of each model. In this dissertation, we build upon one multi-label machine learning approach, where each task has its own model. Our extension includes a selection phase for each prediction task prior to model learning. This selection phase subsamples a shared larger dataset by identifying appropriate learning examples for the learning task.

In this dissertation, we present the concept of *high-throughput machine learning* which is a specialized version of multi-label classification where each task has its own model. In high-throughput machine learning, we construct a pipeline, that with minimal manual intervention, can both 1) select the appropriate training examples and control criteria where appropriate, and 2) train a model to predict that task using the selected training data. This differs from a one-off multi-label classification where there is no selection of training examples. Such a pipeline can be readily applied to the task of predicting a large set of health outcomes where each event needs specialized controlling criteria. For example, while we may always wish to control for age and sex when predicting diagnoses, we may only wish to control for pregnancy stage when predicting a pregnancy complication. This dissertation additionally contributes the approach of *dynamic definition refinement*, a methodology for automatically learning potential appropriate control criteria on a per-prediction-task basis. For example, dynamic definition refinement can learn that when predicting a diabetic retinopathy, a potential complication of diabetes, both positive and negative examples should already be diabetic patients.

In this dissertation, we apply high-throughput machine learning to the task of predicting all diagnoses in a large healthcare system with more than 1.5 million patients' electronic health records. We build several predictive models per diagnosis ranging from predicting 1-month to 20-years in advance. We explore trends in these models across major chapters of the diagnostic hierarchy as well as temporal patterns. We then consider three tasks where high-throughout machine learning

can be leveraged or extended. First, we present a high-throughput approach for learning new medical knowledge from models by combining their feature importances with a text-mining-extracted knowledge base from PubMed literature. Next, we explore the use of machine learning approaches learning from multiple hospital sites and present an approach to ameliorate the privacy concerns inherent to sharing healthcare data. Finally, we present a novel approach for evaluating multi-class classification models motivated by the prediction of health events that could be one of multiple outcomes. Ultimately, this dissertation is concerned with the application of machine learning algorithms to healthcare data at large scales, be they quantity of data, tasks, and/or institutions.

1.1 Thesis Statement

Using the high-profile setting of medicine and electronic health records, this dissertation supports the following thesis:

> *High-throughput machine learning yields insights not gleaned from standard one-off or multi-label modeling.*

1.2 Document Organization

My dissertation is organized as follows.

Chapter 1 presents the overarching opportunity for applying high-throughput machine learning to medical data.

Chapter 2 presents background on machine learning and the multiple areas of healthcare we apply it to.

Chapter 3 shows hows machine learning can be used to both predict and better understand a single disease, Calciphylaxis.

Chapter 4 introduces the concept of high-throughput machine learning and its application in the prediction of thousands of diagnoses.

Chapter 5 discusses the need for privacy preserving machine learning algorithms when incorporating data from multiple hospitals.

Chapter 6 presents a use of high-throughput machine learning to identify novel diagnostic uses of laboratory tests.

Chapter 7 presents a new measure for evaluating multi-class machine learning models motivated by inherently multi-class medical prediction tasks.

Chapter 8 provides a summary of the dissertation and its key contributions.

Chapter 9 provides supplementary tables and proofs for this dissertation.

2 BACKGROUND

This dissertation presents research at the nexus of machine learning and healthcare. As such, this chapter presents the necessary background and context to understand the significance of the contributions. In Section 2.1 we present background on the use of electronic health records and the type of data they store. In Section 2.2 we present high-level background on the field of machine learning and algorithms that are heavily used in this dissertation. Section 2.3 provides a survey of some of the notable applications of machine learning to healthcare data. In Section 2.4 we discuss learning multiple outcomes using machine learning and examples of healthcare tasks that would require such an approach. Finally, Section 2.5 explores the privacy challenges inherent in using healthcare data from both individual healthcare institutions and when combining data from multiple institutions.

2.1 Electronic Health Record Data

The use of EHRs to digitize patient data has quickly become common practice in healthcare systems in the United States (Hsiao et al., 2014). This exponential growth has been fueled both by advances in computational power and EHR software as well as government incentives and regulations through the Health Information Technology for Economic and Clinical Health (HITECH) Act of 2009. Now, a typical U.S. healthcare system stores and interacts with patient data almost entirely in a digital fashion. This change has provided unprecedented opportunities for secondary use of EHR data, including applying machine learning algorithms to predict disease diagnosis and outcome.

Healthcare systems can vary greatly in how their EHR systems are implemented. Some use a single vendor, such as Epic or Cerner, for all of their departments and systems. Others employ a "best-of-breed" approach choosing the best product on a per-department/system basis. A "best-of-breed" approach can result in requiring careful integration of systems. Because of the variance in implementation

Diagnoses			
MHN	Date	Diagnosis	
1357	6/3/97	Type II Diabetes	
1357	6/3/97	High Blood Pressure	
1357	8/15/97	Hypercholesterolemia	
Laboratory			
MHN	Date	Test	Value
1357	6/3/97	Blood Glucose	142
1357	8/15/97	Total Cholesterol	297
Vitals			
MHN	Date	Test	Value
1357	6/3/97	Systolic BP	152
1357	6/3/97	Diastolic BP	103

Table 2.1: Synthetic data illustrating the tabular nature of EHR data for a toy patient with medical history number (MHN) 1357.

by healthcare systems, this background will focus solely on how the Marshfield Clinic (Marshfield, WI) has structured their EHR, as that is most relevant to the work presented in this document. The Marshfield Clinic utilizes (as of this writing) a single system, "Cattails", to record their EHR data. Cattails stores patient data as a series of tables as illustrated by the synthetic EHR patient data in Table 2.1. While Table 2.1 shows only diagnoses, labs, and vitals, EHRs also store procedures, medications, vitals, notes, images, etc.

EHR data often cannot be taken at face value and may contain errors that need to be corrected before any generalizations can be made from the data. One of the most common errors present is the entry of codes for "rule-out diagnoses". Consider the case of a patient entering the emergency room complaining of a severe headache: the physician may be concerned that this patient has a head injury requiring immediate treatment and will need to perform a computed tomography (CT)-scan of the patient's head and neck to confirm or rule out this suspicion. However, the patient's insurance company may require a diagnosis prior to allowing payment for the CT-scan resulting in a diagnosis of concussion on the patient's record. Should the scan come back negative for concussion it may still be the case

that the diagnosis of concussion remains on the patient's record as it was needed for billing purposes. Here, the diagnosis of concussion was a rule-out diagnosis; the diagnosis represented a necessary step in the physician's workup of the patient, but not the reality of the patient's health. Rule-out diagnoses are a serious issue when labeling patient data and thus they must be addressed via *electronic phenotyping* of EHR data. Electronic phenotyping is the task of identifying the patterns amongst a set of variables that indicate a particular phenotype (for example a diagnosis). While electronic phenotyping is useful for handling rule-out diagnoses, it has many additional and important uses. A simples rule for phenotyping is the "rule-of-2" stating that any diagnosis on a record that appears twice or more (on two separate days) is likely to be legitimate. There are many more advanced forms of electronic phenotyping utilizing rule learning to label patient records (Peissig et al., 2014; Pathak et al., 2013) including by committee (eMEREGE; Overhage et al., 2012).

2.2 Machine Learning

Machine learning, an area of computer science that grew out of artificial intelligence, is broadly focused on a class of algorithms capable of 'learning' from data (Mitchell, 1997). While algorithms that work with various and exotic types of data do exist, the vast majority of algorithms expect data that is in a matrix format. This matrix is said to be composed of samples (rows) and features (columns). A sample is an individual observation and each observation has associated features. For example, in a data set of hotel reservations a sample may be a particular reservation and features may include the customer name, room, dates of stay, and rental rate.

Machine learning algorithms are roughly divided into 'unsupervised' and 'supervised' algorithms based on the task of interest and the type of data available . Unsupervised machine learning algorithms are generally focused on learning patterns amongst the various samples in the data based upon their features. Clustering is a common unsupervised machine learning task wherein we wish to partition our original data set into subsets of samples. Ideally, the samples in a particular subset are more similar to one another than they are to the samples in other subsets. In an

unsupervised task, our data set, $D = \{\mathbf{x}^{(i)}\}_{i=1}^{n}$, has n samples, where a particular sample, $\mathbf{x}^{(i)}$, is a vector with m features. In supervised learning, there is a target variable, y, that we are interested in predicting by using previous data that we have collected. In supervised learning, our data set, $D = \{\langle \mathbf{x}^{(i)}, y^{(i)} \rangle\}_{i=1}^{n}$, contains pairs of samples and labels, with each label $y^{(i)}$ serving as the target variable for its associated sample, $\mathbf{x}^{(i)}$. If y is a continuous variable we call this task "regression"; if y is a discrete variable with a finite number of values we call this task "classification". The research presented in this dissertation is focused on classification tasks, and hence in this chapter we present background on two common classification algorithms: logistic regression and random forests.

2.2.1 Logistic regression

Logistic regression, unlike its moniker, is a machine learning algorithm used for classification tasks in which each sample has an associated binary label. Commonly, binary classification algorithms can be used to answer yes/no questions such as "Should we purchase advertising in this particular city?", or "Does this patient have heart disease?" Logistic regression, one of the simplest classification algorithms, models the probability that an instance will take on a particular label given the observed features, $P(y = 1 \mid \mathbf{x})$. This probability is a soft classification that can be converted to a hard label by thresholding the probability space. The logistic regression model is described in Equation 2.1

$$y = \frac{1}{1 + e^{-\mathbf{w}\mathbf{x}}} \tag{2.1}$$

The vector $\mathbf{w}$ is learned by fitting the model to some available training data. For the jth feature in our dataset, D, $\mathbf{w}_j$ is a coefficient that, if positive, indicates that larger values of x_j increase the probability that y is of the positive class, and if negative, indicates that larger values of x_j decrease the probability that y is of the positive class. It is standard to include an intercept variable $x_0 = 1$ for all instances. When $w_0 > 0$, it is more likely that the instance belongs to the positive class when no information is known about that instance (that is, $\mathbf{x} = 0$). The coefficient vector,

$\mathbf{w}$ is learned by minimizing cross-entropy, or log-loss, detailed in equation 2.2. Procedurally, a gradient descent algorithm can be used to efficiently compute a $\mathbf{w}$ that minimizes cross-entropy. Here $f_w(\mathbf{x}^{(i)})$ is the evaluation of Equation 2.1 when parameterized by a given $\mathbf{w}$, that is $P(y^{(i)} = 1 \mid \mathbf{x}^{(i)}; \mathbf{w})$.

$$J(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \text{Cost}(f_w(\mathbf{x}^{(i)}), y^{(i)}) \tag{2.2}$$

$$\text{Cost}(f_w(\mathbf{x}, y) = \begin{cases} -\log\left(f_w(\mathbf{x})\right), & \text{if } y = 1. \\ -\log\left(1 - f_w(\mathbf{x})\right), & \text{if } y = 0. \end{cases}$$

A common variant of logistic regression includes a penalty function in the form of a regularizer. These penalty terms are a function of the learned weight vector, $\mathbf{w}$, and are used to prevent overfitting. Two popular choices of regularizers are the L_1 penalty, known as LASSO (Tibshirani, 1996), and the L_2 penalty, known as ridge regression (Hoerl and Kennard, 1970). In LASSO-penalized logistic regression, sparsity is encouraged in the learned weight vector; in this way, an L_1 penalty not only combats overfitting but also serves as a feature selection phase. Equation 2.3 is the LASSO-penalized logistic regression loss function which adds the L_1 penalty to the cross-entropy error in Equation 2.2. Minimizing Equation 2.3 requires replacing standard gradient descent with either coordinate descent or a proximal method such as the "Iterative Shrinking-Threshold Algorithm (ISTA)" to account for the lack of a derivative whenever a coefficient changes sign. Ridge logistic regression incorporates an L_2 penalty to combat overfitting by encouraging smaller weights. Equation 2.4 is the ridge logistic regression loss function which can be minimized using standard gradient descent. Both Equations 2.3 and 2.4 incorporate a parameter, λ, which controls the strength of the penalty; larger choices of λ result in stronger effects from the regularization. It is standard to tune the value of λ on some held-aside validation data during the model training process.

$$L(\mathbf{w}) = J(\mathbf{w}) + \lambda |\mathbf{w}|_1 \tag{2.3}$$

$$L(\mathbf{w}) = J(\mathbf{w}) + \lambda|\mathbf{w}|_2 \tag{2.4}$$

2.2.2 Random forests

Ensemble machine learning methods have proven to perform very well in practice (Opitz and Maclin, 1999). An ensemble combines the predictions of multiple base learners through some (potentially weighted) voting scheme to produce a single prediction. Learning theory has shown that successful ensembles are comprised of base learners with uncorrelated errors. That is, given two random base learners in the ensemble, if the first learner forms an incorrect prediction, the second learner isn't much more likely to form an incorrect prediction.

One very successful ensemble method is random forests, which is comprised of multiple decision trees as the base learners (Breiman, 2001). Given a dataset, $D = \{\langle \mathbf{x}^{(i)}, y^{(i)} \rangle\}_{i=1}^{n}$, a random forest constructs k decision trees $\{T_i\}_{i=1}^{k}$. Prior to building a given tree, T, a new dataset, $\tilde{D}$, is formed by selecting n samples with replacement from D. This method, known as bootstrapping, increases the variety of trees learned and thus reduces error correlation among the trees. To further reduce correlation, for each tree (or in an alternative formulation, at each split), only a uniformly, randomly-selected subset of features is considered. Random forests have been shown to be one of the best performing classification algorithms (Fernández-Delgado et al., 2014).

The construction of a particular tree in the forest, T_i, occurs by iteratively splitting its associated bootstrapped training data $\tilde{D}_i$. At each node in the tree, a true/false split test is learned based on one of the features of the dataset. Those instances that satisfy the test proceed down one branch and the remaining instances proceed down the second branch. The split test at each node is chosen during the training process by first randomly selecting a subset of the m features to consider. The number of features considered at each split, f, is a forest-wide hyper-parameter of the random forest that can be used to control the learning algorithm. Amongst these f features, the feature f_j that best separates the positive and negative examples

from one another is selected. Typically, for classification tasks, either Gini impurity or information gain is used as the selection criterion, though alternative criteria exist for different variants of the random forest algorithm. The samples in T_i are iteratively split in this binary fashion until either some stopping condition is met (some minimum number of samples per leaf node), or until a node has only examples from one class. Once the learning algorithm has finished splitting the instances in $\tilde{D}_i$, the training of tree T_i is complete. This process is repeated for each of the k trees in the forest, each with its own bootstrapped $\tilde{D}$. The prediction process for random forests involves taking an instance $\mathbf{x}$, and for each tree starting at the root, following branches according to split tests, and then receiving a label at a leaf. Thus, each $\mathbf{x}$ receives k predictions, one per tree, and the soft-classification from the random forest is the fraction of the k trees that label it positive. Much like logistic regression this soft-classification can be converted to a hard label by the use of a threshold.

2.3 Machine Learning and Healthcare

In recent years, the use of EHRs in healthcare systems has grown substantially (Hsiao et al., 2014). This wealth of data has created an unprecedented opportunity for the use of computational approaches to not only augment existing knowledge of various diseases but also produce predictive models to assess patient risk. Diseases such as breast cancer (Gail et al., 1989) and myocardial infarction (Weiss et al., 2012) have already been successfully modeled through the use of machine learning algorithms. In the context of healthcare data, machine learned models can be used to predict the disease risk of a patient based on the information present in their health record. In some cases, the models produced by a machine learning algorithm can be manually inspected to understand which variables suggest different patient outcomes.

While a variety of machine learned models have been used for healthcare applications, there is often emphasis placed on interpretability, that is, understanding how and why a model makes its predictions Doshi-Velez and Kim (2017). This

often lends favor to naturally interpretable models such as LASSO-penalized logistic regression (Tibshirani, 1996). However, LASSO-penalized logistic regression can produce interpretable, but not credible, models when there are co-linearities in the data (Wang et al., 2018). One of the most common risk assessment tools currently used in practice was a result of the epidemiological Framingham heart study (Dawber et al., 1951). This study resulted in a multitude of logistic regression based risk calculators including a tool for the prediction of coronary heart diseases (Wilson et al., 1998). These tools were created with the express purpose of assisting healthcare providers in evaluating the cardiovascular risk of patients based on their current and historic health. Models such as logistic regression have the benefit of providing an intelligible tool for both patient and physician. An interpretable model is thus often appealing for translation into clinical applications. One does not need to fully understand logistic regression to understand that the coefficients of the Framingham model, $\mathbf{w}$, indicate protective and aggravating factors in the development of cardiovascular disease. While less popular with healthcare data, there have been efforts to explain model predictions for any type of model (including non-linear models). For example, Ribeiro et al. (2016) introduced the approach of "Local Interpretable Model-Agnostic Explanations" (LIME).

It is worth distinguishing between two similar, but unique tasks that apply machine learning to EHR data: electronic phenotyping and disease prediction/diagnosis. The task of electronic phenotyping involves labeling a patient as positive or negative for a history of the particular disease(s). As EHR data can have errors and often cannot be taken at face value, the task of electronic phenotyping aims to better label patient EHR data. Whereas electronic phenotyping is inherently retrospective, disease prediction and diagnosis is a prospective task. Given a patient's health history, a researcher or physician may wish to use machine learning to identify health risks and/or diagnoses of a patient based on their complaints. The tasks of electronic phenotyping and disease prediction/diagnosis can be complementary to one another. A model that is learned to diagnose a patient will often perform better if it is learned from phenotyped data.

2.4 Learning Multiple Outcomes

Machine learning problems are typically divided into classification or regression tasks, where a classification problem requires assigning a category, or label, to an instance, and a regression problem requires predicting a numerical value for an instance. Within classification problems there is typically a division between binary classification tasks with two categories and problems with $k > 2$ categories. Classification problems with $k > 2$ categories are divided into multi-label problems, where $1 <= j <= k$ of k labels could be assigned, and multi-class problems, where only 1 of k categories can be correct. An example of multi-label learning is predicting which of k possible diseases a patient has, as they could well have more than one. An example of a multi-class problem is predicting which one of several manufacturers made a car based on an image.

2.4.1 Multi-label classification

While the typical application of machine learning to EHR data has focused on a single disease or small subset of diseases, some researchers have begun to look at machine learning tasks considering many diseases at once. These multi-disease tasks typically utilize multi-label learning algorithms, as human disease is inherently a multi-label domain. There are two common approaches for multi-label learning: 1) transformation methods where the problem is transformed to one of producing multiple binary classifiers, and using those for labeling, and 2) algorithm adaptation methods that utilize an algorithm to directly output multi-label predictions (Tsoumakas and Katakis, 2007). There are many approaches to problem transformation, and they vary on both how the original multi-labeled training data are manipulated, and the number of classifiers that are produced in the transformed problem. One common approach to problem transformation is to produce k classifiers, one for each of the k classes (Boutell et al., 2004). The training data labels for this approach are manipulated on a per-classifier basis. When predicting the jth of k classes, any example that has label j is considered positive, and any example without label j is considered negative. On the other hand, problem transformation

methods manipulate the original training data to produce multiple binary classifiers, while algorithm adaptation methods manipulate originally binary classifiers to produce multi-label output. The tree based approach proposed by Clare and King (2001) adapts the C4.5 algorithm such that it learns multi-label output. A variety of other learning algorithms have also been adapted for multi-label learning.

While most applications of machine learning to EHR data have focused on a single or a small subset of diseases, there has been some work regarding larger, more diverse, sets of diseases. Miotto et al. (2016) utilized deep learning to create new representations of patient data and then compared those new representations on prediction tasks for a variety of diseases. Similarly, Rajkomar et al. (2018) used a deep network representation of EHR data (from hospital stays) and performed a variety of prediction tasks including in-hospital mortality, unplanned hopsital readmission, and discharge diagnosis prediction. While Rajkomar et al. (2018) predicted many thousands of diagnosis codes, they did not perform any automated phenotyping or case/control selection as we do in Chapter 4. Che et al. (2015) explored the retrospective task of high-throughput electronic phenotyping through the use of deep-learning. Ho et al. (2014) also performed high-throughput electronic phenotyping but through the use of sparse non-negative tensor factorization.

2.4.2 Multi-class Classification

Multi-class classification algorithms are used to model tasks where an instance can belong to one of $k > 2$ distinct categories. Some classification algorithms are naturally compatible with a multi-class setting such as neural networks where the final layer can be made up of k output nodes, one for each class. Then, the output node with the highest score can be selected as the label. k—nearest neighbors needs no adjustment as it is a lazy learner that will simply label the query instance based on the neighbors selected from the training data (which itself is inherently multi-class). Additionally, decision trees and random forests are also naturally multi-class classifiers as long as the splitting criterion used is appropriate for a multi-class setting. Both information gain and Gini impurity, two common splitting criteria for

tree-based learners, are naturally compatible with multi-class settings.

However, while not all two-class classifiers can be gracefully extended to the multi-class setting, it is still possible to use these classifiers by means of converting the multi-class problem into many binary class problems. Three common decomposition techniques are *one-versus-all* (Sejnowski and Rosenberg, 1987), *all-versus-all* (Hastie and Tibshirani, 1998), and *error-correcting output codes* (Dietterich and Bakiri, 1995). In the *one-versus-all* approach, k binary classifiers are trained to distinguish one of the classes versus any of the other classes. A test instance is then classified by choosing the class which has the highest score amongst all k models. The *all-versus-all* approach builds $k(k-1)/2$ binary classifiers, one for each unique pair of classes. During the prediction phase, a test instance is run through all classifiers and for each class, a score is assigned by summing the individual scores of the $k-1$ classifiers to which it belongs. Then, the class with the highest score is assigned as the label for the test instance. The *error-correcting output codes* approach assigns each class a "codeword" which is a binary string of length m. Then, m binary classifiers are trained where positive and negative instances are assigned based on each of the k codewords. A new instance is classified by applying each of the m models to the instance and choosing the codeword whose hamming distance is closest to the m model outputs.

2.5 Machine Learning, Privacy, and Healthcare

Nowadays, machine learning models are deployed for prediction in many privacy sensitive scenarios (*e.g.*, personalized medicine or genome-based prediction). A classic example is disease diagnosis, where a model predicts the risk of a disease for a patient by simply looking at his/her health records. Such models are constructed by applying learning methods from the literature to specific data collected for this task (the training data—instances for which the outcome is known—such as health records of patients phenotyped or labeled for a given disease). Prior experience in machine learning model training suggests that having access to a large and diverse training dataset is a key ingredient in order to enhance the efficacy

of the learned model, *e.g.* (IWPC, 2009). A large training dataset can be created by merging several silos of data collected locally by different entities. Therefore, sharing and merging data can result in mutual gain to the entities involved in the process and, finally, to the broader community. For example, hospitals and clinics located in different cities across a country can locally collect clinical data that is then used to run a collaborative analysis with the potential to improve the healthcare system of the entire country. However, in privacy sensitive scenarios, sharing data is hindered by significant privacy concerns and legal regulations (*e.g.*, Health Insurance Portability and Accountability Act (HIPAA) laws in the United States and General Data Protection Regulation (GDPR) for the European Union). In the example described before, sharing clinical data directly competes with the need for healthcare providers to protect the privacy of each patient and respect current privacy policies and laws.

Based on the preceding discussion, we often face the following dilemma: share data to improve accuracy, or keep data and information secret to protect privacy? Notice that de-identification cannot resolve this standoff: several works (Homer et al., 2008; Narayanan and Shmatikov, 2008) demonstrated that sharing de-identified data is not a secure approach since in many contexts the potential for re-identification is high. More sophisticated anonymization criteria (*e.g.*, k-anonymity, l-diversity, t-closeness, etc.) were proposed by the database community. While arguably better than de-identification, all such "syntactic" approaches work only in presence of assumptions regarding the adversary's background knowledge. Conversely, cryptographic tools can guarantee perfect privacy of shared data in more general situations. For example, a number of *privacy-preserving training* algorithms have been proposed since the seminal paper of Lindell and Pinkas (Lindell and Pinkas, 2000) introduced this concept in 2000. These algorithms use advanced cryptographic tools in order to allow different parties to run known learning algorithms on the merge of local datasets without revealing the actual data. This approach guarantees privacy at the price of high communication and computation overhead. Once the model is learned, we face another privacy problem: using the model to compute a prediction for new instances while both the model and the

instances data are sensitive information privately held by different parties. This problem can be solved using again cryptographic tools, and an algorithm designed for this task is called *privacy-preserving scoring*. In conclusion, a solution that uses the current tools to guarantee privacy at all levels (*e.g.*, for the data providers, model providers, model users) deploys two privacy-preserving systems, a first one for training and a second one for scoring. It should be noted that even the prediction for a given patient could potentially reveal some information about the training patients. While differential privacy (Dwork and Roth, 2014) can address this concern, it comes at the cost of incorporating noise into training that can reduce model efficacy (Fredrikson et al., 2014). Differential privacy is beyond the scope of the present work.

3 SINGLE DISEASE PREDICTION: CALCIPHYLAXIS

In this chapter we begin by exploring the task of predicting a single disease, Calciphylaxis, through the use of machine learning. We present several tasks that are part of the overall machine learning process, including model selection, feature engineering, and nested cross-validation. Moreover, we demonstrate how a machine learning algorithm can be used to better understand a disease through the use of feature importance inspection. This chapter serves as a primer on the use of machine learning for disease prediction and is meant to serve as a foundation for the task of predicting thousands of diseases using high-throughput machine learning as presented in Chapter 7.

3.1 Background

Calciphylaxis, also known as Calcific Uremic Arteriolopathy (CUA), is a highly morbid disorder that presents with necrotic lesions of the skin resultant from ischemia caused by calcification of the small and medium sized arteries (Coates et al., 1998). The mortality rate of Calciphylaxis has been measured in excess of 50% following one year from initial diagnosis (Weenig et al., 2007). One of the most commonly associated comorbidities with Calciphylaxis is Chronic Kidney Disease (CKD) (Coates et al., 1998). In particular, Calciphylaxis is found most often among patients in the late stages of CKD and end stage renal disease (ESRD) (Coates et al., 1998). While Calciphylaxis was originally named by Hans Selye in 1962, there is still much that is unknown about its risk factors. Known risk factors include: CKD, ESRD, mineral and bone disorders, diabetes mellitus, hyperphosphatemia, female gender, obesity, warfarin use, and ethnicity (Sowers and Hayden, 2010). Furthermore, to the best knowledge of the authors of this work, there are no risk assessment models currently used in practice for identifying patients at risk for development of Calciphylaxis.

In recent years, the use of electronic health records (EHRs) in healthcare systems

has grown substantially (Hsiao et al., 2014). This wealth of data has created unprecedented opportunity for the use of computational approaches to not only augment existing knowledge of various diseases, but also produce predictive models to assess patient risk. Diseases such as breast cancer (Gail et al., 1989) and myocardial infarction (Weiss et al., 2012) have already been successfully modeled through the use of machine learning algorithms. Machine learning, a branch of artificial intelligence, focuses on producing algorithms that learn rules or relationships about a particular set of variables to predict the value or outcome of an unknown variable. In the context of healthcare data, machine learned models can be used to predict the disease risk of a patient based off of the information present in their health record. In some cases, the models produced by a machine learning algorithm can be manually inspected to understand which variables suggest different patient outcomes. In this chapter, we present the use of two machine learning algorithms, lasso-penalized logistic regression (Tibshirani, 1996) and random forests (Breiman, 2001), in the context of prediction and risk factor analysis for Calciphylaxis. We present these methodologies on several sub-populations of CKD and ESRD and show that they produce strong prediction of Calciphylaxis.

3.2 Experimental Methodology

Data

Our dataset is comprised of patients who visited the Marshfield Clinic, a health care system that serves Northern and Central Wisconsin. Data include diagnosis codes, in the forms of both ICD-9 and ICD-10 codes, demographic information, laboratory values, vitals readings, procedures, and medications present on patient records. These values were gathered for patients who were identified as case or control patients for our experiments. Case patients were manually identified by their attending surgeon, who confirmed Calciphylaxis diagnosis via inspection of patient records and histological examination of excised tissue. Control patients were required to have no diagnosis codes indicative of Calciphylaxis in their records including ICD-9 codes 275.49 and 709.1, and ICD-10 codes L95.9 and E83.59. Ad-

	Any Stage	**Stage 3**	**Stage 4**	**Stage 5**	**ESRD**
ICD-9 Code(s)	585-585.9	585.3	585.4	585.5	585.6
ICD-10 Code(s)	N18-N18.9	N18.3	N18.4	N18.5	N18.6
# Cases	38	10	15	12	17
# Controls	363	100	148	117	165
Total Patients	401	110	163	129	182

Table 3.1: Included diagnosis codes and number of cases and controls included in each experimental condition.

ditional criteria were included for control patients that were dependent upon the various experiments we completed. In all experiments, we attempted to match up to 10 cases for each control. We chose a 10-to-1 ratio based on the small number of confirmed cases. We present research on 5 different CKD patient populations: patients with any stage of CKD, referred to as any stage, patients with stage 3 CKD (stage 3), patients with stage 4 CKD (stage 4), patients with stage 5 CKD (stage 5), and patients with stage 5 CKD requiring chronic dialysis, which we will refer to as end stage renal disease (ESRD). Further details of these patient populations are in Table 3.1.

In addition to our experimentation based on stage of chronic kidney disease, we explored the effects of binary versus continuous feature representations. The raw dataset was structured such that each non-demographic variable, e.g., a particular lab test, had a value for the number of times the patient had received that particular variable on their record. The counts for each variable were done from the start of the patient's record until an experiment-specific date, e.g., their first entry of a particular stage of chronic kidney disease. One potential concern about using count data is the likelihood of chronically sick patients to have an increased number of healthcare encounters. In this way, uncorrelated variables that are recorded often such as a height measurement vital could become inappropriately correlated with disease status. This was of particular concern as Calciphylaxis tends to occur in much sicker patients. For this reason, we also explored the use of binary features. We constructed these features by setting any non-demographic feature value to 1 if

the patient had ever had it entered on their record, and 0 if they had never had it on their record. In the case of continuous features, we simply scaled the counts for each feature to be between 0 and 1, which was of use when comparing the relative importance of features when evaluating our models.

Model Construction Methods

Our preliminary research into predicting Calciphylaxis began with matching Calciphylaxis positive cases with controls who were not positive for Calciphylaxis. Candidate controls had no Calciphylaxis diagnosis codes on their records, and a minimum of 10 unique diagnosis codes on their records. The minimum data requirement was implemented to ensure sufficient data for prediction. We selected up to 10 controls for each case with the requirement that the control and case have the same gender, and birthdates within 30 days of one another. Additionally, controls were required to have data both before and after the diagnosis date of Calciphylaxis for the case patient. In this way, we attempted to ensure that the control patient was alive at the time of the case patient's diagnosis and was negative for Calciphylaxis. Both case and control data were right censored 30 days prior to the diagnosis date of the Calciphylaxis patient.

These initial experiments resulted in modest predictive models with an overwhelming quantity of the predictive features related to CKD. Because CKD showed such a strong relationship with Calciphylaxis, we wanted models that could differentiate between CKD patients and guide us in understanding why certain CKD patients go on to develop Calciphylaxis while others don't. We additionally wanted to explore how well Calciphylaxis could be predicted, and how far in advance. While we were interested in risk of Calciphylaxis for patients at any stage of CKD, we found we had insufficient data to build predictive models for stages 1 and 2, as these had less than 10 cases each. To these ends we produced predictive models of Calciphylaxis among 5 different types of patients: any stage of CKD, stage 3 CKD, stage 4 CKD, stage 5 CKD, and stage 5 CKD requiring chronic dialysis (ESRD).

We further explored the predictive capacities of two different models: random forests and logistic regression. Random forest models are well known for both their

strong accuracy and their resilience to high dimensional data. Furthermore, random forests are capable of capturing nonlinear interactions in data. While random forests are strong predictors, they are often difficult to interpret, and interpretation is often critical in a healthcare context. For this reason, we additionally explored logistic regression models. Because logistic regression is not inherently resilient to high dimensional data, we employed lasso-penalized logistic regression, which utilizes an L1-regularization term in the objective function of the model to penalize features that were only marginally informative and to encourage a small set of strongly predictive features in the final model. This is necessary not only to obtain comprehensible models but also to obtain accurate ones, as the various types of CKD we explore have between 5,000 to 10,000 unique features present on the records of the patients and logistic regression without some form of dimensionality reduction can perform very poorly in such situations. The L1 or "lasso" penalty is a widely-used approach to dimensionality reduction in regression.

For both the random forest and the logistic regression models we performed a case-control matched leave-one-case-out cross validation. For each experiment we chose k-folds, one fold for each of the k case patients and its matching controls. In this procedure, for each round of cross validation one fold was chosen to be left out and was comprised of a single case patient and the up to 10 control patients matched with it. In the case of random forests, models were constructed using the remaining training data. We built 500 decision trees for each forest and used the square root of the number of features as the number of features to consider at each split. Trees were grown to leaf purity where possible and splits were chosen via Gini gain that was calculated in a balanced fashion in that case patients were weighted more heavily such that their total combined weight equaled that of the control patients.

For logistic regression we performed an additional layer of internal leave-one-case-out cross validation to tune the penalty coefficient for the L1 regularization term. In this tuning procedure we consider 10 different penalty values logarithmically spaced between 10^{-4} and 10^{4}. Each penalty value was evaluated via an internal cross validation layer and the optimal penalty value was chosen based on

the performance of the various models as judged by a weighted accuracy measurement that ensured equal combined weights for the cases and the controls. Thus, for each external cross-validation fold the remaining folds were used to select a penalty value and a new predictive model was trained on those folds and finally evaluated on the original held out fold. Logistic regression models were all constructed using a balanced class weight approach similar to that employed in the random forest models.

For each of the 20 pairs of experimental condition and model, e.g. predicting CKD stage 4 with random forests using binary features, we repeated the model construction 30 times. These replications were done because random forests are stochastic by nature, and logistic regression may have multiple equally good solutions to their optimization problem, both of which can produce varying results across multiple runs from the same data set. In this way, we were able to better estimate the predictive quality achieved under each experimental condition.

Model Evaluation Methods

Model evaluation is done quantitatively via the construction of receiver operating characteristic (ROC) curves and precision recall (PR) curves. We primarily use the area under the ROC curve (AUC-ROC) as a numerical evaluation for the quality of our models. We do this because AUC-ROC provides a metric of efficacy that is unbiased by class skew of a dataset. Because we attempt to achieve a 10-to-1 control to case ratio this may not be representative of the true population skew, and thus we provide a fairer analysis of our results via ROC-curves (Boyd et al., 2012). We demonstrate PR-curves for our best performing algorithm for completeness, but do not use them in ranking the quality of the models. Both ROC-curves and PR-curves were constructed for each model in a similar fashion. For a single repetition of a single experiment we first produced a vector of predicted probabilities corresponding to model estimated risk for each patient. We produced this vector during the leave-one-case-out k-fold cross validation by using the model built on the training folds to predict the labels of the test fold. Our final predicted probability vector was the union of these predictions as each patient was only in a test fold once.

From these predicted probabilities and the true class labels we constructed either a ROC- or PR- curve in the typical fashion. Then across the 30 repetitions for each experiment, we performed vertical averaging of the 30 resulting curves to yield a final curve.

For each of the five experimental conditions, we wish to know which of the four models performed best. We evaluated model performance via a sign test and evaluated 12 hypotheses which were of the form: "Binary feature random forest is significantly better than binary feature logistic regression". These twelve hypothesis were all of the unique ordered pairs of the four models types drawn without replacement. For each experimental condition we chose an experiment-wide p-value of $\alpha = 0.05$ as our threshold for significance. We performed a Bonferroni correction and required $p_i \leqslant 0.05/12$ for an individual hypothesis to be significant. If a particular model was significantly better than the three other models, then we considered it to be the best model for that experimental condition.

Our p-values for each hypothesis were calculated via the aforementioned sign test, the details of which we provide now. Because the 30 repetitions for each experimental condition and model are not truly independent from one another, we cannot use bootstrapping to calculate confidence intervals on our AUCs. Instead, we use a sign test to determine if a particular model is correct more often than another model a statistically significant number of times. For each model we first construct an ROC-curve in the fashion described above. To convert the probabilities from each algorithm to labels, we choose a threshold corresponding to an 80% true positive rate, or recall, for our algorithms. Because of the serious nature and high mortality rate of Calciphylaxis we chose 80% recall to reflect the significant cost difference between false negatives and false positives. Thus, for each experiment the threshold corresponding to 80% recall was used to label predictions as either positive or negative. For every patient we then checked the label assigned by both models and its true label, if both models assigned the same label, that patient was considered a tie and not counted, however if the models differed then if the first model received either a win or a loss if it was correct or incorrect respectively. Then, each hypothesis was tested by first counting n_{wins}, the number of times the first

model predicted a patient correctly and the second model predicted that patient incorrectly, and n_{loss}, the converse statement. Finally, we calculate the cumulative distribution of the binomial function $\texttt{Binom}(n_{wins}, n_{loss}, 0.5)$ to determine a p-value for a particular hypothesis.

We perform a qualitative evaluation of our models by inspecting the top performing features returned from our models. In the case of logistic regression, the top features are those with the highest absolute value of their coefficients. In the case of random forests, we used the approach detailed in Breiman 1984 to determine feature importance values via the Gini importance (Breiman, 1984).

3.3 Results

In general, we find that we achieve superior performance predicting Calciphylaxis from a specific stage of CKD compared to an arbitrary stage. We present in Figure 3.1 a grouped bar plot detailing model performance by stage of CKD and choice of model and feature set. We note that in general random forests outperform logistic regression and that use of binary features roughly outperforms use of continuous features. In Table 3.2 we present the precise ROC-AUC values achieved in each experiment. In two of the five experimental conditions we found random forests with binary features to outperform the other three approaches with a statistically significant difference.

Given the strong performance of random forests using binary features, we chose to visualize the ROC- and PR-curves for this particular model and feature engineering scheme across the five experimental conditions and present this in Figure 3.2. We note that the models predicting CKD at a specific stage appear to be clustered together and roughly separated from the ROC-curve representing prediction at an arbitrary stage of CKD. Additionally, there appears to be some separation between the PR-curve representing prediction of ESRD as compared to the other stages of CKD.

Additionally, for each of the five experimental conditions, we present in Table 3.3 the top 10 features used in prediction for the binary feature random forest.

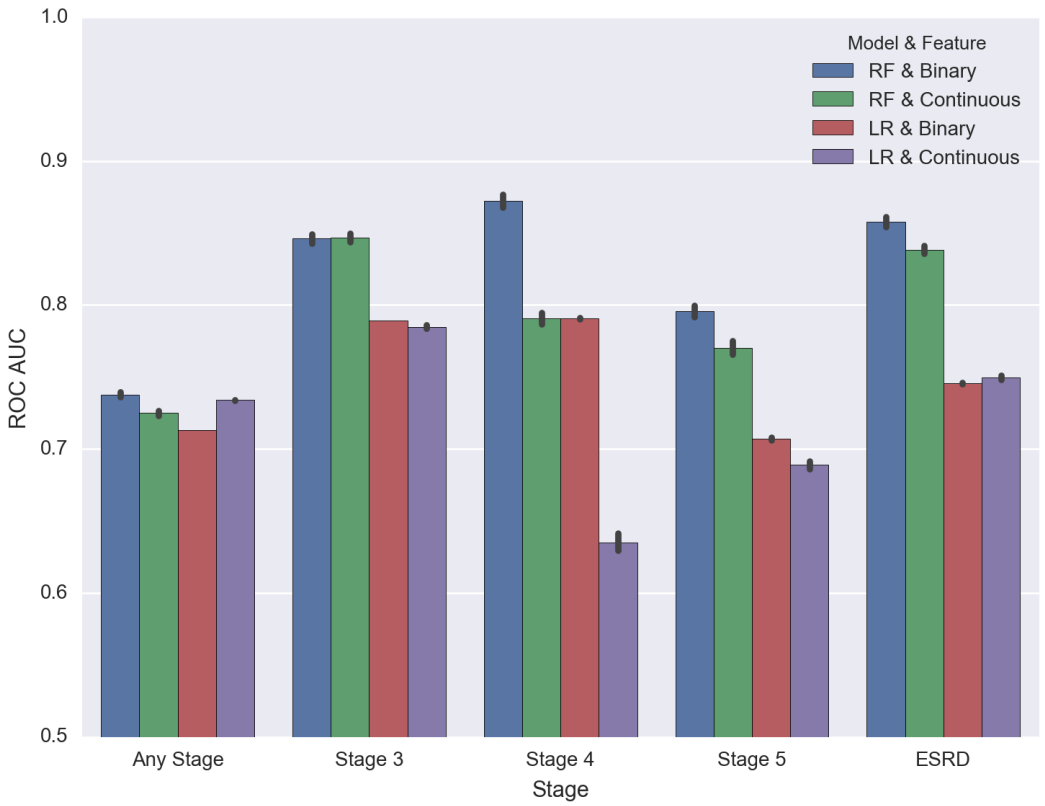


Figure 3.1: Mean ROC-AUC achieved on the 20 experimental predictions for the task of predicting Calciphylaxis.

We note that many of these features are strongly related to many already known risk factors for Calciphylaxis. Additionally, these features are suggestive that we successfully control for CKD, as few CKD related features appear among the top features for these models. Furthermore, we find that there may be some temporal confounding, as the feature "Prescription transmit via erx system" is present in the top ten features for both stage 4 and stage 5 models. This particular feature can appear when a model is attempting to distinguish cases and controls temporally, as the introduction of electronic transmission of prescriptions can be used to infer if a patient's data is from a more recent period by inspecting if this feature is present on their record.

	RF Bin.	**RF Cont.**	**LR Bin.**	**LR Cont.**
Any Stage	0.7377	0.7248	0.7126	0.7335
Stage 3	0.8487	0.8492	0.7940	0.7897
Stage 4	**0.8718**	0.7907	0.7901	0.6349
Stage 5	0.7960	0.7702	0.7076	0.6892
ESRD	**0.8575**	0.8377	0.7450	0.7501
Average	0.8223	0.7945	0.7499	0.7195

Table 3.2: AUC-ROC values for each experimental condition and algorithm. For each experimental condition, the algorithm that performed best is highlighted in bold if it is statistically significantly better than the other three algorithms as determined by a sign test. Note that just binary random forests achieved statistical significance and did so in two experimental conditions. For CKD stage 4 binary random forests outperformed its competitors with $p \leqslant 3.78e-4$. For ESRD stage 4 binary random forests outperformed its competitors with $p \leqslant 1.92e-4$.

3.4 Discussion

In this work we explored prediction of Calciphylaxis at various stages of CKD, through the use of random forest or logistic regression models and either binary or continuous features. We find that overall prediction of Calciphylaxis is achievable with strong results and note that in each of the experimental conditions in which we predict Calciphylaxis during a specific stage of CKD, we find at least one model with a AUC-ROC of nearly 0.8 or above. Such AUC-ROC values suggest models of high quality capable of predicting Calciphylaxis with reasonable efficacy. Given the high mortality rate of Calciphylaxis, we feel that early prediction of Calciphylaxis would be of great benefit for physicians who wish to monitor patient risk. In particular, we find some of our strongest results when predicting Calciphylaxis using binary feature random forests for the experimental conditions of stages 3 and 4 where patients have moderate to advanced CKD.

It is worth noting that for a given task and feature type, random forests nearly always outperformed logistic regression. We feel that this is likely due to the ability for random forests to capture nonlinear interactions in data. Furthermore, for a given task and model type, binary features achieved AUC-ROC values that were

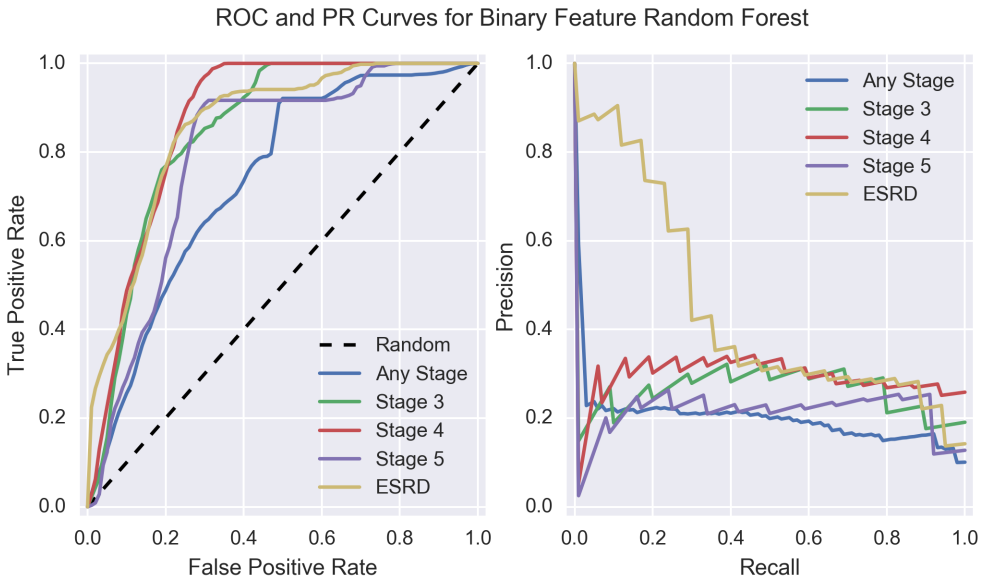


Figure 3.2: ROC- and PR-Curves for binary feature random forest across the five experimental conditions.

typically equal to or better than those achieved with the use of continuous features. This is of interest as our constructions for binary and continuous features ensured that continuous features had strictly greater information. For every feature other than age, the binary feature could be obtained from the continuous feature by thresholding such that a continuous feature of value 0 becomes a binary feature of value 0, and a continuous feature of value anything greater than 0 becomes a binary feature of value 1. In the case of random forests this could be expressed as splitting on a particular continuous feature, f_i, with $f_i > 0$ being the logical test for indicating which branch to choose. The poorer performance from continuous features suggests that there may be some overfitting occurring. By using binary features, we are limiting the expressiveness of our models, and such an action could cause improvement in the case of a model that is overfitting.

In our analysis of the top performing features for the various models and experimental conditions, we found that features could be grouped roughly into one of three categories: previously known Calciphylaxis risk factors, indicators of

Rank	Any Stage	Stage 3	Stage 4	Stage 5	ESRD
1	Hepatitis B Surface Ag	Obesity	Anemia in CKD	Obesity	Thyroxine (T4)
2	Anemia in CKD	Lactescence/ Chylomicrons	Morbid Obesity	Morbid Obesity	Obesity
3	Secondary Hyperparathyroidism of Renal Origin	Chylomicrons	Thyroid Stimulating Hormone-Reg'l C	Parathyroid Hormone (PTH),1-84	Amylase-Pancreatic
4	Direct Microscopy	ALT (GPT)	Obesity	Radiologic exam knee complete 4/more views	Age
5	Ulcer of Lower Limb, Unspecified	Differential Polychromatophili	Chylomicrons	Instrument Neutrophil #	Frac.O2 Hb, Arterial
6	Iron Defic Anemia Nos	Differential Poikilocytosis	Lactescence/ Chylomicrons	Age	Arthropathy, unspecified
7	Hepatitis B Surface (HBs) Ab	Uric Acid, Blood	Thyroxine (T4)	Uric Acid,Bld	Prothrombin Time (PT)
8	% O2 Saturation	Differential Activated Lymph	Secondary Hyperparathyroidism of Renal Origin	Non-HDL Cholesterol	Abdominal Pain
9	Differential Poikilocytosis	Sodium serum plasma or whole blood	Prescription transmit via erx system	Prescription transmit via erx system	Chronic Liver Disease Nec
10	Blood Urea Nirtrogen-Post-Dial	Platelet Estimate	Hypercholesterolemia	Skin Suture Nec	Blood count complete automated

Table 3.3: Ranking of 10 top random forest model features for each of the 5 CKD experiments: any CKD stage, stage 3 CKD, stage 4 CKD, stage 5 CKD, and ESRD. For each experiment, feature importance values were averaged first across each of the k-folds for cross-validation, and then those values were averaged across the 30 repetitions completed to produce a final importance value for each feature.

potential confounding in the model, or potential Calciphylaxis risk factors with minimal or no previous support. Of those risk factors for Calciphylaxis that are previously discovered, our models pick up most strongly on obesity and its related risks, anemia/low iron, and hyperparathyroidism. It is worth noting that we see minimal use of CKD features, which suggests that our method for controlling for CKD was largely successful. There are some features which suggest either temporal confounding. In particular, the feature "Prescription transmit via erx system" is likely being used to separate patients by how recently they visited the Marshfield Clinics. Such a feature can be exploited during cases of temporal confounding as the use of an electronic prescription transmittal system is a relatively recent addition to workflows in healthcare systems. Finally, we find some of our top features are related to liver disease and/or liver function, a relationship that is not strongly established with Calciphylaxis. In particular, we see both various liver function labs and testing for hepatitis B as strong features for predicting Calciphylaxis.

3.5 Conclusion

Patients suffering from late stage CKD and ESRD are known to be at a much higher risk for Calciphylaxis (Coates et al., 1998), a vascular disorder with a one-year mortality rate in excess of 50% (Weenig et al., 2007). By identifying at risk patients, we provide an opportunity for these patients to take actions to mitigate their risk factors and avoid a potentially deadly disease. Additionally, by modeling Calciphylaxis to predict patient risk, we invite the opportunity to discover more about the potential risk factors of what is a poorly understood disease. In this work we explored prediction of Calciphylaxis at various stages of CKD, through the use of random forest or logistic regression models and either binary or continuous features. From this work, we found that random forests using binary features tended to outperform other methodologies and that we could typically predict Calciphylaxis with strong efficacy. Furthermore, we found that our models not only largely exploited known risk factors for Calciphylaxis, but also found some evidence for a previously unsupported connection to liver function and hepatitis.

We note that our results show these models to be of high quality and intend to explore potential for translating these models to a clinical setting, where they could be used to identify patients who are of higher risk for developing Calciphylaxis.

3.6 Summary

In this work we demonstrated a machine learning pipeline as applied to a single healthcare condition. It is standard in these cases to perform some combination of feature engineering, model selection and validation. Moreover, in the healthcare domain it is often advantageous to inspect these models both for the validation of their quality and for their potential insights into the condition of interest. However, this methodology is time consuming and does not scale well to the prediction of the thousands of interesting health events that exist. For this reason, in Chapter 4 we turn our attention to the task of predicting many diseases at once.

Co-authorship and Acknowledgements

The work presented in Chapter 3 is the result of a collaboration with multiple researchers that was published in the proceedings of the 2018 AMIA Joint Summits. This work was completed with the following authors and affiliations in order of their appearance as co-authors: Ross Kleiman[1,2] M.S., Eric R. LaRose, B.S.[1], Jonathan C. Badger, Pharm.D.[1,3], David Page, Ph.D.[2,3], Michael D. Caldwell, M.D., Ph.D.[4], James A. Clay, M.D.[4], and Peggy L. Peissig, Ph.D.[1]; [1]Marshfield Clinic Research Institute, Marshfield, WI; [2]Department of Computer Sciences, University of Wisconsin, Madison, WI; [3]Department of Biostatistics and Medical Informatics, University of Wisconsin, Madison, WI; [4]Department of Surgery, Marshfield Clinic. This work was supported by the NIH BD2K grant: U54 AI117924, the NIG NIGMS grant: R01 GM097618, and the NLM training grant: 5T15LM007359.

4 HIGH-THROUGHPUT MACHINE LEARNING FROM ELECTRONIC HEALTH RECORDS

4.1 Introduction

Much of the existing literature related to predicting disease risks via machine learning algorithms focuses on single diseases or small subsets of diseases (Gail et al., 1989; Weiss et al., 2012; Huang et al., 2014; Himes et al., 2009). Small-scale learning tasks often allow for manual selection of the features used by the learning algorithm and manual definition of the cases and controls. However, the framework to predict a single disease is not sufficient to assess risks for the thousands of possible illnesses a patient can face. The current paradigm, of each publication or tool targeting a specific disease, creates a labyrinth of different implementations and data collection assumptions. Therefore, to answer the question of how well all diagnoses can be predicted, we cannot simply aggregate the existing models of varying approach and data source but must rather produce a single pipeline capable of to predict all diagnoses at once using the same set of rules, assumptions, and data source. Although there has been some work that has predicted multiple patient diagnoses, to the best of our knowledge it has either been done to test representations of patient data (Miotto et al., 2016) or in the context of a retrospective phenotyping task (Che et al., 2015), to "predict" or specify who really had an event such as an MI, rather than who will have the event in the current year. We believe that our work is greater in breadth than the previous research and that we are the first to produce a machine learning pipeline that attempts to learn thousands of unique models predicting the various diagnosis risks a patient may receive in the future.

We hypothesize that it is possible to predict nearly all disease risks simultaneously at a competitive level of performance. We call the approach of learning thousands of predictive models at once "high-throughput machine learning," and we call the application of the approach to predicting risks for many diagnosis codes from EHR data "pan-diagnostic machine learning." To test our hypothesis that

accurate pan-diagnostic machine learning is possible, we produced a machine learning pipeline that builds a unique model for each diagnosis code in the 9th revision of the International Classification of Disease (ICD-9) (ICD-9 codes were used rather than ICD-10 because (1) we initiated this study before our data source, Marshfield Clinic Health System (MCHS), switched to ICD-10 in October, 2015, and (2) all historical data were coded in ICD-9). A future incorporation of ICD-10 codes would require a standard lexicon, such as the Systematized Nomenclature of Medicine — Clinical Terms (SNOMED CT), that is capable of handling both ICD-9 and ICD-10 codes. Although ICD-9 codes primarily serve billing purposes and do not always correspond with diseases, they are useful for both their ubiquitous and often standardized presence in healthcare systems and the fundamental role they play in phenotyping patients. The pipeline is general enough to be run on ICD-10 codes, with very little modification once sufficient longitudinal data is available.

In this chapter, we present a high-throughput machine learning pipeline that learns diagnosis-specific models for risk prediction. Starting with only EHR data, the pipeline learns one model per diagnosis code. Our modeling approach automatically learns appropriate case-control matching rules, including controlling for pre-existing conditions, for each diagnosis code. Furthermore, each model learns rules about which populations of patients are appropriate for the model to be applied to. In this way, our pipeline would automatically learn that a model predicting 10-year Alzheimer's disease risk should not be applied to adolescents. While our pipeline is designed to require minimal intervention to allow for ease of use, it is also highly customizable and can accommodate prior knowledge such as phenotyping rules for user-specified case-control definitions. We showcase such an example with custom rules for identifying individual pregnancies when predicting pregnancy complications. In this work we present the results of our pipeline using data from the MCHS and introduce a new dataset of performance measures and variable importance values for tens of thousands of clinical prediction tasks. Moreover, the code for this pipeline is made publicly available so that it may be readily applied to data from other healthcare entities.

4.2 Methods and Materials

Electronic Health Record Data

Data preparation, de-identification, and patient consent

Our dataset consisted of the demographics, diagnoses, labs, procedures, and vitals of 1.5 million patients who received care at the MCHS. Prior to sharing the data, the MCHS fully de-identified all data by not only removing any protected health information (PHI) but also by de-identifying codes and values. All names for ICD-9 codes, procedures, lab tests, and vital measurements were de-identified, and a separate mapping file was produced. Additionally, the values for lab and vital measurements were first mapped to a reference range and then de-identified. The mapping files for re-identifying data were used only for summary statistics and to properly analyze results and produce figures. The UW-Madison Institutional Review Board (IRB) deferred to the MCHS IRB under the Wisconsin IRB consortium and waiver of consent was obtained from the MCHS IRB.

Pre-processing data

Patients with insufficient data were removed from the dataset. Our original dataset was comprised of some 1.5 million patients of which 1.1 million met our data minimum requirement. We removed patients with fewer than four distinct diagnoses recorded over their lifetime or whose medical data did not span multiple entry dates. This was done to help ensure patients in the dataset received regular care at MCHS. If a patient did not have both a sex and date of birth recorded, then he/she also was removed from the dataset. Our dataset also contained many ICD-9 codes not related to diseases. These non-disease codes included ICD-9 procedure codes and codes beginning with the letter 'E' or 'V'. E-codes describe the external cause of a disease or injury and V-codes do not pertain to disease or injury and are used for supplementary documentation purposes. We did not build predictive models for V-codes, E-codes, or ICD-9 procedure codes, though these data were kept as candidate (input) features in the training data. Additionally, we did not

build predictive models for two major categories in the ICD-9 hierarchy: "Perinatal Diseases" (ICD-9 codes 760-779) as there was insufficient data in the records of newborns to predict disease, and "Symptoms, Signs, And Ill-Defined Conditions" (ICD-9 codes 780-799) as these are not diseases and were outside of the scope of this research.

Feature Extraction

Patient EHR data in a research data warehouse is distributed across many tables in a relational database, with tables for diagnoses, labs, vitals, procedures, demographics, etc. Furthermore, each patient's health record can be viewed as an irregularly-sampled timeline of medical events. In order to use random forests, we needed to convert this rich, irregular patient data into a single relational table with one row, or record, per patient, and with patient features arranged by column. To accomplish this, all features were divided into multiple counts of events occurring within time ranges. The time ranges we used were "last 1 year", "last 3 years", "last 5 years", and "ever" (Lantz, 2016). These time ranges are relative to a specific date, that is prior to a case patient's first entry of a diagnosis of interest. For example, a surgical procedure 6 months prior to a diagnosis we are predicting would add a single count to all four columns for that surgical procedure feature in a given patient's row. Lab value event counts within a time range were further subdivided based on outcome (normal, abnormal, high, low etc.).

Different lab tests have varying numbers of associated outcomes; for example, one test's result could be continuous while another could be discrete, such as either normal or abnormal. All continuous lab results were first discretized by the MCHS using reference ranges. We then constructed one feature per (test, reference-value) tuple, and each such unique tuple received separate counts for the four time ranges. For example, a blood sodium measurement would have separate time range counts for values of normal, high, low, etc. The process of discretizing the continuous lab values and counting (test, reference-value) tuples as unique features was also done for vitals (e.g. high/low systolic blood pressure).

Pan-Diagnostic Machine Learning Experiments

While our system can use a wide variety of possible machine learning algorithms, we employ random forests (Breiman, 2001), which form a prediction by taking a majority vote amongst an ensemble of decision trees. Random forests are known for their competitive accuracy (Caruana and Niculescu-Mizil, 2006) and resilience to high-dimensional data (Breiman, 2001); because of their use of decision trees, they can capture some non-linear interactions of multiple variables without the need to predefine interaction terms; because of their ensemble nature, they are more robust against overfitting than are individual decision trees (Breiman, 2001). There are many alternative popular options for tree-based ensemble learners, such as Extremely Randomized Trees (Geurts et al., 2006) and XGBoost (Chen and Guestrin, 2016). Our dataset consisted of the labs, vitals, diagnoses, procedures, and demographic information of 1.1 million patients from the MCHS which serves patients in the Northern and Central regions of Wisconsin. Using these data, we set out to predict all disease-specific ICD-9 code risks prior to patient diagnosis. The models can be constructed to make predictions for arbitrary user-defined time-periods prior to diagnosis. In this work, we explore how predictive performance varies with time by evaluating models constructed to predict initial diagnosis at seven different time windows: 1-month, 6-months, 2-years, 5-years, 10-years, 15-years, and 20-years. It is worth noting that predicting health events with a long forecast window is likely to truly be a disease risk prediction, whereas a shorter window such a 1-month could be considered a blend of risk prediction and diagnosis, as the patient may be undiagnosed for a disease they physiologically already have. Of course, where the shift from risk prediction to diagnosis occurs is disease-specific. Additionally, we wished to explore the relationship between model efficacy and disease type and did so by leveraging the disease categories present in the ICD-9 hierarchy. We evaluated model performance by measuring Area Under the Receiver-Operating Characteristic Curve (AUC) via ten-fold cross-validation, a robust form of hold-out testing commonly used for evaluation in machine learning.

A recurring challenge we faced in many steps of the pan-diagnostic machine

learning pipeline was the need for algorithms to be general, flexible, and able to automatically account for differences in the disease mechanisms and affected populations of different diagnoses. Several diagnosis codes required specialized refinements to our general approach for case-control matching (see Case-control matching and Dynamic definition refinement), to avoid artificially high accuracies for trivial reasons. This arose when predicting risks for diseases that are complications of preexisting states of health. For example, when predicting risk for a pregnancy complication, we wish to match a case with a control who is not only female, but also pregnant, and specifically at a similar stage of pregnancy. Simply choosing each control to be of the same age and sex as the case makes prediction artificially easy, because pregnancy and pregnancy-stage become accurate predictive features. To automatically determine if a diagnosis requires a set of preexisting diagnoses we created a novel approach which we call "dynamic definition refinement" (DDR) (see Dynamic Definition Refinement). For each diagnosis code, DDR learns a set of prerequisite diagnoses that we require both the case and control to have on their record prior to the prediction date. For example, DDR would learn a rule that a patient who has a diabetic complication should previously have diabetes on their record, and hence any matched control should also have diabetes on their record.

Building random forests

To construct the random forest models from summary table data and to calculate AUCs we used scikit-learn 0.16.1, an open source machine learning library (Pedregosa et al., 2012). We used scikit-learn's implementation of the RandomForestClassifier. For each model, 500 trees were constructed and 10% of features were randomly selected as candidates for each split. All other model specifications were default and are listed in Table 9.1 for completeness.

For each ICD-9 code a separate random forest model was trained on a minimum of 1,000 and a maximum of 10,000 randomly selected case-control paired patients sampled from the dataset of 1.1 million patients. While we chose a maximum of

10,000 patients due to computational constraints, many of our models certainly would have benefited from the use of additional case patients, or even a greater control to case patient ratio. We randomly sampled a maximum of 10,000 case-control paired patients by first randomly selecting a case patient, and then randomly selecting a control patient that met the criteria that follow. If a code did not meet the minimum of 1,000 case-control paired patients, it was not modeled. We note that while there were 22,396 unique ICD-9 codes present in our dataset, the majority of these were very infrequently used for diagnosis during clinical care either due to redundancy or lack of prevalence of the corresponding disease. By requiring a minimum of 1,000 case-control paired patients we built at most 3,586 models for a given truncation window. Cases were considered positive for a diagnosis if the ICD-9 code appeared two or more times in the patient's record ('rule of two') (Rasmussen et al., 2014). Controls paired to cases must have never had the diagnosis being modeled recorded in their record. Cases and controls were also required to have the same sex and no more than a 30-day difference in date of birth. Further matching criteria were utilized and are detailed in the Dynamic Definition Refinement (DDR) and Break Point Analysis (BPA) methods sections. After matching, we identified a truncation date for each pair of case-control patients based on the date of the first entry of the diagnosis of interest on the case patient's record and the length of the truncation window. If day t was when the case patient had their first entry of the diagnosis being predicted, and the window length is w, then any data following day $t - w$ was excluded from the training data. This truncation is essential to prevent class label information from leaking into the training data, which would bias resulting estimates of model performance on future patients, specifically making them overly-optimistic. Even more aggressive truncation was applied when we tested prediction ability even further in advance, requiring truncation from one year to 20 years prior to the event to be predicted. Before carrying out cross-validation for a given prediction task, unsupervised feature selection was employed to retain only features populated for greater than 1% of patients (without regard to case vs. control label). In this work, we chose to use cross-validation rather than the out-of-bag (OOB) estimates for our random forest models because OOB error can in

some cases provide an incorrect assessment of model performance (Mitchell, 2011).

Case-control matching

During case-control matching, two patients are considered to have a sufficiently close date of birth if the date of birth of the control patient is within 30 days of that for the corresponding case patient. We did not consider case patients or control patients where the patient's date of birth would be after the truncation date, e.g., we would not train on an adolescent's data when building a model to predict 20-years in advance. Furthermore, we considered the concept of a last known contact or the most recent data point in a patient's record. We required that control patients have a last known contact greater than or equal to the date of diagnosis of the case patient. This helps ensure that the control patient had the potential of being diagnosed but was not.

Dynamic definition refinement (DDR)

Due to the massive number of models built, it would be infeasible for a human to individually develop a prerequisite diagnosis list for each model. Therefore, we developed DDR to perform most this work, which we could then inspect and update. The DDR approach involves two stages: an algorithmic stage to identify potential prerequisite diagnoses and a manual stage to correct over- or under-controlled diagnoses. While a serious effort was made to choose realistic controls for each disease and only predict appropriate ICD-9 codes, we realize that pan-diagnostic machine learning is not without limitations and some of the models we produced may still be somewhat overly optimistic, influenced in part by limitations of the ICD-9 hierarchy.

For each diagnosis code, DX_i, DDR first identifies all diagnosis-positive patients (established via "rule of two"). Among these patients the algorithm considers all ICD-9 codes on their records prior to their entry of DX_i. Any diagnosis code that occurred in at least 85% of the case (positive) patients became a candidate prerequisite diagnosis for DX_i. To prevent controlling for ICD-9 codes that describe

healthcare encounters, DDR does not consider a specific subset of very general ICD-9 codes (see Table 9.4). We found that these general (non-diagnosis-specific) algorithmic refinements were largely effective, but a small number of the pregnancy complication codes failed to have pregnancy as a prerequisite diagnosis. Therefore, we manually added ICD-9 V22 (Normal Pregnancy) as a prerequisite diagnosis for all pregnancy complication codes (ICD-9 630-679).

Predicting pregnancy-related complications

Case-control matching for pregnancy complications required additional consideration beyond the standard date of birth, sex, 'rule of two', and DDR-control that all of the other codes received. To ensure cases and controls are in similar stages of pregnancy, we required that their pregnancies must have begun within two weeks of one another. The beginning and end dates of a pregnancy, which we use to define the 'pregnancy era', were determined via the use of break point analysis using the 'segmented' package for R (Muggeo, 2003, 2008). Across all patients with two or more V-22 (Normal Pregnancy) ICD-9 codes, the time gaps between adjacent codes were calculated. These times gaps were then used by break point analysis (Kuang et al., 2016) to calculate a threshold for determining if two adjacent codes in a patient's record belonged to the same pregnancy. Breakpoint analysis returned a threshold of 84 days, which is the maximum amount of time two adjacent pregnancy codes on a given patient's record can be spaced from one another and still belong to the same pregnancy era. This threshold value was used to determine the unique pregnancy eras, and their associated start dates, for each patient with two or more V-22 ICD-9 codes. Finally, a case-control pair was matched with the additional constraint that their pregnancy eras began within 14 days of one another. This enforces the simple notion that case-control pairs should roughly be in the same stage of pregnancy.

Truncation date analysis

The experimental results presented in Figure 4.1 were generated via kernel density estimation (KDE) (Parzen, 1962) using the function and parameters detailed in Table 9.2. For these results, we do not show confidence intervals on the density estimates of the AUC distributions as they are very tight. The maximum achievable 95% confidence interval with 500 cases and 500 controls is AUC $\pm$0.036, and with 5,000 cases and 5,000 controls is AUC $\pm$0.011 (see the simulated prospective study analysis section for details on calculating AUC confidence intervals). For this reason, confidence intervals are not included in Figures 4.1 or 4.2.

High-throughput construction of models

In order for models to be constructed in a reasonable amount of time, we used the University of Wisconsin âŁ“ Madison's HTCondor framework (Thain et al., 2005). HTCondor is an open-source high-throughput computing infrastructure that distributes work among many execute nodes running in parallel. We used HTCondor version 8.5.1 which allows for encryption of data transferred to and from worker nodes. Since a separate model is built for each ICD-9 code, pan-diagnostic machine learning can easily be partitioned into parallel tasks by assigning a separate machine to build a predictive model for each diagnosis code. Our term "high-throughput machine learning" acknowledges our intellectual debt to the computing philosophy of HTCondor and the high-throughput computing environment provided by UW-Madison's Center for High-Throughput Computing (CHTC).

Simulated Prospective Study

Simulated prospective study design

We simulated a prospective study by randomly selecting a test cohort of patients and forming predictions across all diagnoses for these patients over a one-year period. We chose to perform our study during the calendar year of 2014 as this

maximized the amount of usable training data for building predictive models. A test cohort was generated by randomly selecting 100,000 patients who had at least one contact with the Marshfield Clinic in the calendar year of 2013 and were not known to be deceased at the end of 2013. This was done to simulate how a health system would perform a prospective study, by first identifying a cohort of active patients to follow.

Two sets of ICD-9 code models were built using pan-diagnostic machine learning with a 1-month prediction window and a 6-month prediction window. These models were constructed in the same fashion as the models built for the pan-diagnostic machine learning experiments. Training data for each model was produced by randomly selecting a set of patients using 1:1 case-control matching, as well as adhering to the prerequisite diagnoses established by DDR. Additionally, no patients present in the testing cohort were a part of the training data. All training data following the calendar year of 2013 were truncated to prevent leakage of any information.

For each ICD-9 code model, a subset of the testing cohort was selected based on eligibility criteria, to avoid overly-optimistic accuracy estimates from predicting for patients for whom such predictions are trivial to make correctly. The eligibility criteria were determined by the case patients chosen for training the model and included the following: 1) the test patient must have all DDR prerequisite diagnoses, 2) the test patient must have been aged between the 1st and 99th percentile of ages of the training case patients, and 3) the test patient must have been of the same sex as the majority of the training case patients if there was a 99% or greater proportion of one sex. This ensured that the results of our models were not artificially elevated by testing on patients who would not have been considered for this diagnosis in a healthcare setting (e.g. prostate cancer for women, or Alzheimer's disease for an adolescent). Each eligible test patient received a score from the model as their predicted value and their true value was indicated by entries on their record during the year of 2014.

Simulated prospective study analysis

The results of the prospective study yielded, for each modeled ICD-9 code and the subset of the testing cohort eligible for the model, the risk scores as predicted by the model on this subset, and the corresponding ground truth indicating if a given patient received the predicted code during the calendar year of 2014. For each diagnosis code, we required that there be sufficient patients to construct both a 1-month and 6-month predictive model. For a given patient and ICD-9 code, the risk for that patient was calculated as the maximum of the risks predicted by the 1-month and 6-month models. In this way, we roughly predict risk in the 1-year window of our study. Like the analysis performed for the experiment shown in Figure 4.1, we use KDE to visualize the distribution of model AUCs in the simulated prospective study. However, the models produced in the first experiment had a minimum of 500 case and 500 control patients each. That amount of data is large enough to produce very good estimates of the AUC for each model. In the prospective study, there were significantly less positive patients for each ICD-9 code, with the majority of models having less than 10 patients who received the code during 2014. Fewer patients, in combination with heavy class skew, led us to much less tight estimates of AUC for each model; hence for the prospective trial, we also show 95% confidence intervals of AUC for each model. To compute these confidence intervals, we first calculated the standard deviation using the formula (Bamber, 1975) in Equation 4.1, where n_p and n_n are the number of positive (case) and negative (control) patients respectively, and AUC represents the AUC of a particular model.

$$\sigma = \sqrt{\frac{AUC(1-AUC) + (n_p - 1)(P_{xxy} - AUC^2) + (n_n - 1)(P_{xyy} - AUC^2)}{n_p n_n}} \tag{4.1}$$

$$P_{xxy} = \frac{AUC}{2 - AUC} \qquad P_{xyy} = \frac{2AUC^2}{1 + AUC}$$

We then computed the upper and lower bounds of the two-sided 95% confidence interval as AUC $\pm z_{.025}\sigma$ where z_α was calculated via the percent point function for the normal distribution N(0,1). As AUC can only take on values between 0 and 1, bounds were truncated at 0 and 1: computed lower bounds below 0 were set to 0, and computed upper bounds above 1 were set to 1. We then produced two additional KDE distributions of the lower and upper bound AUC values. Note that some of these confidence interval densities, in addition to the density of AUC scores, have a small mass outside of the 0,1 range due to kernel density estimation, specifically the use of a Gaussian kernel to estimate the densities. This confidence interval calculation requires that the number of positive and negative examples not be too small or else the AUC may not be normally distributed. For this reason, we required each diagnosis code to have a minimum of 30 case and 30 control patients in the study cohort to be included in the results of this work. We retained a total of 2,538 models that contributed to the results presented in Figure 4.3A, and 2,130 models that for Figure 4.3B.

Figures 4.3A and 4.3B differ based on the inclusion (A) or exclusion (B) of so-called repeat diagnoses. In Figure 4.3B we predict a diagnosis for a patient only if that patient has never had this diagnosis on their record. In this way we are performing a similar task to our original experiments presented in Figures 4.1 and 4.2. However, in Figure 4.3B we predict an outcome for a patient even if they have had that diagnosis before. There are many acute diseases for which an additional code corresponds to a unique event separate from previous entries; however, there are also many other diseases for which this is not true. Including only first diagnoses may be somewhat pessimistic for some diseases, but including repeat diagnoses for all diseases is certainly overly optimistic. We believe the true aggregate performance falls somewhere between the two.

In-depth analysis of high impact diseases

In Figure 4.4 we present an in-depth analysis of the models produced for three high impact diseases. The results for each model included only patients who met

the model's eligibility criteria and had no entries of the ICD-9 code prior to 2014. For each disease, we generated a modified Kaplan-Meier curve, a ROC-curve, and a PR-curve. The modified Kaplan-Meier curve details the relationship between time since prediction (start of 2014) and the fraction of patients yet to receive the diagnosis. The ROC-curve depicts the relationship between the false positive rate and the true positive rate as the threshold is varied. The PR-curve shows the relationship between precision (positive predictive value) and recall (sensitivity, or true positive rate) as the threshold is varied.

Additional Details

Additional software tools used

Our pan-diagnostic machine learning code also makes use of NumPy (Van Der Walt et al., 2011) and Pandas (McKinney, 2010). Figures were constructed using matplotlib (Hunter, 2007) and seaborn (Waskom et al.).

4.3 Results

Prediction quality depends on disease category and time window

We demonstrated the efficacy of the pan-diagnostic machine learning pipeline on the tasks of predicting diagnosis risks at the seven different time windows. The prediction window of 1 month is achieved by truncating all training data following the date 1 month prior to the case patient's diagnosis; each control patient for a given ICD9 code belongs to a case-control pair and has his/her data truncated at the same date as does the case patient. An analogous process is executed for the six other prediction time windows. We observed mean AUCs ranging from 0.803 ± 0.062, across the 3,586 1-month models, to 0.524 ± 0.028 across the 3,288 20-year models. Fewer models were constructed at the 20-year window than at the 1-month window. Because we required a minimum number of case-control pairs that all have data prior to the truncation window, some models had sufficient patients for the 1-month

window but not the 20-year window. We use KDE to visualize and compare the distributions of model performances when predicting at the several time intervals (Figure 4.1). Additionally, we investigated how the predictive accuracy varies across and within the fifteen different highest-level chapters, or diagnosis categories, of the ICD hierarchy (Figure 4.2), using the 1-month and 6-month prediction time windows.

We see in Figure 4.1 that the majority of diagnoses (with sufficient data) can be predicted 1-month in advance with an AUC of 0.8 or greater. We note an inverse relationship between the length of the prediction window and the quality of the model. This observation is likely owed to the decrease in number of patients available for long term predictions, the smaller amount of data these patients have as we necessarily have access to less of their records, and the importance of a patient's recent health state on their immediate future. Interestingly, there are some diagnoses which can predict with AUC at or above 0.7 even 20 years in advance. These diagnoses were largely ocular disorders and congenital anomalies.

Encapsulated in Figure 4.1 are the AUC scores of some 24,462 models across the 7 different truncation windows. While this figure provides a high-level understanding of the general trends of how predictive efficacy varies with time, and while Figure 4.2 delves deeper in presenting how individual diagnostic chapters vary from one another, there is a tremendous opportunity for exploring the results of this work in greater detail. For example, each random forest model has feature importance (Breiman, 2001) values corresponding to an estimate of how valuable a particular health event is in predicting a diagnosis. These variable importance values might be used to better understand both individual diseases and how different diseases relate to one another. Furthermore, a more fine-tuned approach could be taken to the analysis by exploring a subset of diagnoses or how the prediction of a single diagnosis code changes over time. Due to the wealth of opportunities for additional research, we publish alongside this work a dataset containing the AUC and feature importance values for all 24,462 models contributing to Figure 4.1. We believe that this dataset will be both a source for additional research and a baseline of comparison for other prediction strategies in future research. We note

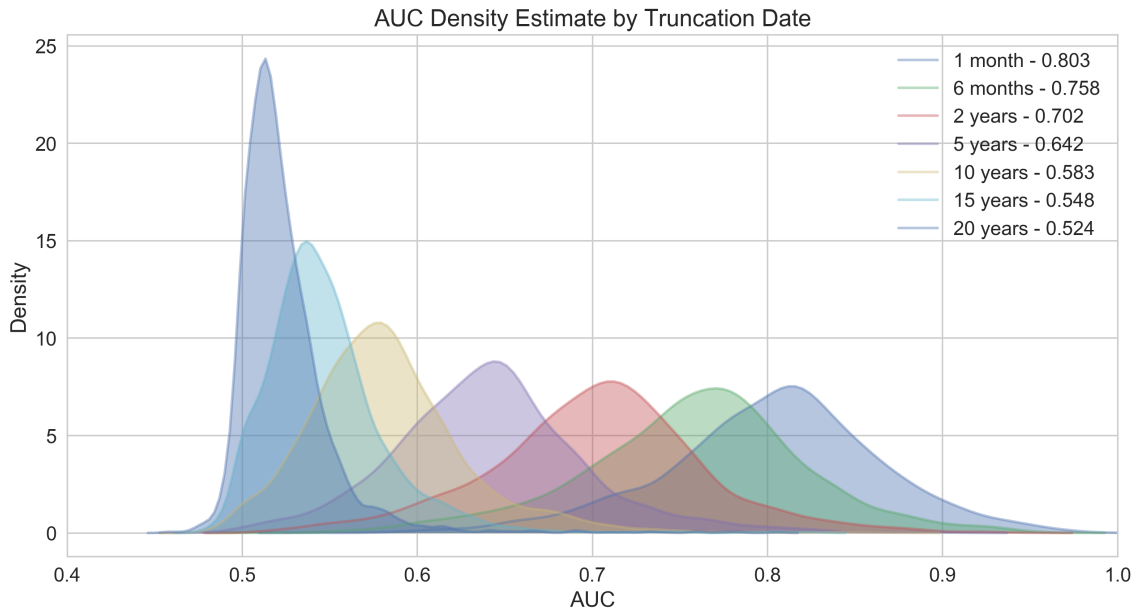


Figure 4.1: Comparison of Kernel Density Estimate (KDE) of AUC Distributions by Truncation Date. The shaded regions show the KDE distributions of AUC for the models built to predict ICD-9 codes at various time intervals ranging from 1 month to 20 years prior to first diagnosis. Note the significant performance increase attributed to an additional data leading up to diagnosis. Additionally, we find it of interest that these distributions are approximately normal. Some of the distributions, in particular those predicting far in advance, have a heavy right tail stretching well into AUCs of 0.7 and greater.

that because the random forest algorithm is stochastic that the feature importance values are as well (that is they can vary between runs). However, with a sufficient number of trees, such variance is minimal. Nevertheless, it is critical to inspect the features returned as a sanity check on the model, and thus in Table 4.1 we present top feature importances for three models predicting common health conditions: acute myocardial infarction, lung cancer, and influenza. We find that these models are largely using reasonable features matching known risk factors for their respective conditions.

Table 4.1: Feature importance values for three common diseases. We present the top 10 unique features for each model, i.e., if a feature was useful in more than one time window, it was collapsed into a single feature for clarity. Note that while we controlled for age, sex, and date of birth, these features are consistently top predictors and they still have value through interactions which other features. Features seen in the acute myocardial infarction model match known risk factors. Of interest is the "Procedure Cytopathology, Pap Smear" feature in lung cancer; we believe this feature is useful as it is a proxy for sex and also is a procedure only performed at specific ages. Similarly, we believe the ICD-9 367, seen in two of the models, may act as a useful proxy for age as eyesight worsens with age. The influenza model shows a mix of respiratory related features and features suggesting regular contact with the healthcare system (office visits and blood collection). We believe that features suggesting greater contact with the healthcare system may be used to differentiate sicker patients from healthier patients, which could indicate a greater risk for influenza.

Rank	410.4 Acute Myocardial Infarction (AUC 0.703)	162.9 Lung Cancer (AUC 0.728)	487.1 Influenza (AUC 0.836)
1	Date of Birth	Date of Birth	Date of Birth
2	Age	Age	Age
3	ICD-9 410.9 Unspecified Essential Hypertension	ICD-9 305.1 Nondependent Tobacco Use Disorder	ICD-9 465 Acute Upper Respiratory Infections
4	ICD-9 414 Ischemic Heart Disease	ICD-9 305 Nondependent Abuse of Drugs	Procedure Office Visit
5	ICD-9 305.1 Nondependent Tobacco Use Disorder	ICD9 496 Chronic Airway Obstruction	ICD-9 V72 Special Investigations and Examinations
6	Sex	ICD-9 518 Other Diseases of Lung	ICD-9 780 General Symptoms
7	ICD-9 305 Nondependent Abuse of Drugs	Sex	Procedure Routine Venipuncture Collection
8	ICD-9 786 Respiratory System and Other Chest Symptoms	ICD-9 786.6 Swelling, Mass, or Lump in Chest	ICD-9 462 Acute Pharyngitis
9	ICD-9 367 Disorders of Refraction and Accommodation	ICD-9 518.8 Other Diseases of Lung	ICD-9 490 Bronchitis
10	ICD-9 786.5 Chest Pain	Procedure Cytopathology, Pap Smear	ICD-9 367 Disorders of Refraction and Accommodation

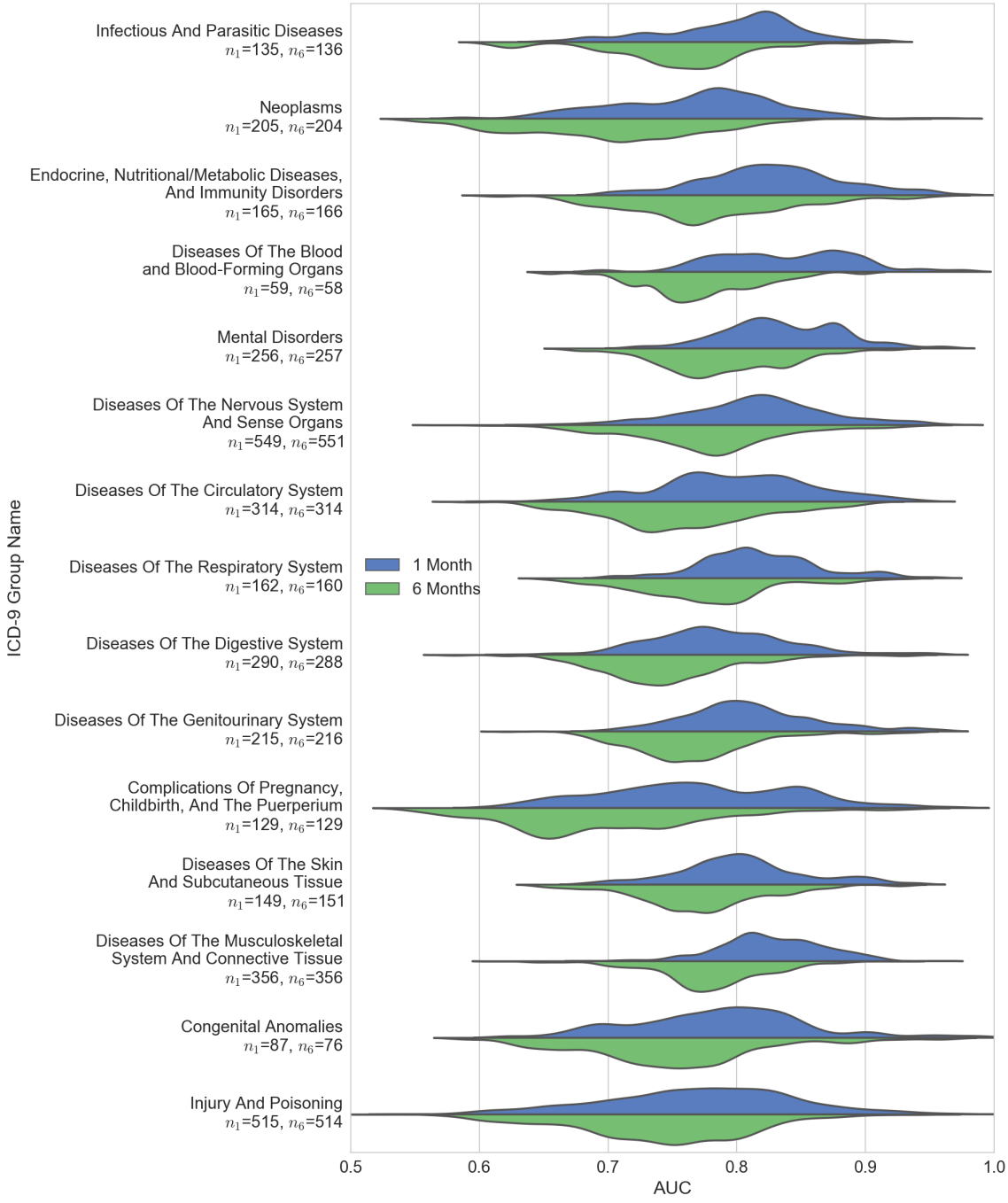


Figure 4.2: (Continued on the following page.)

Figure 4.2: Split Violin Plot Comparing Kernel Density Estimates of AUC Distribution by Truncation Date and ICD-9 Diagnosis Group. Across each diagnosis group a comparison of KDE distributions of AUC is shown for censor dates of 1 and 6 months in the blue and green curves respectively. The number of codes predicted at 1 month and 6 months are listed respectively. The largest decrease in performance is among pregnancy complications where the mean AUC drops from 0.772 ± 0.075 at 1 month to 0.697 ± 0.077 at 6 months. Additionally, note the bi- and sometimes tri-modal nature of distributions, suggesting that many of these categories may have interesting clusters within them.

Translational Validation via Simulated Prospective Study

In addition to evaluation by cross-validation in our case-control samples, to better estimate the accuracy of pan-diagnostic machine learning if it were employed in a clinical setting, we performed a simulated prospective study. The goal of this experiment was to gain insight into the potential efficacy of the models if used in a live healthcare setting, by simulating the translation of our pan-diagnostic risk prediction system.

In our initial pan-diagnostic machine learning experiments we intentionally use a one-to-one case-control matching scheme. Because the lifetime incidence rates for diseases vary greatly from disease to disease, a 1-to-1 ratio can skew certain evaluation metrics such as precision and recall. Therefore, we utilized a truly random sample in our simulated prospective study to demonstrate results without case-control matching. Our prospective study ran during the calendar year of 2014 and followed 100,000 randomly selected patients who had visited the Marshfield Clinic at least once during 2013 and were alive as of the study start (see Simulated prospective study design in Methods and Materials).

The prospective study assessed the predictive quality under two different paradigms, given the complexities in disease mechanisms. These paradigms addressed first diagnoses vs. repeat diagnoses. Some illnesses may be coded multiple times over a patient's life, but each diagnosis may be independent of one another (e.g., influenza), while other, chronic diseases may be coded multiple times as part of disease treatment (e.g., diabetes). We see higher performance when including repeat diagnoses as compared to the first entry of a code, illustrating the higher

difficulty of predicting risks for first-time diagnoses. Figure 4.3 shows a detailed view of the distribution of model efficacies when including or excluding repeat diagnoses. Only models corresponding to diagnosis codes with at least 30 case and 30 control patients in the 100,000-patient cohort were included in the prospective study evaluation. This minimum patient requirement was put in place as our evaluation metric, AUC, becomes unstable when one class has a small number of samples. For the task of predicting new diagnoses, we find that the prospective study shows a slight decrease in performance as compared to the models showcased in our initial experiments shown in Figure 4.1. In our initial experiments with 1:1 case-control matching we achieved a mean AUC of 0.702 predicting diagnoses 2 years in advance. In the simulated prospective study, we see a mean AUC of 0.697 predicting over the course of a year. One potential explanation for this decreased AUC is the inclusion of patients who did not return during the study year in the evaluation data set. We included these patients in the evaluation set as their exclusion would provide a biased view of the results.

Limitations of our Experiments

While receiver operating characteristic (ROC) curves are arguably the most common summary of predictive ability in both machine learning and medicine, a weakness (and a strength) is that they are not sensitive to disease incidence, or the natural "class skew" of cases to controls. In many domains with strong class skew, such as information retrieval and web search, precision-recall (PR) curves are used instead. Nevertheless, PR curves are less appropriate for comparing the results of many disease risk models at once because their different class skews can artificially make one disease look easier to predict than another. But for any one disease at a time, they can provide useful insights, so we explore PR curves and other metrics further in the next subsection*, in the context of a few specific diseases of interest.

Another limitation of even our prospective trial is that we do not compare our predictive models with current clinical practice, to see if any of them would currently improve clinical care. A true, randomized prospective trial could provide such a

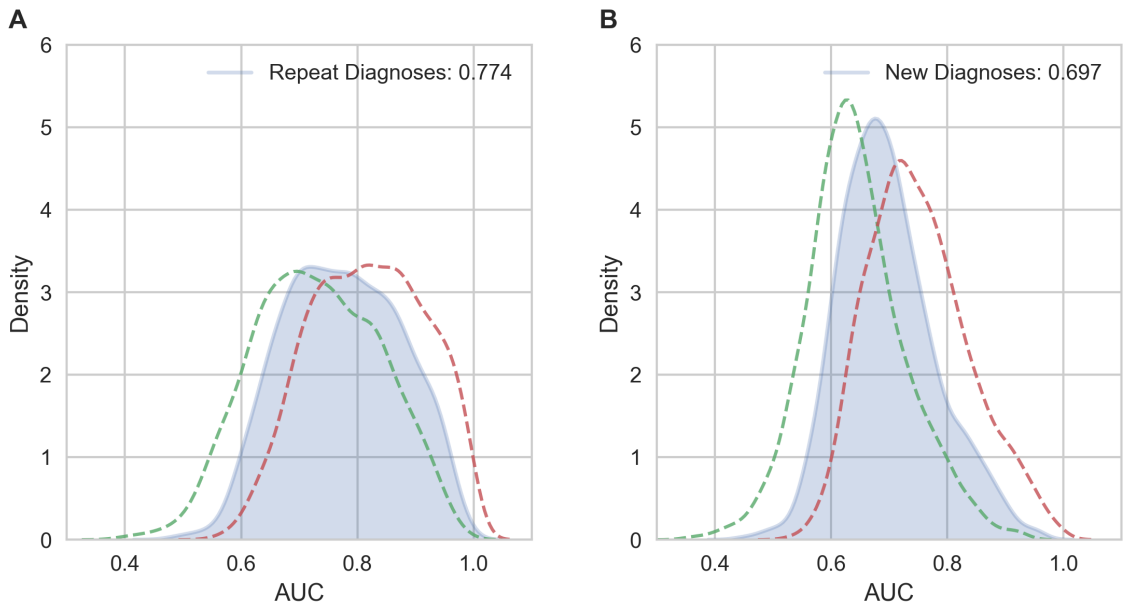


Figure 4.3: Comparison of KDEs of AUC Distributions by Minimum Number of Positive Patients and Inclusion of Repeat Diagnosis. In Figures **A** and **B**, the blue region represents the KDE distribution of AUC scores for models in the simulated prospective study; the green and red dotted lines represent the upper and lower bounds of the 95% CI respectively. **A**, 2,538 models with mean AUC 0.774, 95% CI [0.730, 0.819]; including repeat diagnoses. **B**, 2,130 models with mean AUC 0.697, 95% CI [0.643, 0.751]; including only first diagnoses. While including test patients who previously received the diagnosis provides a more optimistic distribution of AUC density, a conservative estimate of model efficacy would only include evaluation on new diagnoses.

comparison but is beyond the scope of our present work; our present work simply shows how well all coded diagnoses can be predicted across different lengths of time into the future. A few such predictive models have in fact already been translated into the clinic by others, such as the Framingham model for cardiovascular risk (Dawber et al., 1951), the Gail model for breast cancer risk (Gail et al., 1989), and models used in the emergency room such as Charlson comorbidity index (Charlson et al., 1987). These models were constructed by human-selected features and logistic regression, rather than the full EHR and random forests. While the latter approach often yields slightly more accurate models than the former approach (Weiss et al.,

2012; Lantz, 2016), models such as the Framingham risk score and the Gail model include identified input features and features found in text, and also use such data for improved phenotyping, while our modeling approach used only de-identified coded data for privacy reasons (Gail et al., 1989; Dawber et al., 1951). Before a clinical trial and translation, each of our models could almost certainly benefit significantly from being retrained with the inclusion of such data.

A Closer Look at Some Specific Diagnosis Models

In addition to the coarse analysis of our simulated prospective study, we present a deeper analysis of predicting the initial entry of three high impact diagnoses: type II diabetes (Zimmet et al., 2001), chronic kidney disease (CKD) (Levey et al., 2005), and acute myocardial infarction (MI) (Tunstall-Pedoe et al., 1994) in Figure 4.4. Most notable from this analysis is the often tenuous relationship between ROC curve performance and PR curve performance. In learning tasks with high class skew (e.g., relatively rare diagnoses) ROC curves, which are skew independent, can give a much more optimistic picture of the results than do PR curves, which are skew dependent. We would like to aggregate PR areas as we did ROC areas in Fig 1, but because each ICD-9 code has a unique skew, aggregating PR areas across diagnoses is inappropriate (Boyd et al., 2012); therefore, instead we use these three individual PR curves to illustrate the limitations of our models despite their high overall distribution of ROC areas.

4.4 Discussion

This research introduces the concept of high-throughput machine learning and demonstrates its successful application to the task of predicting patient diagnosis risks, which we call pan-diagnostic machine learning. Additionally, we demonstrate an automated procedure for translation of these models into practice wherein appropriate subpopulations are automatically identified for each model. We find that many coded diagnoses can be predicted with reasonable performance — AUC

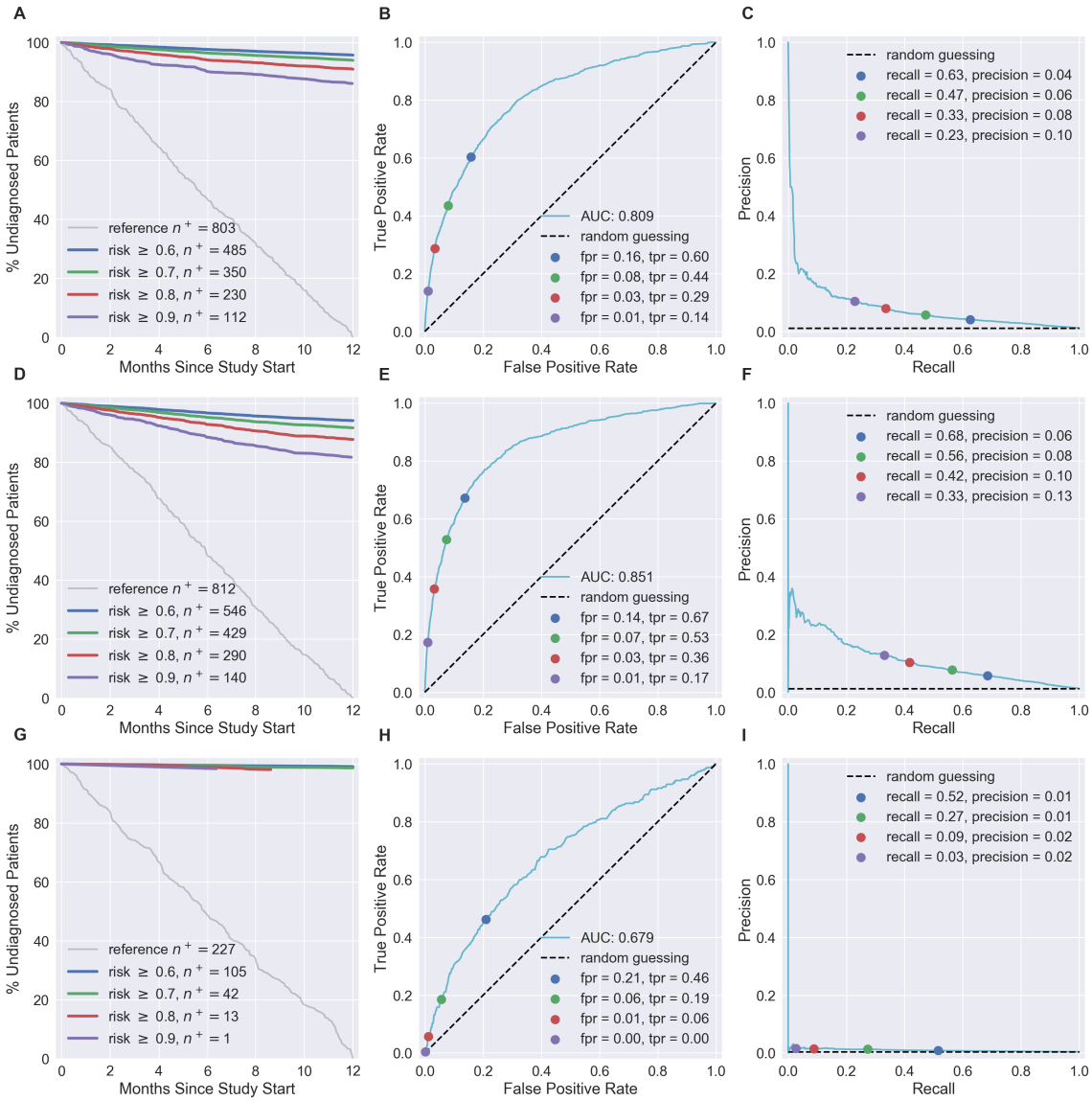


Figure 4.4: In-Depth Analysis of High Impact Diseases. Figures **A-C**, **D-F**, and **G-I** analyze Type II Diabetes (ICD-9 250.00), Chronic Kidney Disease (CKD) (ICD-9 585), and Acute MI (ICD-9 410) respectively. In all figures, blue, green, red, and purple dots correspond to the prediction thresholds of 0.6, 0.7, 0.8, and 0.9 respectively. **A**, Kaplan-Meier Curve for Type II Diabetes depicting the relationship between months since prediction and percent of predicted positives that had not received an entry of ICD-9 250.00 in the study year. **B**, ROC-Curve for Type II Diabetes with overlaid operating points corresponding to risk thresholds identified in **A**. **C**, PR-Curve for Type II Diabetes with overlaid operating points corresponding to risk thresholds identified in **A**. **D**, Kaplan-Meier Curve for CKD. **E**, ROC-Curve for CKD. **F**, PR-Curve for CKD, note the poor performance in comparison to **E**. **G**, Kaplan-Meier Curve for Acute MI. **H**, ROC-Curve for Acute MI. **I**, PR-Curve for Acute MI.

greater than 0.8 — at least one month prior to diagnosis as shown in Figure 4.1. As we attempt to predict diagnosis risks further in advance, we become less confident in the quality of our predictions. We do, however, note that even 20 years prior to the first diagnosis, we can still predict with some efficacy the first entry of diagnosis codes categorized as congenital disorders (ICD-9 320-389) and diseases of the nervous system and sense organs (ICD-9 740-759). Amongst the 43 models predicting congenital anomalies, 10 were above 0.6 AUC (23%) at this timeframe; amongst the 506 models predicting diseases of the nervous system and sense organs, 30 were above 0.6 AUC (5.6%). Many of the highest performing models, in both congenital and nervous categories, were related to ocular disorders.

Of additional interest are the differences in predictive performance amongst models when stratified by the disease categories of the ICD-9 hierarchy as presented in Figure 4.2. The multi-modal distributions of some of these categories suggest that some of these categories can be further partitioned by predictive quality, such as those models predicting mental disorders or pregnancy complications at the 1-month window. We believe there are additional insights to gain from further analysis of these diagnoses. With a simulated prospective study, we estimate the translational efficacy of a pan-diagnostic machine learning pipeline in the healthcare setting. We note that the performances of these models under this evaluation methodology are slightly pessimistic as compared to the performances achieved in a 10-fold cross validation.

We introduce a new medical dataset consisting of the AUC scores and feature importance values of the 24,482 predictive models belonging to Figure 4.1. We believe that this dataset will be of interest both as a baseline for comparison for future work that predicts diagnoses and as a data source that can be mined for relationships amongst diagnoses and health record events. Many of our open questions such as which diagnoses cluster together on predictive quality and how predictive efficacy changes over time can be investigated with this dataset.

The limitations of this research fall largely into two categories: those that are opportunities for improvements to various steps in our high-throughput machine learning pipeline, and those that arise from the aggregation of unique prediction

tasks both in model building and evaluation. As the focus of this work is on the introduction of high-throughput machine learning and generating a baseline for the performance of predicting diagnosis code risks, we believe there are some areas of our pipeline that, while functional, could be improved to produce even higher quality models. For example, in place of rule-of-n for case-control definition, more advanced forms of electronic phenotyping (Peissig et al., 2014) could be used. Future work could additionally incorporate existing disease ontologies to group together ICD-9 codes. Doing so would both provide computational benefits and aggregate codes into disease-specific models. Because our pipeline is flexible and modular, these enhancements could be easily created and incorporated into future research. Furthermore, it would be interesting to compare the results of this work to a multi-label learning task wherein a single model outputs risks for all diagnoses. This multi-label approach would allow us to leverage differences between diagnoses that co-occur versus those that occur independently. We additionally note that some limitations arise when creating general rules for the construction and evaluation of thousands of models. For example, by picking a single prediction time window, e.g. 6-months, there are some diagnoses for which we are losing valuable data and others for which we would have preferred an even more greatly truncated window. For example, truncating by 6-months may be too strict for the prediction of an acute disease such as influenza, but may be too short of a window in the case of predicting a disease whose symptoms mount gradually leading to diagnosis, such as Parkinson's disease. While we present a variety of truncation windows between 1-month and 20-years for this reason, we do not have a data-driven method to automatically determine the appropriate truncation window for a given diagnosis. A final limitation of this work is the difficulty in analyzing the quality of any particular model relative to the current quality achieved at a clinic. Because we do not have baseline AUC values for how well healthcare providers can predict these diagnoses, it is impossible to make claims on the impact of any particular model. It is possible that a model with an AUC of 0.6 for one disease would be of more value than a model with an AUC of 0.8 for another disease if healthcare providers currently cannot predict the first disease at all, but can perfectly predict

the second. This limitation is not unique to our work and it emphasizes the importance of clinical trials when considering the translation of any decision support tool into the clinic. Finally, we note that because our dataset is comprised of patients belonging to the North/Central region of Wisconsin, the models learned in this work have a population-based bias. For example, because model quality is affected by availability of training data, our models are sensitive to the prevalence of the modeled diseases in our particular population. Moreover, generalizing these results or applying these models to a population with a different demographic make-up would be inappropriate.

In this chapter, we demonstrate that a single system can predict risks for thousands of different coded diagnoses. Perhaps the most important contribution of this study is to provide an initial baseline for how accurately the wide range of disease phenotypes can be predicted from EHR data. We believe this is a step toward a much broader incorporation of machine learning-based prediction into clinical care.

4.5 Summary

In this chapter we introduced and explored the use of high-throughput machine learning methods to learn models for thousands of diseases and multiple time points. We showed that by predicting thousands of diseases at once we can reveal a rich landscape of prediction quality across different levels of the diagnostic hierarchy. We believe that there are several exciting avenues for the extension of high-throughput machine learning to additional data types, prediction tasks, and applications. For this reason, the remaining chapters of this thesis are all organized as distinct extensions or applications of high-throughput machine learning.

Learning models using data from multiple hospital sites

Whereas in this chapter we build our pipeline on a single set of EHR data, there are many reasons for which we may wish to build models using data from multiple

sites. Even with the millions of patients in the Marshfield Clinic EHR system, there are some rare conditions for which there are so few examples it can be challenging to accumulate enough data to learn a strong model. In these cases we may wish to incorporate additional data from other healthcare systems. However, there are both privacy and business reasons that hospitals may not wish to openly share their patient data with one another. Therefore, in Chapter 5 we present a cryptographic solution to such a challenge in which multiple hospitals can collaboratively learn a model in a secure fashion that does not require the sharing of data.

Discovery of new medical knowledge using high-throughput machine learning feature importances

When applying machine learning to healthcare data it is often highly important to know that the model not only performs well but also that it is using sensible features to make its decision. For this reason, we showed in this chapter the feature importances for a handful of these models. However, if a feature is heavily relied on but not congruent with current medical knowledge it may be that the model has discovered a hitherto unknown relationship. Such relationships may provide valuable medical insight. Hence, in Chapter 6 we explore an approach for identifying candidate epidemiological relationships between diagnoses and laboratory tests.

Evaluating models for multi-class classification tasks

In this chapter, we constructed thousands of predictive models for two-class classification problems that were predicting whether or not a patient had a particular disease. However, there are some conditions that share the same constellation of symptoms for which it would be valuable to construct a multi-class classification model. These models could answer questions such as "Does this patient have Parkinson's Disease, Alzheimer's Disease, or no disease?" However, the evaluation of such multi-class models cannot use the standard AUC measure which is only valid for two classes. Existing multi-class performance measures have key flaws such as failing to properly score perfectly separated examples, interpretability dif-

ficulties, and combinatorial computational time complexities. For this reason, in Chapter 7 we introduce a new multi-class performance measure that adheres to those critical properties of AUC. We apply this new measure for the multi-class task of predicting the diagnosis of one of seven different digestive cancers.

Co-authorship and Acknowledgements

The work presented in Chapter 4 is the result of a collaboration with multiple researchers. This work was completed with the following authors and affiliations in order of their appearance as co-authors: R. S. Kleiman[1], P. S. Bennett[1], P. L. Peissig[2], Z. Kuang[1], R. L. Berg[2], S. J. Hebbring[2], M. D. Caldwell[2], C. D. Page[1,3]; [1]Department of Computer Sciences, University of Wisconsin âŁ“ Madison; [2]Marshfield Clinic Research Institute; [3]Department of Biostatistics and Medical Informatics, University of Wisconsin. This work was supported by the NIH BD2K grant: U54 AI117924, the NLM grant: R01LM011028, and the NLM training grant: 5T15LM007359.

5 PRIVACY-PRESERVING COLLABORATIVE PREDICTION USING RANDOM FORESTS

Increasing the available training data for a machine learning algorithm improves its overall generalization performance. However, the EHR data from a particular healthcare system only grows as the services are needed by the patients. This implies that any significant increase in training data must necessarily come from the inclusion of additional EHR data from other healthcare systems. However, such efforts are met with many privacy and business concerns that discourage separate healthcare systems from openly sharing data with one another. Hence, there is a need for machine learning algorithms that can learn from siloed data without ever sharing the data amongst the parties. In Chapter 5 we introduce such an approach by constructing a privacy preserving random forest that utilizes cryptographic tools to ensure security.

5.1 Introduction

In this work, we notice that for *ensemble methods*, for which the learned model is formed by a set of more basic models and the prediction for a new instance is computed by blending together the basic predictions, there can be an easier and more efficient solution that needs only one system; we refer to this solution as the *"locally learn then merge"* approach. Each entity with a local data silo (*i.e.*, providers) can train its own local model M_i, and then the prediction given by these models can be merged at the moment when the scoring for a new (eventually private) instance is computed. That is, a user with input $\mathbf{x}$ gets $y = \Phi(M_1(\mathbf{x}), \ldots, M_t(\mathbf{x}))$ for a specific merging function Φ. Here $M_i(\mathbf{x})$ indicates the prediction of the local model M_i for the instance $\mathbf{x}$. In this approach, privacy concerns coming from data sharing in the training phase are not present since, clearly, local training does not require data sharing. Moreover, there is no overhead for the training phase (this is run as in the standard machine learning scenario), while the final prediction can benefit from

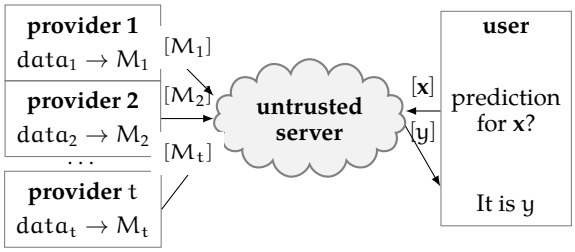


Figure 5.1: Overview of the new "locally learn then merge" approach in the cloud model. The providers upload the encrypted models to the server and then go off-line. The server is on-line to answer to the prediction requests of the user.

merging the local predictions via the function Φ. On the other hand, accuracy loss (with respect to a model learned from the merged data) and information leakage can happen during the merging/scoring phase. In particular, a challenge remains with this simple and elegant approach to collaborative machine learning: if we want to guarantee model and user's input privacy (*i.e.*, the user learns y and no other information on the models M_i, the providers learn nothing about $\mathbf{x}$), then even after the training phase each provider must maintain its own on-line server and communicate with the client and the other providers each time a new prediction is requested. Since in a real-world scenario (*i.e.*, healthcare environment), this requirement can be cumbersome to implement, we design our system in the *cloud model*, where the computation of the prediction from the local models is outsourced to a central server and providers are not required to be on-line during the scoring process (Fig. 5.1). Since we do not require the server to be trusted, each model M_i is sent to the server in encrypted form (*i.e.*, $[M_i]$). Once this is done, the providers (*e.g.*, clinics) can go off-line and when a user (*e.g.*, medical research institution) requires access to the models to compute predictions for new data, the server communicates with it and computes the answer from the encrypted models.

In this work, we specify and evaluate the "locally learn then merge" paradigm in the cloud model for a widely-used ensemble method: random forests. *Random forests* (Breiman, 2001) are among the most accurate and widely-used machine learning ensemble models and are employed across a variety of challenging tasks, including predictive modeling from clinical data (Lantz, 2016), that are character-

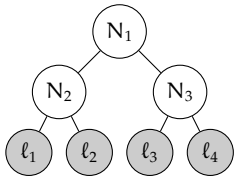
ized by high dimension and variable interactions, or other non-linearities in the target concept. A random forest is a collection of simple decision trees. By the use of different trees, a random forest can capture variable interactions without the need for the learner to know or guess all relevant interactions ahead of time in order to represent them with new variables (interaction terms); by their ensemble nature, random forests effectively reduce the over-fitting often observed with ordinary decision tree learning. A less-recognized advantage of random forests is that they can be learned in a distributed manner. In this circumstance, separate random forests can easily be learned locally by entities with data silos, and then the prediction for a new instance is computed as the arithmetic mean of the predictions of all the trees in the locally trained random forests (*i.e.*, the merging function Φ is the arithmetic mean). We design a system implementing this approach for random forest using standard and fast cryptographic primitives. While our scheme is efficient even for forests of many trees, not surprisingly its run-time and communication complexity grow exponentially with maximum tree depth in a forest. Therefore we also provide empirical evidence that across a variety of data sets and tasks, increasing the number of trees can effectively make up for any accuracy or AUC lost by incorporating a stringent limit on tree depth, such as 8 or even 6. *Related Work:*

There is an extensive research that propose *privacy-preserving training* (Brisimi et al., 2018) protocols and few of them focus on training decision tree. After that, Lindell and Pinkas (2000) presented a system for two data-providers, Xiao et al. (2005) and Samet and Miri (2008) considered the case of more than two providers. While the former works consider horizontally partitioned data, another line of work (Vaidya et al., 2008; de Hoogh et al., 2014) assumes data that are vertically partitioned among two or more data holders. Lastly, Vaidya et al. (2014) proposed a method to learn and score *random trees* in a privacy preserving manner. Like our approach, their approach requires encryption-based collaboration to make predictions. Unlike our approach, their approach also requires interaction and collaboration at training time. One party proposes a random tree structure, and all parties must contribute information to the distributions at the leaf nodes. In our approach, learning is

completely independent for each party, and hence training is much faster. An advantage of Vaidya et al. is the ability to also address vertically partitioned data. *Privacy-preserving scoring* protocols for decision trees have been designed using different cryptographic tools (*e.g.*, levelled homomorphic encryption (Bost et al., 2015), LHE (Tai et al., 2017), secret-sharing (De Cock et al., 2017), OT-channels (Joye and Salehi, 2018)). Backes et al. (2017) improved and extended to random forests the algorithm presented by Brickell et al. (2007) Another line of research focuses on constructing *differentially private decision trees*, see for example the work of Jagannathan et al. (2012) and Rana et al. (2015). Our approach is orthogonal to differential privacy since we consider a different threat model.

5.2 Methods: Decision Trees (DTs) and Random Forests (RFs)

Decision trees (DTs) are a nonparametric machine learning model used for both classification and regression problems. While there are a myriad of algorithms for constructing DTs, we focus here on describing the model representation of the scoring procedure. A decision tree (DT), T, can be viewed as mapping a column vector $\mathbf{x} = (\mathbf{x}[1], \ldots, \mathbf{x}[n])^\top$ of features to a prediction value y. In practice, we assume that T is represented as a directed acyclic graph with two types of nodes: *splitting nodes* which have children, and *leaf nodes* which have no children. Moreover, T has a single root node, which is also a splitting node, that has no parents. For an input $\mathbf{x} \in \mathbb{R}^n$, we traverse the tree T starting from the root and reach a leaf. Each splitting node N_i is defined by a pair (j_i, t_i) where j_i is an index



$$P_{2,1}(x_1, x_2) = (x_1 - 1)(x_2 - 1)$$
$$P_{2,2}(x_1, x_2) = (x_1 - 1)(x_2 + 1)$$
$$P_{2,3}(x_1, x_3) = (x_1 + 1)(x_3 - 1)$$
$$P_{2,4}(x_1, x_3) = (x_1 + 1)(x_3 + 1)$$

Figure 5.2: Polynomial representation of a depth 2 complete DT.

in $\{1, \ldots, n\}$ and $t_i \in \mathbb{R}$ is a threshold value. In the root-leaf path, at node i we take the right branch if $\mathbf{x}[j_i] \geqslant t_i$. Otherwise, we take the left one. Thus, each splitting node N_i is associated with the function $N_i(\mathbf{x}) = \mathbf{e}_{j_i}^\top \cdot \mathbf{x} - t_i$ and the value $n_i = \text{sign}(N_i(\mathbf{x}))$ (where $\cdot$ is the standard row-by-column multiplication). Here the vector $\mathbf{e}_i$ is the column vector in $\mathbb{R}^n$ with all zeros except for a 1 in position i and $\mathbf{e}_i^\top$ is its transpose. Moreover, if $x \in \mathbb{R}$, then $\text{sign}(x) = 1$ if $x \geqslant 0$ and $\text{sign}(x) = -1$ otherwise. In this way we traverse the tree and we reach a leaf node. The i-th leaf node is associated with the label ℓ_i, which is defined to be the prediction of the query $\mathbf{x}$ that reaches the i-th leaf (*i.e.*, $y = T(\mathbf{x}) = \ell_i$). The format of the labels $\{\ell_i\}_i$ depends on the specific machine learning problem (regression, multiclass classification or binary classification). In this work, we assume $\ell_i \in [0, 1]$ representing the probability of $\mathbf{x}$ being classified as $+$ in a binary classification problems with labels $\{+, -\}$. The *depth* of a tree is the maximum number of splitting nodes visited before reaching a leaf. In general, DTs need not be binary or complete. However, all DTs can be transformed into a complete binary tree by increasing the depth of the tree and introducing "dummy" splitting nodes. Without loss of generality, here we only consider complete binary DTs. A complete binary tree of depth d has 2^d leaves and $2^d - 1$ splitting nodes. *Random forests* (RFs), proposed by Breiman (2001), are an ensemble learning algorithm that are based on DTs. An ensemble learner incorporates the predictions of multiple models to yield a final consensus prediction. More precisely, a random forest RF consists of m trees, $T_1, \ldots, T_m$, and scoring RF on input $\mathbf{x}$ means computing $y = \frac{1}{m} \sum_{i=1}^{m} T_i(\mathbf{x})$. Let d be the maximum of the depths of the trees in RF, we refer to d and m as the *hyperparameters* of the forest.

Polynomial representation: We can represent a tree using polynomials. Let T be a complete binary tree of depth d, then we associate each leaf with the product of d binomials of the form $(x_i - 1)$ or $(x_i + 1)$ using the following rule: in the root-leaf path, if at the node N_i left turn is taken, choose $(x_i - 1)$ otherwise choose $(x_i + 1)$. We indicate with $P_{d,i}$ the polynomial of degree d corresponding to the i-th leaf. Notice that $P_{d,i}$ contains only d variables, out of the $2^d - 1$ total possible variables (one for each splitting node). We call $\mathcal{I}_{d,i}$ the set of indices of the variables that appears

in $P_{d,i}$ and we write $P_{d,i}((x_j)_{j\in\mathcal{J}_i})$ to indicate this; in Fig. 5.2, $\mathcal{J}_{2,1} = \mathcal{J}_{2,2} = \{1,2\}$ and $\mathcal{J}_{2,3} = \mathcal{J}_{2,4} = \{1,3\}$. Now $T(\mathbf{x})$ can be computed by evaluating the polynomials $\{P_{d,i}((x_j)_{j\in\mathcal{J}_i})\}_{i=1,\dots,2^d}$ on the values $\{n_j\}_{j=1,\dots,2^d-1}$. Indeed, if i^* is the unique value for the index i for which $P_{d,i}((n_j)_{j\in\mathcal{J}_i})) \neq 0$, then $T(\mathbf{x}) = \ell_{i^*}$.

Methods: Cryptographic Tools

A *linearly-homomorphic encryption* (LHE) scheme is defined by three algorithms: The key-generation algorithm Gen takes as input a security parameter and outputs the pair of secret and public key, $(\mathbf{sk}, \mathbf{pk})$. The encryption algorithm Enc is a randomized algorithm that takes $\mathbf{pk}$ and an input $\mathbf{x}$, and outputs a ciphertext, $c \leftarrow \mathsf{Enc}_{\mathbf{pk}}(\mathbf{x})$. The decryption algorithm Dec is a deterministic function that takes $\mathbf{sk}$ and c, and recovers the original input $\mathbf{x}$ with probability 1 over Enc's random choice. The standard security property (semantic security) states that it is infeasible to gain extra information about an input when given only its ciphertext c and the public key. Moreover, we have the homomorphic property: informally, linear functions of encrypted data can be computed without decrypting (*e.g.*, from $\mathsf{Enc}_{\mathbf{pk}}(\mathbf{x}_1)$ and $\mathsf{Enc}_{\mathbf{pk}}(\mathbf{x}_2)$ we can compute $\mathsf{Enc}_{\mathbf{pk}}(\mathbf{x}_1 + \mathbf{x}_2)$ without knowing $\mathbf{x}_1$ and $\mathbf{x}_2$). Efficient instantiations of this primitive are known (Joye and Libert, 2013). In the design of the privacy-preserving system presented later on in this work, we will deploy the *secure comparison protocol* Π_{SC}. The latter (Giacomelli et al., 2018) is a modification of the protocol presented by Kerschbaum and Terzidis (2006) and has the following structure: party 1 has an encryption of an integer a, while party 2 knows the corresponding secret key. They run the protocol and the output is a multiplicative sharing of the sign of a and no extra information about a. In particular, party 1 receives a share $\alpha \in \{-1,+1\}$ and party 2 receives a share $\beta \in \{-1,+1\}$ such that $\alpha\beta = \mathrm{sign}(a)$ and the knowledge of only one share gives no information on $\mathrm{sign}(a)$.

5.3 Results: the Proposed System

In this section we describe our system, where the prediction for a new instance is computed using the RFs trained by different and mutually distrustful parties on their local data silos. We start by describing the role of the parties involved and the security model.

Providers: There are t providers, the k-th one, P_k, has a forest $RF_k = \{T_1^k, \ldots, T_{m_k}^k\}$ with m_k DTs; we assume that the forest hyperparameters (m_k and maximum tree depth d_k) are public values, while the description of the trees is the secret input of P_k to the system. The providers have no output.

Server: The server has no input and no output; it is trusted to handle neither private data nor proprietary models. Its function is providing reliable software and hardware to store encrypted version of the models RF_k and handling prediction request from the user in real-time.

User: Its secret input is an instance $\mathbf{x} \in \mathbb{R}^n$ and the output is the prediction for $\mathbf{x}$ according to all the trees T_j^k (n is public); more precisely, the user's output from the system is $y = \frac{1}{m} \sum_{k=1}^{t} \sum_{j=1}^{m_k} T_j^k(\mathbf{x})$ with $m = m_1 + \cdots + m_k$.

We assume that all the parties involved are honest-but-curious (*i.e.*, they always follow the specifications of the protocol but try to learn extra information about other parties, secret input from the messages received during the execution of the protocol) and non-colluding (*e.g.*, in real world applications, physical restrictions or economic incentives can be used to assure that the server has no interest in colluding with another party). Using the cryptographic tools described before and other standard tools, we design a system where only the user gets to know y and it gets no other information about the private models held by the providers. Moreover, the providers and the server gain no information about the input $\mathbf{x}$. The system we present has two phases: an off-line phase, during which each provider uploads an encrypted form of its forest to the server, and an on-line phase, during which the prediction for a specific input $\mathbf{x}$ is computed by the server and the user. Notice that the off-line phase is independent of the actual input of the user and needs to be executed only once (*i.e.*, when the providers join the system). After

that, the providers can leave the system and the server will manage each prediction request. In particular, for each request, a new on-line phase is executed by the server together with the user making the request (it is possible to have more than one user requesting predictions). Each phase is described below:

Off-line Phase: The goal of this phase is to transmit all the trees to the server, but in encrypted form. That is, the server will know the public hyperparameters of each locally learned forest but have no knowledge about the specific structure of the trees in the forests (*i.e.*, it does not know the indices i_j, the thresholds t_i, or the leaf values ℓ_i). This is achieved by having each provider execute a new *model-encryption procedure* we design (Giacomelli et al., 2018). Using this, the P_k encrypts the thresholds and leaf values and hides the vectors $\mathbf{e}_{j_i}$ using a standard PRF (pseudorandom function); then it sends the encrypted forest to the server; after this P_k can leave the system. We indicate the encrypted forest, which is a collection of encrypted trees, with the notation $\{[\,T_j^k\,]\}_{j=1\ldots,m_k}$.

On-line Phase: For each prediction request, this phase is executed. A user with input $\mathbf{x}$ joins the system sending $\mathsf{Enc}_{\mathbf{pk}}(\mathbf{x})$ to the server. Now, the user and the server run the *tree evaluation protocol* Π_{TE} for each encrypted tree $[\,T_j^k\,]$ of the forests that were uploaded to the server. This protocol returns an additive sharing of $T_j^k(\mathbf{x})$ (*i.e.*, the server gets the share $r_j^k \in [0,1]$ and the user gets the share $s_j^k \in [0,1]$ such that $T_j^k(\mathbf{x}) = s_j^k + r_j^k$ and the knowledge of only one share does not reveal $T_j^k(\mathbf{x})$). Finally, the server sends the sum $r = \sum_{k=1}^{t}\sum_{j=1}^{m_k} r_j^k$ of its shares (one for each tree) to the user, which computes y as $(s+r)/m$, where $s = \sum_{k=1}^{t}\sum_{j=1}^{m_k} s_j^k$ is the sum of the user's shares. The security of our system against a honest-but-curious server follows from the security of the encryption scheme: all the messages received by the server are ciphertexts. Moreover, the user does not learn any extra information about the local models since it does not see the individual predictions (*i.e.*, for each tree the user only see the share s_j^k).

High level description of protocol Π_{TE} (more details in the full version (Giacomelli et al., 2018)): Recall that, given a tree T and an input $\mathbf{x}$, finding the index i^* such that the polynomial P_{d,i^*} evaluates 0 on the values $\{n_j\}_j$ is equivalent to compute $T(\mathbf{x})$ (*i.e.*, $T(\mathbf{x}) = \ell_{i^*}$). Therefore, finding i^* is sufficient in order to then

compute an additive sharing of $T(\mathbf{x})$. In the privacy-preserving scenario, the main challenges in doing this are: 1) First of all, notice that neither the server or the user can see i^* in the clear, indeed knowing the index of the reached leaf can leak information about the inputs and the tree structure (when more than a request is made). We solve this using a simple *tree randomization* stratagem that hides i^* for the user (*i.e.*, the user gets to know i^* for a tree T' equivalent to T but with nodes randomly permuted by the server) and a standard cryptographic tool called *oblivious transfer* that hides i^* for the server (*i.e.*, once that the user gets i^* for T', the oblivious transfer protocols allows it to receive ℓ_{i^*} without revealing i^* to the server); 2) Then observe that neither the server or the user can see the $\{n_j\}_j$ in the clear, indeed also these values can leak information about $\mathbf{x}$ or T. To solve this we use the homomorphic property of the underlying LHE [a], the secure comparison protocol Π_{SC} and an algebraic property of the polynomials $\{P_{d,i}\}_i$. Since each $n_j = N_j(\mathbf{x})$ is a linear function of $\mathbf{x}$, the server can compute $\mathsf{Enc}_{\mathbf{pk}}(N_j(\mathbf{x}))$ from $\mathsf{Enc}_{\mathbf{pk}}(\mathbf{x})$ (assuming that the underlying scheme is an LHE scheme); then the server and the user run protocol Π_{SC}: the server is party 1 with $a = N_j(\mathbf{x})$ and the user is party 2; at the end they know α_j and β_j, respectively and such that $\alpha_j\beta_j = n_j$. However, the value n_j is kept secret. Finally, notice the following: for each $i = 1, \ldots, 2^d$ we have $P_{d,i}((\beta_j)_{j\in\mathcal{J}_i}) \prod_{j\in\mathcal{J}_i} \alpha_j = P_{d,i}((n_j)_{j\in\mathcal{J}_i})$ and therefore $P_{d,i}((n_j)_{j\in\mathcal{J}_i}) = 0$ if and only if $P_{d,i}((\beta_j)_{j\in\mathcal{J}_i}) = 0$. This implies that i^* can be computed locally by the user that knows $\{\beta_j\}_j$.

Complexity of the system in terms of cryptographic operations: During the off-line phase, the providers run the model-encryption procedure to encrypt their models $\mathsf{RF}_1, \cdots, \mathsf{RF}_t$. Assume that RF_k has hyperparameters d_k and m_k, then for P_k the model-encryption procedure costs $\Theta(m_k\, 2^{d_k})$ encryptions. Moreover, P_k sends to the server $m_k\, 2^{d_k+1}$ ciphertexts. The complexity of the on-line phase is dominated by the m repetitions of protocol Π_{TE}. The latter requires $\Theta(n\, 2^d)$ operations ($\Theta(2^d)$ for the user and $\Theta(n\, 2^d)$ for the server) and generates $\Theta(2^d)$ ciphertexts exchanged

[a] A party (different form the server) runs $\mathsf{Gen}(\kappa)$, makes $\mathbf{pk}$ public and safely stores $\mathbf{sk}$. The user needs to authenticate itself with this party in order to get the secret key $\mathbf{sk}$. Notice that the role of this party can be assumed by the user itself or by one or more of the providers.

among the server and the user to score a tree of depth d on an instance with n features. Therefore, the on-line phase has complexity proportional to $n\, m\, 2^d$, where $d = \max_k d_k$. Finally, notice that many steps of our system can be easily *run in parallel*. For example, the m needed instances of protocol Π_{TE} can be executed concurrently.

Discussion: Random Forest (RF) Hyperparameters

Since the depth and number of trees (*i.e.* model hyperparameters) affect the efficiency of our system, we provide here an empirical demonstration that bounding them can be done without adversely affecting the prediction efficacy.

Bounded depth: Typically, during the training phase a RF is grown such that each tree may split in a greedy fashion without concern for the depth of the trees. We provide here an empirical inspection of the effect of bounding the depth to a maximum value d on the efficacy (AUC value) of the learned forest. We utilize the public Kent Ridge Colon-cancer dataset from the UCI repository (reference in Table 5.2) and we looked at various combinations of d and the number of trees in the forest, m. Specifically, we consider values of d in $\{1, 2, \ldots, 28, 30\}$ and 25 different choices of m in $\{1, 5, 10, 25, 50, 100, 200, 300, \ldots, 1900, 2000\}$. For each pair of values, we performed 30 replicates of a RF model construction and evaluation process. For each model, the construction began with choosing a random 70% of the data to serve as training data and the remaining 30% as testing data. A model was then built with the specified hyperparamters and AUC was measured on the testing data. In Fig. 5.3 we present the results of this investigation as a heatmap. For this task even a maximum depth of 6 was competitive with larger depth choices if 300 trees are considered. This suggests that while the standard learning algorithm may greedily grow trees very deeply, the overall performance is not substantially impacted by bounding the maximum depth.

Tuning methodology: Common practice for training machine learning algorithms involves some selection method for determining a choice for the hyperparameters. One standard selection method is a grid-based search wherein a researcher

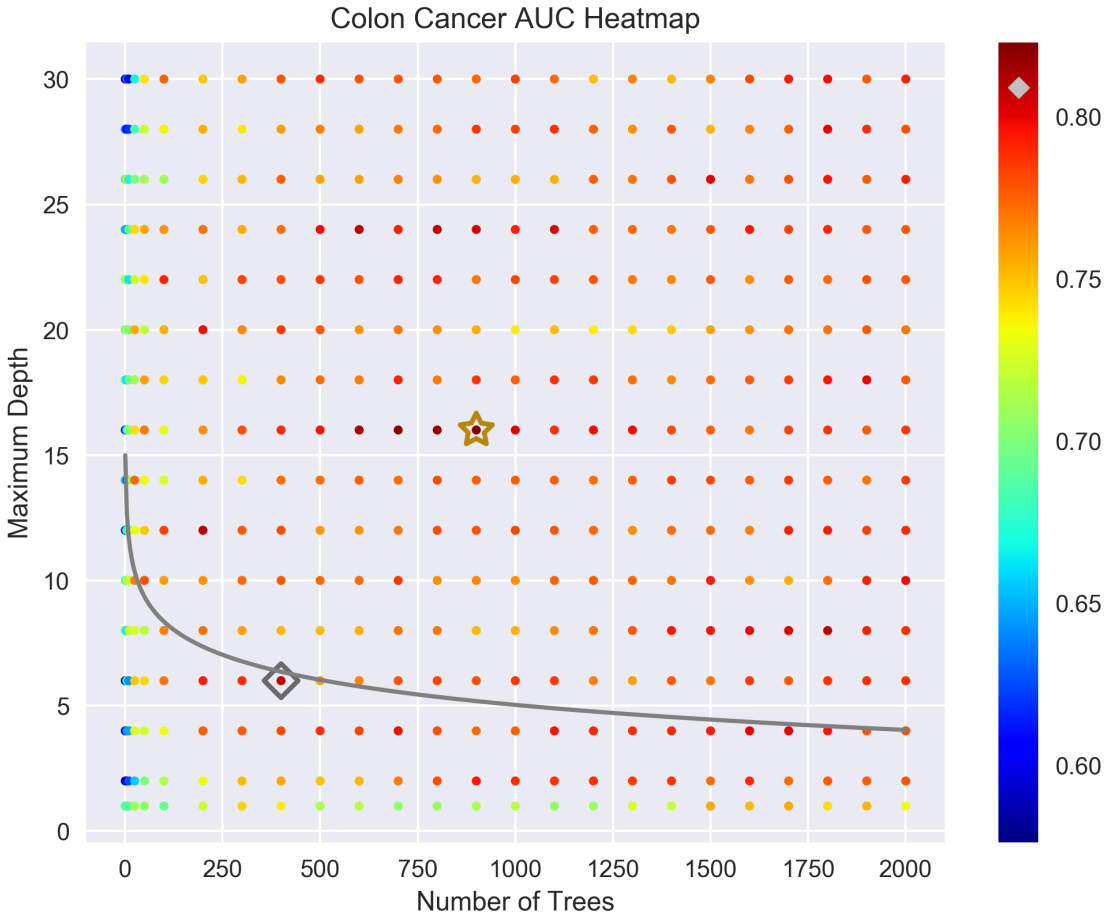


Figure 5.3: Heatmap of mean AUC values for various combinations of d and m. The gray line indicates $m2^d = 2^{15}$. The gold star indicates the best overall combination of d and m (AUC=0.823), the silver diamond indicates the best overall combination constrained by $m2^d \leqslant 2^{15}$ (AUC=0.809). The silver diamond is also on the colorbar indicating the corresponding AUC.

will predefine some set of choices for each hyperparameter and then compute the cross product of these sets and choose the combination that maximized the AUC of the model. For example, for RF, we pick the hyperparameters as $d^*, m^* = \arg\max_{(d,m)\in D\times M} AUC(RF(d,m))$, where $RF(d,m)$ is a RF trained with hyperparameters d and m, $AUC(\cdot)$ is the AUC of a given RF on some held aside validation data and D, M are fixed sets. However, this procedure searches all combinations of d and m, whereas we are interested in controlling the value $m\,2^d$ because the overhead of our system its directly proportional to it. Therefore, between two hyperparameters choices giving the same efficacy, we are interested to choose the one that produces smaller overhead. In other words, our approach for tuning is the following: we fix a value s and then we maximize the model efficacy constrained to choosing the hyperparameters d and m in the set $Q_s = \{(m,d) \in \mathbb{Z}^+ \times \mathbb{Z}^+ \mid m\,2^d \leqslant s\}$. The gray line in Fig. 5.3 depicts the boundary of Q_s when $s = 2^{15}$ and dictates that choices above it are too large, and choices below are of acceptable overhead. Even if the number of acceptable choices is relatively small compared to the total number of combinations, it is worth noting that we saw competitive performance as both depth and number of trees exceeded some minimum choices. This suggests that we may be able to achieve both good performance and small overhead.

5.4 Performance: Efficacy

To conclude our work, we want to experimentally validate our system. First, we study the effect of the "locally learn then merge" approach on the prediction accuracy. In particular, we want to compare the accuracy of the proposed method with the one of the standard "merge then learn" approach[b]. We provide an empirical investigation of this in two fashions: across three disease prediction tasks using EHR data from the Marshfield Clinic in Marshfield, WI, and across five predictions tasks from the UCI database.

[b] If there are no privacy concerns, parties can simply share their data with one another and learn a single model. Otherwise a privacy-preserving training algorithm can be used to achieve the same result.

Real EHR Data. We consider the tasks of predicting three diseases one month in advance: Influenza (ICD-9 487.1), Acute Myocardial Infarction (ICD-9 410.4), and Lung Cancer (ICD-9 162.9). Each dataset was comprised of up to 10,000 randomly selected case-control matched patients on age and date of birth (within 30 days), with cases having 2 or more positive entries of the target diagnosis on their record and the control having no entries (rule of 2). Data for each case-control pair were truncated following 30-days prior to the case patient's first diagnosis entry to control for class-label leakage. Features were comprised of patient demographics, diagnoses, procedures, laboratory values and vitals. Unsupervised feature selection was performed on a per-disease basis first with a 1% frequency-based filter to remove very uncommon features and then followed up with principal component analysis to reduce the overall dimension of the data to 1,000 (this was done to improve the performance speed of our algorithm). For each of the three diseases, we constructed, as a performance baseline, a RF model with 500 trees, a maximum depth of 8, and 10% of features considered at each split in the tree. Models were trained on 90% of the data and tested on a held aside 10%. We compared these baseline models (*i.e.*, "merge then learn" approach) with our "locally learn then merge" approach by again constructing a forest with the same hyperparameters except the training data were partitioned between two simulated providers each with 45% of the original data that were used to train two smaller forests of 250 trees each and then merged together. Model performance was measured using the area under the receiver operating characteristic curve (AUC), a common machine learning accuracy metric. We present in Table 5.1 both the dataset information and results of our experimentation. We find that the AUC achieved on partitioned data for these three tasks is less than the shared data. While this efficacy loss is meaningful, it is possible that with the additional data providers it may be mitigated.

UCI Datasets. We use five UCI datasets (references in Table 5.2) to investigate the effect of the number of providers sharing data on the performance of a RF. To simulate a dataset being shared amongst t providers, we randomly split each UCI dataset into t equal sized and unique chunks, $D_1, \ldots, D_t$, with each chunk

ICD-9	Disease	Samples	Features	Base AUC	LLM AUC	Prediction Time (s)
487.1	Influenza	10,000	8,211	0.8011	0.7640	105.37±14.70
410.4	Acute MI	9,284	9,136	0.6797	0.6658	121.75±9.43
162.9	Lung Cancer	10,000	9,021	0.6313	0.5786	125.94±8.19

Table 5.1: Efficacy testing results for 3 EHR datasets. Number of features are calculated before applying PCA (post-PCA selected the top 1,000 components). Base AUC refers to a forest learned on the whole dataset ("merge then learn" approach) and LLM AUC refers to a forest learned in our "locally learn then merge" fashion. Prediction Time refers to the mean±std time required for our system to return a prediction for a single patient query.

belonging to a single provider. Each chunk was then split into a training (70% of the data) and testing set (30% of the data), *i.e.* $D_i = \text{Train}_i \cup \text{Test}_i$. We then learned models in three different ways. To simulate the effect of "zero sharing" (*i.e.*, providers with silo data do not share data or models), provider i learns a forest on Train_i and tests on Test_i achieving AUC_i with the average silo AUC taken as the mean across all t providers. Each forest was learned with 50 trees of maximum depth 8. To simulate the effect of "locally learn then merge", each provider learns a RF on their own training data, the forests are merged together, and the AUC is calculated on the merged testing data, $\cup_i \text{Test}_i$. Again, each provider learned 50 trees of maximum depth 8 and the final merged forest being of size $50\,t$ trees. To simulate the effect of a merged dataset ("merge the learn") we learn a single forest with $50\,t$ trees and maximum depth 8 from $\cup_i \text{Train}_i$ and then evaluate the AUC on $\cup_i \text{Test}_i$. This process was repeated 50 times to produce confidence intervals and performed for each of the five datasets in Table 5.2 across five choices of $t \in \{2, 3, 4, 5, 6\}$. We present the results of these experiments in Fig. 5.4. We see from it that the effect of locally learning the merging has neither a strictly positive or negative effect on the quality of the model. Indeed, our results indicate that the effect is dataset dependent. Therefore, we believe that it would be critical for a provider to investigate how the quality of their predictions are impacted by merging their learned models with another hospital system as compared to using their own data.

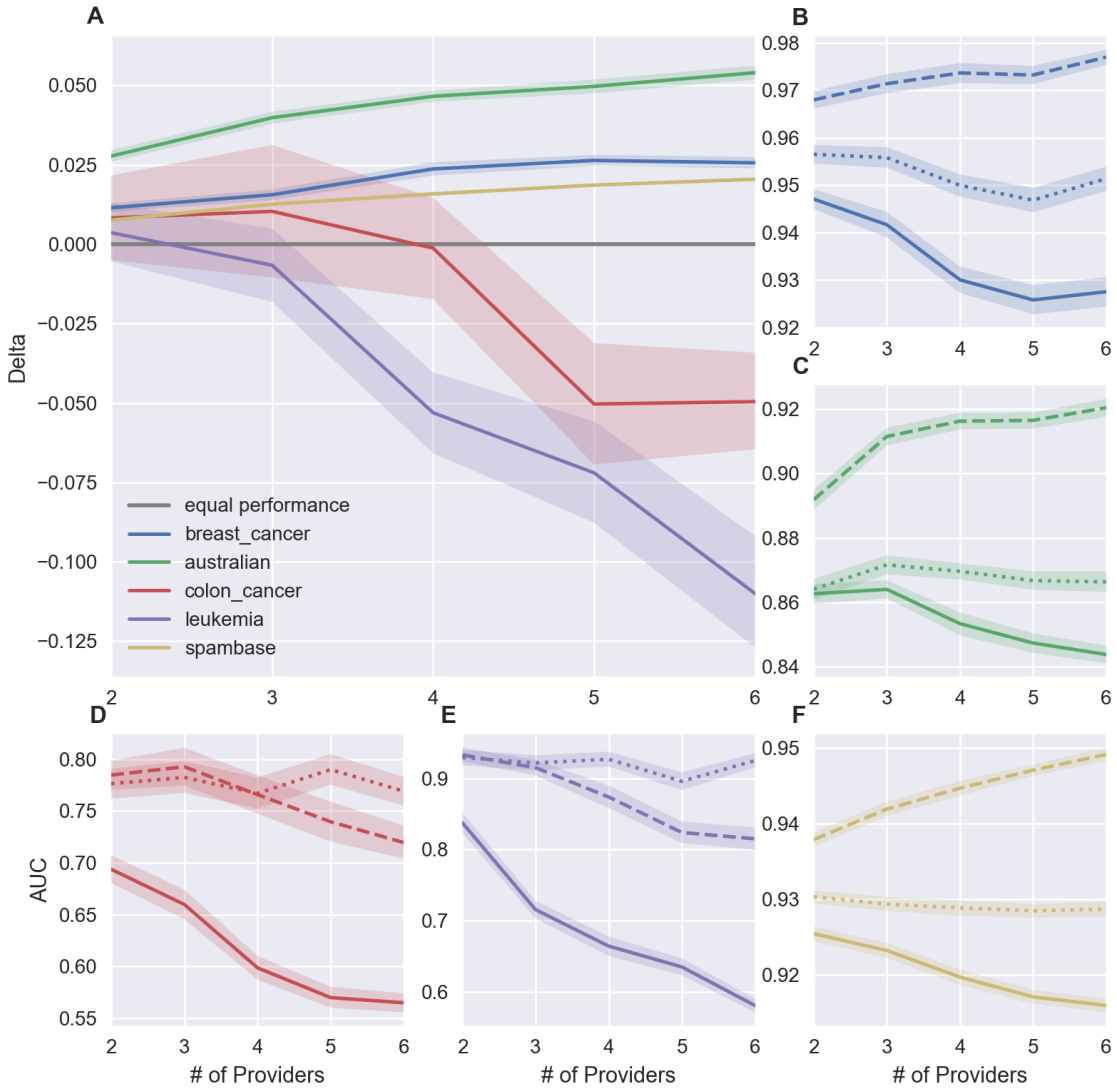


Figure 5.4: Effect of locally learning then merging compared to learning from a merged dataset. Subfigures **B-F** shows on the datasets of Table 5.2 how AUC is impacted by the number of providers. The dashed, solid and dotted lines shows AUC values for the locally learn then merge, the zero-sharing and the merge then learn approach, respectively. Subfigure **A** shows the AUC difference between the locally learn then merge and merge then learn (positive values indicate an improvement using our approach).

5.5 Performance: Efficiency

Implementation details. To test efficiency (*i.e.*, bandwidth and running time) we implemented our proposed system in Python3.5.2. As underlying LHE we use Joye-Libert's scheme (Joye and Libert, 2013) with $\mathcal{M} = \mathbb{Z}_{2^{64}}$ and 100-bit security. We assume all inputs are real number with absolute value less or equal to $2 \cdot 10^3$ and at most 3 digits in the fractional part. To convert them into values in $\mathcal{M}$, we multiply each value by 10^3. This allows to represent all inputs with 21-bits values (we represent negative values using the upper half of $\mathbb{Z}_{2^{21}}$) and avoid overflow in the secure comparison protocol. The $\binom{2^d}{1}$-OT protocol for 20148-bit strings is implemented (Naor and Pinkas, 1999) using d calls to a standard $\binom{2}{1}$-OT protocol (*i.e.*, emp-toolkit) for 100-bit strings and $2d$ calls to a PRF (*i.e.*, AES_{128}). We provide an empirical investigation of the efficiency in two fashions: using a commodity machine and using the HTCondor system.

Commodity machine. We report the performance of our system executed on a commodity machine (60GB memory and 48core CPU, Intel Xeon CPU E5-2680 v3) for the UCI datasets of Table 5.2 in the setting described before (*i.e.* each provider knows a RF with 50 trees of maximum depth 8). Several tasks in the implementation were parallelized by multi-threading; all the timing values are averaged on five repetitions of the same experiment. Table 5.2 (last two columns on the right) reports the running time of the model-encryption procedure executed by one provider during the *off-line phase*; it also reports the size of the encrypted model obtained via this procedure. The number n of features influences both results, however even for the high dimensional cases (*i.e.*, thousands of features) the encrypted model has size less than 1 GB and is produced in less than a minute. The *on-line phase* of our system consists of three steps: first, the user submits its encrypted input to the server. Clearly, the performance of this step is influenced only by the encryption scheme used and by the dimension of the input (*i.e.*, number of features n). In our experiments, even for the largest value of n, this step takes less than a second (*e.g.*, 0.17 seconds for $n = 7129$). Then, the server and the user execute m times the protocol Π_{TE} to evaluate each tree in the merge of all the forests. Fig. 5.5 illustrates

	samples	features	Model-encryption	
			Time (s)	Size (MB)
Australian	609	14	0.84	7.96
Breast cancer	569	30	0.90	9.6
Spambase	4601	57	1.01	12.35
Colon cancer	62	2000	10.57	210.54
Leukemia	72	7129	41.7	733.69

Table 5.2: References for the UCI datasets. The last two columns show the overhead of the off-line phase of our system.

the performance of this part (the most expensive one in the on-line phase): The two graphs on the left depict the running time of the protocol Π_{TE} run on $50\,t$ trees as function of the parameter t, number of providers; the results are dataset dependent since the server executes $\Theta(n\,2^d)$ cryptographic operations. The graph on the right of Fig. 5.5 reports the size of the messages exchanged by the server and the user as function of t. This value is not influenced by n (dataset size) and it only increases linearly with the number of trees; in our experiment, even for 300 trees the bandwidth required is always less than 60 MB. In the last step of the on-line phase, the server and the user sum their shares; the overhead of this step is independent of n and influenced only by the total number of trees (*e.g.*, in our experiment this needs less than 8 ms for 300 trees).

HTCondor. The experiments for the real EHR data were executed using the HTCondor system, a high-throughput computing architecture that we utilized in a "master-worker" fashion. For each forest, one tree was learned as a separate "job" exploiting the heavy parallelization available to RFs. Thus, both training and prediction were performed in a high-throughput manner. We report the running time of the on-line phase in this setting in the last column on the right of Table 5.1. We find that this parallelized version of our algorithm allows us to provide near real-time interactions as predictions are returned on average within two minutes of providing a query to the system. We believe that this would be reasonably fast enough to support the workflow of a physician who wishes to query the model for a patient.

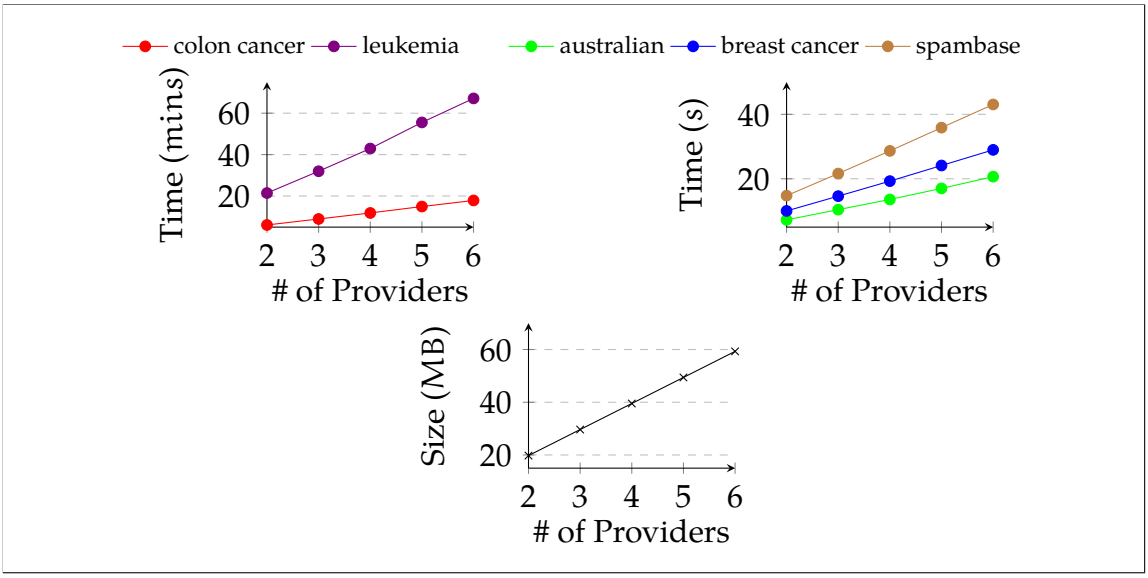


Figure 5.5: Performance of protocol Π_{TE} on $(50 \times \#\text{ of Providers})$ trees of depth 8 for the datasets of Table 5.2.

5.6 Conclusion

We propose a new approach for computing privacy-preserving collaborative predictions using random forests. Instead of a system composed by a training algorithm, which usually has high overhead in the privacy-preserving setting, followed by a scoring algorithm, we propose a system based on locally learning and then privacy-preserving merging. To avoid the need for providers to be on-line for each prediction request, we instantiate the new approach in the cloud model. That is, an untrusted server collects the locally trained models in encrypted form and takes care of scoring them on a new private instance held by the user. Our system is secure in the honest-but-curious security model and extending it to the malicious model, especially for a corrupted server, is an interesting direction for future work. We evaluate the performance of our system on real-world datasets, the experiments we conducted show that (1) the efficacy of the new approach is dataset dependent; we intend to pursue this future work by applying our methodologies data

from multiple hospital systems, and (2) the efficiency is influenced by the forest hyperparameters, and by the number of features n, which is given by the specific application; avoiding the dependency on n is another interesting direction that may lead more efficient implementation of this new approach.

Co-authorship and Acknowledgements

The work presented in Chapter 5 is the result of a collaboration with multiple researchers that was published in the proceedings of the 2019 AMIA Informatics Summit. This work was completed with equal efforts from the following authors and affiliations listed alphabetically: Irene Giacomelli[1], Somesh Jha[2], Ross Kleiman[2], David Page[2], and Kyonghwan Yoon[2]; [1]ISI Foundation, Turin, Italy; [2]University of Wisconsin-Madison, Madison, WI, USA. This work was supported by the Clinical and Translational Science Award (CTSA) program, through the NIH National Center for Advancing Translational Sciences (NCATS) grant UL1TR002373, by the NIH BD2K Initiative grant U54 AI117924 and by the NLM training grant 5T15LM007359.

6 MACHINE LEARNING ASSISTED DISCOVERY OF NOVEL PREDICTIVE LAB TESTS USING ELECTRONIC HEALTH RECORD DATA

In Chapter 4 we introduced the concept of high-throughput machine learning as a means of learning models for many thousands of diagnoses at once. Each of these random forest models can be partially understood by inspecting the feature importances learned by the forests. Amongst these features are laboratory tests that may be of some clinical value in diagnosis and/or better understanding the modeled disease. In some cases, these laboratory tests were not previously mentioned in the literature and thus may be of medical value. In this chapter, we introduce a high-throughput approach to identification of novel predictive lab tests by combining these feature importances with a literature derived knowledge base.

6.1 Introduction

Epidemiological studies associating changes in biological markers (often measured by laboratory tests) and disease states are an invaluable tool for better understanding disease mechanism, such as the Framingham heart study (Dawber et al., 1951). Moreover, many diagnostic criteria exploit such associations by testing for abnormal changes in the associated biological marker. For example, the diagnostic criteria for Type II diabetes (a metabolic disorder that impacts how the cells uptake glucose) includes a measurement of fasting blood glucose levels. The benefits of a well performed epidemiological study are clear, but such studies can be time-consuming, expensive, and like any scientific study they require some intuition of the link searched for. Further, not all hypothesized associations will be borne out in the data, and thus the higher the quality of initial hypothesis, the more likely the study yields valuable medical information. In this work, we attempt to generate a ranked set of high-quality candidate hypotheses through a combination of machine

learning, text-mining based literature searches, and traditional logistic regression analysis.

Our method identifies "novel predictive lab tests," which we define as a previously unknown association between a given disease state and a given biological marker (measurable by a laboratory test) that changes prior to diagnosis. Our approach hinges on the assumption that a novel diagnostic lab test is 1) useful for predicting a given diagnosis (via a machine learning model) and 2) not currently discussed in published medical literature on PubMed. Using these criteria we hypothesize that given a set of candidate diagnoses, we can generate a set of high-quality novel diagnostic lab tests in a five step procedure:

1. For each diagnosis, produce a machine learning model to predict it, and select the top k lab tests used for prediction by their feature importance.

2. Map the clinic-used name for each diagnosis and lab test to the literature-used name (clinic names in our data source commonly used names and abbreviations that may not have been found in literature).

3. Perform automated text-mining to search for literature associations between diagnoses and lab tests. This provides a pseudo knowledge-base of known diagnostic lab tests.

4. Rank novel diagnostic lab tests in a way that encourages high predictive usefulness and low literature presence.

5. Evaluate the top candidates with traditional logistic regression analysis to only retain hypotheses that are statistically significant both with and without inclusion of potential confounders.

The use of Electronic Health Records (EHRs) to digitally capture patient health encounters has grown substantially in recent years (Hsiao et al., 2014). This has created unprecedented opportunity for secondary use of EHR data in combination with machine learning algorithms to build predictive models for critical patient health events such as breast cancer (Gail et al., 1989) or heart attack (Dawber et al.,

1951) risk. Machine learning algorithms flexibly learn relationships in a data set without the need for hard coded rules. In this work we first utilize a machine learning algorithm, specifically random forests (Breiman, 2001), to build predictive models of disease for which we are interested in finding novel diagnostic lab tests. Random forests work by forming an ensemble of multiple decision trees, each learned on a randomly bootstrapped sample of the original dataset. They are well known for their strong predictive performance (Caruana and Niculescu-Mizil, 2006) and resilience to applications with very large numbers of features (Breiman, 2001) (which is true of EHR data).

While machine learning algorithms can give a quantitative prediction, or even a confidence of the prediction, one of their common critiques is interpretability. That is, machine learning algorithms do not offer reasoning along with their predictions. One way that data scientists can interpret machine learned models is by observing which features most impact their predictions. The random forest algorithm has support for "feature importances" (Breiman, 2001) which provide a window into the model by ranking the contribution of each feature to the construction of the model. In random forests, feature importances are non-negative values for which larger values suggest a greater contribution of that feature towards prediction. In this work, we use feature importances to discover potential novel predictive lab tests.

While a researcher might consider conducting manual literature searches to validate potential novel discoveries from a trained model, the size and exponential growth in scientific literature (Pautasso, 2012; Bornmann and Mutz, 2015) make this approach infeasible. The literature search space to validate tens or hundreds of lab tests across tens or hundreds of diagnoses is too large for an individual to reasonably explore. We therefore use a text mining approach to search for associations between diagnoses and lab tests. For this work, we rely on KinderMiner, a previously developed algorithm designed to filter and rank a list of target terms by their association in the literature with a key phrase of interest (Kuusisto et al., 2017). In our particular application, a diagnosis name is the key phrase of interest, and the names of important lab features are the target terms to be ranked by association

with the diagnosis. In contrast to the original intent of KinderMiner, which is to rank the target terms by their positive association with the key phrase within the literature, we are looking for labs deemed useful for prediction, but which are not already well known in the literature. Thus, we modify KinderMiner to suit our needs by instead filtering and ranking terms by significant *lack* of association with the diagnosis in the literature.

In this work, we demonstrate our proposed method of identifying novel predictive lab tests by gathering a set important diagnoses and their most predictive lab tests from machine learning models. We then use text mining to filter and rank hypothesized predictive lab tests based on negative association within the literature. Finally, we evaluate the top hypotheses proposed by our method and find several to be promising candidates for further investigation. In the following sections we specifically describe the pipeline: the selection of diagnoses and labs, the construction of predictive models, text mining based filtration and ranking, and final evaluation of the hypotheses.

6.2 Selection of Diagnoses and Lab Tests

To select the set of important diagnoses to consider, we started with the 100 most common diagnoses by patient count in our EHR dataset. We then manually filtered out diagnoses that we considered unlikely to be diagnosed via lab tests or that were effectively a restatement of an abnormal lab value. For example, we removed ICD-9 code 719.46 (Pain in joint; lower leg) because it is likely a result of a mechanical ailment, and we removed ICD-9 code 272 (Pure hypercholesterolemia) because it is a diagnosis of an abnormal lab value. We also manually curated our diagnosis descriptions to better reflect what we would expect to find in the literature and to include synonyms. For example, "gout; unspecified" became just "gout" and "dysthymic disorder" became "dysthymic disorder" or "dysthymia." See Table 6.1 for our final chosen list of 69 diagnoses and the search terms we used for each.

Table 6.1: The 69 diagnoses that we considered along with all search terms we used for each. Alias search terms are separated by semicolons.

ICD Code	Diagnosis Name	Curated Search Terms
162.9	Malignant neoplasm of bronchus and lung; unspecified site	malignant lung cancer
174.9	Malignant neoplasm of breast (female); unspecified site	malignant breast cancer
274.9	Gout; unspecified	gout
300.01	Anxiety state; unspecified	anxiety; gad; generalized anxiety disorder
300.4	Dysthymic disorder	dysthymic disorder; dysthymia
305.1	Tobacco use disorder	tobacco use
309.28	Adjustment disorder with mixed anxiety and depressed mood	adjustment disorder
314.00	Attention deficit disorder of childhood without mention of hyperactivity	attention deficit disorder
314.01	Attention deficit disorder of childhood with hyperactivity	attention deficit hyperactivity disorder; adhd
327.23	Obstructive sleep apnea (adult) (pediatric)	obstructive sleep apnea
362.51	Nonexudative senile macular degeneration of retina	senile macular degeneration
366.10	Unspecified senile cataract	senile cataract; senile cataracts
366.16	Nuclear sclerosis	nuclear sclerosis
367.0	Hypermetropia	hypermetropia; farsightedness; hyperopia; farsighted
367.1	Myopia	myopia
367.4	Presbyopia	presbyopia
372.30	Unspecified conjunctivitis	conjunctivitis
379.21	Vitreous degeneration	vitreous degeneration
382.9	Unspecified otitis media	otitis media
388.70	Unspecified otalgia	otalgia

Table 6.1: (continued)

ICD Code	Diagnosis Name	Curated Search Terms
389.9	Unspecified hearing loss	hearing loss
410.71	Acute myocardial infarction; subendocardial infarction; initial episode of care	acute myocardial infarction; heart attack; subendocardial infarction
411.1	Intermediate coronary syndrome	intermediate coronary syndrome
413.9	Other and unspecified angina pectoris	angina
414	Other forms of chronic ischemic heart disease	chronic ischemic heart disease
424.1	Aortic valve disorders	aortic valve disorder
427.31	Atrial fibrillation	atrial fibrillation; afib
427.89	Other specified cardiac dysrhythmias	cardiac dysrhythmia
427.9	Unspecified cardiac dysrhythmia	cardiac dysrhythmias
428.0	Congestive heart failure; unspecified	congestive heart failure
434.91	Unspecified cerebral artery occlusion with cerebral infarction	ischemic stroke
440.9	Generalized and unspecified atherosclerosis	atherosclerosis
443.9	Unspecified peripheral vascular disease	peripheral vascular disease
461.9	Acute sinusitis; unspecified	acute sinusitis
462	Acute pharyngitis	acute pharyngitis
465.9	Acute upper respiratory infections of unspecified site	acute upper respiratory infection
466.0	Acute bronchitis	acute bronchitis
472.0	Chronic rhinitis	chronic rhinitis
473.9	Unspecified sinusitis (chronic)	chronic sinusitis
477.9	Allergic rhinitis; cause unspecified	allergic rhinitis; hay fever; seasonal allergies
486	Pneumonia; organism unspecified	pneumonia

Table 6.1: (continued)

ICD Code	Diagnosis Name	Curated Search Terms
490	Bronchitis; not specified as acute or chronic	bronchitis
493.90	Asthma; unspecified; unspecified status	asthma
496	Chronic airway obstruction; not elsewhere classified	chronic airway obstruction
521.00	Unspecified dental caries	dental caries; dental cavity
530.81	Esophageal reflux	esophageal reflux; gerd
558.9	Other and unspecified noninfectious gastroenteritis and colitis	non-infectious gastroenteritis; noninfectious gastroenteritis; non-infectious colitis; noninfectious colitis
562.10	Diverticulosis of colon (without mention of hemorrhage)	colon diverticulosis
564.0	Unspecified constipation	constipation
564.1	Irritable bowel syndrome	irritable bowel syndrome
574.20	Calculus of gallbladder without mention of cholecystitis or obstruction	gallbladder calculus; gallstones; gallstone
584.9	Acute kidney failure; unspecified	acute kidney failure; acute renal failure
585.3	Chronic kidney disease; Stage III (moderate)	stage 3 chronic kidney disease; ckd stage 3
592.0	Calculus of kidney	kidney calculus; kidney stone; nephrolithiasis
593.9	Unspecified disorder of kidney and ureter	kidney disorder; ureter disorder
599.0	Urinary tract infection; site not specified	urinary tract infection
600.0	Hypertrophy (benign) of prostate	benign prostate hypertrophy
611.72	Lump or mass in breast	breast mass; breast lump
616.10	Unspecified vaginitis and vulvovaginitis	vaginitis; vulvovaginitis
625.3	Dysmenorrhea	dysmenorrhea

Table 6.1: (continued)

ICD Code	Diagnosis Name	Curated Search Terms
626.2	Excessive or frequent menstruation	excessive menstruation; frequent menstruation; menorrhagia; polymenorrhea; hypermenorrhea
627.2	Symptomatic menopausal or female climacteric states	symptomatic menopause; symptomatic menopausal
692.9	Contact dermatitis and other eczema; due to unspecified cause	contact dermatitis; eczema
702.0	Actinic keratosis	actinic keratosis
706.1	Other acne	acne
709.9	Unspecified disorder of skin and subcutaneous tissue	skin disorder
723.4	Brachial neuritis or radiculitis NOS	brachial neuritis
724.4	Thoracic or lumbosacral neuritis or radiculitis; unspecified	thoracic neuritis; thoracic radiculitis; lumbosacral neuritis; lumbosacral radiculitis; thoracic radiculopathy; lumbosacral radiculopathy
729.1	Unspecified myalgia and myositis	myalgia; myositis

To select the lab tests to consider, we assembled the union of the top 10 most important lab features (according to our random forest models) from each of our 69 chosen diagnoses. There was substantial overlap of important features between diagnoses, leaving us with a total of 52 different lab features from our EHR dataset. Just as with the diagnoses, we curated the lab test names to better reflect what we would expect to find in the literature and to include synonyms. See Table 6.2 for our list of 52 lab tests and the search terms we used for each.

Table 6.2: The 52 lab tests that we considered along with all search terms we used for each. Alias search terms are separated by semicolons.

Lab Name	Curated Search Terms
ALT (GPT)	alanine aminotransferase
AST (GOT)	aspartate aminotransferase test
Anion Gap	anion gap
Bacteriuria Screen (Esterase)	bacteriuria esterase; bacteriuria screen; bacteriuria test
Bacteriuria Screen (Nitrate)	bacteriuria nitrate; bacteriuria screen; bacteriuria test
Bicarbonate (CO2)	blood bicarbonate; blood co2; serum bicarbonate; serum co2
Bilirubin, Total-Neonatal	neonatal bilirubin; neonatal bile
Calcium	blood calcium; serum calcium
Chloride (Cl)	blood chloride; serum chloride
Cholesterol	cholesterol blood; serum cholesterol
Creatinine, Blood	blood creatinine; serum creatinine
Culture Organism	culture organism
Differential Segment Neut-Segs	segmented neutrophils; segmented pmn
Direct Bilirubin	direct bilirubin; conjugated bilirubin
Glom Filter Rate (GFR), Est	estimated glomerular filtration rate; egfr
Glucose	blood glucose; serum glucose
HDL Cholesterol	high density lipoprotein; hdl cholesterol
Hematocrit (Hct)	hematocrit
Hemoglobin (Hgb)	hemoglobin
Low Density Lipoprotein(LDL-C)	low density lipoprotein; ldl cholesterol
MCH	mean corpuscular hemoglobin
MCHC	mean corpuscular hemoglobin concentration
Mean Corpuscular Volume (MCV)	mean corpuscular volume
Phosphorus	blood phosphorus; serum phosphorus
Platelet Count (Plt)	platelet count
Potassium (K)	blood potassium; serum potassium

Table 6.2: (continued)

Lab Name	Curated Search Terms
Prothrombin Time (PT)-INR	prothrombin time; international normalized ratio
Rapid Strep Antigen	rapid strep test
Red Blood Cell (RBC) Count	red blood cell count
Red Cell Distribute Width(RDW)	red cell distribution width
Sodium, Bld (Na)	blood sodium; serum sodium
Thyroid Stimul Hormone-Mfld	thyroid stimulating hormone
Total Cholesterol/HDL Ratio	cholesterol ratio
Triglycerides	triglycerides blood; triglycerides serum
Unconjugated Bilirubin	unconjugated bilirubin
Urea Nitrogen,Bld	urea nitrogen blood; serum urea nitrogen
Uric Acid,Bld	uric acid blood; uric acid serum
Urinalysis-Coarse Gran	urine coarse granular casts
Urinalysis-Color	urine color
Urinalysis-Fine Gran	urinary cast fine
Urinalysis-Hyaline	urine hyaline
Urinalysis-RBC	urine red blood cell
Urinalysis-Renal Epi	renal epithelial cells urine
Urinalysis-Spec Type	urinalysis specimen
Urinalysis-Specific Gravity	urine specific gravity
Urinalysis-Turbidity	urine turbidity
Urine Bile	urine bile; urine bilirubin
Urine Blood	urine blood
Urine Ketones	urine ketones
Urine Urobilinogen	urine urobilinogen
Urine pH	urine ph
White Blood Cell Count (WBC)	white blood cell count

6.3 Predictive Models and Feature Importance

For each of the 69 diagnoses of interest we constructed a random forest model using case-control matched patient EHR data from Marshfield Clinic in Wisconsin. We phenotyped cases and controls from the EHR data using the "rule of 2" with cases having 2 or more entries of the diagnosis on their record and controls having no entries. We matched cases and controls based on age and date of birth (within 30 days) and we truncated all data for a case-control pair following 30 days prior to the case patient's first entry of the diagnosis of interest. In this fashion we generated 5,000 case-control pairs (a total of 10,000 patients). Our patient data included demographics, diagnoses, labs, vitals, and procedures. Demographic features included age, sex, and date of birth. We summarize this information in Table 6.3. For all features except demographics, we extracted the features as counts in the time windows: 1-year, 3-years, 5-years, and ever. In this manner, our features were of the form "4 influenza diagnoses in the last 3-years", or "2 high blood glucose labs in the last 1-year". We used the random forest implementation from the Python package scikit-learn (Pedregosa et al., 2012) version 0.15. Each forest was trained with 500 trees and 10% of the features randomly selected at each split. We chose these setting a priori, as our prior research has performed well with these choices. All other settings for the forest used default parameters. We extracted feature importance values using scikit-learn's built in functionality which uses the standard random forest feature importance calculation method (Breiman, 2001).

Table 6.3: Demographic summary for the Marshfield electronic health record population.

Characteristic	Women	Men	Total
n	565,011 (51.5%)	532,083 (48.5%)	1,097,094
Mean age, yrs	46.7 ± 25.7	44.9 ± 25.5	45.8 ± 25.6
$<$ 18 y.o	84,917 (15.0%)	98,183 (18.5%)	183,100 (16.7%)
18-39 y.o.	157,827 (27.9%)	137,967 (25.9%)	295,794 (27.0%)
40-59 y.o.	132,280 (23.4%)	123,642 (23.2%)	255,922 (23.3%)
$\geqslant$ 60 y.o.	189,987 (33.6%)	172,291 (32.4%)	362,278 (33.0%)

6.4 Text-Mining

We modified the KinderMiner algorithm for the text mining portion of this work to determine which diagnostic lab tests are likely novel in the literature. KinderMiner filters and ranks a list of target terms by their association with a key phrase of interest. It accomplishes this through simple string matching and document counting within a given text corpus. For each search, the user must specify the key phrase representing a concept of interest along with the list of target terms to be filtered and ranked by their association with the key phrase. KinderMiner then searches a given text corpus for article counts matching the target terms and key phrase. Specifically, it computes a contingency table of counts for each target term. For each target term, KinderMiner computes the number of articles containing both, either, and neither of the target term and key phrase. The result of this procedure is a list of contingency tables of document counts, one table for each target term. KinderMiner then performs a one-sided Fisher's exact test on each contingency table, filtering out target terms that do not demonstrate statistically-significant co-occurrence with

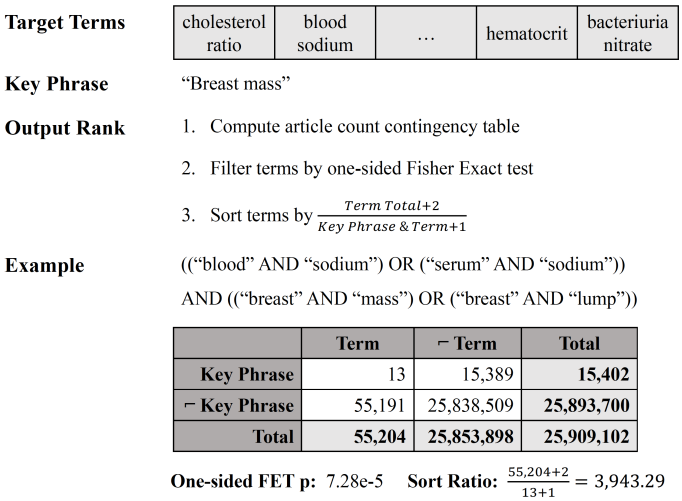


Figure 6.1: Visual example of our modified KinderMiner, with contingency table and disassociation Fisher's Exact Test (FET) analysis of the diagnosis key phrase "breast mass" and the lab target term "blood sodium." Target terms are filtered by significance of disassociation with the key phrase and then sorted by the inverted co-occurrence ratio.

the key phrase according to a specified p-value threshold. Finally, the remaining target terms are ranked by the co-occurrence ratio, which is the number of articles in which a target term co-occurs with the key-phrase divided by the total number of articles in which the target term appears.

In this work, we make four modifications to the original KinderMiner algorithm (see Figure 6.1 for a visual representation). First, while the original KinderMiner algorithm finds exact string matches for target terms and key phrases, we extend this by breaking target terms and key phrases into their constituent tokens and matching on all tokens in any order or location within the document. For example, in the original KinderMiner a key phrase like "stage 3 chronic kidney disease" would need to match that string exactly to be counted and would not match the similar phrase "chronic kidney disease, stage 3." Our modification breaks this key phrase into five tokens ("stage", "3", "chronic", "kidney", and "disease") which must all be present in the document, but which do not need to match exactly in the original phrasing order.

Second, we extend KinderMiner to accommodate alias matching for target terms and key phrases. For example, we may expand a target term like "blood sodium" with an alias like "serum sodium." Similarly, we can expand a key phrase like "breast mass" with an alias like "breast lump." This is important for key phrases and target terms that may be referred to in multiple ways within the literature.

Third, in contrast to the original goal of KinderMiner, we wish to identify targets that are negatively associated with the key phrase in the literature. To accomplish this, we modify KinderMiner's filtration step by changing the one-sided Fisher's exact test to the opposite side test, thereby testing for significant negative association between each target term and key phrase.

Fourth, we also change how KinderMiner ranks the final filtered set of associations. KinderMiner typically ranks results by the co-occurrence ratio, the proportion of articles in which both the key phrase and target term occur over all articles in which the target term occurs. This is useful because it gives a rough estimate of the magnitude of association between the key phrase and the target term. In this work, we instead use the inverted co-occurrence ratio because it gives a rough estimate

of the disassociation. When computing this ratio, we also add a pseudo-count of one to each of the article counts for when the key phrase and term co-occur, and when the target term appears without the key phrase. We add these pseudo-counts because it is not rare to find a significant disassociation when there are zero articles in which the term and key phrase co-occur.

As part of the filtration step, KinderMiner requires a p-value threshold for the Fisher's exact test. While the original KinderMiner paper used 0.00001 in all cases, we loosen that threshold to 0.05 for our work. Because there are already few candidate lab-diagnosis pairs that appear unexpectedly disassociated in the literature, we care more about getting sufficient candidate discoveries than filtering out false positives.

KinderMiner also requires a text corpus to search. We constructed our text corpus from the National Library of Medicine's MEDLINE/PubMed publicly available citation records (US National Library of Medicine). We downloaded the annual baselines in XML format, parsed, and then ingested them into an Elasticsearch index (version 2.4.6). Our initial ingest of the 2017 annual baseline was performed in June and July 2017, and we updated to the 2018 baseline in November 2017. The dataset contains 27,947,480 citation records, with the abstracts indexed by Elasticsearch using two analysis chains. The default analysis that we use for all of the searches in our work is Elasticsearch's standard analyzer, which applies a grammar-based tokenizer and lowercase filter to the text. We then use the Elasticsearch Query Domain Specific Language to construct each of our queries in JSON. Altogether, a search for the key phrase "breast mass" (with alias "breast lump") and target term "blood sodium" (with alias "serum sodium") would be equivalent to the following:

```
((‘‘breast’’ AND ‘‘mass’’) OR (‘‘breast’’ AND ‘‘lump’’)) AND ((‘‘blood’’
            AND ‘‘sodium’’) OR (‘‘serum’’ AND ‘‘sodium’’))
```

Hypothesis Ranking and Evaluation

Once we have gathered a set of hypothesized lab-diagnosis pairs to consider, we must rank them to help prioritize the best candidates for further investigation.

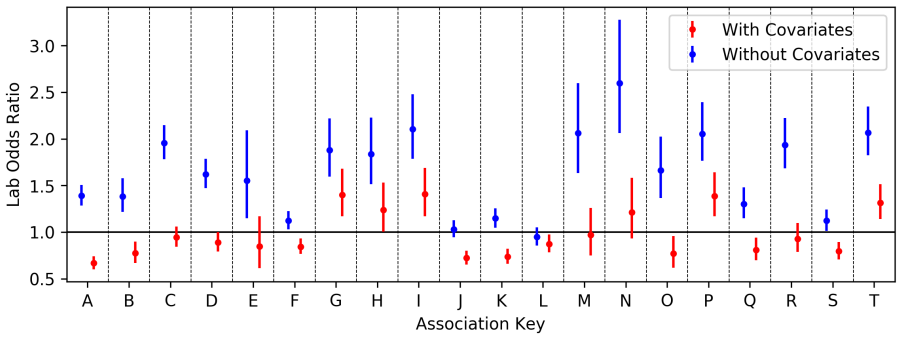


Figure 6.2: The odds ratios and confidence intervals of all 20 lab-diagnosis hypotheses. Includes odds ratios for both with (right, red) and without (left, blue) covariates.

Recall that we initially rank lab-diagnosis pairs by the inverted co-occurrence ratio. While this provides a lab test ranking for a particular diagnosis, we have several hypotheses from different diagnoses and we want to identify the most promising hypotheses overall. To do this, we construct a combined rank score defined as the product of a literature score and a feature score. For the literature score, we simply use the inverted co-occurrence ratio, which takes on large values when a lab is infrequently mentioned with a diagnosis. For the feature score, we use the feature importance of the lab value in the diagnosis model multiplied by the number of features in that diagnosis model (as to not bias models with small numbers of features). The product of the literature score and feature score thus gives a combined estimate of the novelty and the diagnostic importance of the lab.

With all the lab-diagnosis pairs ranked, we consider the top 20 hypotheses in more detail. To evaluate each candidate, we perform logistic regression analyses on the same dataset that was used to train the random forest. First, for each hypothesis, we perform a logistic regression with the diagnosis as the response variable and the laboratory test as the sole covariate. We use this to calculate the odds ratio of the lab in question. Second, we assess the odds ratio of the lab test in the presence of potential confounders. Finally, we perform manual literature search for important findings and decide if each is in fact novel.

To select potential confounders for a given diagnosis, we perform L1-regularized

Table 6.4: Summary and logistic regression odds ratios for the top 20 hypotheses. We compute the odds ratio of the lab test for a given hypothesis in two ways: as the sole covariate, "Odds", and including potential confounders, "Adjusted Odds". For both odds ratio calculations, we present the 95% confidence interval and bold 4 of the top 20 hypotheses whose 95% confidence intervals exclude 1.0 and whose odds are in the same direction, both before and after including confounders.

Key	ICD-9	Diagnosis	Lab Test	Odds	Adjusted Odds
A	702.0	Actinic Keratosis	Glucose	1.4 [1.3, 1.5]	0.67 [0.6, 0.74]
B	702.0	Actinic Keratosis	Creatinine, Blood	1.4 [1.2, 1.6]	0.78 [0.67, 0.9]
C	367.4	Presbyopia	Glucose	2.0 [1.8, 2.1]	0.95 [0.84, 1.1]
D	600.0	Benign Prostate Hypertrophy	LDL Cholesterol	1.6 [1.5, 1.8]	0.89 [0.79, 1.0]
E	702.0	Actinic Keratosis	Sodium, Bld (Na)	1.6 [1.2, 2.1]	0.85 [0.61, 1.2]
F	162.9	Malignant Lung Cancer	LDL Cholesterol	1.1 [1.0, 1.2]	0.84 [0.76, 0.93]
G	461.9	Acute Sinusitis	HDL Cholesterol	**1.9 [1.6, 2.2]**	**1.4 [1.2, 1.7]**
H	461.9	Acute Sinusitis	LDL Cholesterol	1.8 [1.5, 2.2]	1.2 [1.0, 1.5]
I	472.0	Chronic Rhinitis	Glucose	**2.1 [1.8, 2.5]**	**1.4 [1.2, 1.7]**
J	162.9	Malignant Lung Cancer	HDL Cholesterol	1.0 [0.94, 1.1]	0.72 [0.65, 0.8]
K	496	Chronic Airway Obstruction	LDL Cholesterol	1.1 [1.0, 1.3]	0.74 [0.66, 0.82]
L	521.00	Dental caries	LDL Cholesterol	0.95 [0.85, 1.1]	0.87 [0.78, 0.97]
M	461.9	Acute Sinusitis	Hemoglobin (Hgb)	2.1 [1.6, 2.6]	0.97 [0.75, 1.3]
N	473.9	Chronic Sinusitis	Hemoglobin (Hgb)	2.6 [2.1, 3.3]	1.2 [0.93, 1.6]
O	367.0	Hypermetropia	Hemoglobin (Hgb)	1.7 [1.4, 2.0]	0.77 [0.62, 0.96]
P	461.9	Acute Sinusitis	Cholesterol	**2.1 [1.8, 2.4]**	**1.4 [1.2, 1.6]**
Q	496	Chronic Airway Obstruction	Triglycerides	1.3 [1.1, 1.5]	0.81 [0.7, 0.94]
R	530.81	Esophageal Reflux-Gerd	WBC Count	1.9 [1.7, 2.2]	0.93 [0.79, 1.1]
S	162.9	Malignant Lung Cancer	Cholesterol	1.1 [1.0, 1.2]	0.8 [0.71, 0.89]
T	466.0	Acute Bronchitis	Cholesterol	**2.1 [1.8, 2.3]**	**1.3 [1.1, 1.5]**

logistic regression on a feature set containing demographics, laboratory tests, and the top-level, whole integer ICD-9 codes. Moreover, our features for laboratory tests and demographics are binary features capturing if the patient did or did not have an entry of a particular health event in the last one year. We perform the L1-regularized logistic regression with scikit-learn (Pedregosa et al., 2012) and choose the minimum number of covariates greater than or equal to five by slowly increasing the regularization parameter. Note that as the cases and controls for this data were already age and sex matched we do not see these as discovered confounders. We then use these five (or more) selected features as confounders in logistic regression analysis (with R version 3.3.1) where we compute the odds ratio (and 95% confidence interval) of the lab in question both with an without the identified confounders. If the lab in question has an odds ratio that both maintains the same sign in both evaluations, and if the 95% confidence interval for both odds ratios exclude 1.0 (no change in odds), then we consider the hypothesis to be corroborated by the logistic regression analysis.

6.5 Results and Discussion

In Table 6.4 we present, in rank order, the top 20 hypotheses found between labs and diagnoses. In Figure 6.2 we plot the odds ratios of all 20 hypotheses both with and without potential confounders. We find that four of the 20 hypotheses passed the secondary logistic regression analysis and maintained an odds ratio 95% confidence interval above 1.0 both with and without potential confounders. In all hypotheses except one, hypothesis L, we see that the inclusion of confounders either diminishes or even reverses the trend found without confounders. For the four bolded hypotheses that passed our logistic regression analysis, we present in Table 6.5 the covariates selected by the L1-regularized logistic regression.

The four hypotheses that met our odds ratio criteria for further consideration effectively represented three distinct hypotheses: cholesterol for acute sinusitis, cholesterol for acute bronchitis, and blood glucose for chronic rhinitis. While we

Table 6.5: Covariates included as potential confounders for the 3 diagnoses included in the four hypotheses that passed the regression analysis. A minimum of five covariates were identified for each diagnosis, with Chronic Rhinitis including a sixth covariate as there was no L1 penalty that achieved five.

Acute Sinusitis	Chronic Rhinitis	Acute Bronchitis
V72: Examination	461: Acute Sinusitus	V72: Examination
Lab: Hemoglobin	473: Chronic Sinusitis	Lab: MCHC
786: Respiratory Symptoms	V72: Examination	Lab: Hemoglobin
465: Acute URI	465: Acute URI	786: Respiratory Symptoms
462: Acute Pharyngitis	493: Asthma	465: Acute URI
	786: Respiratory Symptoms	

expected to find few hits by design, we manually searched PubMed for articles related to these findings.

First, manual literature search for an association between cholesterol and acute sinusitis did not turn up direct associations. It did, however, turn up several hits for cholesterol granuloma of the maxillary sinus, which describes cysts containing cholesterol crystals and other fluids surrounded by fibrous tissue (Chao, 2006). Symptoms are vague, and there are only two noted specific symptoms: clear golden yellow antral washout fluid, and washout containing cholesterol crystals. A family history of hypercholesterolemia was noted in one study (Dilek et al., 1997). This tangential association between cholesterol and sinus ailments within the literature, suggests to us that this discovered hypothesis is a promising lead for further investigation.

Second, literature search for an association between cholesterol and chronic bronchitis turned up two relevant studies. One study notes that low plasma lipid levels, particularly HDL cholesterol, is indicative of bacterial infection, and that low total cholesterol is predictive of adverse outcomes in patients with lower respiratory infections (Gruber et al., 2009). Another study suggests that lipid levels in airway mucus may be diagnostic for infection (Bhaskar et al., 1987). While the presence of these studies suggests prior awareness of an association, the literature is limited and can be viewed as confirmatory of our approach.

Third, literature search for an association between blood glucose and chronic

rhinitis turned up one study. The study suggests that some antihistamine medications may affect blood glucose levels (Lal, 2000). If true, this indicates that our hypothesis may instead be a confounded result of patients with chronic rhinitis having abnormal blood glucose as a result of antihistamine prescription, rather than blood glucose being predictive of rhinitis. Given the limited literature, however, the hypothesis may still warrant further investigation.

6.6 Conclusion

In this work, we propose a high-throughput pipeline for generating high-quality hypotheses for novel lab tests to predict diagnoses. We test our pipeline on a large electronic health record dataset and the PubMed corpus and, after manual evaluation, find several promising hypotheses in the top candidates.

One limitation of this work is the current need for manual mapping of diagnosis and lab terms to curated search terms. We expect that future work incorporating standardized naming and coding schemes in EHR datasets may obviate this need or that the process may be automated more completely in the future. This would facilitate higher throughput of diagnostic lab discovery by allowing us to run our pipeline on all diagnoses and all laboratory tests rather than a subset.

Our pipeline leverages the latent knowledge and patterns present in electronic health record data and the PubMed corpus to identify potentially interesting epidemiological findings. However, our proposed method does not eliminate the need for experimental design and further investigation of findings. On the contrary, it augments this process, and we argue that it represents a valuable addition by assisting with the prioritization of experiments when identifying biomarkers of disease.

Co-authorship and Acknowledgements

The work presented in Chapter 6 is the result of a collaboration with multiple researchers that was published in the proceedings of the 2019 AMIA Informatics Sum-

mit. This work was by the following authors and affiliations: Ross Kleiman[1,*], Finn Kuusisto[2,*], Ian Ross[1], Peggy Peissig[3], Ron Stewart[2], C. David Page[1], Jeremy Weiss[4]; [*]equal contribution; [1]University of Wisconsin - Madison; [2]Morgridge Institute for Research; [3]Marshfield Clinic Resaerch Institute; [4]Carnegie Mellon University. The authors acknowledge support from the National Institutes of Health (NIH) grant U54-AI117924 and the National Library of Medicine (NLM) grant 5T15LM007359. The authors also thank Marv Conney for a grant to Ron Stewart and Finn Kuusisto.

7 AUC MU: A PERFORMANCE METRIC FOR MULTI-CLASS MACHINE LEARNING MODELS

In Chapters 3 and 4 we present disease prediction tasks in which we are interested in whether a patient will or will not present with a particular disease. However, there are some conditions that share the same constellation of symptoms for which it would be valuable to construct a multi-class classification model. These models could answer questions such as "Does this patient have Parkinson's Disease, Alzheimer's Disease, or no disease?" However, the evaluation of such multi-class models cannot use the standard AUC measure which is only valid for two classes. Existing performance measures have key flaws such as failing to properly score perfectly separated examples, interpretability difficulties, and combinatorial computational time complexities. Therefore, in this chapter we introduce a new multi-class performance measure that adheres to those critical properties of AUC. We apply this new measure for the multi-class prediction task of seven different digestive cancers.

7.1 Introduction

The area under the Receiver Operating Characteristic (ROC) curve, commonly referred to as the AUC, is ubiquitous in machine learning, yet it is limited to classification tasks with only two classes. There have been a variety of prior attempts to extend AUC to the multi-class setting but there is no consensus on the appropriate way to proceed. Multi-class AUC analogs must deal with new challenges in both computational complexity and decisions of which properties of the binary AUC are most important to preserve. Current approaches largely fall into two camps: those that are theoretically rooted and those that are focused on ease of use. We believe that the community has implicitly stated a preference for practicality, as the most widely used measure, M, introduced by (Hand and Till, 2001), is an easy to use multi-class AUC analog. However, in our work we show that M can fail to

return a score of 1 (perfect performance), even when for every example a model gives the correct label the highest probability.

In our work we first consider those properties of AUC that we believe to be most critical to its use and interpretation. The properties are based on work by Fawcett (2006).

1. If a model gives the correct label the highest probability on every example, then AUC = 1

2. Random guessing on examples yields AUC = 0.5

3. AUC is insensitive to class skew

We note that these three properties are all a consequence of the relationship between AUC and the Mann Whitney U-Statistic (Hanley and McNeil, 1982). The U-statistic, and hence the two-class AUC, is the probability the model will correctly rank two instances of difference classes. Therefore, rather than generalizing the ROC curve to handle $K > 2$ classes as others have done before, we instead turn our attention to generalizing the U-statistic for $K > 2$. We call our measure AUC_μ using the Greek letter mu (μ) as an abbreviation for "**m**ulti-class **U**-statistic."

In Section 7.2 we present a survey of prior work performed on extending AUC to the multi-class setting. In Section 7.3 we present background on the U-statistic form of AUC, multi-class AUC, and partition matrices (a tool we use in computing AUC_μ). In Section 7.4 we formulate the AUC_μ statistic. In Section 7.5 we provide several theoretical results for AUC_μ and we also demonstrate some special cases for AUC_μ. In Section 7.6 we present an empirical evaluation of three multi-class AUC measures on the task of predicting one of seven digestive cancers. Finally, in Section 7.7 we provide concluding remarks on the work and some interesting future directions.

7.2 Prior Work on Multi-Class AUC

Prior work on extending AUC to the multi-class setting has focused on both the theoretical aspects of the problem and producing useable measures for real world problems. Interestingly, one of the earliest works in this area was a theoretical piece by Srinivasan (1999) who proved which classifiers may be optimal in an n-dimensional ROC space. Given a set of possible hard-labeling multi-class classifiers, it was shown that regardless of the choice of misclassification cost matrix, the optimal classifier lies on the convex hull of the n-dimensional ROC "surface". This is an extension of a known property of AUC that was shown by Provost and Fawcett (1997); that is, we may consider only those classifiers on the convex hull of the ROC curve regardless of the misclassification costs. While Srinivasan (1999) did not suggest how one would construct such an n-dimensional ROC space, the contributions were useful for future work.

A reasonable notion for the construction of a multi-class analog of AUC is that if in the two-class case we integrate under the ROC curve, then for the K-class case we should integrate under the ROC surface. This resulted in work on computing the volume under the ROC surface (VUS), though there is a disagreement on exactly how one should construct an ROC surface. In the two-class case, an ROC curve is plotted using the true positive rate and false positive rate values that are derived from the 2×2 confusion matrix. In general, a problem with K classes has a $K \times K$ confusion matrix from which we would construct the ROC surface. Two schools of thought arose on how to construct an ROC surface. Mossman (1999) believed that one needed only K dimensions for construction of the ROC surface, while Ferri et al. (2003) believed that $K(K-1)$-dimensions were necessary.

While a VUS-based approach is a reasonable extension to AUC, it suffers greatly from both computational complexity and interpretability. Both the construction of the ROC surface and computation of its volume are computationally intense problems. Lane (2000) notes that finding the convex hull of N points in d dimensions requires $O(N \log N + N^{\lfloor \frac{d}{2} \rfloor})$ time. This makes finding the ROC surface itself challenging for problems with even a moderate number of classes and instances. Be-

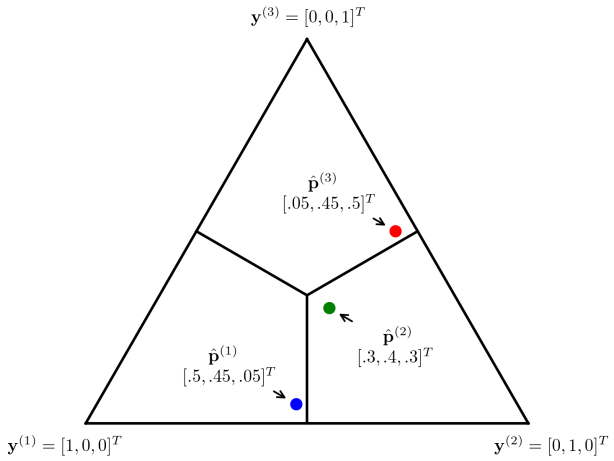


Figure 7.1: The M measure proposed by (Hand and Till, 2001) can yield a result much less than 1 even when a model assigns the correct label the highest probability on every example. In this figure, the larger outside equilateral triangle is the space of all possible model outputs and is known as the 2-simplex. Consider three model predictions plotted on the 2-simplex, $\hat{\mathbf{p}}^{(1)}$, $\hat{\mathbf{p}}^{(2)}$, and $\hat{\mathbf{p}}^{(3)}$, belonging to classes 1, 2, and 3 respectively. We divide the simplex using the argmax partitioning, that is, a prediction is assigned to the class for which it has the highest probability. The points are separable and correctly classified, however, M returns a value of 0.67 for this example, suggesting the model quality is much closer to random guessing than it is to perfect performance. This behavior is also true of the metric proposed by Provost and Domingos (2000).

cause of this, both Mossman (1999) and Ferri et al. (2003) choose to approximate the points on the ROC surface, which ultimately leads to inexact and underestimated volume. Further, even with an exact computation of volume, VUS no longer adheres to the same scale that AUC does, namely when AUC is 1 a classifier is perfect and when AUC is 0.5 it is equivalent to random guessing. VUS-based approaches have scales that get increasingly smaller as the number of classes grows and this makes interpreting how good a multi-class model is with VUS a challenge.

Perhaps it is for these reasons that the most widely used multi-class AUC approach is not VUS-based but rather an average of pairwise AUCs amongst the k classes. Hand and Till (2001) propose the measure M, an easy to compute and

class-skew insensitive performance measure for multi-class problems. However, M loses many of the properties that we believe are crucial for successful use and interpretation. Most importantly M can return values much less than 1 even when all points are correctly labeled. Consider the example in Figure 7.1 where three predictions, $\hat{\mathbf{p}}^{(1)}$, $\hat{\mathbf{p}}^{(2)}$, and $\hat{\mathbf{p}}^{(3)}$, are all correctly classified using the standard argmax rule. For each pair of classes, i and j, M considers all points whose true label is i or j, and computes the AUC amongst these instances twice, once with the ith component considered positive, and once with the jth component considered positive. These two calculations can yield different results and thus in cases such as Figure 7.1, M can return a score as low as 0.67 even though the points are perfectly labeled. Finally, M loses the elegance of a simple probabilistic interpretation as it is no longer equivalent with the U-statistic. That is, M is not the probability that two random instances will be ranked correctly.

While not nearly as widely used, Provost and Domingos (2000) proposed another method of extending AUC to the multi-class domain. Their approach performs a weighted average of K one-versus-all calculations of AUC for each of the individual class probabilities. However, because this approach weighs each individual AUC calculation by its class weight, it is inherently sensitive to class skew and thus violates Property 3. Additionally, like M it does not satisfy Property 1 and can return values less than 1 even when all examples would be accurately labeled using the argmax rule. Using the example in Figure 7.1, the method proposed by Provost and Domingos (2000) would also return a value of 0.67.

7.3 Background

Here we provide background material necessary for our derivation of AUC_μ. First, in Section 7.3 we discuss the relationship of AUC and the aforementioned Mann-Whitney U-statistic. The U-statistic is a metric based on the ranking of probabilistic model predictions from the two-class case. We then discuss in Section 7.3 how the probabilistic predictions of a model differs when there are more than 2 classes as multi-class predictions are specified as categorical distributions. Finally, in Section

7.3 we discuss partition matrices and decision boundaries, two tools that we use eventually use to rank categorical distributions.

AUC and the Mann-Whitney U-Statistic

True to its moniker, AUC is most commonly understood as an integration under the ROC curve. The Mann-Whitney U-statistic relationship shows a probabilistic interpretation of AUC. That is, AUC is the probability that a random instance whose label is positive will receive a higher ranking than a random instance whose label is negative. Let $\hat{Y}^+$ and $\hat{Y}^-$ represent the sets of model predictions for positive and negative instances respectively (e.g. if $\hat{y}^{(i)} \in \hat{Y}^+$, then $\hat{y}^{(i)}$ is some probability in $[0, 1]$, and the true label $y^{(i)}$ for instance $\mathbf{x}^{(i)}$ is positive). Further, let $n_+ = |\hat{Y}^+|$ and $n_- = |\hat{Y}^-|$ be the number of positive and negative instances respectively. Then we can calculate AUC as specified in Equation 7.1,

$$\text{AUC} = U = \frac{1}{n_+ n_-} \sum_{\hat{y}^{(i)} \in \hat{Y}^+} \sum_{\hat{y}^{(j)} \in \hat{Y}^-} \tilde{I}(\hat{y}^{(i)} - \hat{y}^{(j)}), \tag{7.1}$$

where $\tilde{I}(\cdot)$ is a modified indicator function that returns 1 if the argument is positive, 0 if the argument is negative, and 0.5 if the argument is 0.

Multi-Class Classification Models and Predictions

Whereas binary classification problems are concerned with labeling an instance as one of two categories, we call a task where an instance can belong to one of K categories a multi-class classification problem.

Definition 7.1. *$\mathcal{M}$ is a **multi-class model** over a domain $\mathcal{X}$ of possible examples that maps each $\mathbf{x} \in \mathcal{X}$ to a categorical distribution $\hat{\mathbf{p}} = [\hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_K]^\top$, where $\hat{\mathbf{p}}_j$ is the probability $\mathbf{x}$ belongs to category j. The domain of possible model predictions for a task with K classes is described by the $(K-1)$—simplex, Δ_{K-1}.* Figure 7.1 *shows Δ_2 along with 3 different $\hat{\mathbf{p}}$ model outputs.*

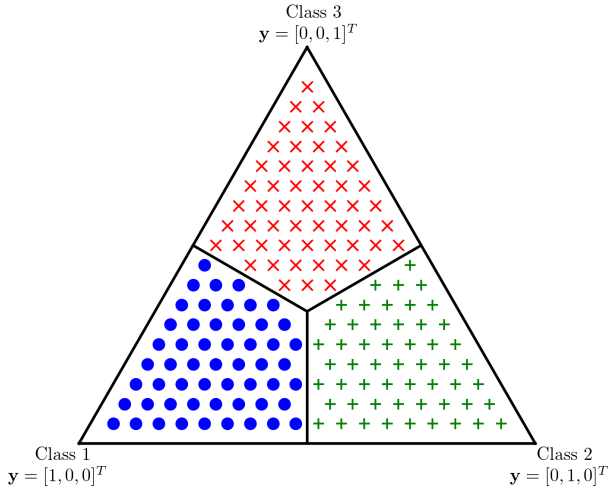


Figure 7.2: A partitioning of the 2-simplex, Δ_2, for a 3-class classification problem. Here we show the argmax partitioning, $\tilde{A}$, and the decision boundaries it induces. The regions with blue circles, green pluses, and red crosses are assigned to classes 1, 2, and 3 respectively.

While in two-class tasks a scalar threshold is used to map $\hat{\mathbf{p}}$ to a label, we need more complex tools for a multi-class task. With $K > 2$ classes, we now must define a partitioning of the $(K-1)-$simplex that maps a categorical distribution, $\hat{\mathbf{p}}$, to a hard label. That is, we divide Δ_{K-1} into K regions corresponding to values of $\hat{\mathbf{p}}$ that map to each of the K labels. Consider Figure 7.2 where we demonstrate what a partitioning of the 2-simplex looks like for a classification task with three classes.

Partition Matrices and Decision Boundaries

Recall that in Equation 7.1, the U-statistic computation involves ranking of predictions for two instances of different classes. Thus, extending the U-statistic to $K > 2$ classes requires some way of ranking the categorical distributions outputted by a multi-class model. We propose the use of a partition matrix, which divides the probability space of model outputs into distinct labeling regions. The partition matrix is

analogous to a threshold value in the two-class case and it has been shown that the two-class threshold can be derived from a 2×2 partition matrix (O'Brien et al., 2008). The misclassification cost matrix, that specifies the cost of mislabeling an instance of one label as another, is in fact a partition matrix. In general, a partition matrix is any matrix that divides the probability space into labeling regions for each of the K classes. We note the connection between the partition matrix and the decision boundaries learned by a linear-kernel multi-class SVM (Weston and Watkins, 1999). Both the linear-kernel multi-class SVM and the partition matrix produce K regions, each corresponding to a class. However, a linear-kernel multi-class SVM divides the feature space whereas the partition matrix divides a probablity space, Δ_{K-1}. Moreover, a partition matrix has at least one *equal-risk point*, where all classes are equally likely, whereas a multi-class SVM may produce solutions with no *equal-risk point*. Here we present background on partition matrices and their relationship with calculating AUC. The work presented by O'Brien et al. (2008) relies heavily on partition matrices, and their study provides many useful properties, proofs, and definitions. We restate some of their results (Definitions 7.2 and 7.3) as they are useful building blocks for our work.

Definition 7.2. ***Partition matrix:*** *Let A be a $K \times K$ matrix and let $A_{i,j}$ be the cost of classifying an instance as class i when its true class is j. Then A defines a partition on the $(K-1)$—simplex and induces decision boundaries between the K classes.*

Further, as shown in (O'Brien et al., 2008), any partition matrix A can be expressed by some other matrix A' with the properties $A'_{i,i} = 0 \; \forall i$ and $A'_{i,j} \neq 0 \; \forall i \neq j$. From here forward, when referring to a partition matrix we assume it is in this form with all diagonal entries zero.

Definition 7.3. ***Decision boundary:*** *A decision boundary between class i and class j, $i \neq j$, is the hyperplane that separates the two classes in Δ_{K-1}. The decision boundary is calculated using the partition matrix to solve for the hyperplane of solutions that have*

equal costs if assigned to class i *or class* j*.*

$$\sum_{k=1}^{K} A_{i,k}\hat{\mathbf{p}}_k = \sum_{k=1}^{K} A_{j,k}\hat{\mathbf{p}}_k \tag{7.2}$$

Here, $\hat{\mathbf{p}}_k$ *is the* k*th element of the categorical distribution* $\hat{\mathbf{p}}$*, i.e., the probability that the instance belongs to class* k*.*

An equivalent formulation of Equation 7.2 can use dot products and may be written as $A_{i,\cdot}\hat{\mathbf{p}} = A_{j,\cdot}\hat{\mathbf{p}}$.

Definition 7.4. ***Argmax partition matrix,*** $\tilde{A}$***:*** *When the costs in a partition matrix are 1 everywhere except the diagonal where they are 0, we call this the argmax partition matrix. It is so named because the label it assigns to any prediction,* $\hat{\mathbf{p}}$*, is* $\arg\max_k \hat{\mathbf{p}}_k$*. Because we reference this heavily in this work, we give it a special identifier:* $\tilde{A}$*.*

Figure 7.3 shows how the choice of partition matrix can change not just the label of a point, but in fact reverse the orientation of two points. By this we mean that two points that are both correctly labeled (correctly oriented with respect to the decision boundary) with one partition matrix, can become incorrectly labeled with another partition matrix.

7.4 Algorithm Derivation

In this section we derive the formula for AUC_μ. We wish for AUC_μ to be a multi-class extension of the U-statistic presented in Equation 7.1. Thus, AUC_μ must compute the probability that two random instances from different classes are ranked correctly by a model. However, recall that in multi-class classification our model output $\hat{\mathbf{p}}$ is a categorical distribution which makes extending the concept of ranking unclear. What does it mean for one categorical distribution to be of a "higher rank" than another? We must provide some means to map a categorical distribution to a scalar value which can be used for ranking. In Section 7.4 we demonstrate how

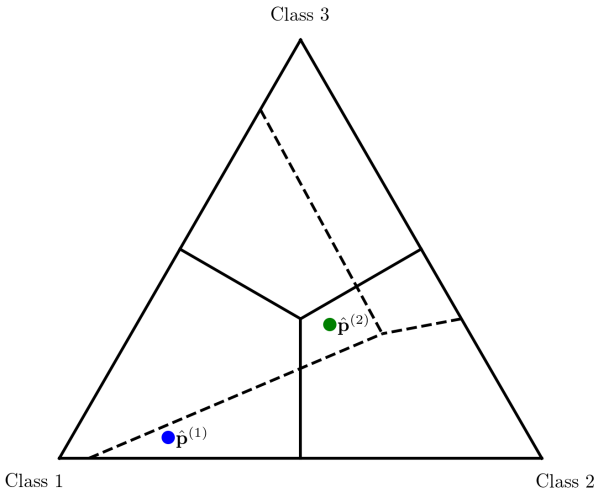


Figure 7.3: The choice of partition matrix can reverse the labeling of two points. Consider first the argmax partitioning of Δ_2, using $\tilde{A}$, shown in solid lines. Then $\hat{\mathbf{p}}^{(1)}$ and $\hat{\mathbf{p}}^{(2)}$ are assigned to the correct classes, 1 and 2 respectively. Now consider an alternative partitioning shown in dashed lines. Now $\hat{\mathbf{p}}^{(1)}$ is assigned to class 2, while $\hat{\mathbf{p}}^{(2)}$ is assigned to class 1. The first choice of partition matrix correctly labels both points, while the second choice of partition matrix incorrectly labels both points. It is also possible to choose a partition matrix that labels one point correctly and one point incorrectly.

such rankings can be performed using a partition matrix. Then, in Section 7.4 we use this method of ranking to derive the expression for AUC_μ.

Ranking Categorical Distributions

The intuition behind our approach is most easily described through an analogy to standard linear kernel support vector machines (SVMs). A linear SVM generates a decision hyperplane which divides the feature space into two regions, one where instances are labeled as positive and the other negative. The further an instance is from the decision hyperplane the more confident the SVM is in its label. In this way, model confidence for an SVM is measured by the orthogonal distance of an instance to the decision hyperplane. Similarly, when ranking two instances from

different classes, we use the decision hyperplane between those two classes that is derived from the partition matrix.

Recall that Equation 7.2 describes the decision boundary as a set of categorical distributions for which the expected cost of labeling an instance as class i or class j is equal. Let $\mathbf{v}^T = A_{i,\cdot} - A_{j,\cdot}$, then $\mathbf{v}^T\hat{\mathbf{p}} = 0$ is the equation of the hyperplane and an equivalent formulation of Equation 7.2. This decision boundary divides our $(K-1)-$simplex into two regions, one where we are more confident to label an instance class i and one where we are more confident to label an instance class j. If $\mathbf{v}^T\hat{\mathbf{p}}$ is positive, then we see it is more costly to assign the label of class i than class j. The more positive $\mathbf{v}^T\hat{\mathbf{p}}$, the larger the difference in cost, and therefore the more favorable a labeling class j becomes. Therefore, $\mathbf{v}^T$ provides a way to rank various points in terms of their cost difference between assignments of class i and j. It should be noted that $\mathbf{v}^T$ is the orthogonal vector to our equal-cost hyperplane and $\mathbf{v}^T\hat{\mathbf{p}}$ is proportional to the length of the projection of $\hat{\mathbf{p}}$ onto $\mathbf{v}^T$. We are in essence calculating an unscaled orthogonal distance of our prediction, $\hat{\mathbf{p}}$, to the equal cost hyperplane. This scalar value provides the ranking that is the critical piece that is needed to extend the indicator function I in Equation 7.1 and thus derive multi-class AUC.

We now show how to determine if two model outputs are ranked correctly in a multi-class problem. Let $\hat{\mathbf{p}}^{(i)}$ and $\hat{\mathbf{p}}^{(j)}$ be the categorical output of our model for two instances $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$. Further, without loss of generality, let the true classes of $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ be classes 1 and 2 such that $\mathbf{y}^{(i)} = [1, 0, \ldots, 0]^T$ and $\mathbf{y}^{(j)} = [0, 1, \ldots 0]^T$ are the true class vertices on the $(K-1)-$simplex for these two instances. Let A be our partition matrix. We first calculate our normal vector to our decision boundary as $\mathbf{v}^T = A_{1,\cdot} - A_{2,\cdot}$. Note that $\mathbf{v}^T\mathbf{y}^{(1)}$ and $\mathbf{v}^T\mathbf{y}^{(2)}$ are the unscaled distances of our class vertices from the hyperplane. This provides us the "correct" orientation of two points projected onto $\mathbf{v}^T$. That is, if $\mathbf{v}^T\mathbf{y}^{(i)} > \mathbf{v}^T\mathbf{y}^{(j)}$, then if $\mathbf{v}^T\hat{\mathbf{p}}_i > \mathbf{v}^T\hat{\mathbf{p}}_j$ we know that our model correctly ranked the two points. Figure 7.4 illustrates an example of this projection and how we can use our partition-matrix-derived decision boundary to induce rankings on multi-class predictions. We can efficiently compute if two points are ranked correctly through the introduction of an orientation function.

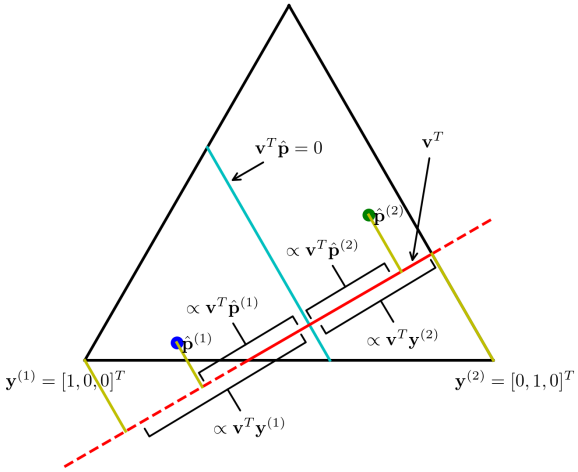


Figure 7.4: A depiction of how the partition-matrix-derived decision boundary, $\mathbf{v}^\mathsf{T}\hat{\mathbf{p}} = 0$, (in cyan) can be used to induce a ranking of categorical distributions. The normal vector to the decision hyperplane, $\mathbf{v}^\mathsf{T}$ (shown in red), provides a means to rank points in the simplex. The dot product of $\mathbf{v}^\mathsf{T}$ with the true labels and model outputs form an un-normalized projection onto $\mathbf{v}^\mathsf{T}$ and thus a means of ranking categorical distributions. The ranking of $\hat{\mathbf{p}}_1$ and $\hat{\mathbf{p}}_2$ is correct here as their orientation with respect to the decision boundary is the same as the orientation of their labels $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$.

Definition 7.5. *An orientation function,* O*, returns a positive value if the predictions are ranked correctly, a negative value if they are ranked incorrectly, and 0 if their rank is tied.*

$$O(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}, \hat{\mathbf{p}}^{(i)}, \hat{\mathbf{p}}^{(j)}, \mathbf{v}^\mathsf{T}) = (\mathbf{v}^\mathsf{T}(\mathbf{y}^{(i)} - \mathbf{y}^{(j)}))(\mathbf{v}^\mathsf{T}(\hat{\mathbf{p}}^{(i)} - \hat{\mathbf{p}}^{(j)})) \tag{7.3}$$

$\mathbf{AUC}_\mu$

Here we detail our derivation of AUC_μ as an extension of the U-statistic such that we satisfy Properties 1-3 listed in Section 7.1. We restate the two-class U-statistic, Equation 7.1, for reference.

$$U = \frac{1}{n_+ n_-} \sum_{\hat{y}^{(i)} \in \hat{Y}^+} \sum_{\hat{y}^{(j)} \in \hat{Y}^-} \tilde{I}(\hat{y}^{(i)} - \hat{y}^{(j)}),$$

We begin by modifying the indicator function, $\tilde{I}$, so that it is compatible with multi-class model outputs. Recall that $\tilde{I}$ returns 1 if two instances are ordered correctly, 0 if they are ordered incorrectly, and 0.5 if there is a tie in their rank. We utilize the orientation function described in Equation 7.3 as the new argument for $\tilde{I}$. Now, for two instances indexed by i and j from different classes, $\tilde{I}O(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}, \hat{\mathbf{p}}^{(i)}, \hat{\mathbf{p}}^{(j)}, \mathbf{v}^T)$, will indicate if the two instances are ordered correctly, incorrectly, or tied. However, we note that O requires the two class decision hyperplane normal vector, $\mathbf{v}^T$, as an argument. What is the right choice of $\mathbf{v}^T$? To answer this question we refer to Property 1, that if all instances are labeled according the their highest probability, then AUC_μ should return a score of 1. We note that this division of the $(K-1)$-simplex is exactly the argmax partitioning, and thus we argue that $\tilde{A}$ is the appropriate partition matrix to use in AUC_μ . We later show in the proof in Section 7.5, if there is no *a priori* preference of a particular partition matrix, $\tilde{A}$ is the appropriate choice. We further show in Section 7.5 how to compute an alternative formulation of AUC_μ when there is a preference for a particular partition matrix. From here forward, unless otherwise specified we assume that the decision boundaries used when calculating AUC_μ are derived from $\tilde{A}$.

For a problem with K classes, let us first consider, without loss of generality, two classes $i < j \leqslant K$. Similar to (Hand and Till, 2001), we aim to construct a separability measure between i and j; we call this measure $S(i,j)$. Let $D(i)$, and $D(j)$ be the sets of all instances whose true label is i and j respectively. Further, let n_i, n_j be the number of instances in each set respectively. Then we define,

$$S(i,j) = \frac{1}{n_i n_j} \sum_{a \in D(i), b \in D(j)} \tilde{I}(O(\mathbf{y}^{(a)}, \mathbf{y}^{(b)}, \hat{\mathbf{p}}^{(a)}, \hat{\mathbf{p}}^{(b)}, \mathbf{v}^T)).$$

If $K = 2$, then $S(i,j)$ reduces to the U-statistic, and thus AUC. We discuss and prove this equivalence in Section 7.5.

Next we turn our attention to Property 3, that AUC_μ should be insensitive to class skew. While in the two-class case if two instances from different classes are randomly selected we always get equal representation from both classes (one from each class). However, if instances are randomly selected from different classes when

Table 7.1: Comparison of the various multi-class metrics discussed in this work: VUS-3 (Mossman), VUS (Ferri), H&T (Hand and Till), P&D (Provost and Domingos), and AUC_μ. A $\surd$ indicates a proven property, a $\times$ indicates a proven absence of a property, and a "?" indicates an unknown. Not only is AUC_μ the only metric to preserve Properties 1, 2, and 3, but also its time complexity is as fast or faster than all other methods.

	VUS-3	VUS	H&T	P&D	AUC_μ
Perfect = 1	$\surd$	$\times$	$\times$	$\times$	$\surd$
Random = 0.5	$\times$	$\times$	$\surd$	$\surd$	$\surd$
Skew Insensitive	?	?	$\surd$	$\times$	$\surd$
Time Complexity	exponential	exponential	polynomial	polynomial	polynomial

$K > 2$, we are more likely to sample classes with more instances. For this reason, we construct AUC_μ such that each choice of i and j is weighted equally. This approach is inspired by how (Hand and Till, 2001) construct their measure M such that it is also class skew insensitive. The final formulation for AUC_μ is as follows.

$$AUC_\mu = \frac{2}{K(K-1)} \sum_{i<j} S(i,j) \tag{7.4}$$

Comparison of Algorithms

In Table 7.1 we present a comparison of AUC_μ to the four other multi-class classification metrics presented in this work (Mossman, 1999; Ferri et al., 2003; Hand and Till, 2001; Provost and Domingos, 2000). Of these five metrics, AUC_μ is the only one to preserve the three critical properties of AUC: 1) a perfect classification results in a score of 1, 2) random guessing results in a score of 0.5, and 3) skew insensitivity. Moreover, AUC_μ has time complexity that is equal or faster than all other algorithms. In Section 7.5 we present a variety of theoretical analyses and proofs for these claims.

7.5 Algorithm Analysis and Extensions

In this section we provide several properties of AUC_μ, as well as several extensions for special cases. In calculating AUC_μ we use the argmax partition matrix, $\tilde{A}$, yet a partition matrix is notably absent in the two-class case for calculation of AUC. Thus, we provide a proof that the calculation of AUC with only two classes is a special case that does not require a partition matrix. Further, we present a corollary of this theorem showing that AUC_μ simplifies to the standard two-class AUC when there are only two classes. We then present a proof that when there are $K > 2$ classes a partition matrix is required. In Appendix A, we provide proofs that AUC_μ satisfies Properties 1, 2, and 3. Finally, we present two special cases of AUC_μ for domains in which there is a strong concern about misclassification costs and/or skew.

Partition Matrices and AUC

The reader has likely noted that the requirement of a partition matrix seems unnatural, since the standard AUC measure does not require any information about a threshold or partition matrix (the former derivable from the latter). Recall though, as demonstrated in Figure 7.3, that the choice of partition matrix influences the ranking of points and thus AUC_μ is sensitive to the choice of partition matrix. In Theorem 7.6 we claim that for two-class classification problems we do not require a partition matrix as the the relative ranking of two points is indifferent to choice of partition matrix. This theorem is proved in Appendix A.

Theorem 7.6. *Let $\mathcal{M}$ be a model trained to perform a binary classification task. Let A be a 2×2 partition matrix with diagonal zeros and all other entries positive. Then A has no effect on the ranking of predictions from $\mathcal{M}$*

A desirable corollary of Theorem 7.6 is that AUC_μ simplifies to the two-class AUC presented in Equation 7.1. This corollary is proved in Appendix A.

Corollary 7.7. *When $K = 2$, AUC_μ simplifies to the Mann-Whitney U-statistic formulation of AUC.*

When there are more than 2 classes, the choice of partition matrix can impact the ranking of instances and thus it is necessary to specify the partition matrix in calculating AUC_μ. In Figure 7.3 we showed that the choice of a partition matrix can affect how two instances are ranked in the 3-class case. We claim in Theorem 7.8 that for any $K > 2$ we must provide a partition matrix to rank predictions and we prove this theorem and Appendix A.

Theorem 7.8. *Let $\mathcal{M}$ be a model trained for a multi-class classification task with $K > 2$ classes. Then the ranking of predictions from $\mathcal{M}$ is not independent of the choice of $K \times K$ partition matrix, hence calculating AUC_μ requires a partition matrix.*

The Argmax Partition Matrix

In Section 7.4 we argue that $\tilde{A}$ is a good choice as it satisfies Property 1. Here we claim that if there is no *a priori* preference for choice of partition matrix, then $\tilde{A}$ is the appropriate choice as it is the expectation over all possible partition matrices. Theorem 7.9, states that the expectation over all partition matrices, uniformly distributed, is the argmax partition matrix, $\tilde{A}$. We prove this theorem in Appendix A.

Theorem 7.9. *The expectation over all partition matrices, uniformly distributed, for a task with K classes is the argmax partition matrix, $\tilde{A}$, where $\tilde{A}_{i,i} = 0 \; \forall i$ and $\tilde{A}_{i,j} = 1 \; \forall i \neq j$.*

Unsurprisingly, choosing uniform misclassification costs results in an argmax partitioning of Δ_{K-1}. That is, when we have no knowledge of misclassification costs, we label an instance with the category which contains the highest probability in $\hat{\mathbf{p}}$.

Time Complexity of AUC_μ

The time complexity of AUC_μ is $O(Kn \log n)$ when using the argmax partition matrix, $\tilde{A}$, where K is the number of classes, and n is the number of instances. This is equivalent to the time complexity of M proposed by (Hand and Till, 2001). While (Fawcett, 2006) claims that M has a complexity of $O(K^2 n \log n)$, we show that the

bound is in fact tighter and that the complexity is $O(Kn \log n)$. The derivation of both of these results is in Appendix A.

Extensions of AUC_μ

Motivated by the initial important properties we listed for AUC, we use the argmax partition matrix, $\tilde{A}$, in the calculation of AUC_μ. While we believe for most cases our initial presentation of AUC_μ is a suitable measure, there are exceptions. It is not uncommon for domains to have highly skewed class distributions or unequal misclassification costs between classes. AUC_μ can be easily modified to accommodate both of these scenarios and thus we present two such extensions. In Section 7.5 we show that for tasks with unequal misclassification costs one can incorporate an alternative partition matrix when calculating AUC_μ. In Section 7.5 we show an alternative formulation of AUC_μ that can account for class skew in problems where this may be desirable in the performance measure.

Use of an Alternative Partition Matrix

Recall that in the calculation of AUC_μ we rely on the orientation function presented in Equation 7.3. This function ranks two instances based on the two-class decision boundary derived from the partition matrix. In the standard calculation of AUC_μ, we use $\tilde{A}$ to perform ranking. Thus, we note that it is straightforward to use an alternative partition matrix in this calculation as well. (O'Brien et al., 2008) note that if the partition matrix is the misclassification cost matrix for a particular domain, then instances will be labeled in such a manner as to minimize the expected cost for a given instance. Therefore, if the misclassification cost matrix is well established for a particular domain, it may be appropriate then to use that as the partition matrix in place of $\tilde{A}$. We show in Appendix A that using an alternative partition matrix has time complexity to $O(Kn(K + \log n))$.

Incorporating Class Skew into AUC_μ

Like Hand and Till (2001), we believe that a multi-class extension of AUC should be insensitive to class skew and thus AUC_μ is designed to remove the effects of any skew. However, there are tasks with heavy class skew where this property may become problematic. Therefore, we provide here an alternative formulation of AUC_μ that incorporates a weight for each pair of classes. For a task with K classes, let $i < j \leqslant K$ be the class labels for two different classes. Let $n = n_1 + \ldots n_K$ be the number of total instances and number of instances in each class and let $\tilde{n} = \sum_{i<j} n_i n_j$. Finally, let $w_{i,j} = \frac{n_i n_j}{\tilde{n}}$ be the weight assigned for classes i and j. Here, $w_{i,j}$ is the probability that a pair of instances randomly selected from different classes belong to class i and class j. Then, we formulate the class skew sensitive formulation, AUC_μ^S, as follows.

$$AUC_\mu^S = \sum_{i<j} w_{i,j} S(i,j) \tag{7.5}$$

This alternative formulation of incorporates the natural class skew in the dataset as the weighting factor for each separability function, $S(i,j)$. We note that while choosing $w_{i,j} = \frac{n_i n_j}{\tilde{n}}$ is a natural option, any weighting scheme may be used so long as $\sum_{i<j} w_{i,j} = 1$ so that AUC_μ^S is still bounded between 0 and 1.

7.6 Empirical Comparisons

This section presents an empirical investigation of the three computationally tractable multi-class measures: H&T (Hand and Till, 2001), P&D (Provost and Domingos, 2000), and AUC_μ. Consider the task of predicting digestive cancer in one of seven locations (esophagus, stomach, colon, rectum/anus, liver, gallbladder, and pancreas). As discussed in Section 7.2, H&T and P&D are prone to calling a pair of correct predictions a tie, and thus under-reporting the multi-class AUC. For the task of predicting digestive cancer location, AUC_μ returns a score that is significantly higher than H&T (p=2.3e-32) and P&D (p=7.7e-34).

Table 7.2: Dataset details for patients included in the digestive cancer multi-class study. Note that ICD-9 codes 152, 158, and 159 had no patients with these diagnoses.

ICD-9 Code	Location	Count (%)
150	Esophagus	1,253 (7.7%)
151	Stomach	1,171 (7.2%)
153	Colon	7,899 (48.5%)
154	Rectum/Anus	2,574 (15.8%)
155	Liver	922 (5.7%)
156	Gallbladder	487 (3.0%)
157	Pancreas	1,989 (12.2%)
Total	All	16,292 (100%)

Experimental Design

We consider the task of predicting the location of digestive cancers one month prior to diagnosis. The study included patients based on the diagnoses present on their patient record in the digestive cancer chapter of the ICD-9 hierarchy (codes 150-159). For a particular digestive cancer code, for example ICD-9 150, patients were labeled as positive for that diagnosis if they both satisfied the rule-of-2 and had no entries of other digestive cancers on their record prior to the diagnosis of interest (ICD-9 150 in the example). Patient EHR data was sampled from the same dataset used in Chapter 4 and records were summarized into feature vectors using the procedure detailed in that chapter. Table 7.2 contains details of the patient population used in this study.

The following model training and evaluation procedure was repeated 30 times and consisted of a data splitting phase, a model training phase, and a model evaluation phase. The 16,292 patient samples were split into a training set (70%) and testing set (30%) via a random assignment that preserved the class percentages in both datasets. A random forest model with 500 trees was trained using the same parameters used for the models in Chapter 4. Recall that random forests naturally extend to multi-class tasks, and thus there was no need to make adjustments when

switching from a binary to multi-class domain. Probability vectors, in the form of categorical distributions, were predicted for the testing data. Scores were then computed for the three computationally tractable multi-class AUC measures: H&T, P&D, and AUC_μ.

Experimental Results

The mean scores across the 30 repetitions were 0.623 for H&T, 0.612 for P&D, and 0.662 for AUC_μ. ANOVA with repeated measures was used as a statistical test to calculate if there was a significant difference amongst the three means. The p-value of 2.1e-58 was below the threshold of $\alpha = 0.05$. To test if the mean score of AUC_μ was significantly higher than the mean score of H&T or P&D, two two-sided paired t-tests were performed. Significance was tested at $\alpha = 0.025$ which was chosen using a Bonferroni correction to account for the two tests performed such that the experiment-wide $\alpha = 0.05$. The mean of AUC_μ was significantly higher than both H&T (p2.3e-32) and P&D (p=7.7e-34).

7.7 Conclusion

In this chapter we introduce AUC_μ, a multi-class classification performance measure that aims to maintain the many desirable properties of AUC. Prior work focused on multi-class analysis of volume under the ROC surface has proven to be computationally intensive and requires stochastic sampling methods for computation (Ferri et al., 2003; Mossman, 1999; Srinivasan, 1999; Lane, 2000). These measures are not well suited to large datasets or tasks such as hyper-parameter tuning that require fast calculation of the model quality. The most popular approach as of the time of this writing, that does not utilize an ROC surface, is the measure M introduced by Hand and Till (2001). However, as we demonstrated in Figure 7.1, M can return values much less than 1 even when all predictions are separable and would be labeled correctly following the common argmax labeling rule. Thus, we employ an alternative approach to multi-class AUC that is motivated by the relationship of

AUC and the Mann Whitney U-statistic, and through this relationship we derive AUC_μ, a measure that is easy to compute and interpret.

We provide several theoretical observations of AUC_μ and some extensions for domains with particular concern regarding misclassification costs and class skew. We prove in Theorems 7.6 and 7.8 that while a partition matrix is not needed for two-class AUC, it is needed for ranking model outputs for more than two classes. As the argmax labeling rule is common in multi-class problems, we suggest that the use of the argmax partition matrix is the appropriate choice for most tasks and thus we use this in our computation of AUC_μ. However, we also note in Section 7.5 that an alternative partition matrix can and should be used in domains with known unequal misclassification costs. We additionally present in Section 7.5 an alternative of AUC_μ that can intentionally incorporate class skew where this may provide a more sensible evaluation of the model performance.

An empirical comparison of AUC_μ , H&T, and P&D was performed in Section 7.6 on the task of evaluating a multi-class digestive cancer model. The significantly higher mean for AUC_μ as compared to H&T and P&D supports the claim that both H&T and P&D can under-score an algorithm. We assert that this is due to H&T and P&D incorrectly calling a pair of instances a tie, even when the instances are correctly labeled, as shown in the example in Figure 7.1.

There are several exciting avenues for analysis of AUC_μ. While empirical confidence intervals and p-values can be calculated through a bootstrap approach, it would be interesting to see if there exist closed-form solutions for AUC_μ as they do for the binary AUC. Additionally, we note that the calculation of AUC_μ involves two dot products and that if either of these dot-products are 0 then the ranking of two instances is tied. This could become troublesome for tasks with very high numbers of classes as the probability of orthogonality between two random vectors increases with dimension. Whether AUC_μ is susceptible to this or not is a matter for future exploration and could suggest that the U-statistic is not a reliable measure of model performance for tasks with large numbers of classes.

By naturally extending the Mann-Whitney U-statistic, we both introduce a new method for computing multi-class AUC and provide several theoretical observa-

tions on how AUC_μ behaves in multi-class tasks. We believe that a renewed interest in performance metrics for multi-class machine learning models is warranted, as many interesting problems in machine learning are not binary class problems. Ultimately, we claim that AUC_μ is a fast, reliable and easy to interpret method for assessing the performance of a multi-class classification model.

Co-authorship and Acknowledgements

The work presented in Chapter 7 is the result of a collaboration with multiple researchers. This work was by the following authors and affiliations: Ross Kleiman[1], C. David Page[1], [1]University of Wisconsin - Madison. The authors acknowledge support from the National Library of Medicine (NLM) grant 5T15LM007359.

8 CONCLUSION

In this thesis we introduced high-throughput machine learning, an extension of one-off multi-label classification wherein each label requires special consideration for the training data. We presented groundwork for high-throughput machine learning with a particular focus on healthcare domain applications. High-throughput machine learning differs from typical multi-output tasks in that its goal is to utilize the trained models to learn larger scale meta-information about the individual predictive models. By constructing one model per prediction task we can combine both the predictions and feature importances of these models with ancillary domain information to identify trends that emerge across multiple prediction tasks. This dissertation addressed the opportunities, challenges, and limitations of performing high-throughput machine learning.

This thesis claims that high-throughput machine learning yields insights not gleaned from standard one-off or multi-label modeling. In Chapter 3 we began by presenting the procedure for predicting a single medical condition in Calciphylaxis. We motivated and described the need for high-throughput machine learning in Chapter 4 where we showed how predicting all diagnoses can identify both how prediction quality changes across the hierarchy of diseases as well as temporally. In Chapter 5 we considered the unique privacy challenges of healthcare data and our focus on large scale machine learning approaches, and presented a cryptographic approach to random forest model learning where multiple parties with siloed data wish to collaborate on learning a single model. We then presented an additional task for high-throughput machine learning in Chapter 6 where the learned feature importances of the models could be used to identify candidate epidemiological hypotheses for novel diagnostic uses of existing laboratory tests. Finally in Chapter 7, we explored prediction tasks where a multi-class approach is appropriate and showed that the current methodologies for evaluating performance are insufficient and hence we proposed a new means of evaluating such multi-class models. Therefore, this thesis has both introduced high-throughput machine as an efficient

method for constructing many thousands of unique predictive models, and showcased how high-throughput machine learning can provide unique insights when applied to the high-profile context of healthcare data.

REFERENCES

Backes, Michael, Pascal Berrang, Matthias Bieg, Roland Eils, Carl Herrmann, Mathias Humbert, and Irina Lehmann. 2017. Identifying personal DNA methylation profiles by genotype inference. In *2017 IEEE symposium on security and privacy, SP 2017, san jose, ca, usa, may 22-26, 2017*, 957–976.

Bamber, Donald. 1975. The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of Mathematical Psychology* 12(4):387–415.

Bhaskar, KR, DD O'Sullivan, H Opaskar-Hincman, and LM Reid. 1987. Lipids in airway secretions. *European Journal of Respiratory Diseases* 153:215–221.

Bornmann, Lutz, and Rüdiger Mutz. 2015. Growth rates of modern science: a bibliometric analysis based on the number of publications and cited references. *Journal of the Association for Information Science and Technology* 66(11):2215–2222.

Bost, Raphael, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. 2015. Machine learning classification over encrypted data. In *22nd annual network and distributed system security symposium, NDSS 2015, san diego, california, usa, february 8-11, 2015*.

Boutell, Matthew R, Jiebo Luo, Xipeng Shen, and Christopher M Brown. 2004. Learning multi-label scene classiÿcation. *Pattern Recognition* 37:1757–1771.

Boyd, Kendrick, Vítor Santos Costa, Jesse Davis, and C David Page. 2012. Unachievable Region in Precision-Recall Space and Its Effect on Empirical Evaluation. *Machine learning : proceedings of the International Conference. International Conference on Machine Learning* 2012:349. 1206.4667.

Breiman, Leo. 1984. *Classification and Regression Trees*. Belmont, California: Wadsworth International Group.

———. 2001. Random forests. *Machine Learning* 45(1):5–32.

Brickell, Justin, Donald E. Porter, Vitaly Shmatikov, and Emmett Witchel. 2007. Privacy-preserving remote diagnostics. In *Proceedings of the 2007 association of computing machinery conference on computer and communications security, CCS 2007, alexandria, virginia, usa, october 28-31, 2007*, 498–507.

Brisimi, Theodora S., Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch. Paschalidis, and Wei Shi. 2018. Federated learning of predictive models from federated electronic health records. *International Journal of Medical Informatics* 112: 59–67.

Caruana, Rich, and Alexandru Niculescu-Mizil. 2006. An empirical comparison of supervised learning algorithms. *Proceedings of the 23rd International Conference on Machine Learning* C(1):161–168.

Chao, Ting-Kuang. 2006. Cholesterol granuloma of the maxillary sinus. *European Archives of Oto-Rhino-Laryngology and Head & Neck* 263(6):592–597.

Charlson, M E, P Pompei, K L Ales, and C R MacKenzie. 1987. A new method of classifying prognostic comorbidity in longitudinal studies: development and validation. *Journal of Chronic Diseases* 40(5):373–83.

Che, Zhengping, David Kale, Wenzhe Li, Mohammad Taha Bahadori, and Yan Liu. 2015. Deep Computational Phenotyping. *Proceedings of the 21st International Conference on Knowledge Discovery and Data Mining* 507–516.

Chen, Tianqi, and Carlos Guestrin. 2016. XGBoost : Reliable Large-scale Tree Boosting System. In *Conference on knowledge discovery and data mining*. `1603.02754`.

Clare, Amanda, and Ross D King. 2001. Knowledge Discovery in Multi-label Phenotype Data. In *Proceedings of the 5th european conference on principles of data mining and knowledge discovery*, 42–53. PKDD '01, London, UK, UK: Springer-Verlag.

Coates, T, G S Kirkland, R B Dymock, B F Murphy, J K Brealey, T H Mathew, and A P Disney. 1998. Cutaneous necrosis from calcific uremic arteriolopathy. *American*

Journal of Kidney Diseases : the Official Journal of the National Kidney Foundation 32(3): 384–91.

Dawber, T R, G F Meadors, and F E Moore. 1951. Epidemiological approaches to heart disease: the Framingham Study. *American Journal of Public Health and the Nation's Health* 41(3):279–81.

De Cock, Martine, Rafael Dowsley, Caleb Horst, Raj Katti, Anderson Nascimento, Wing-Sea Poon, and Stacey Truex. 2017. Efficient and private scoring of decision trees, support vector machines and logistic regression models based on precomputation. *IEEE Transactions on Dependable and Secure Computing*.

Dietterich, Thomas G, and Ghulum Bakiri. 1995. Solving Multiclass Learning Problems via Error-Correcting Output Codes. Tech. Rep.

Dilek, Fatma Hüsniye, Muzaffer Kiris, and Serdar Ugras. 1997. Cholesterol granuloma of the maxillary sinus. *Rhinology* 35:140–141.

Doshi-Velez, Finale, and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv*.

Dwork, Cynthia, and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*.

eMEREGE. eMERGE.

Fawcett, Tom. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27(8):861–874.

Fernández-Delgado, Manuel, Eva Cernadas, Senén Barro, Dinani Amorim, and Dinani Amorim Fernández-Delgado. 2014. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research* 15:3133–3181.

Ferri, César, José Hernández-Orallo, and Miguel Angel Salido. 2003. Volume under the ROC Surface for Multi-class Problems. In *Proceedings of the 14th european conference on machine learning*, 108–120. Springer, Berlin, Heidelberg.

Fredrikson, Matthew, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. 2014. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. *Proceedings of the USENIX Security Symposium. UNIX Security Symposium*.

Gail, M H, L A Brinton, D P Byar, D K Corle, S B Green, C Schairer, and J J Mulvihill. 1989. Projecting individualized probabilities of developing breast cancer for white females who are being examined annually. *Journal of the National Cancer Institute* 81(24):1879–86.

Geurts, Pierre, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine Learning* 63(1):3–42.

Giacomelli, Irene, Somesh Jha, Ross Kleiman, David Page, and Kyonghwan Yoon. 2018. Privacy-preserving collaborative prediction using random forests (full version). *http://www.cs.wisc.edu/ dpage/PrivateForests.pdf*.

Gruber, Maja, Mirjam Christ-Crain, Daiana Stolz, Ulrich Keller, Christian MÃ¼ller, Roland Bingisser, Michael Tamm, Beat MÃ¼ller, and Philipp Schuetz. 2009. Prognostic impact of plasma lipids in patients with lower respiratory tract infections-an observational study. *Swiss Medical Weekly* 139:166–72.

Hand, David J., and Robert J. Till. 2001. A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. *Machine Learning* 45(2): 171–186.

Hanley, J A, and B J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* 143(1):29–36.

Hastie, Trevor, and Robert Tibshirani. 1998. Classification by pairwise coupling. *Annals of Statistics*.

Himes, Blanca E, Yi Dai, Isaac S Kohane, Scott T Weiss, and Marco F Ramoni. 2009. Prediction of chronic obstructive pulmonary disease (COPD) in asthma patients using electronic medical records. *Journal of the American Medical Informatics Association* 16(3):371–9.

Ho, Joyce C., Joydeep Ghosh, and Jimeng Sun. 2014. Marble. In *Proceedings of the 20th international conference on knowledge discovery and data mining*, 115–124. New York, New York, USA: ACM Press.

Hoerl, Arthur E., and Robert W. Kennard. 1970. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*. `9809069v1`.

Homer, Nils, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. 2008. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS genetics* 4(8):e1000167.

de Hoogh, Sebastiaan, Berry Schoenmakers, Ping Chen, and Harm op den Akker. 2014. Practical secure decision tree learning in a teletreatment application. In *Financial cryptography and data security - 18th international conference, barbados, march 3-7, 2014, revised selected papers*, 179–194.

Hsiao, Chun-Ju, Esther Hing, and Jill Ashman. 2014. Trends in electronic health record system use among office-based physicians: United States, 2007-2012. *National Health Statistics Reports* 75:1–18.

Huang, Sandy H, Paea LePendu, Srinivasan V Iyer, Ming Tai-Seale, David Carrell, and Nigam H Shah. 2014. Toward personalizing treatment for depression: predicting diagnosis and severity. *Journal of the American Medical Informatics Association* 21(6):1069–1075.

Hunter, John D. 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering* 9(3):90–95.

IWPC. 2009. Estimation of the Warfarin Dose with Clinical and Pharmacogenetic Data. *New England Journal of Medicine* 360(8):753–764.

Jagannathan, Geetha, Krishnan Pillaipakkamnatt, and Rebecca N. Wright. 2012. A practical differentially private random decision tree classifier. *Transactions on Data Privacy* 5(1):273–295.

Joye, Marc, and Benoît Libert. 2013. Efficient cryptosystems from 2^k-th power residue symbols. In *Advances in cryptology - EUROCRYPT 2013, 32nd annual international conference on the theory and applications of cryptographic techniques, athens, greece, may 26-30, 2013. proceedings*, 76–92.

Joye, Marc, and Fariborz Salehi. 2018. Private yet efficient decision tree evaluation. In *Data and applications security and privacy XXXII - 32nd annual IFIP WG 11.3 conference, dbsec 2018, bergamo, italy, july 16-18, 2018, proceedings*, 243–259.

Kerschbaum, Florian, and Orestis Terzidis. 2006. Filtering for private collaborative benchmarking. In *Emerging trends in information and communication security, international conference, ETRICS 2006, freiburg, germany, june 6-9, 2006, proceedings*, 409–422.

Kuang, Zhaobin, James Thomson, Michael Caldwell, Peggy Peissig, Ron Stewart, and David Page. 2016. Computational Drug Repositioning Using Continuous Self-Controlled Case Series. In *Proceedings of the 22nd international conference on knowledge discovery and data mining - kdd '16*, 491–500. New York, New York, USA: ACM Press.

Kuusisto, Finn, John Steill, Zhaobin Kuang, James Thomson, David Page, and Ron Stewart. 2017. A simple text mining approach for ranking pairwise associations in biomedical applications. *American Medical Informatics Association Summits on Translational Science Proceedings* 2017:166–174.

Lal, Avtar. 2000. Effect of a few histamine(1)-antagonists on blood glucose in patients of allergic rhinitis. *Indian Journal of Otolaryngology and Head and Neck Surgery* 52(2):193–195.

Lane, Terran. 2000. Position paper: Extensions of ROC analysis to multi-class domains. *Proceedings of 2000 International Conference on Machine Learning: Workshop on Cost-Sensitive Learning*.

Lantz, Eric. 2016. Machine learning for risk prediction and privacy in electronic health records. Ph.D. thesis, The University of Wisconsin-Madison.

Levey, Andrew S., Kai-Uwe Eckardt, Yusuke Tsukamoto, Adeera Levin, Josef Coresh, Jerome Rossert, Dick D.E. Zeeuw, Thomas H. Hostetter, Norbert Lameire, and Garabed Eknoyan. 2005. Definition and classification of chronic kidney disease: A position statement from Kidney Disease: Improving Global Outcomes (KDIGO). *Kidney International* 67(6):2089–2100.

Lindell, Yehuda, and Benny Pinkas. 2000. Privacy preserving data mining. In *Advances in cryptology - CRYPTO 2000, 20th annual international cryptology conference, santa barbara, california, usa, august 20-24, 2000, proceedings*, 36–54.

McKinney, Wes. 2010. Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference* 1697900(Scipy):51–56.

Miotto, Riccardo, Li Li, Brian A. Kidd, and Joel T. Dudley. 2016. Deep Patient: An Unsupervised Representation to Predict the Future of Patients from the Electronic Health Records. *Scientific Reports* 6:26094.

Mitchell, Matthew W. 2011. Bias of the Random Forest Out-of-Bag (OOB) Error for Certain Input Parameters. *Open Journal of Statistics* 1:205–211.

Mitchell, Tom M. (Tom Michael). 1997. *Machine Learning*. McGraw Hill.

Mossman, Douglas. 1999. Three-way ROCs. *Medical Decision Making* 19(1):78–89.

Muggeo, Vito M. R. 2003. Estimating regression models with unknown breakpoints. *Statistics in Medicine* 22(19):3055–3071.

———. 2008. Segmented: An R Package to Fit Regression Models With Broken-Line Relationships. *R News* 8(1):20–25.

Naor, Moni, and Benny Pinkas. 1999. Oblivious transfer and polynomial evaluation. In *Proceedings of the thirty-first annual association of computing machinery symposium on theory of computing, may 1-4, 1999, atlanta, georgia, USA*, 245–254.

Narayanan, Arvind, and Vitaly Shmatikov. 2008. Robust de-anonymization of large sparse datasets. In *2008 the institute of electrical and electronics engineers symposium on security and privacy, 18-21 may 2008, oakland, california, USA*, 111–125.

O'Brien, Deirdre B., Maya R. Gupta, and Robert M. Gray. 2008. Cost-sensitive multi-class classification from probability estimates. In *Proceedings of the 25th international conference on machine learning*, 712–719. New York, New York, USA: ACM Press.

Opitz, D., and R. Maclin. 1999. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research* 11:169–198.

Overhage, J. Marc, Patrick B. Ryan, Christian G. Reich, Abraham G. Hartzema, and Paul E. Stang. 2012. Validation of a common data model for active safety surveillance research. *Journal of the American Medical Informatics Association*.

Parzen, Emanuel. 1962. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics* 33(3):1065–1076.

Pathak, Jyotishman, Abel N Kho, and Joshua C Denny. 2013. Electronic health records-driven phenotyping: challenges, recent advances, and perspectives. *Journal of the American Medical Informatics Association*.

Pautasso, Marco. 2012. Publication growth in biological sub-fields: patterns, predictability and sustainability. *Sustainability* 4(12):3234–3247.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2012. Scikit-learn:

Machine Learning in Python. *Journal of Machine Learning Research* 12:2825–2830. 1201.0490.

Peissig, Peggy L., Vitor Santos Costa, Michael D. Caldwell, Carla Rottscheit, Richard L. Berg, Eneida A. Mendonca, and David Page. 2014. Relational machine learning for electronic health record-driven phenotyping. *Journal of Biomedical Informatics* 52:260–270.

Provost, Foster, and Pedro Domingos. 2000. Well-Trained PETs: Improving probability estimation trees. *CDER Working Paper #00-04-IS*.

Provost, Foster, and Tom Fawcett. 1997. Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions. *Knowledge Discovery and Data Mining 1997 Proceedings*.

Rajkomar, Alvin, Eyal Oren, Kai Chen, Andrew M. Dai, Nissan Hajaj, Peter J. Liu, Xiaobing Liu, Mimi Sun, Patrik Sundberg, Hector Yee, Kun Zhang, Gavin E. Duggan, Gerardo Flores, Michaela Hardt, Jamie Irvine, Quoc Le, Kurt Litsch, Jake Marcus, Alexander Mossin, Justin Tansuwan, De Wang, James Wexler, Jimbo Wilson, Dana Ludwig, Samuel L. Volchenboum, Katherine Chou, Michael Pearson, Srinivasan Madabushi, Nigam H. Shah, Atul J. Butte, Michael D. Howell, Claire Cui, Greg S. Corrado, Jeffrey Dean, Peter J. Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, Patrik Sundberg, Hector Yee, Kun Zhang, Yi Zhang, Gerardo Flores, Gavin E. Duggan, Jamie Irvine, Quoc Le, Kurt Litsch, Alexander Mossin, Justin Tansuwan, De Wang, James Wexler, Jimbo Wilson, Dana Ludwig, Samuel L. Volchenboum, Katherine Chou, Michael Pearson, Srinivasan Madabushi, Nigam H. Shah, Atul J. Butte, Michael D. Howell, Claire Cui, Greg S. Corrado, and Jeffrey Dean. 2018. Scalable and accurate deep learning with electronic health records. *Nature Partner Journals Digital Medicine*. 1801.07860.

Rana, Santu, Sunil Kumar Gupta, and Svetha Venkatesh. 2015. Differentially private random forest with high utility. In *2015 institute of electrical and electronics engineers international conference on data mining, ICDM 2015, atlantic city, nj, usa, november 14-17, 2015*, 955–960.

Rasmussen, Luke V., Will K. Thompson, Jennifer A. Pacheco, Abel N. Kho, David S. Carrell, Jyotishman Pathak, Peggy L. Peissig, Gerard Tromp, Joshua C. Denny, and Justin B. Starren. 2014. Design patterns for the development of electronic health record-driven phenotype extraction algorithms. *Journal of Biomedical Informatics* 51:280–286.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?". In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining - kdd '16*.

Samet, Saeed, and Ali Miri. 2008. Privacy preserving ID3 using gini index over horizontally partitioned data. In *The 6th the institute of electrical and electronics engineers international conference on computer systems and applications, AICCSA 2008, doha, qatar, march 31 - april 4, 2008*, 645–651.

Sejnowski, T. J., and C. R. Rosenberg. 1987. Parallel networks that learn to pronounce English text. *Complex Systems*.

Shimabukuro, David W, Christopher W Barton, Mitchell D Feldman, Samson J Mataraso, and Ritankar Das. 2017. Effect of a machine learning-based severe sepsis prediction algorithm on patient survival and hospital length of stay: a randomised clinical trial. *British Medical Journal Open Respiratory Research* 4(1):e000234.

Sowers, Kurt M, and Melvin R Hayden. 2010. Calcific uremic arteriolopathy: pathophysiology, reactive oxygen species and therapeutic approaches. *Oxidative Medicine and Cellular Longevity* 3(2):109–21.

Srinivasan, Ashwin. 1999. Note On The Location Of Optimal Classifiers In N-Dimensional ROC Space. Tech. Rep., Oxford University Computing Laboratory.

Tai, Raymond K. H., Jack P. K. Ma, Yongjun Zhao, and Sherman S. M. Chow. 2017. Privacy-preserving decision trees evaluation via linear functions. In *Computer security - ESORICS 2017 - 22nd european symposium on research in computer security, oslo, norway, september 11-15, 2017, proceedings, part II*, 494–512.

Thain, Douglas, Todd Tannenbaum, and Livny Miron. 2005. Distributed computing in practive: the condor experience. *Concurrency and Computation: Practice and Experience* 17(2):325–356.

Tibshirani, Robert. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B: Methodological* 58:267–288.

Tsoumakas, Grigorios, and Ioannis Katakis. 2007. Multi-Label Classification. *International Journal of Data Warehousing and Mining* 3(3):1–13.

Tunstall-Pedoe, H, K Kuulasmaa, P Amouyel, D Arveiler, A M Rajakangas, and A Pajak. 1994. Myocardial infarction and coronary deaths in the World Health Organization MONICA Project. Registration procedures, event rates, and case-fatality rates in 38 populations from 21 countries in four continents. *Circulation* 90(1):583–612.

US National Library of Medicine. Medline/pubmed citation records. `https://www.nlm.nih.gov/databases/download/pubmed_medline.html`.

Vaidya, Jaideep, Chris Clifton, Murat Kantarcioglu, and A. Scott Patterson. 2008. Privacy-preserving decision trees over vertically partitioned data. *Transactions on Knowledge Discovery from Data* 2(3):14:1–14:27.

Vaidya, Jaideep, Basit Shafiq, Wei Fan, Danish Mehmood, and David Lorenzi. 2014. A random decision tree framework for privacy-preserving data mining. *Institute of Electrical and Electronics Engineers Transactions Dependable Secure Computing* 11(5): 399–411.

Van Der Walt, Stéfan, S. Chris Colbert, and Gaël Varoquaux. 2011. The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering* 13(2):22–30. `1102.1523`.

Wang, Jiaxuan, Jeeheh Oh, Haozhu Wang, and Jenna Wiens. 2018. Learning credible models. In *Proceedings of the 24th acm sigkdd international conference on*

knowledge discovery & data mining, 2417–2426. KDD '18, New York, NY, USA: ACM.

Waskom, Michael, Olga Botvinnik, Drewokane, Paul Hobson, Yaroslav Halchenko, Saulius Lukauskas, Jordi Warmenhoven, John B. Cole, Stephan Hoyer, Jake Vanderplas, Gkunter, Santi Villalba, Eric Quintero, Marcel Martin, Alistair Miles, Kyle Meyer, Tom Augspurger, Tal Yarkoni, Pete Bachant, Constantine Evans, Clark Fitzgerald, Tamas Nagy, Erik Ziegler, Tobias Megies, Daniel Wehner, Samuel St-Jean, Luis Pedro Coelho, Gregory Hitz, Antony Lee, and Luc Rocher. seaborn: v0.7.0 (January 2016).

Weenig, Roger H., Lindsay D. Sewell, Mark D.P. Davis, James T. McCarthy, and Mark R. Pittelkow. 2007. Calciphylaxis: Natural history, risk factor analysis, and outcome. *Journal of the American Academy of Dermatology* 56(4):569–579.

Weiss, Jeremy C., Sriraam Natarajan, Peggy L. Peissig, Catherine A. McCarty, and David Page. 2012. Machine Learning for Personalized Medicine: Predicting Primary Myocardial Infarction from Electronic Health Records. *Artificial Intelligence Magazine* 33(4):33.

Weston, J, and C Watkins. 1999. Support Vector Machines for Multi-Class Pattern Recognition. *Proceedings of the 7th European Symposium on Artificial Neural Networks (ESANN-99)*.

Wilson, P W, R B D'Agostino, D Levy, A M Belanger, H Silbershatz, and W B Kannel. 1998. Prediction of coronary heart disease using risk factor categories. *Circulation* 97(18):1837–47.

Xiao, Mingjun, Liusheng Huang, Yonglong Luo, and Hong Shen. 2005. Privacy preserving ID3 algorithm over horizontally partitioned data. In *Sixth international conference on parallel and distributed computing, applications and technologies (PDCAT 2005), 5-8 december 2005, dalian, china*, 239–243.

Zimmet, Paul, K. G. M. M. Alberti, and Jonathan Shaw. 2001. Global and societal implications of the diabetes epidemic. *Nature* 414(6865):782–787.

9 APPENDIX

A Proofs for AUC Mu

Properties of AUC_μ

Here we show that AUC_μ satisfies the properties we listed in Section 7.1.

Theorem 1. *AUC_μ has the following properties:*

1. *If a model gives the correct label the highest probability on every example, then $AUC_\mu = 1$*
2. *Random guessing on examples yields $AUC_\mu = 0.5$*
3. *AUC_μ is insensitive to class skew*

Proof. Property 1. Let $\mathcal{M}$ be a multi-class classification model for a task with K classes and let n be the number of examples in the test set. Property 1 assumes that for all n instances $\mathcal{M}$ assigns the correct label the highest probability. The formula for AUC_μ is an average over separability functions between pairs of classes. Consider the separability function $S(i,j)$ between classes $i < j \leqslant K$. Let $\hat{\mathbf{p}}^{(a)}$ and $\hat{\mathbf{p}}^{(b)}$ be predictions for instances from classes i and j respectively. Further, let $\mathbf{v}^T$ be the normal vector to the decision hyperplane between classes i and j derived from the argmax partition matrix $\tilde{A}$. As $\hat{\mathbf{p}}^{(a)}$ and $\hat{\mathbf{p}}^{(b)}$ are labeled correctly we have $\tilde{I}(O(\mathbf{y}^{(a)}, \mathbf{y}^{(b)}, \hat{\mathbf{p}}^{(a)}, \hat{\mathbf{p}}^{(b)}, \mathbf{v}^T)) = 1$ as the instances are oriented correctly. We have n_i and n_j instances from classes i and j respectively. Then $S(i,j) = 1$ for any pair of classes i and j.

$$S(i,j) = \frac{1}{n_i n_j} \sum_{a \in D(i), b \in D(j)} \tilde{I}(O(\mathbf{y}^{(a)}, \mathbf{y}^{(b)}, \hat{\mathbf{p}}^{(a)}, \hat{\mathbf{p}}^{(b)}, \mathbf{v}^T)).$$

There are $K(K-1)/2$ choices of unordered pairs of i and j and thus $K(K-1)/2$ choices of $S(i,j)$, each of value 1.

$$\text{AUC}_\mu = \frac{2}{K(K-1)} \sum_{i<j} S(i,j)$$

$$\text{AUC}_\mu = \frac{2}{K(K-1)} \sum_{i<j} 1 = \frac{2}{K(K-1)} \frac{K(K-1)}{2} = 1$$

Therefore, if a model gives the correct label the highest probability on every example, then $\text{AUC}_\mu = 1$.

Property 2: Random guessing yields $\text{AUC}_\mu = 0.5$. If a model randomly guesses the prediction for all points, then any two predictions are equally likely to be oriented correctly or incorrectly via Equation 7.3. Then, for any separability function $S(i,j) = 0.5$ from the modified indicator function, $\tilde{I}$. As AUC_μ is an unweighted average of separabilities functions over all choices of i and j, the average of all separability functions is 0.5. Therefore, if a model randomly guesses for all points, then $\text{AUC}_\mu = 0.5$.

Property 3: AUC_μ is insensitive to class skew. AUC_μ is an unweighted average over separability functions. Each separability function, $S(i,j)$, can be computed using the standard two class AUC algorithm by first ranking all instances in classes i and j by using the two-class decision boundary derived from the partition matrix. As AUC is insensitive to class skew, so too are the individual separability functions. The calculation of AUC_μ is an unweighted average of each $S(i,j)$, and thus changes in class skew will not change the value of AUC_μ . Therefore, AUC_μ is insensitive to class skew. □

Partition Matrix

Theorem 2. *Let $\mathcal{M}$ be a model trained to perform a binary classification task. Let A be a 2×2 partition matrix with diagonal zeros and all other entries positive. Then A has no effect on the ranking of predictions from $\mathcal{M}$*

Proof. A typical binary classification model will output a single value, e.g. $\hat{y}^{(i)} = u$, corresponding to the probability of an instance belonging to one of the two classes (typically the positive class). Here we will use an equivalent model output in

its categorical distribution form, e.g. $\hat{\mathbf{p}}^{(i)} = [u, 1-u]$. Let $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ be two instances with the true labels of these two instances $\mathbf{y}^{(i)} = [1, 0]$ and $\mathbf{y}^{(j)} = [0, 1]$. Let $\hat{\mathbf{p}}^{(i)} = [u, 1-u]$ and $\hat{\mathbf{p}}^{(j)} = [v, 1-v]$ be the predicted categorical distributions for $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ respectively. Recall that in the standard calculation of AUC, if $u > v$ then $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are ranked correctly. We define our partition matrix, A, with $\alpha, \beta > 0$, following the rules presented in Section 7.3.

$$A = \begin{bmatrix} 0 & \alpha \\ \beta & 0 \end{bmatrix}$$

Without loss of generality, let the first class be our positive class and the second class be our negative class. We calculate our orthogonal vector to our hyperplane $\mathbf{v}^T = [-\beta, \alpha]$. We can now plug in our values into Equation 7.3 to inspect how our orientation function is influenced by our choice of A. We calculate $(\mathbf{y}^{(i)} - \mathbf{y}^{(j)}) = [1, -1]$ and $(\hat{\mathbf{p}}^{(i)} - \hat{\mathbf{p}}^{(j)}) = [u - v, v - u]$. Further, $\mathbf{v}^T(\mathbf{y}^{(i)} - \mathbf{y}^{(j)}) = -(\beta + \alpha)$ and $\mathbf{v}^T(\hat{\mathbf{p}}^{(i)} - \hat{\mathbf{p}}^{(j)}) = (\beta + \alpha)(v - u)$. Our final expression yields $(\beta + \alpha)^2(u - v)$. This expression is only positive when $u > v$. Thus, the orientation is solely determined by u and v, and that if $u > v$ our orientation is correct. This is the same result as the standard AUC calculation and we therefore conclude that binary classification is a special case for the calculation of AUC that does not require a partition matrix. □

Corollary 3. *When* $K = 2$, *AUC*$_\mu$ *simplifies to the Mann-Whitney U-statistic formulation of AUC.*

Proof. Let $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ be two instances with the true labels of these two instances $\mathbf{y}^{(i)} = [1, 0]$ and $\mathbf{y}^{(j)} = [0, 1]$. Let $\hat{\mathbf{p}}^{(i)} = [u, 1-u]$ and $\hat{\mathbf{p}}^{(j)} = [v, 1-v]$ be the predicted categorical distributions for $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ respectively. As shown in the proof of Theorem 7.6, $\hat{\mathbf{p}}^{(i)}$ and $\hat{\mathbf{p}}^{(j)}$ will be ranked correctly only if $u > v$. This is exactly the test performed in Equation 7.1, the Mann-Whitney U-statistic version of AUC. Therefore, when $K = 2$, AUC_μ is equivalent to AUC. □

Theorem 4. *Let* $\mathcal{M}$ *be a model trained for a multi-class classification task with* $K > 2$ *classes. Then the ranking of predictions from* $\mathcal{M}$ *is not independent of the choice of* $K \times K$ *partition matrix, hence calculating AUC*$_\mu$ *requires a partition matrix.*

Proof. Assume that the AUC_μ for $\mathcal{M}$ is independent of the choice of partition matrix, A, for the task. Then for any two points, $\hat{\mathbf{p}}^{(i)}$ and $\hat{\mathbf{p}}^{(j)}$, the ranking of these points is not changed by A. We show that there is at least one counterexample to our assumption for any $K > 2$. We first show this for $K = 3$ and then describe how this can be easily generalized to choices of $K > 3$.

Let $K = 3$, $\hat{\mathbf{p}}^{(i)} = [.5, .45, .05]^T$, $\hat{\mathbf{p}}^{(j)} = [.35, .4, .25]^T$, and $\mathbf{y}^{(i)} = [1, 0, 0]^T$, $\mathbf{y}^{(j)} = [0, 1, 0]^T$. Let A and A' be two partition matrices. Let $A_{1,\cdot} = [0, 1, 1]$ and $A_{2,\cdot} = [1, 0, 1]$. We find $\mathbf{v}^T = [-1, 1, 0]$ and $O(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}, \hat{\mathbf{p}}^{(i)}, \hat{\mathbf{p}}^{(j)}, \mathbf{v}^T) = 0.2$ showing that our points are oriented correctly. Now let $A'_{1,\cdot} = [0, 4, 1]$ and $A'_{2,\cdot} = [1, 0, 1]$. We find $\mathbf{v}'^T = [-1, 4, 0]$ and $O(\mathbf{y}^{(i)}, \mathbf{y}^{(j)}, \hat{\mathbf{p}}^{(i)}, \hat{\mathbf{p}}^{(j)}, \mathbf{v}'^T) = -0.25$ showing that our points are oriented incorrectly. Therefore, for $K = 3$ there exists at least one pair of points whose ranking is dependent on the choice of partition matrix. This can be readily generalized to $K > 3$ by padding 0s to the end of the vectors $\hat{\mathbf{p}}^{(i)}$, $\hat{\mathbf{p}}^{(j)}$, $\mathbf{y}^{(i)}$, and $\mathbf{y}^{(j)}$. □

Theorem 5. *The expectation over all partition matrices, uniformly distributed, for a task with* K *classes is the argmax partition matrix,* $\tilde{A}$*, where* $\tilde{A}_{i,i} = 0\ \forall i$ *and* $\tilde{A}_{i,j} = 1\ \forall i \neq j$.

Proof. Let $\mathcal{A}$ be the set of all $K \times K$ partition matrices with diagonal elements zero and off-diagonal elements positive. Now let A be a partition matrix drawn randomly according to a uniform distribution, $\mathcal{U}$, from $\mathcal{A}$. Consider two off diagonal elements in A, $A_{i,j}$ and $A_{k,l}$ with $(i, j) \neq (k, l)$. We find that $\mathbb{E}_{\mathcal{U}} A_{i,j} = \mathbb{E}_{\mathcal{U}} A_{k,l}$ as there is exactly one other $A' \in \mathcal{A}$ where $A'_{i,j} = A_{k,l}$, $A'_{k,l} = A_{i,j}$, and all other elements equal. Therefore, if the expectation of any two random off-diagonal elements in A are equal, then the expectations of all off-diagonal elements in A is equal. Let $\mathbb{E}_{\mathcal{U}} A_{i,j} = \sigma$; then A is equal to $\sigma\tilde{A}$, where $\tilde{A}$ is the argmax partition matrix with uniform misclassification costs. As noted in (O'Brien et al., 2008), $\tilde{A}$, induces the same partitioning as A. Because we chose A at random from $\mathcal{A}$, we find that $\mathbb{E}_{\mathcal{U}} A$ is the argmax partition matrix with uniform misclassification costs. □

Time Complexity

Time Complexity of AUC_μ Using Argmax Partition Matrix $\tilde{A}$

$$\text{AUC}_\mu = \frac{2}{K(K-1)} \sum_{i<j} S(i,j)$$

Here we compute the time complexity of AUC_μ when the argmax partition matrix, $\tilde{A}$ is used. Let K be the number of classes and $n = n_1 + \ldots n_K$ be the total number of instances, and the number of instances for each class respectively. Computing AUC_μ requires $K(K-1)/2$ repetitions of computing a separability function between an unordered pair of classes. We breakup the time complexity calculation into two parts; we first calculate the time complexity of a single separability function $S(i,j)$, and then we use this to calculate the time complexity of AUC_μ.

We calculate the time complexity of $S(i,j)$ by first noting its relationship with AUC. $S(i,j)$ shares an equivalence to AUC as it is the probability that two instances are ranked correctly by first using the two class decision boundary, $\mathbf{v}^T\hat{\mathbf{p}} = 0$, to map the categorical predictions to scalar values that can be ranked. As such, the time to compute $S(i,j)$ is the time to perform the mapping and then the time to calculate the AUC. There are $n_i + n_j$ predictions for each $S(i,j)$ computation, and mapping these categorical distributions to scalar values involves a dot product between two vectors of dimension K. This procedure would normally be of complexity $O(K(n_i + n_j))$, however we next show that when using the argmax matrix this can be performed in time $O(n_i + n_j)$. All elements in $\tilde{A}$ are 1 except the diagonal entries which are 0. Thus all elements of $\mathbf{v}^T = \tilde{A}_{i,\cdot} - \tilde{A}_{j,\cdot}$ are 0 except $\mathbf{v}^T{}_i = -1$ and $\mathbf{v}^T{}_j = 1$. Thus we may calculate $\mathbf{v}^T\hat{\mathbf{p}} = \hat{\mathbf{p}}_j - \hat{\mathbf{p}}_i$, which is a constant time calculation for each of the $n_i + n_j$ instances. We next compute $S(i,j)$ by calculating the AUC of these ranked instances, and using the AUC time complexity result from (Fawcett, 2006). The time complexity of computing AUC with $n_i + n_j$ instances is $O((n_i + n_j)\log(n_i + n_j))$. Thus, time complexity to calculate $S(i,j)$ is $O((n_i + n_j)\log(n_i + n_j))$ as the time complexity of the ranking is dominated by the complexity of calculating the AUC of the ranked points.

The time complexity for AUC_μ can be determined by summing the time complexities for each of the $K(K-1)/2$ choices of i and j.

$$O(\sum_{i<j}(n_i + n_j)\log(n_i + n_j))$$

We note here that $\log(n_i+n_j) < \log n$ as $n_i+n_j < n$ and we use this relationship so that we may pull this term out of the sum.

$$< O(\log n \sum_{i<j} n_i + n_j)$$

Next we note that for a given n_i it appears in the sum $K-1$ times as it shows up each time it class i is paired with one of the other $K-1$ classes. We therefore exchange the two sums

$$= O(\log n \sum_{i=1}^{K}(K-1)n_i)$$
$$= O(\log n(K-1)n)$$

Therefore, the time complexity of AUC_μ is $O(Kn\log n)$.

Time Complexity of AUC_μ With Any Partition Matrix

For the general case of computing AUC_μ with any partition matrix, A, the time complexity is $O(Kn(K+\log n))$. When computing the time complexity of AUC_μ using $\tilde{A}$, we noted that $\tilde{A}$ had the special property that dot products could be performed in constant time. This characteristic is not true for all choices of A and thus in general the ranking of points has complexity $O(K(n_i+n_j))$. Therefore, computing $S(i,j)$ has complexity $O((n_i+n_j)(K+\log(n_i+n_j)))$. Finding the total time complexity of AUC_μ can be done in the same manner as the computation above, however this time we substitute $K+\log n$ for $K+\log(n_i+n_j)$ and pull the former term out of the sum. Therefore, the time complexity of AUC_μ for a general choice of A is $O(Kn(K+\log n))$.

Time Complexity of M

The measure, M, proposed by (Hand and Till, 2001) is claimed to have a time complexity of $O(K^2 n\log n)$ by (Fawcett, 2006). An additional contribution of our

work is to show that the time complexity of this measure is in fact $O(Kn \log n)$. The calculation follows almost identically our approach for computing the complexity of AUC_μ using the argmax partition matrix. Let K be the number of classes and $n = n_1 + \ldots n_K$ be the total number of instances, and the number of instances for each class respectively.

$$M = \sum_{i<j} \hat{A}(i,j)$$

Here, $\hat{A}(i,j)$ is computed by averaging two AUC calculations each with $n_i + n_j$ instances. Therefore, the time complexity of computing $\hat{A}(i,j)$ is $O((n_i + n_j) \log(n_i + n_j))$. The expressions of M and AUC_μ differ only in their separability functions $\hat{A}(i,j)$ and $S(i,j)$. As these functions have the same time complexity, the overall algorithms do as well. Therefore, the time complexity of M is $O(Kn \log n)$.

B Tables for High-Throughput Machine Learning

Table 9.1: Parameter values used to construct RandomForestClassifier

Parameter	Value
n_estimators	500
criterion	'gini'
max_depth	None
min_samples_split	2
min_samples_leaf	1
min_weight_fraction_leaf	0.0
max_features	0.1
max_leaf_nodes	None
bootstrap	True
oob_score	False
n_jobs	1
random_state	None
verbose	0
warm_start	False
class_weight	None

Table 9.2: Parameter values input to the function seaborn.kdepolot to construct Figures 4.1 and 4.3

Parameter	Value (Fig. 4.1)	Value (Fig. 4.3)	Value (Fig 4.3 CI)
shade	True	True	False
vertical	False	False	False
kernel	'gau'	'gau'	'gau'
bw	0.008	'scott'	'scott'
gridsize	100	100	100
clip	None	None	None
legend	True	False	False
cumulative	False	False	False
shade_lowest	True	True	True

Table 9.3: Parameter values input to the function seaborn.violinplot to construct Figure 4.2

Parameter	Value (Fig. 4.2)
hue_order	None
bw	0.2
cut	2
scale	‘area’
scale_hue	True
gridsize	100
width	0.8
inner	None
split	True
orient	’h’
linewidth	None
color	None
palette	‘muted’
saturation	0.75

Table 9.4: ICD-9 Diagnosis codes not allowed to be used as a prerequitie diagnosis in DDR.

Code	Description
89	Diagnostic Interview, Consultation, and Evaluation
V20	Health supervision of infant or child
V24	Postpartum care and examination
V25	Encounter for contraceptive management
V28	Encounter for anteatal screening of mother
V29	Observation and evaluation of newborns for suspected conditions not found
V51	Aftercare involving the use of plastic surgery
V56	Encounter for dialysis and dialysis catheter care
V58	Encounter for other and unspecified procedures and aftercare
V60	Housing household and economic circustances
V61	Other family circumstances
V62	Other psychosocial circumstances
V63	Unavailability of other medical facilities for care
V64	Persons encountering health services for specific procedures not carried out
V65	Other persons seeking consultation
V66	Convalescence and palliative care
V67	Follow-up examination
V68	Encounters for administrative purposes
V69	Problems related to lifestyle
V70	General medical examination
V71	Observation and evaluation for suspected conditions not found
V72	Special investigations and examinations
V73	Special screening examination for viral and chlamydial diseases
V74	Special screening examination for bacterial and spirochetal diseases
V75	Special screening examination for other infectious diseases
V76	Special screening for malignant neoplasms
V77	Special screening for endocrine nutritional metabolic and immunity disorders
V78	Special screening for disorders of blood and blood-forming organs
V79	Special screening for mental disorders and developmental handicaps
V80	Special screening for neurological eye and ear diseases
V81	Special screening for cardiovascular respiratory and genitourinary diseases
V82	Special screening for other conditions