University of Nevada, Reno

Clinical Dataset Analysis and Patient Outcome Prediction via Machine Learning

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Chemical Engineering

by

Alex Buettner

Víctor R. Vásquez, Ph.D., Thesis Adviser

December 2018
Declaration of Authorship

I, Alex Buettner, declare that this thesis titled, “Clinical Dataset Analysis and Patient Outcome Prediction via Machine Learning” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.

- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.

- Where I have consulted the published work of others, this is always clearly attributed.

- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.

- I have acknowledged all main sources of help.

- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: [Signature]

Date: 12/18/18
THE GRADUATE SCHOOL

We recommend that the thesis prepared under our supervision by

ALEX BUETTNER

Entitled

Clinical Dataset Analysis And Patient Outcome Prediction Via Machine Learning

be accepted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Dr. Victor Vasquez, Advisor

Dr. Charles Coronella, Committee Member

Dr. Ronald Lembke, Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

December, 2018
Abstract

We analyze and evaluate relevant machine learning methods for use in extracting and understanding clinical data sets in the context of optimization of clinical processes. Three data sets were considered to demonstrate the types and style of data found in the healthcare field: (a) the Pima Indians diabetes dataset (PIDD), a non-time-dependent diabetes onset study, (b) an alcoholism EEG dataset (AED), studying responses of alcoholic and control subjects when exposed to image stimulus, and (c) the diabetes readmission dataset (DRD), that focuses on factors that relate to diabetic patient readmission times. Each dataset is modeled using a variety of machine learning methods, including Bayesian, neural network, and decision tree methods, to better understand the advantages and disadvantages as applicable to rapid dependency extraction and understanding of the information contained therein. The goal of this work is to analyze the potential of machine learning for use in management of clinical processes and operations.

Neural network models are used to assess all three data sets; two using dense neural networks, and one using convolutional neural networks. The dense neural network model used on the PIDD resulted in a maximum prediction accuracy of $81.77\%$. In contrast, the use of neural network (NN) models on the much larger DRD demonstrated some drawbacks that were not expected upon initial analysis of the data. We found that the NN model performs poorly on this dataset, with classification accuracy no higher than $61.17\%$, due to the complexity of the dataset and potential need for more data. The use of convolutional neural networks for analysis of time series data was demonstrated on the alcoholism EEG dataset, resulting
in subject classification accuracies between 91.41% and 98.82% depending on the training and testing sets used to analyze the model.

Bayesian methods are used to analyze all three datasets, both in supervised and unsupervised manners. Supervised learning analysis on the PIDD showed improvement over published results, but are generally in agreement with the literature. Classifications accuracies of resulted in a maximum of 84.49% on a preprocessed dataset, and 79.75% on an unmodified dataset. Similarly, supervised learning was applied to the DRD, resulting in maximum classification accuracies on a three-class and two-class model of 58% and 62%, respectively. Unsupervised Bayesian methods are applied to the alcoholism EEG dataset in order to extract the true number of classes present in the model, in which all trials correctly identified two subject classes without the aid of labeling. Hidden Markov models are also applied to the alcoholism EEG dataset in an unsupervised fashion, allowing us to extract characteristic states in each EEG sample, for each subject class.

The PIDD and DRD sets are also processed using decision tree models; gradient boosting classifiers (GBC) are applied to the PIDD, and extreme gradient boosting classifiers (XGBC) are applied to the DRD. The GBC model used to analyze the PIDD resulted in a maximum classification accuracy of 82.48% on preprocessed data, and 80.60% on an unmodified dataset. The DRD showed difficulty in model development, with maximum classification accuracy reaching approximately 55%, and with insensitivity to two of three data labels. The model seems unable to capture the components of the third class, showing that the distinguishability of the classes may be lacking.
This thesis marks a significant accomplishment in my academic career, being one of the most challenging and rigorous endeavors I have ever completed. However, to say this was completed entirely by my own volition is a falsity. It has taken the encouragement and support of many different people in my life to get me here this day. Of the many who have given me drive, the most influential and important are my parents, Juliet and Hermann Buettner. It is to them that I dedicate this thesis, and to them that I owe my gratitude.

As a child, my mother always told me I would go to college. She read to me every night, aided in my learning, and gave me everything I needed to succeed as a student. Atop this, she gave me love and support that never faded, pushed me to do my best, and to go the extra mile in everything I did. She instilled in me the courage to take the path less followed and not look back, to be kind, gentle, and caring to those around me, and most importantly, to never give up on my dreams. It was her dedication to giving me the best future possible that has enabled me to reach my goals, and to complete this masters degree.

As a young man, my father gave me the guidance and drive to be the best man I could be. He gave me his advice, passed on his knowledge, and taught me that discipline is an important part of ones life. He taught me to be respectable, kind, and well mannered, no matter the
time or place. He shared his skills with me, taught me to cook, and enabled me to live on my own without fear of failure. My father is the best man I know, and to this day I strive to be as good and as strong of a man as he is. His dedication to my upbringing, his family, and to doing things the right way has given me an exceptional role model after whom to model my own life. My ability to complete this thesis is a direct result of everything he has given me.

Throughout my time on this earth, I have been fortunate enough to have two wonderful parents who have filled my life with joy, support, and love. They have given me opportunities to grow, shown me the world, and always included me in their lives. It is quite obvious, then, that this thesis be completed in their honor. They have given me everything, and to them I owe my undying gratitude. I love you Mom and Dad. Now, and forever onward.
Acknowledgements

I would like to thank my advisory committee, namely Dr. Victor Vasquez, Dr. Charles Coronella, and Dr. Ron Lembke, who guided me through my collegiate career, and through this masters thesis. I am grateful for their patience and support in overcoming numerous obstacles I was faced with during my studies, and for being patient and kind when I was struggling to move forward.

I would like to thank my fellow masters and doctoral students for their feedback, friendship, and for keeping me smiling when times got rough. I will never forget that laughter can be the best medicine when stress seems to be the only thing that exists.

I would like to thank my dear friend and colleague Dr. Kevin Schmidt for convincing me to undertake the masters degree in the first place, and for spending all that time cooped up with me in the office, discussing topics ranging from penguins to quantum mechanics.

I would also like to thank my wonderful girlfriend Kristen Demko, who has shown me endless support and love since I began this endeavor, and continues to do so every day.

And finally, I would like to thank Juliet and Hermann Buettner, my mother and father, who instilled in me the desire to better myself and do good in this world, to push myself further every single day, and never give up. From the day I was born, they have shown me undying love
and support, care, and compassion, in every aspect of my life. Without them, I would be nothing, and I am extremely grateful to have them as my parents.

"Family and friends are hidden treasures, seek them and enjoy their riches." - Wanda Hope Carter
# Contents

1 **Introduction**  
   1.1 Clinical Information Systems ............................................. 3  
   1.2 Clinical Processes ............................................................. 4  
   1.3 Process Systems Engineering Approach ................................. 6  
   1.4 Current Uses of Big Data in Industry ................................. 7  
   1.5 Model-Based Optimization .................................................. 9  
      1.5.1 Deterministic Methods ..................................................... 9  
      1.5.2 Bayesian and Markovian Statistical Inference Models .......... 10  
   1.6 Motivation for Machine Learning and Big Data Analysis ............ 13  
      1.6.1 Machine Learning ........................................................ 13  
   1.7 Scope and Objectives ......................................................... 14  
      1.7.1 Scope of Work .......................................................... 14  
      1.7.2 Objectives ............................................................ 14  

2 **Distributed Healthcare** ......................................................... 16  
   2.1 Mobile Health Systems ....................................................... 18  
      2.1.1 Trends in Remote Patient Monitoring ................................. 19  
      2.1.2 Efficacy of Mobile Health Monitoring Systems .................. 20  
   2.2 Clinical Information Systems .............................................. 21
2.2.1 Commercial Clinical Information Systems .......................... 22
  2.2.1.1 Examples of Clinical Information System ..................... 24
  2.2.2 Limitations of Current Clinical Information Systems ........... 25

3 Clinical Data sets
  3.1 Electronic Medical Records ........................................... 28
  3.2 Data sets ............................................................... 30
  3.3 Ethical use of Health Data ............................................ 32
  3.4 Generation of Synthetic Patient Data ............................... 33

4 Machine Learning Methods ................................................. 36
  4.1 Framework ............................................................. 36
  4.2 Reasons for Dataset Analysis ......................................... 39
  4.3 Dataset Structure Analysis and Metrics ............................ 40
    4.3.1 Terminology ....................................................... 40
    4.3.2 Feature Importances .............................................. 43
    4.3.3 Model Insights ................................................... 43
    4.3.4 Dimensionality Reduction, Embedding, and Clustering Methods 47
    4.3.5 Labeling Routines ................................................. 51
  4.4 Supervised Learning ................................................... 55
    4.4.1 Neural Networks ................................................... 55
    4.4.2 Bayesian Classifiers .............................................. 63
    4.4.3 Decision Trees .................................................... 64
  4.5 Unsupervised Learning ................................................ 69

5 Preprocessing and Data Preparation Techniques ......................... 73
  5.1 Imputation and data removal ......................................... 73
  5.2 Scaling ................................................................. 75
A Data and Code List

A.1 Pima Indians Dataset ................................................. 155
A.2 Alcoholism EEG Dataset ............................................. 156
A.3 Diabetes Readmission Dataset ..................................... 157
List of Tables

4.1 Infection Data ................................................................. 44
4.2 Covariance Types in GMMs .................................................. 55
4.3 Examples of Activation Functions ......................................... 57

6.1 Pima indians dataset column names and % missing values. ........ 81
6.2 Diabetes readmission dataset variable types and descriptions. ..... 85
6.3 Mapping of Admission type ID ............................................. 86
6.4 Mapping of Admission source ID ......................................... 86
6.5 Mapping of Discharge disposition ID .................................... 87

7.1 Resultant dataset dimensions under different pre-processing regimes.
   Dimensions shown as (rows, columns) ................................. 95
7.2 Classification accuracy on cleansed PIDD. Data was scaled using Min-
   Max scaling to (0,1). Table displays accuracy of train/test using k-fold
   crossvalidation (5 folds) with the Adam optimizer. Confidence inter-
   vals are reported using the t-score method at 95% confidence using 5
   samples. ................................................................. 98
7.3 Prediction accuracy using Naive Bayes and Bayes Classifier Models.
   Confidence intervals (CI) are reported using the t-score method at 95%
   confidence using 10 samples. ......................................... 101
7.4 Default gradient boosting classifier parameters ........................................ 104
7.5 Optimized parameters for the gradient boosting classifier model used to analyze the PIDD ............................................................. 105
7.6 Classification accuracies on the PIDD using both the default parameter values for the model, and optimized parameter values. Confidence intervals calculated using t-score with a sample size of 20 and confidence of 95%................................................................. 105
7.7 Layer configuration for the convolutional neural network used in this study .......................................................... 109
7.8 Validation and post-training validation accuracies of CNN model using the alcoholism EEG dataset. Values in parenthesis represent the post-validation accuracy of the test measured by the percentage of correctly classified subject............................................................. 110
7.9 Cluster weights defined by the GMM after fitting to the alcoholism EEG dataset............................................................. 112
7.10 Unique number of classes identified by the HMM classification. Results contain the number of classes as well as the respective frequency of class labels in the EEG alcoholism dataset............................................................. 116
7.11 Confusion matrix of the multivariate Gaussian 3-class model............................................................. 122
7.12 Confusion matrix of the independent normal 3-class model............................................................. 123
7.13 Confusion matrix of the multivariate Gaussian 2-class model............................................................. 123
7.14 Confusion matrix of the independent normal 2-class model............................................................. 124
7.15 $F_1$-scores for the 3-class models............................................................. 125
7.16 $F_1$-scores for the 2-class models............................................................. 125
7.17 Confusion matrix of the neural network model used to analyze the diabetes readmission dataset............................................................. 130
7.18 $F_1$-scores for the 2-class NN model on the training data.  

7.19 Descriptions and default values of parameters used in the XGB\textsuperscript{[67]} classifier model.  

7.20 Confusion matrix produced by H2O.ai framework on training set. Matrix rows: predicted class; Columns: actual class.  

7.21 $F_1$-scores for the 3-class XGB model on the training data.  

7.22 Confusion matrix produced by H2O.ai framework on crossvalidation set. Matrix rows: predicted class; Columns: actual class.  

7.23 $F_1$-scores for the 3-class XGB model after crossvalidation.
List of Figures

1.1 Diagram of PSE optimization cycle. This work will focus on the analysis component of the cycle in an effort to improve clinical process management. ........................................ 7

1.2 Example of a five-state outpatient Markov process. ...................... 10

2.1 Studies surrounding mobile devices used in the telehealth field. Data from Vegesna et al.\textsuperscript{[1]} ................................................ 19

3.1 Generic structure of a CDA record. ........................................... 29

3.2 Basic CDA document entry in XML. Elements in square brackets represent replaceable information. ........................................ 29

3.3 Basic CDA document entry in XML. The formerly undefined fields are completed and displayed as they would be in a complete CDA document. 30

3.4 A rendered CDA document populated with synthetic patient data. The document contains demographics data, as well as patient medical history in 11 subcategories. ........................................ 33

3.5 A rendered CDA document populated with synthetic patient data. The document displays patient demographics, vital sign history, and immunization history. Each subcategory of the document is formatted similarly. ........................................ 34
4.1 Flow chart diagram describing methods applicable to processing particular data types. Terminating nodes shown in green are used in this work. Terminating nodes shown in yellow may prove useful for analysis of clinical datasets, but are not used formally in this work.

4.2 A diagram of a single neuron used in a neural network model. Inputs are weighted and summed, a bias is introduced, and the output is passed through an activation function to limit the output range.

4.3 A simple feed-forward artificial neural network. Contains three distinct layers: (a) the input layer with 5 units, (b) the hidden layer with 3 units, and (c) the output layer with a single unit. This network maps an input of five variables to an output of a single variable.

4.4 In-depth diagram of an LSTM cell. The cell consists of a cell state, \( c \), a hidden state of the cell (also the output), \( h \), and 9 operators that control the flow of information within the cell.

4.5 Example of a convolution across a matrix with a \( 3 \times 3 \) kernel matrix. Each element of the output matrix is the element-wise summation of a \( 3 \times 3 \) region of the original matrix multiplied by the kernel matrix.

4.6 A simple convolutional neural network composed of a single convolutional layers, a max pooling layer, a fully connected layer, and a softmax classifier layer (1 \( \times \) 2).

4.7 A Markov model in discrete time. The model consists of three states, \( x_i \in x \), with probability of transition \( a_{ij} \) to state \( x_j \). Each state \( x_i \) has likelihood \( b_{ik} \) of emitting some \( y_k \in y \). The chain is fully defined, as each state will emit a particular set of emissions specific to that state with a known probability. Note that some states cannot emit particular emissions \( y_k \).
4.8 A hidden Markov model in discrete time. The model consists of three states, \( x_i \in \mathbf{x} \), with probability of transition \( a_{ij} \) to state \( x_j \). Each state \( x_i \) has likelihood \( b_{ik} \) of emitting some \( y_k \in \mathbf{y} \). The model is called hidden because we do not know the exact probabilities of transition or emission. Note that all states can emit any \( y_k \in \mathbf{y} \).

6.1 Histograms of each column of the Pima indians dataset.

6.2 EEG samples of a control subject (left) and an alcoholic subject (right). The response of the alcoholic shows distinct regions of activation that are not present in the control subject.

6.3 Admission Source ID. The admission source ID defines the means by which the patient arrived to the clinical encounter. Generally patients arrived through sources 1, 7, and 17, which correspond to physician referral, emergency room, and NULL (information not available). In general, diabetic encounters occur on scheduled time periods or in emergency situations. The shape of the distribution is of less importance to this work as it is a mapped categorical record, and not a numerical observation.

6.4 Admission Type ID. The admission type ID defines the department by which the patient was admitted. A majority of admissions occurred through the emergency, urgent care, elective, and unknown routes, aligning common admission routes for diabetic patients. Like the admission source, the admission type is a mapped categorical record, thus the distribution shape is not numerically significant.

6.5 Discharge disposition ID. Discharge disposition ID refers to the method of discharge and outcomes after leaving the hospital.
6.6 **Number of Lab Procedures.** The distribution of the number of lab procedures is approximately Gaussian distributed with a mean of 43 and a standard deviation of 23. The plotted values are shown on a log-scale.  

6.7 **Number of procedures.** The number of procedures, including surgery, the patient underwent during their current visit.  

6.8 **Number of medications.** The number of medications the patient is currently taking in regard to the treatment of diabetes.  

6.9 We can see that the number of emergency visits is exponentially distributed, with most patients having encountered 5 or less emergency visits in the past year.  

6.10 The number of all-time outpatient visits related to diabetes management and treatment.  

6.11 The number of all-time inpatient visits related to diabetes management and treatment.  

6.12 Patient length of stay (LOS) is described by the number of days spent in the hospital during the current visit.  

7.1 Schematic of dense neural network used to classify the PIDD.  

7.2 Examples of control subject and alcoholic subject time series (a and b) and equivalent Markov transition field (c and d).  

7.3 A diagram of the convolutional neural network used to process the alcoholism EEG dataset.  

7.4 Diagram of the HMM configuration used in scheme one (unsupervised class identification) of the HMM analysis.  

7.5 Analysis of alcoholism EEG dataset using an HMM with two hidden states.
7.6 Analysis of alcoholism EEG dataset using an HMM with three hidden states. ............................................................. 119

7.7 Analysis of alcoholism EEG dataset using an HMM with four hidden states. ............................................................. 119

7.8 Analysis of alcoholism EEG dataset using an HMM with six hidden states. ............................................................. 120

7.9 Flow diagram of the neural network model used to classify the diabetes readmission dataset. ............................................. 129
Chapter 1

Introduction

With rapid development of computer-assisted healthcare operations, including the implementation of intraoperative monitoring systems, the increasing scale of operational data sets is increasing the use of automation as a primary tool for improving healthcare. The development of automated data-mining techniques continues to increase, and combined with a need for optimal resource allocation within clinical processes, healthcare research and development is moving toward personalized, highly dynamic patient care. Many problems in healthcare depend on the ability to identify underlying patterns of a dynamic process, determine a useful characterization of the noisy or sparse data, and implementing personalized changes.

In the past, many efforts in the field of clinical optimization have focused on frequentist scheduling models to better understand and optimize clinical processes. These methods rely on fixed statistical estimators to best predict flow patterns, patient readmission rates, and patient needs, but lack the ability to adapt to sudden changes. Disease outbreaks, emergencies, and other unpredictable events may potentially cause traditional models to be inadequate. Adaptive clinical process optimization has become an important field of study in recent years, as increasing patient loads and health costs affect the quality of care. Modeling of clinical processes
using stochastic models has gained popularity in recent decades\textsuperscript{[2]}, with big-data style analytics becoming a new frontier in optimization.

Clinical process data sets tend to be high dimensional, containing numerous variables, features, and relationships. The effective utilization of big data in healthcare is contingent on the speed and accuracy of processing and compilation into predictive models. Other factors such as intermittent data collection and cost restrictions limit most clinical processes from implementing full-scale optimization using conventional methods. Variables like average patient length-of-stay (ALOS) may be better predicted using active inference models that account for current operational conditions. Other operational variables such as patient intake rate, number of occupied beds, expected resource usage, available medical staff, available rooms, and triage condition also affect the ALOS\textsuperscript{[3]}.

Predictive models for ALOS assist hospital operations by reducing length-of-stay, improving patient satisfaction\textsuperscript{[4]}, and operation throughput; in turn translating into more acute care to larger groups of patients and increasing reimbursement in the operations. Modern healthcare settings are more value driven practices, with political and financial factors placing pressure on hospitals and medical staff to reduce operating costs in every department, while ensuring patient safety, quality services, and low costs. Efforts to improve these criterion have shown that reducing ALOS is a very significant means for achieving these goals. In addition to patient satisfaction, reduced ALOS also reduces the potential for patients’ to acquire secondary conditions during their stay, reducing potential costs and improving patient outcomes\textsuperscript{[5]}.

Dynamic models of clinical processes, with sensor-driven data and on-the-fly stochastic analysis can be very practical for improving both operational efficiency and patient care\textsuperscript{[6]}. Patient and staff flow and other operational variables can be monitored and analyzed to identify crowding, staff shortages, and needs of individ-
ual patients, as well as provide opportunities for optimization. Unlike traditional methods that rely on in-stream sampling of the process before optimization, active collection of process data can be used in real-time optimized models dynamically. Big data analysis provides unique opportunities for the characterization of clinical processes in new ways and for the extraction of useful patterns and relationships from the available data.

1.1 Clinical Information Systems

Modern patient monitoring systems generate large amounts of data that are fed into semi-automated or human monitored alert systems. For example, tier one intensive care units (ICUs) generally monitor patient’s vital health statistics from a centralized location; integrated networks handle both monitoring and staff response, allowing staff to better optimize patient care timelines. Although major improvements in patient monitoring have been developed, there are a significant number of opportunities for incorporating signal processing, computer science methods, biomedical engineering advancements, and mathematical models in advancing patient monitoring and care, in general\textsuperscript{[6]}. Lack of integration of standardized (non-proprietary) data formats cause difficulties for the integration with other monitoring systems, limiting advancements in patient care, particularly in intensive care environments\textsuperscript{[6]}

Dynamic analysis of patient care is critical for improving patient outcomes in a number of clinical settings, primarily critical care environments like the ICU\textsuperscript{[6]}. Low sampling rates of physiological data, including vital statistics and clinical notes, reduces the accuracy and speed at which a patient may be diagnosed or treated. Understanding dynamical behavior of patient health requires frequent sampling and the use of numerous data analysis techniques (linear, non-linear, stochastic, etc), many of which are overlooked or unused\textsuperscript{[6]}. Integration and synchronization of clinical
information systems remains a relevant and desirable means for improving physician understanding of acute patient health, allowing for rapid, actionable decision making and improvement of patient outcomes.

1.2 Clinical Processes

A clinical process is a system of discrete interacting units (DIUs) within the hospital or clinical setting. A basic process model may assess the flow of patients through a clinical process, account for interaction with both staff and equipment, and assess the efficiency of the process once the patient is discharged. Within each DIU are physicians, specialists, and support staff who may interact with other DIUs; equipment and services may also be utilized in a cross-unit fashion.

From a modeling and optimization standpoint, the units of analysis (UoA) form the primary components of a clinical process model; each UoA introduces different optimization goals, resource requirements, and interaction parameters to an optimization/efficiency model. Each UoA may be modeled as an individual entity, but may also take input from, or output to, other UoA’s within the same clinical process\cite{7}.

To define a UoA, in addition to modeling a process of interest, each unit should be a decision-making unit (DMUs). This characteristic requires some conversion of input to output within the unit, with decisions surrounding personnel, patients, and resources required by the unit. The choice and components of each UoA should be made such that it is comparable to other units within different processes. Similar objective models are critical if analysis across multiple processes is performed. Vertical integration of certain processes allows for better management of resources, but may become significantly affected by other units within the process. It is advantageous to focus on smaller and more focused units within a clinical process, e.g. individu-
als, teams, processes, etc. that can be better defined and more easily assessed for efficiency\cite{7}.

Staff are considered a semi-permanent component of a DMU, as the flow of staff is not restricted in the process model. Staff often work across sub-units or departments, and as such must be considered a mobile resource within a clinical process. Staff are grouped based on skill set, and often remain within a limited group of sub-units related to that skill set. Optimization of the staff-to-patient ratio ensures that staff (a labor input) remain both effective and economically feasible\cite{8}. Efficient scheduling of staff is also critical for quality of patient care. Over or under scheduling of staff is a major operational inefficiency within a healthcare operation; emergency departments are particularly susceptible to these staffing issues, due to their oscillating, and many times unpredictable, patient inflow\cite{9,10}.

Much like staff in a DMU, patient flow across DMUs is permitted by process models. Most efficiency outcomes are patient focused, dealing with the improved health of the patient, and not with the services provided. Patients, unlike staff, are not a labor input, and as such the indicator of patient success is primarily an improvement in health. Other indicators include patient satisfaction, and minimization of residence time within the process. Inside a process (or DIU), a patient is defined objectively by a number of factors. Primary, secondary, and tertiary health factors aid clinical staff in diagnosing and better understanding patient health status upon arrival; these factors are used by physicians and nursing staff to more accurately and rapidly approach a diagnosis. Primary demographic factors such as age, sex, and height are collected upon intake and provide some baseline for expected health. Secondary lifestyle factors such as drug and alcohol use, weight, diet and exercise regimen, etc. offer physicians additional insight into patient activities and potential areas of improvement\cite{11}. Tertiary factors such as social status and education, yield
useful public health data and help guide the physician in best improving patient quality of life and safety during treatment.

The nature of clinical processes and DMUs allows them to be modeled as flow processes. Different methods have been proposed surrounding economic and actor driven flow models, each deriving optimal flow parameters and efficiencies from different metrics\cite{9,12}. Traditionally, deterministic or analytical models are used to develop patient flow processes, but they lack generally dynamical adjustments with patient flow change.\cite{2}.

### 1.3 Process Systems Engineering Approach

Clinical processes may be modeled much like a flow process using a systems engineering approach. Clinical settings act as the process, and patient and staff act as inputs and outputs to the process. Optimization of this type of process is typical of process systems engineering (PSE) in that resources, supply chains, and interactions may be adjusted to improve efficiency of the process. In the context of this work, we wish to analyze and improve the efficiency of the system by understanding the expected behavior and outcomes of patients, in order to better understand the optimization goals of the process. The ability to predict outcomes enables the adjustment of specific parameters of the system to better accommodate and distribute the relevant resources to meet the needs of patients, staff, and other applicable members of the system. Figure 1.1 shows the PSE optimization cycle, which is composed of multiple stages of optimization, including identification of inefficiencies, analysis of applicable processes, adjustment of the process based on the analysis, and staging of the modified process back into the system flow. This work will identify inefficient processes in clinical settings, and will maintain focus on the analysis of these processes in the context of patient outcomes. Predictive analysis will act as the input
Figure 1.1: Diagram of PSE optimization cycle. This work will focus on the analysis component of the cycle in an effort to improve clinical process management.

upon which adjustments can be made to a clinical process in order to improve efficiency. Knowledge of patient outcomes is a critical component in a clinical setting, and the ability to infer outcomes may enable better management, decrease ALOS, and improve care.

1.4 Current Uses of Big Data in Industry

Advances in continuous processes, such as clinical processes, are prompting the use of large sensor arrays and monitoring tools to better serve the needs of patients and staff. The use of process monitoring systems have been the standard for monitoring different processes, including patient health and operational efficiency. As clinical processes become more heavily imbued with sensors, the volume of data being collected and analyzed by clinical staff via single displays has become overwhelming. As a direct result of this excess of process data, many sensor data streams are ignored or disregarded. In many cases these processes are monitored in ways that are
similar to non-clinical industries, such as the oil and gas sectors of chemical engineering. The use of machine learning and advanced modeling to handle continuous, and often high-dimensional, data can be applied to clinical processes using similar methodologies.

Approaches to compile the data into manageable streams have been applied, including multivariate analysis (MVA) techniques\textsuperscript{13}. MVA is able to capture covariance in processes; whether that be patient health statistics, or process statistics, is dependent on the process being monitored. Within these data driven models, a variety of machine learning and big data approaches are also applied to enable important data to be extracted and presented, providing better monitoring of processes by a single staff member. Of these approaches, projection-based methods, such as principal component analysis, are used to identify key variables in a process to use as indicators of the process performance and efficiency\textsuperscript{14}.

However, in many clinical processes, degradation of the model and rapid changes in data streams may cause these methods to fail. Efforts to integrate modern machine learning methods, for example, Bayesian Gaussian mixture models, and neural network models, have shown great stride in creating robust and reliable methods to handle process data\textsuperscript{15,16}. Current research suggests that there is no one-size-fits-all methodology for every scenario, however appropriate model choice and understanding of process data is becoming more possible that we have ever seen before.\textsuperscript{13}.
1.5 Model-Based Optimization

1.5.1 Deterministic Methods

In many operations, planning and management is handled with simple deterministic approaches, using metrics such as patients ALOS, resource requirements, bed occupancy, etc\textsuperscript{[17]}. Steady-state analytical models are common; these consist of fully-defined analytical methods used for optimization of clinical processes; frequentist statistical techniques were used to model the clinical flow environment, and a process model designed around that particular operation. Models are developed from historical data particular to a clinical process. The fitting and optimization of these models requires significant data collection from within the process, often disrupting patient and staff flow. Analytical models generally perform very well for the process they are designed for, but lack generalizability and the ability to model different clinical environments. While often very accurate, the time and costs required to build analytical models inhibit their use in large-scale operations\textsuperscript{[18]}.

Other fully-defined analytical methods include queuing theoretic models (QTMs) and queue-based optimizations\textsuperscript{[2]}. Originally proposed in the 1950’s, QTMs are still used today to simulate and optimize clinical flow processes. However, like many analytical methods, queuing theoretic models fall short in transferability and ease of implementation in similar clinical processes\textsuperscript{[2]}.

Dynamic stochastic methods incorporate some kind of regression-based stochastic model\textsuperscript{[12,19]} in order to account for dynamic system behavior. Although these methods are suitable for modeling patient flows and pathways, they lack efficacy when analyzing performance measures related to waiting times or random events\textsuperscript{[2]}. Stochastic models are designed around a process in a steady-state fashion. As a result, models are generally non-transferable to other clinical processes without ex-
tensive modification of the process model.

1.5.2 Bayesian and Markovian Statistical Inference Models

Alternative modeling methods include Bayesian Networks (BN) and Markov models. These methods utilize Bayesian statistics to iteratively predict future states of a system. Unlike frequentist statistics, Bayesian statistics relies on prior observations to predict the likelihood of newly emitted observations, effectively using past observations as guides to future behavior. These models present a unique method for predicting expected future behavior in a stochastic framework; models adapts to acute changes in behavior in the near past, allowing dynamic responses during unpredictable operational modes.

The network flow of patients within a hospital setting may be described as a Markov process, with some number of states progressing from intake to discharge. Each state can be thought of as a process within the hospital, with some probability of transition from one state to the next. The patient-flow network structure gives insight into allowed and disallowed transition probabilities, as well as bottle-neck regions within the operation.

![Figure 1.2: Example of a five-state outpatient Markov process.](image)

It may be advantageous to incorporate multiple Markov processes into a flow network; each component of which may interact with other components or share resources. While not all patients will access all components, sharing of resources and
scheduling of staff within departments is an overall constraint. Within the whole operation, each state has some probability of transition to the absorbing state (discharge), and similarly to other related states. The flow of patients through each state may occur strictly in the forward direction (that is, there is no probability of transition to a previous state), as is the case with many specialty care operations[20]. More general operations may form sub-processes which incur backward transitioning of states, as seen in Fig. 1.2, in which patients flow through each state before reaching the absorbing (or exit) state of the model. Each state \( i \) has some probability of transitioning to another state \( j \) with some probability \( p_{ij} \). States 1 and 2 have a single transition likelihood and only occur in the forward direction, while states 3 and 4 have multiple transition likelihoods in both directions. In this example, patients may receive multiple consultations and tests before reaching the absorbing (exit) state[2]. Consultation and testing states form a loop in the flow network (e.g. diagnostic testing, analysis, further testing, etc.), commonly observed in hospital settings[8]. Each transition probability is dependent on current patient status, and the duration of time spent in the previous state. States inherently have a distribution of emissions (or expected patient status, type of procedure, etc.) that dictates its activation. Duration of activation (of the state) is also a factor in transition, as the time to reach the absorption state (discharge) is directly affected by these other transitions. Because each state has some probability of transition to discharge, the overall probability of transition to discharge can be described by a phase-type distribution composed of the distributions of each state.

Hidden Markov Models (HMM) have a structure similar to a general Markov model, but with more restrictions and less definition of the model parameters. By nature, HMM’s serve an inferential purpose; models aid in making decisions based on the maximum likelihood of emission origin, and are based on limited or fuzzy
data. HMM’s must be designed and fit to a model data set which can be used to define emission probabilities and transition probabilities for the model based on statistical likelihood estimates.

If a model topology can be formulated for a process, observable data (emissions) can be used to ‘train’ state emission probabilities. Emission data is inherently grouped into bins based on which state is most likely to emit it. HMM’s flow in time, and as such states have transition probabilities which describe their likelihood to ‘jump’ from state to state. Transition probabilities reflect logical ‘next steps’ in the model progression. As emission data is introduced at each time step \( t \), the previous state of the model has a likelihood of transitioning to a new state. However, this transition is based on which state the emission fits best in, as well as the probability that the current state can move to the next state predicted by the emission. This model design generates a conditional dependency on the current state of the operation, having a direct affect on the flow and interaction characteristics of staff and patients within the model. Modes of operation may be defined in relation to a particular actions of staff and patients, providing evidence of the different underlying states of the operation. For example, changes in staff allocation or patient ALOS can indicate changes in the underlying state of an operation.

Neural networks are another class of models that are very popular for analysis of non-linear data sets, and may be formulated for the classification and prediction of patient outcomes within a clinical setting. Patient data may relate to numerous non-obvious factors within a clinical process. Their analysis using state-of-the-art techniques such as neural networks (NN) may provide a basis for modification of patient care by detecting new opportunities for process optimization. The use of NNs in image processing has shown great advances as a way to accurately detect and identify numerous medical phenomena, including cancerous cells biopsy results, tumors
and legions in MRI results, and fractures and fluid build-ups in x-ray imaging[21].

1.6 Motivation for Machine Learning and Big Data Analysis

Big data collection within clinical processes continues to increase and has the potential for organizations to improve performance analysis. The mass collection of data within healthcare operations provides an excellent opportunity for data-driven optimization models and adaptive budgeting schemes.

Clinical processes are an excellent example of a big data system; information regarding fundamental operating conditions, resource allocation, staff coverage, patient needs, patient history, and budgetary changes are tracked across entire operations. Important variables within each process give indication of the level of efficiency of that process, and as a result can be used to optimize and improve the process. In addition, the availability of data surrounding patient outcomes creates a unique opportunity to predict and theorize the means by which patient outcomes are affected by different factors.

1.6.1 Machine Learning

Clinical processes involving significant patient and staff flow contain enormous amounts of data regarding flow patterns, bottle-necking, and resource allocation. From a big data perspective, these processes present a highly complex optimization problem. Machine learning (ML) techniques may provide useful tools for processing large data sets in clinical processes. The application of ML methods to big data is practical with modern computing hardware, advances in computing, and data storage; which provide data analysts with more powerful and diverse dataset processing tools for
optimizing complex processes in near real-time. NNs, HMMs, and BNs define a full-scope toolkit, that to the best of my knowledge, has not been explored fully in analysis and optimization of clinical processes. The use of these methods have the potential to model and predict patient outcomes, needs, and operational bottlenecks in near real-time, while learning and adapting from operational data continuously.

1.7 Scope and Objectives

1.7.1 Scope of Work

This work aims to analyze clinical processes and healthcare operations data sets using machine learning and advanced stochastic methods to analyze and optimize patient care, improve operational efficiency, and predict patient outcomes. NNs, HMMs, and BNs will be the primary tools used for analysis and prediction of clinical data sets. These methods will be applied to three representative data sets to demonstrate the techniques capacity for model development and extraction of useful information. Advantages and disadvantages of each model type will be discussed in context of the dataset they are applied to, in an effort to demonstrate the effectiveness and applicability of machine learning in the healthcare field.

1.7.2 Objectives

The objectives of this thesis are:

- Evaluate potential sources of clinical and patient-centered data with an emphasis on modern data sets.

- Analyze healthcare-relevant data sets, such as human EEG data and diabetic patient data, in the context of big data and machine learning.
• Predict and classify features of healthcare systems and healthcare data relevant to the improvement of a process or improvement of patient care.

• Study the effects of different model types and methods on the extraction of useful features of a clinical dataset.
Chapter 2

Distributed Healthcare

Distributed healthcare is defined by the decentralized monitoring and treatment of patients, both in a clinical and home environment\cite{22}. Early in this century, patient healthcare was driven primarily by in-home visits from a family physician, a practice that is no longer commonplace today. However, the evolution of modern medicine has prompted the renewal of personalized, in-home care through the use of digital technologies. In general, modern healthcare is carried out in clinical environments, such as hospitals or clinics, in which advanced technologies and diagnostic tools may be housed and afforded through large-scale financial conglomerates. These institutions have high overhead and cost of care, in turn reducing the effectiveness of clinicians in the treatment and management of common conditions, both acute and chronic. A shift from centralized care toward distributed, decentralized digital healthcare is proving to be an effective means for reducing cost of care while still maintaining positive patient outcomes in the process\cite{1}.

Moving services into patient’s homes has the potential to improve the patient’s experience while also ensuring rapid diagnosis and treatment of numerous conditions. The use of digital technologies such as networked smartphones have provided clinicians with an alternative means for monitoring of patient health, while simul-
taneously reducing overhead cost and frustration. These alternative means of monitoring present significant benefits to patients who traditionally are at odds when receiving care, such as elderly or immobile individuals. In addition to more rapid identification and control of chronic illness, distributed healthcare systems provide a platform of communication that enables both caregivers and patients to better understand the care they are providing or receiving. Serving patient goals rather than simply treating a disease, modern applications of decentralized healthcare have the potential to improve quality of life of millions of patients.

Distributed healthcare practices do present certain challenges, namely in the transition from the traditional healthcare model to a home-based one. Coordination of services, both medical and financial, are leading challenges which require communities, providers, and organizations to take forward-reaching steps to successfully integrate into the new model. In order to meet these goals, common data platforms and dynamic medical information systems must be in place to provide seamless care environments and access to medical records without incompatibilities or missing information. Teams must be able to work across numerous care settings, including the patients home and smaller clinical environments.

In 2015, a case study performed by the Canterbury District Health Board (CDHB)[23] regarding the reduction of acute hospital admissions via better management of patient care through distributed healthcare systems[23] was performed. Several studies focus on reductions of cost associated with acute care, in which there has been mixed success. For example, work performed by Steventon et al[24] in conjunction with the the Partnership for Older People Projects[25], found insignificant reduction in net costs associated with avoiding unplanned admissions to hospitals. The CDHB study[23] focused on whether integrating patient-driven healthcare systems would have a positive impact on reducing acute hospital visits in an aging population, but
instead with patient outcomes being the primary metric for success\textsuperscript{[23]}. Canterbury DHB\textsuperscript{[23]} was able to demonstrate a measurable reductions in demand for acute hospital and long-term residential care services, all with an increasing number proportion of patients age 65 and older\textsuperscript{[23]}. The observed reduction in acute hospital care requirements provides evidence that distributed healthcare services, both in terms of community health education programs and personalized healthcare driven by patient-oriented healthcare models, is an effective practice.

\section{2.1 Mobile Health Systems}

Recently, the use of non-invasive remote patient monitoring has become increasingly reliable and an effective tool in healthcare\textsuperscript{[1]}. With this technology, there is better sharing and use of health care data, resulting in improved consultation and management of patients. Mobile health (mHealth) technologies, namely wireless wearable devices, allows for both automated and semi-automated gathering of patient health data. In chronically ill patients, use of automated data capture is ideal for monitoring and treatment without direct patient interaction; reducing error and providing more accurate diagnosis\textsuperscript{[1]}.

The use of mobile health systems (mHealth) is very promising emerging for collection and processing of patient wellness data. Driven primarily by wireless technologies, the use of mobile devices for data transfer have become standard in everyday practice, including use in clinical settings\textsuperscript{[26]}. Personalized medicine is also a key factor in the use of mobile devices in healthcare, as monitoring and integrated sensors provide a platform for rapid evaluation of patient health. Figure 2.1 shows the utilization of different mobile devices in research applications for the acquisition of mobile health data. In recent years, the use of personal digital assistants (PDAs) and smartphones have become dominant for collection of mHealth information. The
use of biosensors and wearable devices show less usage, though many smartphone devices support the use and integration of these technologies directly.

From a consumer perspective, mHealth benefits both acute and chronically ill patients. Acute conditions present up to 34% of all physician office visits, many of which could be treated in ambulatory settings\cite{27}. Self- and assisted-diagnosis of acute conditions can be achieved by teleconferencing or tele-monitoring by physicians, reducing the need for patient transportation as well as reducing costs associated with in-office visits. Chronically ill patients also benefit from remote monitoring of vital signs such as blood pressure, heart rate, and blood oxygen saturation, among others; while further reducing costs associated with laboratory work and clinical visits\cite{26}.

2.1.1 Trends in Remote Patient Monitoring

Remote patient monitoring (RPM) has become a central part of healthcare, in the form of patient-to-clinician monitoring. In recent studies\cite{1} with young adult pa-
tients ages 21 to 39 years old (5%), adults ages 40 to 64 years old (18%), and seniors ages 65+ years old (61%), it was found that remote monitoring was an effective tool for improving quality of care and maintenance of illness. Typically, patients studied had chronic illness related to respiratory diseases, cardiovascular diseases, weight management conditions, and metabolic disorders. In self-monitoring conditions, the primary illness is weight management. Of these patients, monitoring was typically routed to clinicians at a rate of approximately 50%, with only 18% being self-monitoring by the patient\[1\]. The primary route of data acquisition was through smartphone devices to monitor lifestyle changes and behaviors, such as caloric intake and exercise.

2.1.2 Efficacy of Mobile Health Monitoring Systems

Steventon et al.\[24\] performed a study involving chronically ill patients and the performance of mHealth in reducing healthcare needs. The study focused on patients with diabetes, chronic obstructive pulmonary disease, and heart failure. The intervention trial group evaluated the use of remotely exchanged tele-health data between clinicians and patients during their diagnosis and treatment. The control trial group included all clinically available services, without the use of tele-health technologies. The study shows that patients in the intervention group had statistically significant reductions in hospital admission, length-of-stay, and overall mortality rates compared to the usual care control group\[24\].

The use of mHealth technologies reduces physician intervention, offering potential cost reduction and improved care. Common acute and chronic conditions may be managed through passive monitoring of patients in ambulatory settings, reducing the incidence of office visits and use of specialist care. Patients with remotely manageable conditions may be better served through alert-based care systems when
abnormal readings are detected. In these cases, simple text/email/phone alerts are sent to patients and clinicians to proactively address issues before becoming significant risks to the patient\textsuperscript{[26]}. The use of machine learning algorithms for early detection of abnormal readings could allow for automated alerts and reduce the need of human-based monitoring of patients. Additionally, the reduction physicians in non-life threatening cases could reduce global overhead and cost of care. Combined, mobile health technologies and low-cost monitoring devices provide a framework with good potential to improve patient self-management and lower financial costs\textsuperscript{[24,26]}. 

2.2 Clinical Information Systems

Modern patient monitoring systems generate large amounts of data during use, which is feed into semi-automated or human-monitored alert systems; for example, tier one intensive care units (ICUs) generally monitor patient’s vital health statistics from a centralized location. Integrated networks handle both monitoring and staff response, allowing staff to better optimize patient care timelines. Although major improvements in patient monitoring have been developed, a systemic lack of incorporation of signal processing, computer science methods, biomedical engineering advancements, and mathematical models have prevented widespread advancement of patient monitoring and care\textsuperscript{[6]}. Limitations in data formatting and integration of standardized (non-proprietary) data collection techniques have also given rise to incompatibility with other monitoring systems, in turn limiting advancements in patient care, particularly in intensive care environments\textsuperscript{[6]}. In context of these limitations, traditional analysis techniques provide little insight into overall patient health or acute conditions. Dynamic analysis of patient illness is critical for improving patient outcomes in a number of environments, primarily critical care environments such as the ICU\textsuperscript{[6]}. Low sampling rates of physio-
logical data, including vital statistics and clinical notes, reduces the accuracy and speed at which a patient may be diagnosed or treated. Understanding dynamical behavior of patient health requires frequent sampling and the use of numerous data analysis techniques (linear, non-linear, stochastic, etc), many of which are overlooked or unused\cite{6}. Integration and synchronization of clinical information systems remains a relevant and desirable means for improving physician understanding of acute patient health, allowing for rapid, actionable decision making and improvement of patient outcomes.

2.2.1 Commercial Clinical Information Systems

The annual market in the United States for emergency, operative, and intensive care software solutions is expected to exceed $1.46 billion by 2018\cite{28}. The availability of commercial clinical information systems (CIS) is increasing, though no one company maintains dominance in the market. Advances in end-to-end software platforms have been primarily driven by smaller market acquisition\cite{6}. In general, CIS systems are equipped with four major operative subunits: electronic medical record databases, care orchestration and staff management systems, client/patient and clinical logistics management systems, and operations and infrastructure management systems. Each subunit is a critical part of ensuring uninterrupted operation in the clinical environment, and each provider of these services designs them differently.

The most common and similar subunit from any provider of CIS systems is the electronic medical record database, which serves as a basis for management of patient data in a secure and reliable fashion. Electronic medical records are a norm in modern healthcare environments, and thus have become interchangeable among different service providers. These databases generally contain patient intake data,
such as insurance, housing, and past medical history, as well as updated information that is managed within the clinical environment. Benefits of these electronic databases include rapid access and up-to-date information that is available to clinicians, without the need for later manual data entry. Many modern healthcare environments utilize laptop computers and other mobile devices/platforms during treatment and consultation, enabling direct update of patient medical records. These systems are managed in a secure fashion, often routed through numerous private networks within the clinical operation only, with limited access through public networks.

Care orchestration and staff management systems vary by vendor, but typically maintain functionality related to resource scheduling, staff scheduling, chart management and coding, and patient coordination services. These services provide management relating to quality of care and smooth operation within the clinical environment. Staffing needs are dependent on numerous factors including intake rates of patients, both acute and chronic, as well as operational needs like equipment maintenance and facilities maintenance.

Clinical logistics management systems handle intra-operation needs, such as patient tracking, clinical messaging, and device coordination, as well as financial and enterprise planning. These systems are generally considered ‘behind the scenes’ management as related to patient care, and provide information to clinicians and staffers simultaneously. These systems also handle equipment tracking and inventories.

Operations and infrastructure management systems handle data, networking, IT operations and device integration. These subunits also encompass security needs of the clinical environment, including, but not limited to, network and staff safety and threat avoidance. Many large healthcare operations maintain large databases.
of private information, which must be safeguarded from both digital and physical access. Server hardware and network integration must be redundant and secure. In addition to network and data security, infrastructure management systems handle resources used by buildings, such as energy, water, and waste management. In large operations, optimization of resources assists in reducing operational costs and ensuring longer lifetimes of equipment and clinical spaces.

2.2.1.1 Examples of Clinical Information System

The Hewlett-Packard healthcare information system (HCIS) is a modernized service solution stemming from the original Hewlett-Packard CareVue clinical information system developed in 1997. HCIS technology is implemented through sets of application services including: secure access solutions such as wired and wireless network access for health information exchange and point of care systems, business integrations including content and information management, business intelligence, and document capture management, clinical and research applications such as master patient indexes, clinical analytics, and device and workflow management, communications and facilities management technologies related to voice and video connectivity, patient interaction systems, digital signage, and building management, and infrastructure solutions such as data warehousing, virtualization, information technologies, and client virtualization. The IBM Digital Hospital Solution Framework is an end-to-end framework for management and integration of digital technologies in the modern clinical environment. IDHS leverages the use of health analytics, staff management, care orchestration, electronic medical records, patient services, business management, clinical logistics, operation and infrastructure management, and intelligent buildings to unify the complex networks present in clinical environments.
GE developed the Centricity Critical Care System (GE-CCCS) as a combination of smaller acquisitions that provide the components of the software/hardware platform. Through the combination of Marquette Medical Systems\textsuperscript{[31]}, Intrumentarium\textsuperscript{[32]}, and iPath\textsuperscript{[33]}, GE’s CCCS offers patient monitoring, anesthesia equipment, and operating room management systems, respectively. Outside of direct care, GE also acquired MedicaLogic and IDX systems for records management and billing/operations management. GE’s Centricity Critical Care system provides data collection through the proprietary Unity Interface Device Network (ID Network) from patient monitors and centralized data organization much like that of traditional ICU paper charting systems.

Philips Healthcare also provides end-to-end ICU management and organization systems. Most recently, Phillips released its IntelliVue Clinical Information Portfolio (ICIP) in 2007\textsuperscript{[34]}. Much like the GE-CCCS, the ICIP system is a culmination of acquisitions of technologies to provide better overall management of care in the critical care environment. Agilent Technologies Healthcare Solutions Group\textsuperscript{[35]}, Witt Biomedical\textsuperscript{[36]}, and Emergin\textsuperscript{[37]} were acquired to provide general patient monitoring, hemodynamic monitoring, and alarm systems to the ICIP system, respectively. Philips uses TOMCAT Systems\textsuperscript{[38]} and VISICU\textsuperscript{[39]} for cardiac monitoring and tele-ICU technologies.

\section*{2.2.2 Limitations of Current Clinical Information Systems}

Amid advancement in modern clinical information systems (CIS), many limitations still exist. Foremost is the inability to both capture and store high-frequency physiological data; even in state-of-the-art systems offered by leaders in healthcare such as GE or Phillips, real-time analysis of incoming data are not standard. This is in part due to memory requirements of capturing high-resolution physiological data and
lack of standardized/centralized storage within the monitoring system. This limitation is present in almost all commercially available enterprise-grade monitoring systems available today. Due to the intense amount of data collection required, many systems opt for less frequent data collection and storage, in an effort to provide useful patient metrics without the significant overhead of maintaining high-resolution data. The result of this trade-off is the maintenance of snapshots of clinical data which may or may not be useful in making informed decisions regarding patient care\cite{6}. In addition, no standards for the frequency of collection of patient data have been established by major commercial software providers.

There is also limited post-processing, statistical analysis, and risk assessment of collected data\cite{40}. Generic statistical trends, such as mean, median, or mode, are often the only analysis performed on data, severely reducing potential insight from more dynamic analyses. Research surrounding the dynamics of physiological data have shown that nonlinear and stochastic models provide significantly better predictive insight about overall patient health than do frequentist analysis techniques\cite{40}. Emerging methods from the engineering and information sciences have shown an ability to more quickly and accurately identify issues in patient health before they become critical. Promising advancements in critical care are then heavily dependent on the use of advanced analysis to best predict complex relationships among physiological data. In turn, these analysis may serve as a means to identify and personalize treatment options for improved patient care in both critical and conventional settings. Ideally, these systems should provide both direct data and proposed multimodal models to physicians, improving actionable decision making processes on a real-time basis\cite{40}.

In the context of intensive care, little advancement has been made in the use of visual monitoring systems for identification of patient health. With increased vol-
umes of collected clinical data, traditional visualization methods have become less effective in terms of readability and comprehension \cite{41}. Imhoff et al \cite{42} noted that when caring for critically ill patients, clinicians incurred difficulty judging and relating two or more patient-oriented variables simultaneously. Large numbers of visual representations of patient vital statistics and the inability to effectively correlate them with other patient information results in less effective patient care and preventable errors \cite{41,42}. 
Chapter 3

Clinical Data sets

3.1 Electronic Medical Records

Electronic medical records (EMR) are the primary means of recording patient medical histories. In terms of both maintenance and transferability, EMR management has provided significant benefit to the healthcare administration. Standardized formatting has also allowed for better systemization of care, as medical records may be viewed and edited by most healthcare providers without the need for non-standard software. In addition, the ability to rapidly transfer records through digital channels, both intra- and inter-operationally, allows for smooth patient transfers without loss of information or care aptitude.

Medical records are stored in common file formats based on either XML or JSON scripting, allowing for easy interchange, modification, and readability on a variety of platforms. The Clinical Document Architecture (CDA) is a document markup standard that specifies the structure and semantics of clinical documents for the purpose of exchange in healthcare. CDA is a base standard which provides a common architecture, coding, semantic framework, and markup language for the creation of electronic clinical records\[43\]. CDA documents are coded in HTML and XML,
remain human and machine readable, are templated in context to clinical objectives, and provide object-oriented associations and classes to allow for reuse and flexible implementation\cite{43}.

![Figure 3.1: Generic structure of a CDA record.](image)

Figure 3.1: Generic structure of a CDA record.

![Figure 3.2: Basic CDA document entry in XML. Elements in square brackets represent replaceable information.](image)

Figure 3.2: Basic CDA document entry in XML. Elements in square brackets represent replaceable information.

An extension of this standard is the Consolidated Clinical Document Architec-
<entry>
  <observation classCode="OBS" moodCode="EVN">
    <templateId root="2.16.840.1.113883.10.20.6.2.10"/>
    <code code="50373000" codeSystem="2.16.840.1.113883.6.96" codeSystemName="SNOMED-CT" displayName="Body height"/>
    <statusCode code="completed"/>
    <effectiveTime value="20121114"/>
    <value xsi:type="PQ" value="177" unit="cm"/>
  </observation>
</entry>

Figure 3.3: Basic CDA document entry in XML. The formerly undefined fields are completed and displayed as they would be in a complete CDA document.

CaT (C-CDA), which specifies templated document libraries and prescribes their use for specific applications and document types. Coded discrete data elements may be coded in all clinical documents, including transition of care, clinical summaries, vital health summaries, quality reporting, and financial documentation.

### 3.2 Data sets

Data sets often contain multiple optimization-relevant features generally related to patient focused or operation focused metrics. In order to better understand each feature set in more detail, efficient extraction and separation of each subclass is required. Real-time metrics such as patient vital signs may be useful for the identification of declines in patient health, while long-term metrics such as past medical conditions and procedures may be useful in the identification of future health risks and alternative treatment options.

Data sets used in this work consist of patient and operation oriented variables. Patient oriented variables consist of general census statistics such as place of resi-
dence and insurance information, vital health statistics and associated timestamps, allergies, medications and immunizations, diagnostic results, medical encounters, plans of care, and social histories. Operation oriented variables consist of medical order times and dates, as well as bed types, associated costs, and financial information. Each subset of data yields different insight surrounding pre-visit, during-visit, and post-visit patient behavior and expected outcomes.

**Pre-visit Data**  
Historical patient data, often in the form of an electronic medical record, as well as registration information such as patient name (ID) and patient address comprise pre-visit information. Primary constituents of a patient's medical history include patient date of birth, ethnicity, social habits (smoking, drug use, etc.), laboratory results, current medications, allergies, and previously recorded vital signs. Emergency room patients and walk-in patients are likely to not have fully defined pre-visit records. Other information such as patient origin and preliminary gathered information may yield valuable relationships between locale and general health. Pre-visit data is also valuable to clinicians for analysis of the general health of a population, specifically epidemiological analysis such as the spread of influenza or other seasonal diseases.

**During-visit Data**  
Monitoring and maintenance of patient health care during the length of stay remains the primary goal of clinical operations. Each patient represents a continuous stream of information, with varying outcomes associated with each statistic. Vital health statistics that correlate with negative outcomes during patient residence become of particular importance for predictive and inferential statistical models, and decision making surrounding patient outcomes. Analysis of historical process data allows for outlier detection in live data sets.
Post-visit Data Analysis of post-release patient behavior is of high importance for a clinical operation. Declining health after release from a clinical process is a leading cause of patient re-admission in the time period shortly after discharge. Mistakes, missed tests, and other operational errors can promote a rapid decline in health (RDH) of newly released patients; tracking re-admissions, patient paths, and staff actions may reveal factors unconsidered in previous visits, in turn reducing the number of required future visits. Other factors considered in post-visit data are those related to financial reimbursement. Patient groups who are unlikely to pay medical costs may be offered financial aid or guidance prior to discharge, improving payment likelihood over time.

3.3 Ethical use of Health Data

The adoption of electronic medical records (EMRs) in healthcare practices has greatly accelerated interest in the direct use of health data for research and personalized medicine[44]. Detailed collection of patient health data in EMRs present new opportunities for use of advanced modeling methods. The real-life characteristics of modern EMRs make them attractive resources for public health and data science research. However, medical confidentiality regulations and the inclusion of personal information in these records requires patient consent before use in any research or public health field. In addition, these records are a target for abuse by third parties, in turn limiting their availability for legitimate use in an effort to protect patients right to privacy[45]. Even in the case of select patient consent, the refusal of consent of other patients in a study group may form a biased sample set, which severely undermines record-based research efforts and reduces the legitimacy of study outcomes[45]. Combined, these restrictions create a poor environment for real-life record-based research practices, introducing the need for sanitized data sets or
3.4 Generation of Synthetic Patient Data

Synthetic medical records are often an ideal choice for methods development in public health and related fields as they provide complete, unbiased reports that can be used to test and model hypotheses that normally require consensually acquired patient medical records. The compilation of these records vary between research interests; demographics data alone is often useful for large scale studies focused on public health, whereas the addition of detailed medical histories may be better suited to patient-oriented studies[44].

Figure 3.4: A rendered CDA document populated with synthetic patient data. The document contains demographics data, as well as patient medical history in 11 subcategories.

The generation of in-silico patients that include detailed demographics and historical medical records for patients has potential to be a useful source of data when
Figure 3.5: A rendered CDA document populated with synthetic patient data. The document displays patient demographics, vital sign history, and immunization history. Each subcategory of the document is formatted similarly.
real data is unavailable. This type of data may be generated using a patient generation framework, such as the Synthea synthetic patient population generator\cite{46}, which uses United States public health statistics to accurately simulate patients. Generated EMRs are relevant to specific geographic areas, and contain realistic, statistically and demographically accurate synthetic patient histories from birth to death. Patient histories are generated using Monte Carlo simulation methods\cite{46}, in which patients move through simulated events based on statistical likelihood of occurrence since time of birth. The resulting data includes patient demographics, allergies, medications, conditions, care plans, observations, encounters, procedures, immunizations, and relevant imaging studies. Figures 3.4 and 3.5 detail a synthetic medical record with full patient history and demographics. Currently implemented events and encounters include routine infant/child health check, essential hypertension, diabetes mellitus, normal pregnancy, respiratory infections (pharyngitis, bronchitis, sinusitis), general adult medical examination, disorders of lipoid metabolism, ear infections (otitis media), asthma, and urinary tract infections. In addition, resulting causes of death include natural causes, ischemic heart disease, lung cancer, Alzheimer’s disease, COPD, cerebrovascular disease, road injuries, self-harm, diabetes mellitus, colorectal cancer, and drug use disorders (limited to opioids).
Chapter 4

Machine Learning Methods

4.1 Framework

This work focuses on the use of methods for rapid evaluation of data sets and identification of important features for predictive analysis. Using a combination of machine learning methods, it is possible to create a flow chart of applicable techniques based on the type of data being processed.

Figure 4.1 shows a generalized process pathway for preparation and analysis of a dataset. Preprocessing of data is a crucial component of any machine learning method, especially when working with missing data or magnitude-sensitive methods, such as neural networks. The proposed framework segments data sets along a tree-like pathway based on the makeup of the data, beginning with preprocessing and concluding with model development. Prepared data may be analyzed for dataset structures, which may prove beneficial for model development and design. The framework contains general methods for extraction of clusters in the dataset, as well as artificial labeling routines to enable use of both supervised, and unsupervised learning methods. There exist numerous techniques for analysis of a given dataset, based on the type and structure of the data present. We wish to develop a
generalized framework as a guide, with respect to the type of data being analyzed, which can used to identify relevant pathways for study and methods applicable to that study.

The framework is defined in three stages: preprocessing, structure analysis, and model development. The preprocessing stage aims to standardize a dataset in such a way that it is compatible with a majority of machine learning methods. Specifically, data is checked for leakage, converted to consistent numerical representations, and stripped of missing or incomplete values.

Once the data has been preprocessed, it is branched to either structure analysis or model development, based on the needs of the researcher. Structure analysis is highly relevant to data that is unlabeled, of unknown origin, or lacking obvious structure. Methods proposed in this branch include clustering and labeling routines such as t-stochastic neighbor embedding, k-means clustering, and HDBSCAN clustering. Each method provides different means for identification of data similarities. Model development begins by splitting into two main pathways, related to the labeled or unlabeled nature of the data. Unlabeled data may be processed in an unsupervised manner, but is bifurcated by the need for time-series handling using techniques such as hidden Markov models or restricted Boltzmann machines, respectively. Labeled data is processed similarly, but with a larger set of applicable methods for analysis. This work emphasizes the use of supervised learning techniques such as Bayesian classifiers, neural networks, and decision tree models for processing of labeled data, and unsupervised techniques such as hidden Markov models and general mixture models, for the processing of unlabeled data. Additional techniques are presented as alternatives, or as supplemental to these methods.
Figure 4.1: Flow chart diagram describing methods applicable to processing particular data types. Terminating nodes shown in green are used in this work. Terminating nodes shown in yellow may prove useful for analysis of clinical datasets, but are not used formally in this work.
4.2 Reasons for Dataset Analysis

Modern health care processes have ongoing efforts to improve speed of care and ease over-crowding in clinical settings. Amid operational improvements, errors in patient care and efficient scheduling of staff may still affect operations. Optimum staffing allocations may considerably reduce delays in clinical operations, improving patient care quality. Health care operations provide opportunities for the care and development of real-time, state-of-the-art operations management and scheduling technologies. Through the use of historical clinical data, important relationships within an operation may be identified and modeled to better understand specific impacts on the efficiency of the operations. High-frequency patient monitoring data may serve as a means for identifying critical interactions which lead to declines in patient outcomes. These relationships may stem from staff-patient interactions, operational procedures, or a combination thereof. This work aims to identify and tune ML methods for determining critical patient-oriented areas of improvement, as well as identify key areas for operations improvement within clinical settings.

Many operations monitor and collect numerous time-oriented variables or attributes related to patients, staff, operating conditions, and budgets, creating high dimension data sets. Inherent non-linear and hidden relationships within high-dimensional data sets may be useful during optimization; the challenge of uncovering these relationships without substantial pre-processing or assumptive modeling remains a primary objective of study. Data sets generally contain a large number of discrete or continuous variables, usually on the order of hundreds of variables. The task of interpreting these still relies on empirical analysis, in particular the use of visualization methods to identify correlations. However, this may fail when processing high-dimensional data, as visualization is only applicable to two and three dimensions, limiting the number of comparisons that can be made simultaneously.
Because of this, valuable relationships within the data may be overlooked, or require significant effort to discover. For example, 2-variable comparisons on 100 variables requires about 5000 separate analyses. It is generally desirable to map higher dimensional spaces onto a 2-or-3D space to distinguish groups and provide better labeling and classification.

4.3 Dataset Structure Analysis and Metrics

4.3.1 Terminology

**Dimensionality.** Data dimensionality refers to the number of random variables contained within a dataset. These are often classified as having low or high dimensionality, with low indicating three or less random variables.

**Dimensionality Reduction.** This is the process of reducing the number of random variables in a dataset. The embedded space is designed such that it preserves the information from the high dimensional data in a lower dimensional space. This process is also referred to as embedding, and it is a common goal of dataset analysis.

**Mutual Information (MI).** This metric between a set of two random variables describes the inherent mutual dependence between the two variables. Often seen as a measure of the information (and thus the strength of the correlation) between two variables, the MI is an objective metric that determines how alike the product of the marginal probability distributions, $P(x)P(y)$, is to the joint probability distribution, $P(x,y)$. Mutual information analysis is commonly used as a predictor of variable/feature importance.
**Dataset Topology.** Data can be represented by its geometry in embedded 2D/3D space, most commonly as clusters. The shape of each cluster can drastically affect the success of certain unsupervised learning techniques, such as K-means clustering and other labeling routines that inherently rely on the shape of clusters they are processing. The convexity of a cluster is roughly a measure of roundness, where highly convex is encompassed by a circle or sphere, in 2D or 3D, respectively. Convexity is a critical measure when describing clustered data, as deviations from ideal convexity present a challenge for labeling routines which rely on distance from cluster centers for determination of cluster identity. Intra-cluster spread is also an important factor for labeling routines, and is commonly described by the variance about the cluster center. Isotropic clusters are defined by having uniform variance about the cluster center, and non-isotropic clusters defined by a non-uniform variance about the center.

**Kullback-Leibler Divergence.** Also known as *relative entropy* or *discrimination information*, the Kullback-Leibler divergence (KLD) is a measure of how one probability distribution varies (or diverges) from another.\textsuperscript{[47]} When comparing two statistical inference models, the KLD represents information gain obtained by using the mapping from a distribution $Q$ to a parent distribution $P$, where the dimensionality of $Q$ is lower than that of $P$. Given the discrete probability distribution $P$ and its low-dimensional embedding, $Q$, the Kullback-Leibler divergence is defined as:

$$D_{KL}(P||Q) = \sum_i P_i \log \frac{P_i}{Q_i}. \tag{4.1}$$

**Bayes Rule.** Given a model $M$, and data $D$, we can write Bayes’ Rule\textsuperscript{[48]} as follows:

$$P(M|D) = \frac{P(M) \prod_i^d P(D_i|M_i)}{P(D)} \tag{4.2}$$
Where $P(M)$ is known as the prior knowledge, and represents the likelihood that a sample is of a certain class before drawing the sample. In general, this is simply the frequency of each class in the model. If one class is more likely to be observed, a sample drawn is more likely to belong to the more frequently observed class. $P(D|M)$ is the probability of observing a given piece of data $D$ under the current model $M$. $P(M|D)$ is known as the posterior probability of a class in the model being the class that generated the data. This equation forms the basis of most probabilistic modeling, with interesting priors ($P(M)$) allowing for useful information to be encoded in the model before calculation.

**Akaike Information Criterion (AIC).** AIC is a method for determining the relative quality of a statistical model on a given set of data. The AIC estimates the quality of the model relative to other models of similar type.

$$
AIC = 2k - 2\ln(\hat{L}).
$$

(4.3)

For a given statistical model of a dataset, $S$; $k$ is an estimate of the model parameters and $\hat{L}$ the maximum value of the likelihood function for the model $S$. For a set of models, the one with minimum AIC is preferred. $AIC$ rewards goodness of fit based on the likelihood function, with a penalty that increases as the number of model parameters increases. The penalty discourages overfitting of the model, because goodness of fit is always increased by increasing the number of parameters.

**Bayesian Information Criterion (BIC).** BIC is a method for calculating the relative quality of a statistical model on a given set of data. Much like the AIC, the BIC uses a model likelihood function and a penalty term that penalizes model overfitting. BIC is defined as:
\[ BIC = \ln(n)k - 2\ln(\hat{L}) \quad (4.4) \]

Where \(x\) is the observed data, \(n\) is the number of samples in the dataset, \(k\) is an estimate of the number of model parameters, and \(\hat{L}\) is the maximum value of the likelihood function for our model \(M\), such that \(\hat{L} = p(x|\hat{\theta}, M)\) where \(\hat{\theta}\) are the parameters that maximize \(\hat{L}\).

### 4.3.2 Feature Importances

Feature importance is a critical component of understanding data and its ability to be used for predictive analysis. Data sets often contain a large number of features or variables, many of which may not be insightful to an objective or target. Identification of important and unimportant features should be an initial step when working with large data sets, particularly when the features are not intuitive. In many cases, elimination of low- or high-variance features is the end goal of feature selection. In cases where the target feature has a high variance, low-variance predictive features in the dataset offer little benefit to models, and can lead to over-fitting. In the opposite case, a low-variance target may not respond well to high-variance predictive features, where under-fitting of a dataset may occur and limit the predictive power of the model. The latter case is less problematic than the former, as the non-linear nature of many predictive models may still gain insight using high-variance predictors.

### 4.3.3 Model Insights

In many applications, machine learning models have a tendency to be ‘black boxes’, referring to the fact that they are able to make good predictions, but lack a solid logical explanation for these predictions. A lack of understanding of model features is
often a contributing factor to the unknown nature of machine learning predictors. In certain cases, a misunderstanding of dataset variables can misrepresent the predictive power of a model as being greater than it truly is. This occurs when variables of a dataset are treated as predictive, when in reality they are falsely correlational to the target variable or outcome. False correlation, also known as *data leakage*, occurs when a variable represents information that would only be available after the target variable has been defined. Model insights are then a useful means for determining the usability of a feature for debugging, informing feature engineering, guiding future data collection, informed decision making, and most importantly, generating trust in the model.

**Data Leakage.** One of the most important factors when building good models is the prevention of leaky data. Specifically, data leakage causes a model to present high prediction accuracy until it is used for new decision making, when the model becomes inaccurate.

There are two main types of data leakage: (a) target leakage, and (b) validation leakage. Target leakage is the most common form, and usually stems from features that are updated or created after the target value has been evaluated for the dataset. For example, take a dataset shown in Table 4.1, used to predict whether a patient has an infectious disease, based on a variety of health metrics.

<table>
<thead>
<tr>
<th>age</th>
<th>sex</th>
<th>took_antibiotics</th>
<th>body_temp</th>
<th>has_infection</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>M</td>
<td>no</td>
<td>37</td>
<td>no</td>
</tr>
<tr>
<td>29</td>
<td>F</td>
<td>yes</td>
<td>39</td>
<td>yes</td>
</tr>
<tr>
<td>42</td>
<td>F</td>
<td>no</td>
<td>37.3</td>
<td>no</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>M</td>
<td>yes</td>
<td>38.2</td>
<td>yes</td>
</tr>
</tbody>
</table>

The dataset contains a target variable *has_infection* and a predictive variable.
took_antibiotics. In this case, the predictive variable was created after the target variable; it can be rationally argued that someone who took antibiotics was successfully diagnosed with an infection and was treated accordingly. We note the similarity between took_antibiotics and has_infection; these two variables demonstrate perfect correlation, but offer little predictive insight when trying to determine if the patient had an infection. Under validation conditions, the predictive variable would not be available (as we are trying to predict whether a patient will contract an infection), and as such is unusable in model development. Leakage is easier to discover in well-labeled data sets that contain informative label names, such as Table 4.1, that can be identified through human intuition about the time/tense of the variable. In data sets that are un-intuitively labeled, mutual information scores and other feature selection routines may be beneficial to the discovery of leakage, as is simply learning more about the nature of the dataset and its origins.

Validation leakage is less common than target leakage, and pertains primarily to preprocessing techniques. It is important to maintain full segmentation of training and validation data when fitting a model. For example, a common preprocessor is dataset matrix completion (imputation), which relies on filling in missing data based on the mean or median values of a feature in a dataset. Validation is designed to measure how well a model performs on data it has not yet been trained on; if a dataset is completed using information from the full dataset before being split into train and test data, the subtle modifications from the validation data may affect the preprocessing and result in a model that has excellent validation scores, but performs poorly when deployed on new data.

While leaky data is not always present in a model, it is best to test for it before selecting and fitting a model. Heuristically, if a model is fit using the complete dataset and presents an unusually high prediction accuracy (>95%), there may be
data leakage. While this approach is not always accurate, models that predict with such high accuracy should be investigated to ensure they are indeed free of leakage. A simple data comparison can be performed to assist the identification of leakage.

**Permutation Importance.** Once a dataset is leakage free and has been fit to a model, identification of the critical factors to model performance is the next step. There are many ways to measure feature importance, some applicable to pre-fitting and others post-fitting. However, for model development, we care most about feature importances as related to the post-fitting predictive capacity of a model. One such method is permutation importance (PI)\[^{49}\]. PI works on the principal of variable sensitivity and the importance of each variable to the predictive capacity of a model. In addition, it is calculated after a model is fit, eliminating the need for multiple fittings; an added benefit when a dataset is very large or a model highly complex. Compared to other methods, permutation importance stands out as a fast, easily understood method and demonstrates good measures as a feature importance metric.

PI operates on a fitted model by sequentially shuffling each variable (column), and performing prediction on the modified dataset using the fitted model parameters. The accuracy of the shuffled prediction is scored against the accuracy of the predictions made by the unshuffled dataset. PI tests the predictive loss of a model caused by shuffling a feature that is used by the model. This ‘shuffling and predicting’ is performed on each feature of the input dataset, and the new predictions are compared to the true target values of the original model. The deterioration of predictive accuracy measures the importance of the feature being tested, where higher predictive loss indicates greater importance. Implicitly, this method tests how important a variable or attribute is to the predictive power of the model, and not the dataset as a whole. Thus for models that perform poorly on the non-shuffled data,
permutation importance offers little insight about the model.

The stochastic nature of permutation shuffling does introduce some variation into performance/deterioration of the shuffled feature. In order to capture the randomness in the PI calculation, the process is repeated with multiple shuffles of the data. The variance in performance is recorded and reported with the mean of the tests to evaluate the effect of random shuffling. In some cases, the mean PI values return a negative number, indicating that the shuffled column improved predictive performance; this occurs when an unimportant feature, which should have had an importance close to zero, caused predictions on shuffled data to be more accurate. The occurrence of negative PI values is indicative of a low-variance feature of the dataset, and is more common in smaller data sets where shuffling has a higher chance of maintaining order of some of the information in the feature. This is particularly prevalent in categorical variables, especially those with few unique classes. Shuffling a column containing a small number of unequally distributed classes may result in many rows containing the same class as before shuffling, artificially reducing the permutation of the column and producing less degradation of prediction, in turn reducing the implied importance of the feature.

4.3.4 Dimensionality Reduction, Embedding, and Clustering Methods

Dataset clustering can be considered one of the most fundamental unsupervised learning techniques used in modern big data analysis\[50\]. Its application extends to the discovery of structures within an unlabeled dataset, a task rooted in finding similarities between subsets of data. By definition, a cluster is a collection of objects that exhibit similarity among themselves; inherently, the task of defining ‘similar’ (or dissimilar) data is not simple, as data sets may be sparse, non-linear, and large.
The process of embedding a dataset from high-to-low dimensional space is not a novel task; numerous techniques have been proposed which have the capacity to reduce dimensionality\[^{51,52}\]. However, these methods often impose significant drawbacks that restrict their application to select cases and data sets. Commonly implemented methods include multidimensional scaling, principal component analysis, and local linear embedding, each prompting different challenges\[^{53}\].

Another important factor to consider when handling large data sets is the learning computational footprint. Algorithmic speed and scalability influence practical usage of different learning techniques. Under these constraints, clustering is a popular approach to segment and parallelize analysis of large data sets.

**Stochastic Neighbor Embedding.** Analysis of high-dimension data sets is common practice in an increasing number of fields. Autonomous dimensionality reduction is a practical and common technique used on large data sets as a pre-processing step or data analysis, whether to simplify and compress a classification task or produce a tractable 2D or 3D dataset that can be visualized\[^{53}\].

Stochastic neighbor embedding (SNE), proposed by Hinton and Roweis\[^{53}\], is a method for probabilistically embedding high-dimension objects (data points) onto a low-dimensional space such that the embedding preserves initial neighbor similarities and identities without forced mixing in the low-dimension embedded space. The algorithm attempts to minimize the Kullback-Leiber divergence between the high dimensional data and the desired reduced dimensional space. This method relies on a relatively simple cost function, in particular the Kullback-Leibler divergence (see Section 4.3.1), which produces a simple gradient used to adjust embedded point positions. In general, this method yields areas of similarity on the newly embedded space, such that data near each other are related by some \(t\)-distributed probability function. In practice, the method is used to embed a high-dimension dataset on to
a 2D or 3D space in an effort to better identify clusters of data via visualization or secondary cluster labeling techniques\textsuperscript{[51]}. For example, related but unique parameters such as *heart rate* and *heart medication*, will tend to be close to one another on the mapping, without directly associating *rate* and *medication* as part of the same cluster.

**t-Stochastic Neighbor Embedding.** t-SNE is an adaptation of the Stochastic neighbor embedding method proposed by Hinton and Roweis\textsuperscript{[53]}, that improves upon ease of optimization and output visualization. The cost function of the SNE method is difficult to optimize and generally results in a crowding of clusters around the center of the embedded mapping. A modified technique, titled *t*-distribution stochastic neighbor embedding, defines the cost function by using a symmetrized version of the original SNE cost function, and uses a Student *t*-distribution in place of a normal distribution. The use of a modified cost function provides a univariate objective function upon which gradient based minimization approaches may converge more rapidly. The use of a long-tailed *t*-distribution within the embedded space promotes separation of clusters based on similarity (or dissimilarity) of their cluster centers. In general, t-SNE decreases the tendency for cluster crowding in the center of the embedded space, in turn improving the identification of cluster structures across a variety of scales. The method has gained popularity in many fields that utilize highly non-linear data sets\textsuperscript{[51]}. Much like its parent method, t-SNE is a primary technique for embedding a high-dimension dataset onto a 2D or 3D space in an effort to better identify clusters of data through visualization and other labeling techniques.

**Autoencoders.** As an alternative to conventional methods such as t-SNE, the use of autoencoders have gained popularity due to their simplicity and application to non-linear data sets\textsuperscript{[54]}. Autoencoders are a type of artificial neural network\textsuperscript{[54] used
to encode data into an efficient, parameterized representation typically with the aim of dimensionally reduction. It is a type of data compression algorithm in which the compression and decompression functions are specific to a dataset and are inferred directly from the data. In general, autoencoders require no specific human input or engineering, but can result in lossy compression\textsuperscript{[55]} due to their data-specific nature. Autoencoders do not generalize well to other data sets. This stems mostly from the nature of neural networks, in which specific training examples build the weights matrix which is used to compress the data. For example, an autoencoder trained on a dataset of car pictures will do poorly encoding a set of faces. This however can be a useful feature, as autoencoders learn automatically, meaning it is easy to create specialized instances of the algorithm that perform well on a specific input type. The method requires no crafted input, just relevant training samples\textsuperscript{[55]}.

Building an autoencoder requires three principal components: (a) an encoding function, (b) a decoding function, and (c) a loss metric by which to calculate the information loss during compression and decompression. The encoder and decoder functions are typically neural networks, which are parametric by nature, and the loss function is typically a mean square error function between the input data and the reconstructed data. Autoencoders excel at particular tasks, notably data denoising and dimensionality reduction. Under appropriate conditions, autoencoders can learn more representations of a dataset than conventional methods like principal component analysis (PCA)\textsuperscript{[55]}.

**Markov Transition Fields.** The study of time-series data is often focused on predictive analysis of future events. However, time-series data can also be used for classification tasks such as determination of process regime or point-of-origin identification. One such method for analysis of time-series as a classifier is the Markov transition field (MTF)\textsuperscript{[56]}.
We first define the Markov transition matrix, which given a time series $X$, we identify a number of quantile bins $Q$ and assign each $x_i \in X$ to a bin $\{q_j | j \in [1, Q]\}$. Using the binned data, we construct a $Q \times Q$ weighted adjacency matrix $W$ that describes the frequency of transition from each bin to the next. The weights are described by the frequency that a point in $q_j$ is followed by a point in $q_i$. This weight matrix is then aggregated across each point in the original time series by assigning the probability from the quantile at that time step $i$ to the quantile at the next time step $j$ within a matrix $M$, called the Markov transition field matrix. This final matrix $M$ describes the time series data in terms of probability of transition to a new state, much like a one dimensional Markov chain.

The utility of this transformation lies in the output representation of a square matrix. The matrix encodes the original time series from 1D to 2D, allowing it to be processed as an image, rather than a stochastic time series. The natural next step in this analysis is the use of image analysis techniques such as convolutional neural networks, which may be used to classify the ‘image’ based on the features contained therein.

### 4.3.5 Labeling Routines

**K-Means.** One of the simplest and widely used clustering techniques is the k-means algorithm. Contextualized around uniform, convex, isotropic clusters, the k-means algorithm classifies a dataset through some fixed number of clusters $k$. The K-Means algorithm attempts to partition $X$ samples into $k$ distinct clusters of equal variance. Each cluster may be represented by the mean of the samples in that cluster, $\mu_k$, often called the cluster centroid. The centroid is not required to be a point in $X$, but it does reside in the same space (dimension) as the samples in $X$. The simplest means for defining initial clusters centroid is to randomly sample
(farther apart is better) points in the dataset. Theoretically, sampling random initial centroids guarantees a solution, but in practice has shown to produce poor results and to be slow-converging\footnote{57}. Augmenting this initial selection process with an improved initial guess, the \(k\)-means++ algorithm proves beneficial in both accuracy and speed of clustering convergence\footnote{57}.

The \(k\)-means++ algorithm consists of three main steps:

1. Take one centroid \(\mu_1 \in X\), chosen uniformly at random from \(X\).

2. Take a new center, \(\mu_i = x \in X\), choosing \(x\) with:

\[
p(x) = \frac{D(x)^2}{\sum_{x \in X} D(x)^2}
\]

(4.5)

3. Repeat Step 2 until all \(k\) centroids have been chosen.

Where \(\mu\) is the centroid of a given cluster, and \(D(x)\) is the shortest distance between a given data point and the closest centroid that has already been chosen. After initialization of the cluster centroids, the nearest neighbors of the initial clusters are enumerated and assigned to their associated centroid. This process completes the initialization stage of the algorithm.

The new barycenter of the centroid groups is calculated and used as the new centroid. In order to find the best fit, a scoring function called the inertia is minimized for each cluster. Inertia is also known as the intra-cluster sum-of-squares error, and relates to the distance each value is to its cluster centroid. The inertia of the fitting is defined as:

\[
I = \sum_{i=0}^{n} \min_{\mu_k \in N} (||x_i - \mu_k||^2),
\]

(4.6)

where \(I\) is the sum of the minimum distance from \(x_i\) to its nearest cluster center \(\mu_k\), and \(n\) is the number of samples in \(X\).
The smaller the inertia, the more internally consistent the clusters; as such a fitting with an inertia of 0 would indicate a perfect fit. However, the inertia function is susceptible to some drawbacks, namely:

(a) It assumes a convex and isotropic cluster when sampling, which may not always hold true. Elongated or highly anisotropic data can result in poor performance.

(b) Inertia is not normalized. High-dimension data sets can inflate Euclidean distance metrics (i.e. the curse-of-dimensionality) and artificially induce ill-fitting clusters. In this case, it is advisable to reduce the dimensionality of the dataset before attempting to run the k-mean algorithm, usually with a manifold embedding technique such as t-SNE.

The process of finding cluster centers and calculating the resulting inertia is repeated iteratively until the variance of the updated centroids does not change significantly.

**HDBSCAN.** This is a recent algorithm\[^{58}\] with the goal of allowing varying density clusters to be identified without hard labeling. The method is based on the density of similar values, but allows points to fall out of a cluster and be labeled as noise if it does not fit well in the cluster. The algorithm begins by calculating the pairwise distances between each point and populates a matrix of these values.

Next, the algorithm calculates the maximum distance to its $k^{th}$ nearest neighbor, denoted the core distance as,

$$ core_k(x_i) = d(x_i, x_k), \quad (4.7) $$

where $x_i$ is a point in $X$, and $x_k$ is the $k^{th}$ nearest neighbor to $x_i$. 
Next, the algorithm calculates the reachability of each point from a starting point $x_i$, formally called the mutual reachability distance, for each point to its neighbors as:

$$d_{MR}(x_i, x_j) = \max\{\text{core}_k(x_i), \text{core}_k(x_i), d(x_i, x_j)\},$$ \hspace{1cm} (4.8)

where $d(x_i, x_j)$ is the original distance metric between points $x_i$ and $x_j$.

Once the mutual reachability is tabulated, the algorithm generates a minimum spanning tree for the mutual reachability distance matrix using Prim’s algorithm. This minimum spanning tree is then converted into a hierarchy of connected components, and is performed by creating a disjoint-set data structure (DSDS). The DSDS tracks a set of elements partitioned into a number of disjoint subsets, effectively grouping the points by their mutual reachability. This forms a dendrogram for the mutual reachability of each point. At this point, HDBSCAN implements a pruning algorithm that is based on the minimum cluster size, which removes any arms of the dendrogram with less than the specified number of points. The resultant clusters are then allowed to split again based on a metric of cluster lifetime, with some value between the minimum and maximum lifetime designating the point at which a cluster may be broken into a separate cluster. The method is detailed further in the paper published by Campello et al.\textsuperscript{58}.

**General Mixture Models (GMM).** These can be used for dataset labeling, and in some cases dataset clustering. Their use as a labeling routine is based on the ability to determine an ideal number of clusters within a dataset in its high-dimensional form. A GMM is defined by two main parameters: (a) the covariance type, and (b) the number of components. The covariance of the model defines the dimensional relationships within the dataset, or more concisely, the dependence among components. Table 4.2 lists the four covariance types used in GMMs.
Table 4.2: Covariance Types in GMMs

<table>
<thead>
<tr>
<th>Covariance Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>Each component has its own general covariance matrix</td>
</tr>
<tr>
<td>Tied</td>
<td>All components share the same general covariance matrix</td>
</tr>
<tr>
<td>Diagonal</td>
<td>Each component has its own diagonal covariance matrix</td>
</tr>
<tr>
<td>Spherical</td>
<td>Each component has its own single variance</td>
</tr>
</tbody>
</table>

In order to determine the ideal number of components of the model, a grid search of model components can be performed using the Akaike information criterion (AIC) or Bayesian information criterion (BIC) as an estimator for the goodness of fit of the model. Both the AIC and BIC are model-specific relative metrics that compare the information loss between similar statistical models. By fitting a GMM to a variable number of components and comparing the AIC/BIC of each model, the model representing the best fit is selected from the lowest AIC/BIC score.

4.4 Supervised Learning

This type of learning relies on the use of labeled data during training to predict and classify data into distinct groups. Supervised learning methods build models around input and output data in an effort to classify or regress new data sets from the same source. The use of supervised models is advantageous when a dataset contains labels that are relevant to an outcome that is non-obvious from the structure of the data.

4.4.1 Neural Networks

Dense Neural Networks. An artificial neural network (ANN) is a type of non-linear weighted regression model that utilizes multiple layers in series to classify
or regress a dataset. ANNs are composed of multiple layers, with each layer being composed of a series of stacked input-output units called neurons. Figure 4.2 shows a schematic of a neuron, which is a computational unit that takes inputs, performs a weighted summation of the inputs, and outputs the value of the summation, with an additional bias term added to the summation if needed. The final summation is then passed through an activation function $f$. This process is described by,

$$\text{Output} = f(\mathbf{X} \cdot \mathbf{W} + b),$$

(4.9)

where $\mathbf{X}$ is the vector of inputs, $\mathbf{W}$ is the vector of weights, $b$ is the bias term, and $f$ is the activation function.

Figure 4.2: A diagram of a single neuron used in a neural network model. Inputs are weighted and summed, a bias is introduced, and the output is passed through an activation function to limit the output range.

The dot product of the input vector and weight vector in a neuron acts as a linear regressor with an output range between $-\infty$ and $+\infty$. In order to limit the output values to a more manageable range, a squashing function, also called an activation function, is applied to the output. Activation functions are scaling functions that introduce non-linear effects to subsequent summations, and give neural networks their strong ability to work with non-linear data sets. Different activation functions produce different ranges of output. Four commonly used activation functions and
their output ranges are shown below in Table 4.3

<table>
<thead>
<tr>
<th>Function</th>
<th>Form</th>
<th>Range of output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigmoid</td>
<td>$y_a = \frac{1}{1 + \exp(-x_a)}$</td>
<td>$(0,1)$</td>
</tr>
<tr>
<td>Tanh</td>
<td>$y_a = \tanh(x_a)$</td>
<td>$(-1,1)$</td>
</tr>
<tr>
<td>SoftPlus</td>
<td>$y_a = \ln(1 + \exp(x_a))$</td>
<td>$(0, +\infty)$</td>
</tr>
<tr>
<td>ReLU</td>
<td>$y_a = \max(0, x_a)$</td>
<td>$(0, +\infty)$</td>
</tr>
</tbody>
</table>

Figure 4.3 shows how ANNs are typically arranged. An input layer, some number $N$ hidden layers, and an output layer. Each connection from one neuron to the next represents a weight that modifies the input of the neuron before being processed. The weights applied to each neuron and across each layer are adjusted during the training procedure to maximize the output of desired information. The hidden layer has full connectivity to the output layer, where each arrow represents a weight in the network. It is also common for an artificial neural network (ANN) to contain a variety of different layer types. Each layer type dictates the mathematical calculation performed within the neurons of that layer.

**Time-series (LSTM) Neural Networks**  In time-series data handling applications, information is time distributed. If we wish to perform predictive modeling of future events, the use of a recurrent network layer such as long short-term memory (LSTM) is a practical method. LSTM layers are a type of recurrent neural network layer that was originally developed by Hochreiter et al.\[^{[60]}\]. In traditional recurrent neural networks, issues with training became a major problem for practical use. Back-propagation through time methods (BPTT) and real-time recurrent learning (RTRL) methods suffered from issues related to loss of gradients during
Figure 4.3: A simple feed-forward artificial neural network. Contains three distinct layers: (a) the input layer with 5 units, (b) the hidden layer with 3 units, and (c) the output layer with a single unit. This network maps an input of five variables to an output of a single variable.
back-propagation training of network weights\cite{60}. Error signals sent back tended to negatively impact the gradient of the loss function by causing it to either diverge to infinity or collapse to zero. This created fluctuation in learned weights, or loss of learning updates altogether\cite{60}. To remedy this, the gated network cell (LSTM) was developed that selectively allowed information to propagate into or out of the cell, reducing the effect of gradient degradation during updates.

A single LSTM unit makes decisions by considering the current input, previous output and previous memory. The cell generates a new output and alters its internal cell memory during each training sequence\cite{61}. The network shown in Figure 4.4 takes three inputs, namely $x^{(t)}$, the input of the current time step, $h^{(t−1)}$, the output from the previous LSTM unit and $c^{(t−1)}$, the cell state of the previous unit. Each LSTM cell maintains a cell state, which acts as a conveyor of information across the cell. The cell is able to modify the information flow across the cell state by adjusting parameters of the gate operators within the cell. Gates act as a restrictor to updates, and are composed of a sigmoid neural network layer and a pointwise multiplication operations. The sigmoid layer restricts output between 0 and 1, effectively regulating the amount of information received by the cell state at each stage of the cell. Each standard LSTM cell maintains three gates to control the information flow to the cell state. Gate one is a forget gate, which acts on $x^{(t)}$ and $h^{(t−1)}$ and decides what information to discard in the cell state update. The sigmoid layer acts on the input values and squashes them to between 0 and 1. 0 represents complete removal of the data, and 1 represents complete retention of the data. The second gate is the input gate layer, which decides what values are updated in the cell state. A sigmoid layer limits the input, followed by a hyperbolic tangent layer that creates a new candidate vector of values between -1 and 1 that may be added to the cell state. The previous cell state $c^{(t−1)}$ is then modified with the new information and updated to
the current cell state $c^{(t)}$. Finally, we create a vector of output with gate three by passing the current cell state through a hyperbolic tangent function and multiply the sigmoid-adjusted values of the previous hidden state $h^{(t-1)}$ to create the new hidden state $h^{(t)}$. The previous cell state remains one of the most critical inputs of the LSTM cell, as this is the unit that controls back-propagation of information in time from one cell to the next.

**Figure 4.4**: In-depth diagram of an LSTM cell. The cell consists of a cell state, $c$, a hidden state of the cell (also the output), $h$, and 9 operators that control the flow of information within the cell.

**Convolutional Neural Networks.** In other applications such as image processing, convolutional neural network (CNN) layers prove useful for extraction of unique and important features of an image relevant to classification and regression tasks. CNNs have made a mark on the machine learning field surrounding their use as top-tier image classification frameworks in competitions such as the ImageNet large scale visual recognition challenge (ILSVRC)[62]. Unlike typical neural networks that require 1D samples, CNNs require 2D samples (generally images).

Each input image is passed through a series of convolution layers, shown in Fig-
4.5, is defined by,

\[ f[x, y] \ast g[x^f, y^f] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] \cdot g[x-n_1, y-n_2], \quad (4.10) \]

where \( x \) and \( y \) are discrete indices of the initial matrix, \( x^f \) and \( y^f \) are the discrete indices of the kernel matrix, \( f \) is the initial matrix, \( g \) is the kernel matrix, and \( n_i \) is the index of the convolution element.

Figure 4.5 shows the process of convolution, in which a kernel matrix is multiplied along an input matrix in a sliding fashion, and each resulting multiplication summated and stored as an element of the output matrix. The output matrix may be smaller than the input matrix due to edge effects of the matrix multiplication, but may be avoided by the introduction of padding to the outside edges of the input matrix. The resultant filtered inputs are then passed through pooling layers, which slide across each matrix with a filter of size \((p_1, p_2)\) in stride of \( s_1 \) and take the maximum value present in each filter region. This process reduces the size of the matrices by a constant factor of \((p_1, p_2)\) and extracts the most prevalent feature of that region. Each matrix is then flattened into a single dimensional vector and processed by dense neural network layers, as shown in Figure 4.3.

Finally a classification layer, often called a softmax layer, is used to classify the image into a number of classes. Softmax layers take the maximum normalized probability of a sample to create distinct predictions of sample class. The operation is defined by,

\[ P(y = j|x)_{\text{max}} = \frac{\exp(x^T w_j)}{\sum_{k=1}^{K} \exp(x^T w_k)}, \quad (4.11) \]

where \( y \) is the output class, \( j \) is the input sample index, \( x \) is the sample vector, \( w \) is the weight vector, \( K \) is the desired number of output classes.

Most practical CNNs have multiple convolution and pooling steps before the
dense layers, as subsequent convolution and pooling aids in extraction of features from an image. In practice a CNN with multiple blocks of convolution and pooling are able to process more unique features of an image and improve classification accuracy.

Figure 4.5: Example of a convolution across a matrix with a $3 \times 3$ kernel matrix. Each element of the output matrix is the element-wise summation of a $3 \times 3$ region of the original matrix multiplied by the kernel matrix.

Figure 4.6: A simple convolutional neural network composed of a single convolutional layers, a max pooling layer, a fully connected layer, and a softmax classifier layer ($1 \times 2$).

**Bayesian Neural Networks.** A Bayesian neural network (BNN), unlike a traditional neural network (NN), is a network with a prior distribution on its weights\(^\text{63}\). Unlike traditional NNs, BNNs present distinct features that distinguish them from other means of learning models. Prior background knowledge (beliefs) may be de-
rived and utilized as initializations for models parameters\cite{63}. This presents a useful means for fitting models that behave differently under variable initial conditions.

As an example, we may define the probabilistic argument for a normally distributed BNN.

Consider a data set \{(x_n, y_n)\} in which each data point is comprised of variables \(x_n \in \mathbb{R}\) with output \(y_n \in \mathbb{R}\). We define the likelihood for each data point as:

\[
p(y_n | w, x_n, \sigma^2) = \text{N}(y_n | \text{NN}(x_n ; w), \sigma^2),
\]

(4.12)

where NN is the neural network, and \(w\) the associated weights and biases. We then define the prior probabilities of the weights and biases \(w\), and identity matrix \(I\), to be:

\[
p(w) = \text{N}(w | I),
\]

(4.13)

where N is the normal distribution of weights with variance defined by the identity diagonal covariance matrix \(I\). This model assumes that the weights of each neuron are normally distributed, which may not always be the case.

### 4.4.2 Bayesian Classifiers

**Naive Bayes Classifier (NBC).** These have gained wide popularity for their powerful probabilistic modeling and computational efficient fitting procedure. Data are separated into classes, where each class in the model can be represented by a probability distribution describing the samples drawn from that class. Each variable in the model can be represented by a different type of distribution, as each is modeled as an independent distribution. Modeling variables independently produces a diagonal covariance matrix in the model, simplifying fitting and updating procedures. Fitting data to a model requires the application of an expectation maximization (EM)
calculation to determine under which class data belongs. EM is a simple algorithm that calculates the most likely distribution a sample originated from by comparing the probability of finding that sample within each distribution in a model. Prediction is achieved by applying Bayes’ rule to each new sequence of data to determine the most likely class from which it was most likely sampled from.

**General Bayes Classifier (GBC).** The use of univariate distributions to model each variable in a dataset results in an identical model to that of a naive Bayes classifier. A divergence occurs under the conditions introduced by a multivariate probability distribution; the GBC assumes variable dependence, consequently producing a full covariance matrix within the model. As a result, the GBC may capture more complex interactions and covariance between variables of the dataset. In general, we may generate a GBC from a naive Bayes classifier by assuming a multivariate probability distribution in place of the product of univariate distributions. The most fundamental difference is the full covariance matrix in the model, in place of the previous diagonal covariance matrix present with the independent distributions.

### 4.4.3 Decision Trees

Known for their accuracy and robustness to outliers, decision trees have a long history as powerful models in machine learning\(^{[64]}\). In addition to their predictive ability, decision trees maintain variable importance measures that are used in applications requiring interpretability as a primary result of model development\(^{[65]}\). Decision trees split data along variables that decrease the impurity of the resulting data, in effect creating branches that describe particular subsets of data as the tree grows in size. The process by which data is split apart inherently defines the importance of a variable by how much it affects the loss criterion brought by that variable. In more general terms, the better a variable split is able to segment, or purify, the resulting
data, the more important that variable is to the tree. Decision trees are also used as classifiers and regressors, in which the underlying data is sequentially arranged into branches and leaves, defining a pathway by which to split and categorize values found in different variables into their designated class or output.

Tree models partition a feature space $\mathbf{X}$ into $T$ non-overlapping regions. For each region $R_j \in T$, a prediction is made based on a constant model of the form:

$$f(x) = \sum_{j=1}^{T} w_j I(x \in R_j)$$  \hspace{1cm} (4.14)

Where $w_j$ is the weight of the number of samples in region $R_j$, and $I$ is the indicator function, yielding 1 if $x$ is in region $R_j$, and 0 otherwise. The structure of the tree is defined by the placement of the splits (branches), and can be modeled with the use of a gain and loss function. The loss function can take the form:

$$L(f) = \sum_{i=1}^{n} \sum_{j=1}^{T} \sum_{i \in R_j} L(y_i, w_j) = \sum_{j=1}^{T} L_j$$  \hspace{1cm} (4.15)

where $L_j$ represents the loss at some node $j$. If we take $k$ to be a node $k \neq j \in T$, we may define a before and after loss as:

$$L(f_{\text{before}}) = \sum_{j \neq k} L_j + L_k$$  \hspace{1cm} (4.16)

$$L(f_{\text{after}}) = \sum_{j \neq k} L_j + L_{kL} + L_{kR}$$  \hspace{1cm} (4.17)

From these losses, we can define the gain of such a split as:

$$\text{Gain} = L(f_{\text{before}}) - L(f_{\text{after}}) = L_k - (L_{kL} + L_{kR}).$$  \hspace{1cm} (4.18)

Each possible split is considered for each feature, and split that achieves the max-
imal gain is taken. Once the tree structure has been defined, the weights of each region $R_j$ must be defined, commonly as:

$$w_j = \arg\min_w \sum_{i \in I_j} L(y_i, w)$$

(4.19)

Where $w_j$ is the weight of region $R_j$ that minimizes the sum of the loss function over the specified output $y_i$

**Gradient Boosting Machines.** An adaptation to base-level decision trees was proposed by Friedman in 2001\[^{[66]}\]. Named gradient boosting regression (GBR), or gradient boosted regression trees (GBRT), the method constructs a forward stage-wise additive model by integrating a gradient decent optimization in function space, as opposed to parameter space. The model utilizes multiple decision trees to better adapt a highly robust and interpretable procedure for classification and regression.

The initial parameterization of the model begins with the definition of the gradient at a stage $m$:

$$-g_m(x) = \left[ \frac{\partial L(y, f(x))}{\partial f(x)} \right] \text{ where } f(x) = f^{m-1}(x).$$

(4.20)

For each iteration $m$, the model is fit to a regression/classification tree to predict the gradient loss. For tractability, a squared error is used as a loss function for the gradient,

$$\phi_m = \arg\min_{\phi} \sum_{i=1}^{n} \left[ (-g_m(x_i) - \phi(x_i))^2 \right],$$

(4.21)

where $\phi_m$ is the gradient loss at iteration $m$, and $\phi$ is the current gradient estimate for $x_i$.

The parameter $\rho_m$ defines the step taken a stage $m$, and can be derived using a
line search:
\[
\rho_m = \argmin_{\rho} \sum_{i=1}^{n} L(y_i f^{(m-1)}(x_i) + \rho \phi_m(x_i)).
\] (4.22)

Friedman\textsuperscript{[66]} proposed a special enhancement for performing this line search over each region \( R_{i \in T} \). Friedman also proposed a shrinkage parameter \( \eta \in (0,1] \) which can be interpreted as a learning rate parameter. In combination, the function to be optimized at each step is then defined as:
\[
f(x) = f^{(M)}(x) = \sum_{m=0}^{M} f_m(x) + \sum_{m=1}^{M} \eta \rho_m \phi_m(x).
\] (4.23)

We may initialize \( f_0(x) \) by utilizing a single decision tree model,
\[
f_0(x) = \theta = \argmin_{\theta} \sum_{i=1}^{n} L(y_i, \theta).
\] (4.24)

We define the new loss function in terms of the gradient,
\[
L(\phi_m) = \sum_{i=1}^{n} \left[ (-g_m(x_i) - \phi_m(x_i)) \right]^2 \\
= \sum_{i=1}^{n} \left[ g_m^2 + 2g_m(x_i)\phi_m(x_i) + \phi_m^2(x_i) \right] \\
= \sum_{j=1}^{T} \left[ \sum_{i \in R_{mj}} g_m^2(x_i) + 2w_{jm} \sum_{i \in R_{mj}} g_m(x_i) + n_{jm}w_{jm}^2 \right],
\] (4.25)

where \( n_{jm} \) is the number of points in a region \( j \).

If we define \( G_{jm} \) as the sum of the gradients in a region \( j \), then we can simplify the loss function as:
\[
L(\phi_m) = \sum_{j=1}^{T} \left[ 2G_{jm}w_{jm} + n_{jm}w_{jm}^2 \right] + \text{constant.}
\] (4.26)

The weights that minimize the loss function are:
\[
w_{jm} = -\frac{G_{jm}}{n_{jm}} \quad \text{for} \quad j \in 1, ..., T_m.
\] (4.27)
Finally, we can write the gain of the split as:

$$ \text{Gain} = \frac{G_{jmL}^2}{n_{jmL}} + \frac{G_{jmR}^2}{n_{jmR}} - \frac{G_{jm}^2}{n_{jm}}. \quad (4.28) $$

**Extreme Gradient Boosting.** An modern adaptation of gradient boosting called extreme gradient boosting (XGBoost) has become popular since its introduction by Chen et. al in 2014\cite{67}. XGBoost derives from the original GBR method, its main difference being the formulation of the gradient and subsequent loss. Unlike GBR, XGBoost attempts to solve for the gradient step directly for each $x$ in the data set:

$$ \frac{\partial L(y, f^{(m-1)}(x) + f_m(x))}{\partial f_m(x)} = 0. \quad (4.29) $$

In order to achieve this numerically, a second-order Taylor-series expansion around the gradient estimate $f^{(m-1)}(x)$ is performed:

$$ L(y, f^{(m-1)}(x) + f_m(x)) \approx L(y, f^{(m-1)}(x)) + g_m(x)f_m(x) + \frac{1}{2}h_m(x)f_m(x), \quad (4.30) $$

where $g_m(x)$ is the gradient, and $h_m(x)$ is the Hessian at the current estimate $m$.

The Hessian is defined as:

$$ h_m(x) = \frac{\partial^2 L(y, f^{(m-1)}(x))}{\partial f^{(m-1)}(x)^2}. \quad (4.31) $$

Combining these expansions, we rewrite the loss function as:

$$ L(f_m) \approx \sum_{i=1}^{n} \left[ g_m(x_i)f_m(x_i) + \frac{1}{2}h_m(x_i)f_m(x_i)^2 \right] + \text{constant} $$

$$ \propto \sum_{j=1}^{T_m} \left[ G_{jm}w_{jm} + \frac{1}{2}H_{jm}w_{jm}^2 \right], \quad (4.32) $$

where, similar to the GBR derivation, $G_{jm}$ and $H_{jm}$ are the sums of the gradient and Hessian in region $j$, respectively. We may rewrite the loss function and weights,
given a fixed model structure as:

\[
L(\phi_m) = \sum_{j=1}^{T_m} \left[ 2G_{jm}w_{jm} + \frac{1}{2}H_{jm}w_{jm}^2 \right],
\]  

(4.33)

with

\[
w_{jm} = -\frac{G_{jm}}{H_{jm}} \text{ for } j \in 1, \ldots, T_m.
\]

(4.34)

The loss function is of the general form:

\[
L(f_m) \propto -\frac{1}{2} \sum_{j=1}^{T_m} \frac{G_{jm}^2}{H_{jm}}.
\]

(4.35)

According to Chen et. al\textsuperscript{[67]}, this loss function is the structure score of the defined tree, in which a smaller score indicates a better structure. Therefore for each split, the gain can be written as:

\[
\text{Gain} = \frac{1}{2} \left[ \frac{G_{jmL}^2}{H_{jmL}} + \frac{G_{jmR}^2}{H_{jmR}} - \frac{G_{jm}^2}{H_{jm}} \right],
\]

(4.36)

where \(L\) and \(R\) denote the left and right node of the split, respectively.

### 4.5 Unsupervised Learning

**Markov Chains.** These are stochastic processes that satisfy the Markov property of \textit{memoryless-ness}, in which inference about the future of the process is equally valid whether the process history is available to the inference or not. The future and past states are statistically independent of the current state of the process. Markov chains prove useful when modeling sequences of events in a highly independent state space in which the emissions of each state are bounded by a finite number of possible outcomes.
Markov Models. A Markov model is a sub-type of a Markov chain, and is the functional discrete-time model that a Markov chain produces when observed in discrete time\textsuperscript{68}. Figure 4.7 shows a simple Markov model in discrete time. Effectively, a Markov model is defined by a set of $x_i \in \mathbf{x}$ states each having a probability of emitting a particular observable, $y_k \in \mathbf{y}$, as well as the probability of transitioning to a new state, $x_j \in \mathbf{x}$. Markov models are defined by an explicit set of prior emission probabilities describing the likelihood of observation of an output based on the state of the model, written as:

$$\{P(y_k|x_i)^{init} \geq 0 \| x_i \in \mathbf{x}\},$$

(4.37)

where $P$ is the probability greater than or equal to 0 of observing $y_k$ while in state $x_i$.

If each distribution is known before constructing the model, the inferential behavior of the model can be excellent. Generally, Markov models are defined within a known process space, that is, they require prior knowledge of the process that is being modeled to be fully defined. Markov models are particularly useful when the process being modeled is ongoing over time, or may recur back to a previous state of activity as time goes on. For example, processes like surgical procedures may incur returns to previous risk states. For example, during operation, a patient may be well (state), and suddenly experience loss of patient heartbeat (observable), indicating the patient is unwell (state). Soon after, the patient may experience return of patient heartbeat (observable), become well again (state). Each event is independent of previous events, and has potential to occur more than once, both satisfying the Markov property and occurring in a time-ongoing fashion.

Hidden Markov Models. (HMM) If the state space of the data is unknown, it is still possible to define a Markov model that describes the operation, with some re-
Figure 4.7: A Markov model in discrete time. The model consists of three states, $x_i \in \mathbf{x}$, with probability of transition $a_{ij}$ to state $x_j$. Each state $x_i$ has likelihood $b_{ik}$ of emitting some $y_k \in \mathbf{y}$. The chain is fully defined, as each state will emit a particular set of emissions specific to that state with a known probability. Note that some states cannot emit particular emissions $y_k$.

This is a specific example of a Markov model in which each state is assumed to have an initial probability greater than zero of emitting a given $[y_k \in \mathbf{y}]$ for each state $[x_i \in \mathbf{x}]$. We call this the initial prior distribution of the model. This can be described mathematically as:

$$\{ P(y_k | x_i)^{init} > 0 \ | \ x_i \in \mathbf{x} \}$$

(4.38)

where $P$ is the probability greater than 0 of observing $y_k$ while in state $x_i$.

This initial condition is defined because the emission distribution of each state is initially unknown, and as such cannot be assumed to be zero without evidence.

In the general Markov model, initial priors can have zero probabilities, as denoted
by the greater than or equal operator in the set. This stems from partial knowledge of the states and their emission distributions when defining the non-hidden Markov model.

HMMs may be generalized as a ‘fuzzy’ versions of a Markov model. Their unspecified state space provides a framework for machine learning across a mutable dataset; the lack of full definition allows the model to generalize to many different types of data, providing an empirical set of references for important modes in a dataset without prior knowledge of the actual (hidden) network structure. This ability to classify arbitrary data is what make HMMs a useful tool for the optimization of clinical processes, including resource allocation, patient flow, and random event scenarios.

![Hidden Markov Model Diagram](image)

Figure 4.8: A hidden Markov model in discrete time. The model consists of three states, $x_i \in \mathbf{x}$, with probability of transition $a_{ij}$ to state $x_j$. Each state $x_i$ has likelihood $b_{ik}$ of emitting some $y_k \in \mathbf{y}$. The model is called hidden because we do not know the exact probabilities of transition or emission. Note that all states can emit any $y_k \in \mathbf{y}$. 
Chapter 5

Preprocessing and Data Preparation Techniques

5.1 Imputation and data removal.

Data sets containing missing values require preprocessing, as most machine learning methods do not support missing values in model training or prediction[69]. A large majority of supervised learning methods require missing data to be removed or completed by some external method, often called imputation. We will refer to each sample as a row, and each variable in the sample as a column.

This work focuses on five major methods for handling missing data:

1. Complete removal of missing values;

2. Removal of most incomplete columns followed by data removal;

3. Complete imputation of the missing values;

4. Removal of most incomplete columns followed by data imputation; and

5. Replacement of missing data with fixed negative integer.
Removal of missing values. Data sets often contain rows that contain missing values. In an effort to generate a model, it may be necessary to remove the row to ensure the dataset is dense and will not introduce biases. The removal of rows containing one or more missing values can be unsuitable if a large fraction of the dataset is lost, potentially removing useful and informative features of the data. Therefore, care must be taken when removing data. An alternative to direct row removal is to identify and remove columns in the dataset that present a high proportion of missing values. The removal of these columns can then reduce the number of rows that contain missing values, and thus reduce the number of rows that must be removed from the dataset.

Imputation of the missing values. An alternative to removal of data is to artificially complete (impute) missing data based on the available data in the set. Data completion, also called imputation, is performed by filling in missing values based on the mean, median, or mode, of the column that contains the missing information. The use of simple column statistics has some notable drawbacks, including introduction of bias to the dataset, lack of evidence for the imputed data, and an assumption that the data is normally distributed. In cases where only a small fraction of data is missing, data imputation has less of an effect on the integrity of the data. In order to reduce the number of values that must be artificially completed, columns containing a large fraction of missing values may be removed from the dataset before imputation. This technique is more suitable when a small subset of columns have many missing values, and the remaining columns do not. Imputation is generally performed using the mean, median, or mode of the column that the missing value originated from. The use of simple column statistics have some notable drawbacks, including introduction of bias to the dataset, lack of evidence for the imputed data, and an assumption that the data is normally distributed. Other methods, such as
the use of Bayesian networks, may be used for dataset imputation, but require a processing time that is proportional to the exponential of the number of samples being fit. In addition, Bayesian network methods often require discrete valued data, and thus data sets containing a high number of continuous-valued columns are generally unsuited for these methods.

**Replacement of missing data with fixed negative integer.** In some cases, the removal or imputation of missing data is unfavorable due to the introduction of bias or because the missing data may have importance in the dataset. To address this, missing values can be replaced with a value that does not mimic other data in the model, but instead creates a consistent representation of a missing value. For example, replacing missing values with a fixed, high magnitude, negative integer across the dataset will enable a model to train on the data, but will identify missing values as belonging to the surrogate value and handle the input accordingly. This method works best when a majority of columns contain some missing values, or when future testing data contains missing values in the same columns as the training data. If a model is not trained to identify missing values in a column of input data, deployment of the model may fail if presented with missing values in the future.

### 5.2 Scaling

Many machine learning methods are sensitive to the magnitude of data. Neural networks in particular perform best when data is close to the unit normal distribution, that is, centered with a mean of 0 and a standard deviation of 1. In order to maximize the effectiveness of several ML methods, scaling of the data is often performed\[69\].

MinMax scaling is one of the simplest approaches to scale a dataset. Each column of the dataset is scaled independently to a target range, (low, high), by the following
transformation:

\[ S_{factor} = \frac{(X_i - \text{min}(X))}{\text{max}(X) - \text{min}(X)} \]  

(5.1)

\[ X_{scaled} = S_{factor} \cdot (\text{high} - \text{low}) + \text{low} \]  

(5.2)

where \( X \) is vector of values in the column to be scaled, \( S_{factor} \) is the computed scaling factor for the column, \( \text{high} \) and \( \text{low} \) are the desired scaling range as defined by the user, and \( X_{scaled} \) is the scaled output. MinMax scaling is reversible, so long as the original minimum and maximum values of the dataset columns, as well the target range, are maintained.

Another approach is to use standard scaling on a column by removing the mean and scaling the values to a unit variance. Centering and scaling occur independently on each column by computing the relevant mean and variance of the samples. This method of scaling is useful for machine learning methods that rely on data to be centered around 0 and maintain variance in the same order across all columns, such as neural networks and support vector machines. Each column of the dataset is scaled independently through the following transformations:

\[ X_{shift} = X - \mu_X \]  

(5.3)

\[ X_{scaled} = \frac{X_{shift}}{\sigma_X} \]  

(5.4)

where \( X \) is vector of values in the column to be scaled, \( X_{shift} \) is the shift required to bring the data to a zero mean, \( \sigma_X \) is the standard deviation of the column, and \( X_{scaled} \) is the scaled output. Standard scaling works well when the column being scaled is approximately normally distributed, but may still be applicable if the data is not extremely scattered.
5.3 Encoding and Embedding

Encoding techniques are useful for machine learning methods that handle input values in a strictly numeric fashion, rather than categorical. For example, if a column contains categorical values ranging from 1 to 5, a strictly numeric algorithm will treat each category as though it were continuously valued, such that the average of categories 1 and 5 is equal to category 3. Because categorical variables do not relate in this way, they must be encoded to conform with the method they are being applied to.

OneHot encoding is a popular method of encoding columns of categorical variables that are dense, that is they contain relatively few total categories. OneHot encodes categorical integer features using a one-of-K scheme, such that each category is represented by a column vector describing the sample as on or off. In order to eliminate numerical bias in the target variable, the output is converted into a binary matrix. Each class of the target is encoded into an independent column of the OneHot matrix, for example:

\[
\begin{bmatrix}
0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 \\
2 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 \\
\vdots & \vdots & \vdots & \vdots 
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
\vdots & \vdots & \vdots 
\end{bmatrix}.
\]

Another popular method for the conversion of categorical data into a continuous valued form is direct embedding. This works well with categorical variables that are sparse, that is they contain a large number of unique categories. Much like OneHot encoding, embedding converts categorical data into a fixed-length vector
that is representative of the original data, but is not limited to binary values. Often a single column will be embedded into a replacement vector of the same length as the column, but with a vector of values representing each category. For example:

\[
\begin{bmatrix}
\text{big\_dog} \\
\text{small\_dog} \\
\text{bird} \\
\text{big\_dog} \\
\text{small\_dog} \\
\vdots
\end{bmatrix}
\rightarrow
\begin{bmatrix}
0.5 & -0.8 & 0.9 & 0.1 \\
0.4 & -0.3 & 1.0 & 0.2 \\
-0.7 & 0.1 & 0.8 & -0.4 \\
0.5 & -0.8 & 0.9 & 0.1 \\
0.4 & -0.3 & 1.0 & 0.2 \\
\vdots & \vdots & \vdots & \vdots
\end{bmatrix}
\]

(5.6)

Column embedding is performed as an in-line process within a model, for example, individual columns may be embedded directly within a neural network model and trained along with other layers. In many cases, the resulting vector embeddings maintain similarities across each feature, as it relates to the dataset. In Equation 5.6, we see similar values representing the \textit{big\_dog} and \textit{small\_dog} categories, and dissimilar values representing the \textit{bird} category. These similarities are directly inferred from data, and can often be treated as representations of the category in vector form, allowing for vector operations to be performed on the embedding, much like in Word2Vec style encodings\textsuperscript{[70]}.

### 5.4 Special Transformations

Certain machine learning techniques require data to be specially prepared prior to use in a model. If a dataset input dimension is not directly compatible with a particular method, it must be transformed to meet the requirements of the model. An example of this is the use of time series data with an image processing model such as convolutional neural networks, in which the input data is single dimensional, and
the required model input is two dimensional.

The application of Markov transition fields to time series data is a means for converting the one dimensional data into a two dimensional image representation, based on the transition of the time series across different states within the sample. In order to adapt to the required input dimension, time series data is transformed by use of Markov transition fields (MTF), as described in Section 4.3.4.
Chapter 6

Dataset Description and Analysis

6.1 Pima Indians Dataset

The Pima indians dataset is a compilation of information collected by medical professionals studying the onset of diabetes in women 21 to 81 years of age who belonged to the Pima indians tribe. The dataset was originally owned by The National Institute of Diabetes and Digestive and Kidney Diseases of the NIH\cite{71}. The dataset records include information taken from 768 patients, each with 9 numeric attributes. Out of the nine condition variables, six attributes describe the result of physical examination, the remaining two variables are the result of chemical examinations. The target objective is the binary variable ‘Outcome’, in which 1 indicated a positive result for diabetes (diabetes = yes), and 0 indicated a negative result (diabetes = no).

The dataset has been widely studied in machine learning fields as a baseline for model development and preprocessing techniques\cite{72}. The dataset contains a relatively low number of variables and samples, and thus is tractable for a variety of machine learning methods that have generally high convergence orders, such as support vector machines (SVM), and non-linear kernel approximation methods. We use Bayesian methods and artificial neural networks as an alternative to these methods,
Table 6.1: Pima indians dataset column names and % missing values.

<table>
<thead>
<tr>
<th>Column</th>
<th>% Missing</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pregnancies</td>
<td>14.45%</td>
<td>Number of times pregnant</td>
</tr>
<tr>
<td>Glucose</td>
<td>0.65%</td>
<td>Plasma glucose concentration at 2 hours in an oral glucose tolerance test</td>
</tr>
<tr>
<td>Blood Pressure</td>
<td>4.56%</td>
<td>Diastolic blood pressure (mm Hg)</td>
</tr>
<tr>
<td>Skin Thickness</td>
<td>29.58%</td>
<td>Triceps skin fold thickness (mm)</td>
</tr>
<tr>
<td>Insulin</td>
<td>48.7%</td>
<td>2-Hour serum insulin (mu U/ml)</td>
</tr>
<tr>
<td>BMI</td>
<td>1.43%</td>
<td>Body mass index ((kg/(m)^2)))</td>
</tr>
<tr>
<td>Diabetes Ped. Function</td>
<td>0.0%</td>
<td>Diabetes pedigree function, family history</td>
</tr>
<tr>
<td>Age</td>
<td>0.0%</td>
<td>Age of the patient</td>
</tr>
<tr>
<td>Outcome</td>
<td>0.0%</td>
<td>Target variable: value of 0 or 1, indicating whether the patient has diabetes</td>
</tr>
</tbody>
</table>

in the effort to develop a generalizable method for the rapid analysis of this kind of dataset and compare the results to other published works.

The variables in the PIDD are represented relatively well by common probability distributions, in particular the Gaussian distribution (normal distribution) and exponential distribution. Figure 6.1 shows the shapes of each column in the PIDD; by direct observation of the histograms, we define normally distributed features. For example, BMI, blood pressure, glucose, and skin thickness. Similarly, we find approximately exponentially distributed features, such as age, diabetes pedigree function, insulin, and pregnancies. The use of appropriate distributions for modeling generally improves prediction accuracy, and is particularly important in probability-based models such as naive Bayes classifiers and other Bayesian methods.
Figure 6.1: Histograms of each column of the Pima indians dataset.

6.2 Alcoholism EEG Dataset

The use of electroencephalogram (EEG) technology is very actively used by researchers to study brain waves in a variety of animal subjects and humans. Samples of anonymized EEG data was collected during research on the genetic predisposition to alcoholism in human subjects. It contains measurements from 64 electrodes placed on subject’s scalps which were sampled at 256 Hz for 1 second during the experimental proceedings. The experiment maintained two groups of subjects, alcoholic and control. Each subject was exposed to a single stimulus or two stimuli originating from the 1980 Snodgrass and Vanderwart picture set\cite{374}. In the case of two stimuli, subjects were presented with either a matched condition (each stimulus was the same), or a non-matched condition (different stimuli). Control group and alcoholic subject brain waves are shown in Figure 6.2. We note the distinct activation of the alcoholic subject during the recording, in which the voltage of select channels reach a very high level compared to the control subject.

This dataset was provided by the UCI Knowledge Discovery in Databases Archive (KDD)\cite{71}.
83

Figure 6.2: EEG samples of a control subject (left) and an alcoholic subject (right). The response of the alcoholic shows distinct regions of activation that are not present in the control subject.

6.3 Diabetes Readmission Dataset

This dataset is a diabetes centered collection of publicly available data, compiled by Strack et al.\cite{74}. The data extracted is designed to meet five qualifying criteria: (1) it is an inpatient encounter, (2) it is a diabetic encounter, (3) the patient length of stay was between 1 and 14 days, (4) laboratory tests were performed during the encounter, and (5) medications are administered during the encounter. It should be noted that not all encounters in the dataset are directly related to a diabetic encounter, but instead that the patient had diabetes coded as a prior condition.

The dataset contains 50 features related to diabetic patients, with a target feature of readmission. Features of the dataset include relevant metrics for prediction of patient readmission statistics. Patient readmission is a major component of patient health and operational efficiency, as an ideal operation would minimize readmission wherever possible. Predicting patient readmission is then a critical effort for the improvement of clinical process efficiency.

Table 6.2 shows the variables included in the diabetes readmission dataset, as well as the variable type, description of the variable, and the percentage of missing values present in each variable. The dataset is composed primarily of categorical and
continuous variables, with some variables directly referencing another categorical feature present in supplementary dataset, such as the admission type (Table 6.3), admission source ID (Table 6.4), discharge disposition (Table 6.5), and payer code.

Histogram analysis of each variable in the dataset is performed to identify relevant distributions of the data as well as potential outliers in continuous or categorical features of the dataset. Initial analysis shows that many continuous variables are normally distributed, for example number of lab procedures, number of medications, and patient length of stay, whereas others are exponentially distributed, for example number of procedures, number of emergency visits, number of outpatient visits, and number of inpatient visits. Categorical variable distributions are not analyzed for distribution shape, but instead for regions of unusual occurrence, such as the low number of samples with admission type ID of 4 and 7, corresponding to newborn and trauma center, respectively, and the low number of samples with discharge disposition ID of 10, 12, 16, 17, 20, 21, and 26, corresponding to neonatal discharge, current patient, discharged to different outpatient facility, discharged for outpatient care at current facility, expired in medical facility, expired in unknown place, and unknown, respectively. These regions of interest are useful for the identification of unlikely outcomes for a patient in this dataset, in particular those marked as expiring in the care facility.
Table 6.2: Diabetes readmission dataset variable types and descriptions.

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Type</th>
<th>Description of variable</th>
<th>% missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encounter ID</td>
<td>INT</td>
<td>Unique encounter identifier</td>
<td>0%</td>
</tr>
<tr>
<td>Patient Number</td>
<td>INT</td>
<td>Unique patient identifier</td>
<td>0%</td>
</tr>
<tr>
<td>Race</td>
<td>CAT</td>
<td>Values: Caucasian, Asian, African American, Hispanic, Other</td>
<td>2%</td>
</tr>
<tr>
<td>Gender</td>
<td>CAT</td>
<td>Values: Male, Female, unknown</td>
<td>0%</td>
</tr>
<tr>
<td>Age</td>
<td>CAT</td>
<td>Groups: [0,10), [10,20), ..., [90,100)</td>
<td>0%</td>
</tr>
<tr>
<td>Weight</td>
<td>NUM</td>
<td>Weight in pounds</td>
<td>97%</td>
</tr>
<tr>
<td>Admission type</td>
<td>MAP</td>
<td>9 distinct values, corresponding to emergency, urgent, etc.</td>
<td>0%</td>
</tr>
<tr>
<td>Discharge disposition</td>
<td>MAP</td>
<td>29 distinct values, corresponding to discharged to home, expired, etc.</td>
<td>0%</td>
</tr>
<tr>
<td>Admission source</td>
<td>MAP</td>
<td>21 distinct values, corresponding to physician referral, emergency room, transfer, etc.</td>
<td>0%</td>
</tr>
<tr>
<td>Time in hospital</td>
<td>NUM</td>
<td>Number of days in hospital (range: 1 to 14 days)</td>
<td>0%</td>
</tr>
<tr>
<td>Payer code</td>
<td>MAP</td>
<td>23 distinct values, corresponding to Medicare, self-pay, etc.</td>
<td>52%</td>
</tr>
<tr>
<td>Medical Specialty</td>
<td>CAT</td>
<td>Identifier of care type, for example, cardiology, family practice, surgical, etc.</td>
<td>53%</td>
</tr>
<tr>
<td>Num. of lab procedures</td>
<td>NUM</td>
<td>Number of lab tests performed during patient encounter</td>
<td>0%</td>
</tr>
<tr>
<td>Num. of procedures</td>
<td>NUM</td>
<td>Number of procedures performed during the encounter</td>
<td>0%</td>
</tr>
<tr>
<td>Num. of medications</td>
<td>NUM</td>
<td>Number of medications administered during the encounter</td>
<td>0%</td>
</tr>
<tr>
<td>Num. outpatient visits</td>
<td>NUM</td>
<td>Number of outpatient visits within the past year</td>
<td>0%</td>
</tr>
<tr>
<td>Num. emergency room visits</td>
<td>NUM</td>
<td>Number of emergency visits within the past year</td>
<td>0%</td>
</tr>
<tr>
<td>Num. inpatient visits</td>
<td>NUM</td>
<td>Number of inpatient visits within the past year</td>
<td>0%</td>
</tr>
<tr>
<td>Diagnosis 1</td>
<td>CAT</td>
<td>Primary diagnosis, coded in ICD9</td>
<td>0%</td>
</tr>
<tr>
<td>Diagnosis 2</td>
<td>CAT</td>
<td>Secondary diagnosis, coded in ICD9</td>
<td>0%</td>
</tr>
<tr>
<td>Diagnosis 3</td>
<td>CAT</td>
<td>Additional secondary diagnosis, coded in ICD9</td>
<td>1%</td>
</tr>
<tr>
<td>Num. of diagnoses</td>
<td>NUM</td>
<td>Number of diagnoses on patient record</td>
<td>0%</td>
</tr>
<tr>
<td>Glucose serum test results</td>
<td>CAT</td>
<td>Range of results. Values: &quot;&gt;200&quot;, &quot;&gt;300&quot;, &quot;normal&quot;, &quot;none&quot;</td>
<td>0%</td>
</tr>
<tr>
<td>A1c test results</td>
<td>CAT</td>
<td>Range of results. Values: &quot;&gt;8%&quot;, &quot;&gt;7%&quot;, &quot;normal&quot;, &quot;none&quot;</td>
<td>0%</td>
</tr>
<tr>
<td>Change of medication</td>
<td>CAT</td>
<td>Whether patient medication regimen was adjusted. Values: &quot;Yes&quot;, &quot;No&quot;</td>
<td>0%</td>
</tr>
<tr>
<td>Diabetes medication</td>
<td>CAT</td>
<td>Whether diabetic medication was prescribed. Values: &quot;Yes&quot;, &quot;No&quot;</td>
<td>0%</td>
</tr>
<tr>
<td>Medication features (x24)</td>
<td>CAT</td>
<td>Prescribed dosages of diabetes medications. Values: &quot;Up&quot;, &quot;Down&quot;, &quot;Steady&quot;, &quot;No&quot;</td>
<td>0%</td>
</tr>
<tr>
<td>Readmitted</td>
<td>CAT</td>
<td>Number of days for readmission. Values: &quot;&gt;30&quot;, &quot;&gt;30&quot;, &quot;No&quot;</td>
<td>0%</td>
</tr>
</tbody>
</table>

Descriptions: INT = Integer, CAT = Categorical, NUM = Numeric, MAP = Mapped integer
Table 6.3: Mapping of Admission type ID

<table>
<thead>
<tr>
<th>admission_type_id</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Emergency</td>
</tr>
<tr>
<td>2</td>
<td>Urgent</td>
</tr>
<tr>
<td>3</td>
<td>Elective</td>
</tr>
<tr>
<td>4</td>
<td>Newborn</td>
</tr>
<tr>
<td>5</td>
<td>Not Available</td>
</tr>
<tr>
<td>6</td>
<td>NULL</td>
</tr>
<tr>
<td>7</td>
<td>Trauma Center</td>
</tr>
<tr>
<td>8</td>
<td>Not Mapped</td>
</tr>
</tbody>
</table>

Table 6.4: Mapping of Admission source ID

<table>
<thead>
<tr>
<th>admission_source_id</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Physician Referral</td>
</tr>
<tr>
<td>2</td>
<td>Clinic Referral</td>
</tr>
<tr>
<td>3</td>
<td>HMO Referral</td>
</tr>
<tr>
<td>4</td>
<td>Transfer from a hospital</td>
</tr>
<tr>
<td>5</td>
<td>Transfer from a Skilled Nursing Facility (SNF)</td>
</tr>
<tr>
<td>6</td>
<td>Transfer from another health care facility</td>
</tr>
<tr>
<td>7</td>
<td>Emergency Room</td>
</tr>
<tr>
<td>8</td>
<td>Court/Law Enforcement</td>
</tr>
<tr>
<td>9</td>
<td>Not Available</td>
</tr>
<tr>
<td>10</td>
<td>Transfer from critical access hospital</td>
</tr>
<tr>
<td>11</td>
<td>Normal Delivery</td>
</tr>
<tr>
<td>12</td>
<td>Premature Delivery</td>
</tr>
<tr>
<td>13</td>
<td>Sick Baby</td>
</tr>
<tr>
<td>14</td>
<td>Extramural Birth</td>
</tr>
<tr>
<td>15</td>
<td>Not Available</td>
</tr>
<tr>
<td>17</td>
<td>NULL</td>
</tr>
<tr>
<td>18</td>
<td>Transfer From Another Home Health Agency</td>
</tr>
<tr>
<td>19</td>
<td>Readmission to Same Home Health Agency</td>
</tr>
<tr>
<td>20</td>
<td>Not Mapped</td>
</tr>
<tr>
<td>21</td>
<td>Unknown/Invalid</td>
</tr>
<tr>
<td>22</td>
<td>Transfer from hospital in patient/same facility</td>
</tr>
<tr>
<td>23</td>
<td>Born inside this hospital</td>
</tr>
<tr>
<td>24</td>
<td>Born outside this hospital</td>
</tr>
<tr>
<td>25</td>
<td>Transfer from Ambulatory Surgery Center</td>
</tr>
<tr>
<td>26</td>
<td>Transfer from Hospice</td>
</tr>
<tr>
<td>discharge_disposition_id</td>
<td>description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Discharged to home</td>
</tr>
<tr>
<td>2</td>
<td>Discharged/transferred to another short term hospital</td>
</tr>
<tr>
<td>3</td>
<td>Discharged/transferred to skilled nursing facility</td>
</tr>
<tr>
<td>4</td>
<td>Discharged/transferred to intensive care facility</td>
</tr>
<tr>
<td>5</td>
<td>Discharged/transferred to another type of inpatient care institution</td>
</tr>
<tr>
<td>6</td>
<td>Discharged/transferred to home with home health service</td>
</tr>
<tr>
<td>7</td>
<td>Left against medical advice</td>
</tr>
<tr>
<td>8</td>
<td>Discharged/transferred to home under care of Home IV provider</td>
</tr>
<tr>
<td>9</td>
<td>Admitted as an inpatient to this hospital</td>
</tr>
<tr>
<td>10</td>
<td>Neonate discharged to another hospital for neonatal aftercare</td>
</tr>
<tr>
<td>11</td>
<td>Expired</td>
</tr>
<tr>
<td>12</td>
<td>Still patient or expected to return for outpatient services</td>
</tr>
<tr>
<td>13</td>
<td>Hospice / home</td>
</tr>
<tr>
<td>14</td>
<td>Hospice / medical facility</td>
</tr>
<tr>
<td>15</td>
<td>Discharged/transferred within this institution to Medicare approved swing bed</td>
</tr>
<tr>
<td>16</td>
<td>Discharged/transferred/referred another institution for outpatient services</td>
</tr>
<tr>
<td>17</td>
<td>Discharged/transferred/referred to this institution for outpatient services</td>
</tr>
<tr>
<td>18</td>
<td>NULL</td>
</tr>
<tr>
<td>19</td>
<td>Expired at home. Medicaid only, hospice.</td>
</tr>
<tr>
<td>20</td>
<td>Expired in a medical facility. Medicaid only, hospice.</td>
</tr>
<tr>
<td>21</td>
<td>Expired, place unknown. Medicaid only, hospice.</td>
</tr>
<tr>
<td>22</td>
<td>Discharged/transferred to another rehab fac including rehab units of a hospital</td>
</tr>
<tr>
<td>23</td>
<td>Discharged/transferred to a long term care hospital.</td>
</tr>
<tr>
<td>24</td>
<td>Discharged/transferred to a nursing facility certified under Medicaid but not</td>
</tr>
<tr>
<td></td>
<td>certified under Medicare.</td>
</tr>
<tr>
<td>25</td>
<td>Not Mapped</td>
</tr>
<tr>
<td>26</td>
<td>Unknown/Invalid</td>
</tr>
<tr>
<td>27</td>
<td>Discharged/transferred to a federal health care facility.</td>
</tr>
<tr>
<td>28</td>
<td>Discharged/transferred/referred to a psychiatric hospital or psychiatric unit</td>
</tr>
<tr>
<td>29</td>
<td>Discharged/transferred to a Critical Access Hospital (CAH).</td>
</tr>
</tbody>
</table>
Figure 6.3: Admission Source ID. The admission source ID defines the means by which the patient arrived to the clinical encounter. Generally patients arrived through sources 1, 7, and 17, which correspond to physician referral, emergency room, and NULL (information not available). In general, diabetic encounters occur on scheduled time periods or in emergency situations. The shape of the distribution is of less importance to this work as it is a mapped categorical record, and not a numerical observation.
Figure 6.4: *Admission Type ID*. The admission type ID defines the department by which the patient was admitted. A majority of admissions occurred through the emergency, urgent care, elective, and unknown routes, aligning common admission routes for diabetic patients. Like the admission source, the admission type is a mapped categorical record, thus the distribution shape is not numerically significant.

Figure 6.5: *Discharge disposition ID*. Discharge disposition ID refers to the method of discharge and outcomes after leaving the hospital.
Figure 6.6: *Number of Lab Procedures.* The distribution of the number of lab procedures is approximately Gaussian distributed with a mean of 43 and a standard deviation of 23. The plotted values are shown on a log-scale.

Figure 6.7: *Number of procedures.* The number of procedures, including surgery, the patient underwent during their current visit.
Figure 6.8: *Number of medications.* The number of medications the patient is currently taking in regard to the treatment of diabetes.

Figure 6.9: We can see that the number of emergency visits is exponentially distributed, with most patients having encountered 5 or less emergency visits in the past year.
Figure 6.10: The number of all-time outpatient visits related to diabetes management and treatment.

Figure 6.11: The number of all-time inpatient visits related to diabetes management and treatment.
Figure 6.12: Patient length of stay (LOS) is described by the number of days spent in the hospital during the current visit.
Chapter 7

Dataset Analysis Results

7.1 Pima Indians Diabetes Dataset

7.1.1 Neural Network Model

The Pima Indians diabetes dataset (PIDD) contains numerous zero/missing values in various columns, indicating a lack of information available from particular patients. In order to examine the effect of preprocessing on the PIDD, we use three methods for preprocessing the data: (1) the complete, unmodified dataset in which missing values are denoted as 0, (2) the dataset after applying the complete removal of missing values preprocessing technique, and (3) the dataset after applying the imputation with a fixed negative integer technique, described in Section 5.1, respectively. Table 7.1 shows the size of the resultant dataset under each possible preprocessing regime, where rows in bold indicate the preprocessing methods used during analysis.

Because the PIDD is processed via a neural network, dataset scaling is required to ensure proper learning updates and model fitting. The input variables of the PIDD were scaled using MinMax scaling as described in Section 5.2. The categorical target
Table 7.1: Resultant dataset dimensions under different pre-processing regimes. Dimensions shown as (rows, columns)

<table>
<thead>
<tr>
<th>Method</th>
<th>Resulting dataset dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unmodified dataset</td>
<td>(768, 8)</td>
</tr>
<tr>
<td>Removal of all missing values</td>
<td>(336, 8)</td>
</tr>
<tr>
<td>Removal of worst column + removal of remaining missing values</td>
<td>(455, 7)</td>
</tr>
<tr>
<td>Removal of worst column + imputation of remaining missing values with median</td>
<td>(768, 7)</td>
</tr>
<tr>
<td>Removal of worst column + imputation of remaining missing values with mean</td>
<td>(768, 7)</td>
</tr>
<tr>
<td>Removal of worst column + imputation of remaining missing values with negative integer</td>
<td>(768, 7)</td>
</tr>
<tr>
<td>Full imputation of dataset with median value</td>
<td>(768, 8)</td>
</tr>
<tr>
<td>Full imputation of dataset with mean value</td>
<td>(768, 8)</td>
</tr>
<tr>
<td><strong>Full imputation of dataset with negative integer value</strong></td>
<td><strong>(768, 8)</strong></td>
</tr>
</tbody>
</table>
variable *outcome* was encoded using the OneHot method (Section 5.3). The MinMax scaling approach was chosen due to the reversibility of the scaling properties. Each column is independently scaled to values between [0,1] on preprocessing regimes 1 and 2, and regime 3 was scaled and the indices of missing values replaced with the value -1. The model was fit to each dataset and the model accuracy compared.

The proposed model architecture is shown in Figure 7.1. Dense layers one and three contain 64 and 32 neurons, respectively. Layers used in this model utilize exponential linear unit (ELU) activation functions, which are similar to ReLU (Section 4.4.1) activation functions at the tail ends. ELU activation has been shown to improve classification accuracy, and improve model training times by allowing the weights and activations to remain near a unit normal mean of zero. Maintaining the activation mean near zero speed up learning by ensuring a gradient closer to the unit natural gradient\[75\]. Layers two and four are intermediate dropout layers, with 25% and 50% dropout rate, respectively. The model is closed with a softmax classification layer that outputs the most likely target class based on the input data. The model is trained to classify individuals into either 0 (no diabetes) or 1 (diabetes) based on predictive variables in the dataset.

The PIDD contains 8 predictive variables and the single binary objective variable *outcome*, lending to the use of a feed-forward neural network. We use two hidden layers, each with a dropout layer to act as an activity regularizer. The latter is designed to artificially remove connections between neurons in order to prevent overfitting of the dataset, which occurs when the model develops highly specific pathways from input variables to the target, and loses the generality of fit. This is unfavorable in deployment, as new data that is processed by the model may be misclassified. Layers are composed of numbers of neurons that are powers of two, as the number of attributes in the dataset is also a power of two. In addition, GPU training benefits
Figure 7.1: Schematic of dense neural network used to classify the PIDD.
from the use of matrices that conform to powers of two, improving training times.

The model was trained using the Adam optimizer\textsuperscript{[76]} at different learning rates, including 0.0001, 0.001, 0.005, 0.01, 0.02, 0.05, to sample the effect of learning rate on the model outcomes. Each of the three dataset preprocessing regimes are tested using 5-Fold crossvalidation\textsuperscript{[77]} to sample the accuracy of the model across different regions of the dataset. K-fold crossvalidation creates multiple train-test splits of the dataset and builds independent models using each of the splits. This method validates the model by reducing the potential of a random split producing exceedingly high model accuracy by random chance.

Zehra \textit{et. al}\textsuperscript{[72]} report an upper confidence classification accuracy of 75.39\% using a neural network model on an unmodified PIDD (regime 1), where this work shows 81.61\%. Our results are in agreement with those of the literature, with a significant improvement in accuracy. This may be attributed to the use of dropout layers to reduce overfitting, but further analysis is required to confirm this.

Table 7.2: Classification accuracy on cleansed PIDD. Data was scaled using Min-Max scaling to (0,1). Table displays accuracy of train/test using k-fold crossvalidation (5 folds) with the Adam optimizer. Confidence intervals are reported using the $t$-score method at 95\% confidence using 5 samples.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mean Training Acc.</th>
<th>Mean Test Acc.</th>
<th>Best Adam LR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regime 1</td>
<td>$77.83%\pm2.35%$</td>
<td>$77.22%\pm3.54%$</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>$CI_{95}$ [74.91%, 80.75%]</td>
<td>$CI_{95}$ [72.83%, 81.61%]</td>
<td></td>
</tr>
<tr>
<td>Regime 2</td>
<td>$77.65%\pm1.15%$</td>
<td>$78.57%\pm2.58%$</td>
<td>0.005</td>
</tr>
<tr>
<td></td>
<td>$CI_{95}$ [76.22%, 79.08%]</td>
<td>$CI_{95}$ [75.37%, 81.77%]</td>
<td></td>
</tr>
<tr>
<td>Regime 3</td>
<td>$77.92%\pm0.92%$</td>
<td>$77.21%\pm3.49%$</td>
<td>0.005</td>
</tr>
<tr>
<td></td>
<td>$CI_{95}$ [76.78%, 79.06%]</td>
<td>$CI_{95}$ [72.88%, 81.54%]</td>
<td></td>
</tr>
</tbody>
</table>

Results under preprocessing regimes 2 and 3 demonstrate similar accuracies, where regime 2, \textit{full removal of missing values}, resulted in an upper confidence accuracy of 81.77\%, and regime 3, \textit{imputation with a fixed negative integer}, resulting in an upper confidence accuracy of 81.54\%. The improvement in model accuracy
in regime 2 is attributed to the completeness of the dataset, in which all input parameters were fully defined and did not introduce missing or imputed values to the training procedure. Comparing this to the accuracy of 81% reported by Zehra et al.\cite{72} on a PIDD with missing values removed, we find that the model performed nearly identically. The preprocessed dataset presented by Zehra et al.\cite{72} also included a discretization procedure on the continuous variables present in the dataset, associating a finite value with a range of continuous variables. Discretization has been shown to reduce fitting times and improve classification accuracy under certain circumstances, and will be studied in future work.

We note the improvement in accuracy when using the dataset developed in regime 2, as the presence of missing values does not interfere with model training. Results are in congruence with literature values\cite{72} for this dataset, confirming that the model captured the most common features of the dataset. Crossvalidation results show that the model performs well on subsets of data, indicating that the model is not over-fit to the dataset. Improved prediction accuracy may be achieved by adjusting model parameters, preprocessing techniques, or a combination of both.

The use of neural network type models for the analysis of clinical data sets is promising, as these models have a high capacity for learning complex relationships within a dataset. On advantage of neural networks with dropout layers is the development of robust models that are not over-fit to data, allowing them to be used in general cases with samples drawn from a similar data space. Although the classification accuracies presented in this work do not significantly exceed those published in the literature\cite{72}, the ability to apply the model to future data, as well as the ability to learn from new data atop the current model, is a useful feature that makes neural networks a powerful and realizable model type for rapid analysis of clinical data sets.
7.1.2 Bayes Models

The PIDD was preprocessed in the same manner as in Section 7.1.1, producing three dataset regimes. The values were not scaled after initial preprocessing, as the Bayesian methods used for analysis do not require data to be scaled near unit mean and unit variance like neural network models.

Three model Bayesian classifiers were developed to analyze the PIDD using Bayesian methods, namely (1) a naive approach, (2) a multivariate Gaussian approach, and (3) a two-component general mixture model. Naive Bayes classifiers use independent probability distributions to model each variable in a dataset, and thus can contain a variety of different distributions. Two models were developed, one using exclusively normal distributions, and another using a mixture of normal and exponential distributions, to model the PIDD. In addition, a general Bayes classifier was developed using a tied multivariate Gaussian distribution. Bayes classifiers generally use covariance between input variables when fitting, and result in potentially robust classifiers. However, if input variables are poorly modeled by a Gaussian distribution, the model will not perform well. Finally, a general Bayes classifier using general mixture models (GMM) in place of a traditional probability distribution allows the model to contain sub-distributions that describe certain pieces of data, potentially improving classification ability and dataset segregation.

Each model was tested using 10-fold crossvalidation regimen described in Section 7.1.1, all results are shown in Table 7.3. Regime 2, full removal of missing values, produced the best results of all regimes in the Bayes model study. Regime 2 presents a dataset that is free of missing values, a critical importance to Bayesian models. Because each model is based on the ability to divide the data into sets of probability distributions, the presence of missing values or outliers creates a skewed representation of the data, resulting in poor classification accuracy.
Table 7.3: Prediction accuracy using Naive Bayes and Bayes Classifier Models. Confidence intervals (CI) are reported using the $t$-score method at 95% confidence using 10 samples.

<table>
<thead>
<tr>
<th>Regime</th>
<th>Naive Bayes Methods</th>
<th>Bayes Classifier Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
<td>Mixed</td>
</tr>
<tr>
<td>1</td>
<td>75.38% ± 5.30%</td>
<td>76.55% ± 4.48%</td>
</tr>
<tr>
<td></td>
<td>CI$_{95}$ [71.59%, 79.17%]</td>
<td>CI$_{95}$ [73.35%, 79.75%]</td>
</tr>
<tr>
<td>2</td>
<td>77.67% ± 7.20%</td>
<td>78.33% ± 7.07%</td>
</tr>
<tr>
<td></td>
<td>CI$_{95}$ [72.52%, 82.82%]</td>
<td>CI$_{95}$ [73.27%, 83.39%]</td>
</tr>
<tr>
<td>3</td>
<td>75.51% ± 5.21%</td>
<td>76.16% ± 4.84%</td>
</tr>
<tr>
<td></td>
<td>CI$_{95}$ [71.78%, 79.24%]</td>
<td>CI$_{95}$ [72.70%, 79.62%]</td>
</tr>
</tbody>
</table>
The mixed distribution naive model resulted in an upper confidence accuracy of 83.39%, a 0.57% improvement over the upper confidence of the normal naive model, due to the presence of columns that follow a non-normal distribution. The utility of mixed distributions within the naive model is the ability to capture more accurately the underlying distributions of the data, and improve classification accuracy without additional effort aside from identification of the likely distributions in the dataset.

The application of mixed models such as the multivariate Gaussian model shows excellent classification accuracy, but with a higher standard deviation than other models. The performance of the multivariate model is likely due to the existence of dependent features in the dataset that benefit from covariant relationships available to the model during fitting, a feature the naive models do not have. However, because the data is not entirely normally distributed, the multivariate model is still unable to capture certain aspects of the data, resulting in similar performance to that of the mixed naive model.

The applicability of these methods stems from their ease of design and relative speed of model fitting. This work showed the best mean classification accuracy using mixed naive models and multivariate Gaussian models, although we do not see significant improvements in these models over that of the literature. Zehra et al.\[72\] show an upper confidence classification accuracy of 76.3% using a naive Bayes classifier on an unmodified PIDD, and 80.3% using a preprocessed model with missing values removed. In comparison, this work found similar accuracies to that of the unmodified dataset, but exceeded the accuracy estimates reported by up to 3%.

The outcome of the Bayes models demonstrate the effectiveness of these methods for extraction of model dependencies from a dataset in a rapid manner, as well as the ability to combine different distributions to better describe a dataset. In an effort to ensure a generalized method for extraction of dataset features, Bayesian
methods enable data sets composed of well defined clusters of data to be modeled in an unsupervised manner.

### 7.1.3 Decision Tree Model

Decision tree models are useful methods for classification and regression of mixed data sets; that is, data sets containing both continuous and discrete variables. In addition, decision trees are able to handle categorical data inherently, as the decision function is based on variable relationships and not direct values. Here we use a more advanced decision tree model, called a gradient boosted classifier, to analyze the PIDD.

The PIDD was preprocessed in the same manner as in Section 7.1.1, producing three dataset regimes. We study the same group of data sets, however, because decision trees are often robust to the need for preprocessing, steps to further scale or modify the dataset were not taken before model development.

We use a gradient boosting classifier (GBC) to predict the outcomes associated with the PIDD. The dataset is composed of eight input variables and a single target variable, *outcome*, representing a patient with “no diabetes” or “with diabetes”, respectively. Decision tree style models require parameter optimization to achieve maximum classification accuracy on a specific dataset. Table 7.4 shows the parameters that are optimized for each dataset regime, including descriptions. Optimized parameters were found using a grid search with crossvalidation over the relevant parameters of the model.

Table 7.5 shows the optimized parameters for each of the three dataset preprocessing regimes. The optimized parameters are significantly different than the default parameters of the GBC model, however for each regime, we find that the parameters are similar. Because GBC models require dataset specific optimization, it is
Table 7.4: Default gradient boosting classifier parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n_estimators</td>
<td>[1,inf)</td>
<td>100</td>
<td>Number of trees used in the GBC ensemble</td>
</tr>
<tr>
<td>learning_rate</td>
<td>(0,inf)</td>
<td>0.1</td>
<td>Update fraction of model weights</td>
</tr>
<tr>
<td>subsample</td>
<td>(0,1)</td>
<td>1.0</td>
<td>Fraction of dataset to sample from when generating splits</td>
</tr>
<tr>
<td>min_samples_split</td>
<td>[1, N)</td>
<td>2</td>
<td>Min. number of samples allowed to create a split in the tree</td>
</tr>
<tr>
<td>min_samples_leaf</td>
<td>[1, N)</td>
<td>1</td>
<td>Min. number of samples to denote a leaf</td>
</tr>
<tr>
<td>min_weight_fraction_leaf</td>
<td>(0,1]</td>
<td>0.0</td>
<td>Min. fraction of data defined by a single leaf</td>
</tr>
<tr>
<td>max_depth</td>
<td>[1, inf)</td>
<td>3</td>
<td>Max. number of branch levels in the tree</td>
</tr>
</tbody>
</table>

not surprising that under different preprocessing regimes the model converges near the same group of parameters, as the underlying structure of the dataset remains mostly intact regardless of preprocessing technique. We notice the higher optimum learning rate for regime 2, which likely required less distinction between regions of the data, and thus the model weight updates could be more drastic than regimes in which the data contained missing or placeholder values. The min_samples_split parameter showed the greatest deviation from default values, and remained constant across all models.

Both the default and optimized GBC models are used to predict the target binary class variable outcome for each respective dataset regime. Each analysis uses a 20-fold crossvalidation (CV) of the dataset to ensure the model was not over or under-fit. The results of the analysis are shown in Table 7.6, which details the mean classification accuracy and standard deviation of the CV, as well as 95% confidence intervals (CI) for each result. Classification accuracies between the models built using default parameters and optimized parameters show little overall improvement in classifica-
Table 7.5: Optimized parameters for the gradient boosting classifier model used to analyze the PIDD

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Regime 1</th>
<th>Regime 2</th>
<th>Regime 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>n_estimators</td>
<td>100</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>learning_rate</td>
<td>0.1</td>
<td>0.005</td>
<td>0.02</td>
<td>0.005</td>
</tr>
<tr>
<td>subsample</td>
<td>1.0</td>
<td>0.9</td>
<td>0.9</td>
<td>1.0</td>
</tr>
<tr>
<td>min_samples_split</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>min_samples_leaf</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>min_weight_fraction_leaf</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>max_depth</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

The upper 95% confidence interval (CI) for regime 1 was 79.74%, suggesting a maximum possible accuracy of approximately 80% using the default model. This estimate was only improved by 0.65% when optimized parameters were used in the model. We note that regime 3 showed identical results using the default model, demonstrating how decision trees handle data values as pseudo-categorical, in which...
a 0 value and a -1 value were explained by the same grouping. This is also evident in the results of the optimized model, with a slight deviation between regime 1 and regime 3, likely due to the difference in model parameter choice.

Regime 2 shows the best classification accuracy of all regimes, with a 0.66% upper improvement over the default model. The results also demonstrate better classification accuracy compared to regimes 1 and 3, due in part to the completeness of the data (no missing or placeholder values).

In comparison to other classifier methods presented in this work, the GBC model performed better than Bayesian methods, and are in line with neural network models. Results by Zehra et al.[72] for multiple model types (excluding decision tree models) using unprocessed data (regime 1 in this work) report the upper 95% CI value average accuracy as 74.37%. The use of GBC models show a significant improvement over the average accuracy reported, and demonstrates the utility of these models for predictive analysis even under default conditions.

The use of decision trees for dataset classification comes with some drawbacks, namely the need for extensive parameter optimization that must be performed for every dataset being processed, and the potential for overfitting. In addition, if a dataset does not contain a sufficient range of values for each variable, application of the model to future data may yield poor classification results. The advantage of decision trees lies in their interpretability and function as a feature selector, which may be applicable to the development of different model types. Upon comparison of the prediction accuracy of the gradient boosted decision tree model to that of the neural network and Bayesian models, we see general agreement of prediction accuracy, with only slight improvement over neural network or Bayesian models.
7.2 Alcoholism EEG Study

7.2.1 Convolutional Neural Network Model

The use of convolutional neural networks (CNN) for time series analysis requires conversion of the dataset into a valid form of input. Conversion of time series input to a valid pseudo-image was performed by applying a Markov transition field transformation to the data. Each MTF plot is a representation of the transition of states within the parent time series, and can be interpreted as a heat map of recurrent states along the time series. Figure 7.2 demonstrates the recurrence of states in each time series, where brighter colors indicate a transition back to the same state.

Input data was originally in the form of a $(60 \times 256 \times 64)$ tensor. The first dimension represents the subject, the second dimension a sample, and third dimension a specific channel of EEG output. Using the first dimension, labels were derived for each sub-matrix based on the subject origin, forming a two-class label set of control and alcoholic subjects. Each sub-matrix was then broken into its individual channels, forming 64 sets of 256 samples per subject, then transformed into a MTF plot, creating a total of 64 images per individual. The dataset is composed of 60 subjects, 30 control and 30 alcoholic, generating 3840 MTF plots.

The alcoholism EEG dataset (Section 6.2) contains 60 sets of 64-channel EEG data split evenly between control and alcoholic subjects. The use of convolutional neural networks (CNNs) to classify this dataset was chosen because of the unique ability to transform time series into pseudo-images for processing. The goal of this model is to classify a single channel of EEG data as belonging to either a control or alcoholic subject. The network used in this model was designed to mimic popular CNN architectures used for image classification tasks, which employ convolution and max pooling layers for extraction of image features. The network design shown
Figure 7.2: Examples of control subject and alcoholic subject time series (a and b) and equivalent Markov transition field (c and d).
in Figure 7.3 is based on the structure of the image classification models VGG Net\textsuperscript{[78]} and AlexNet\textsuperscript{[79]}. The complete makeup of the network is shown in Table 7.7, which is composed of numerous layer types including convolution and pooling layers, dropout layers and flattening layers, dense layers, and softmax classification layers.

![Figure 7.3: A diagram of the convolutional neural network used to process the alcoholism EEG dataset.](image)

Table 7.7: Layer configuration for the convolutional neural network used in this study

<table>
<thead>
<tr>
<th>Index</th>
<th>Layer type</th>
<th>Input shape</th>
<th>Output shape</th>
<th>Activation</th>
<th>Kernel size</th>
<th>Num. filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Input</td>
<td>(256, 256, 1)</td>
<td>(256, 256, 1)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>Convolution</td>
<td>(256, 256, 1)</td>
<td>(256, 256, 32)</td>
<td>ReLU</td>
<td>(3, 3)</td>
<td>32</td>
</tr>
<tr>
<td>2</td>
<td>Convolution</td>
<td>(256, 256, 32)</td>
<td>(256, 256, 32)</td>
<td>ReLU</td>
<td>(3, 3)</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>Max Pooling</td>
<td>(256, 256, 32)</td>
<td>(128, 128, 32)</td>
<td>-</td>
<td>(2, 2)</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>Convolution</td>
<td>(128, 128, 32)</td>
<td>(128, 128, 64)</td>
<td>ReLU</td>
<td>(3, 3)</td>
<td>64</td>
</tr>
<tr>
<td>5</td>
<td>Convolution</td>
<td>(128, 128, 64)</td>
<td>(128, 128, 64)</td>
<td>ReLU</td>
<td>(3, 3)</td>
<td>64</td>
</tr>
<tr>
<td>6</td>
<td>Max Pooling</td>
<td>(128, 128, 64)</td>
<td>(64, 64, 64)</td>
<td>-</td>
<td>(2, 2)</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>Dropout</td>
<td>(64, 64, 64)</td>
<td>(64, 64, 64)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>Flatten</td>
<td>(64, 64, 64)</td>
<td>(262144,)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>Dense</td>
<td>(262144,)</td>
<td>(256,)</td>
<td>ReLU</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>Dropout</td>
<td>(256,)</td>
<td>(256,)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>Prediction</td>
<td>(256,)</td>
<td>(2,)</td>
<td>Softmax</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The alcoholism EEG dataset (AED) contained stimulus responses of subjects under three regimes of stimulation, termed single, S1, and S2 stimulation. Single stimulation indicated the use of a single image stimulus, S1 indicates the use of two non-matched stimuli (different images), S2 indicates the use of two matched stimuli (same image repeated). The model was fit separately to each of these groups of data, and model accuracy validated on the fitted set and the unseen remaining sets.
to determine the ability of the model to identify subject class under limited training.

The model was also fit to the entire dataset and validated in a similar fashion. Each subset of data used to train the model used 90% of the input image data, and validated using the remaining 10%. To ensure robustness, a small fraction of random images were retained from training and used as validation after model fitting. Each image was classified as originating from either a control or alcoholic subject.

From Table 7.8, we can see that the model performed best when trained on the entire dataset, which included single, S1, and S2 stimuli responses. In general, CNNs perform best when given a large set of images to train on, as well as a variety of different input examples that are mapped to the desired output. Each stimulus response varied between single, S1, and S2 sets, and thus the model was more robust to differences across each image when trained using a larger sample set. In particular, we note the performance across all test sets of the full-dataset trained model, which maintained an overall classification accuracy of > 90%, even during post-validation testing.

<table>
<thead>
<tr>
<th>Trained on</th>
<th>Tested on</th>
<th>Single</th>
<th>S1</th>
<th>S2</th>
<th>Full dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td></td>
<td>100%</td>
<td>67.96%</td>
<td>49.51%</td>
<td>66.57%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(68.75%)</td>
<td>(48.88%)</td>
<td>(80.8%)</td>
<td>(80.76%)</td>
</tr>
<tr>
<td>S1</td>
<td></td>
<td>63.11%</td>
<td>100%</td>
<td>76.69%</td>
<td>79.38%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(64.06%)</td>
<td>(80.07%)</td>
<td>(76.17%)</td>
<td>(76.16%)</td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td>60.19%</td>
<td>80.58%</td>
<td>100%</td>
<td>78.55%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(65.23%)</td>
<td>(75.95%)</td>
<td>(82.03%)</td>
<td>(82.03%)</td>
</tr>
<tr>
<td>Full dataset</td>
<td>$\mathbf{100%}$</td>
<td>$\mathbf{98.05%}$</td>
<td>$\mathbf{100%}$</td>
<td>$\mathbf{95.26%}$</td>
<td>$\mathbf{91.43%}$</td>
</tr>
</tbody>
</table>

$\mathbf{98.82\%}$ $\mathbf{98.83\%}$ $\mathbf{91.41\%}$ $\mathbf{91.43\%}$

The use of methods such as CNNs as a classifier of time series is a relatively new
technique, and requires further analysis to determine alternative uses and potential drawbacks of the method. This work applied the use of CNNs in a supervised fashion, but further work may reveal their use outside of this domain.

### 7.2.2 General Mixture Model

The use of general mixture models (GMM) to analyze the alcoholic EEG dataset takes on a similar role as the CNN. We may use a GMM for classification of full-duration time series, by modeling the individual channels within each component of the GMM as components of the GMM, thus creating a separable classification scheme of each subject type. Classification of full-duration time series performs a similar task as the analysis using CNNs, specifically that each one-second sample may be labeled as belonging to an alcoholic subject or a control subject.

Preprocessing of the input dataset is not a critical first step when using a general mixture model. Because of the nature of the model, each value is fit using a Gaussian distribution, which can be centered around any mean value present in the data. In general, if the dataset contains samples that are relatively well centered about a single baseline value, such as that of the alcoholism EEG dataset, which are centered around 0 mV, the model is able to capture the underlying distributions of values. In the event that data sets contain multiple mean values, the input can be scaled using a standard approach, as outlined in Section 5.2.

The GMM model was developed as an N-component model using multivariate Gaussian distributions to represent each component. The dimensions of the dataset are (60, 256, 64), described by 60 subjects, each containing a 256 sample, 64-channel EEG readings. We wish to define the individual subject classes as either alcoholic or control, based solely on the input data without direct labeling. Because we use no predefined labels during training, the GMM is developed using a range of com-
ponents in order to ensure the best model is derived and the results compared to
determine the correct number of classes.

The use of the GMM model for dataset classification is used in an expanded fash-
ion, that is the 60 individual subject arrays of (256, 64) are transformed into a two
dimensional array of shape (15360, 64). The new array has corresponding labels that
are not used during training, but instead for validation of accuracy of fitting. Three
models were fit using 2, 3, and 4 components, respectively. The resultant weights for
each dataset generated are presented in Table 7.9.

Table 7.9: Cluster weights defined by the GMM after fitting to the alcoholism
EEG dataset.

<table>
<thead>
<tr>
<th>Components</th>
<th>Class 1</th>
<th>Class 2</th>
<th>Class 3</th>
<th>Class 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.4999</td>
<td>0.5001</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>0.0448</td>
<td>0.4674</td>
<td>0.4878</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>0.0682</td>
<td>0.0962</td>
<td>0.3819</td>
<td>0.4537</td>
</tr>
</tbody>
</table>

The weights of each class indicate the relative fraction of data belonging to a
particular class of the dataset. The ordering of classes is unimportant, class order
is defined by arbitrary means and thus does not present useful information. Notice
however that under each model, there exists a class with weight of approximately
50%, followed by a class of similar weight, followed by classes of low weight relative
to the best two. By adjusting the number of components of the GMM, we find a
tendency to converge to an approximately two-class model, indicative of the data
actually being composed of two classes. Mapping of class labels from the arbitrary
mapping of the model to the actual labels is performed by unique label count, in
which the model class label is mapped to its appropriate true class by comparative
analysis of the indices of the predicted labels and true labels.

We know the input data contains two classes of equal weight, which is described
by the 2-component model very well. Testing of model generated labels against
known labels for this model result in 100% classification accuracy. Alcoholic and control subjects exhibit unique characteristics in respective EEG samples that are captured well by the multivariate Gaussian distribution in 64 dimensions. The label accuracy of this model indicates that there is a significant separability of the data into distinct classes, made clear by the ability of the model to predict each class with 100% accuracy. Use of GMMs is most applicable to processes that do not maintain a direct dependence on time, but instead present modes of data that are distinct across the dataset and can be extracted by comparing the likelihood of existence in one mode or another via probabilistic means. For example, the GMM model used to classify the alcoholism EEG dataset did not consider the time-dependence of the data, but rather the span of values produced throughout the time series. The model was trained using the individual subject responses concatenated together into a large vector, rather than considering each subject individually, in turn removing the time dependence of the model and considering the data to be stationary.

This type of analysis is a useful means of handling time series when the process-specific information is to be extracted. The model is able to identify the existence of two major classes within the dataset, which were not time dependent, but process dependent. Knowledge of the original dataset, exemplified in Figure 6.2, shows a unique set of modes in the control and alcoholic subjects, in which alcoholic subjects exhibit a more volatile response to stimulus than the control subject. This trend of volatile brain activity in alcoholic subjects is not a time dependent feature, but a process specific feature related to the subject type.

7.2.3 Hidden Markov Model

An alternative analysis of the alcoholism EEG dataset is made by the use of hidden Markov models (HMM). These may be used to model individual time series to deter-
mine interesting modes of data present in the time series automatically. This type of modeling is useful for identification of modes in the data that are unlike other regions, based on both value and order in a time series.

The EEG alcoholism dataset presents a large number of channels of EEG data per subject, and as a result may contain low-variance or uninformative channels. Notice that certain channels in Figure 6.2 show little variation between control and alcoholic subjects. These channels offer little distinguishing information (they do not vary across subject type), and may introduce unneeded complexity. To control this, we introduce principal component analysis (PCA) decomposition, which identifies the eigenvalues of the matrix (per column), and keeps some user defined number of columns based on those with the highest eigenvalues. PCA is performed on 64 channels of EEG data, with 16 channels kept for use in model development.

We fit the alcoholism EEG dataset using a hidden Markov model to perform unsupervised identification of class type, and tagging of the onset of unique regions of data. Unsupervised classification is a useful tool if a dataset is unlabeled, and may contain unique classes of data. The identification of important regions of data within a time series is useful for analysis of outliers in a dataset, as well as classification of normal and outlier modes. We perform two types of analysis on the EEG alcoholism dataset, first a model to identify classes of subjects in an unsupervised manner, and second to identify critical regions of each sample.

Figure 6.2 shows the presence of unique activity in the alcoholic subject group, specifically the magnitude of brain activity in the latter portion of the sample. Certain channels present very high amplitude relative to the rest of the sample, and thus may be useful in the identification of important regions in the sample in an unlabeled manner.

In the first, fully unlabeled scheme, a system of HMMs, \( h \in H \), are trained using
the full dataset, each with 2, 3, 4, 6 and 8 hidden states denoted as $N$. The resulting models are then combined into a master HMM, $Q$, consisting of the previously compiled models $H$, with equal probability of transition to one of $h \in Q$. A diagram of the model architecture is shown in Figure 7.4. Models $h \in H$ are trained using independent normal distributions, each representing a channel of EEG data in each subject sample. The master model is trained using the full dataset in order to identify significant classes of subjects. To ensure the correct number of classes are identified, the model is configured with a varying number of hidden states, number of models $H$, and number of channels of data in the dataset. The resulting model acts as an unsupervised classifier, identifying relevant classes within the dataset. Identification of classes is performed by passing in each subject sample as a (256, 16) matrix, and the matrix labeled as belonging to one of $h \in Q$ models. The number of samples labeled as belonging to each class is counted and the results analyzed for the total number of identified classes. Ideally, the model will identify only certain states as relevant, thus indicating the total number of recognized classes in the dataset, as well as their frequency of occurrence.

Figure 7.4: Diagram of the HMM configuration used in scheme one (unsupervised class identification) of the HMM analysis.
Under the second scheme, a single HMM is trained on the full dataset using 2, 3, 4, and 6 hidden states, denoted $N_s$. The model is then used to predict the most likely state at each time step for each of the 60 subject samples, producing 60 predictions each time step in the sample. Predictions are then split into groups based on their subject class of origin, and each unique state counted along the time axis, producing a vector of class counts with shape $(256, N_s)$, with values corresponding to the number of instances of a particular state at a particular time step across all subjects in the control or alcoholic group. The counts of each class are plotted along a time axis according to the time step the classes were sampled at, displaying the activity of certain states that are active for each subject class at that time step. The presence of outlier regions in some of the subject time series, in particular alcoholic subjects, present an opportunity to discover unique regions of data, which may be useful for later identification of characteristic behavior in either control or alcoholic subject EEG responses.

The outcome of scheme one allows the determination of the number of classes in the dataset. Using an array of hidden states to define the the HMM, the dataset is consistently classified into two distinct classes, as expected, given that the dataset contained two real classes of control and alcoholic subjects. The results of this classification testing is shown in Table 7.10.

Table 7.10: Unique number of classes identified by the HMM classification. Results contain the number of classes as well as the respective frequency of class labels in the EEG alcoholism dataset.

<table>
<thead>
<tr>
<th>Models</th>
<th>Number of hidden states</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2 : (0.55,0.45)</td>
</tr>
<tr>
<td>3</td>
<td>2 : (0.55,0.45)</td>
</tr>
<tr>
<td>4</td>
<td>2 : (0.58,0.42)</td>
</tr>
</tbody>
</table>

Each configuration indicated the presence of two distinct class types in the dataset.
Models $Q$, with a number of sub-models $M > 2$ are able to define up to $M$ classes; however, due to the probabilistic evaluation of the dataset, the models converged toward only two outstanding models based on the probability of finding a sample in a particular class. The use of HMMs for this kind of analysis demonstrates powerful ability for the segregation of data in an unsupervised manner, based on the underlying features of the dataset.

After converging on the total classes present in the dataset, each subject sample analyzed using scheme two. We use the true labels identified in the dataset when feeding data to the model. If the dataset is truly unlabeled, the segregation and labeling of the dataset may be performed using scheme one before performing scheme two. The model is defined using a varying number of hidden states, $N_s$, as each subject class may contain unique characteristics that define the time series, and thus a sufficient number of hidden states, defined by trial and error, should be used to capture multiple phenomena in each subject class. This work will analyze four sets of models with a number of hidden states $N_s \in [2, 3, 4, 6]$. Each model is used to uncover regions of interest in each time series relative to control and alcoholic groups, and analyzed for similarity to models with fewer hidden states and evolution of new or interesting states.

Figure 7.5 shows the results of the simple two-state model, in which the presence of certain state activity along the time axis varies for the control and alcoholic group, demonstrating the emergence of different phenomena in each group of subjects. Control subjects demonstrate a dominance of state 0 in the early stages of stimulus ($t < 0.3s$), followed by a distinct loss of characterization, exemplified by the presence of roughly equal state frequency, in the latter half of the time period. In contrast, alcoholic subjects exhibit a sustained dominance of state zero in the time period ($t < 0.7s$), followed by a rapid approach toward equal state frequency for a
short time \((0.7s < t < 0.95s)\), with final divergence back to state zero dominance. The difference in these plots shows a late response to stimulus in alcoholic subjects.

![Figure 7.5: Analysis of alcoholism EEG dataset using an HMM with two hidden states.](image)

The three-state model, shown in Figure 7.6, demonstrates some of the same types of activity as the two-state model; however, we note a distinct dominance of state two over time period \((0.3s < t < 0.6s)\). At time \((t > 0.6s)\), the appearance of states zero and one begin to progress upward, but do not overtake state two throughout the duration of the samples. Alcoholic subjects demonstrate the opposite occurrence of states, with state one beginning as dominant for time \((t < 0.4s)\), where after states one and two become entangled and lose characterization. There is a distinct rise in activity of state zero in the latter portion of the alcoholic sample at time \((t > 0.7s)\), similar to the response of the two state model, shown in Figure 7.5.

Figure 7.7 shows the activity of the four state model, which demonstrates many of the same characteristics as the three-state model, particularly for the control group. We see a dominance of state one over time period \((0.3s < t < 0.6s)\), with similar response as the three state model in states zero and two. State two shows relatively low impact across the model, with only a small number of occurrences across the time axis. The alcoholic group shows new activity in states zero and two,
Figure 7.6: Analysis of alcoholism EEG dataset using an HMM with three hidden states.

but generally remains about the midline of the plot, indicating non-dominant behavior. The presence of state three at time \( t > 0.7s \) maintains a similar shape and frequency as that in the three state model, indicating a potentially static occurrence in the dataset. The activation of state three late in the response is unique to the alcoholic group, and is likely a feature that is unique to those subjects.

Figure 7.7: Analysis of alcoholism EEG dataset using an HMM with four hidden states.

Figure 7.8 shows the activity of the six-state model, again demonstrating highly similar activity to the four-state model, with control group activation of state one in the time period \( 0.3s < t < 0.6s \). States zero and five show similar activity across the control group, indicating a similar underlying probability distribution. State three
shows little activation and may be an unnecessary state. The alcoholic group shows similarity to its corresponding four-state model, described by overlapping states zero and two along the centerline, the peaking of state three at time \((t > 0.7s)\), and little activation of states four and five. We note that exchange of state one activity across the control group and alcoholic group, indicating that state one is present primarily in control subjects and not alcoholic subjects.

Figure 7.8: Analysis of alcoholism EEG dataset using an HMM with six hidden states.

The identification of different regions of activity within the control and alcoholic groups demonstrates the utility of HMMs as an autonomous classification model. We are able to easily identify important regions within the dataset based on state activity of the model, presenting a means to extract useful information about data structures and critical regions. The analysis performed in this work may be expanded to include a larger number of hidden states or different underlying probability distributions, which may exemplify different features of the data. The robust capability of HMMs is advantageous for handling multiple types of data with a time-dependent nature. Because HMMs handle continuous data, they are able to capture the variance of features that explain unique phenomena.
7.3 Diabetes Readmission Study

The diabetes readmission dataset contains a very large amount of sample data, with each sample originating from a different class of patient. In order to maximize effectiveness of the methods, we choose to use subsets of the dataset that apply to specific groups of patients. In particular, we will analyze caucasian women of age 50+. This demographic group is heavily affected by diabetes, and may present different characteristics that stand out from other demographic groups that make up the dataset. The selection of a particular demographic is in the effort to build more specific models to better understand groups individually and create more accurate predictive models.

7.3.1 Bayes Models

In the case of the diabetes readmission dataset, there are highly non-linear dependencies throughout the data that may introduce difficulty in designing a Bayesian model that is able to capture the relevant phenomena and analyze this dataset effectively.

We use the same methods to preprocess this dataset as in the neural network model (Section 7.3.2), and select the rows that correspond to the group described in Section 7.3.2. The resulting dataset contains 7493 patient records, roughly 7.6% of the original dataset.

Two models are developed to study the selected group of patients: (a) Bayes classifier with multivariate normal distributions, and (b) Bayes classifier with independent normal distributions. Each distribution type is trained on two variants of the dataset, one containing three readmission classes (no readmission, less than 30 days for readmission, and greater than 30 days for readmission), and the other containing two readmission classes (no readmission, and readmitted). The latter set is created
by modifying the labels of the dataset such that there is no distinction between read-mission times, forming a two-class labeling of the dataset.

Each model is trained in a labeled fashion, and the results compared using confusion matrices of the predicted results. The precision metric of the confusion matrix is the fraction of values correctly classified according to the actual data labels. The recall of the confusion matrix is the fraction of the values that were consistently classified according to the model fitting. We use the $F_1$ score as a metric to analyze the prediction accuracy results for each model, which is simply the harmonic mean of the precision and recall scores of each model, and represents the ability of the model to correctly classify each input into the correct class. The output range of the $F_1$ score metric is $[0, 1]$, where 0 indicates no ability to classify, and 1 indicates perfect classification.

Table 7.11 shows the results of the three-class multivariate Gaussian model, which demonstrated generally poor precision and recall. A majority of the data was grouped into the “No” and “>30” classes, with little sensitivity to the “<30” class. This is likely due to relative similarities of data points in the “>30” and “<30” classes, in which the distinction between patients is not significant. These results lend to the dataset being inseparable based solely on the provided variables, and may require expansion of the recorded variables in the dataset.

Table 7.11: Confusion matrix of the multivariate Gaussian 3-class model.

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th>Actual sums</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td>&gt;30</td>
<td>&lt;30</td>
</tr>
<tr>
<td>No</td>
<td>2127</td>
<td>1182</td>
<td>339</td>
</tr>
<tr>
<td>&gt;30</td>
<td>1548</td>
<td>1026</td>
<td>311</td>
</tr>
<tr>
<td>&lt;30</td>
<td>535</td>
<td>339</td>
<td>86</td>
</tr>
<tr>
<td>Predicted sums</td>
<td>4210</td>
<td>2547</td>
<td>736</td>
</tr>
<tr>
<td>Recall</td>
<td>50.52%</td>
<td>40.28%</td>
<td>11.68%</td>
</tr>
</tbody>
</table>

Table 7.12 shows the results of the three-class independent normal model, and
similarly to the multivariate model, demonstrates poor precision and recall. This model is more sensitive to the “No” and “>30” classes, with zero data points identified as being part of the “<30” class. Based on the findings of this model, we confirm the non-distinguishability of the two readmission classes, in which patients may not demonstrate any significant difference in operational variables that predict their duration of absence from the clinical environment.

Table 7.12: Confusion matrix of the independent normal 3-class model.

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th>No</th>
<th>&gt;30</th>
<th>&lt;30</th>
<th>Actual sums</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td></td>
<td>3099</td>
<td>549</td>
<td>0</td>
<td>3648</td>
<td>84.95%</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>2535</td>
<td>348</td>
<td>2</td>
<td>2885</td>
<td>12.06%</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>811</td>
<td>149</td>
<td>0</td>
<td>960</td>
<td>0.00%</td>
</tr>
<tr>
<td>Predicted sums</td>
<td></td>
<td>6445</td>
<td>1046</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Recall</td>
<td></td>
<td>48.08%</td>
<td>33.27%</td>
<td>0.00%</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

In the case of the binary readmission labels, the two-class models show a better acuity for prediction in terms of both precision and recall. Table 7.13 shows the results of the two-class multivariate model, which produced a higher precision for the “No” readmission class and the “Yes” readmission class. Although the model did not capture more than 62.94% of the data accurately, the reduction of the number of readmission outcomes from to two to three shows better sensitivity to the data. We also note that the dataset is more balanced using the two-class model, in which there are approximately equal numbers of patients in each category.

Table 7.13: Confusion matrix of the multivariate Gaussian 2-class model.

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th>No</th>
<th>Yes</th>
<th>Actual sums</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td></td>
<td>2296</td>
<td>1352</td>
<td>3648</td>
<td>62.94%</td>
</tr>
<tr>
<td>Yes</td>
<td></td>
<td>2287</td>
<td>1558</td>
<td>3845</td>
<td>40.52%</td>
</tr>
<tr>
<td>Predicted sums</td>
<td></td>
<td>4583</td>
<td>2910</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Recall</td>
<td></td>
<td>50.10%</td>
<td>53.54%</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 7.14 shows the results of the two-class independent normal model, which performed worse than the multivariate model. The lack of covariance relationships in the independent normal model is the leading cause of the poor precision and recall rates of prediction, as relationships in the data may be dependent on combinations of variables to segregate classes accurately. The high precision of the “No” class and low precision of the “Yes” class produced by this model is not a sign of better performance, but rather the opposite. Because we reduce the dataset to a binary classification problem, there exists a 50/50 chance of guessing the correct class based on zero evidence. Thus we expect that under poor model fit, the precision should be at least 50%, which is not the case in these models.

Table 7.14: Confusion matrix of the independent normal 2-class model.

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th>Actual sums</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>3203</td>
<td>445</td>
<td>3648</td>
</tr>
<tr>
<td>Yes</td>
<td>3441</td>
<td>404</td>
<td>3845</td>
</tr>
<tr>
<td>Predicted sums</td>
<td>6644</td>
<td>849</td>
<td>-</td>
</tr>
<tr>
<td>Recall</td>
<td>48.21%</td>
<td>47.59%</td>
<td>-</td>
</tr>
</tbody>
</table>

Analysis of the confusion matrices shows a clear inability of the model to reliably classify the expected readmission rate of each patient. The $F_1$ scores for each model, shown in Table 7.15 and Table 7.16, are used to gauge the predictive power of the model and its ability to separate the data into the appropriate classes. We note that no model is able to achieve an $F_1$ score above 0.6224, indicating that these models are unable to capture the underlying conditions representative of each class in the dataset to a high degree of accuracy.

The reason for lack of classification resolution could be attributed to dataset labels not defined by the distributions of the input data, but rather correspond to some other variable not present in the dataset. The capture of certain characteris-
Table 7.15: $F_1$-scores for the 3-class models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Data class</th>
<th>No</th>
<th>&gt;30</th>
<th>&lt;30</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV Gaussian</td>
<td></td>
<td>0.5414</td>
<td>0.3777</td>
<td>0.1014</td>
</tr>
<tr>
<td>Indep. Normal</td>
<td></td>
<td>0.6141</td>
<td>0.1771</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 7.16: $F_1$-scores for the 2-class models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Data class</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV Gaussian</td>
<td></td>
<td>0.5579</td>
<td>0.4613</td>
</tr>
<tr>
<td>Indep. Normal</td>
<td></td>
<td>0.6224</td>
<td>0.1721</td>
</tr>
</tbody>
</table>

tics of the dataset by the Bayesian models does indicate some kind of correlation; however, the overlap of these underlying distributions may not be separable by the labels applied to them. There is likely a patient-specific variation in readmission that does not correlate with the operational variables contained in the dataset, and thus the operational features are unable to accurately predict patient readmission with a high accuracy. The addition of more patient-specific data may aid in the discovery of underlying phenomena for readmission. Logically, we conjecture that each patient maintains highly personal and specific factors that dictate their readmission likelihood, including elements that are not represented by operational data alone. Conditions outside of the clinical environment, such as lifestyle and nutritional constraints, may greatly affect readmission rates of certain patients. The dataset used in this work does not contain variables that represent information related to patient lifestyle, and thus we cannot determine outside factors that may contribute to the readmission time of a patient.
7.3.2 Neural Network Model

In order to utilize any of the approximately 50 variables present in the diabetes readmission dataset (DRD), preprocessing of each variable is performed before the being fed to the neural network model. Preprocessing is broken into three major categories: (a) data cleansing, (b) categorical value encoding, and (c) continuous value scaling. As with many data sets, missing values should be removed or completed (imputed) before proceeding with other preprocessing techniques. The DRD contains both columns and rows that contain missing information, and must be cleansed before other preprocessing can occur. As a result of the size of the dataset, as well as the number of unique values present, imputation of missing values is impractical and would introduce severe bias to the data. Table 6.2 identifies three variables: weight, payer code, and medical specialty, that contain greater than 50% of missing values, and so these variables were removed from the dataset. The presence of missing values in the remaining columns is small, and so the rows containing these values can be removed without the loss of a significant portion of the dataset. The final size of the dataset after cleansing is (96437, 46).

Variables containing categorical data are defined in two ways: (a) continuous categories, such as age and glucose, and (b) static categories, such as gender and race. In the former case, each value is reduced to an equivalent numerical value that is representative of the category. In the latter case, each value is encoded with a representative categorical label. For example, if a sample in the age column is “[40,50)”, it will be reduced to “40”. Similarly, gender may be encoded to an integer, for example, from Male to 0 and Female to 1. The immediate difference between these two methods is the conversion of a value to a continuous variable, or from a string-valued class to a integer representation of the class.

Because neural networks are based on the use of continuous variables, it is im-
important to handle the integer labeled classes in a way that is applicable to this requirement. Encoded integer classes may be further encoded during preprocessing by the OneHot method (Section 5.3), or by the use of a vector encoding layer in the neural network model.

For example, we count the total number of categorically encoded variables in a dataset as \( N \). In any given variable, there exist some number of categories, \( C \), that make up the values in the variable. If \( C \) is small, OneHot encoding is suitable for encoding, as the size of the resulting binary matrix will also be small. Conversely, if the number of categories \( C \) is very large, the resultant OneHot encoding will be very large. If this is accounted for each categorical variable \( N \) in the original dataset, using strictly OneHot encoding with increase the number of effective variables in the dataset by \( \sum_i N_i C_N \). This can be avoided by the use of an encoding layer in the neural network, which encodes each class label into a user defined vector of length \( E \), which represents the class in a unique and continuous manner that is suited for continuous valued neural network implementations. The resulting vector encoding will increase the number of perceived columns of input by \( \sum_i N_i E_N \). If we define \( E_i \ll C_i \) for each column to be encoded, we reduce the total number of resultant columns in the dataset significantly compared to a full OneHot encoding scheme.

Finally, columns containing continuous variables must be scaled to reduce training time and model bias to large input values. Columns are scaled individually using one of the scaling methods described in Section 5.2. The choice of dataset scaling technique is somewhat dependent on the activation function used in the hidden layers of the network, but is relatively robust to this choice. Linear scaling to values between \([-1, 1]\) and \([0, 1]\) is common, as is scaling to unit mean and unit variance across a column. The choice also depends on the spread of the data, and if it contains outliers.
Preprocessing of the dataset is dependent on the analysis to be performed by
the model. Due to the high complexity and multiple groups of patients represented
by the diabetes readmission dataset, we choose to use a subset of the dataset that
surrounds a particular patient type, in this case caucasian women age 50+, with a
minimum length of stay greater than or equal to seven days. This group of patients is
particular affected by diabetes and is studied independently in an effort to discretize
the likelihood of readmission. Rows that correspond to the group described above
were selected for use by the NN model. The resulting dataset contains 7493 patient
records, roughly 7.6% of the original dataset. We choose to reduce the number of
output classes to a binary set by adjusting the labels of the input dataset such that
readmission as a whole is represented by a single class.

The model used to analyze the diabetes readmission dataset consists of a feed-
forward neural network that is fed a combination of continuous variables and em-
bedded variables. We choose to use vector embedding inline with the model such
that training can be performed on the embedding layers in concurrence with the
other layers of the model. In addition, we use a single feed-forward dense layer of
1024 hidden units for model segmentation, and a single SoftMax prediction layer[^80]
for classification of the readmission rates. A diagram of the model layers, shown in
Figure 7.9, describes the architecture of the neural network model used to classify
the diabetes readmission dataset. The use of embedding layers is critical for the
handling of categorical variables such as race and gender.

The outcomes of the NN model are in alignment with those of the Bayesian mod-
els, showing little ability of the model to create definitive predictions of output class.
The confusion matrix in Table 7.17 describes the ability of the model to classify each
sample; we see that the output is similar to the Bayesian models, indicating a lack of
underlying model segmentation. Table 7.18 shows the $F_1$ scores for the NN model,
Figure 7.9: Flow diagram of the neural network model used to classify the diabetes readmission dataset.
where No, and Yes classes are 0.56 and 0.48, respectively.

Table 7.17: Confusion matrix of the neural network model used to analyze the diabetes readmission dataset.

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th>Actual sums</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>2268</td>
<td>1380</td>
<td>3648</td>
</tr>
<tr>
<td>Yes</td>
<td>2199</td>
<td>1646</td>
<td>3845</td>
</tr>
<tr>
<td>Predicted sums</td>
<td>4467</td>
<td>3026</td>
<td>-</td>
</tr>
<tr>
<td>Recall</td>
<td>50.77%</td>
<td>54.40%</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 7.18: $F_1$-scores for the 2-class NN model on the training data.

<table>
<thead>
<tr>
<th>Model</th>
<th>Data classes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>NN</td>
<td>0.56</td>
<td>0.48</td>
</tr>
</tbody>
</table>

The diabetes readmission dataset is complex and likely requires more in-depth model development in order to accurately classify patient readmission. This dataset contains many operationally-defined variables and few patient-centered variables. The inability of the model to find adequate decision boundaries may be the result of lack of sufficient information regarding patient health and lifestyle, which may be a contributing factor to their probability of readmission. Future work will be required to generate a more accurate model, possibly with different network architecture composed of more dense layers and normalization layers, or other types of preprocessing techniques to ensure patient characteristics are maintained and represented accurately. Other factors such as learning rate and loss metrics may also influence decision boundaries, forcing the model toward the accurate classification of a single class in the dataset. Neural networks are known for requiring very specific training parameters in order to capture the dataset without over or under fitting issues.
7.3.3 Decision Tree Model

For decision tree based models, the dataset does not need significant preprocessing before analysis. This work uses extreme gradient boosting (XGB)\[^{67}\] (Section 4.4.3) for analysis, performed using the \textit{H2O.ai} machine learning framework\[^{81}\]. The latter is a modern toolkit built using the Java language that contains numerous machine learning methods relevant to business analysis, and an excellent dataset handling interface. A built-in web-based UI called the \textit{Flow UI} is used to process and analyze the DRD. In particular, \textit{H2O.ai} maintains an implementation of XGB that is coupled with automated handling of categorical data sets. XGB is a modern abstraction of gradient boosting that has shown outstanding results as a classifier in the literature\[^{67}\]. We choose to use this method to analyze the DRD because of the complexity of the dataset and the relatively poor performance of more common methods like Bayesian analysis and neural networks.

Similar to the preprocessing performed on previous models, we choose a subset of the dataset for analysis of a particular patient type, namely caucasian women age 50+, with a minimum length of stay greater than or equal to seven days. Rows are selected that correspond to the group chosen; the resulting dataset contains 7493 patient records, roughly 7.6\% of the original dataset, and is free of missing values. We use similar methods to preprocess this dataset as in the neural network model (Section 7.3.2), however categorical data is not directly encoded or transformed in any way. The \textit{H2O.ai} framework is capable of encoding categorical data internally, and thus this encoding was not performed before model development.

We use the \textit{H2O.ai} framework for ease of handling categorical data, as well as the fast implementation of the XGB method. Default model parameters are used for the XGB classifier\[^{67}\], as the analysis of the PIDD using decision trees (Section 7.1.3) showed that model parameter optimization did not significantly improve outcomes.
We wish to analyze the use of XGB in terms of its potential for successful modeling of the complex DRD, which will be evident in model improvement over other models such as Bayesian methods or neural networks. Columns that specified individual medications are removed before feeding the data to the model, as these columns are summarized by the “diabetesMed” column in the dataset (Table 6.2), but all other columns are included.

The model parameters used in the XGB classifier model are shown in Table 7.19, which include similar parameters to those in the GBC model using to analyze the PIDD, but with some additional parameters specific to the XGB model. In particular, the method of tree growth, “grow_policy”, and convergence parameters, “alpha” and “nround”, are specific to the XGB model.
Table 7.19: Descriptions and default values of parameters used in the XGB\textsuperscript{[67]} classifier model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_leaves</td>
<td>0</td>
<td>Max number of leaves per branch, 0 is unlimited</td>
</tr>
<tr>
<td>eta</td>
<td>0.1</td>
<td>Learning rate parameter</td>
</tr>
<tr>
<td>objective</td>
<td>multi:softprob</td>
<td>Accuracy metric for learning updates, handles multiple categorical outputs</td>
</tr>
<tr>
<td>colsample_bylevel</td>
<td>1</td>
<td>Fraction of each column sampled for splitting at each level of the tree</td>
</tr>
<tr>
<td>nthread</td>
<td>72</td>
<td>Number of threads to use when fitting</td>
</tr>
<tr>
<td>seed</td>
<td>-2121776215</td>
<td>Random number seed</td>
</tr>
<tr>
<td>num_class</td>
<td>3</td>
<td>Number of output classes</td>
</tr>
<tr>
<td>min_child_weight</td>
<td>1</td>
<td>Min. number of samples to generate a split</td>
</tr>
<tr>
<td>max_depth</td>
<td>6</td>
<td>Maximum levels of the tree</td>
</tr>
<tr>
<td>colsample_bytree</td>
<td>1</td>
<td>Fraction of each column sampled for splitting at each new tree</td>
</tr>
<tr>
<td>lambda</td>
<td>0</td>
<td>Convergence parameter</td>
</tr>
<tr>
<td>gamma</td>
<td>0</td>
<td>Convergence parameter</td>
</tr>
<tr>
<td>alpha</td>
<td>0</td>
<td>Convergence parameter</td>
</tr>
<tr>
<td>booster</td>
<td>gbtree</td>
<td>Type of boosting algorithm</td>
</tr>
<tr>
<td>grow_policy</td>
<td>lossguide</td>
<td>Method for growing the tree. Loss guide is growing out, instead of deeper</td>
</tr>
<tr>
<td>nround</td>
<td>1000</td>
<td>Number of individual trees in the ensemble</td>
</tr>
<tr>
<td>max_bins</td>
<td>256</td>
<td>Max number of bins to divide the sample into, using in tree growth</td>
</tr>
<tr>
<td>subsample</td>
<td>1</td>
<td>Fraction of column to use for sampling</td>
</tr>
</tbody>
</table>
Outcomes of the extreme gradient boosted tree model are specified in terms of confusion matrices and accuracy of classification. The use of the H2O.ai framework provides useful output for the interpretability of the model results, including confusion matrices for cross validation of the dataset. Table 7.20 shows the confusion matrix for the model fit to the training dataset, which demonstrates a significant improvement over other model types using to process the DRD. Both precision and recall scores show excellent ability to segment the data into the proper classes, with approximately > 90% precision for each readmission class. Table 7.21 shows the $F_1$-scores for the training model, which far exceed the results of other models presented previously; however, these results are misleading, as they represent only the model accuracy on the training data, which is only a single sampling of the possible values present in the DRD. Crossvalidation is required to test the ability of the model to classify data on different subsets of the DRD.

Table 7.20: Confusion matrix produced by H2O.ai framework on training set. Matrix rows: predicted class; Columns: actual class

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th>Actual sums</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>3660</td>
<td>296</td>
</tr>
<tr>
<td></td>
<td>&gt;30</td>
<td>157</td>
<td>2590</td>
</tr>
<tr>
<td></td>
<td>&lt;30</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Predicted sums</td>
<td>3818</td>
<td>2886</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>95.86%</td>
<td>89.74%</td>
</tr>
</tbody>
</table>

Table 7.21: $F_1$-scores for the 3-class XGB model on the training data.

<table>
<thead>
<tr>
<th>Model</th>
<th>Data classes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td>XGB</td>
<td>0.93</td>
</tr>
</tbody>
</table>

Table 7.22 shows the confusion matrix for the model after being processed via crossvalidation, which shows a large decline in accuracy from that of the training
Precision and recall scores are reduced to levels similar to the Bayesian or neural network methods, and the model once again shows evidence of insensitivity to readmission times, “<30” and “>30”, as the precision and recall of the “<30” class are 20.96% and 7.29%, respectively. Table 7.23 shows the $F_1$-scores for the cross-validated model, which are on par with those of other models used in this section. Readmission classes “No”, “>30”, and “<30” present $F_1$-scores of 0.62, 0.47, and 0.11, respectively. These results indicate that the dataset remains inseparable to a high degree of accuracy, and may require the addition of patient lifestyle related variables in order to better classify readmission statistics. The inability of the model to adequately segment the data under cross validation indicates an overfitting to the training data. Notice that the XGB method does perform better than all previous methods, in both two-class and three-class models. This improvement shows the applicability and effectiveness of the XGB method for use on complex data sets, including those that are difficult to separate.

**Table 7.22: Confusion matrix produced by H2O.ai framework on crossvalidation set.**

Matrix rows: predicted class; Columns: actual class

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th>Actual sums</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>2561</td>
<td>1405</td>
<td>482</td>
</tr>
<tr>
<td>&gt;30</td>
<td>128</td>
<td>1346</td>
<td>408</td>
</tr>
<tr>
<td>&lt;30</td>
<td>129</td>
<td>135</td>
<td>70</td>
</tr>
<tr>
<td>Predicted sums</td>
<td>3818</td>
<td>2886</td>
<td>960</td>
</tr>
<tr>
<td>Recall</td>
<td>67.08%</td>
<td>46.64%</td>
<td>7.29%</td>
</tr>
</tbody>
</table>

**Table 7.23: $F_1$-scores for the 3-class XGB model after crossvalidation.**

<table>
<thead>
<tr>
<th>Model</th>
<th>Data classes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td>XGB</td>
<td>0.62</td>
</tr>
</tbody>
</table>
The use of extreme gradient boosted trees result in more positive outcomes than previous models; however, the crossvalidation accuracies and confusion matrices lead us to believe the model is overfitting the dataset, and thus still remains unable to classify the general dataset with a high degree of accuracy. The results show opportunity to improve the model and potentially reduce overfitting in future development. Adjustment of the model parameters or the addition of patient lifestyle data may greatly improve model outcomes. The utility of extreme gradient boosting shows promise that this dataset contains some informative features that can be used to classify the dataset, however these features may not provide enough specificity to sufficiently segment the data. Based on the previous results of other models, as well as the reduced performance under crossvalidation, the dataset may not contain relevant information for the segmentation of patient readmission times. Readmission may rather correspond to some other variable not present in the dataset, such as those related to patient nutrition and other factors not recorded within the clinical environment.
8.1 Concluding Remarks

This work evaluated the relevancy and applicability of a variety of machine learning methods on three different datasets found in the healthcare field. Each dataset represented a type and style of data that may be analyzed to improve operational efficiency, improve patient health, or discover trends that would have otherwise gone unnoticed. The goal of this work was to apply machine learning methods to each dataset and evaluate their effectiveness and potential for use in optimization and management of clinical processes and operations.

The three datasets studied each contributed in particular ways to the work, providing an overview of the kinds of data found in healthcare, and how one might analyze these using ML methods. Although there are many applicable ML methods that could have been applied to each dataset, the ones used represent a sample of different methods and applications for analysis. This work focuses on popular methods, including neural networks, Bayesian, and decision tree models. Each model type
presents different advantages and disadvantages in use, each of which is discussed in the context of the dataset being analyzed.

**Neural Networks.** In the context of this work, neural network models were used to analyze all three datasets studied, with varying results. Working with the smaller and more conventional Pima Indians dataset, NN models perform relatively well compared to other published results on the same dataset\(^7\). The model is designed in a general way and was not highly optimized for the dataset, as this work surrounds the use of ML methods for rapid analysis of a variety of datasets, which require generalized methods and model structures in order to be applied in various settings. The reason for this decision lies in the idea that for an unknown dataset, optimization is a secondary step to initial model development, and may be performed if initial results lend to positive outcomes and predictive accuracy. The neural network model used on the Pima Indians dataset resulted in a maximum prediction accuracy of 81.77\% using the complete dataset, and a maximum of 81.61\% prediction accuracy on the preprocessed dataset, which may be improved with future parameter optimization and network architecture modification, such as the addition of dense layers, input normalization layers, or alternative and novel layers that are custom to the data types being processes. Other published works\(^7\) using alternative models on this dataset reveal that neural networks are not highest performing method, but instead the use of ensemble methods that combine hierarchal K-means clustering and support vector machines (SVM) produced the best classification accuracy on the PIDD\(^7\). This work identified potential benefits and caveats of using a non-linear model like feed-forward neural networks, for the rapid analysis of a small, non-time-dependent data. The relative simplicity of the model and its ability to classify the dataset with approximately 81\% accuracy without extensive optimization shows that neural networks are a powerful tool for predictive analysis and classifi-
The use of convolutional neural networks (CNN) has been widespread in the field of image classification and segregation. In this work, we combine the power of image handling in CNNs with time-series analysis to produce a useful model for the classification of EEG data. The model developed for use on the alcoholism EEG dataset used a medium-deep, classic network architecture found in early CNN image classification models, fed with EEG data that was converted from time series into a Markov transition field image. The model performed very well as a supervised classifier, achieving classification accuracies between 91.41% and 98.82% on unseen data. The method was proposed as an alternative to conventional time-series analysis techniques, which are typically used to predict future values of a time-series rather than define their class of origin. In addition, the high classification accuracy presented by this model demonstrates a potential for the use of out-of-scope models, such as the use of a CNN for time series analysis, on other datasets. CNN models show a capacity for feature segmentation and extraction that is unlike traditional neural network models, and presents an opportunity for exploration of alternative uses on both time-dependent and time-independent datasets.

In contrast to the Pima Indians dataset, the use of neural network (NN) models on the much larger diabetes readmission dataset demonstrated some drawbacks that were not expected upon initial analysis of the data. We found that the NN model, when fed a preprocessed but nearly complete readmission dataset performed poorly. The model was composed of embedding layers to handle categorical data, dense feed-forward layers for non-linear segmentation, and concluded with a discrete classification layer (SoftMax) for readmission class prediction. Each layer parameter was tuned and adjusted to maximize performance, but the model never learned to segment data in a meaningful way. The lack of high accuracy of prediction indicates
that the dataset may contained an overwhelming number of dependencies, or that the information contained was insufficient to develop a consistent classifier model for the three output classes. The maximum prediction accuracy on the DRD was 62.17%.

The use of NNs for analysis of different types of datasets showed great promise as a general use model that can be applied nearly universally to a variety of data types. Some drawbacks are noted including the training times required to fit the models, as well as the extensive preprocessing required if a dataset is non-conformant to the continuous variables requirements of neural networks. In the case of the CNN, preprocessing was less intensive, but the methods used to encode the time series data require input regularity and thus may induce more requisite preprocessing if data is of variable length or missing values. However, the applicability and popularity of these methods demonstrates their utility in machine learning.

**Bayesian and Markovian Methods.** This work utilized Bayesian and/or Markovian methods for analysis of all three datasets. Bayesian methods, as well as Markovian methods, rely on Bayesian statistics for both supervised and unsupervised prediction and classification. The methods are based on separability of data into distinct regions, as represented by probability distributions. These techniques perform better when underlying information governing the process is known, such as the types of distributions or frequency of occurrence within a process. However, we show that these models are powerful unsupervised classifiers that reveal useful metrics about dataset based solely on the values of the data.

We used supervised (labeled) Bayesian techniques for the analysis of the Pima Indians dataset using two types of classifiers: (a) naive Bayes, and (b) Bayes. Analysis of the dataset prior to model development revealed that the columns of the data were represented by a mixture of normal distributions and exponential distributions,
based on the histograms of the values in the columns. This prior knowledge enabled
the development of a mixed naive model, improving classification performance in-
herently, without adjustment or additional preprocessing of the dataset. Using the
more complex Bayes classifier, we found that predictive accuracy is improved over
that of the naive model, but only on the cleansed dataset. This is likely due to the
underlying data being composed of truly mixed distributions, preventing the multi-
variate Gaussian distributions of the Bayes classifier to accurately capture the data.
Each method showed acuity in classifying the Pima Indians dataset, and resulted
in similar classification accuracy of that in the literature\cite{72} and of other techniques
studied in this work.

The alcoholism EEG dataset presented an unsupervised learning challenge that
allowed us to uncover the power of Bayesian and Markovian analysis. Although
the data was labeled according to subject type, we were able to extract the number
of subject classes from the unlabeled data, as well as analyze the individual sub-
ject samples in the dataset for interesting characteristics. We analyzed the dataset
in two ways: (a) in an unlabeled fashion to determine the number of unique sub-
ject types present, and (b) on a subject-type basis to determine interesting modes of
data present in the data of each subject type. Using a nested hidden Markov model
(HMM), the subject classes were segregated into two classes, matching the original
dataset and demonstrating the ability of HMMs to segregate a dataset based on the
unique characteristics captured by the components of the model. We also showed the
presence of a number of hidden states within each group of subjects that described
unique features of the group, and may be used to identify subjects based on rapid
analysis of their EEG data.

Bayesian analysis of the diabetes readmission dataset showed little acuity for
separation or prediction of subject classes based on the labeled data, but did identify
alternative groupings of data that may represent some other underlying phenomena. Using both naive Bayes and Bayes classifiers, the preprocessed dataset was fit to each model and scored for accuracy, resulting in low accuracy scores across each output class. We attribute the inability of the model to capture the dataset accurately to the lack of discernable values across each output class.

Bayesian and Markovian methods demonstrate a powerful ability to segregate data based on its underlying representations via probability analysis. Under many conditions, the use of these methods provide excellent data segregation and classification results. We also note that if a dataset is highly complex, these models may not provide generalized performance as compared to more organized datasets.

**Decision Tree Methods.** This work evaluated the use of decision tree methods, specifically gradient boosting methods, on two of the three datasets. Decision trees offer utility as a classifier and segmenter on both linear and non-linear data, generally require little preprocessing of data, and provide highly interpretable analysis of feature importance within a dataset.

Analysis of the Pima Indians dataset using gradient boosted trees showed similar classification accuracy to both the neural network model and the Bayesian models. Because decision trees require per-dataset parameter optimization, their use as a general first step is often limited to the determination of feature importance for other models. This work found that using default parameters of the model resulted in a similar classification accuracy as that of the optimized parameters, indicating that optimization is critical only in certain cases. The model itself did perform in line with literature data\cite{72], and required little input preprocessing, except for the conversion of categorical variables into a numerical representation.

Analysis of the diabetes readmission dataset was performed using extreme gradient boosted trees (XGB), a variant of the gradient boosted tree model used to pro-
cess the Pima Indians dataset. Much like the other methods applied to the diabetes readmission dataset, the results of the XGB analysis yielded low classification accuracy. Although we see a slight increase in accuracy using the XGB model, the overall results of any method applied to the diabetes dataset tended to be poor. Again, we attribute this to the complexity and potentially uncorrelated nature of the predictive variables to the labeled output classes.

In general, decision tree models are a useful tool for classification tasks in machine learning. Their application is particularly well suited for datasets that contain larger numbers of output classes, or for regression of a dataset. The nature of decision trees breaks apart data into branches and leaves, classifying them as the tree becomes deeper. If the data presented requires too many splits before reaching an output class, the results may be substandard in quality. However, the use of these models as a feature selector for the development of other models may be of more utility.

\subsection*{8.2 Future Work}

The use of machine learning methods for rapid analysis of arbitrary datasets is a great challenge, as no model is perfect for every task. This work shows that datasets that contain highly structured and complete data are generally better suited to automated analysis, with less structured data requiring more preprocessing and human analysis. Future work includes developing and refining the techniques to enable more autonomous data extraction.

Neural networks are a particular interest for future study, as the structure of the model and tuning of parameters may lead to better performance in extraction of data dependencies. It would be of interest to study the conditions under which a neural network is able to learn features in a more natural or cohesive way, with the
development and fine tuning of interconnected and complex network structures. The use of feature engineering toolkits such as H2O.ai and FeatureTools, and methods for preprocessing of data such as encoding and embedding, may be used as a means to merge columns of a dataset to produce better predictors of output data. The use of convolutional neural networks as a general purpose feature extractor is also of interest, as the layers of these model types are designed to extract relevant features of input images so as to classify and regress datasets. The combinations of these methods may be a novel means for defining useful features within a dataset that can be used as input variables in other models.

The application of Bayesian and Markovian techniques to more complex datasets, in an effort to develop novel methods for unsupervised learning and analysis of time-invariant and time-dependent data, has significant potential for use in unsupervised environments, where data labels may be impossible to generate. The power of Bayesian statistical analysis has been shown in many fields, and may be used in combination with other model types studied in this work to form a hybrid model capable of identifying hidden structures and features of a dataset.

The study of diabetes readmission proved that many real-world datasets are filled with non-ideal relationships and potentially lack information to efficiently develop models to predict and classify outcomes. We find that the blind use of popular and well documented machine learning techniques is often not enough to handle this type of dataset, and as a result requires extensive research and development of new methods and models capable of handling these types of data sets.
References


[22] Orion Health | From Integration to Precision Medicine | Orion Health. 16


[31] Marquette Medical Systems. 25

[32] Intrumentarium. 25
[33] iPath. 25

[34] Philips. IntelliVue Clinical Information Portfolio Information. 25

[35] Agilent Technologies Healthcare Solutions Group. 25

[36] Witt Biomedical. 25

[37] Philips - IntelliSpace Event Management. 25

[38] Tomcat Systems Ltd. 25

[39] VISICU. 25


[55] Francois Chollet. Building Autoencoders in Keras. 50


[57] David Arthur and Sergei Vassilvitskii. k-means++: The Advantages of Careful Seeding. 2006. 52


[70] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. Technical report. 78
[71] Stephen D Bay, Dennis Kibler, Michael J Pazzani, and Padhraic Smyth. The UCI KDD Archive of Large Data Sets for Data Mining Research and Experimentation. Technical report. 80, 82


[77] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-Validation. Technical report. 98


[80] Wikipedia. Softmax function. 128

Appendix A

Data and Code List

Here listed are files containing scripts and raw data used to generate models, figures, and tables present in this thesis.

A.1 Pima Indians Dataset

<table>
<thead>
<tr>
<th>Filename</th>
<th>Location</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataImporter.py</td>
<td>.//*</td>
<td>Script</td>
<td>Script for importing datasets</td>
</tr>
<tr>
<td>pima_diabetes.csv</td>
<td>./NN</td>
<td>Data</td>
<td>Raw PIDD data set</td>
</tr>
<tr>
<td>best_model_<em>_</em><em>*</em>.hdf5</td>
<td>./NN</td>
<td>Data</td>
<td>Model parameters for NN. Format: name_regime_LR_CV.hdf5</td>
</tr>
<tr>
<td>pima_model_ffnn.py</td>
<td>./NN</td>
<td>Script</td>
<td>Model development for both NN and BN analysis of PIDD</td>
</tr>
<tr>
<td>pima_decision_tree.py</td>
<td>./DT</td>
<td>Script</td>
<td>Model development for DT analysis of PIDD</td>
</tr>
<tr>
<td>best_params_*_.pkl</td>
<td>./DT</td>
<td>Data</td>
<td>Best model parameters for each pre-processing regime of PIDD</td>
</tr>
</tbody>
</table>
## A.2 Alcoholism EEG Dataset

<table>
<thead>
<tr>
<th>Filename</th>
<th>Location</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataImporter.py</td>
<td>/*</td>
<td>Script</td>
<td>Script for importing datasets</td>
</tr>
<tr>
<td>control/alcoholic</td>
<td>/*</td>
<td>Folder</td>
<td>Data folder containing the single stimulus data of AED</td>
</tr>
<tr>
<td>control/alcoholic_m</td>
<td>/*</td>
<td>Folder</td>
<td>Data folder containing the matched stimulus data of AED</td>
</tr>
<tr>
<td>control/alcoholic_n</td>
<td>/*</td>
<td>Folder</td>
<td>Data folder containing the non-matched stimulus data of AED</td>
</tr>
<tr>
<td>eeg_model_cnn.py</td>
<td>./CNN</td>
<td>Script</td>
<td>Script for development of CNN model on AED dataset</td>
</tr>
<tr>
<td>best_model_*.hdf5</td>
<td>./CNN</td>
<td>Data</td>
<td>Best model parameters for each subset of AED data</td>
</tr>
<tr>
<td>trainHistoryDict_*.pickle</td>
<td>./CNN</td>
<td>Data</td>
<td>Best NN model parameters for each stimulus response of AED</td>
</tr>
<tr>
<td>eeg_model.py</td>
<td>./HMM</td>
<td>Script</td>
<td>Script for development of HMM model of AED dataset</td>
</tr>
<tr>
<td>figure_gen.py</td>
<td>./HMM</td>
<td>Script</td>
<td>Script to generate the state count plots of the HMM analysis of the AED</td>
</tr>
<tr>
<td>raw_counts_*_states.npy</td>
<td>./HMM</td>
<td>Data</td>
<td>Raw state count data for the respective number of states used in the model</td>
</tr>
<tr>
<td>smoothed_counts_*_states.npy</td>
<td>./HMM</td>
<td>Data</td>
<td>LOESS smoothed state count data for the respective number of states used in the model</td>
</tr>
</tbody>
</table>
### A.3 Diabetes Readmission Dataset

<table>
<thead>
<tr>
<th>Filename</th>
<th>Location</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataImporter.py</td>
<td>./</td>
<td>Script</td>
<td>Script for importing datasets</td>
</tr>
<tr>
<td>diabetic_data.csv</td>
<td>./</td>
<td>Data</td>
<td>Raw DRD dataset (non-numeric)</td>
</tr>
<tr>
<td>diabetic_data_numeric.csv</td>
<td>./</td>
<td>Data</td>
<td>Numerically encoded DRD dataset</td>
</tr>
<tr>
<td>diabetic_model_keras.py</td>
<td>./NN</td>
<td>Script</td>
<td>Script used to develop the NN model on the DRD</td>
</tr>
<tr>
<td>best_model.hdf5</td>
<td>./NN</td>
<td>Data</td>
<td>Best model parameters for NN model of the DRD</td>
</tr>
<tr>
<td>trainHistoryDict.pickle</td>
<td>./NN</td>
<td>Data</td>
<td>Training history of the NN model of the DRD</td>
</tr>
<tr>
<td>diabetic_model_bayes.py</td>
<td>./Bayes</td>
<td>Script</td>
<td>Script used to develop Bayesian models on the DRD</td>
</tr>
</tbody>
</table>