12-2019

Extracting Patterns in Medical Claims Data for Predicting Opioid Overdose

Ryan Sanders
University of Arkansas, Fayetteville

Extracting Patterns in Medical Claims Data for Predicting Opioid Overdose

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

by

Ryan Parker Sanders
University of Arkansas
Bachelor of Science in Industrial Engineering, 2018

December 2019
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

W. Art Chaovalitwongse, Ph.D.
Thesis Co-Director

Shengfan Zhang, Ph.D.
Thesis Co-Director

Bradley C. Martin, Ph.D.
Committee Member

Chase Rainwater, Ph.D.
Committee Member

Abstract

The goal of this project is to develop an efficient methodology for extracting features from time-dependent variables in transaction data. Transaction data is collected at varying time intervals making feature extraction more difficult. Unsupervised representational learning techniques are investigated, and the results compared with those from other feature engineering techniques. A successful methodology provides features that improve the accuracy of any machine learning technique. This methodology is then applied to insurance claims data in order to find features to predict whether a patient is at risk of overdosing on opioids. This data covers prescription, inpatient, and outpatient transactions. Features created are input to recurrent neural networks with long short-term memory cells. Hyperparameters are found through Bayesian optimization. Validation data features are reduced using weights from the best model and compared against those found using unsupervised learning techniques in other classifiers.

Contents

1. Introduction

Opioids are an effective pain-management tool, and can be used to treat a variety of conditions with great success [1]. Unfortunately, they are not always taken appropriately, and abuse of opioids can lead to adverse, many times fatal, events [2]. Opioids are highly addictive, and incorrect prescription of opioids has been a driving factor in the national increase of overdose-related deaths in the past 20 years [3]. In 2017 alone there were over 47,000 overdose deaths in the U.S. involving opioid [4]. Of those prescribed opioids for chronic pain, an estimated 21 - 29% misuse them and 8 - 12% develop an opioid use disorder [5]. 4 - 6% of those who misuse prescription opioids transition to heroin and 80% of heroin users first misused opioids [6].

Over the past few years, several changes have been made to the control of prescription opioids that have proven effective in reducing the number of fatal overdoses. The Centers for Disease Control and Prevention released a new, more restrictive set of guidelines for prescribing opioids [7]. This has been at the cost, however, of an over-restriction of the drug, which is an effective pain-management tool when prescribed appropriately [8]. The opioid epidemic has peaked, but the battle with opioids is not yet over.

1.1 Research Motivation

Much research is available concerning which populations are most vulnerable to opioid addiction. Risk factors are often measured for patients by using one of many screening tools before prescribing them opioids. These are based on multivariate statistics on the static attributes of opioid abusers such as age and sex. It is useful to prescribers (physicians or nurse practitioners) to create risk screening tools based on these statistical analyses. These tools are summarized by Lawrence et al. [9].

While many doctors use screening tools before prescribing opioids to a patient, there are not any tools available to monitor patients after treatment begins. A patient who is not classified as vulnerable according a risk tool may become vulnerable over time. Monitoring for patient state changes is currently done only by the prescriber based on their knowledge

and training. There is inevitably a degree of subjectivity in their judgement. In addition, risk tools do not include the patient's full history of prescriptions and doctors visits. There may patterns in a patient's history that can be exploited to predict their risk of addiction on an individual basis.

Many datasets have been underutilized in combating the opioid epidemic from a data perspective. These include Electronic Health Records (EHR), Prescription Drug Monitoring Programs (PDMP), and Insurance Claims data, all of which are transaction datasets. Transaction data can be defined as any temporal sequence where a number of entities are recorded performing different events. Transaction data is much more utilized in areas other than healthcare, but more research is needed in order to improve usability. In the field of data mining, association rules and sequence matching have been the target of most algorithms utilizing transaction data [10]. Extraction of more informative features to use in prediction models is still a budding research area [11].

Research has been done to predict other medical responses fully utilizing this type of transaction data, but there is a dearth of research using similar methodologies for opioid-related responses. Many times in the medical field, features are created using expert knowledge. While this does not yield bad predictions, better may be obtained by exploited the full complexity of the data. Time-dependent features created are usually relegated to small window of the entire data. In addition, features are only created using variables of interest. There may be complexities in the data that are not known to be related to the response in the medical literature. Fully utilizing the time-dependent aspect of the data will allow complex interactions in the data to be exploited for better predictions.

1.2 Research Goals

The goals of this research are to (1) develop a framework for feature extraction from transaction data, (2) explore deep learning techniques using those extracted features for predicting patients as at risk of adverse, opioid-related events (ORE).

These goals will be accomplished through experimenting with various strategies transforming raw data into usable features in combination with different modeling strategies. This research focuses on deep learning techniques for both feature extraction and prediction. The best combination of techniques found will be used to develop a framework for feature extraction and prediction using transaction data.

1.3 Thesis Organization

Chapter 2 begins with a review of literature related to the prediction of ORE for individual patients. These include studies that use both point-in-time, or cross-sectional data, as well as those using time-dependent, or panel data. The application of the methodologies developed in these studies is then discussed.

The literature review continues with studies related to prediction of events from transaction data that use feature extraction and deep learning techniques. Although studies using healthcare-related data are primarily looked at, studies using data from other fields are investigated as well. The focus is on prediction using machine learning and deep learning techniques, as well as feature extraction.

Chapter 3 describes the methodology for data processing, feature extraction, and prediction. The data subsection includes data sources, cohort selection, response identification, as well as transformations used to prepare the data as an input to train different models. Transformation is especially important due to the research goal of extracting appropriate features from the data. In order to get the most informative features, different time aggregation windows with varying sequence lengths are tested

Prediction methodology begins with using those extracted features in a recurrent neural network with long short-term memory cells to predict ORE. Hyperparameters of each model are tuned using Bayesian optimization. To improve predictions, autoencoders are used to find better representations of the original inputs. Features from the autoencoders are then input into several different prediction models for comparison. These are compared to using the learned weights from the neural networks as input for the same models.

Results are presented, compared, and discussed. Also discussed are potential applications of this methodology, medical implications of the results, and limitations of this research.

2. Literature Review

This chapter focuses on a review of the methodology used in this research. The first section of this chapter covers previous work related to the prediction of ORE from transaction data. These works are divided by prediction methodology and data source. The second section covers the methodologies used in this research. It begins with a description of deep learning and continues to describe research using deep learning models with healthcare-related transaction data. Finally, research using representational learning to better represent healthcare data is discussed.

2.1 Prediction of Opioid-Related Adverse Events

EHR, PDMP, and Claims data have all been used with varying degrees of success to predict the risk of a patient of having an opioid-related adverse event. Methodologies for prediction patient-level adverse events range from simple statistical tests to advanced deep learning techniques. The following review is broken up by the primary prediction methodology used in each study.

Logistic regression (LR) is a very popular methodology in the literature and has been applied to many different datasets for ORE prediction as reviewed by Turk et al. [12]. The value of LR lies in its explanatory power. In these papers, the primary use of the model is to learn which variables are most related to a positive response as opposed to predicting which patients will have an ORE.

The Cox proportional hazards (CPH) model is one of the most predominately used models in individualized patient predictions. The model is in essence a regression of survival time on the patient variables. Its popularity is also due to the explanatory nature of the model. For each feature used to build the model, the proportional likelihood that the level of that feature will correspond to a response is calculated. Several papers have used this methodology to predict adverse events related to opioids.

Decision trees are another popular classifier. Both the random forest (RF) and gradient boosting (GB) algorithms have been used for ORE prediction. Neural networks (NN) have

also been used, and these are detailed in the next section. A summary of ORE prediction literature is shown below in Table 1.

Table 1. Prediction of ORE from Transaction Data

Author	Data Type	Data Source	Sample Size	Positive Responses	Number of Variables	Classifier
Zedler et al. [13]	EHR	Veteran's Health Administration	1,877,841	817	15	LR
Karhade et al. [14]	EHR	5 Hospitals	5,507	345	4	LR
Cauley et al. [15]	EHR	Nationwide Inpatient Sample database	11,317,958	9,458	11	LR
Ali et al. [16]	Medicare; Private Claims	IBM MarketScan	4,535,623; 1,604,143	31,163; 44,994		LR
Chang et al. [17]	PDMP	Maryland	25,487	827	25	LR
Geissert et al. [18]	PDMP	Oregon	879,402	1,409	6	LR
Levin et al. [19]	PDMP	New York	881,558	1,118		LR
Cochran et al. [20]	Private Claims	Thomson Reuters MarketScan Commercial Claims database	284,1793	2,913	38	LR
Sun et al. [21]	Private Claims	Optum Clinformatics claims database	5,293,880		40	LR
Liang et al. [22]	Private Claims	Aetna Health Maintenance Program	206,869	1,386		LR

Table 1 continued

Author	Data Type	Data Source	Sample Size	Positive Responses	Number of Variables	Classifier
Rice et al. [23]	Private Claims	Ingenix Employer Solutions	821,916	6,380		LR
Lo et al. [24]	Medicare Claims		560,057	3,188	268	LR, GB, NN
Calcaterra et al. [25]	EHR	Denver Health Medical Center	27,705	1,457	13	LR, RF
Shah et al. [26]	Private Claims	IMS Lifelink+ database	1,353,902	33,019	6	CP
Glanz et al. [27]	Private Claims	Kaiser Permanente Colorado health plan	42,828	121	9	CP
Li et al. [28]	Private Claims	IMS LifeLink PharMetrics PlusTM database	1,246,642	2,274	278	CP
Ellis et al. [29]	EHR	Mount Sinai Medical Center	716,533	9,518		RF
Che et al. [30]	EHR	Rochester Epidemiology Project	102,166	749		NN

2.2 Introduction to Deep Learning

Deep Learning is a catch-all term used to describe many different types of neural networks. This section provides a brief description of the way neural networks are constructed and trained. For further reading, please refer to *Deep Learning* by Goodfellow et al. [11]. A neural network, at its essence, is simple a graph of nodes and weights used to transform an input to a desired output. They are usually structured with layers of nodes so that every

node in one layer is connected to every node in the layers on either side. The connections between these nodes have are a assigned a weight that changes in as the network is trained.

Training a neural network begins with initializing the weights. This is most typically done using random values from a normal distribution, but sometimes different functions are used. The inputs are then passed through the graph. At each node, the input is passed through an activation function that adds some nonlinearity. The most commonly used activation function is the sigmoid function. These are prone to forcing values to become either really big of really small in large networks, or what is commonly known as the vanishing and exploding gradient problem. Using a rectified linear unit (ReLU) activation has recently gained popularity to mitigate this issue. As values pass from node to node they are multiplied by the weights between those nodes. At the last layer, the output layer, the values are compared to the desired output using some loss function. The loss function measures the difference between the original input and the network output. The goal is to minimize that different for all training examples.

The weights of the network are adjusted to decrease the value of the loss function through a process known as backpropagation. To do this, the first derivative of the loss function is taken with respect to each of the output nodes. The derivative of each of those function is taken with respect to each of the nodes connected to that node from the previous layer. Then for each of those nodes, the process is repeated again until the first derivative is found for all possible paths through the graph. The weights between each node are adjusted using the corresponding derivative of the loss function so that the value of the loss function is decreased. How much the weights get adjusted each iteration is referred to as the learning rate.

Different optimizers perform backpropagation in different ways. Gradient descent is the easiest to understand. In order to minimize the loss function, we simply want to find the gradient (or first derivative with respect to each variable) that decreases the function the fastest. However, this approach is susceptible to getting stuck in local minimums. Local

minimums are relatively low compared to the surrounding points, but not the lowest point the loss function can be in general. An alternative, stochastic gradient descent, adds an element of randomness to the search in order to avoid the local minimum issue. The most popular algorithm to use in deep learning research is the Adam optimizer. It takes less time to train than stochastic gradient descent while achieving similar performance.

The forward and back propagation process is done for each observation in the training data. One pass through all observations is referred to as an epoch. Networks are trained using several epochs until performance is deemed adequate or shows no sign of improvement. To speed up training, batches of observations are used to train. A batch of observations are forward propagated through the network and the average of the loss function calculated for the entire batch before performing backpropagation.

2.3 Individualized Healthcare Predictions Using Deep Learning

Neural networks are used in many other healthcare-related event predictions besides OREs. The most relevant to this research are those which utilize to the most extent a similar type of transaction type data. By utilizing the temporal nature of this type of data, the following studies were able to make much better predictions than those predicting based on static features. They also make use of the entirety of information available in the data by using representational learning techniques.

2.3.1 Types of Networks Used

There are many different approaches to take when trying to classify patients. Neural networks have the most potential according to the literature, so we focus on investigating them [31]. Neural networks have been around for a very long time, but only in the last decade become popular due to increased computing power and data collected. The basic concept of neural networks is briefly explained in this section

The most basic type of a neural network is the feed-forward neural network (FNN). They are composed of an input layer (X), hidden layers (h), and an output layer (y) shown

below in Figure 1. Every cell is connected to every cell in the next layer. The weights for these connections are trained based on the gradient of the loss function, a process called back-propagation. Non-linearities are introduced in each cell by an activation function [11].

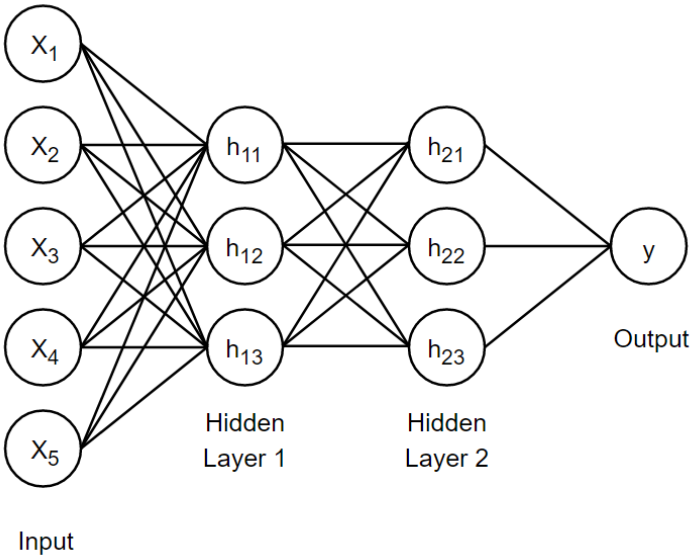

Figure 1. A two-layer FNN with one output

Recurrent Neural Networks (RNNs) are more effective to use in making predictions based on sequential inputs[32]. Unlike FNNs, they accept a sequence of inputs for each observation where information from past sequence elements is incorporated into each cell. The structure of a RNN with two layers of recurrent cells is illustrated in Figure 2. In the figure, X is the input sequence, h is a dense hidden layer, y is the output, and A are the recurrent cells. They are widely used in natural language processing tasks, but can be applied to any sequential data as they are designed to deal with long-range temporal dependencies [33].

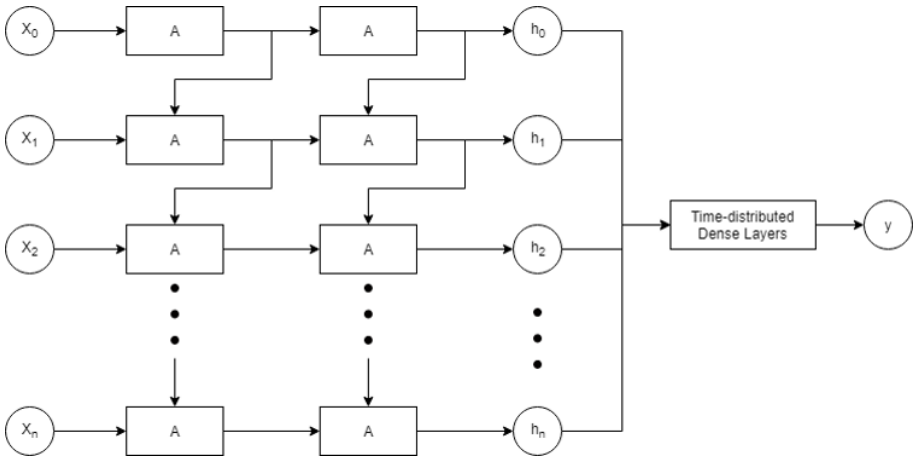

Figure 2. Structure of an RNN with two layers of recurrent cells

Training ordinary RNNs is difficult as they are prone to both vanishing and exploding gradients, but this can be remedied by using gates to restrict how much information gets passed from cell to cell [34]. There are two popular types of gated RNNs, those using gated recurrent units (GRU) [35] and those using long short-term memory cells (LSTM) [36]. The difference between the two lies in the number of gates and the exact purpose they serve. While LSTM networks have proved more effective for natural language processing tasks, GRU networks are computationally more efficient while providing comparable results [35]. Deep learning models have many hyperparameters that must be tuned. Grid search and random grid search are typically used, but Bayesian optimization can be used to fine the best combinations of hyperparameters in a more efficiently [37], [38].

2.3.2 Deep Learning for ORE Prediction

The most promising methodology in ORE prediction is deep learning. Deep learning refers to the use of neural networks with more than one hidden layer. The advantage of deep learning models is their ability to learn representations of the data with multiple levels of abstraction. The multiple hidden layers allow the model to discover complex interactions that exist between variables [32].

Che et al. [30] used both FNN and LSTM to classify patients in the Rochester Epidemiology Project (n=102,166) as either short-term, long-term, or opioid-dependent users.

To prepare the temporal data, they used one-hot encoding to categorical features and aggregated by year. They found FNN to predict better than RNN, with 90% accuracy and 62% recall. This research will use a similar approach but with more granular time windows. Another difference will be the response of ORE instead of opioid dependence.

Claims data (n=186,686) was used by Lo-Ciganic et al. [24] along with logistic regression, random forest, gradient booting machine, and feed-forward neural network to predict the risk of ORE. The neural network had the best performance of any model. They used 3 month time windows instead of the more granular windows used in this study. In addition, features were developed based on domain instead of the more naive approach of this research.

2.3.3 Other Deep Learning Uses in Healthcare Prediction

Choi et al. [39] used GRU to predict different diagnoses based on EHR data (n=263,706). The model was trained on the sequence of patient visits with all corresponding information one-hot encoded. They were able to achieve 79% recall and 64% accuracy with this model. This work was later extended [40] by using fixed-sized time windows which aggregated events instead of using events themselves to predict the occurrence of heart failure using a similar EHR dataset. Esteban et al. [41] used a novel structure to incorporate static features into an RNN network that reduced the dimensionality of the input. They found that a GRU network outperformed LSTM and ordinary RNN networks in predicting outcomes from kidney transplants using EHR data.

Nickerson et al. [42] predicted the occurrence of adverse post-operative events from time-window aggregated EHR data using both FNN and LSTM networks. The DeepCare framework proposed by Pham et al. [43] uses the sequence of admissions recorded in EHR data, classified into diagnosis events and intervention events, in an LSTM network. Several modifications to the traditional LSTM network were made, and in different experiments they predicted both diabetes and mental health related outcomes with a relatively high F-score. Razavian et al. [44] trained a LSTM on the results of lab tests recorded in claims

data (n=298,000) in order to predict the onset of 133 different diseases. To deal with class imbalance, they use a weighted log-likelihood function with changing weights for each batch.

LSTM cells were used by Lipton et al. [45] to predict the occurrence of any of 128 different diagnoses using irregularly sampled tests in EHR data from an intensive care unit. They found LSTM outperformed other models, including FNN. Their best performing model utilized an ensemble of LSTM and FNN networks. Puruchotham et al. [46] compared different machine learning models on a publicly available EHR dataset. They also found that an ensemble model of FNN for static features and RNN for time dependent features consistently yielded the best predictions.

2.3.4 Methods of Information Extraction

Many machine learning techniques have a limited ability to utilize raw data. Prepossessing to transform the raw data usually requires domain experts to manually engineer usable features. Representational learning is a set of methods that automatically learn usable features from the data. Deep learning methods of representational learning allow for complex functions to be learned from the raw data, with no domain expertise needed [32].

The DeepCare algorithm developed by Pham et al. [43] involves pooling same type features over time windows and then concatenating the different feature vectors. In this way they embed variable number of events into a continuous distributed vector space.

Skip-gram embeddings are a popular preprocessing technique for natural language processing tasks popularized by Mikolov et al. [47]. E. Choi et al. used skip-gram representations of different medical codes assigned to each patients as an input to an RNN [48], [49], [40]. Y. Choi et al. used skip-gram embedding on several different data and found that in all cases the results conceptual similarity and medical readiness of the data was improved [50].

Autoencoders (AEs) were used by Miotto et al. [51] to improve the performance of their prediction model. The best AE structure for their data was using three hidden layers with 500 nodes each. The advantage of AEs over standard dimensionality reduction techniques such

as principal component analysis or singular value decomposition is their ability to capture complex interactions between variables [33].

AEs are not the only way to develop better features with deep learning. Che et al. [52] proposed a framework for feature extraction from heterogeneous healthcare time series data using a form of greedy layerwise pretraining. This methodology, similar to that proposed by Bengio et al. [53], increases the performance of a deep network by training a model with one hidden layer, then adding a layer at a time, retraining the weights learned in the last iteration.

3. Data Processing and Prediction Methodology

This chapter begins by describing the data used, the cohort selection process, and the methods by which the data was cleaned and transformed. The data were divided into 20 distinct datasets for experiments. Each of these datasets was used to train a RF classifier, a sparse AE whose features were then used to train a RF classifier, and a RNN with LSTM cells. For the LSTM models, hyperparameter tuning was done using Bayesian optimization. A simple, denoising, and sparse AE was trained on each dataset and outputs used to train other classifiers. This research utilizes Tensorflow [54], an open-source tool for applying machine learning methodologies, using Keras [55]. All data preprocessing was completed using Python with pandas [56] and NumPy [57]. Classifiers from Scikit-learn [58] are used to compare models. Bayesian optimization for hyperparameter tuning was accomplished using GPyOpt [59].

3.1 Introduction to Data and Data Processing

This section describes the dataset used and the processing techniques to transform it. Data preprocessing is as important as, if not more important than, the prediction models tested, since the results from the models can only be as good as the data used to train them. Transaction data can be tricky to use effectively in machine learning. This section lays out a framework to use transaction data to construct inputs to any classification or regression tool.

3.1.1 Data Description

The LifeLink PharMetrics Plus database is used in this research. It contains inpatient, outpatient, and pharmacy (RX) claims for millions of unique patients enrolled in major healthcare insurance plans from 2006 to 2015. Of these patients, approximately 2.6 million received at least one opioid prescription during this time frame.

3.1.2 Patient Cohort and Response Identification

The same criteria developed by Li et al. [28] was used for selecting the cohort of patients. The cohort are those who receive their first recent opioid prescription while covered in our

dataset. The first opioid prescription for each patient is considered the index claim. Patients with no opioid prescriptions are excluded. We only consider patients who are continuously enrolled in the insurance plan 6 months before and after the index claim. Patients with cancer or in hospice care, as well as patients younger than 13 are excluded as well. The cohort consists of 1,376,760 patients. Those patients average 134.27 events recorded after their first opioid prescription over the course of 688.78 days. The largest gap between the first opioid prescription and the last event was 3464 days with 9004 events in between.

The response, OREs, are identified using ICD-9 diagnoses codes. Modeled after one of the responses used by Seal et al. [60], a patient diagnosed with one of the codes shown in Table 2 is considered to be a positive response for ORE. This differs in the response identified by Li et al. [28] by not considering patients who are only coded with a respiratory depression diagnosis. The narrower definition will yield fewer false positives at the risk of missing some OREs experienced by patients. There are 1,533 patients who exhibit an ORE by this definition in the cohort.

Table 2. ICD-9 Codes Identifying ORE

ICD-9 Code	Description
965	Poisoning by opium (alkaloids), unspecified
965.02	Poisoning by methadone
965.09	Poisoning by other opiates and related narcotics
E850.1	Accidental poisoning by methadone
E850.2	Accidental poisoning by other opiates and related narcotics
E935.1	Methadone causing adverse effects in therapeutic use
E935.2	Other opiates and related narcotics causing adverse effects in therapeutic use

3.1.3 Combining Data Sources

Inpatient, outpatient, and RX claims are recorded in separate tables, each having a distinct set of associated variables. There also exists a table of patient information. These tables are structured as a relational database where each table of events is connected to the patients table by a unique identifier. This research considers each event (claim or transaction), as an observation for a patient. As such, events in the inpatient, outpatient, and RX tables

were all combined into one table of events. Static patient information was added to this table of events as well. The tables had some overlapping variables. The final variables used for modeling are shown in Table 3. The three event tables (inpatient, outpatient, and RX) were appended together with respect to the variables they shared as shown in Figure 3. In total, they consist of 184,852,600 events. Labels across the top of the figure are the sets of variables included in the tables. A categorical variable indicating the source table of each event was added which is not shown.

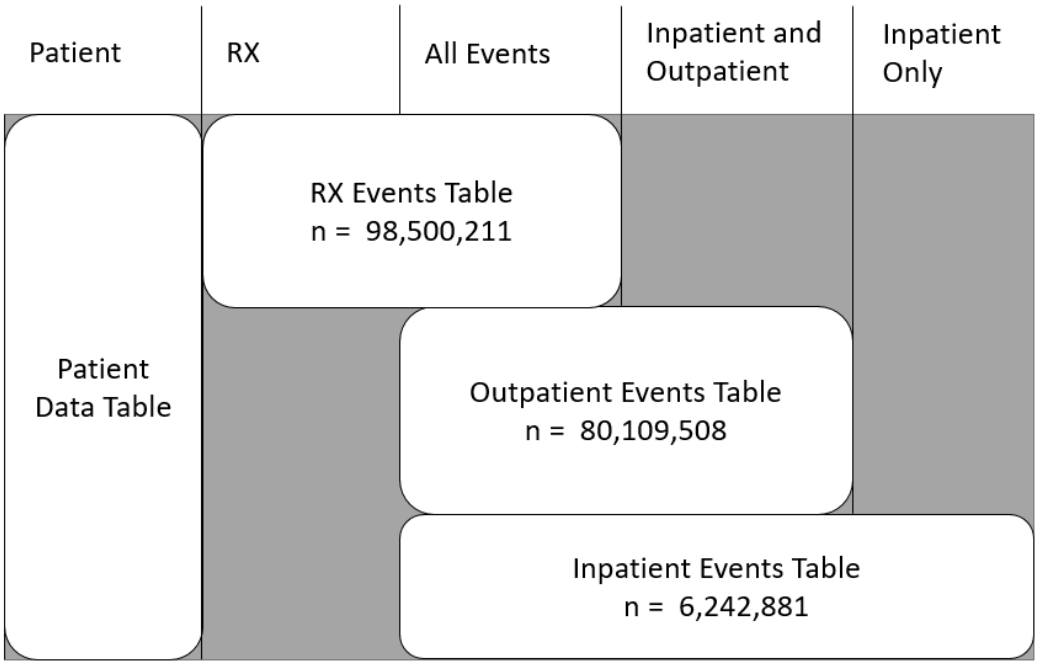

Figure 3. Combining Sources of Events

3.1.4 Data Cleaning and Preprocessing

For the selected cohort of patients, there were a few variables that were no longer needed to be included. Several were removed because there was no variation throughout between patients. For example, one variable indicated whether the claim was paid for or denied. For the given cohort every claim was paid for, so this variable was removed. Other variables with very low response rate or high redundancy with other variables were removed as well. For the categorical variables remaining, many had a large number of categories that a very small proportion of the events fell into. These categories were recoded so that any category

containing less than 1% of events were grouped into an "other" category. This was done to reduce dimensionality once one-hot encoded. Continuous variables, shown in Table 3, were scaled between 0 and 1. Scaling continuous variables is a common preprocessing step that helps avoid vanishing and exploding gradient problems [61].

Diagnosis and procedure codes for inpatient and outpatient events are both very important to a patient's history. They contain all information about conditions a patient was diagnosed with and how they were treated. There are up to 11 supplementary diagnosis codes in addition to the diagnosis code for admission. There are up to 6 procedure codes for each event. In this data, both procedures and diagnoses are coded using the ICD-9 convention. According to this condition, there are over 15,000 possible diagnosis codes and nearly 4,000 procedure codes. If these original variables were one-hot encoded it would result in more than 204,000 columns!

In order to reduce the high dimensionality of the ICD-9 code combinations, we used the Agency for Healthcare Research and Quality's Healthcare Cost and Utilization Project (HCUP) clinical classification software (CCS). This software classifies ICD-9 codes into a smaller number of categories. Diagnosis codes are mapped to 285 categories, and procedure codes to 232. The HCUP's chronic condition indicator (CCI) was also used for diagnosis codes. This is a binary variable indicating whether a diagnosis is chronic or not. HCUP's procedure classes (PC) code was added as well to indicate the severity of the procedure preformed, of which there are four categories. After recoding each of the 12 diagnosis code columns in 2 ways (CCS and CCI) and and 6 procedure code columns in 2 ways (CCS and PC), each of the 36 new columns were one-hot encoded. Columns for similar categories were then summed for each event to reduce the sparsity of the data since it does not matter which diagnosis column a diagnosis was recorded in, just that the diagnosis occurred for that event. This resulted in adding 523 columns for diagnosis and procedure codes instead of 204,000. 1010 total features resulted from data preprocessing after one-hot encoding categorical variables.

Table 3. Variable Descriptions

Variable Type	Variable	Data Type	Number of Categories
Patient	Age at first opioid claim	Continuous	
	Group or individual coverage	Categorical	3
	Sex	Categorical	3
	Enrollee relationship to patient	Categorical	6
	State	Categorical	51
Event	Event type	Categorical	3
	Days after first opioid prescription	Continuous	
	Provider type flag	Categorical	3
	Billing provider's primary specialty	Categorical	65
	Rendering provider's primary specialty	Categorical	65
	Place of service code	Categorical	9
RX	Days of supply of the prescription	Continuous	
	Drug Group (first two digits and first subset of the GPI code) classifies general drug products	Categorical	69
	Brand is trademark or generic	Categorical	2
	Prescriber's primary specialty	Categorical	63
	Dispensed as written code indicating whether or not the prescriber's instructions regarding generic substitution were followed	Categorical	4
	Route of administration - how the medication's dosage form is administered to the patient	Categorical	7
	Whether the prescription drug was paid as included in the plan's formulary at the record level	Categorical	2
	Submission type code	Categorical	5
Inpatient and Outpatient	Admitting diagnosis; ICD-9 diagnosis patient was admitted with	Categorical	~15,000
	Diagnosis codes 1–11; ICD-9 diagnosis codes of the event	Categorical	~15,000
	Procedure codes 1–6; ICD-9 procedure codes of the event	Categorical	~4,000
	Specialty of the attending provider. For non-physicians, this reflects the type of provider/facility	Categorical	56
	Specialty of the primary care physician	Categorical	43
Inpatient Only	Length of stay (only for inpatient events)	Continuous	

3.1.5 Transforming the Data to Inputs

There are multiple ways to transform our original dataset into features for prediction. The simplest way is to strictly consider a patient's transaction history as a sequence of events. This is already how the data is structured, with each transaction (event) having a number of descriptive variables. The other approach is to aggregate the sequence of events over time windows. This strategy essentially collapses the events into one observation for each time window. Time windows could be of either fixed or variable length, but for simplicity of replication with other data, this study only uses fixed-length time windows. Each patient's set of time windows ends the day before the last non-ORE event in their event sequence.

Time aggregates are calculated for each monthly, bi-weekly, weekly, and daily windows. Aggregates are found by summing both continuous variables and one-hot encoded categorical variables over each time window. One-hot encoded categorical variables are then clipped to be either 0 or 1. Clipping categorical variables reduces the amount of redundant information in the data. For example, consider a patient who has three outpatient events during a time window and receives the same diagnosis all three times. Once the patient has been diagnosed with a condition during the time window, subsequent diagnoses of the same condition should not matter. The one-hot encoded variable indicating source table (RX, Inpatient, Outpatient) was not clipped after aggregating in order to inform the model of the number of event transpiring during the time window. For example, it would be important information to know if a patient went to the pharmacy 10 times in a month. Static variables about the patient remain the same with aggregation. If a patient had no events occur during a time window, those variables are filled with 0s. There are 5 datasets being investigated: time window aggregates of monthly, bi-weekly, weekly, and daily events, as well as using the events with no aggregation.

The length of the sequence used in prediction is also important to consider. The sequence of data for every patient must be the same length. Events occurring closer to the ORE response are hypothesized to have more influence on the ORE than those events occurring

farther in the past. For this reason, the sequences are aligned by the last event for each patient, the event directly preceding an ORE for positive responses and the last event recorded for negative responses. Then, the sequence is trimmed to the desired length from the left, excluding the earlier events for patients with sequences longer than that length. This process is illustrated in Figure 4 below, where the sequence length is chose to be 4. Patients with less than 4 events are excluded. For each of time aggregate datasets, only patients having events covering the entire time window will be included in the dataset for that time window. In this experiment, sequence lengths of 5, 10, 20, and 40 observations were tested. These lengths were chosen to encompass a broad range of sequence lengths. Combining these with the 5 time aggregation methods yields a total of 20 sets of data to test.

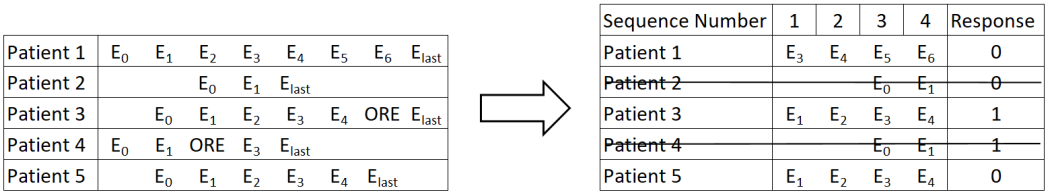

Figure 4. Visualization of Sequence Alignment and Trimming

Only transactions recorded on the day of or after the date of a patient's first opioid prescription are used. The transactions of patients with an ORE are only included up until the day before the first ORE identified. All transactions of patients without an ORE are included. Transactions are also only included for a patient's continuous enrollment period. That is, if the patient becomes dis-enrolled from the insurance plan, transactions after that time are excluded from the data.

3.1.6 Batch Processing

Given the large size of this data, special techniques had to be used for processing. There are over 184 million rows of 1010 columns in the final dataset. That is 185 billion data points. This is way to much data to be held in memory so a data chunking technique was used for processing. Data was queried in chunks from the combined events table, taking care not to

break a patient's data between chunks. Events for each patient were cropped to the specified sequence length. The min and max of continuous columns were updated and the data chunk stored in hdf5 format [62]. After the query was complete, each chunk was then reopened for processing. Continuous variables were scaled between zero and one and categorical variables were one hot encoded. All possible categories for each categorical column were hard coded so that each chunk would be one hot encoded to the same number of columns. The events for each patient were then aggregated for each time window as described in the previous subsection. Time windows with no events were filled with zeros.

3.2 Deep Learning

Each dataset is split into train, validation, and testing sets. First, a random 20% of the data was selected as the test set. The remaining data was split 80% training and 20% validation as shown in Figure 5. For each split, the data was stratified so that the same proportion of ORE patients where in each set.

In order to see how well different models work, we must start out with a simple model to get a baseline for prediction. Random forest was chosen due to its popularity of use in medical research [25], [29]. RNNs with LSTM cells were then trained for each dataset. The hyperparameters of each model are found using Bayesian optimization and the validation data. Models are then trained on both training and validation data and tested on the remaining testing data. A sparse AEs was trained, and the outputs fed into the same baseline model.

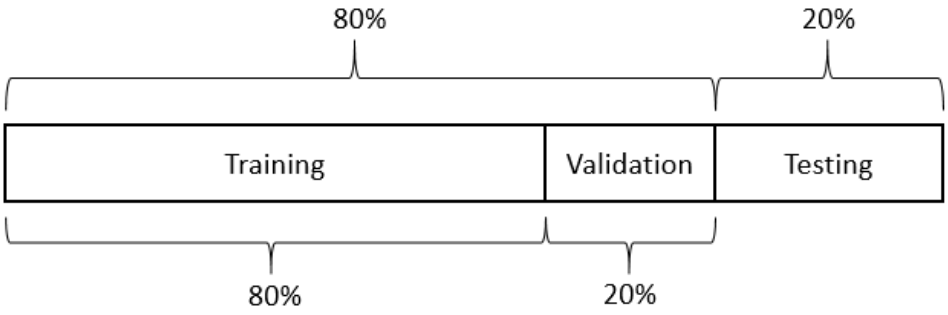

Figure 5. Visualization of Training, Validation, and Testing Split

Given the size of the data, it cannot be all kept in memory. Neural networks can be trained online and do not need to be input all the data at once. The other classifiers used are not online and all training data must be loaded into memory at one time. It is not possible to train on the entire dataset due to computational limitations. Classifiers are trained on the validation data, 16% of the entire dataset, and tested on the same holdout testing data, 20% of the entire dataset. This is done 5 times, each with a different portion of the training data so that all training observations are using in training a model.

3.2.1 Models Tested

RNN with LSTM cells will were tested using all 20 datasets. A basic structure of two layers of recurrent nodes and two layers of dense nodes was chosen. This number of layers was chosen because having more hidden layers allows the model to learn higher level abstractions from the input data [11].

To train the RNN, we used log loss, otherwise known as binary cross-entropy in Keras. Since the data is incredibly imbalanced, class weighting was used to weight the loss function [63]. Rectified linear unit (ReLU) activation functions were used as it reduced the exploding and vanishing gradient problem [11]. Dropout method was used to help prevent overfitting of the training data [64], [65]. Batch normalization is used as well [66].

Good combinations of hyperparameters for training with each dataset were found using Bayesian optimization [37], [38], [67]. Hyperparameters considered were the number of nodes in each layer, dropout rate of each layer, learning rate, batch size, and number of epochs to train for. The model is trained using the training set and tuned using the validation set in each iteration of the algorithm. Once the best hyperparameters are found, they are used to train the model on both the training and validation set and tested with the testing set that was held out.

The output of each model comes from a sigmoid function and is between 0 and 1. The threshold for which the model classifies an observation as either positive or negative is very important. For each model, the threshold that maximizes weighted accuracy metric is used

to determine the final model. Mathews correlation coefficient [68] was also considered as a metric to maximize.

3.2.2 Representation Learning

The raw inputs to the models listed above should produce fairly well performing classifiers due to the size of the raw input and the power of neural networks to find hidden dependencies in the data. The goal, however, is to create features that can be used in other classifiers that will achieve an even better performance.

Supervised representation is where observation labels are used to help create the features. One supervised feature representation technique is to use the learned weights from a neural network. After the network is trained, the prediction layer is removed and the outputs from the last hidden layer are used as the new features.

A similar unsupervised feature representation technique is to use AEs [69]. These are neural networks that are designed to encode information from the input to a lower dimension. The simplest versions have one hidden layer, but deep AEs with multiple hidden layers may be more effective. AEs can be stacked, oftentimes making them more effective [70]. The method of training stacked AEs, greedy layerwise unsupervised pretraining, is very similar to greedy layerwise supervised pretraining in that each layer of the AE is trained with weights initialized as the weights learned from training the previous layer. Unsupervised feature representation yields better results in many cases [71].

In this research, we train a sparse AE using each dataset. A sparse AE with 5 layers is used to find higher level abstractions in the data. These five layers consist of 1250, 2500, 5000, 2500, and 1250 nodes respectively. Its output has the same dimension as the original data. Our loss function to train the sparse AE is mean squared error between the inputs and outputs. The outputs from this method was tested using a random forest classifier.

4. Results

4.1 Data

The number of patients in each of the datasets tested is shown below in Table 4. These differ because of the exclusion criteria applied when creating different datasets. Patients whose sequence of events is not as long as the sequence length are excluded. Patients who have no events recorded during that time window are also excluded.

Table 4. Size of Dataset Used

	Sequence Length			
Time Aggregate	5	10	20	40
Events	1,265,087	1,309,926	1,207,803	1,005,234
Days	302,643	481,942	684,805	955,942
Weeks	904,586	1,081,934	1,205,549	1,259,576
Bi-Weeks	1,081,934	1,205,549	1,259,576	1,198,651
Months	1,213,888	1,260,371	1,183,033	879,132

4.2 Random Forest Classifier

Random forest classifiers were trained as a baseline for comparison. For each dataset, 5 models where trained using different portions of the training set. Average results models trained using each dataset are shown in Table 5. This methodology had a relatively high accuracy and area under the receiver operating characteristic curve (AUC). For most datasets, recall was poor.

Table 5. Random Forest Classifier Results

Time Aggregate	Sequence Length	Accuracy	Recall	Precision	AUC
Events	5	0.718	**0.733**	0.002	**0.725**
Events	10	0.783	0.483	0.003	0.633
Events	20	0.781	0.395	0.003	0.588
Events	40	0.702	0.392	0.002	0.547
Days	5	0.638	0.417	0.004	0.527
Days	10	0.592	0.503	0.003	0.548
Days	20	0.805	0.303	0.003	0.555
Days	40	**0.854**	0.261	0.002	0.558
Weeks	5	0.755	0.477	0.003	0.616
Weeks	10	0.781	0.405	0.002	0.593
Weeks	20	0.717	0.418	0.002	0.568
Weeks	40	0.762	0.373	0.002	0.567
Bi-Weeks	5	0.76	0.424	0.002	0.592
Bi-Weeks	10	0.789	0.355	0.002	0.572
Bi-Weeks	20	0.764	0.361	0.002	0.563
Bi-Weeks	40	0.741	0.409	0.001	0.575
Months	5	0.765	0.437	0.002	0.601
Months	10	0.837	0.28	0.002	0.559
Months	20	0.701	0.443	0.001	0.572
Months	40	0.815	0.258	0.001	0.537

4.3 Sparse Autoencoder

An AE that expands the input to 5000 nodes in order to uncover abstractions is trained. The output was fed into a random forest classifier. Results are shown below in Table 6. For each dataset, this methodology performed worse than the baseline classifier. This is a substantially worse methodology than only using RF.

Table 6. Random Forest using Sparse AE with 5000 nodes

Time Aggregate	Sequence Length	Accuracy	Recall	Precision	AUC
Events	5	0.903	**0.416**	0.003	**0.66**
Events	10	0.941	0.343	0.006	0.643
Events	20	0.966	0.212	0.007	0.589
Days	5	0.919	0.104	0.003	0.513
Days	10	0.893	0.133	0.003	0.514
Days	20	0.827	0.229	0.002	0.529
Weeks	5	0.913	0.189	0.003	0.552
Weeks	10	0.973	0.092	0.004	0.533
Weeks	20	0.986	0.048	0.004	0.518
Bi-Weeks	5	0.986	0.046	0.004	0.517
Bi-Weeks	10	0.989	0.019	0.002	0.505
Bi-Weeks	20	**0.991**	0.024	0.003	0.508
Months	5	0.961	0.093	0.003	0.528
Months	10	0.983	0.024	0.001	0.504
Months	20	**0.991**	0.015	0.002	0.503

4.4 LSTM RNN Predictions

Before performing Bayesian Optimization to tune the hyperparameters, arbitrary hyperparameters were chosen to see how well the tuning improved training the models. The dropout rate parameter was set to 0 and 0.2 for the two recurrent layers, respectively, and 0.1 for the dense layer. There were 128 nodes in the recurrent layers and 64 nodes in the dense layer. Batch size of 500 with a learning rate of 0.001 was used to train the each model for 10 epochs. The results are shown below in Table 7.

Table 7. LSTM Classifier Before Tuning Hyperparameters

Time Aggregate	Sequence Length	Accuracy	Recall	Precision	AUC
Events	5	0.895	0.571	0.003	0.839
Events	10	0.795	0.819	0.004	0.869
Events	20	0.857	0.723	0.005	0.857
Events	40	0.337	0.796	0.001	0.566
Days	5	0.412	0.896	0.004	0.753
Days	10	0.514	0.774	0.003	0.698
Days	20	0.437	0.805	0.002	0.675
Days	40	0.001	1	0.001	0.5
Weeks	5	0.725	0.716	0.003	0.792
Weeks	10	0.809	0.637	0.004	0.763
Weeks	20	0.775	0.603	0.003	0.731
Weeks	40	0.849	0.553	0.004	0.752
Bi-Weeks	5	0.778	0.625	0.003	0.752
Bi-Weeks	10	0.872	0.532	0.005	0.74
Bi-Weeks	20	0.835	0.545	0.003	0.722
Bi-Weeks	40	0.999	0	0	0.5
Months	5	0.842	0.532	0.004	0.733
Months	10	0.84	0.538	0.003	0.757
Months	20	0.872	0.522	0.004	0.742
Months	40	0.839	0.5	0.002	0.715

Using the best hyperparameters found from 25 iterations of Bayesian optimization, a model is trained using the both the training and validation data, and tested using the holdout set. Results from selected models are shown below in Table 8.

Table 8. LSTM Classifier After Tuning Hyperparameters

Time Aggregate	Sequence Length	Accuracy	Recall	Precision	AUC
Events	5	0.854	0.652	0.003	0.822
Events	10	0.999	0	0	0.303
Events	20	0.793	0.788	0.004	0.823
Events	40	0.999	0	0	0.5
Days	5	0.187	0.993	0.003	0.712
Days	10	0.075	0.979	0.002	0.409
Days	20	0.003	1	0.002	0.579
Days	40	0.023	0.984	0.001	0.514
Weeks	5	0.187	0.872	0.001	0.531
Weeks	10	0.001	1	0.001	0.331
Weeks	20	0.001	1	0.001	0.5
Weeks	40	0.999	0	0	0.5
Bi-Weeks	5	0.701	0.683	0.003	0.758
Bi-Weeks	10	0.806	0.58	0.003	0.707
Bi-Weeks	20	0.846	0.518	0.003	0.722
Bi-Weeks	40	0.999	0	0	0.5
Months	5	0.735	0.632	0.003	0.728
Months	10	0.905	0.369	0.004	0.67
Months	20	0.999	0	0	0.5
Months	40	0.999	0	0	0.5

The 10 best RNN models are summarized below in Table 9. Around half of these models were trained using the baseline hyperparameters as opposed to those found through Bayesian optimization hyperparameter tuning. The AUC for most of the datasets is much higher than the baseline model. The smaller sequence lengths did better for most of the datasets. The only dataset that was not time-aggregated performed the best as well.

Table 9. Comparison of the best LSTM Models

Time Aggregate	Sequence Length	Accuracy	Recall	Precision	AUC
Events	10	0.795	0.819	0.004	**0.869**
Events	20	0.857	0.723	0.005	0.857
Events	5	**0.895**	0.571	0.003	0.839
Weeks	5	0.725	0.716	0.003	0.792
Weeks	10	0.809	0.637	0.004	0.763
Bi-Weeks	5	0.701	0.683	0.003	0.758
Months	10	0.84	0.538	0.003	0.757
Days	5	0.412	**0.896**	0.004	0.753
Weeks	40	0.849	0.553	0.004	0.752
Months	20	0.872	0.522	0.004	0.742

4.5 Comparison of Classifier Results

The recall and accuracy of all models is plotted below in Figure 6. It is easy to see from this graph that LSTM RNNs give the best balance between accuracy and recall out of the three methodologies tested. It is most important to have a high recall in order to not miss many patients who are at risk of experiencing of experiencing an ORE. It is clear that LSTM is the best classifier for this task.

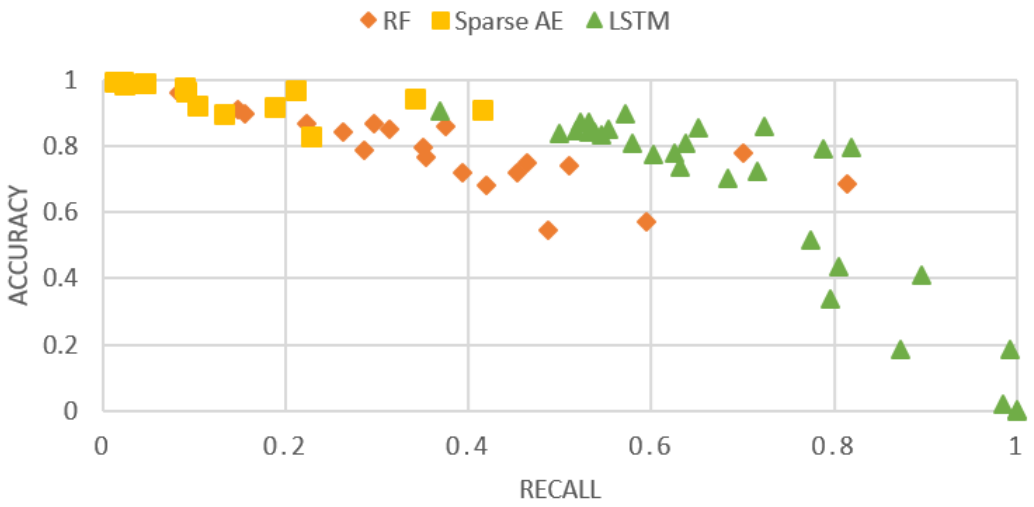

Figure 6. Recall vs Accuracy for all Models

4.6 Variable Importance

Variable importance can be easily calculated from random forest models. For the sake of brevity, only the best model's variable importances are shown. There are a few different ways to look at the importance of each variable. It is important to remember that for this model each patient's sequence of events must be flattened so that the two-dimensional sequence becomes one long vector. Each variable importance then is the importance of a variable at a certain step in the sequence. The top variable importances are shown in Table 10. These are averaged over each of the five models trained during cross-validation. Since each feature really appears a sequence length number of times in the variables, we can get total importance for that feature by summing its importance for each sequence step. The top variables from this method are shown in Table 11. Conversely, we can sum over all the variable importances for each sequence step in order to see which sequence step is most important, as shown in 12.

Table 10. Variable Importance for Random Forest with 5 Events Data

Variable	Sequence Number	Average Importance	Standard Deviation
Attending Provider Specialty: Anesthesiology	3	0.036	0.044
Attending Provider Specialty: Nephrology	5	0.025	0.02
Attending Provider Specialty: Other	3	0.02	0.039
Procedure Code: NG tube	3	0.019	0.027
Billing Provider's Specialty: Psychiatry	3	0.019	0.038
Diagnosis Code: Other eye dx	3	0.019	0.037
Diagnosis Code: Natural/environment	3	0.018	0.037
Diagnosis Code: Oth skin dx	4	0.018	0.036
Diagnosis Code: Other eye dx	4	0.016	0.032
Diagnosis Code: Ot dx kidney	3	0.016	0.031

Table 11. Average Variable Importance for each Feature for Random Forest 5 Events Data

Variable	Average Importance	Standard Deviation
Attending Provider Specialty: Anesthesiology	0.037	0.044
Diagnosis Code: Other eye dx	0.035	0.043
Attending Provider Specialty: Nephrology	0.029	0.028
Diagnosis Code: Ot dx kidney	0.024	0.03
Attending Provider Specialty: Ambulatory Surgery Center	0.022	0.041
Attending Provider Specialty: Other	0.022	0.042
Diagnosis Code: Ot infl skin	0.021	0.029
Procedure Code: NG tube	0.019	0.027
Diagnosis Code: Natural/environment	0.019	0.036
Procedure Code: No Procedure	0.019	0.038

Table 12. Total Variable Importance for each Sequence Step for Random Forest 5 Events Data

Sequence Step	Importance
1	0.000
2	0.000
3	0.274
4	0.428
5	0.298

We also looked at which variables were the most informative for other models. Table 13 shows the variables that were on average the most important across all models.

Table 13. Average Variable Importance for each Feature for All Models

Variable	Average Importance	Standard Deviation
Age at First Opioid RX Claim	0.017	0.016
Quantity of RX Claim	0.017	0.012
Procedure Code: No Procedure	0.016	0.014
Attending Provider Specialty: Other	0.015	0.016
Days Supply of RX Claim	0.015	0.017
No RX Drug Goup	0.014	0.016
Is an RX Claim	0.014	0.013
Dispensed as Written Code Blank	0.014	0.012
Diagnosis Code: No Diagnosis	0.014	0.013
Diagnosis Code: Chronic condition	0.013	0.012

5. Discussions

The main contribution of this research is the framework laid out to extract features from transaction data. This framework can be applied to any set of data, not just healthcare-related data, where entities are recorded preforming events. Extracting features this way allows machine learning models the ability to fully learn from the time-dependent aspect of the data instead of point-in-time predictions.

This feature extraction and modeling process developed can be used to predict any event. ORE was our chosen response in this research, but any event of interest can have be chosen. The success of this approach is promising that the methodology might be implemented in the real world.

5.1 Quality of Results and Implications

The extremely imbalanced nature of this data brings an added layer of difficulty in making good predictions. The datasets on average had a 1000 to 1 ratio of negative to positive responses. Given this, it would be very easy to have a near 100% accuracy by classifying all patients as negative responses. Due to the nature of the response, however, it is important to keep the false negative rate low. If a patient has any perceptible risk of an ORE they should be flagged as such by the model as intervening for a patient is in most cases much less costly than an ORE. To keep the false positive rate low, the metric we focus on improving is recall, the proportion of true positives to ground-truth positives. Precision was very poor for everyone model due to the extreme imbalance.

The best predictions made on the test set were with using the dataset with no time aggregation. With a sequence of 10 events a LSTM network had 81.9% recall while keeping accuracy at 79.5%. Comparatively to other research, this is a useful model. This model will be especially useful to healthcare insurance providers. Using their data in this model will allow them to predict with high confidence who is at the most immediate risk or an ORE. Ideally, this model would be run for a patient every time there had a new transaction recorded.

The downside to this approach is the black box nature of neural networks. The impact each variable has on the output can be calculated, but its interactions with other variables cannot be. Other models can be interpreted more easily. Interpretation is important in the medical field because prescribers need to know what variables to be aware of in patients that could lead to an ORE. This model is solely good for predicting ORE likelihood given a sequence of claims.

A shorter series of events was found to make better predictions. This is most likely due to a couple of different reasons. Events closer to an ORE are more likely to influence the outcome. Leaving in a large number of data points that had little to do with the ORE since they occurred so far before that event. This extra dimensionality hindered model performance. The greater the sequence length, the less useful information is recorded in the data.

The sparse AE performed worse than using the original features in a classifier. This is most likely due to the important features of the data being considered noise. Sparse AEs are good at finding abstractions, but they also remove outliers in a set of data by smoothing all observations to be more similar. The goal of this research is the opposite - to detect outliers. It stands to reason that sparse AEs gave poor results.

5.2 The State of Healthcare Data

There as a data-connectivity hurdle that must be jumped on the track to more accurate personalized healthcare. State prescription monitoring programs are a great start and lead to life saving interventions, but they alone are not enough. Prescription history is just one part of a patient's medical history. Data concerning diagnoses and procedures are siloed in separate EHR systems. Outcomes are hard to track due to this. Separate systems results in a lack of standardization across patient records. Fast Healthcare Interoperability Resources (FHIR) specification was developed in an attempt to standardized EHR data [72]. Higher adoption rates of the FHIR standard will improve researchers ability to develop more accurate

predictions. Higher data quality and connectivity will lead to a revolution in personalized healthcare.

Data quality was detrimental to this study. Patients come in and out of the data as they are enrolled and disenrolled in the health insurance plan. Important events may occur while a patient is disenrolled which may have a significant impact on whether they will have an ORE. There also could be an ORE before or after the time they appear in the data. This is related to the issue of right-censoring of the response. Since the sequence of events for a patient inevitably ends, either due to disenrollment or the ending of the data's time frame, there is an issue with right-censoring of the response. This means many of the patients classified as negative responses actually should be labeled as positive responses.

5.3 Limitations of Research

Using Bayesian optimization to tune the hyperparameters did not yield the best results. Many of the models trained using the hyperparameters found after 25 iterations of Bayesian optimization predicted every observation as the majority class. It might be that more than 25 iterations are needed to tune the hyperparameters. 25 iterations is very computationally expensive. This might have been better done using a random grid search methodology. A smarter selection of the hyperparameter space to search might have also improved results.

Using a validation method for time series data would have been prudent. Walkforward validation is one of the simplest. In this method, a patient would have a sequence of events created for each time step. This was not done due to the size of the data.

Using a smaller subset of patients may have been a better technique. A smaller cohort would have sped up computation times and possibly led to a more accurate model. In a similar vein, stratifying the cohort based on different demographics and training a separate model for each strata might have yielded more accurate results as well.

Even the best model found is not well calibrated. Though not included in this paper, the calibration curve (sometimes referred to as a reliability plot) is very poor. A majority of test set predictions for positive responses are around 0.5, with no observations around 0

or 1. Even though a threshold was found that showed relatively good accuracy and recall, the model needs to be calibrated. Both Platt's scaling [73] and isotonic regression [74] are common calibration techniques that could improve the model. Temperature scaling is a simple variant of Platt's scaling that has been shown to work well for neural networks [75].

5.4 Future Work

While the results are very good, it does not mean that the best strategy for modeling this data has been found. There are several strategies that still need to be investigated to improve results further.

Variable length time windows may provide better results in certain situations. In particular, using an expanding time windows that get larger for events further in the past may be useful. This research only focused on fixed length time windows and it would be interesting to see if predictions could be improving by varying the sizes of the windows.

Patient data was static throughout the sequence of events for each patient. There is no reason to input static data into a RNN; FNNs are much more appropriate for handling data that is not sequence dependent. It might give better results to separate static and event data, train a FNN using the static and a RNN using events, then ensemble the two. Similarly, it may help to split up events. We treated every event as equal in this research. In reality this is not the case. There is a reason the data was broken up into prescription, inpatient, and outpatient event tables. We could keep these tables separate, train and model for each, and ensemble the resulting models.

Combining the sparse AE and RNN methods by adding a few layers of sparsity before the recurrent layers might improve results as well. Training them at the same time might have more computational expense, but could generate better results. Similarly, we could use a supervised learning approach to train the AE alone before combining the models. In this method a final output layer would be added to the AE for training, so that it would be a prediction model minimizing a log loss function. Both of these methods are opposed to the

previous unsupervised approach in which the goal of training was to minimize mean squared error between the input and output sequences.

Given the success of the methodology, it should be used for predicting a number of other healthcare events using the same data. The methodology should be used with other transaction data as well to test the how well it generalizes to other domains.

References

[1] N. Vadivelu, S. Mitra, and D. Narayan, "Recent advances in postoperative pain management," *The Yale journal of biology and medicine*, vol. 83, no. 1, p. 11, 2010.

[2] A. S. Bohnert, M. Valenstein, M. J. Bair, D. Ganoczy, J. F. McCarthy, M. A. Ilgen, and F. C. Blow, "Association between opioid prescribing patterns and opioid overdose-related deaths," *Jama*, vol. 305, no. 13, pp. 1315–1321, 2011.

[3] J. C. Maxwell, "The prescription drug epidemic in the united states: A perfect storm," *Drug and alcohol review*, vol. 30, no. 3, pp. 264–270, 2011.

[4] L. Scholl, P. Seth, M. Kariisa, N. Wilson, and G. Baldwin, "Drug and opioid-involved overdose deaths—united states, 2013–2017," *Morbidity and Mortality Weekly Report*, vol. 67, no. 5152, p. 1419, 2019.

[5] K. E. Vowles, M. L. McEntee, P. S. Julnes, T. Frohe, J. P. Ney, and D. N. van der Goes, "Rates of opioid misuse, abuse, and addiction in chronic pain: A systematic review and data synthesis," *Pain*, vol. 156, no. 4, pp. 569–576, 2015.

[6] P. Muhuri, J. Gfroerer, and M. Davies, "Cbhsq data review: Associations of nonmedical pain reliever use and initiation of heroin use in the united states," *Center for Behavioral Health Statistics and Quality, Substance Abuse and Mental Health Services Administration*, 2013.

[7] C. for Disease Control, Prevention, *et al.*, "Guideline for prescribing opioids for chronic pain," *Journal of pain & palliative care pharmacotherapy*, vol. 30, no. 2, pp. 138–140, 2016.

[8] J. Hoffman, "Medicare is cracking down on opioids. doctors fear pain patients will suffer," *The New York Times*, vol. 27, 2018.

[9] R. Lawrence, D. Mogford, and L. Colvin, "Systematic review to determine which validated measurement tools can be used to assess risk of problematic analgesic use in patients with chronic pain," *BJA: British Journal of Anaesthesia*, vol. 119, no. 6, pp. 1092–1109, 2017.

[10] M. J. Zaki, W. Meira Jr, and W. Meira, *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[12] D. C. Turk, K. S. Swanson, and R. J. Gatchel, "Predicting opioid misuse by chronic pain patients: A systematic review and literature synthesis," *The Clinical journal of pain*, vol. 24, no. 6, pp. 497–508, 2008.

[13] B. Zedler, L. Xie, L. Wang, A. Joyce, C. Vick, J. Brigham, F. Kariburyo, O. Baser, and L. Murrelle, "Development of a risk index for serious prescription opioid-induced respiratory depression or overdose in veterans' health administration patients," *Pain Medicine*, vol. 16, no. 8, pp. 1566–1579, 2015.

[14] A. V. Karhade, J. H. Schwab, and H. S. Bedair, "Development of machine learning algorithms for prediction of sustained postoperative opioid prescriptions after total hip arthroplasty," *The Journal of Arthroplasty*, 2019.

[15] C. E. Cauley, G. Anderson, A. B. Haynes, M. Menendez, B. T. Bateman, and K. Ladha, "Predictors of in-hospital postoperative opioid overdose after major elective operations: A nationally representative cohort study," *Annals of surgery*, vol. 265, no. 4, p. 702, 2017.

[16] M. M. Ali, A. B. Tehrani, R. Mutter, R. M. Henke, E. Cutler, J. M. Pines, and M. Mazer-Amirshahi, "Factors associated with potentially problematic opioid prescriptions among individuals with private insurance and medicaid," *Addictive Behaviors*, 2019.

[17] H.-Y. Chang, N. Krawczyk, K. E. Schneider, L. Ferris, M. Eisenberg, T. M. Richards, B. C. Lyons, K. Jackson, J. P. Weiner, and B. Saloner, "A predictive risk model for nonfatal opioid overdose in a statewide population of buprenorphine patients," *Drug and alcohol dependence*, 2019.

[18] P. Geissert, S. Hallvik, J. O. Van, N. O'Kane, L. Alley, J. Carson, G. Leichtling, W. Wakeland, R. A. Deyo, *et al.*, "High-risk prescribing and opioid overdose: Prospects for prescription drug monitoring program-based proactive alerts.," *Pain*, vol. 159, no. 1, pp. 150–156, 2018.

[19] D. L. Levin, D. Paone, E. Tuazon, E. Goldmann, G. Li, and S. Martins, "Using prescription drug monitoring data to predict prescription opioid misuse," *Drug and Alcohol Dependence*, no. 140, e121, 2014.

[20] B. N. Cochran, A. Flentje, N. C. Heck, J. Van Den Bos, D. Perlman, J. Torres, R. Valuck, and J. Carter, "Factors predicting development of opioid use disorders among individuals who receive an initial opioid prescription: Mathematical modeling using a database of commercially-insured individuals," *Drug and alcohol dependence*, vol. 138, pp. 202–208, 2014.

[21] J. W. Sun, J. M. Franklin, K. Rough, R. J. Desai, S. Hernandez-Diaz, K. F. Huybrechts, and B. T. Bateman, "A novel claims-based algorithm to predict opioid overdose in the united states," in *PHARMACOEPIDEMIOLOGY AND DRUG SAFETY*, WILEY 111 RIVER ST, HOBOKEN 07030-5774, NJ USA, vol. 27, 2018, pp. 373–374.

[22] Y. Liang, M. W. Goros, and B. J. Turner, "Drug overdose: Differing risk models for women and men among opioid users with non-cancer pain," *Pain medicine*, vol. 17, no. 12, pp. 2268–2279, 2016.

[23] J. B. Rice, A. G. White, H. G. Birnbaum, M. Schiller, D. A. Brown, and C. L. Roland, "A model to identify patients at risk for prescription opioid abuse, dependence, and misuse," *Pain Medicine*, vol. 13, no. 9, pp. 1162–1173, 2012.

[24] W.-H. Lo-Ciganic, J. L. Huang, H. H. Zhang, J. C. Weiss, Y. Wu, C. K. Kwoh, J. M. Donohue, G. Cochran, A. J. Gordon, D. C. Malone, *et al.*, "Evaluation of machine-learning algorithms for predicting opioid overdose risk among medicare beneficiaries with opioid prescriptions," *JAMA network open*, vol. 2, no. 3, e190968–e190968, 2019.

[25] S. Calcaterra, S. Scarbro, M. Hull, A. Forber, I. Binswanger, and K. Colborn, "Prediction of future chronic opioid use among hospitalized patients," *Journal of general internal medicine*, vol. 33, no. 6, pp. 898–905, 2018.

[26] A. Shah, C. J. Hayes, and B. C. Martin, "Factors influencing long-term opioid use among opioid naive patients: An examination of initial prescription characteristics and pain etiologies," *The Journal of Pain*, vol. 18, no. 11, pp. 1374–1383, 2017.

[27] J. M. Glanz, K. J. Narwaney, S. R. Mueller, E. M. Gardner, S. L. Calcaterra, S. Xu, K. Breslin, and I. A. Binswanger, "Prediction model for two-year risk of opioid overdose among patients prescribed chronic opioid therapy," *Journal of general internal medicine*, vol. 33, no. 10, pp. 1646–1653, 2018.

[28] X. Li, W. Chaovalitwongse, G. Curran, J. Tilford, H. Felix, and B. Martin, "Using machine learning to predict opioid overdoses among prescription opioid users," *Value in Health*, vol. 21, S245, 2018.

[29] R. J. Ellis, Z. Wang, N. Genes, and A. Ma'ayan, "Predicting opioid dependence from electronic health records with machine learning," *BioData mining*, vol. 12, no. 1, p. 3, 2019.

[30] Z. Che, J. S. Sauver, H. Liu, and Y. Liu, "Deep learning solutions for classifying patients on opioid use," in *AMIA Annual Symposium Proceedings*, American Medical Informatics Association, vol. 2017, 2017, p. 525.

[31] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: Review, opportunities and challenges," *Briefings in bioinformatics*, vol. 19, no. 6, pp. 1236–1246, 2017.

[32] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[33] B. Shickel, P. J. Tighe, A. Bihorac, and P. Rashidi, "Deep ehr: A survey of recent advances in deep learning techniques for electronic health record (ehr) analysis," *IEEE journal of biomedical and health informatics*, vol. 22, no. 5, pp. 1589–1604, 2017.

[34] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013, pp. 1310–1318.

[35] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, 3. MIT press Cambridge, MA, 2006, vol. 2.

[38] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," 2013.

[39] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun, "Doctor ai: Predicting clinical events via recurrent neural networks," in *Machine Learning for Healthcare Conference*, 2016, pp. 301–318.

[40] E. Choi, A. Schuetz, W. F. Stewart, and J. Sun, "Using recurrent neural network models for early detection of heart failure onset," *Journal of the American Medical Informatics Association*, vol. 24, no. 2, pp. 361–370, 2016.

[41] C. Esteban, O. Staeck, S. Baier, Y. Yang, and V. Tresp, "Predicting clinical events by combining static and dynamic information using recurrent neural networks," in *2016 IEEE International Conference on Healthcare Informatics (ICHI)*, IEEE, 2016, pp. 93–101.

[42] P. Nickerson, P. Tighe, B. Shickel, and P. Rashidi, "Deep neural network architectures for forecasting analgesic response," in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2016, pp. 2966–2969.

[43] T. Pham, T. Tran, D. Phung, and S. Venkatesh, "Deepcare: A deep dynamic memory model for predictive medicine," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2016, pp. 30–41.

[44] N. Razavian, J. Marcus, and D. Sontag, "Multi-task prediction of disease onsets from longitudinal laboratory tests," in *Machine Learning for Healthcare Conference*, 2016, pp. 73–100.

[45] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel, "Learning to diagnose with lstm recurrent neural networks," *arXiv preprint arXiv:1511.03677*, 2015.

[46] S. Purushotham, C. Meng, Z. Che, and Y. Liu, "Benchmarking deep learning models on large healthcare datasets," *Journal of biomedical informatics*, vol. 83, pp. 112–134, 2018.

[47] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[48] E. Choi, A. Schuetz, W. F. Stewart, and J. Sun, "Medical concept representation learning from electronic health records and its application on heart failure prediction," *arXiv preprint arXiv:1602.03686*, 2016.

[49] E. Choi, M. T. Bahadori, E. Searles, C. Coffey, M. Thompson, J. Bost, J. Tejedor-Sojo, and J. Sun, "Multi-layer representation learning for medical concepts," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 1495–1504.

[50] Y. Choi, C. Y.-I. Chiu, and D. Sontag, "Learning low-dimensional representations of medical concepts," *AMIA Summits on Translational Science Proceedings*, vol. 2016, p. 41, 2016.

[51] R. Miotto, L. Li, B. A. Kidd, and J. T. Dudley, "Deep patient: An unsupervised representation to predict the future of patients from the electronic health records," *Scientific reports*, vol. 6, p. 26 094, 2016.

[52] Z. Che, S. Purushotham, D. Kale, W. Li, M. T. Bahadori, R. Khemani, and Y. Liu, "Time series feature learning with applications to health care," in *Mobile Health*, Springer, 2017, pp. 389–409.

[53] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[54] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `http://tensorflow.org/`.

[55] F. Chollet *et al.*, *Keras*, `https://keras.io`, 2015.

[56] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, Austin, TX, vol. 445, 2010, pp. 51–56.

[57] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, p. 22, 2011.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[59] T. G. authors, *Gpyopt: A bayesian optimization framework in python*, `http://github.com/SheffieldML/GPyOpt`, 2016.

[60] K. H. Seal, Y. Shi, G. Cohen, B. E. Cohen, S. Maguen, E. E. Krebs, and T. C. Neylan, "Association of mental health disorders with prescription opioids and high-risk opioid use in us veterans of iraq and afghanistan," *Jama*, vol. 307, no. 9, pp. 940–947, 2012.

[61] D. Sussillo, "Random walks: Training very deep nonlin-ear feed-forward networks with smart ini," *arXiv preprint arXiv*, vol. 1412, 2014.

[62] The HDF Group. (1997-NNNN). Hierarchical Data Format, version 5. http://www.hdfgroup.org/HDF

[63] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge & Data Engineering*, no. 1, pp. 63–77, 2006.

[64] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[65] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[66] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio, "Batch normalized recurrent neural networks," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2016, pp. 2657–2661.

[67] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[68] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using matthews correlation coefficient metric," *PloS one*, vol. 12, no. 6, e0177678, 2017.

[69] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[70] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[71] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[72] A. Rajkomar, E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun, *et al.*, "Scalable and accurate deep learning with electronic health records," *NPJ Digital Medicine*, vol. 1, no. 1, p. 18, 2018.

[73] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[74] R. E. Barlow, D. J. Bartholomew, J. M. Bremner, and H. D. Brunk, "Statistical inference under order restrictions: The theory and application of isotonic regression," Wiley New York, Tech. Rep., 1972.

[75] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, *On calibration of modern neural networks*, 2017. arXiv: `1706.04599 [cs.LG]`.