Machine learning and electronic health records

Sivert Stavland

Master's thesis in Software Engineering at

Department of Computing, Mathematics and Physics,
Bergen University College

Department of Informatics,
University of Bergen

June 2020

Western Norway
University of
Applied Sciences

Acknowledgments

I would like to thank Mohn Medical Imaging and Visualization Centre (MMIV) for providing me with equipment and office space to perform my experiments.

A big thanks to Sathiesh Kaliyugarasan my roommate for keeping my spirit up through the writing process and the Covid-19 epidemic. I would also like to thank my fellow master students Sondre Fossen-Ramsaas and Adrian Storm-Johannessen for two great years at the study.

Finally, I would like to thank my supervisor Dr. Alexander Selvikvåg Lundervold for giving me the opportunity to work with this project and the guidance he provided to help me in this challenging field.

Abstract

Electronic health records (EHRs) have over the past decades become the standard way of storing digital information from patients and the broader human population. In recent times, the amount of digital information that is stored in these records has exploded and the trend shows no sign of slowing down. This creates both problems and opportunities related to how to efficiently extract and use the information contained in these vast records, with the ultimate goal of improving health care, patient care, and patient outcomes.

Simultaneous to the establishment and spread of EHR, the field of machine learning has seen tremendous advancements, triggered by increases in computing power, data sets ("big data"), and new methodological developments. As machine learning is one of the most effective ways of using and extracting valuable information from vast, heterogeneous data sets, coupling EHRs and machine learning is a highly attractive prospect.

Although machine learning models based on EHR data have produced impressive results in recent times, there are many obstacles. For researchers to perform machine learning on EHR data they need access to enormous amounts of real, high-quality data. Legal and organizational regulations protect the sensitive data the records contain, making it difficult to generate sufficiently large and rich datasets. Even if one is able to access such data sets, quality and heterogeneity pose additional challenges, typically leading to a lot of work on data cleaning and preprocessing to obtain models that perform sufficiently well, meeting health care standards.

In this work, we investigate the benefits and complications of using machine learning on EHR data. We survey some recent literature and conduct experiments on real data collected from hospital EHR systems.

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

Previously health data was stored and kept in paper archives. This was a workable solution for its time, but it was tedious to access the data and storage required a lot of physical space. This made it interesting for health care worldwide to improve the ways we store and access this data. The rapid technological advances of computers and digital storage devices represented a solution to the problem [2]. Electronic health records (EHR) were invented in mid 1960's, changing the format of health records and thus changing health care. An electronic health record for a patient is a record of information about the patient's health status over time, in a form readable by computers. With the help of EHR, physicians, clinicians and members of the public can access their medical data in a quicker and more feasible way, providing a way to store all necessary health care information in one place. It also enables the ability to easily exchange information across health care organizations. This permits more effective coordination and communication of multiple clinicians, improving the safety of medications, and allowing more rapid evaluation of medications. Today, most hospitals and physicians have adopted an EHR system.

Over the past decade, EHRs have been continuously expanded and improved. Today the records contain a vast amount of various data and information, such as administrative and billing data, medical history, medications, treatment plans, medical images, laboratory and test results, allergies, and much more. Requiring a great amount of data gathering at every level of health care. To keep the data stored as homogeneous as possible, the health personnel collecting the data have to follow certain procedures and codes when filling in the data. It can be a tedious and difficult task to perform, often removing physicians' focus away from the patient. This has been described as one of the greatest challenges of EHRs [3].

However, the rich data stored in these records creates a great opportunity for "second use" of the data, a term used to describe retrospective analysis for research, academic or scientific purposes [4]. The data stored is often

sensitive and is guarded by a safety net of strict regulations for how it can be handled and accessed. Data from EHRs typically has to be processed and deidentified before researchers can use it. This is a challenging and time-consuming task, risking violations to patient privacy if not performed correctly. Because of the restrictions and difficulties with sensitive patient information, there are few available datasets for researchers to explore if they are not a part of an organization that can provide access.

There have been many attempts to analyze and review the amount of data and information stored in an effective manner, as part of the broader fields of health informatics and health analytics [5, 6, 7]. One of the approaches that stand to deliver the most promising results in EHR analytics is machine learning algorithms, particularly the subfield of deep learning [8] which has evolved immensely over the past years. What makes machine learning different from other approaches is the ability to process and learn from large amounts of data through observations instead of rules [9]. This allows machine learning systems to rapidly analyze wast amounts of data in ways humans are not capable of [10], and which is difficult to achieve with other computational approaches. By analyzing vast number of patients, machine learning systems can in principle aid physicians in their diagnosis and prognosis by anticipating future events. The overall goal of machine learning in medicine is to reduce costs without diminishing quality and patient security by using it for preventive health measures, early diagnosing, and patient insight.

However, for machine learning algorithms to achieve accurate and precise prediction scores that meet the health care standard they require the data to be sufficiently structured and consistent. The information stored in EHR systems is typically not yet structured well enough for this to be the case. The main function of EHR's is to store and report clinically collected data. The characteristics of this data are often not suited for machine learning models because of problems with temporality, irregularity, multiple modalities. EHR data can also vary significantly in the density of data from each patient, since events are irregularly sampled. Another huge challenge is the high dimensionality of the data, leading to what is called the *curse of dimensionality* [11]. To attempt to structure the existing data in a format that suits machine learning algorithms the data have to be cleaned, i.e. through preprocessing. This can be extremely time consuming, and often requires domain knowledge about the prediction tasks the machine learning algorithms are going to perform.

This thesis will be mostly be restricted to machine learning approaches to EHR data analysis. We give an overview of what is possible to accomplish and what are the problems by using machine learning with access to real EHR records from hospitals. We review some of the most remarkable recent

papers in the field, looking closely at their results, data sets, techniques, and processes. The thesis concludes with a demonstration using machine learning and deep learning techniques on one of the largest available public EHR data sets from a hospital, MIMIC-III [12]. The demonstration will present machine learning prediction and information gathering on two important pieces of EHR data, the raw clinical data stored in a tabular data format, and the clinical notes written by nurses and clinicians.

The thesis is split into two parts. In Part 1, we go over some of the fundamental theory to give the reader an overview of EHR and machine learning. Chapter 2 introduces EHR systems. In Chapter 3 we give an overview of the basics in machine learning. Chapter 4 presents the theory behind Artificial Neural networks and how they are used in Natural Language Processing. Chapter 5 gives a review of some state-of-the-art work related to this thesis.

Part 2 of the thesis consists of our experiments performed on the MIMIC-III dataset. In chapter 6 we describe our approach for using traditional machine learning algorithms to predict important medical tasks, discussing the methods and materials used and what results were obtained. In Chapter 7 we present deep learning natural language processing on medical notes to predict ICD-9 codes, following it up with methods and materials used and the results. Chapter 8 we conclude our work discussing some of the benefits and problems with machine learning on EHR. Chapter 9 addresses potential further work.

Part I

Background

Chapter 2

Electronic health records

Electronic health records (EHR) is a collection of electronic health information that is gathered in a systematical manner from every hospital visit or any care setting a patient has interacted with over time. Before EHR was invented and adopted by medical organizations all medical information that was recorded had to be kept and maintained on paper and stored in manual filing systems. Each patient had several of these papers that would be stored in medical facilities, requiring a substantial amount of physical space for storing them. With the rise of computers and computer systems came the first electronic medical system released by Lockheed Corporation in the 1960's. This started the progressive development of EHR systems to transform medical records by the usage of information technology that still continues to this day. Where healthcare professionals and consumers can retrieve information in a simpler and more productive manner [13].

The digital information stored in the records is ideally shareable across different healthcare settings. It should be available at the hospital, local doctor, nursing home and in other countries. Some of the information an EHR contains are patient demographics, laboratory test results, medications and allergies, radiology images, medical history and vital signs. EHR has thus the ability to accelerate physician diagnoses and digitalize administrative tasks, as long as it's accessible, which greatly affect the healthcare costs of today's continuous growing society. It also increases patient satisfiability by giving consumers more control and overview over their medical situation [14].

EHR analytics. Analytics is the ability to draw insights from large proportions of data and analyze them to give important insights and information. A great advantage of implementing EHR in comparison to old-fashioned paper records is the ability to automate analytics, potentially improving health care and reduce the chance of medical errors happening. Another advantage of automating processes is the time-saving factor, which

can save both general costs in health care and valuable clinician time. Analytics implemented in EHR systems arose in relation to the enormous amount of valuable data stored in them: with greater access to data come greater analytics opportunities.

Implementation of EHR analytics can give valuable insights and enhance decision-making by evaluating practitioner performance and predict patient risks and medical outcomes. EHR analytics can give new methods to evaluate the performance and effectiveness of health care practitioners at the delivery of care and how it affects patient wellness in real-time. Bringing improved patient care and effective practices of practitioners. Additionally, it can be used to predict risk and medical outcomes, flagging patients that have a higher risk for disease allowing for earlier intervention and preventive efforts.

However, the data stored in the records have to be observed, stored, and formatted. If the collected data is not clean, complete, accurate, and formatted properly then this will severely affect the outcome of analytic tools. A study done at the University of Michigan Medical School performed showed that EHRs only contained 23,5% of the data patients had volunteered to them, due to miscommunication and poor clinical documentation [15]. Due to badly entered and formatted data, a costly process of cleaning (pre-processing) data has to be performed manually. In later years there have also arose automated cleaning techniques using logical rules that can reduce time and expenses while ensuring data integrity [16, 17]. When such problems are solved, then analytics can uncover significant value for healthcare.

Today, a lot of the research into new EHR analytics systems is based on machine learning algorithms that mine EHR data and discover associations between diseases and lesser-known conditions. Much of this research is focused on deep learning because of its advancements in recent years and deep learning models' abilities to learn patterns in large amounts of data. Deep learning is today researched in healthcare to be applied to various clinical tasks such as outcome prediction, phenotyping, de-identification, and information extraction.

Interoperability. One of the greatest challenges in the healthcare industry is interoperability. The ability of different software systems to share and exchange data. In EHR good interoperability is critical, as one of its main traits is the ability to share patients medical history across healthcare organizations, increasing oerational efficiency in healthcare systems as patient data is available in real time allowing for immediate specific patient information to any caregiver. To tackle the issue of interoperability, global standards have been developed. One such standard is FHIR (Fast Healthcare Interoperability Resources). It leveraged the latest Web standards, focusing

on implementation, developed and maintained by HL7 international [18]. These healthcare standards revolve around healthcare messaging, terminology, documents, frameworks, application and even architectures. Healthcare standards ensuring interoperability are continuously being improved and developed to tackle past issues and new issues appearing.

Security. Security is also a great priority and threat to EHR. All information that is shared as a result of a clinical relationship is considered confidential and must be protected [19]. Only persons with permission or as allowed by law should be able to retrieve a patient's information. To preserve the patient confidentiality, EHR systems have to ensure that patient information is only accessible to authorized individuals. With the rise of EHR comes an increasing concern for patient information security. Some of the concerns revolve around increased use of mobile devices, medical identity theft, data exchange between and among organizations, clinicians and patients. But the greatest threat to EHR are hackers, viruses and worms [20]. Reports of theft and loss of sensitive clinical data have appeared in recent years, seriously damaging people's trust of EHRs [21, 22].

International Statistical Classification of Diseases and Related Health Problems. The international classification of diseases (ICD) is a list maintained by the World Health Organization (WHO). The list is used to classify diagnoses with the use of codes. Each disease is corresponded with a six characters long code and is nuanced with a wide variety of symptoms, signs, abnormal findings, social circumstances, etc. This system maps diseases under the same categories to a code within a specified range. An example of how the system functions: all diseases of the respiratory system are coded within the range of 460-519. Each number in that range is a specific disease and subcategory connected to the main category.

Chapter 3

Introduction to machine learning

Machine learning is a subfield of AI, based on training computers to learn over time from observed data, mimicking the way humans learn. The learned information allows computers to correctly generalize to new inputs. The main difference between machine learning and traditional software is that a software developer hasn't written code that instructs it how to operate. For example, to distinguish between pictures of fruits the software developer codes what fruit each picture contains. A machine learning model learns these features by being trained on large amounts of labeled pictures of fruits. Until it can categorize each picture into what fruit it contains. The trick in machine learning is to have lots of data.

Since computers were invented the idea of getting computers to perform and learn as humans have fascinated researchers, software developers, and engineers. Machine learning is believed to be the path to this ambitious idea. Today machine learning is everywhere, in our phones, websites, and train departure screens. We will discuss more how machine learning works in this chapter.

3.1 Data: the key component of machine learning

One of the most important components of machine learning is data. In a report published by Crowdflower in 2017 over 50% of the respondents reported that the issue of getting good quality data or improving the dataset was the greatest cause of bottleneck for completing machine learning projects. In its purest form, is a machine learning algorithm an algorithm that teaches itself based on data that is available to it. Their performance is confined by its quality and quantity. Hence it's important that the data is accurate, complete, and collected or classified from reliable sources. In machine learning there are several different types of data variables:

- Numerical data: Often referred to as quantitative data is any data that is measurable such as prices, measurements, and phone numbers. There are two forms of this data; discrete numbers and continuous numbers. An easy rule for separating the two is to remember that discrete numbers can be counted, while continuous numbers can be measured (fall into a certain range like percentages). Numerical data isn't tied to any specific point in time, they consist simply by raw numbers.
- Categorical data: Data that is sorted by characteristics. Examples include gender, ethnicity, postal-code, and workplace. This data type is non-numerical. Categorical data is often used to group together data that share similar attributes, f.ex. grouping individuals on the bases of some qualitative property. The variable is usually a fixed number.
- Time series data: Consists of data points that are indexed at specific points in time. As a result of the data being collected at time intervals it's prone to be nonuniform, meaning that the data was not collected in-between constant time intervals. Time series data have clear starting and ending points, distinguishing them from numerical data.
- Text data: Data that is words, sentences, or paragraphs that machine learning models can learn features and insights from. Since machine learning models use algorithms that learn by the use of mathematical functions, text data has to be converted into vectors before being fed into models.

Datasets in machine learning have several obstacles, and not all data will be relevant and valuable. The data must be representative of the case you want to generalize to. To get representative data it's crucial that the training data is an accurate representation of the data material faced by the model when deployed in real life. Another issue is the quality of the data. Poor quality data can have a presence of error, outliers, noise, and missing observations.

3.2 Features and feature engineering

The success of a machine learning project can depend critically on finding good features. In some cases, the existing features might be able to distinguish the input data. Other times you need to devise a method of feature extraction to come up with more relevant features for your target domain. This part of machine learning is called feature engineering and is considered one of the most crucial and demanding processes of a machine learning project. The process consists of transforming raw data into features that

better represent the underlying problem for a particular machine learning task.

- *Feature selection*: Selecting the most useful features in the training dataset and ignoring the irrelevant ones for the given problem.
- *Feature extraction*: combining existing features to create a new, more useful feature that is effective for a particular machine learning task.

3.3 Different types of machine learning

In machine learning, there are three main approaches, having to do with how training data is consumed: supervised learning, unsupervised learning, and reinforcement learning.

- In supervised learning, machine learning algorithms learn from labeled training data. The goal is to learn to predict these labels from unseen data in the future. There are two types of problems within this domain: classification problems and regression problems. Classification problems help to predict a discrete value. Each data example is a part of a particular class or group (label), and the goal of the technique is to predict which label a new data example belongs to. Regression problems are used for continuous data. In regression, the models predict real-values, called continuous values. For example, predicting the price of a house in the city, given the area, location, number of rooms, etc.
- In unsupervised learning, the goal is to explore underlying patterns and predict the output by using unlabeled data in a self-organized manner. It can find unknown patterns in data and help to find features that can be useful for categorization. The technique also has a great advantage in terms of access to data, as it's easier to get hold of unlabeled data than labeled data. Basic models in unsupervised learning include Factor Analysis, Principal Component Analysis (PCA), and Independent Component Analysis (ICA) [23].
- Reinforcement learning is about making decisions in an environment to get the highest possible reward in a particular situation. For example, it can learn how to navigate labyrinths by minimizing number of dead ends met over many iterations by learning how to improve one's performance. There have been several impressive reinforcement learning applications released in recent years. In 2019, OpenAI managed to train a reinforcement learning application to learn the complex computer game Dota2. The application managed to beat the reigning champion (team OG), and several other top players [24].

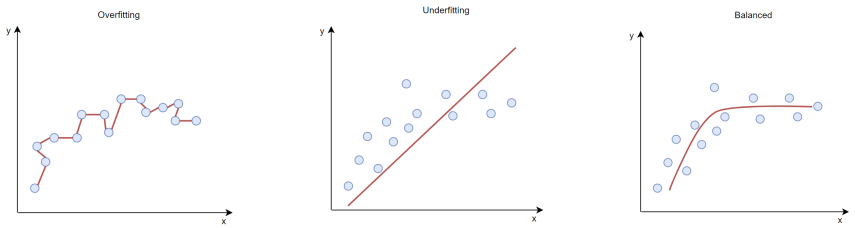
In this thesis we focus on supervised learning and classification problems.

3.4 Training and evaluating machine learning models

3.4.1 Overfitting and Underfitting

The goal of machine learning is to find and train a model that generalizes well from the training data to any data from the problem domain. A complex model usually fits the data better than a simple model. However, these won't necessarily have the greatest generalization ability: models that fit best to the training data don't always perform well on unseen data.

Overfitting occurs when the machine learning models capture the noise in the data and makes predictions based on the noise. The model will then perform well on training data, but perform poorly on new unseen instances. Underfitting is the opposite of overfitting, and happens when the machine learning models are excessively simple and can't capture the underlying trend of the data. These fitting issues are often the main reason for poor performance (generalization) in machine learning models.



(a) Overfitting (b) Underfitting (c) balance

Figure 3.1: An example of overfitting, underfitting, and good balance

3.4.2 Bias and variance

Bias and variance are prediction errors in machine learning. In machine learning, one tries to minimize these errors to improve accuracy while avoiding overfitting and underfitting. This results in models generalizing well to unseen data. Bias is the difference between the model's average prediction and the true value it's trying to predict. Variance is the variability of model prediction, telling us how sensitive the models are too small fluctuations in the datasets. Bias and variance are considered a tradeoff, a more complex and powerful model reduces bias and increases variance, and vice versa [25]. In machine learning, we try to find the balance between bias and variance that minimizes generalization error. There is also a third prediction error,

called irreducible error. An error that refers to the noise in the data, which can only be reduced by dataset cleaning.

Models with high bias and low variance pay very little focus on the training data and oversimplify the model making them unable to capture the underlying pattern of the data, leading to a high error and underfitting. A model that is underfitting can potentially be improved by feeding the model more data or, if the reason is insufficient expressibility of the model, by choosing a more complex model. A general rule in machine learning is that we can never have too much data. This is also the case in overfitting. A model that has seen too few cases can't learn the underlying features in the data.

Beyond this, it is difficult to prevent overfitting, requiring difficult techniques, collectively called regularization techniques (like dropout and early stopping), to prevent it. Dropout and early stopping are explained in section 4.4.2 and 4.4.3.

One fundamental practice to detect and prevent overfitting and underfitting is to split our data into three subsets; training set, validation set, and test set. The models weights are adjusted by the training set to minimize the error on the prediction task. A validation dataset is used to estimate how well your model has been trained and used to tune the hyperparameters and estimate model properties to select the most promising model for the given problem. The model sees this data, and can thus not be used to calculate the model's performance. The final generalization property of the model is estimated by predicting the outcomes of the test set, which remains unseen until the model construction is finalized.

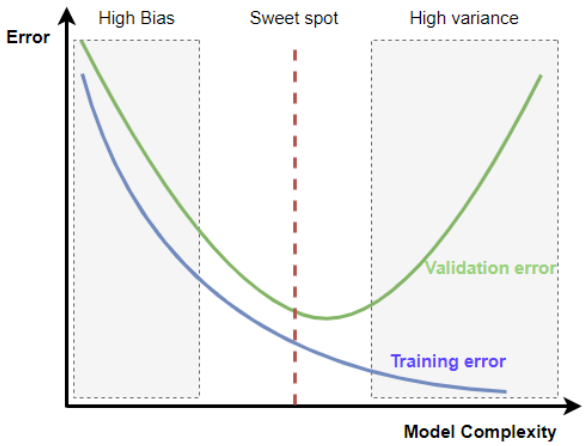


Figure 3.2: The region to the left (high bias), where both training and validation errors high. In the region to the right (high variance) validation error is high, but the training error is low. In the middle is the sweet spot.

3.4.3 Gradient descent

Gradient descent is an optimization algorithm used to find a minimum value of a function. In machine learning, gradient descent plays a key role by being responsible for the update and optimization of parameters in artificial neural network models, in a way that reduces prediction error. The algorithm operates by taking iterative steps in the direction of steepest descent, as defined by the negative (the opposite direction) of the gradient. A learning rate gives the algorithm the size of the step that it should take towards the minimum. A small learning rate leads to slower convergence, while a large one can lead to overshooting, missing the minimum. In other words the gradient tells us the direction and steepness, while the learning rate dictates how far we go in the given direction at each iteration. This process is repeated until a local or global minimum is found.

There are three variants of gradient descent: batch gradient descent, stochastic gradient descent and mini-batch gradient descent. Each of the three variants is differentiated by the amount of data used in calculating the gradient. Variation in the amount of data in each variant results in a trade-off between the time it takes to make an update and the accuracy of the parameter update. Batch gradient descent uses the entire data set each iteration to compute the gradient, making the method slow and computationally intensive. Stochastic gradient descent on the other hand only uses a single random sample from the training data each iteration, thus significantly increasing the speed of the computation. This computation however, has a high variance that causes the movement on the loss surface to fluctuate heavily. Mini-batch gradient descent is a combination of the two methods, where a random subset of the training data, of a fixed size called the batch size, is selected at each iteration to calculate the gradient.

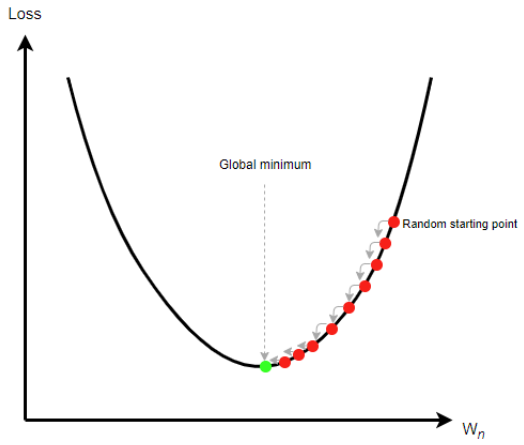


Figure 3.3: Gradient descent to find global minimum.

3.4.4 K-Fold Cross-Validation

K-Fold Cross-validation is a technique used in machine learning to help figure out good settings for hyperparameters of a model. The technique is used to prevent bias and variance. In K-Fold cross-validation, we split our training set into K number of subsets, called folds. The model is then trained k times, each time one of the k subsets is used for validation, and the other subsets are used for training. At the end of the training, the performances of the model on each fold is averaged to come up with the final validation metrics for the model. For hyperparameter tuning, the Cross-validation process is repeated several times, with each iteration using different model settings. The best model is chosen and trained on all the training data, and later evaluated on the test data [26].

Random Search Cross-Validation

In this technique, we define a grid for our hyperparameters and perform for example K-Fold Cross-Validation. The technique picks samples randomly from the grid, performing K-Fold Cross-Validation for each combination of values. By doing so we can narrow the search for optimal hyperparameters and save computation time.

3.4.5 Evaluating classifiers

A machine learning model's success is defined by measurements of the model's performance. Thus, there are several different ways of measuring the performance to see if your model is able to learn the patterns in the data. This measure is called a performance metric, and the choice of this performance metric is dependent on the target task the model is trying to predict [27].

Confusion matrix: A confusion matrix is used in classification to represent the predicted vs. actual classes. For binary classification, the representation describes an output of true and false negatives and positives.

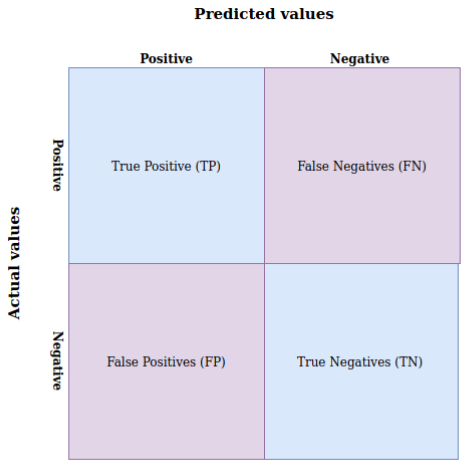


Figure 3.4: An illustration of a Confusion matrix for a binary classifier.

Accuracy: Measures the fraction of the model's prediction that are correct. and is the most commonly used metric for classifiers. However, it is not a very detailed metric, which can be a problem when there is a severe class imbalance in the dataset or if you care, say, more about the number of false negatives than true positives. In such situations, accuracy can give a *false* sense of good performance.

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

Precision: The model's proportion of predicted positive instances that is correct. As a result of low precision in the example of detecting diabetes, many patients will be told that they have T2DM and that will include some misdiagnoses.

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

Recall: The mode's proportion of actual positive instances out of the total actual positive. In other words the ability to find all the positive samples

$$\text{Recall} = \frac{TN}{TN + FP}$$

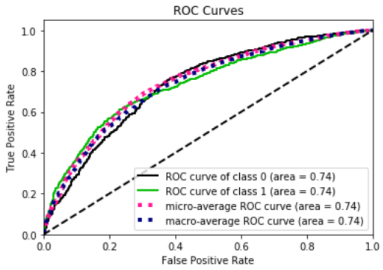


Figure 3.5: An example of a ROC curve. The goal is to have a treshold value that leads the model up to the top-left corner.

To evaluate the effectiveness of a model one has to examine both precision and recall scores. However, improving precision often reduces recall and vice versa, therefore creating a trade-off between the two, and consequently forcing an optimization for metrics most useful for the target task problem.

F1 score: An overall measure of the model's accuracy, measured by combining precision and recall: the harmonic mean between precision and recall. With a high F1 score a model has low false positives and low false negatives.

$$\mathrm{F1} = 2 \times \frac{\mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

ROC curve: Receiver operating characteristic (ROC) is a graph where the false positive rate is plotted against the true positive rate.

$$\text{True positive rate} = \text{Recall} = \frac{TN}{TN + FP}$$

$$\text{False positive rate} = \frac{FP}{TN + FP}$$

3.5 Tree-based ensemble machine learning models

3.5.1 Random Forest

A random forest is an ensemble of decision trees, i.e. a combination of decision trees. A random forest is a non-parametric predictive model used in machine learning for supervised learning and can handle both classification and regression tasks. A decision tree is constructed during learning using algorithms (gini index or entropy) that identify ways to split a data set based on which features result in the optimal data partition. It is viewed as a tree graph, where the data split decision is represented by a node, the answer to the split is represented by branches, and the output is represented by the leaves. Figure 3.6 demonstrates an example of how a decision tree makes a decision in a classification task. The task in the example is for the model to predict which class an instance belongs to; boat, car, bicycle, or sled. It starts at the root node, the node asks if the instance has an engine. The answer decides the path followed in the tree (traverses the three from top to bottom). This step is repeated until we reach a leaf node, where the the model's predictions are found.

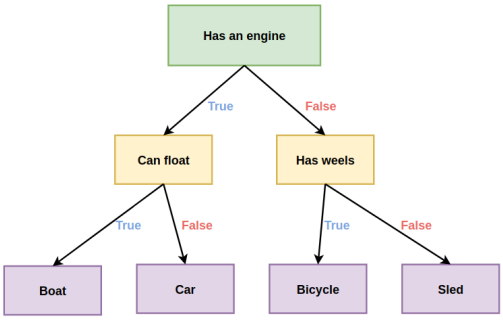


Figure 3.6: Decision tree classification task example.

Random forest contain several Decision Trees operating in an ensemble, thus the name forest. *"Ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a (weighted) vote of their predictions"* [28]. Decisions are made in random forest models by combining the results from the ensemble of decision trees and performing a majority vote to produce a prediction on the target task. An example of this process can be seen in figure 3.7.

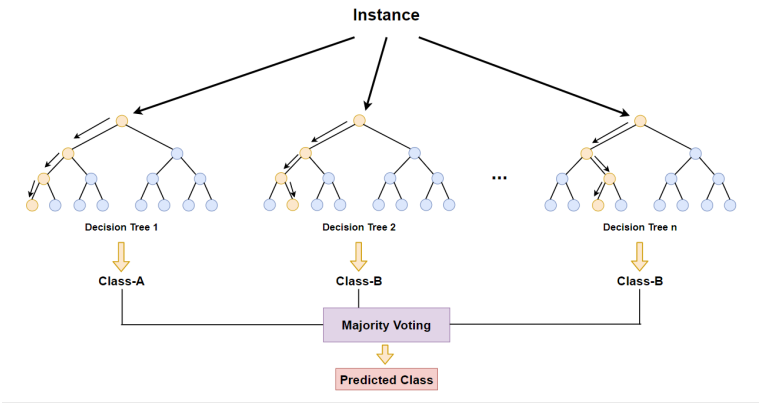


Figure 3.7: An example of the Random Forest model performing a classification task.

In random forests, the key lies between the low correlation between the models. An ensemble of uncorrelated models can produce predictions with improved performance accuracy over each individual model. The decision trees are combined into random forest by the use of bagging. Bagging in random forest trains the decision trees on randomly sampled subsets of the data, while sampling is done with replacement. Each decision tree is sensitive to noise in the data and is very prone to overfitting. Bagging increases the variance of the model without increasing the bias, as long as the individual trees have low correlation. Thus reducing noise in the data and reduces the chances for the model to overfit. Alongside bagging, the model also uses a technique called feature bagging. A technique that only considers a random subset of features at each split in the decision tree. As for bagging, this reduces variance by ensuring that strong predictor variables impact is reduced.

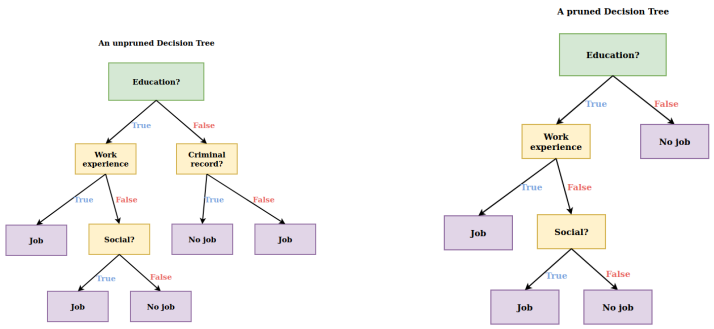
Random forests are considered among the most powerful machine learning models. A random forest can handle large data sets with thousands of input variables without variable deletion. It also gives an indication of what variables are important to the classification. Perhaps its most important feature is the ability to estimate missing data and maintaining accuracy when a large portion of the data is missing [29].

3.5.2 XGBoost

XGBoost stands for "Extreme Gradient Boosting" and is a supervised decision-tree-based ensemble machine learning algorithm that uses a gradient boosting framework. The algorithm was developed by Tianqi Chen and Carlos Guestrin in 2016 [30] and has since shown several state-of-the-art results [31]. The algorithm was developed to efficiently reduce computing time and allocate an optimal usage of memory resources.

Same as Random Forest, gradient boosting also uses an ensemble method that adds a new Decision Tree at each iteration and improves the old ones. However, XGBoost doesn't assign different weight to the classifiers after every iteration, it fixes what it has learned, and add one new tree at the time. The new tree is fitted to the residuals of the previous prediction and then minimizes the loss when adding the latest prediction (optimizes the objective). As a result, the model's parameters are updated using gradient descent. Hence the name, gradient boosting.

The algorithm also contains regularization in the form of tree pruning. Rather than enumerating all possible trees and picking the best one it optimizes one level of the tree at a time. Specifically, it tries to split a leaf, and score its benefit to classifying instances. This is performed iteratively by scoring the new left leaf, then the new right leaf, then the original leaf. If the benefit of adding another branch is smaller then some threshold, then it's discarded. An example of Decision Tree pruning can be seen in figure 3.8.



(a) Before pruning (b) After pruning

Figure 3.8: An example of pruning of a Decision Tree. If a subject doesn't have the measured level of education then the gain from adding that branch is lower than the threshold and is thus discarded.

Chapter 4

Deep learning in Natural Language Processing

4.1 Natural Language Processing

Natural Language Processing (NLP) is a field of study aiming to enable computers to analyze, understand, manipulate, generate natural language. The field combines computational linguistics, computing science, cognitive science, and machine learning. The goal of NLP is to learn or detect context from an input of words to solve meaninfull tasks. Typical applications in NLP include dialogue systems, question answering, sentimental analysis, speech recognition, information retrieval, and natural language generation [32]. In the field of NLP, there are multiple approaches:

- Symbolic approach: The approach uses language rules derived from experts of the given language and implements them into computer systems.
- Statistical approach: This approach uses statistical techniques to find recurring themes in the language text that can be used to form its own specific set of language rules. These rules can then be applied to new input.
- Connectionist approach: This approach combines symbolic and statistical approaches in applications.

Through the development of computers, increased computation power, and increased understanding of NLP, the data-driven statistical approach have become the most popular, and are to this day [33], especially methods based on machine learning. Traditional machine learning algorithms often didn't have the capacity sufficiently large to absorb the large amounts of training data. Because of this, they had problems covering all regularities in languages by designing features manually with domain experience. NLP

required more powerful learning algorithms, methods, and infrastructures to get closer to human-level performance. This gave rise to the current deep learning wave in NLP. Deep learning approaches exploit that powerful artificial neural networks, see Section 4.2 below, are capable of learning representations from data using a cascade of multiple layers of nonlinear processing units for feature extraction. Avoiding the need for linguistic domain experts to design features manually, i.e. "NLP from scratch" [34]. Deep learning approaches can, to a certain extent, provide useful solutions to tasks such as: text summarization, question answering, semantic parsing, sentiment analysis, text classification, semantic role labeling, and more.

4.2 Artificial Neural Networks

Artificial neural networks (ANNs) is a class of machine learning models designed to recognize patterns. The networks are modeled loosely after the human brain and its biological neural network. An artificial neural network recognizes the patterns in data fed to network as vectors, into which all real-world data must be translated to.

The main component of ANNs are neurons: computational units that mimic the neurons in biological brains. An ANN is made up of these neurons stacked in layers, with weighted connections between them (weights). As shown in figure 4.1 the network consists of layers, the first layer called the input layer, the middle layers called hidden layers, and a final layer called the output layer. An ANN with multiple hidden layers is referred to as a deep neural network [35].

The input signals consisting of vectors are fed to the networks through the input layer and into the network. A neuron calculates the dot product between the input signals and its corresponding weights by the computation:

$$sum = \vec{X} \cdot \vec{W} = \sum_{i=1}^{n} x_i w_i$$

This weighted sum is then passed to a nonlinear function called an activation function (Section 4.2.1), which calculates the output of a neuron, determining whether and to what extent that signal should progress further through the network and affect the outcome. Combining several of these steps creates a larger neural network. Note the importance of the nonlinear activation functions: without such nonlinearities, a neural network would simply be a composition of linear functions and thus itself linear. Such models wouldn't be suitable for modelling nonlinear relationships between inputs and outputs.

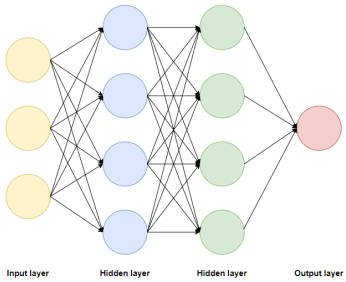
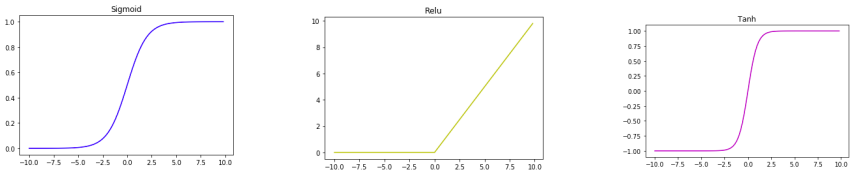


Figure 4.1: Deep Neural Network architecture, consisting of input layer, hidden layers, and output layer.

The ANN model in the figure 4.1 is called a feed-forward network. The data in the network is only passed one way, from the input layer to the hidden layers to the output layer. Hence the name feed-forward. The part of the network that gets updated through training is the weights. They are typically randomly initialized before training, leading to a starting point that doesn't produce useful predictions. Each layer is trained on features derived from previous layer's output. Giving the network's neurons the ability to learn more advanced features in the input data the further into the network it reaches.

4.2.1 Activation functions

Activation functions are used in neurons to calculate their output signal based on the input signals they receive. The purpose of the functions is to introduce non-linearity into the output of the neuron, making an artificial neural network able to learn and tackle complex tasks. The output of the functions evaluates the relevancy of the data the neuron receives, and to what extent it should impact the next neuron. See Fig. 4.2 for examples of some commonly used activation funtions.



(a) Sigmoid (b) ReLU (c) Tanh

Figure 4.2: Sigmoid, ReLU, and Tanh activation functions

Training a neural network

As mentioned above, the weights in an ANN get updated during the training so that the ANN can recognize features in the data. The weights get updated through a technique called backpropagation and the goal of the ANN is to optimize all of the weights to minimize an error. In an ANN the input data get fed to the network and propagated forward through the layers enabling the network to detect advanced features from the input until it reaches the output layer and makes a prediction. The network's prediction error is then calculated using a loss function to calculate the difference between the true value of the input versus the prediction.

$$true\,value - prediction = error$$

This error is then backpropagated (walk the error backward) over the model. Each neurons incoming weights is then scaled up or down according to how much the neuron contributed to the calculated error using an optimization method, such as gradient descent 3.4.3 to minimize the error function. The three-step process of scoring input, calculating loss and applying an update is repeated until we have iterated through our training data. In machine learning terms this is called training our model for one epoch.

$$error \cdot weight's\,contribution\,to\,error = adjustment$$

4.3 Embeddings

An embedding is a mapping of a discrete categorical variable to a vector of continuous numbers in a relative low-dimensional space, which can translate high-dimensional vectors. By performing this mapping the embedding learns features that could be used in machine learning to learn features from large input vectors, like words. The objective of an embedding is to place semantically similar inputs close together in the embedding space, thus capturing some of the semantics of the input. When working with text it must be converted into numbers before it is fed to the model. This is often referred to as vectorization of the text. There are several strategies to perform embedding of words. We describe three below.

4.3.1 One-hot encoding

Each word is encoded in the vocabulary (unique words). The words are represented with a zero vector with length equal to the vocabulary and a one is inputted in the index of the vector that corresponds to that word. Each one-hot encoded word vector are then added to create a vector that

contains the entire encoding of the sentence. The approach is shown in the diagram, with the sentence (learning, machine, learns, model, the).

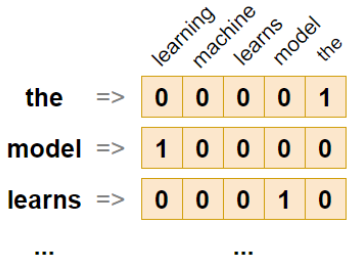


Figure 4.3: An example of a one-hot encoded sentence.

This approach is typically very inefficient. The resulting vectors are very sparse, meaning that most of the indices are zero.

4.3.2 Unique number encoding

The second approach is to represent each word with an encoding of a unique number. The numbers assigned could be any given number as long as it's unique. The example sentence *"The machine learning model learns"* from Figure 4.3 could be represented as a dense vector [5,2,1,4,3]. In contrast to the one-hot encoding approach, where it produces a sparse vector, this approach produces a dense vector (where all elements are non-zero). However, the approach is not without its faults. It doesn't capture relationships between words, because the integer-encoding is arbitrary. A second problem is that it does not capture the underlying relationship that exists between words and their encoding, resulting in a combination of feature-weights that has no meaning. This can make it a challenge for machine learning models to discover useful interpretations of integer-encodings.

4.3.3 Word embeddings

Word embeddings is a representation of words where relationships between words are captured by giving the words a similar encoding. The word embedding is a dense vector of floating point values, which is set to random numbers at the start. These values are trainable parameters, and the weights are learned in the same way during training as the weights in a dense layer. The dimensions of the embedding captures the relationship between words in the vocabulary. A word embedding that has a greater number of dimensions have the ability to capture more specific relationships that exists between words. Common practise is to use smaller dimensions for smaller datasets, and greater dimensions for bigger datasets. When the word embedding is trained, we can encode each word in our vocabulary by looking up the dense

vector it corresponds to in the table, thus it is often referred to as "lookup table"

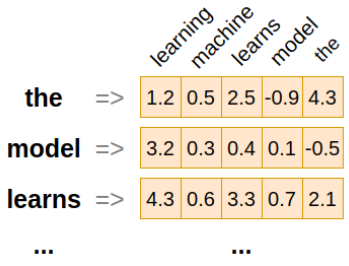


Figure 4.4: An example of a word embedding.

4.4 Regularization

Regularization is techniques that can be implemented to control the complexity of models by for example decreasing coefficients towards zero, thus penalizing complexity which helps models avoid overfitting and improve generalization.

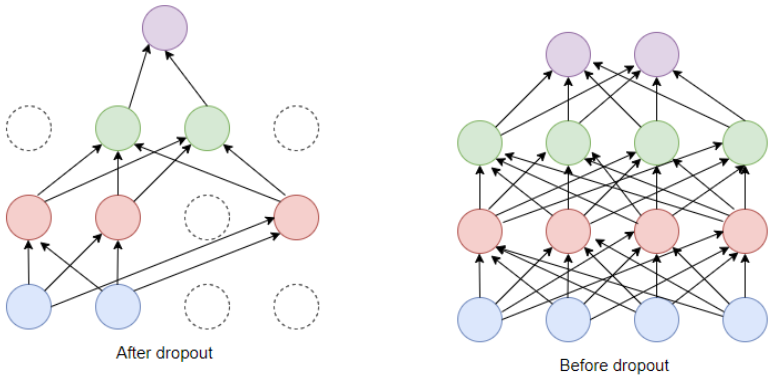
4.4.1 L1 and L2 regularization

L1 regularization L1 regularization uses the absolute sum of the weights to minimize weights towards zero. By enforcing this rule onto the weights it forces some of them to become close to zero, and therefore to not affect the outcome.

L2 regularization L2 regularization is a way of penalizing complexity by imposing a harder penalty to larger weights at each parameter update, without them becoming zero.

4.4.2 Dropout

Dropout is a regularization technique introduced by G.E Hinton in 2012 [36]. The technique's purpose is to reduce overfitting by preventing complex co-adoptions in training data. The central focus of dropout is dropping out units, more specifically neurons in the artificial neural network, by doing this we sever the connection coming in and out. Neurons are ignored based on a predefined probability at each iteration (usually 50%). These neurons will be ignored during the specific training iteration, resulting in a simpler training iteration. Because of this each neuron becomes less finely-tuned to the particularities of other neurons, thus preventing complex co-adoptions in data, and thus lower the risk of overfitting.



(a) Before dropout (b) After dropout

Figure 4.5: An example of a neural network before and after dropout

4.4.3 Early stopping

Early stopping is a widely known and used regularization technique, often used in combination with other techniques to improve generalization. The idea is to stop training before the machine learning model starts to overfit the training data. Overfitting is detected by monitoring and measuring training loss and validation loss, stopping the training process when the validation loss (performance on the validation dataset) becomes significantly worse than the training loss, or before it starts increasing.

4.5 Recurrent Neural Networks

Recurrent neural networks, also known as RNNs, is a class of artificial neural networks. Vanilla neural networks take fixed size vectors as its input. This limits the networks in situations where the input is of a "series" that doesn't have a predetermined size. RNNs are made to accept such series as their input. Each item of a series can be related to the others, making them likely to influence their neighbours. RNNs can capture these relationships across inputs. More importantly, it can remember the relationships it perceived previously in time, and the network's outputs are influenced by that knowledge [37].

RNNs learn similarly to a feed forward network, but they use what they learned from previous inputs while generating outputs. The network does this by not only having weights influence the input, but also a hidden state vector. This hidden state vector represents the context of what the network learned from previous inputs and keeps changing for each input the network is fed. Making it constantly update the hidden state, its "memory", analogous to how humans learn from their experiences. This network design is

often represented as a loop, as shown in figure 4.6.

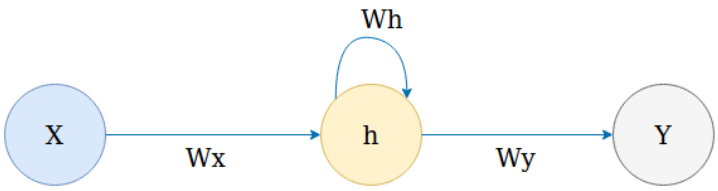


Figure 4.6: An example of part of a RNN A looks at an input X and outputs a value h. The shown loop is what allows information to be passed from one time step to another.

A problem with Recurrent Neural Networks

The design of RNN is made to handle sequences of events that occur in succession, while understanding each event based on the previous events. To get the best possible outcome we would want the network as deep as possible to extend the long-term memory and the possibilities of picking up long-term relationships. But depth is however not necessarily beneficial to the network, because it often leads to a vanishing gradient problem when the gradient is passed through several time steps.

Vanishing gradient problem

Vanishing gradients can occur when training ANN with gradient based learning methods and backpropagation. In some cases the gradients will be vanishingly small, preventing the weight from changing its value, and in a worst case scenario it could stop the training completely. These cases are the cause of the use of common activation functions like sigmoid and tanh that "squish" their input into a very small output range in a nonlinear fashion. Multiplying several of these small numbers for the front layers in a network means that we will get a decreasingly smaller gradient for each of the layers in the network. A RNN will increase the risk of encountering a vanishing gradient problem and losing information learned with each time-step added to the network [38].

4.6 LSTM

Long short-term memory units are a variant of recurrent neural networks developed by Hochreiter and Schmidhuber in 1997 [39]. The network maintains a more constant error, by preserving the error that can be backpropagated

through time. Allowing recurrent nets to continue to learn over several time steps, thus escaping the vanishing gradient problem of RNN.

LSTMs have a gated cell, outside of the normal flow of the RNN. The cell behaves much like computer memory, by having information stored, written and read from it. Inside the cells there are several gates that open and close to make decisions over what to read, write and erase. These gates are implemented with element-wise multiplication by sigmoids (range of 0-1) and react to signals they receive. Blocking and passing information based on its import and strength. Similar to a neural network's nodes, these weights are adjusted via the RNN learning process. Making the cell learn when to allow data to enter, leave or be deleted through the iterative process of the network.

LSTM cells receive information into the cell at multiple points. The combination of the input and the previous cell state is fed into the cell and the three gates within, which decide how to handle the input. Each gate respectively decides to accept or decline the new input, erase the present cell state, or let that state impact the network's output. Cells perform this decision at each time step, and combine their open and shut states at each step to decide if it's going to accept or decline forgetting, writing to or read from its state. Thereby storing or releasing memory, preserving the long term dependencies in the network.

Contained in the cells are three gates.

- Forget gate: responsible for removing information that is no longer necessary for the completion of the task.
- Input gate: adds information to the cells.
- Output gate: selects and outputs necessary information.

Memory cells in a LSTM give different roles to addition and multiplication in transformation of input. The plus sign in a cell is said to be the secret of the network. Instead of determining the subsequent cell state by multiplying the current state with the new input, they add the two. This is what makes the difference and helps preserve a constant error when it must be backpropagated through time and layers.

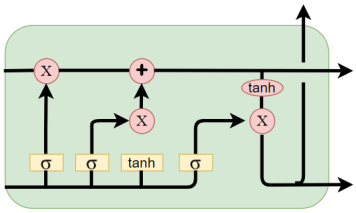


Figure 4.7: An LSTM cell and its components.

By excelling at long-term memory, LSTM networks perform particularly well at predictions based on previous information, such as text prediction and stock prediction. In the category of text prediction the network shines at predicting the next word of a sentence. This task requires the network to retain all the words that precedes it, making LSTMs suitable models.

4.7 Transfer learning

Transfer learning is the process of leveraging existing knowledge of some domain and using it to your advantage by applying it to our problem of interest in the domain. The existing knowledge can have various forms depending on the data. This process has shown to be a powerful technique in NLP [1, 40, 41], and computer vision [42]. In NLP, transfer learning is used to pre-train Language Models that have contributed to the state-of-the-art on a wide range of tasks. The general practice is to pre-train representations on a large unlabelled text corpus and then adapt these representations to a supervised target task using labeled data.

A language model (LM) is a probability distribution over sequences of strings or words, where every word or string is assigned a probability by the use of various statistical and probabilistic techniques. A trained language model can then provide context to distinguish between words and phrases that sound similar by learning features and characteristics of language. With this ability, the LM can be used to predict the next word or sentence in a sequence. Neural network language models capture many facets of language relevant for downstream tasks such as long-term dependencies (reducing the impact of the curse of dimensionality) and can be easily adapted to a classification task.

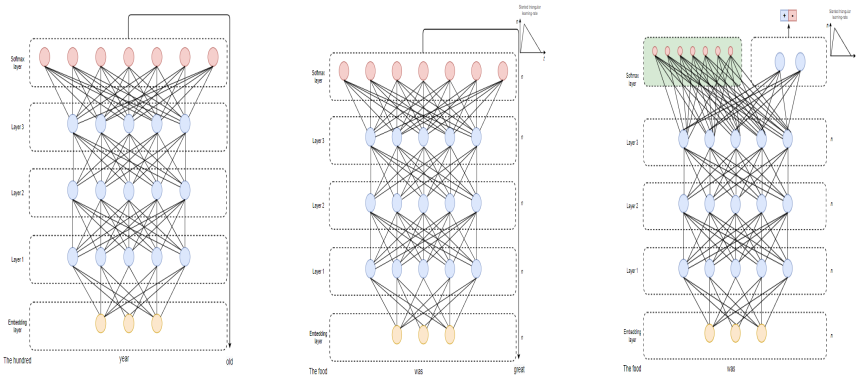
Pre-training language models have a great advantage in the area of NLP. Learning a language is a difficult task even for humans and requires learning about syntax, semantics, and certain facts about the world. For machine learning models to be able to reasonably good at this job they require a large amount of data and parameters. Pre-training language models reduce the need for annotated data labeled by domain experts, and can thus be trained on a wide range of data corpora. With the use of transfer learning these pre-trained language models can achieve similar performance compared to a non-pre-trained model with fever examples [1]. However, pre-training is cost-intensive, requiring a large amount of computational power which is not accessible by everyone. Luckily there is a wide range of shared pre-trained models available for download [43].

4.8 ULMFiT

Universal Language Model Fine-Tuning for Text Classification (ULMFiT) is an effective transfer learning method for text classification that can be applied to any task in NLP, and introduces techniques that are key for fine-tuning a language model. The method was created by Jeremy Howard and Sebastian Ruder in 2018 [1] and at the time outperformed several state-of-the-art methods on six text classification tasks. Deep learning models had at the time achieved state-of-the-art results on many NLP tasks, but they required large datasets because they had to be trained from scratch. Transfer learning had yet to be successful in NLP up until ULMFiT showed that language model fine tuning was possible if larger datasets and different fine-tuning methods were used [1].

ULMFiT uses a regular AWD-LSTM architecture with three layers (without any other complex additions) throughout its three stages. The architecture is a regular LSTM with various tuned dropout hyperparameters [44].

- A pre-trained LM on a large general domain corpus to capture general features of the language in different layers.
- The LM is then fine tuned to the target task data using novel techniques to learn task specific features.
- Then a classifier is fine-tuned on the target-task while preserving low-level representations and adapting high-level ones.



(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

Figure 4.8: ULMFiT model training process. The illustration is inspired by the model architecture illustration in the ULMFiT paper [1].

General domain LM pre-training

This step builds on the idea used in ImageNet pre-training in computer vision [45]. Language modeling has the property of capturing general properties of language, making it a perfect source task for pretraining a network. The AWD-LSTM network is pre-trained on Wikitext-103, consisting of 28,595 preprocessed English Wikipedia articles and 103 million words [44].

Taget task LM fine-tuning

The next step is to fine-tune the language model to the target task (classification data which classification will be performed). This step is done to enhance the classification model on small datasets, such that this technique is viable for alle NLP problems, independent of the data size. To achieve this step the authors propose using the two following methods.

Discriminative fine-tuning

In ANN different layers capture different types of information, and should be fine-tuned to different extents to not lose valuable information. Discriminative fine-tuning achieves this by using different learning rates for all layers of the model, and allows each layer to be tuned with different learning rates. Instead of training each parameter with a learning rate of n, the learning rates were affected by the number of layers in the network $n_1, ..., n_L$. Resulting in greater parameter tuning at the last layers, and a decreasing tuning for the earlier layers.

Slanted triangular learning rates

The technique is used to speed up the rate a model converges to a suitable region of the parameter space at the start of the training and then refine the parameters. This is a more beneficial way to train ANN. Slanted triangular learning rates (SLTR) linearly increase the learning rates at the beginning in a short time of training and then linearly decays it for a longer period of time. STLR is a modification of the triangular learning rates [46]

Target task classifier fine-tuning

In the final step, ULMFiT adds two additional linear blocks to perform text classification. The intermediate layer uses a ReLU activation function, and the final layer uses a softmax layer. Each of the blocks uses batch normalization and dropout.

Gradual unfreezing

Gradual unfreezing is a technique where we freeze all layers in a pre-trained network. One frozen layer at the time, starting from the last layer is unfrozen and fine-tuned for one epoch, this process is repeated until all layers are fine-tuned to convergence. This method is used as the weights in the classifier network are very sensitive to aggressive fine-tuning, avoiding catastrophic forgetting of the pre-trained language model.

4.9 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a machine learning model that began a new era in NLP when it was released in 2018 by Jacob Devlin and his colleagues in Google [40]. Since its release BERT has broken several records in language-based tasks. It was developed around the concept of powerful components that represents words and sentences in a way that captures the underlying meaning and relationships, often referred to as NLP's ImageNet moment. These *components* could be downloaded and used in other models and pipelines - saving time, energy, knowledge, and resources that might have gone into training a language-processing model from scratch. Versions of the model that are pre-trained on massive datasets are available for download, alongside open-sourced code for the model: https://github.com/google-research/bert.

BERT is a trained Transformer Encoder stack - a model that uses attention to boost the training speed of these models. A regular transformer model has both an encoding component and a decoding component and is regularly used to translate text, but BERT only uses the transformer encoder stack (referred to as Transformer Blocks in the paper). The encoders all have identical structure but don't share weights. Each contains two sub-layers - a self-attention layer and a feed-forward neural network layer.

Inside an encoder the input's first flow through the self-attention layer - a layer that looks at other words in the input sentence as it encodes a specific word. The outputs from this layer are then fed to a feed-forward neural network. Each encoder block has the exact same feed-forward network independently applied to each position. In NLP applications the input words are turned into vectors before being fed into the models, this is the same for BERT. All unique words in the sentence are given a token that is represented with a number - before they are fed into a word embedding. Unique for BERT is the [CLS] and [SEP] tokens that are added at the beginning and end of each sentence ([CLS] to mark the start and [SEP] to mark the end). The first encoder stack receives the word embeddings as input, the other stacks get fed the previous stack's output as their input. The size of these initial vectors is set by a hyperparameter, and as a general rule, it is set to the length of the longest sentence in the training dataset (often

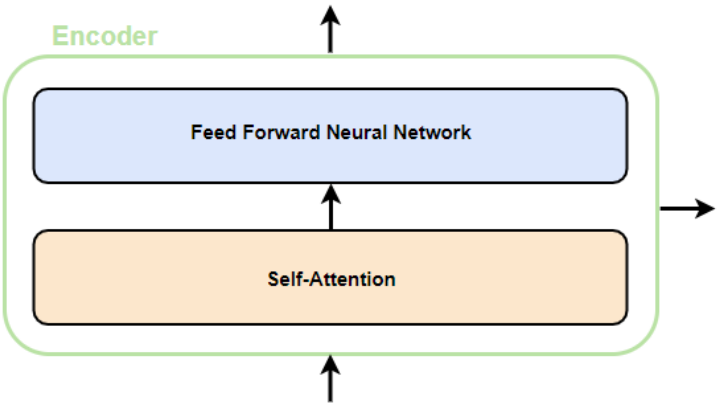


Figure 4.9: A Transformer Encoder with its two layers, Feed-forward Neural Network and Self-Attention.

set to 512). Each embedded word then flows through its own path in the encoder. All of the paths have dependencies between them - until they reach the feed-forward layer in the network, where there are no dependencies, thus the various paths can be executed in parallel while flowing through the layer.

A Transformer's Self-attention layer allows the model to look at other positions in the input sentence for hints that can help the model to make a better encoding for this word (underlying meaning and relationships). It is used to take the "understanding" of other relevant words, and bake them into the current word that the layer is currently processing. This can be shown with a text example: "James was playing computer games with his brother when their mother called". The self-attention allows the model to learn to associate the word "their" with the brother and James when it's processing it. A similar process to what happens in an RNN when the hidden state representation of previous words is incorporated into the word it is processing. For more information about the self-attention layer, we recommend reading the paper "Attention is all you need" [47].

BERT's language model is trained in a special way. The model uses two techniques to train its language model - masked language model and next sentence prediction. The first technique is used to train the Transformer's stack of encoders. BERT performs this by masking 15% of its input with a [MASK] token and tries to predict the correct word, this is also done randomly to some words to improve how the model fine-tunes. The second technique trains the model given two sentences (A and B), and the model is to classify if sentence B is likely to be the sentence that follows A, or not. When choosing the sentences A and B, 50% of the time B is the correct next sentence that comes after A in the actual text, and 50% of the time it's a random sentence from the corpus. With the use of this technique, BERT is

able to improve handling relationships between multiple sentences.

BERT has several different pre-trained models that can be downloaded, differing in model sizes (amount of transformer blocks, hidden units in the feed-forward network, and attention heads). In the original paper, the developers released two models for download - BERT-base and BERT-large [40]. Each model was pre-trained on BookCorpus (corpus containing 800M words) [48] and English Wikipedia (2,500M words). Several other pre-trained models are available for download, contributed by the machine learning community.

Extending BERT to perform supervised classification tasks can be easily implemented by adding a single-layer neural network with softmax as activation function above the encoder stacks. To extend it to multitask classification one adds as many output neurons as there are targets/classes. In figure [figure cite] one can see the architecture of the BERT model performing a classification task.

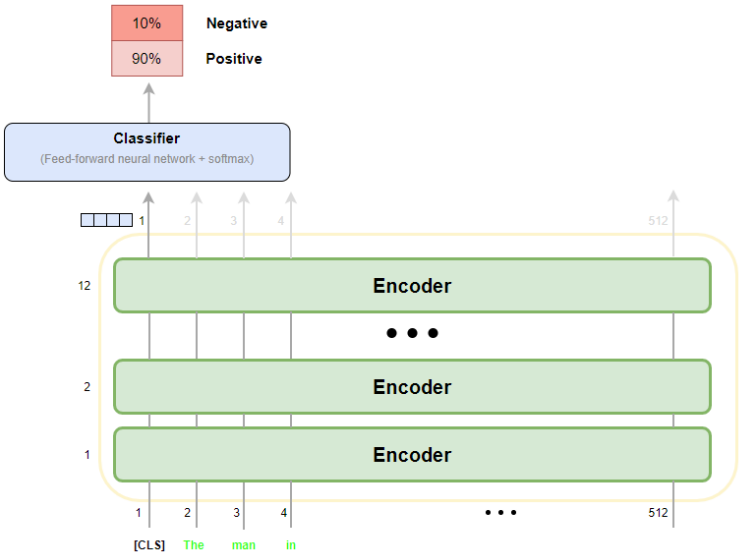


Figure 4.10: BERT model architecture of a classification example.

Chapter 5

Related work

In this chapter, we will present a short review of three papers that resemble our own work: (i) *A machine learning-based framework to identify type 2 diabetes through EHR*, (ii) *Recurrent Neural Networks for Multivariate Time Series with Missing Values*, (iii) *Scalable and accurate deep learning with electronic health records*. Each paper presents state-of-the-art techniques and models for handling prediction tasks on EHR data. The chapter has a natural progressive structure, starting with Paper 1 (Section 5.1), containing traditional models and feature engineering, before moving on to advanced solutions related to deep-learning in Paper 2 (Section 5.2) and Paper 3 (Section 5.3). Together, the survey paints a picture of what the state-of-the-art is, and points out some of the main current obstacles and challenges for successful use of machine learning for EHR analytics.

5.1 Paper 1: *A machine learning-based framework to identify type 2 diabetes through EHR*

5.1.1 Introduction and motivation

Type 2 Diabetes Mellitus (T2DM) is increasing steadily all over the world and is fast becoming an epidemic in some countries [49]. T2DM is a lifestyle disease and most people are undiagnosed early in the diabetes disease process. Increased education about symptoms of diabetes may lead persons to seek out healthcare providers early [50]. The group of researchers in this paper [51] tried to discover diverse features in EHR data associated to T2DM by identifying patients with and without T2DM from EHR via feature engineering and machine learning. This was done in the hope of detecting more associations between features that previous machine learning approaches for identifying patients with T2DM could have missed.

5.1.2 Methods, materials and results

Their investigation focused on three years of EHR data (from 2012-2014), collected from 10 local EHR systems in Changning, Shanghai, ending up with data from 123,241 patients. A filtering strategy was used to pre-select patients whose EHR data was related to diabetes. After filtering they ended up with 23,281 patient samples with diabetes-related information. The pre-selected patients were selected if they satisfied one of three criteria:

1. Diabetes-related diagnosis
2. Diabetes-related medication
3. Diabetic laboratory test

Labeling the entire dataset with disease lables would have required an immense manual work effort. Instead, they picked 300 random samples out of the 23281 patient samples which a pair of clinical experts labeled. The dataset was labeled into three categories by a pair of experts: case (diabetes), control (healthy), and unconfirmed. The unconfirmed cases had severely incomplete EHR documentation and had to be dropped to reduce negative influences on models. In the end, they ended up with 221 samples (161 cases and 60 controls).

Feature construction In their work, they included extra features that were derived from the EHR data to make case/control identification more accurate. These features were not present in previous state-of-the-art machine learning models predicting diabetes. Constructing new, informative features from EHR is usually a must to guarantee a high prediction performance for machine learning models. These features were extracted with the use of feature engineering (feature selection and feature extraction) from seven different sources: *demographic information*, *communication report*, *outpatient diagnosis report*, *inpatient diagnosis report*, *inpatient discharge summary*, *prescription report*. Some of the sources had the same type of features and were highly correlated and could be summarized into 36 features. The features within these sources were also correlated and could be further trimmed down to the final 8 features.

Machine learning For classification, they used several popular classification models such as K-Nearest-Neighbors, Naïve Bayes, Decision Tree, Random Forest (Section 3.5.1), Support Vector Machine, and Logistic Regression to model patterns of cases and controls based on their extracted features. Each classifier was constructed using 4-fold cross-validation and compared using area under the receiver operating characteristic (ROC) curve

(AUC), a metric that demonstrates the trade-off between false positive and true positive rates.

We have already explained the basic principles of Decision Tree and Random Forest in Section 3.5.1 above. We'll quickly go through the other models here.

K-nearest-neighbor

K-nearest neighbor (KNN) is a model that classifies input variables by doing a majority vote of its neighbors. The case is assigned to the majority class measured by a distance function (similarity measure). If there is only one neighbour, then the object is assigned to the neighbor's class that is closest to the point. The number of neighbors that is considered for the majority vote in the algorithm is a number K. K is a hyperparameter that must be set in the beginning and is usually chosen by first inspecting the data. A large K usually is more accurate and reduces the variance that exist in the data. There is no perfect way of finding optimal K, but can be retrospectively determined using hyperparameter tuning algorithms like cross-validation with the use of a validation dataset [52].

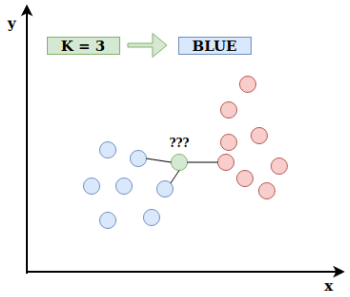


Figure 5.1: An example of how KNN classification works.

Naïve Bayes

Naïve Bayes is a machine learning classification model that is both popular and simple. It is based on Bayes Theorem, a theorem that calculates probabilities and conditional probabilities. The algorithm performs the classification process by assuming that the presence of a feature is unrelated to other features presence. It calculates the probability of a data point (x) belonging to each class by the use of the Bayes theorem equation and outputs the class with the highest posterior probability [53]. The Bayes theorem calculates posterior probability $P(c|x)$ from P(c), P(x), and $P(x|c)$.

$$P(c|x) = \left(\frac{P(x|c)P(c)}{P(x)}\right)$$

Support vector machine

A Support Vector Machine (SVM) is a model that can be used for both classification and regression tasks. The algorithm marks each data point into one of two classes. To find out which data point belong to each class the algorithm finds a hyperplane that differentiates the data points of both classes by the largest possible margin [54]. Imagine that each variable is plotted in an n-dimensional space (n is the number of features in the dataset), where each feature value is a particular coordinate. Figure 5.2 illustrates the idea.

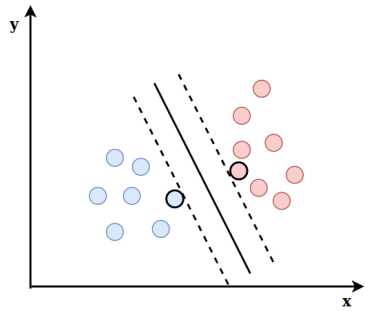


Figure 5.2: Linear SVM classification. Finding the hyperplane that differentiates the classes by the largest possible margin.

Logistic Regression

Logistic regression is named after the logistic function, often referred to as sigmoid function in machine learning. This function is used at the core of the algorithm to calculate a probability value between 0 and 1 that can be mapped to two or more discrete (only specific values or categories are allowed) classes [55]. If the probability value of a data point towards a certain class exceeds a set threshold it is categorized as that particular class. The sigmoid function used to map predictions to probabilities can be viewed below.

$$S(value) = \frac{1}{1 + e^{-value}}$$

Results The project detailed in this paper yielded impressive results, beating the current best machine learning models at the time. Their most stable and accurate models were RF, Decision Tree, and SVM. These models perform well on uncorrelated data and benefit strongly from feature engineering. These three models scored over 0.95 in AUC performance metric, beating the state-of-the-art machine learning models at that time which scored 0.71 in AUC [56]. In their research, they experienced that models predictive performance increased when features were abstracted into higher layers, demonstrating the importance of feature engineering in EHR data

for their approach. However, good feature engineering on EHR datasets requires a deep understanding of medicine and medical terms.

Although they achieved promising results there are some deficiencies with their approach. Perhaps the largest uncertainty of their approach relates to the data used. They used very few samples, and would need more to confirm the scalability and robustness of their models. The Chinese EHR data used might not generalize well on western EHR data. Since their approach for extracting samples requires medical expertise, extending this to larger datasets would be costly.

5.2 Paper 2: *Recurrent Neural Networks for Multivariate Time Series with Missing Values*

5.2.1 Introduction and motivation

Machine learning models are constrained by the quality and quantity of data. They are sensitive to bad data, data containing missing, incomplete, inaccurate, nonrepresentative values. Bad data is a huge discussion field in EHR analytics. A large part of EHR data is comprised of so-called multivariate time series data, data that has been collected by observation and analysis of patients. This data is often observed and collected at irregular intervals because of medical events, saving costs, anomalies, and so on. Thus they contain a variety of missing or incomplete values, heavily affecting machine learning models performances.

There are several different ways of dealing with missing values. The most common is to drop columns of data that consist mostly of missing values. Another common approach is to replace the missing values with the median of that column's values. Although such basic techniques are often better than leaving missing values in the data, they generally don't work as well on EHR data. The researchers in this paper try to tackle this issue by developing a new deep learning model called GRU-D, based on Gated Recurrent Unit (GRU) [16]. They trained the model on MIMIC-III and the PhysioNet datasets and tested their results on experiments of time series classification tasks:

- *Mortality task*: Predicting if a patient is going to die in the hospital. They predicted this binary classification task on both of the datasets.
- *Four tasks*: ICU-mortality, in-hospital mortality, length-of-stay, prolonged length of stay > 3, cardiac condition, recovering from surgery (true or false). This was treated as a multiclassification task on the PhysioNet dataset.
- *ICD-9 Code tasks*: Predicting 20 different ICD-9 diagnosis categories as a multi-task classification problem.

5.2.2 Methods, materials and results

GRU-D The GRU-D model uses the GRU-D component at each time step. The model uses trainable decays at both input variables and the hidden states of the gated recurrent units to handle missing values in the dataset, thus the name GRU-D. A trainable decay these places in the model is motivated by patterns in medical data and RNN's. A missing value in multivariate time series data is often closely related to the last observed value, and as time goes on the relevancy of this value diminishes and will fade away. By mimicking this behavior by adding two learnable decay rates in the model architecture GRU-D, can learn the missing value patterns in the data.

PhysioNet Challenge 2012 A dataset made for the PhysioNet Computing in Cardiology Challenge in 2012, where contestants were trying to predict ICU mortality of patients. The dataset contains records from over 12,000 adult ICU stays, where each patient has stayed over 48 hours. It has 42 variables where each variable has been collected at least once. Six of the variables are general descriptors of the patients and the rest are multivariate time series data, containing one or more observations [57].

MIMIC-III MIMIC critical care database is the largest and most-cited publicly available EHR database. The database comprises information relating to patients admitted to ICU at Beth Israel Deaconess Medical Center. It consists of 60,000 ICU admissions, including demographics, vital signs, laboratory tests, medications, free-text notes, and more. The database is accessed through an application process. The applicant has to pass a training course on biomedical research and research conduct [12]. We discuss MIMIC-III in more detail in Part II of the thesis.

Results To test the GRU-D model's effectiveness on multivariate time series data they compared it to several classification models: RNN, LSTM, logistic regression, support vector machines (SVM), and random forest. All of the models they compared against had the missing values imputed using the mean value of missing values, and were scored using the AUC metric. Their proposed GRU-D model outperformed all of the models in every classification task they tested on the two datasets while having similar running time and space complexity. With these results, they prove that their model captures informative missingness in multivariate time-series data and pull off performances above regular non-deep learning methods. However, their GRU-D model struggled when there was little correlation between the missing values and the prediction task, and they saw patterns that their model might fail or provide limited improvements to these problems. Therefore

their approach requires more research and testing to find its limits and strengths.

5.3 Paper 3: *Scalable and accurate deep learning with electronic health records*

5.3.1 Introduction and motivation

In this work, Google, in collaboration with resarchers at the Universitiy of California, University of Chicago Medicine, and Stanford University, built a machine learning application aiming to improve medical care. Their goal was to predict medical outcomes across a wide range of clinical problems, based on a very large dataset. The medical outcomes they focused on were mortality, readmissions, length of stay, and discharge diagnoses [58].

In recent years the amount of healthcare data has exploded, both in volume and complexity. Most of this data is yet to be used in predictive statistical models. Traditional predictive modeling techniques require a custom dataset with specific variables for each outcome to be predicted. Resulting in many of the potential predictor variables in the EHR might be discarded, particularly the clinical free-text notes from nurses and doctors. These notes are clinical summaries of patients and contain a vast amount of information.

The researchers tried to tackle this problem by not creating a custom dataset for each predictive outcome, but a single data structure that could be used for all predictive outcomes. Instead of harmonizing the data and removing specific variables with preprocessing, they get the statistical predictive models to learn how to harmonize data with the help of new deep learning NLP models and techniques. These deep learning NLP models do not require specification in the preprocessing step of which potential variables to consider, and in what combinations. The neural networks can learn representations of the most important factors and interactions from the data. They also thrive on large volumes of data and can handle unstructured data containing noise.

5.3.2 Methods, materials and results

For their research, they used EHR data from two US academic medical centers, the University of Chicago, San Francisco (USCF) from 2012 to 2016, and the University of Chicago Medicine (UCM) from 2009 to 2016. The EHR data from each hospital were de-identified but otherwise kept intact, except for the addition of two features in the UCM dataset. In the UCM dataset, they kept the dates of service from the de-identification process and added the de-identified, free-text medical notes. Combined the data for their work consisted of 216,221 adult patients that had been hospitalized for more than

24 hours. This resulted in a total of 46 million data points, including the clinical notes.

To fuse the two EHR datasets they developed a preprocessing pipeline that takes EHR data as input and produces FHIR outputs. FHIR represents clinical data in container format that is consistent, hierarchical, and extensible. Making it easier to exchange data between organizations and sites. However, in their work they didn't harmonize the data. Meaning that they did not perform any data quality improvements like dealing with missing variables, duplicate variables, perform feature engineering, etc.

FHIR Fast Healthcare Interoperability Resources, also known as FHIR is a standard for exchanging healthcare information electronically and is developed by Health Level Seven International (HL7). It aims to "simplify implementation without sacrificing information integrity" [59]. FHIR enables this by ensuring that EHR is available, discoverable, and understandable while containing the data structured and standardized.

FHIR is built upon resources, which are modular components of any content that is exchangeable. They act as building blocks that can be assembled into existing systems. The framework aims to be able to extend and adapt resources, to be implemented into any system, independent of how the existing system was developed. Each resource is associated with a unique identifier, making it possible to access information in interest from any application [59].

Machine learning To learn the patterns in their datasets required to predict their target tasks they used three architectures: LSTM, An Attention-Based Time-Aware Neural Network (TANN), and Neural Network with boosted time-based decision stumps. Each model were trained on all four tasks and multiple time points (e.g., before admission, at admission, 24h after admission, and at discharge). For every prediction they used all information available in the EHR up to the predictions were made. The results from each model was in the end combined using ensembling [60]. For information regarding LSTM see 4.6.

Results The produced predictions from their models outperformed every existing EHR model at that current time for predicting mortality (0.92-0.95 vs 0.91) [61], unexpected readmission (0.75-0.76 vs 0.69) [62], and increased length of stay (0.85-0.86 vs 0.77) [63]. Proving the usefulness and ability of deep learning models trained on the entirety of EHR datasets, without the hand-selection of variables by experts. By using the entirety of EHR datasets the models also promote scalability by exposing more data which can be used to make an accurate prediction. Nevertheless, the paper is still a case study and prospective trials are needed to demonstrate that it can be

used to improve care. Future research is also needed to see how the models would generalize on EHR data from different EHR systems and geographical sites. The models in this paper also require intensive computational power and require specialized expertise to develop, which can prevent it from being implemented to improve care as it might be seen as a black-box by health personnel. Unfortunately, the dataset used in their study is not publicly available, because of privacy and security concerns. The work is therefore not easily reproducible, but acts as a gauge of where the state-of-the-art is, and as a guide for other researchers.

5.4 Discussion

The three selected papers introduce techniques and models for performing classification tasks, trained on EHR data to predict important medical tasks and reaching impressive state-of-the-art results. In summary, they contain the information and approaches to perform a successful machine learning project related to EHR data. A common thread in these papers is the focus on preprocessing of the EHR data to prepare it for machine learning, one of the most challenging tasks when working with this kind of data. In Paper 3 we can see the most promising solution to this problem, FHIR [59], which has the potential of streamlining the task and make EHR data more available for second-hand usage. However, there is still research that needs to be performed. A common problem with all of the approaches in these papers is to figure out how their applications would work with geographical different EHR data. To really find out how well EHR machine learning applications can work in healthcare, the creation and sharing of EHR data between organizations worldwide is necessary.

Part II

Experiments

Chapter 6

Machine learning models based on MIMIC-III

6.1 Introduction

Machine learning algorithms applied to EHR datasets for predicting medical events and mining clinical insight has been demonstrated to potentially provide powerful tools to aid clinicians in their work [58, 16, 17]. It's a challenging and time-consuming process for clinicians to get a full overview of patient EHR data. Time becomes critical when a patient is admitted to the ICU, where quick clinical insight into a patient's health data can be life-saving and aid health organizations to structure and plan better.

In order to develop algorithms to perform these tasks, large amounts of data is required. Unfortunately, EHR datasets are difficult to access without a vigorous screening process from the health organizations, or even altogether impossible, because of the sensitivity of the data. Luckily, there are some datasets available that contain real, non-fabricated data that has been anonymized to protect patients and clinicians' identities.

In this chapter we experiment with predicting valuable medical information from the MIMIC-III critical care database, using traditional machine learning algorithms. The experiment revolves around the prediction of two cases; mortality and length of stay. Both cases need to be assessed early when a patient is admitted to a hospital. By trying to predict the mortality of a patient, the trained model can give us an indicator if a patient is going to pass away during the hospital visit. A critical indicator that the patient needs immediate treatment and to be prioritized by health personnel [64]. Length of stay is a factor that helps assess the efficiency of hospital management and patient quality of care, by getting an estimate of how long a patient is going to be hospitalized [65][66]. To predict these clinical tasks we will try two ensemble tree classifiers, Random Forest and XGBoost. Ensemble tree classifiers are robust in the handling of noisy data. A problem

that is often faced in EHR data.

This experiment will be used as an indicator to see if it's possible for a researcher to extract valuable information out of EHR data using regular machine learning algorithms and standardized preprocessing techniques. By doing so we hope to expose and shed light on some of the difficulties and problematic areas of performing EHR analytics with machine learning algorithms on real EHR data from a hospital.

6.2 Methods and materials

Dataset

In this experiment, we used the publicly available MIMIC-III critical care database, the largest and most-cited publicly available EHR database. The database contain information from patients admitted to ICU at Beth Israel Deaconess Medical Center between 2001-2008. It consists of $\sim$ 60,000 distinct intensive care unit admissions, including demographics, vital signs, laboratory tests, medications, discharge summaries and reports, and more. It contains a mean of 4579 charted observations and 380 laboratory measurements for each unique hospital admission. Each table in the relational database is linked through IDs, SUBJECT_ID links to a unique patient, HADM_ID links to a unique hospital admission, ICUSTAY_ID links to unique ICU admission.

The data contained in the records have been deidentified using data cleansing and date shifting. This was performed in accordance with the Health Insurance Portability and Accountability Act (HIPAA) [67]. Eighteen identifying data elements had to be removed to deidentify EHR data. The elements removed/shifted from the database include fields like name, dates, addresses, and telephone numbers.

The database is accessed through an application process. The applicant has to complete a training course on biomedical research and research conduct (https://mimic.physionet.org).

Preprocessing

To process the database into datasets we used MIMIC-Extract, an open-source data extraction and processing pipeline for MIMIC-III. The pipeline addresses three primary challenges to make EHR data prepared for machine learning algorithms: standardization of data through data processing functions (unit conversion, outlier detection, and aggregating semantically equivalent functions), preservation of time series nature of clinical data, and extendability for researchers with similar questions [68]. It's inspired by two similar efforts [69, 70] to make MIMIC-III data preprocessing more accessible for researchers. All code is publicly available and can be customized

after the researcher's needs.

The preprocessing process starts with converting variables to the same measuring units, as patient vitals and lab results were recorded with different measuring units. Then outliers are handled by taking clinically assessed variable ranges (thresholds) made by experts knowledge and associate them with each numerical variable. If the variable is outside the given range it is marked as missing and dropped. The pipeline also applies one extra refined threshold that detects variables outside a physiological valid range. These values are replaced by the nearest valid value.

Another important issue the pipeline addresses is the multivariate time-series aspect of labs and vitals data. The pipeline handles these variables by merging labs and vitals together, before further merging them into hourly buckets to reduce duplicated variables and data missingness. The benefit of the approach is justified in the paper: "Feature robustness in non-stationary health records" [71]. Furthermore, they add binary indicators if common treatments have happened within the hour by adding extra dimensions to the data. Each unique patient then has 24 rows, obtained by transforming the dataset into hierarchical indexing to manipulate and store the data in lower-dimensional structures. All patients that have stayed less than 30 hours are discarded. An excerpt of the final dataset can be seen in figure 6.1.

		LEVEL2	alanine aminotransferase									...	white blood cell count urine					
		Aggregation Function	mask									...	time_since_measured					
		hours_in	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19
subject_id	hadm_id	icustay_id																	
3	145834	211552	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	6.0	7.0	8.0	9.0	10.0	11.0
6	107064	228232	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	30.0	31.0	32.0	33.0	34.0	35.0
9	150750	220597	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	54.0	55.0	56.0	57.0	58.0	59.0
11	194540	229441	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	78.0	79.0	80.0	81.0	82.0	83.0
12	112213	232669	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	102.0	103.0	104.0	105.0	106.0	107.0

Figure 6.1: The hierarchical indexing dataframe (dataset) after pre-processing with MIMIC-Extract pipeline.

Oversampling

A challenge with the MIMIC-III dataset is the highly imbalanced classes, specifically in ICU-mortality, hospital-mortality, and length of stay > 7. In order to deal with this problem, we oversampled the minority classes by duplicating them to increase their signal. We duplicated each sample six times. This technique is called oversamplig and prevents the machine learning algorithm from being biased to predict one class in the majority of the cases. After performing oversampling on the three unbalanced classes we

had ∼24000 samples in each training dataset where ∼7000 were duplicates of subjects containing that class. Length of stay > 3 was already balanced and no oversampling was needed.

	Before upsampling	After upsampling
Training data	15,554	23,961
Labels in training data	1,201	7,488

Figure 6.2: ICU-mortality label/class imbalance in data before and after upsampling.

Our machine learning approach

To perform the EHR analytics classification tasks we picked two ensemble tree machine learning algorithms, Random Forest [72] and XGBoost [30] (Section 3.5). Ensemble three models are known for being robust and capable of achieving good prediction results even with low correlation, missing features, and outliers in the data [73, 74], all common issues for EHR data.

We split our dataset into training (70%), validation (10%), and test set (20%). Each model's hyperparameters were tuned using Sklearns Cross-Validation [72] and their performance was evaluated on four different metrics: accuracy, F1-score, Area Under the Receiver Operating Characteristic Curve (ROC AUC), and Average Precision Score. The model with the best generalization ability on the prediction task was then tested on the test set to give the final scores on each task.

6.3 Experimental results

All of the models were trained using the same dataset and scored using the same evaluation metrics. At the start of the experiment, we fine-tuned the Random Forest classifier on all four tasks. Still, Random Forest didn't perform well. The models did not manage to capture relationships in the time series data, as a result of our preprocessing strategy where we merged vitals and labs into hourly buckets. Random Forest cannot capture the trends in the time-series data if the order of lines are important. The performance metrics results are shown in table 6.1. We can see that the Random Forest models perform significantly worse than the XGBoost models, with an exception of length of stay > 3. To confirm our theory we can look at the models confusion matrices in figure 6.3. By looking at the matrices we can see that the models have a small percentage of correctly predicted labels.dagbladet.no/

We therefore tried a more complex model, XGBoost to see if we could better capture the relationships in the time series data.

Task	Model	acc	F1	AUROC	AUPRC
ICU-Mort	RF	93.0%	20.8	88.6	45.2
	XGBOOST	93.1%	46.9	89.9	50.0
ICU-Hosp	RF	90.4%	26.3	86.5	46.5
	XGBOOST	89.9%	47.8	88.6	50.6
LOS > 3	RF	69.2%	59.4	73.5	69.6
	XGBOOST	69.1%	58.1	67.0	58.1
LOS > 7	RF	92.0%	0.1	76.3	21.3
	XGBOOST	90.0%	18.5	76.7	21.4

Table 6.1: Table of the models performance metrics results on the validation set.

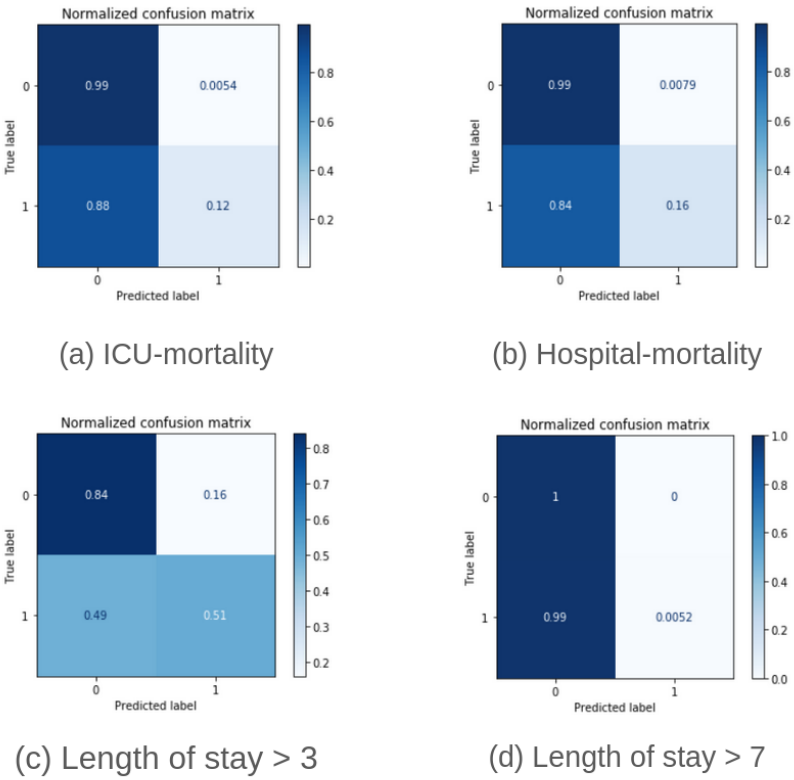


Figure 6.3: Random Forest models normalized confusion matrices showing the models performances on the validation set for all four classification tasks.

Our fine-tuned XGBoost models yielded much better results, especially in predicting ICU-mortality and Hospital Mortality, reaching an AUROC score of ∼89 and increasing the F1 score by 50% in both cases, when evaluated on the validation set. An overview of the full metric scores can be seen in table 6.1 and the confusion matrices can be seen in figure 6.4. Furthermore, the model was able to capture some relations in prolonged length of stay greater than seven days. A classification task the Random Forest model struggled greatly with as we can see in figure 6.3.

By getting these promising results we trained our fine-tuned XGBoost models on the entire dataset and got the final generalization properties of the models by predicting on the test set.

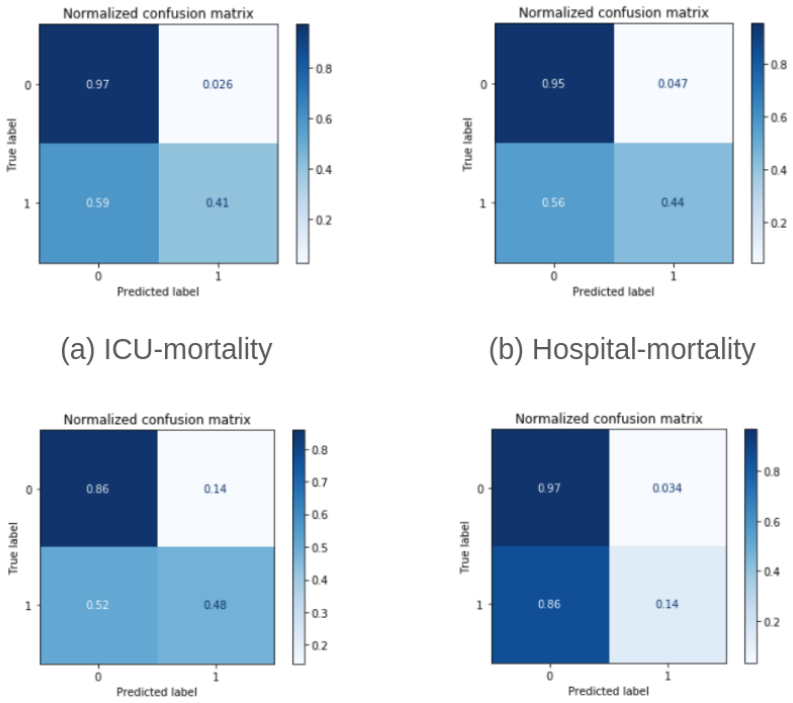


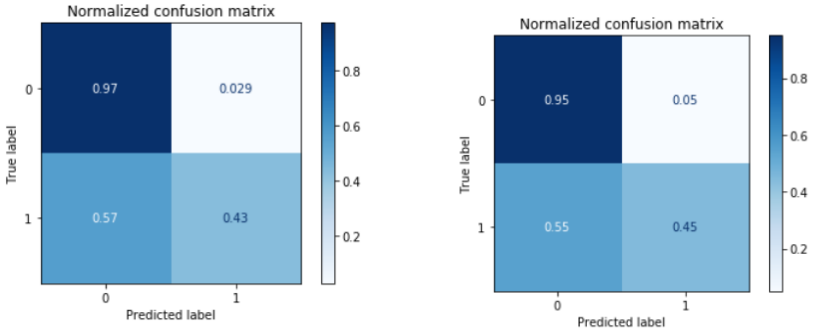
Figure 6.4: XGBoost models confusion matrices showing the models performances on the validation set for all four classification tasks.

In the end we ended up with the metric scores seen in table 6.2 and the number of correct prediction seen from the confusion matrices in figures 6.5 and 6.6. The generalization results for both cases of mortality and length of stay > 3 are promising. However we can quickly see by the case of length of stay > 7, a prediction case very sensitive to time-series data that our model cannot quite capture long term relations in time-series. This could

be an effect of the way we arranged vitals and labs into hourly buckets in the preprocessing step.

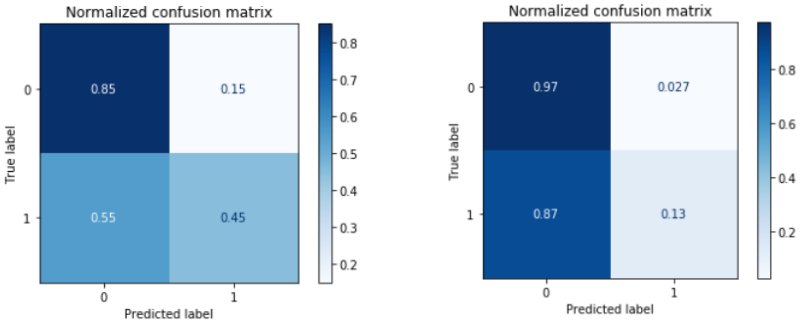
Task	Model	acc	F1	AUROC	AUPRC
ICU-Mort	XGBoost	93.5%	47.1	91.0	48.8
ICU-Hosp	XGBoost	89.9%	42.0	65.4	25.5
LOS > 3	XGBoost	68.3%	54.3	72.2	65.2
LOS > 7	XGBoost	90.7%	18.1	73.2	19.5

Table 6.2: Table of the models performance metrics results on the test set.



(a) ICU-mortality (b) Hospital-mortality

Figure 6.5: ICU-mortality and hospital mortality confusion matrices showing the models performances on the test set.



(a) Length of stay > 3 (b) Length of stay > 7

Figure 6.6: Length of stay > 3 and length of stay > 7 confusion matrices showing the models performances on the test set.

6.4 Discussion

The experiments we conducted did not yield results that reached the performance of other previously existing case studies performed on the MIMIC-III database to perform classification on the same four target tasks. Several other case studies on the MIMIC-III data have gotten better results [75, 76, 77, 78, 79].

However, comparisons to other studies are difficult since the results are greatly affected by the preprocessing each study has performed. The effect of this process on EHR data can be seen in the paper *Early hospital mortality prediction using vital signs* [80] published by Reza Sadeghi and his coworkers in 2018. Their case study on mortality prediction is the current state-of-the-art mortality prediction using MIMIC-III data. They reached an F1, recall, and precision score of 97% using Random Forest. A model that we struggled with, likely because of the arrangement of time series data within the dataset. Thus, indicating the significance of preprocessing to extract important features for the target task.

Preprocessing of EHR data is a time-consuming task that requires domain knowledge to perform it well. We based our experiment on the preprocessing pipeline MIMIC-Extract [79] that transforms the database into datasets that can be used in time-series predictions. By having similar preprocessing it's most relevant to compare our results with their benchmark results on the same four prediction tasks from their paper. They got their top results using the GRU-D model that we reviewed above in Section 5.2 [16]. With this model, they reach 89.1 in ICU Mortality, 87.6 in In-hospital mortality, 73.3 in length of stay > 3, and 71.0 in length of stay > 7 in AUROC (Area Under the Reciever Operating Characteristics). We beat their scores by a slight margin using a fine-tuned XGBoost model, as can be seen in table 6.2.

We expect that our scores would be significantly higher with a better preprocessing technique. However, this would require us to acquire a deeper clinical understanding of the data, which was out of the scope of this study. We also firmly believe that the use of deep learning models could better capture the relations in the time-series data, which could result to better performances.

Chapter 7

Predicting ICD-9 codes from clinical free text notes

7.1 Introduction

Another part of EHR that contains important pieces of information about the patient's health state is the unstructured data, more specifically the freely written clinical notes, transfer notes, and discharge notes, written by nurses and clinicians. These notes are written at different time periods of a patient's hospital stay and give us clinical stories from a patient, written in a clinical narrative in free-form text. The most impactful of these notes are the discharge notes. They are a summarization of patient information before their care is transferred from one clinician to another. Medical notes are essential for clinicians to be able to perform coordinated care and practice effective coordination. The ability to automate the process of extracting information out of these notes in secondary use of EHRs have the potential to save time, aid clinicians in decision making, remove medical errors, and automate coding, an important administrative task [81, 82]. The automation of information extraction from clinical notes is thus essential to unlocking the full potential of EHR data.

Recent advancements in NLP indicate that deep learning can extract associations in unstructured data, leading to great results [33], also for associations between unstructured text and structured data in EHR. NLP can convert data from medical notes into structured text that can be performed analysis on and has the potential to be an invaluable tool for healthcare professionals. This approach has seen great success in information extraction from medical notes from EHRs in the last decade [83, 84, 85, 86].

However, clinical narratives are often informal, and without clear instruction of how to formulate them. The result of the lack of structure and instructions make them prone to errors. Summarization of patients has therefore led to miscommunication between clinicians and even medical

errors in some cases. [87]. The lack of structure and expression variances also makes it hard to perform NLP on them. NLP benefits from similar structure and content in text documents to effectively learn patterns and relationships.

In this chapter, we will investigate the effect of deep learning in combination with NLP's ability to perform multi-label text classification from clinical free form notes. The experiment will revolve around two state-of-the-art deep learning NLP architectures, ULMFiT [1] and BERT [40]. Each model will be trained on discharge notes from the MIMIC-III database [12]. The models will perform multi-label text classification (automatic coding) on ICD-9 codes.

7.2 Methods and materials

The objective is to extract clinical information from unstructured free-text clinical notes. Thus, only the notes section from the `noteevents` table was extracted from the MIMIC-III database. We set our focus on the discharge summaries/notes because they are labeled with ground truth alongside the free-text. Discharge summaries are also written at the end of a patient's stay and thus contain all relevant diagnoses. The data were then preprocessed into a dataset by building on the hierarchical nature of ICD-9 codes. The dataset was comprised of only ICD-9 codes that had over 1000 occurrences, thereupon discharge notes that didn't contain a single occurrence of one of these ICD-9 codes were dropped. Out of the 59.652 hospitalization discharge notes contained in the MIMIC-III database 8.352 were dropped and we were left with 51.300 discharge notes that satisfied the code frequency threshold that was set. As a result of preprocessing after the hierarchical nature of ICD-9 codes and setting an occurrences threshold, we were left with 139 unique ICD-9 codes that we could perform multi-label text classification on.

The raw text in the discharge notes was then further preprocessed to improve their structure by remove text annotations that could negatively affect the machine learning models. We improved the raw text by removing HTML tags, certain punctuations and numbers, and multiple spaces. Thereupon the dataset was split into training (80%) and validation(20%).

Before our models could be trained we had to convert the data into a format the models could operate on, i.e. tokenization and numericalization. Tokenization is the process of splitting a string into a list of tokens, and numericalization is the process of transforming these tokens into numbers. From there the numbers can be fed to embedding layers, converting them into arrays of floats before they are passed through one of our models.

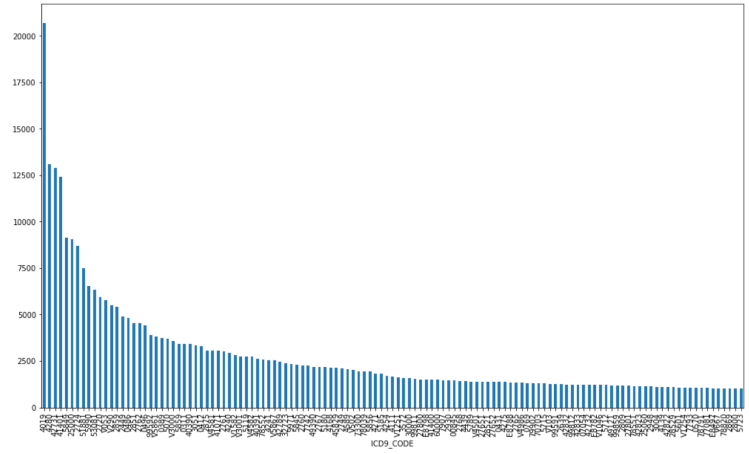


Figure 7.1: ICD-9 codes arranged by frequency within the discharge notes

"(Admission Date: [**2117-5-20**]   Discharge Date: [**2117-5-26**]\n\nDate of Birth: [**2036-11-1*
*]   Sex: F\n\nService: CARDIOTHORACIC\n\nAllergies:\nPatient recorded as having No Known Allergies to
Drugs\n\nAttending:[**First Name3 (LF) 1505**]\nChief Complaint:\nChest pain with exertion\n\n\nMajor Surgical or
Invasive Procedure:\ns/p CABGx3(LIMA->LAD, SVG Y->OM, Diag) [**5-20**]\n\n\nHistory of Present Illness:\nThis is a
n 80-year-old female who presented with chest pain and\nwas found to have a ST-segment elevation myocardial functi
on.\nShe underwent a cardiac catheterization, and this demonstrated\n3-vessel disease. She underwent a bare-\nmeta
l stent of the right coronary artery. We were asked to\naddress the other stenoses.\n\nPast Medical History:\n1. H
ypertension\n2. Rheumatic heart disease as child without any residual murmur\n3. Diverticulitis\n4. Right breast c
ancer status post lumpectomy [**2103**]\n5. Bladder suspension [**2108**]/TAH\n6. CAD, s/p STEMI/stent [**4-7**]\
n\nSocial History:\nPt denies tobacco use. She drinks one glass of wine daily\n\n\nFamily History:\nFather had an
MI at age 59. No sudden death in the patient's\nfamily.\n\nPhysical Exam:\nGEN: 80 y/o in NAD\nHEENT: NCAT, PERR
L, OP Benign\nNECK: Supple, FROM, No JVD\nLUNGS: CTA\nHEART: RRR, No C/C/E\nABD: Benign\nEXT: Warm, well perfused.
[**2-2**]+ pulses, Varicosities noted.\nNEURO: Nonfocal, no carotid bruits.\n\nDischarge\nVitals 98.0, 143/64, 66
SR, 20, RA sat 93%, 67.5 kg\nNeuro A/O x 3 nonfocal\nPulm CTA bilat\nCardiac RRR no M/R/G\nAbd soft, NT, ND, +BS B
M [**5-25**]\nExt warm trace edema pulses palpable\nInc sternal healing no erythema no drainage sternum stable\nLe
ft leg EVH healing thigh ecchymotic\n\n\nPertinent Results:\n[**2117-5-26**] 07:10AM BLOOD Hct-34.5*\n[**2117-5-25
] 07:10AM BLOOD WBC-9.5 RBC-3.38*# Hgb-10.8*# Hct-32.2*#\nMCV-95 MCH-32.0 MCHC-33.6 RDW-15.8* Plt Ct-305\n[211
7-5-20**] 11:29AM BLOOD WBC-7.9 RBC-2.79*# Hgb-8.9*# Hct-25.9*#\nMCV-93 MCH-31.9 MCHC-34.4 RDW-13.1 Plt Ct-155\n[*
*2117-5-26**] 07:10AM BLOOD PT-12.3 INR(PT)-1.1\n[**2117-5-25**] 07:10AM BLOOD Plt Ct-305\n[**2117-5-20**] 11:29AM
BLOOD PT-15.2* PTT-39.4* INR(PT)-1.4*\n[**2117-5-20**] 11:29AM BLOOD Plt Ct-155\n[**2117-5-20**] 11:29AM BLOOD Fib
rino-115*\n[**2117-5-25**] 07:10AM BLOOD Glucose-97 UreaN-10 Creat-0.6 Na-136\nK-4.2 Cl-102 HCO3-24 AnGap-14\n[**2
117-5-20**] 12:42PM BLOOD UreaN-6 Creat-0.5 Cl-109* HCO3-22\n[**2117-5-23**] 08:45AM BLOOD Calcium-8.0* Phos-2.4*
Mg-2.2\n\nCHEST (PA & LAT) [**2117-5-24**] 3:44 PM\n\nCHEST (PA & LAT)\n\nReason: evaluate effusion\n\n[**Hospita
l 93**] MEDICAL CONDITION:\n80 year old woman with\nREASON FOR THIS EXAMINATION:\nevaluate effusion\nTWO-VIEW CHES
T:\n\nCOMPARISON: [**2117-5-22**].\n\nINDICATION: Pleural effusion.\n\nPatient is status post recent median sterno
tomy and coronary\nartery bypass surgery. Cardiac and mediastinal contours are\nstable in the postoperative perio
d. There is improving bibasilar\natelectasis. Small pleural effusions are again demonstrated,\nwith apparent sligh
t increase in the left pleural effusion,\nalthough positional differences slightly limit comparison. A\ntiny retro
sternal gas collection on lateral radiograph, likely a\nnormal postoperative finding given the recent surgery.\n\n

Figure 7.2: An example of a discharge note contained in MIMIC-III.

ULMFiT. Our ULMFiT model was trained using the *three-step process* explained in their paper [1] by utilizing the fastai layered API for deep learning. An API that provides high-level components to practitioners, thus providing them with a flexible tool to more easily develop state-of-the-art deep learning results [88]. A visualization of the *three-step process* can be seen in figure 7.3. The training process starts by using ULMFiT's pre-trained language model, trained on WikiText-103 [44]. The pre-trained language model is then fine-tuned to our target corpus using *discriminative fine-tuning* and *slanted triangular learning rates*. At this stage, our language model has learned semantics in the medical texts by learning to predict the next word in a sentence. After fine-tuning the language model we remove its last layer and add two linear layers. These two layers provide a probability distribution so we can perform multi-label text classification. The ULMFiT classification model was tuned using *gradual unfreezing*. We started out freezing all the layers in the model and gradually unfreezing one layer at the time and

fine-tuning it for one epoch starting from the last layer. This process was repeated for two more times before we unfroze all layers and fine-tuned the entire model for seven epochs. At each fine-tuning step of the classifier we used *discriminative learning rates* and *slanted triangular learning rates*.

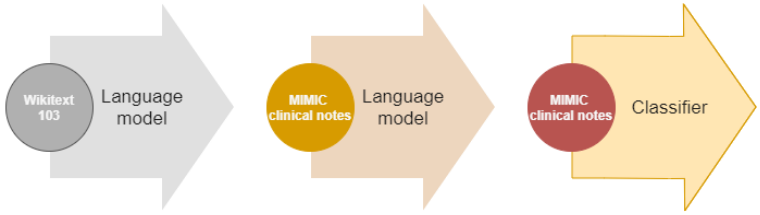


Figure 7.3: The *three-step process* of ULMFiT on medical notes.

BERT. Training BERT's language model from scratch is a computationally and time-intensive task [40]. Similarly to ULMFiT's process, we used a pre-trained BERT model named BlueBERT (Biomedical Language Understanding Evaluation BERT model) [89]. This BERT model has been pre-trained on language representation in the biomedicine domain. More specifically it was pre-trained on PubMed abstracts containing ∼4000M words [90] and clinical notes from MIMIC-III to predict the value of the masked tokens. We then fine-tune the BERT model so it outputs the correct class when its fed a string. To make a BERT classifier we add *BertForSequenceClassification* [43], which adds a single layer consisting of 139 nodes, one for each label in our multi-label text classification task on top of the original BERT model. We can then feed our input data into the model, training both the pre-trained BERT model and the untrained classification layer to our target task. The entire model was fine-tuned for seven epochs. To train our BERT model we utilized HuggingFace's transformer library [43].

7.3 Experimental results

After fine-tuning our models on our target corpus, we predicted the 139 unique ICD-9 codes on the validation set and scored them using the performance metrics: accuracy, F-1, precision, recall, and AUROC. These were each computed using micro and macro-averages. Macro-average computes the metric for each class independently and then take an average, in contrast, micro-average computes the average of the contribution of every class as a whole. Note that we evaluated our models on the validation data set. As the validation performance was monitored during training, and used for early stopping, there might be some performance bias in our results.

By looking at our results from the validation set in table 7.1 we can see that our ULMFiT model outperformed the BERT model. We believe that

this is the result of the effective language model fine-tuning performed on the ULMFiT model by using *slanted triangular learning rates* and *discriminative fine-tuning*. Our fine-tuned language model for the ULMFiT model reached an accuracy of 68.6%, and managed to learn semantics within the medical notes. An example of how good the language model can be seen in figure 7.4, where we get the language model to predict the 15 next tokens of the sentence "*An 80-year old female who presented with chest pain*", a text sample collected from the dataset.

```
an 80-year-old female who presented with chest pain and results to have
  a pericardial effusion on ECHO . The effusion was oncologic .

an 80-year-old female who presented with chest pain and shortness of breath
  stomach with a small right - sided pneumothorax . She was treated
  with
```

Figure 7.4: ULMFiT language model predicting the 15 next tokens of the sentence *"An 80-year old female who presented with chest pain"*.

Task	Model	acc	Micro/Macro	F1	Precision	Recall	AUROC
ICD-9 codes	ULMFiT	0.2%	Micro	50.8%	78.9%	37.5%	68.4%
			Macro	35.4%	58.5%	29.3%	64.2%
ICD-9 codes	BERT	0.15%	Micro	45.9%	75.7%	32,9%	66.1%
			Macro	24.9%	44.6%	20.9%	60.0%

Table 7.1: Table of ULMFiT and BERT performance metrics results on the validation set.

We can compare our results to a case study that performs multi-label text classification on ICD-9 codes from medical notes using the MIMIC-III data. As mentioned in Experiment 1, comparing results with other case studies is difficult when the preprocessing steps are different. In cases of autonomous ICD-9 coding on clinical notes from MIMIC-III, the setup often differ in how many codes each study tries to predict [91, 92, 58, 93]. A case study that we can compare our results too is *Condensed Memory Networks for Clinical Diagnostic Inferencing* [91].

In this case study, they filtered the ICD-9 codes down to the top 50 and top 100 most common ICD-9 codes represented in the MIMIC-III database. In our experiment, we went with a broader approach and chose to perform a multi-label classification on the top-139 ICD-9 codes. When performing multitask classification the number of labels scale-up prediction difficulty by increasing the number of possible combinations exponentially. ICD-9 codes are also similar to each other because of their hierarchical nature. An increase in the number of codes increases the similarity between some of the codes, thus increasing the difficulty for machine learning models to distinguish them. With an increase in complexity, the models need significantly

more data to be able to correctly predict the correct labels.

The case study used a *Condensed Memory Neural Networks* (C-MemNN) [91] to perform their multi-label classification. They scored their model using AUC macro score and achieved a score of 83.33 for top-50 and 76.6 for top-100. In comparison to our ULMFiT model that achieved an AUC macro score of 0.64, we can conclude that our approach performed worse for this multi-label text classification task. However, the numbers of labels we tried to predict were higher so a direct comparison can't be made. Compared to state-of-the-art performance for this task in the field, achieved by Google researchers and collaborators in [58], where they achieved an AUROC score of 0.90 on the prediction of all ICD-9 codes [58], we can do a final conclusion that our approach was not sufficient to tackle this task.

7.4 Discussion

In this study we tried to predict a broad range of ICD-9 codes from discharge notes from the MIMIC-III dataset. Autonomous coding has the potential to save administrative costs, provide insight, and aid clinicians in decision making [94, 95]. However, as we detected in our case study it's a challenging task, requiring large amounts of data and advanced methods for success. Upon completing our case study, we learned that fine-tuning pre-trained language models to EHR notes corpora still require large amounts of training data to perform autonomous coding in medical notes. We expect that a larger training corpus would lead to more accurate machine learning models. In addition, we believe that a different preprocessing approach for the selection of ICD-9 codes to predict could increase prediction performance [93, 92, 96].

Chapter 8

Conclusion and further work

In this thesis, we gave an overview over what is possible to accomplish and what problems we are faced with when using machine learning for EHR analysis. We reviewed some remarkable recent papers in the field and performed two experiments on one of the largest available public EHR data sets from a hospital, MIMIC-III [12]. In our experiments, we investigated prediction and information gathering on two important pieces of EHR data, the raw clinical data stored in tabular data format, and the clinical notes written by nurses and clinicians.

We uncovered that EHR analytics stand to deliver a new era for EHR by delivering clinicians a way to measure their care delivery performance, reduce healthcare costs, aid in decision making, and improve patient care [97]. It also has the possibility to help analyze the vast amount of generated data that is continuously expanding. However, although the field shows promising results of improving health care there are several obstacles to overcome in the field. Perhaps the largest obstacle in the field is the data itself. We discussed that the high dimensional EHR data have problems with temporality, irregularity, and multiple modalities that negatively affect the machine learning models severely. Requiring and intensive cleaning process, i.e. through preprocessing. A process that requires a significant amount of time and domain knowledge to perform well.

Machine learning to perform EHR analysis has a promising future, but the field is faced with several challenges and needs a significant amount of research and testing to evaluate prediction models in real life settings. This will require close collaboration between clinical experts and machine learning practitioners.

In our experiments, there is a lot of interesting work that could be explored in the future:

1. Could we find another source of real hospital EHR data to perform experiments on?

2. How could we improve the preprocessing in the MIMIC-Extract pipeline so it would increase our prediction results on the four medical tasks? Would it be beneficial to find another representation for vitals and lab time series data other than hourly buckets?

3. Would the implementation of a GRU-D model to handle missing values be a better approach than the current one in the MIMIC-Extract preprocessing pipeline?

4. Could other traditional machine learning models get better prediction results in experiment one?

5. Can an LSTM deep learning model better learn the representation of time-series data and improve the prediction results in experiment one?

6. How would further preprocessing of ICD-9 codes down to top-10 and top-50 most commonly represented codes similarly to [93, 92] work on our models in experiment two?

7. In our ULMFiT setup we evaluated performance on a hold-out validation set. One cannot be certain that this set is representative of the data, and a k-fold cross-validation design would be better. Setting aside a representative test set not seen during model construction would also be better. Ideally, a nested k-fold cross-validation where all instances get to be in the test set, while still being unseen during the corresponding model construction steps. Both of these designs would however be extremely computationally demanding.

Bibliography

[1] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.

[2] E. RS, "Electronic health records: Then, now, and in the future," *Yearbook of medical informatics*, vol. 1, no. Supplement 1, pp. S48–S61, 2016.

[3] R. H. Miller and I. Sim, "Physicians' use of electronic medical records: barriers and solutions," *Health affairs*, vol. 23, no. 2, pp. 116–126, 2004.

[4] C. Safran, M. Bloomrosen, W. E. Hammond, S. Labkoff, S. Markel-Fox, P. C. Tang, and D. E. Detmer, "Toward a national framework for the secondary use of health data: an american medical informatics association white paper," *Journal of the American Medical Informatics Association*, vol. 14, no. 1, pp. 1–9, 2007.

[5] P. Yadav, M. Steinbach, V. Kumar, and G. Simon, "Mining electronic health records (ehrs) a survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–40, 2018.

[6] R. Fang, S. Pouyanfar, Y. Yang, S.-C. Chen, and S. Iyengar, "Computational health informatics in the big data age: a survey," *ACM Computing Surveys (CSUR)*, vol. 49, no. 1, pp. 1–36, 2016.

[7] V. Tresp, J. M. Overhage, M. Bundschus, S. Rabizadeh, P. A. Fasching, and S. Yu, "Going digital: a survey on digitalization and large-scale data analytics in healthcare," *Proceedings of the IEEE*, vol. 104, no. 11, pp. 2180–2206, 2016.

[8] G. Hinton, "Deep learning—a technology with the potential to transform health care," *Jama*, vol. 320, no. 11, pp. 1101–1102, 2018.

[9] A. Rajkomar, J. Dean, and I. Kohane, "Machine learning in medicine," *New England Journal of Medicine*, vol. 380, no. 14, pp. 1347–1358, 2019. PMID: 30943338.

[10] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," 2009.

[11] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[12] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, p. 160035, 2016.

[13] N. I. of Health *et al.*, "Electronic health records overview," *National Center for Research Resources. National Institutes of Health, Bethesda*, 2006.

[14] J. G. Anderson, "Social, ethical and legal barriers to e-health," *International journal of medical informatics*, vol. 76, no. 5-6, pp. 480–483, 2007.

[15] N. G. Valikodath, P. A. Newman-Casey, P. P. Lee, D. C. Musch, L. M. Niziol, and M. A. Woodward, "Agreement of ocular symptom reporting between patient-reported outcomes and medical records," *JAMA ophthalmology*, vol. 135, no. 3, pp. 225–231, 2017.

[16] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, vol. 8, no. 1, pp. 1–12, 2018.

[17] M. Yakout, A. K. Elmagarmid, J. Neville, and M. Ouzzani, "Gdr: a system for guided data repair," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pp. 1223–1226, 2010.

[18] J. C. Mandel, D. A. Kreda, K. D. Mandl, I. S. Kohane, and R. B. Ramoni, "Smart on fhir: a standards-based, interoperable apps platform for electronic health records," *Journal of the American Medical Informatics Association*, vol. 23, no. 5, pp. 899–908, 2016.

[19] S. Folkman, "Privacy and confidentiality.," 2000.

[20] J. L. Fernández-Alemán, I. C. Señor, P. Á. O. Lozoya, and A. Toval, "Security and privacy in electronic health records: A systematic literature review," *Journal of biomedical informatics*, vol. 46, no. 3, pp. 541–562, 2013.

[21] P. Chhanabhai and A. Holt, "Consumers are ready to accept the transition to online and electronic records if they can be assured of the security measures," *Medscape general medicine*, vol. 9, no. 1, p. 8, 2007.

[22] L. Zurita and C. Nøhr, "Patient opinion-ehr assessment from the users perspective.," *Medinfo*, vol. 107, pp. 1333–1336, 2004.

[23] Z. Ghahramani, "Unsupervised learning," in *Summer School on Machine Learning*, pp. 72–112, Springer, 2003.

[24] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[25] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[26] J. D. Rodriguez, A. Perez, and J. A. Lozano, "Sensitivity analysis of k-fold cross validation in prediction error estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 3, pp. 569–575, 2009.

[27] A. Zheng, "Evaluating machine learning models: a beginner's guide to key concepts and pitfalls," 2015.

[28] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*, pp. 1–15, Springer, 2000.

[29] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[30] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[31] R. S. Olson, W. La Cava, Z. Mustahsan, A. Varik, and J. H. Moore, "Data-driven advice for applying machine learning to bioinformatics problems," *arXiv preprint arXiv:1708.05070*, 2017.

[32] L. Deng and Y. Liu, *Deep learning in natural language processing*. Springer, 2018.

[33] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *ieee Computational intelligenCe magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[34] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[35] Y. Bengio, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[36] 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.

[37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[38] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

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

[40] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[41] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[42] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[43] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," *ArXiv*, vol. abs/1910.03771, 2019.

[44] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing lstm language models," *arXiv preprint arXiv:1708.02182*, 2017.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[46] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472, IEEE, 2017.

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[48] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

[49] A. B. Olokoba, O. A. Obateru, and L. B. Olokoba, "Type 2 diabetes mellitus: a review of current trends," *Oman medical journal*, vol. 27, no. 4, p. 269, 2012.

[50] P. H. Reddy, "Can diabetes be controlled by lifestyle activities?," *Current research in diabetes & obesity journal*, vol. 1, no. 4, 2017.

[51] T. Zheng, W. Xie, L. Xu, X. He, Y. Zhang, M. You, G. Yang, and Y. Chen, "A machine learning-based framework to identify type 2 diabetes through electronic health records," *International journal of medical informatics*, vol. 97, pp. 120–127, 2017.

[52] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[53] P. Langley and S. Sage, "Induction of selective bayesian classifiers," in *Uncertainty Proceedings 1994*, pp. 399–406, Elsevier, 1994.

[54] B. Schölkopf, A. J. Smola, F. Bach, *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[55] O. Hellevik, "Linear versus logistic regression when the dependent variable is a dichotomy," *Quality & Quantity*, vol. 43, no. 1, pp. 59–74, 2009.

[56] A. N. Kho, M. G. Hayes, L. Rasmussen-Torvik, J. A. Pacheco, W. K. Thompson, L. L. Armstrong, J. C. Denny, P. L. Peissig, A. W. Miller, W.-Q. Wei, *et al.*, "Use of diverse electronic medical record systems to identify genetic risk for type 2 diabetes within a genome-wide association study," *Journal of the American Medical Informatics Association*, vol. 19, no. 2, pp. 212–218, 2012.

[57] I. Silva, G. Moody, D. J. Scott, L. A. Celi, and R. G. Mark, "Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012," in *2012 Computing in Cardiology*, pp. 245–248, IEEE, 2012.

[58] 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.

[59] D. Bender and K. Sartipi, "Hl7 fhir: An agile and restful approach to healthcare information exchange," in *Proceedings of the 26th IEEE international symposium on computer-based medical systems*, pp. 326–331, IEEE, 2013.

[60] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1–39, 2010.

[61] Y. P. Tabak, X. Sun, C. M. Nunez, and R. S. Johannes, "Using electronic health record data to develop inpatient mortality predictive model: Acute laboratory risk of mortality score (alarms)," *Journal of the American Medical Informatics Association*, vol. 21, no. 3, pp. 455–463, 2014.

[62] O. K. Nguyen, A. N. Makam, C. Clark, S. Zhang, B. Xie, F. Velasco, R. Amarasingham, and E. A. Halm, "Predicting all-cause readmissions using electronic health record data from the entire hospitalization: model development and comparison," *Journal of hospital medicine*, vol. 11, no. 7, pp. 473–480, 2016.

[63] V. Liu, P. Kipnis, M. K. Gould, and G. J. Escobar, "Length of stay predictions: improvements through the use of automated laboratory and comorbidity variables," *Medical care*, pp. 739–744, 2010.

[64] A. Barker, K. Mengersen, and A. Morton, "What is the value of hospital mortality indicators, and are there ways to do better?," *Australian Health Review*, vol. 36, no. 4, pp. 374–377, 2012.

[65] H. Bueno, J. S. Ross, Y. Wang, J. Chen, M. T. Vidán, S.-L. T. Normand, J. P. Curtis, E. E. Drye, J. H. Lichtman, P. S. Keenan, *et al.*, "Trends in length of stay and short-term outcomes among medicare patients hospitalized for heart failure, 1993-2006," *Jama*, vol. 303, no. 21, pp. 2141–2147, 2010.

[66] T. Rotter, L. Kinsman, E. L. James, A. Machotta, H. Gothe, J. Willis, P. Snow, and J. Kugler, "Clinical pathways: effects on professional practice, patient outcomes, length of stay and hospital costs," *Cochrane database of systematic reviews*, no. 3, 2010.

[67] C. for Disease Control, Prevention, *et al.*, "Hipaa privacy rule and public health. guidance from cdc and the us department of health and human services," *MMWR: Morbidity and mortality weekly report*, vol. 52, no. Suppl. 1, pp. 1–17, 2003.

[68] S. Wang, M. McDermott, G. Chauhan, M. C. Hughes, T. Naumann, and M. Ghassemi, "Mimic-extract: A data extraction, preprocessing, and representation pipeline for mimic-iii," *arXiv preprint arXiv:1907.08322*, 2019.

[69] H. Harutyunyan, H. Khachatrian, D. C. Kale, G. V. Steeg, and A. Galstyan, "Multitask learning and benchmarking with clinical time series data," *arXiv preprint arXiv:1703.07771*, 2017.

[70] 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.

[71] B. Nestor, M. McDermott, W. Boag, G. Berner, T. Naumann, M. C. Hughes, A. Goldenberg, and M. Ghassemi, "Feature robustness in non-stationary health records: caveats to deployable model performance in common clinical machine learning tasks," *arXiv preprint arXiv:1908.00690*, 2019.

[72] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for

machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.

[73] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[74] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[75] J. Calvert, Q. Mao, J. L. Hoffman, M. Jay, T. Desautels, H. Mohamadlou, U. Chettipally, and R. Das, "Using electronic health record collected clinical variables to predict medical intensive care unit mortality," *Annals of Medicine and Surgery*, vol. 11, pp. 52–57, 2016.

[76] S. Purushotham, C. Meng, Z. Che, and Y. Liu, "Benchmark of deep learning models on large healthcare mimic datasets," *arXiv preprint arXiv:1710.08531*, 2017.

[77] T. Gentimis, A. Ala'J, A. Durante, K. Cook, and R. Steele, "Predicting hospital length of stay using neural networks on mimic iii data," in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pp. 1194–1201, IEEE, 2017.

[78] Y. Jo, L. Lee, and S. Palaskar, "Combining lstm and latent topic modeling for mortality prediction," *arXiv preprint arXiv:1709.02842*, 2017.

[79] S. Wang, M. B. McDermott, G. Chauhan, M. Ghassemi, M. C. Hughes, and T. Naumann, "Mimic-extract: A data extraction, preprocessing, and representation pipeline for mimic-iii," in *Proceedings of the ACM Conference on Health, Inference, and Learning*, pp. 222–235, 2020.

[80] R. Sadeghi, T. Banerjee, and W. Romine, "Early hospital mortality prediction using vital signals," *Smart Health*, vol. 9, pp. 265–274, 2018.

[81] A. Jiwani, D. Himmelstein, S. Woolhandler, and J. G. Kahn, "Billing and insurance-related administrative costs in united states' health care: synthesis of micro-costing evidence," *BMC Health Services Research*, vol. 14, no. 1, p. 556, 2014.

[82] S. Purkayastha, J. W. Gichoya, and A. S. Addepally, "Implementation of a single sign-on system between practice, research and learning systems," *Applied clinical informatics*, vol. 26, no. 01, pp. 306–312, 2017.

[83] C. Friedman, P. O. Alderson, J. H. Austin, J. J. Cimino, and S. B. Johnson, "A general natural-language text processor for clinical radiology," *Journal of the American Medical Informatics Association*, vol. 1, no. 2, pp. 161–174, 1994.

[84] A. R. Aronson and F.-M. Lang, "An overview of metamap: historical perspective and recent advances," *Journal of the American Medical Informatics Association*, vol. 17, no. 3, pp. 229–236, 2010.

[85] G. K. Savova, J. J. Masanz, P. V. Ogren, J. Zheng, S. Sohn, K. C. Kipper-Schuler, and C. G. Chute, "Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications," *Journal of the American Medical Informatics Association*, vol. 17, no. 5, pp. 507–513, 2010.

[86] H. Liu, S. J. Bielinski, S. Sohn, S. Murphy, K. B. Wagholikar, S. R. Jonnalagadda, K. Ravikumar, S. T. Wu, I. J. Kullo, and C. G. Chute, "An information extraction framework for cohort identification using electronic health records," *AMIA Summits on Translational Science Proceedings*, vol. 2013, p. 149, 2013.

[87] J. Callen, J. McIntosh, and J. Li, "Accuracy of medication documentation in hospital discharge summaries: a retrospective analysis of medication transcription errors in manual and electronic discharge summaries," *International journal of medical informatics*, vol. 79, no. 1, pp. 58–64, 2010.

[88] J. Howard and S. Gugger, "Fastai: A layered api for deep learning," *Information*, vol. 11, no. 2, p. 108, 2020.

[89] Y. Peng, S. Yan, and Z. Lu, "Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets," in *Proceedings of the 2019 Workshop on Biomedical Natural Language Processing (BioNLP 2019)*, pp. 58–65, 2019.

[90] D. C. Comeau, C.-H. Wei, R. Islamaj Doğan, and Z. Lu, "Pmc text mining subset in bioc: about three million full-text articles and growing," *Bioinformatics*, vol. 35, no. 18, pp. 3533–3535, 2019.

[91] A. Prakash, S. Zhao, S. A. Hasan, V. Datla, K. Lee, A. Qadir, J. Liu, and O. Farri, "Condensed memory networks for clinical diagnostic inferencing," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[92] J. Huang, C. Osorio, and L. W. Sy, "An empirical evaluation of deep learning for icd-9 code assignment using mimic-iii clinical notes," *Computer methods and programs in biomedicine*, vol. 177, pp. 141–153, 2019.

[93] S. Nuthakki, S. Neela, J. W. Gichoya, and S. Purkayastha, "Natural language processing of mimic-iii clinical notes for identifying diagnosis and procedures with neural networks," *arXiv preprint arXiv:1912.12397*, 2019.

[94] A. Perotte, R. Pivovarov, K. Natarajan, N. Weiskopf, F. Wood, and N. Elhadad, "Diagnosis code assignment: models and evaluation metrics," *Journal of the American Medical Informatics Association*, vol. 21, no. 2, pp. 231–237, 2014.

[95] D. W. Bates, G. J. Kuperman, S. Wang, T. Gandhi, A. Kittler, L. Volk, C. Spurr, R. Khorasani, M. Tanasijevic, and B. Middleton, "Ten commandments for effective clinical decision support: making the practice of evidence-based medicine a reality," *Journal of the American Medical Informatics Association*, vol. 10, no. 6, pp. 523–530, 2003.

[96] T. Baumel, J. Nassour-Kassis, R. Cohen, M. Elhadad, and N. Elhadad, "Multi-label classification of patient notes: case study on icd code assignment," in *Workshops at the thirty-second AAAI conference on artificial intelligence*, 2018.

[97] 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.