PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

EXPLORING REPRESENTATIONS OF ICD CODES FOR PATIENT READMISSION PREDICTION

TAMARA COVACEVICH STIPICICH

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Advisor:
DENIS PARRA

Santiago de Chile, April 2021

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

EXPLORING REPRESENTATIONS OF ICD CODES FOR PATIENT READMISSION PREDICTION

TAMARA COVACEVICH STIPICICH

Members of the Committee:

DENIS PARRA	DocuSigned by: 0F3E21932A2E493...
HANS LÖBEL	DocuSigned by: Hans Lobel 442F10C9559E4B5...
DANIEL CAPURRO	DocuSigned by: 14E3BA3101A14CA...
RICARDO RAINERI	DocuSigned by: Ricardo Raineri B024207090CA453...

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Santiago de Chile, April 2021

To my parents, for their light

ACKNOWLEDGEMENTS

This work was partially funded by ANID - Millennium Science Initiative Program - Code ICN17_002, as well as by ANID - Fondecyt Regular Grant 1191791.

I want to thank everyone who supported me in developing this thesis: My supervisor, Denis Parra, for always providing me insights, support, and trust. The Gaido family, for opening their home to me (most of this thesis was written overlooking their beautiful garden). Flo, Haase, Charles, and my Frauen Power team from TU Delft for the laughs and the mutual encouragement while working on our degree. Friends from Punta Arenas, for the good times, drinking virtual mate while co-working during the pandemic. My family.

This thesis is not only the final product of my Master's Degree but also the end of a 7-year-long journey in Engineering School. This journey was full of wonderful memories and people, and to them, I owe this accomplishment and who I've become. Huge shout out to:

Tribu, the best friends I could've asked for, for their unconditional support.

L'Hopital FC, my teammates. My self-discovery journey wouldn't have been the same without you in my life.

Friends and acquaintances from Generacion 2014. Every day I am amazed by your accomplishments, and I can't wait to see the paths you end up taking—special appreciation to everyone that contributed to our two wins of the Alianzas, great great times.

Everyone I met on our patio. Our conversations shaped my views on social inequality, feminism, diversity, and other relevant issues for our society.

It is always easier to follow a road that has already been traveled. I want to thank my friend Belen Saldias for always giving me great advice, even though I try very hard to rebel against it.

Finally, special thanks to the Department of Computer Science (DCC) community at PUC; teachers, staff, students, and friends. The 4th floor was always a place of happiness, comfort, and passion for me. Thanks for providing an atmosphere of endless opportunities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Hospital readmissions occur frequently, they are expensive, and a high number of them decrease the institution's perceived quality. In machine learning, the readmission task aims to predict a patient's risk of readmission. Several solutions have been proposed using Electronic Health Records (EHR). EHRs have all information related to an admission: lab tests, free notes, demographic data, and International Classification of Diseases (ICD) codes. ICD is an international standard that provides codes for diagnoses and procedures. Initial solutions to the readmission problem used ICD codes as features via categorical representations or representations learned from their local context. Recent solutions ingest all EHR data, adding unnecessary complexity. In this research, we explore new representations for ICD codes. We leverage their text descriptions using Natural Language Processing techniques and their ontological representation through graph embedding algorithms. We provide benchmarks for the readmission task using a novel dataset from a large Chilean hospital, with a clear evaluation framework, and achieve results comparable with the state of the art. Generated ICD mappings and representations are publicly available.

Keywords: ICD, EHR, readmission, representation learning, embeddings, NLP.

RESUMEN

Los reingresos hospitalarios ocurren con frecuencia, son costosos y son usados como medida de calidad de las instituciones. En el aprendizaje de máquina, la tarea de readmisión tiene como objetivo predecir el riesgo de readmisión de un paciente. Se han propuesto diversas soluciones basadas en datos obtenidos desde sistemas de Historia Clínica Electrónica (HCE). Los datos de HCE tienen toda la información relacionada con una admisión: pruebas de laboratorio, notas de texto libre, datos demográficos y códigos de la Clasificación Internacional de Enfermedades (CIE). CIE es un estándar internacional que define códigos para diagnósticos y procedimientos. Las soluciones iniciales al problema de readmisión utilizaron códigos CIE a través de representaciones categóricas o representaciones aprendidas de su contexto local. Las soluciones recientes ingieren todos los datos de HCE, lo que agrega una complejidad innecesaria. En esta investigación, exploramos nuevas representaciones de códigos CIE. Aprovechamos sus textos descriptivos utilizando técnicas de Procesamiento del Lenguaje Natural y también su representación ontológica a través de algoritmos de grafo. Reportamos resultados de referencia para la tarea de readmisión utilizando un nuevo conjunto de datos de admisión de un hospital chileno, con un marco de evaluación claro, y logramos resultados comparables con el estado del arte. Las representaciones y mapeos CIE generados están disponibles públicamente.

Palabras Claves: CIE, HCE, readmisión, aprendizaje de representación, PLN.

1. INTRODUCTION

During a hospitalization, clinicians must continuously assess the patients' conditions to find the best moment when the patient should be discharged versus continuing to receive care in the hospital. Although professionals continuously assess the right time to discharge a patient, the process is far from perfect. A study conducted in 2007 in the Medicare system (USA) reported that 17% of hospital admissions resulted in readmissions within 30 days of discharge, with 76% of these having the potential to be avoided. Readmissions have a monetary cost; they accounted for $15 billion in Medicare spending (MPAC, 2007), and their number is also used as a measure of quality. These are some of the reasons that motivate hospitals to find methods that can predict readmission risk (Fierro, Pérez, & Mora, 2020).

Since the implementation of EHRs, several studies have leveraged this new electronic health data to improve readmission risk prediction. In the beginning, they used mainly demographic data of patients and the admission's ICD codes. These codes represent medical diagnoses and procedures; they are standardized and internationally recognized (Futoma, Morris, & Lucas, 2015). With the emergence of Deep Learning (Goodfellow, Bengio, & Courville, 2016), a field that provided better techniques to consume massive amounts of data, new proposed solutions started consuming all EHR data, including clinical notes (Rajkomar et al., 2018). Increasing the quantity of data consumed is challenging since EHRs are not designed with research as a priority. Even within the same institutions, EHRs might be used differently. Standards for EHRs constantly evolve, and comparing EHRs gathered from different places, and designed with different assumptions, is very challenging (Ching et al., 2018). We hypothesize that it is possible to achieve comparable results only using ICD codes, which are simpler and standard across Chile and other countries. Our central insight is that Deep Learning techniques were used to solve the problem by consuming more data instead of using them to improve the ICD codes' representations inputted to the model. Furthermore, we show that the readmission problem is not clearly solved in the existing literature since reported metrics do not show the methods' actual performance or facilitate comparison. Thus, in this research, we explore old and new ICD codes representations and provide new benchmarks for the readmission task using ICD codes as the main source of information.

Most research on clinical data applies to English text; therefore, it uses data from countries with English speaking institutions (i.e., Australia, USA, UK). Limited research has been done in Spanish, despite the large volume of clinical content generated in this language worldwide (A Miranda-Escalada, 2020). We use a novel dataset with all the admissions of a large Chilean hospital over five years. This is the first work in the country that uses ICD codes to solve the readmission problem in a national institution to the best of our knowledge. We hope it will constitute a basis for future national research in the area. To facilitate this, we identify and address critical challenges in the dataset and make publicly available the generated ICD mappings and representations.

We claim the following contributions:

- Identification and addressing of challenges when working with a DRG dataset
- Creation of a public repository with the English-Spanish translation of the ICD codes used in Chile[1]
- Creation of a public repository with the learned ICD codes representations and the ontologies[2]
- Implementation and generation of two new ICD codes representations (from their text descriptions leveraging pretrained models and from their graph structure leveraging new developments in graph embedding)
- Benchmarking of several ICD codes representations on the readmission task with a clear and relevant evaluation framework

[1] https://github.com/tamycova/icd-cie-codes-chile

[2] https://github.com/tamycova/ICD-embeddings

2. BACKGROUND THEORY

In this chapter, we introduce the essential Machine Learning and Medical concepts needed to understand this work.

2.1. Machine Learning background

2.1.1. Classification

Classification is the task of predicting the class to which an example belongs. The example belongs to only one of several known classes. Each example is represented by a set of values known as features. It is important to find features that encode as much class-discriminatory information (Theodoridis, 2020). Having a feature vector x that corresponds to an example, the goal is to design a function $f(x)$, known as a classifier, capable of predicting the class to which the example belongs.

A binary classification problem consists of a set up with only two classes: an example could belong to one class or the other, as shown in Figure 2.1. A binary classification set up is Email Spam Detection, where a given email can be Spam or not.

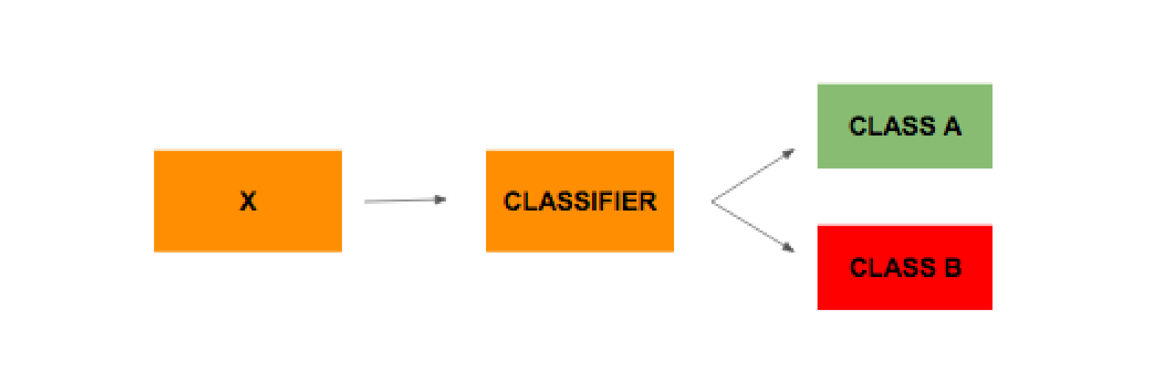


Figure 2.1. Traditional set up for a binary classification problem. To classify an object, it must first be represented as a feature vector x, which is then inputted to the classifier to predict the initial object's class.

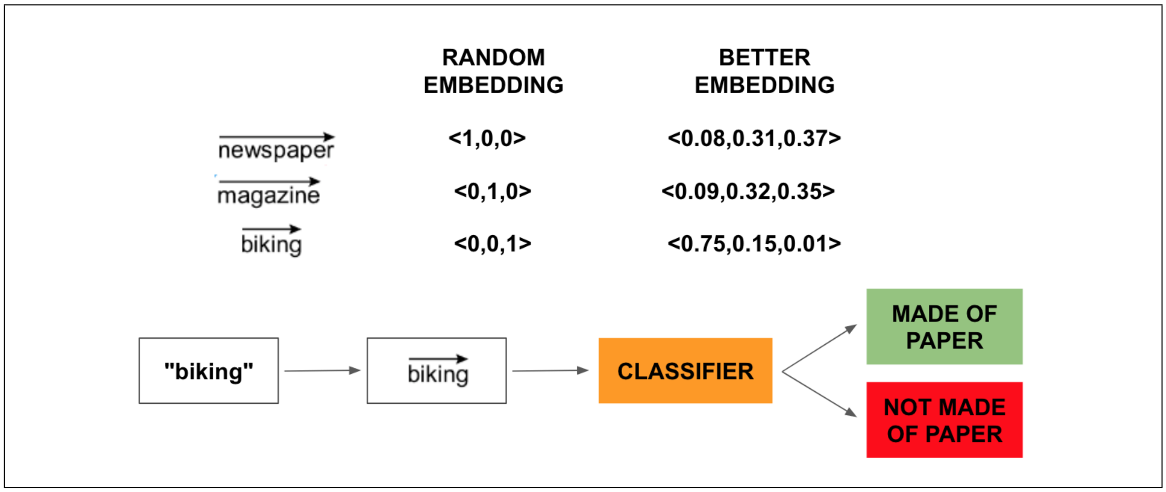
2.1.2. Embeddings

Given an example, there can be multiple ways of building a feature vector to represent it. The vector could have categorical features and numerical features, and they could be discrete or continuous. As mentioned before, the feature vector must encode as much class-discriminatory information as possible, so choosing the right representation is crucial.

One technique used to represent an example is the use of embeddings, which are mappings from discrete objects to vectors of real numbers. Usually, the patterns of location and distance between vectors are used to model the relationships between objects. Since words are discrete objects, this technique has become very popular in Natural Language Processing (NLP).

For example, suppose we have the Spanish vocabulary and want a feature vector to represent a word. In that case, we could assign a random vector to each word, and we know that a given vector is the representation of a given word. A problem with this is that in the vector space, the words' vectors do not represent anything relevant about their meaning or relations with other words. Alternatively, it would be useful to have a better mapping, one where the vectors of words that are synonyms are closer in a lower-dimensional space, for example. The development of these word embeddings is an active area of research, and they have been widely adopted with satisfactory results in several domains (Gutiérrez & Keith Norambuena, 2019). Figure 2.2 shows an example of a classifier that receives the embedding of a word as a feature vector and determines if the object is made or not of paper. With random embeddings for the words is hard to decide on the class (the embedding for ”biking” has no relevant differences with ”newspaper” and ”magazine”). With a better embedding, ”magazine” and ”newspaper” are closer together in the vectorial space; this makes it easier for the classifier to learn the differences between what is made of paper and what is not.

It is important to note that although word embeddings are highly popular, any set of objects can be embedded into a representative vector space, not only words. In the healthcare space, and with the recent developments in Deep Learning methods, several research lines have been exploring the use of embeddings to improve performance in different tasks (Ching et al., 2018). For example, (Miotto, Li, & Kidd, 2016) used a novel architecture to represent patients, and it was able to improve patient-level predictions by up to 15%.



	RANDOM EMBEDDING	BETTER EMBEDDING
$\overrightarrow{\text{newspaper}}$	<1,0,0>	<0.08,0.31,0.37>
$\overrightarrow{\text{magazine}}$	<0,1,0>	<0.09,0.32,0.35>
$\overrightarrow{\text{biking}}$	<0,0,1>	<0.75,0.15,0.01>

Figure 2.2. The use of word embeddings to represent words in a binary classification problem.

2.2. Medical background

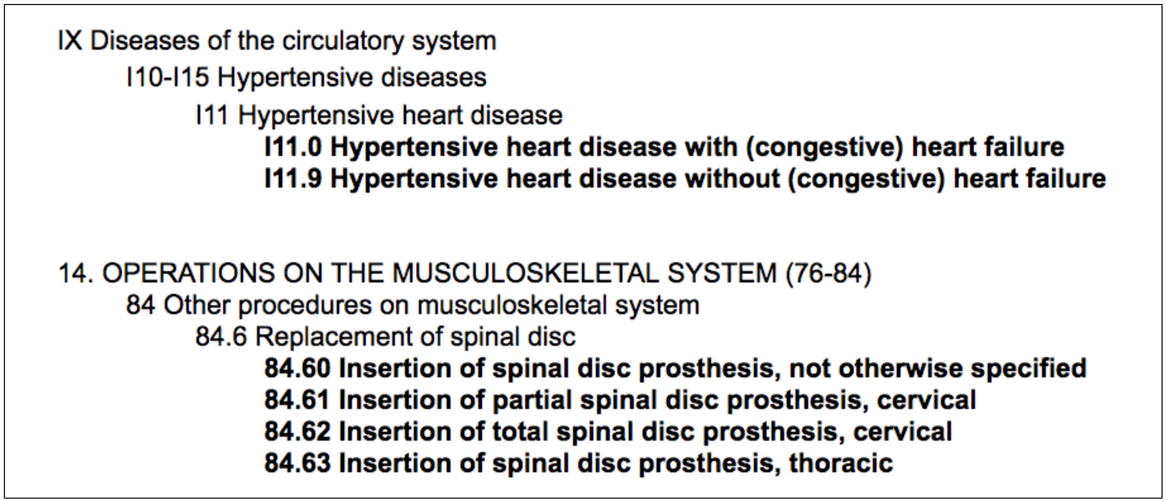
2.2.1. ICD Codes

The International Statistical Classification of Diseases and Related Health Problems (ICD) is the World Health Organization's (WHO) effort to globally identify health trends and statistics. It aims to define the universe of health conditions in an understandable and structured way.

Every code represents a single diagnosis or a single procedure. For example, the diagnosis code 'J15.2' represents 'Pneumonia due to staphylococcus', whereas 'J15.5' is the code used for pneumonia caused by a different bacteria according to the ICD-10 standard of 2008. In procedures, '11.1' stands for 'Incision of cornea'. The ICD standard is hierarchical; codes are divided into several chapters, divided into sub-chapters, etc. The deeper into the hierarchy, the higher the level of specificity. Further information about these codes and their structure is discussed in Sections 3, 5 and 6. Figure 2.3 shows examples of ICD codes for diagnosis and procedure.

In hospitals, clinical coders abstract information from a patient's medical record and assign codes to describe the patient in terms of diagnosis and procedures according to the ICD standard. The set of diagnosis codes and procedure codes of a patient is the input for the DRG grouper, which generates a DRG code for the patient (Karimi, Dai, Hassanzadeh, & Nguyen, 2017). ICD-codes serve a statistical purpose but are also used in billing (Zapata, 2018).

Standards change within different countries, and the standards get updated over time to reflect the advances in medical science.



- IX Diseases of the circulatory system
  - I10-I15 Hypertensive diseases
    - I11 Hypertensive heart disease
      - **I11.0 Hypertensive heart disease with (congestive) heart failure**
      - **I11.9 Hypertensive heart disease without (congestive) heart failure**

- 14. OPERATIONS ON THE MUSCULOSKELETAL SYSTEM (76-84)
  - 84 Other procedures on musculoskeletal system
    - 84.6 Replacement of spinal disc
      - **84.60 Insertion of spinal disc prosthesis, not otherwise specified**
      - **84.61 Insertion of partial spinal disc prosthesis, cervical**
      - **84.62 Insertion of total spinal disc prosthesis, cervical**
      - **84.63 Insertion of spinal disc prosthesis, thoracic**

Figure 2.3. This Figure shows, through indentation, the hierarchical structure of the codes. In bold are the ICD codes used in hospitals to assign diagnoses and procedures to a patient. The top of the Figure corresponds to diagnosis codes, and we can see that "I11.0" is part of the "Diseases of the circulatory system" chapter. The bottom of the Figure corresponds to procedure codes, all associated with the replacement of spinal disc.

2.2.2. DRG System

The Diagnostic-Related Group system was created in the United States in 1983, and it has been gradually introduced in many countries. This system's objective is to increase efficiency in the use of resources and provide transparency of the hospital activities (Mihailovic, Kocic, & Jakovljevic, 2016). This system's idea is that each patient can be classified into a DRG code group, and patients from the same group are expected to use similar amounts of hospital resources. DRG is also the basis for the payment system in several healthcare institutions.

In Chile, DRG was introduced to manage patient variability in 2009 with a starting project that analyzed 16 institutions. The biggest challenge was standardizing the information since, at the moment, all clinical data was on paper. The existing system in Chile's health system corresponds to the IR-DRG (International Refined DRG) and is used both in public and private institutions. When a patient is discharged, the clinical coder assigns ICD-10 codes for diagnoses and ICD-9-CM codes for procedures. The grouper captures all this information and, via an algorithm, classifies the patient into a DRG group (Zapata, 2018) with a DRG weight, as shown in Figure 2.4. The DRG weight represents the average resources required for that particular DRG, relative to the average resources used for all DRGs (per case).

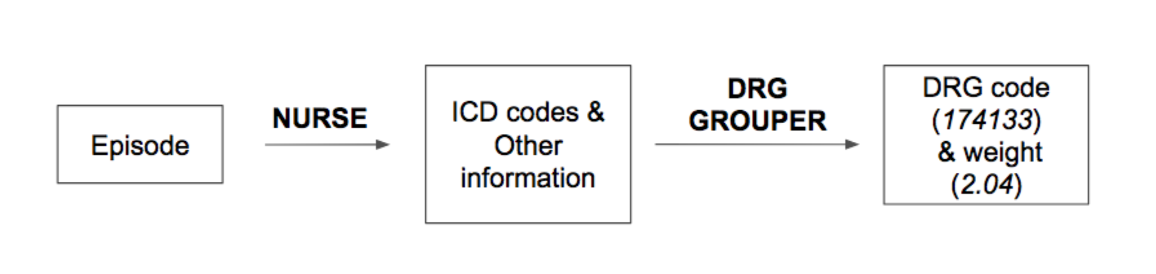

Episode
NURSE
ICD codes & Other information
DRG GROUPER
DRG code (174133) & weight (2.04)

Figure 2.4. The process for DRG code and weight assignment. First, during, and after the clinical episode, nurses codify the episode using ICD codes. This information is inputted to the DRG grouper algorithm, which determines the episode's DRG code and weight.

2.2.3. EHR

An Electronic Health Record (EHR) is the digital version of a patient's paper chart. An EHR entry typically contains the patient's history in the health institution, diagnoses and procedures, lab test results, and clinical notes. EHRs have been widely adopted. Since they provide rich digital information about hospital activity, there has been an increase in research activity that builds predictive models in the medical area (Pham, Tran, Phung, & Venkatesh, 2016).

2.2.4. Readmission problem

A hospital readmission is defined as an admission to a hospital a short time after an original admission (Futoma et al., 2015), they can be planned or unplanned. Accurately predicting the probability of unplanned readmission is clinically significant since it improves efficiency and reduces the burden for the doctors and patients. Also, some countries have set penalties for early readmissions. It is then expected that hospitals will be interested in methods that can flag that a patient has a high risk of being readmitted (Fierro et al., 2020).

In this work, the Readmission problem will be framed as a binary classification task. Define $y_i \in \{0, 1\}$ to indicate whether the ith episode resulted in readmission within 30 days (where a 1 denotes readmission) and x_i the vector representation of the episode. The goal is to train a classifier that assigns to x_i a class label $\hat{y}_i$ ($\hat{y}_i = f(x)$). We use probabilistic classifiers so instead of functions $f(x)$ they are conditional distributions Pr(Y|X), and for a given x_i they assign probabilities to both classes. In order to obtain $\hat{y}_i$ we define probability thresholds to make the decision, for example, with threshold 0.5, which predicts the class with the highest probability (since they sum to one):

$$\hat{y}_i = \begin{cases} 1, & \text{if } P(y_i = 1|x_i) \geq 0.5 \\ 0, & \text{otherwise} \end{cases}$$

It is important to mention that the Readmission problem could be approached in other ways, such as a regression task or defining readmission using 60 days instead of 30 days (Nguyen, Luo, Venkatesh, & Phung, 2018).

3. RELATED WORK

3.1. Readmission problem

Over the years, several research teams have tried to solve the readmission problem or used the readmission classification task to test their architectures. The most relevant to this thesis are listed in Table 3.1.

(Caruana et al., 2015) tackles the trade-off between accuracy and intelligibility, proposing high-performance generalized additive models with pairwise interactions. They report

Table 3.1. Comparison of models, representations and evaluation frameworks used in literature to solve the Readmission Problem.

Research	Representation	Model	Evaluation
Caruana, 2015	-	Generalized Additive Models	AUC (0.78) 30-day readmission
Futoma, 2015	Sparse binary matrix for ICD codes	Logistic Regression (local), Random Forest (local), SVM (local)	AUC (0.82) 30-day readmission
Nguyen, 2017	Learns embedding for ICD codes truncated at level 3	CNN	Accuracy (0.75) 60-day readmission
Pham, 2017	Learns embedding for ICD codes truncated at level 2	LSTM	F-score (79) for diabetes cohort 1-year readmission
Rajkomar, 2018	Sequence of time-ordered tokens	LSTM	AUC (0.75) 30-day readmission
Huang, 2019	Learns aggregated embedding for a set of notes	BERT	AUC (0.71) and RP80 (0.24 ± 0.1) 30-day readmission
Fierro, 2020	Sequence of time-ordered tokens	LSTM	AUC (0.76 without oversampling, 0.74 with oversampling), Precision (0.52) with Recall (0.13) 30-day readmission

that their method scales and provides accuracy comparable to the best unintelligible machine learning methods, reporting an AUC of 0.78 for their model. They do an in-depth analysis of the most important terms for the prediction of three patients. They argue that this makes the model already more intelligible than other traditional models like Random Forest.

(Futoma et al., 2015) describes and compares several predictive models on their performance in the readmission task. They also do a more fine-grained analysis, focusing on the five conditions that were being used to penalize hospitals, so they use local models as well as global models (trained in all the data). Their dataset is of considerable size, with 3.3 million rows of hospital admissions in New Zealand, a country that has a national healthcare system, so few patients are outside the system. They have the set of ICD codes, represented as a sparse matrix, and also demographic and background information for every admission. The best-reported result for the global method is 0.828 AUC using Stochastic Gradient Descent as the classifier.

(Pham et al., 2016) with DeepCare and (Nguyen, Tran, Wickramasinghe, & Venkatesh, 2017) with DeepR both use Deep Learning techniques to solve the problem.

DeepR focuses on detecting predictive clinical motifs from irregular episodic records using a convolutional neural network; they use a time window of 60 days, so it is not comparable to most studies that use a 30-day window. They input the model a sequence of hospital admissions represented with ICD codes, considered tokens, and use bag of words representation with logistic regression as the baseline. They report an accuracy slightly better than the baseline (increase in 0.2%), 0.75.

DeepCare introduces a novel architecture that aims to model the illness trajectory and healthcare processes of a patient encapsulated in a time-stamped sequence of admissions, using an LSTM model. The input to the LSTM is the information extracted from admission (diagnosis and interventions). The output is an illness state at the time of admission, which is then used for several subtasks; one of them is predicting 12-month readmission for a diabetes cohort and 3-month for a mental health cohort. They report an F-Score of 79.0 for the diabetes cohort and 74.7 for the mental cohort, a slight improvement compared to their results with SVM, Random Forest, and Plain RNN.

(Rajkomar et al., 2018) proposed a new sequential representation method for all EHR data (including clinical notes) that uses Deep Learning's ability to handle high volumes of messy data, and then proved that they could accurately predict multiple medical events. One of the events is 30-day readmission, with a reported AUC of 0.75.

Using a pure NLP approach, the advances in the area and only clinical notes (both discharge summaries and the first days of notes in ICU) (Huang, Altosaar, & Ranganath, 2019) outperforms baselines on 30-day readmission prediction. They pre-train the BERT model (Devlin, Chang, Lee, & Toutanova, 2019) on clinical notes and then finetune with the readmission prediction task. On pre-training, they achieve better results than Word2Vec when comparing correlation metrics on medical terms, and on prediction, they report an AUC of 0.714. They are the first ones to report recall, 0.242 (± 0.111) recall at a precision of 80%.

Finally, (Fierro et al., 2020) tries a similar approach than (Rajkomar et al., 2018) but using data from a hospital in Chile (Clinica Las Condes), in Spanish, reporting for the first time results on this task on an unstructured dataset that contains most of the information in this language. They report AUC 0.76, similar to (Rajkomar et al., 2018), and precision 0.52 with recall 0.13 for the positive class.

Overall, as showcased in Table 3.2, most related work lacks explainability in the models or used a specific local method to analyze the predictions.

Table 3.2. Comparison of explainability techniques used in literature to solve the Readmission Problem.

Research	Explainability
Caruana, 2015	Local analysis of three patients with term contribution
Futoma, 2015	-
Nguyen, 2017	t-SNE projection of embeddings
Pham, 2017	-
Rajkomar, 2018	-
Huang, 2019	Visualize BERT attentions (attention matrix)
Fierro, 2020	-

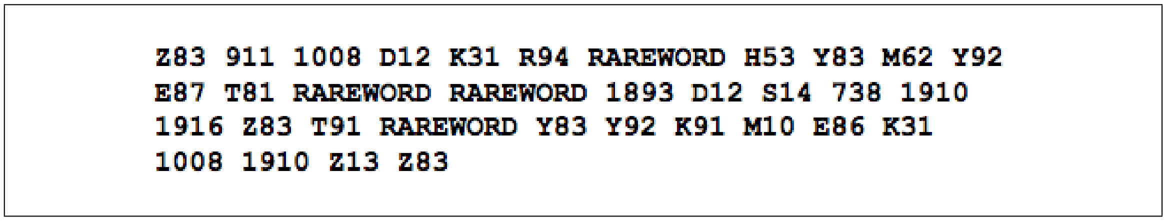
3.2. ICD codes representation

One challenge of dealing with ICD-10 codes directly is their discrete nature. ICD-10 has more than 68,000 codes (Nguyen et al., 2018).

In the previously mentioned works, authors used different representations for ICD codes. In the beginning they used one-hot encodings: a code c is represented by a one-hot vector $v_c \in \mathbb{R}^{|V|}$, where $v_c = (v_c^1, ..., v_c^{|V|})$ and $|V|$ is the number of codes, $v_c = (0, 0, ..., 0, 1, 0, ..., 0)$ because $v_c^i = 1$ if $c = i$, which implies c is the ith code, and $v_c^i = 0$ otherwise. For example, if we have a set of five codes then the embedding size is five and every code is represented having a 1 value in the corresponding position, just like the random embedding in Figure 2.2.

When more advanced encodings started to be used, this approach remained as the baseline (Caruana et al., 2015) (Futoma et al., 2015). This is the most basic approach, and it is not ideal since it fails to capture the ordering and semantic of the embedded objects. It also creates a high-dimensional vector, which leads to overfitting and expensive computations and memory usage.

In NLP, word embeddings that capture semantic representations have replaced one-hot models as the traditional text representation method. (Pham et al., 2016) and (Nguyen et al., 2017) noticed an interesting analogy between natural languages and EHR data (specifically, the ICD codes), where an episode is similar to a sentence, and diagnoses and procedures play the role of nouns and modifiers. With this in mind, they proposed Deep Learning models that are inputted a sentence of codes (so codes are considered the tokens) as Figure 3.1 shows, and they add an embedding layer with learnable weights. Hence, the model ends up learning embedding representations for the tokens (the ICD Codes). This approach was used with random initialization of the weights and also with a Word2Vec representation (Mikolov, Corrado, Chen, & Dean, 2013) trained in an unsupervised fashion. This approach is very interesting, and the authors argue that it can learn semantically sound representations. As the embedding is learned from data, the model does not rely on manual feature engineering. A clear drawback is that representation is learned from the dataset, which is relatively limited. It fails at including previously known information from the codes (for example, that they are clustered in chapters). Furthermore, (Nguyen et al., 2017) truncates to level 3 codes and (Pham et al., 2016) to level 2 codes, losing granular information that might be useful.



```
Z83 911 1008 D12 K31 R94 RAREWORD H53 Y83 M62 Y92
E87 T81 RAREWORD RAREWORD 1893 D12 S14 738 1910
1916 Z83 T91 RAREWORD Y83 Y92 K91 M10 E86 K31
1008 1910 Z13 Z83
```

Figure 3.1. A patient admission seen through the lens of NLP. ICD codes are tokens, and the episode is the sentence (Nguyen et al., 2017).

(Nguyen et al., 2018) formalizes the tasks of identifying patients with similar conditions (patient matching task) via their ICD code sequences. They leverage the literature at the time on representation learning of individual ICD codes and also use the fact that they are sequential in an admission sequence. They do an exhaustive analysis of the drawbacks of different representations and poolings, for example, mentioning that using the codes as discrete input is not enough ('I200 is different than R570 however they are both related to heart issues') and that summing vectors loses the sequentiality of the data. They also obtain embeddings with Word2Vec (size 100) (Mikolov et al., 2013), considering every patient a document and every code a word.

(Karimi et al., 2017) focuses on the automatic diagnosis coding task, which tries to assign ICD codes to radiology reports. In doing so, and even though they don't use learnable embeddings for the ICD codes, they generate an interesting discussion comparing in-domain embeddings with out-of-domain embeddings and static embeddings vs. dynamic embeddings. They conclude that pre-trained word vectors work better than randomly initialized ones, that dynamic word vectors are better than static ones, that in-domain word vectors are better than generic ones, and that larger embedding size does not always lead to better performance—word vectors trained in Medline outperformed word vectors trained using Wikipedia.

So far, we've seen simple representations that treat the codes as individual pieces of information and representations learned from the context they are used in. None of these exploit the fact that the ICD standard is a structured and hierarchical knowledge base and the codes itself represent that structure. (Hema & Justus, 2015) explores the tree representation for the ICD standard.

(Choi, Bahadori, Song, Stewart, & Sun, 2017) proposes a method that supplements EHR with hierarchical information from parent-child medical ontologies; it represents a medical concept as a combination of the ancestors in the ontology. They consider the frequency of the concept and its ancestors, so when a concept is less observed in the data, more weight is given to the ancestors. They concluded that they could get great results using less training data and obtain representations that align with the medical ontologies. One of the ontologies they test is the ICD-9 code hierarchy.

4. PROBLEM FORMULATION

When conducting the literature review, we identified a gap in the Readmission problem's proposed solutions. At first, models used one-hot encoding to represent the ICD codes. Then, (Pham et al., 2016), (Nguyen et al., 2017) and (Nguyen et al., 2018) assume that the meaning of a medical concept can be learned by its context since they treat codes as words and episodes as documents. In their models, ICD codes that co-occur closely (within an episode) are mapped to vectors closer together. Later on, ICD code representations were replaced by Deep Learning models that could ingest all the EHR data, but there were no more attempts to keep on improving the ICD code representations.

It is known that a significant challenge in using more of the data available for a patient is the lack of standards and semantic interoperability of health data from multiple sites (Rajkomar et al., 2018). A dataset that uses ICD codes as the primary source of information is more standard, especially in Chile. Because of the DRG system, the government dictates the ICD standard used for diagnoses and procedures. Thus, in this research, we focus on improving the ICD code representation for the readmission problem.

Moreover, all works that were reviewed on the readmission problem do not have a consistent evaluation system. They use different metrics, and most of them use AUROC as the single reported metric to demonstrate performance. This is not the best metric since, in the health system, precision is crucial. Most of them don't report recall or even mention that recall is relatively low. Only (Huang et al., 2019) reports recall with a specific precision threshold. This study will report benchmarks for different ICD code representations; they will all be compared with the same metrics and contrasted with the literature.

This study is the first of its type in Chile since (Fierro et al., 2020) did not use intervention codes. Most of the other studies were done with the USA or Australian standards, so this exploration of ICD code representation using the Chilean standards can be very useful for future national work in the area.

The main objectives of this research are:

- Explore and learn new representations for ICD codes
- Test the new representations in the readmission task and generate benchmarks for the task using different classifiers with an appropriate evaluation framework
- Make the output of this work publicly available (English-Spanish translation of the ICD codes used in Chile, learned embeddings for the codes and the graph structure of the ICD codes in a simple format)

We hypothesize that it is possible to achieve results close to state of the art using a simpler dataset, just the ICD codes, if they are adequately represented. There is room for improvement in the ICD representation for readmission prediction since ICD codes have a description associated with them that has not been previously used, and NLP techniques have greatly improved in recent years. Also, no readmission prediction works have exploited the hierarchical nature of ICD codes or their sequential nature in the EHR. Previous representations that were learned from the data used local contexts, whereas ICD codes have a global context that should be considered.

5. DATA

In the data science community, it is widely held that 80% of the effort in any modeling process comes from the preprocessing, merging, and cleaning of datasets (Rajkomar et al., 2018). This thesis was not an exception. In this section, we go over the most labor-intensive part of this work, cleaning the original dataset and producing an end version that could be used to meet our needs:

- Have correct demographic data, procedure sets, and diagnosis set for all episodes
- Calculate a readmission flag per episode
- All the diagnosis and procedure codes need to have an English description and be part of a valid and clear hierarchical structure

5.1. Original dataset

After receiving approval from the ethics committee, the dataset was received from our School of Medicine collaborators at PUC Chile. The dataset comes from the Diagnostic Related Group (DRG) Database from the main hospital in Chile's Salud UC CHRISTUS Healthcare Network. The dataset contains all the discharges that occurred during the period 2014-2018. The hospital where the database comes from is a 500-bed academic medical center that includes all main clinical specialties and discharges about 30,000 patients per year. As part of the post-discharge administrative process, a team of expert nurses reviews every episode and generates the DRG entries.

The original dataset has 301400 rows and 268 columns. Each row consists of a DRG entry. The two most important keys are 'episode' and 'patient' (one patient can have many episodes). The dataset represents the hospital activity for five years, and it includes information on demographics, diagnoses, procedures, and readmissions.

Table 5.1 shows a comparison with the dataset used in (Rajkomar et al., 2018) in order to solve the Readmission Problem:

Table 5.1. Comparison between (Rajkomar et al., 2018) and our dataset.

	A - Rajkomar et al.	A - Rajkomar et al.	UC
Size training data	85522	108948	129791 (episodes)
Age median	56	57	44
% female	46.8	62	57
30 day readmission	9136 (10.7%)	15932 (14.6%)	13544 (10.4%)
Hospital stays at least 7 days	20411 (21.9%)	26109 (24%)	21970 (16.9%)
N diagnosis median	12	10	4

The dataset was kept in an access-controlled sandbox.

5.2. Challenges

Several problems and challenges were identified when working with this dataset. They will be described in this section and solved along the preprocessing pipeline. The identification and addressing of these challenges are considered a key contribution of this thesis:

(i) The dataset is in Spanish, which complicates data encoding and NLP work since most models are pretrained with English corpora.

(ii) The dataset is extracted manually by someone inside the hospital. Not knowing how each column was parsed requires a degree of inference and understanding of the problem. For example, procedure codes were parsed as floats when extracted instead of strings, losing an additional zero in the codification that cannot be recovered.

(iii) The dataset includes several years of data, and the systems within hospitals are continuously updated without necessarily guaranteeing backward compatibility.

(iv) Data inconsistencies (i.e., same patient with different demographic values, a diagnosis code with multiple descriptions).

(v) Multiple rows per episode, which leads to subjective analysis to define the correct way of aggregating a feature's values.

(vi) Codes in the dataset correspond to the CIE standard (Spanish translation of the ICD standard), and it is not clear which version of the CIE codes is used and to which ICD version it maps, or even if there is a unique mapping in the dataset.

(vii) Abbreviated descriptions of the codes are used, so using machine translation to get the English representation is impossible.

(viii) The hospital externalizes the coding process. Until 2016 they used a tool called DRGFinder, which was designed to codify both procedures and diagnoses codes in ICD-9-CM. When Chilean institutions started using ICD-10 to code diagnoses, the external company provided a manual mapping between both standards. This is not a perfect process, there are errors, and not all codes have a mapping. These errors are reflected in the dataset (i.e., some diagnoses codes look like procedure codes because there was no proper mapping in the internal tool).

5.3. Preprocessing pipeline

The preprocessing pipeline is represented in Figure 5.1.

There is an initial stage of **preparation**, transforming the original files into a more friendly format. This is followed by the **data extraction** phase that extracts the relevant data from the original dataset. After this, a throughout **code analysis**, for both procedure and diagnosis codes, is performed to solve several of the challenges related to the codes. Finally, the pipeline finalizes with the **integration** stage to get the final dataset used for training.

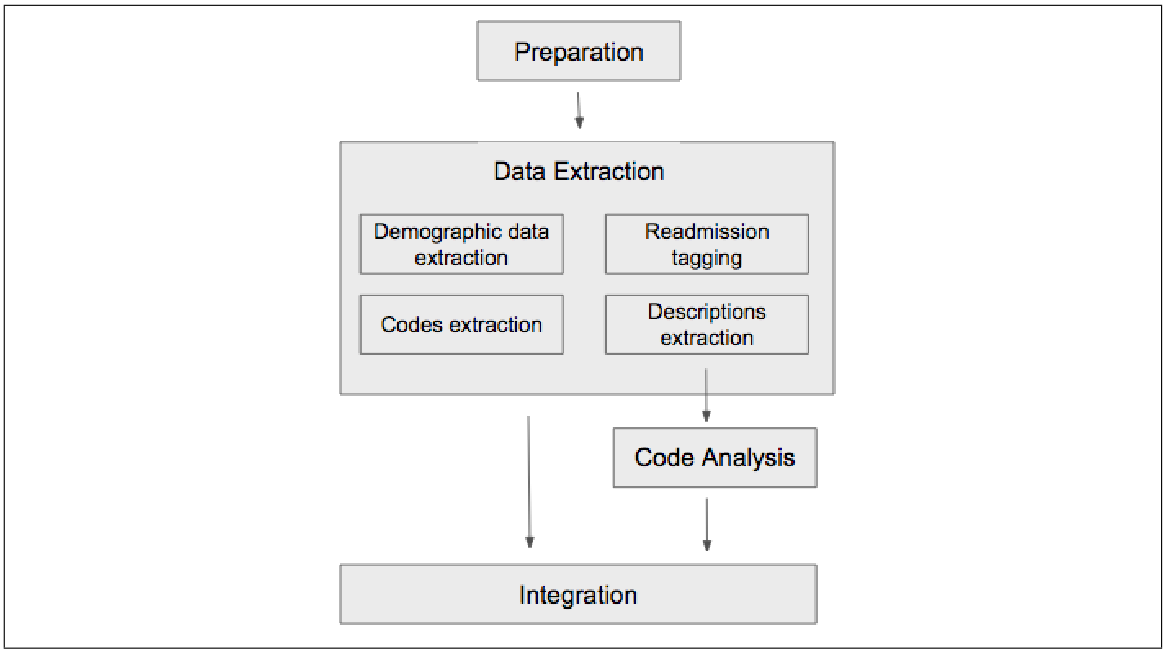


Figure 5.1. Preprocessing pipeline

For conciseness, the details of every step of the preprocessing pipeline are described in Appendix B, and here we only give an overview of every step:

(i) **Preparation**: Merges original files, fixed encoding issues, and outputs a single CSV file.

(ii) **Data extraction**: In four parallel sub-stages it extracts the relevant data from the CSV file (demographic data, readmission tag, codes and descriptions). This stage's outputs are three key-value structures with episodes as keys and the extracted information as values and one key-value structure with codes as keys, and a set of descriptions as values.

(iii) **Code analysis**: Code-description pairs are thoroughly analyzed. Several sources are used to create a ground truth of valid codes with Spanish and English descriptions to solve discrepancies. This stage's outputs are the set of valid codes with their Spanish and English descriptions, a set of codes from the original dataset considered invalid, and the graph structure for the valid procedure and diagnosis codes.

(iv) **Integration**: Final stage that merges the data extracted in the Data Extraction stage, filtering out episodes with missing data and with a high number of invalid codes as defined by the Code analysis stage. The output of this stage is the final dataset used in this work.

The final mappings generated in the Code Analysis stage were published in a Github repository[1]. This repository is a key contribution of this thesis. It provides mappings for both procedures and diagnoses with the current standards used in the Chilean healthcare systems in Spanish and English. This considerably facilitates further researchers that want to work with this data.

5.4. Final dataset

The final dataset has 92933 episodes and 68516 patients, with a 10% readmission rate. 58% of the patients are women. The average age is 44 years, and the average length of stay is five days. The distribution of the number of diagnoses and procedures is plotted in Figures 5.2 and 5.3. The distribution of the other features can be found in Appendix C.

Every episode's features are sex, age, length of stay (LOS), DRG weight, number of procedures, number of diagnoses, set of diagnoses, set of procedures, and readmission flag. For simplicity, from now on, all features that are not the ICD codes and the readmission flag will be referred as demographic features (even though they include features like DRG weight and number of procedures, which are not technically demographic).

[1] https://github.com/tamycova/icd-cie-codes-chile

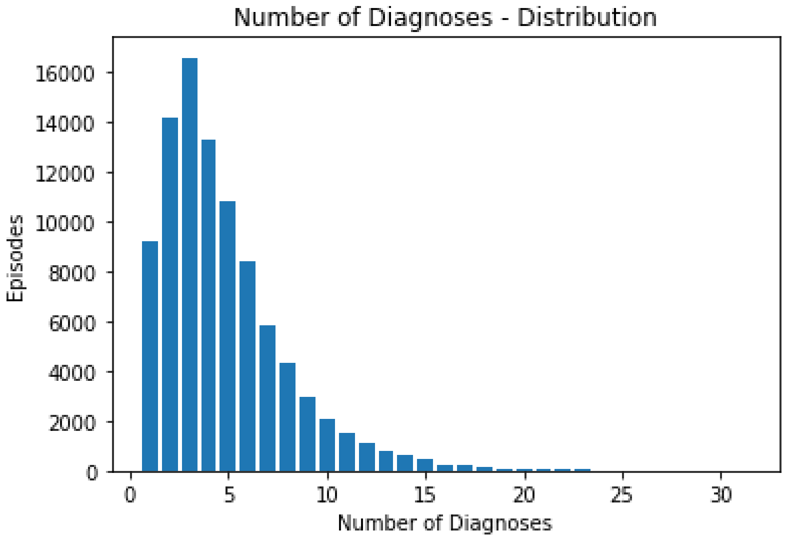


Figure 5.2. Distribution for Number of Diagnoses.

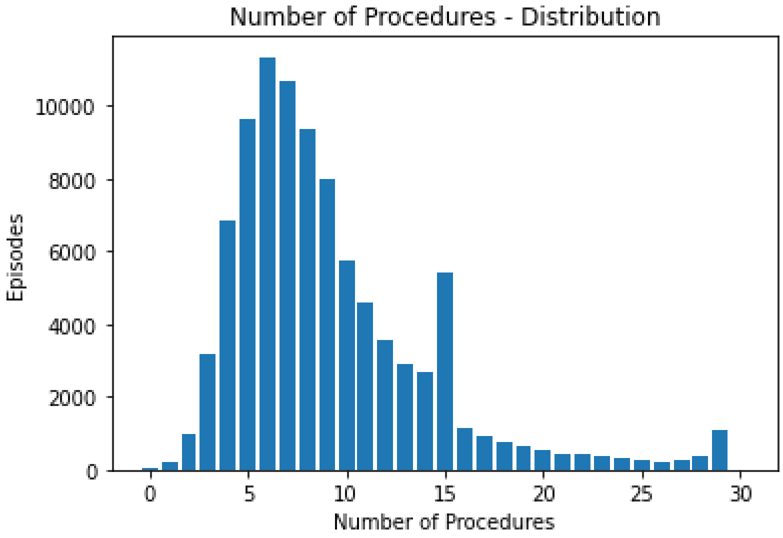


Figure 5.3. Distribution for Number of Procedures.

6. REPRESENTATIONS

6.1. Categorical

Similarly to (Futoma et al., 2015) the first representation will be one-hot encoding, which will also be considered the baseline. The dataset has 7355 different diagnosis codes and 2697 distinct procedure codes, which means that a diagnosis code can be represented with a 7355-sized vector with a ”1” in the position corresponding to the diagnosis. A procedure code can be represented with a 2697-sized vector with a ”1” in the position corresponding to the procedure code. Correspondence is fixed and randomly determined.

With this representation, a single episode can be represented concatenating both representations into a 10052 sized vector, where the first 7355 positions will have a ”1” if the respective diagnosis code is present in the episode and the other 2697 positions will have a ”1” if the respective procedure is present in the episode, as shown on Figure 6.1.

Although this research does not focus on DRG codes, we also built a one-hot encoding for this feature for some experiments; there are 859 DRG codes.

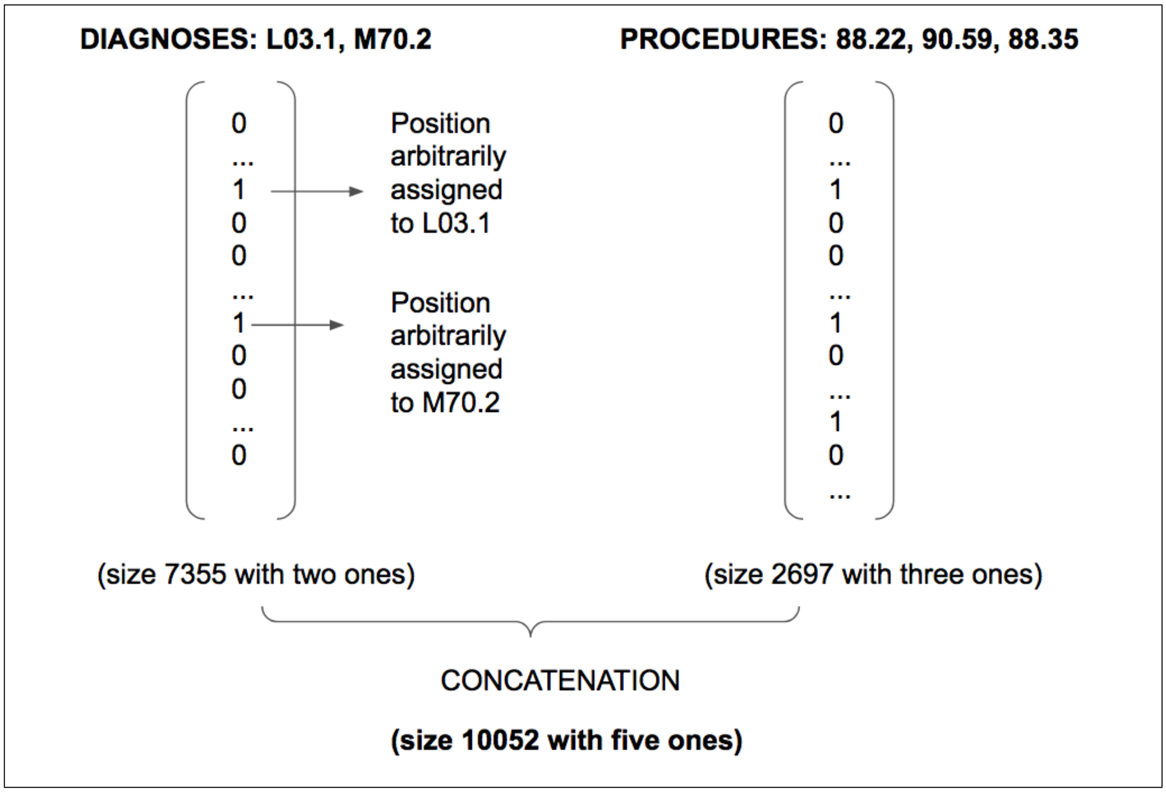

DIAGNOSES: L03.1, M70.2
PROCEDURES: 88.22, 90.59, 88.35
0
...
1
0
0
...
1
0
0
...
0
Position arbitrarily assigned to L03.1
Position arbitrarily assigned to M70.2
0
...
1
0
0
...
1
0
...
1
0
...
(size 7355 with two ones)
(size 2697 with three ones)
CONCATENATION
(size 10052 with five ones)

Figure 6.1. Categorial representation for an episode with two diagnoses and three procedures. The vector that represents the episode is the concatenation of its diagnosis vector with its procedure vector.

6.2. Text description

As we saw in Section 5, ICD codes have a text description associated with them that has never been used to represent them.

BERT (Devlin et al., 2019) is a deep neural network that uses the transformer encoder architecture (Vaswani et al., 2017) to learn embeddings for text. We omit a detailed description of the architecture since it is not the focus of this research, and it can be found in the cited literature. This model is used in (Huang et al., 2019) to learn better representations for the episodes from the discharge summaries, which are later used to classify for readmission (Figure 6.2).

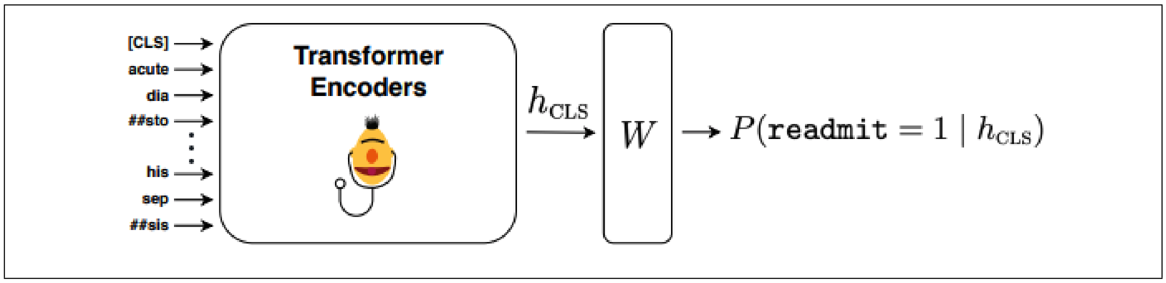


Figure 6.2. Figure from (Huang et al., 2019), with permission of the author. ClinicalBERT models clinical notes and is finetuned on 30-day readmission prediction. The model receives as input a patient's clinical notes, and then the probability of readmission is obtained using a linear layer applied to the classification representation for the CLS token learned by ClinicalBERT.

The quality of learned representations of text depends on what text the model was trained on. Original BERT is trained on Wikipedia and BookCorpus. Fortunately, work has been done to train BERT in the biological domain. This is important because word distribution between general domain corpora and biomedical corpora is shifted. (Lee et al., 2019) continued training BERT on large-scale biomedical corpora. Starting with the pretrained BERT, they trained using PubMed abstracts and PMC full-text articles. From those weights, (Alsentzer et al., 2019) continued training into the clinical domain using all notes from MIMIC-III, a database containing EHR from ICU patients, and made the pre-trained weights publicly available in HuggingFace (Wolf et al., 2019), a large open-source community.

This training chain took significant computational resources. BERT's base model is pretrained for four days on four to sixteen Cloud TPUs. BioBERT pretrained on biomedical corpora for 23 days on eight NVIDIA V100 GPUs. (Alsentzer et al., 2019) took 18 days using a single GeForce GTX TITAN X 12 GB GPU. We can directly benefit from this work.

Since every ICD code has an English description associated with it, every description can be given as input to the BERT model using (Alsentzer et al., 2019) weights and then use the CLS token as a representation of the code, size 768.

An episode can be represented as the concatenation of the descriptions for all the diagnoses and procedures codes and then give this as an input, which in turn provides us with a vector (the one associated with the CLS token) that represents the episode, or we can directly classify with a layer on top of the model (Figure 6.3).

An advantage of this representation is that codes don't need to be truncated to higher levels like they do in (Nguyen et al., 2018), since their descriptions already include (generally) the ancestor information.

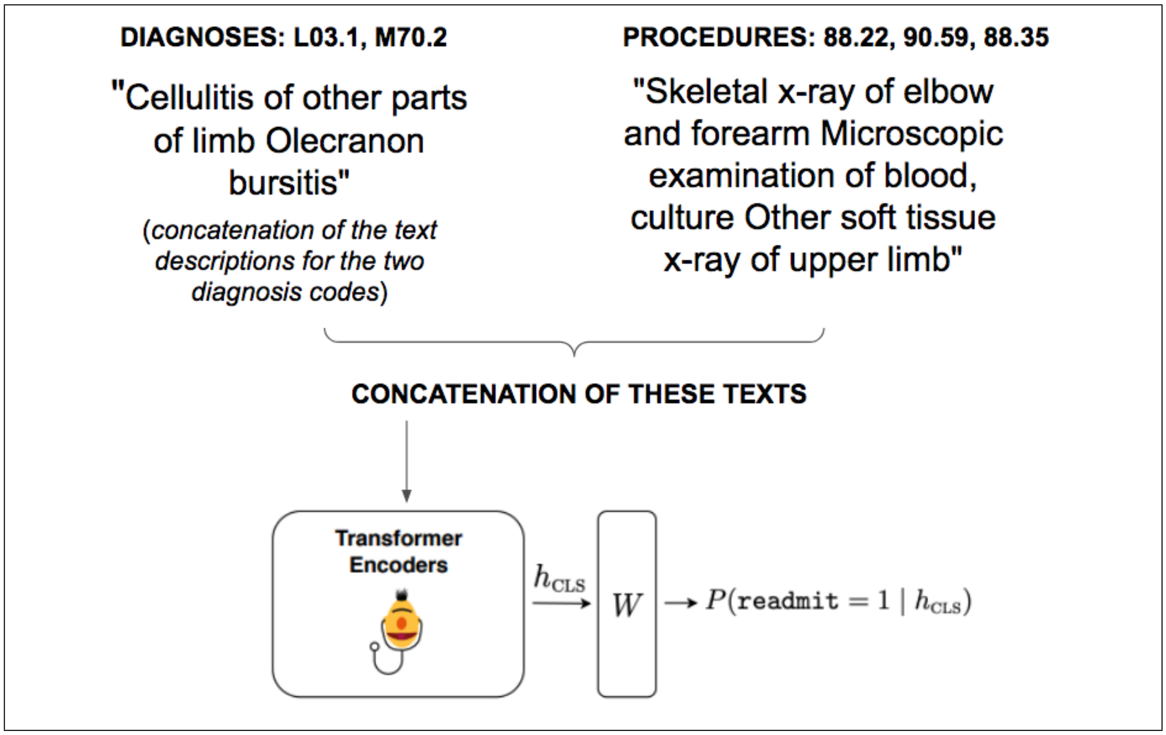

DIAGNOSES: L03.1, M70.2
"Cellulitis of other parts of limb Olecranon bursitis"
(concatenation of the text descriptions for the two diagnosis codes)
PROCEDURES: 88.22, 90.59, 88.35
"Skeletal x-ray of elbow and forearm Microscopic examination of blood, culture Other soft tissue x-ray of upper limb"
CONCATENATION OF THESE TEXTS
Transformer Encoders
h_{CLS}
W
$P(\texttt{readmit} = 1 \mid h_{\text{CLS}})$

Figure 6.3. Text representation for an episode obtained concatenating the text descriptions of its ICD codes. This text is later fed to a pretrained BERT model, similarly to (Huang et al., 2019). This Figure shows one text representation, concatenating diagnosis text with procedure text. Other representations are also possible; they are explored in Section 7.

6.3. Ontology

Table 6.1. Ontology characteristics.

	Edges	Nodes	Structure	Depth	Leaves
ICD-10 Diagnoses	14302	14324	tree	4	11607
ICD-9-CM Procedures	4647	4664	tree	7	3878
Hospital	1403450	180434	graph	-	-

ICD codes for both diagnosis and procedure can be represented with a tree structure. The procedures tree has 18 initial nodes (for example, operations on the nervous system, operations on the eye, operations on the ear, are some of those initial nodes) and depth 4. The diagnoses tree has 22 initial nodes (for example, mental and behavioral disorders, diseases of the respiratory system are some of those initial nodes) and depth 7. In Figure 6.4 and Figure 6.5 there is a visual representation of a subpart of the ontologies, Figure 6.6 provides another view of how the diagnoses ontology looks. We can see that nodes ”D50” and ”D51” have the same parent, ”Nutritional anemias”, but one of them refers to Iron deficiency and the other one to Vitamin B12 deficiency. Further information about these graphs can be found in Table 6.1.

Also, the whole dataset can be represented as a graph, including the ICD ontologies. Patients have episodes, episodes have diagnoses and procedures codes, and the codes are related to each other through the ontologies.

Having the graph representations for each ontology (a list of edges), it is possible to use graph embedding algorithms to generate embeddings for every node in the graph. The idea is that the representation reflects the ontology structure, so nodes related in the tree should be closer than unrelated nodes or farther related. With this in mind, it is possible

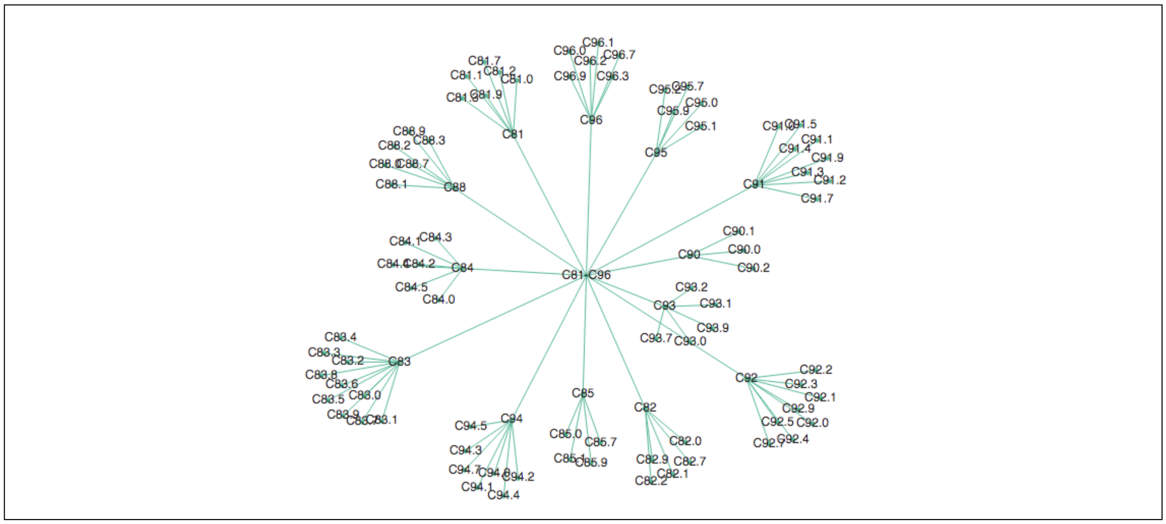


Figure 6.4. Subtree in the diagnoses graph with ”C81-C96 Malignant neoplasms, stated or presumed to be primary, of lymphoid, haematopoietic and related tissue” as root

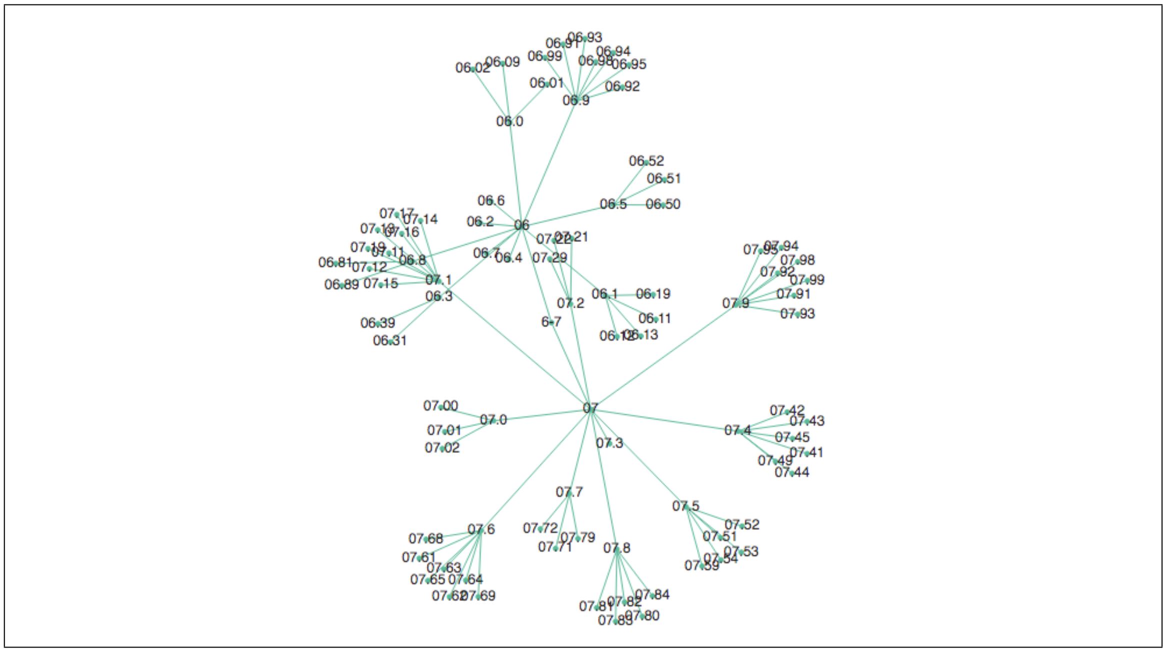
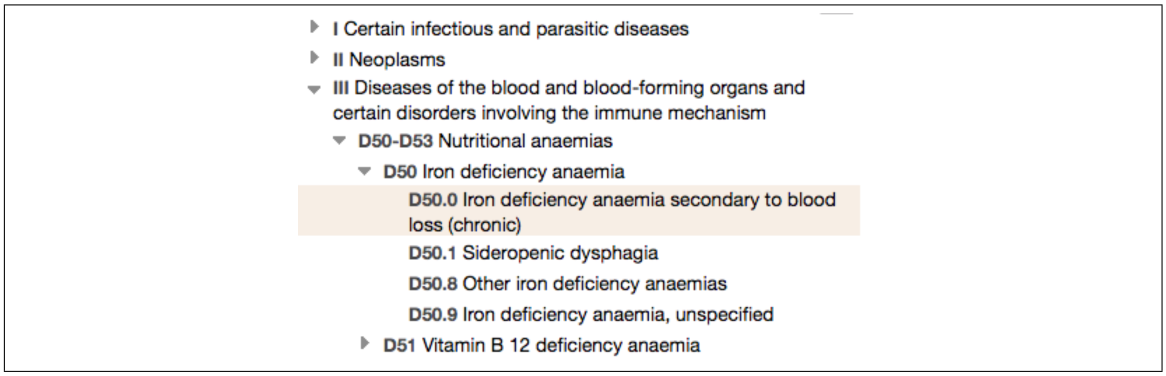


Figure 6.5. Subtree in the procedure graph with ”06-07 Operations on the Endocrine System” as root

to generate embeddings for all ICD diagnoses and procedures from their respective ontology in any determined size. Single episodes can also be represented using their node embedding in the Hospital graph.



- ▶ I Certain infectious and parasitic diseases
- ▶ II Neoplasms
- ▼ III Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism
  - ▼ D50-D53 Nutritional anaemias
    - ▼ D50 Iron deficiency anaemia
      - D50.0 Iron deficiency anaemia secondary to blood loss (chronic)
      - D50.1 Sideropenic dysphagia
      - D50.8 Other iron deficiency anaemias
      - D50.9 Iron deficiency anaemia, unspecified
    - ▶ D51 Vitamin B 12 deficiency anaemia

Figure 6.6. Another view of the ICD-10 Diagnoses structure

In this work, we will explore two different algorithms to generate these embeddings.

6.3.1. Node2Vec

Node2Vec (Grover & Leskovec, 2016) performs representational learning on graphs. That is, given any graph, it learns continuous feature representations for the nodes. They do so by performing biased random walks and maximizing the likelihood of preserving network neighborhoods of nodes with the learned representation. They report their results using the algorithm in networks that range in the number of nodes between 3K and 10K and in the number of edges between 70K and 300K, so it is a good option for small ontologies.

6.3.2. PyTorch-BigGraph

Pytorch-BigGraph (PBG) (Lerer et al., 2019) is a recent contribution of Facebook to the graph embedding open-source frameworks ecosystem. PBG is a distributed system that learns graph embeddings for large graphs (for example, web interaction graphs with billions of nodes and trillions of edges). It is worth noting that PBG is not optimized for small graphs (fewer than 100000 nodes) and the authors highlight that it may not produce high-quality embeddings in those graphs. This is because during training, the algorithm constructs random false edges as negative training examples; it relies on the assumption that any random edge is negative with very high probability (which is valid for large sparse graphs).

7. EXPERIMENTAL DESIGN

7.1. Hardware

Experiments that ran on CPU used an 8 core (Intel i7-7700K CPU @ 4.20GHz) machine with 62.9G of RAM. Experiments that ran on GPU (mostly the text representation experiments) used two GeForce GTX 1080 Ti X 12 GB GPU.

7.2. Data

For every episode, we have the set of diagnoses codes, the set of procedure codes, readmission tag, and extra demographic data (including the DRG weight).

A split consists of splitting the data into a training set (0.8 of the data), a test set (0.1 of the data), and a validation set (0.1 of the data) in a stratified fashion. Hence, the readmission percentage is roughly the same across all datasets. The balanced version of the split has the same testing and validation sets, but the training set increases its size via random over-sampling of the minority class by picking samples at random with replacement until the dataset is balanced. Five splits are generated at random, and all of them have a balanced version. Experiments ran on the five splits, and the reported results correspond to the mean of the metrics obtained over all the splits.

On average, the unbalanced training set has 74346 episodes with 10% readmissions, the balanced training set has 133012 episodes with 50% readmissions, and the testing and validation sets have 9293 episodes with 10% readmissions.

7.2.1. Poolings

Given representation $\vec{u} =< u_1, u_2 >$, $\vec{v} =< v_1, v_2 >$ and $\vec{k} =< k_1, k_2 >$ for three ICD diagnoses code in an episode, they all have the same length because the embedding was generated on the same experiment run (in this case, and for simplicity, the size is two).

When we need to represent the episode that contains these three codes we perform three types of pooling:

(i) Sum pooling: The sum of all the vectors $s\vec{u}m =< u_1+v_1+z_1, u_2+v_2+z_2, u_3+v_3+z_3 >$.

(ii) Mean pooling: The mean of all vectors $\frac{s\vec{u}m}{\|\vec{u}\|}$

(iii) Weighted sum pooling: Vectors are summed but with a linearly decaying weight, in this case the weights would be 0.5 for $\vec{u}$, $0.3\overline{3}$ for $\vec{v}$ and $0.1\overline{6}$ for $\vec{z}$.

7.3. Models

Even though we are focused on the representations (which have their own baseline), we also compare performance on two classifiers that were previously used in literature. Similarly to (Futoma et al., 2015) and (Nguyen et al., 2017), we use Logistic Regression as the baseline classifier. Additionally, we use XGB, which corresponds to a decision tree ensemble (Chen & Guestrin, 2016). Both models are implemented using Scikit-Learn (Pedregosa et al., 2011). The models will not be finetuned, and they will be used with the same (default) parameters for all experiments. We chose these models based on the results in (Futoma et al., 2015) and because we want to have a degree of explainability in the discussion, so using robust models that also give us feature importance is relevant for our analysis.

As mentioned in Section 6, we use BERT (Devlin et al., 2019) loaded with different pretrained weights (Alsentzer et al., 2019) as a classifier for the text representation; the implementation uses the transformers library (Wolf et al., 2019), which also provides the appropriate tokenizer for each model. Every finetuning experiment uses learning rate 3^{-05}, is trained over four epochs, and the rest of the parameters default to the library's Trainer. If the training set is unbalanced, we modify the loss function (cross-entropy) to consider a weight for the classes; the weight will be a training parameter in the experiments. Because of memory constraints, the batch size depends on the number of trained layers, so the pair number of layers/batch size is also a training parameter in the experiments.

7.4. Evaluation

These will be the metrics used to analyze the results of the experiments:

- AUC: Area Under the Receiver Operating Characteristic Curve (ROC AUC) using the predicted probabilities.
- RP60: Recall at precision 60%.
- RP80: Recall at precision 80%.
- Accuracy: Accuracy using 0.5 as the predictive threshold.

AUC and Accuracy were chosen to compare our work to previous studies. RP60 and RP80 were chosen to establish benchmarks for the readmission problem that are more representative of the usefulness of the prediction compared to what has been previously reported. Because of alarm fatigue, useful classification rules for medicine should have high precision (Huang et al., 2019), but this comes at a cost of recall. With different predictive thresholds, different values for precision and recall are obtained. These can be plotted in a precision-recall curve. This curve can also be smoothed by interpolation, as Figure 7.1 shows, by fixing the recall at 11 points (0, 0.1, 0.2, 0.3, ..., 1.0) (Manning, Raghavan, & Schütze, 2008). The precision-recall curve will be used in our analysis, and for every experiment, we will also report recall at a fixed precision of 60% and 80%, relevant metrics for the medical setting.

Additionally, and when possible, we use visualizations to interpret the trained model.

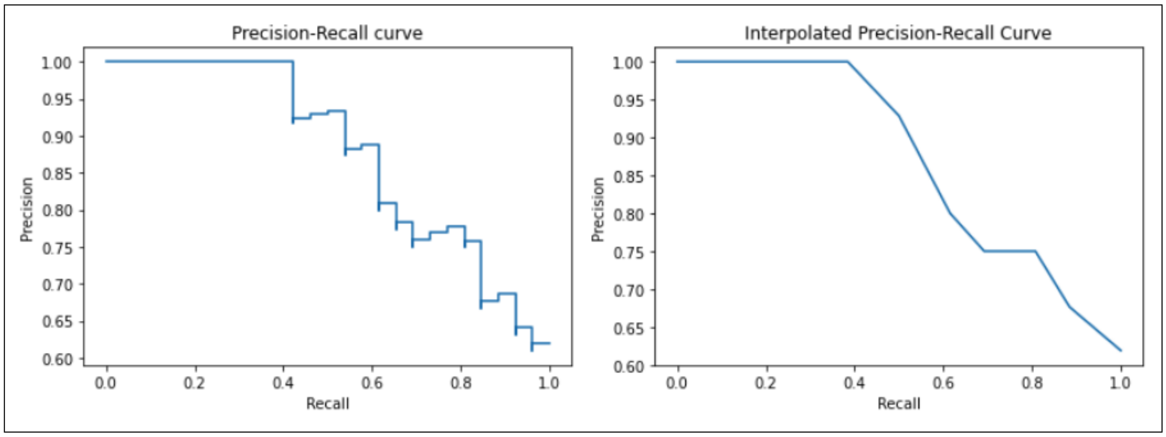


Figure 7.1. Example of a Precision-Recall Curve and its 11-point Precision-Recall curve.

7.5. Experiments

The same experiments were performed on every data split.

7.5.1. Categorical

Since every episode has a vector representing their ICD codes (diagnosis vector concatenated with procedure vector), a set of demographic variables, and a vector representing the DRG code, we performed a series of experiments to evaluate which combination of data was the most useful for readmission prediction.

A combination of these boolean variables defines the dataset used in every experiment:

- Balanced (True if the training data used is balanced)
- Demographic (True if demographic data is part of the dataset)
- DRG (True if DRG vector is in the dataset)
- ICD (True if ICD vector is in the dataset)

Therefore, for every classifier and split, we performed 14 runs.

7.5.2. Text description

7.5.2.1. Static

To confirm that we can benefit from the pre-training on clinical data, we use every episode's representation as the concatenation of their diagnosis and procedure's descriptions and input it to BERT-Base (pretrained on Wikipedia), Bio+Clinical BERT (initialized from BioBERT and trained on all MIMIC notes) and Bio+Discharge Summary BERT (initialized from BioBERT and trained only on discharge summaries) and extract static representations (from the CLS token), which are then used to train the classifiers.

A combination of these variables defines the dataset used in every experiment:

- Balanced (True if the training data used is balanced)
- Demographic (True if demographic data is part of the dataset)
- Input type (Three different ways to input the code's descriptions: diagnoses descriptions concatenated with procedures descriptions to produce a 768-sized representation, diagnoses descriptions interweaved with procedures descriptions to produce a 768-sized representation, diagnoses descriptions separated from procedures descriptions and then both vectors are concatenated to produce a 1536-sized representation)

Therefore, for every classifier, split, and BERT model, we performed 12 runs.

7.5.2.2. Dynamic with classification layer

For this experiment, we use the BERT model with a classification layer on top (a pooler that extracts the CLS token and then two neurons for both classes) initialized with Bio+Clinical BERT's weights.

A combination of these variables defines every experiment:

- Balanced (True if the training data used is balanced)
- Frozen Layers/Batch size (0/10, 5/20, 10/64, 11/74)
- Weights (this is only if Balanced is False, 0.1, 0.15, 0.3, 0.5)

Therefore, for every split, we performed 20 runs (four for the balanced dataset and sixteen for the unbalanced dataset).

7.5.2.3. Dynamic with traditional classification

We finetune a single model (zero frozen layers, batch size 10, unbalanced dataset, weight 0.3, first split) and remove the classification layer. Similarly to the static experiments, we extract the training set representations and then train the classifiers with and without the demographic data. This is done with the same training set used to finetune the initial model (we don't test the classifier's performance on an example that was already seen when the representation was finetuned). In this experiment, two runs are performed per classifier.

7.5.3. Ontology

7.5.3.1. Node2Vec

Using Node2Vec, we obtain representations for every ICD code from their respective ontology. We obtain representations in two different sizes, size 100 and size 768 (so it is comparable to the embeddings obtained from BERT). We then perform the three poolings discussed in Section 6 and obtain a diagnosis and a procedure vector for every episode. In summary, every episode has six possible representations for its diagnoses vector (the combination of two sizes and three different poolings), and the same holds for its procedures vector.

In every experiment, we instantiate a dataset with a different representation and then train the classifiers. A combination of these variables defines an experiment:

- Balanced (True if the training data used is balanced)
- Demographic (True if demographic data is part of the dataset)
- Size (30 or 768)
- Pooling (Mean, Sum or Weighted Sum)
- Concatenate (True if to represent an episode we concatenate the diagnosis vector with the procedure vector, if False we average both vectors)

Therefore, for every classifier and split, we performed 48 runs.

7.5.3.2. PBG codes

The set up for this experiment is the same as with the Node2Vec case, except the representation for every ICD code is learned from the ontologies using Pytorch-BigGraph (Lerer et al., 2019).

7.5.3.3. PBG hospital - code level

This experiment's setup is the same as with the Node2Vec case, except the representation is learned from the dataset represented as a graph as discussed in Section 6 using Pytorch-BigGraph (Lerer et al., 2019).

7.5.3.4. PBG hospital - episode level

Representing the dataset as a graph allows us to get representations for the ICD codes and the episodes. We obtain these representations (in size 30 and size 768) using Pytorch-BigGraph (Lerer et al., 2019) and then train the classifiers.

In every experiment, we instantiate a dataset with a different representation and then train the classifiers. A combination of these variables defines an experiment:

- Balanced (True if the training data used is balanced)
- Demographic (True if demographic data is part of the dataset)

- Size (30 or 768)

Therefore, for every classifier and split, we performed eight runs.

8. RESULTS

This section discusses our results and trade-offs of every representation; for detailed numerical results, see Appendix D.

8.1. One-hot encoding

These experiments had a high memory consumption (8 GB of RAM to store in memory one full dataset) due to the sparse representation. Even though bag-of-words does not add any semantic or local context to the vectors, it provides a clear overview of the episode since there is no pooling of codes.

This representation is our baseline, but it already achieves state of the art results. This is not a surprise considering the literature analysis, with reported results not supported by a solid evaluation framework.

Overall, XGBoost had a better performance than Logistic Regression. Figure 8.1 shows the results for both classifiers trained with all possible data points (ICD codes, DRG code, Demographic data). AUC is similar, but XGBoost has a considerably better recall. From now on, the rest of the analysis will focus on the XGBoost results. Figure 8.1 shows the interpolated precision-recall curves of the XGBoost model for all splits (on the same dataset), there are some differences, but overall the tendency is the same.

Table 8.1. Results for both classifiers on the full dataset (ICD + Demographic + DRG).

Model	Dataset	AUC	rp60	rp80	acc
LR	Balanced	0.8	0.18	0.02	0.9
LR	Unbalanced	0.79	0.2	0.04	0.9
XGB	Balanced	**0.81**	**0.29**	**0.13**	0.78
XGB	Unbalanced	**0.81**	**0.29**	0.12	**0.91**

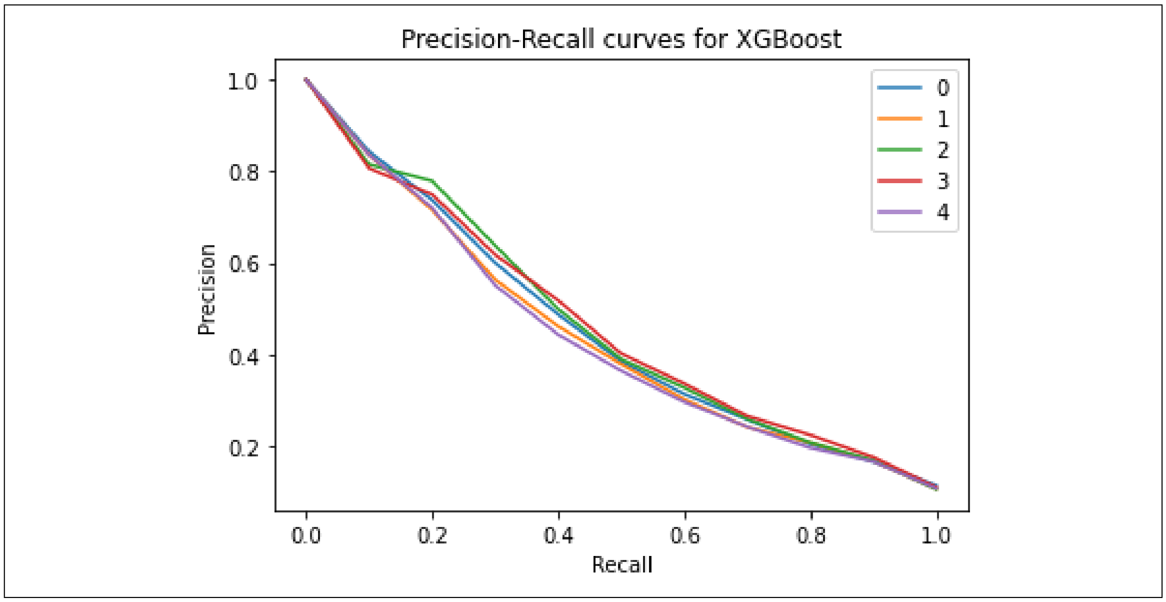


Figure 8.1. Every curve represents the interpolated precision-recall curve of the XGBoost classifier trained with ICD codes, DRG code, and Demographic data on a single split.

The best-performing model achieved a mean AUC of 0.81, comparable to the results obtained in (Futoma et al., 2015) (Rajkomar et al., 2018) (Huang et al., 2019) (Fierro et al., 2020) and mean accuracy of 0.91, similar with (Nguyen et al., 2017). This does not mean that the model is great at solving the problem, obtaining a recall at precision 80% of 0.13 which is slightly low compared to (Huang et al., 2019) but higher than (Fierro et al., 2020). Regardless, the metric is too low to motivate implementation in the clinical sites.

Even though the best model was trained with the unbalanced dataset, demographic data, and codes (no DRG code), the results are very similar to other combinations. As we can see in Table 8.2, ICD codes are the piece of data that provides more information on its own, adding the DRG code does not improve the results. This can be expected since the ICD information is an input to the model that assigns the DRG code to the episode. This means that there is no need to increase the representation's size and sparsity by also adding the DRG code one-hot vector. Demographic data does provide an improvement.

This makes sense considering it includes the DRG weight, which indicates how abnormal this episode is in resource use, surely an important indicator for readmission.

There is no significant difference between the balanced and unbalanced datasets. Furthermore, the unbalanced dataset has better recall than the balanced dataset, which aligns with the results of (Fierro et al., 2020).

Table 8.2. Dataset combinations with one-hot encoding (XGB).

Data	AUC bal	AUC unbal	rp80 bal	rp80 unbal
Demographic	0.76	0.77	0.07	0.07
DRG code	0.77	0.77	0.06	0.03
ICD codes	0.8	0.8	0.09	0.11
ICD + DRG	0.8	**0.81**	**0.11**	0.09
ICD + Demographic	**0.81**	**0.81**	0.09	**0.13**

An advantage of this representation is the inherent meaning of the embeddings. Their meaning does not come from their position in a continuous space but the exact codes composing an episode. This facilitates explainability. In Figure 8.2 we see the explanation for a correct prediction of a positive example using SHAP (Lundberg & Lee, 2017), a method that assigns each feature an importance value for a particular prediction. The plot was generated using Microsoft's InterpretML library (Nori, Jenkins, Koch, & Caruana, 2019). The model gives high importance to the length of stay, the sex of the patient, and the number of diagnoses. This is a clear example that not because a model achieves state of the art in the task at hand implies that its predictions make sense in a clinical setting. Demographic data should not be enough to assess a clinical discharge.

It is also possible to use the model's transparency to get a global explanation. Table 8.3 shows the top features of the model and the importance ranking (in terms of Gain) for

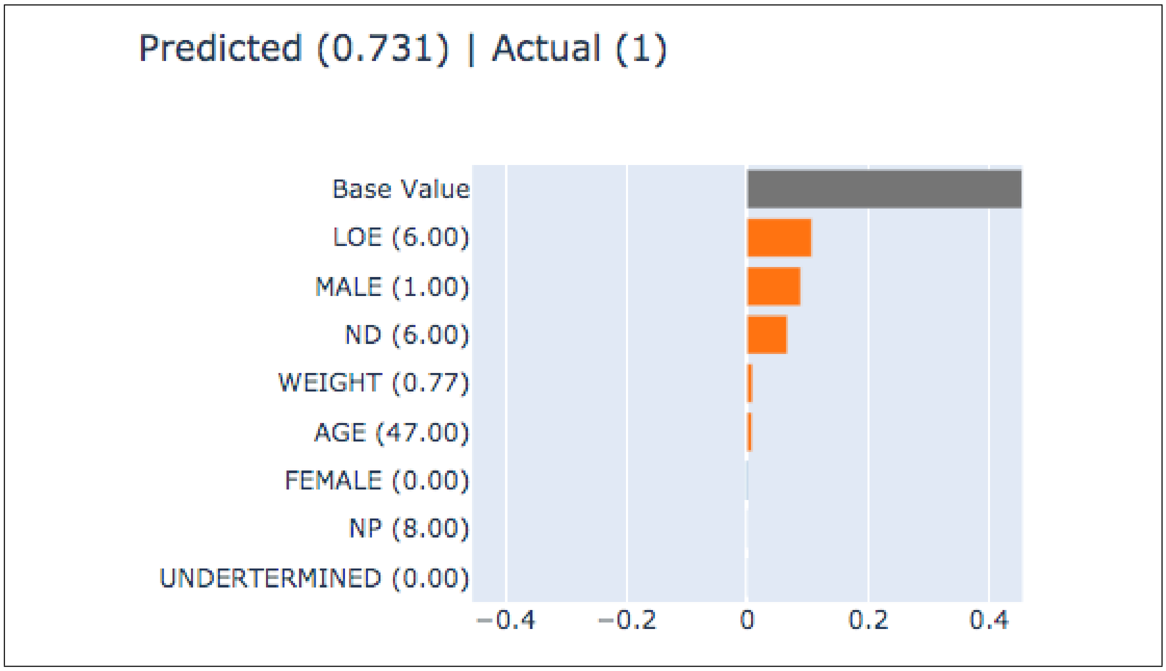


Figure 8.2. SHAP explainability for one-hot encoding using only demographic data.

some of the demographic features. We can see that the two most important features are procedures related to cancer, and the third one is related to birth. Demographic features are no longer the main predictors, and because of the nature of the encoding, it is expected that the model overfits certain codes' presence. This is an interesting insight; it motivates future research lines with analysis performed at the cohort level, similar to (Futoma et al., 2015).

Table 8.3. Feature importances in XGBoost trained with ICD codes.

Ranking	Feature	Weight	Meaning
1	99.25	0.051	Injection or infusion of cancer chemotherapeutic substance
2	51.23	0.009	Laparoscopic cholecystectomy
3	75.34	0.009	Other fetal monitoring
4	D70	0.008	Agranulocytosis
54	LOS	0.002	Length of Stay
159	ND	0.002	Number of Diagnosis
241	Weight	0.001	DRG Weight

Figure 8.3 shows how the SHAP explanation of the prediction changed now that codes are considered. We can see that the model now focuses more on the procedures rather than on the demographic data, which is the desired behavior.

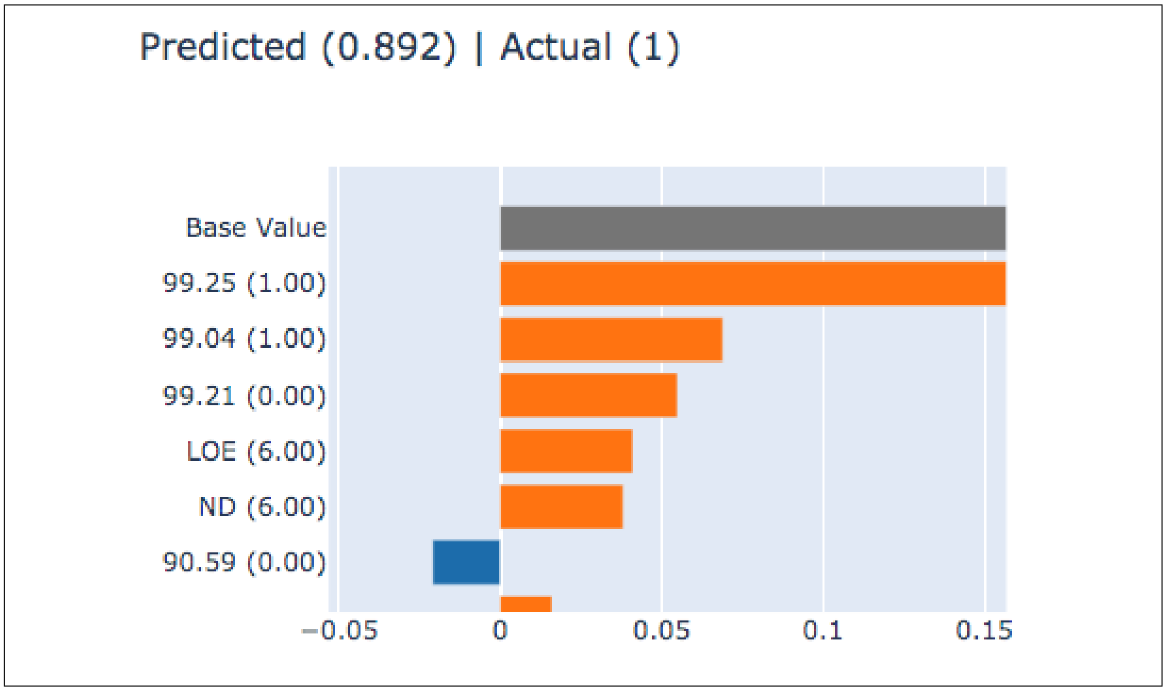


Figure 8.3. New SHAP explainability for one-hot encoding using ICD codes.

8.2. Static BERT

After concatenating text descriptions of all the codes in an episode, static representations were obtained for each episode using Bio+Clinical BERT, Bio+Discharge BERT, and Base BERT. Then classification was performed using these representations.

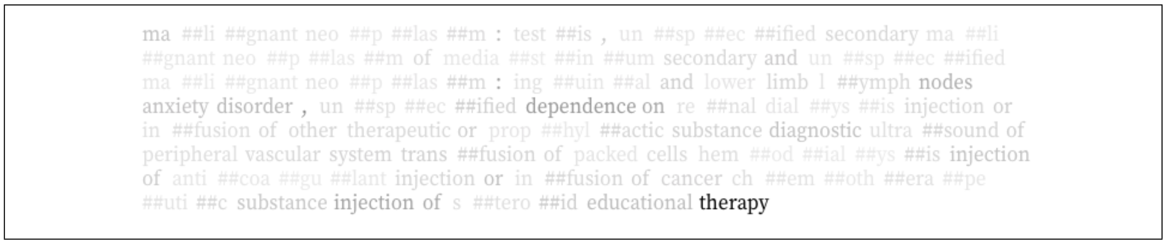
It was expected that Base BERT would underperform compared to the other two instances pretrained on medical specific data. This held, except the performance was not considerably different, as shown in Table 8.4. Again, the unbalanced dataset achieved better results, and XGBoost performs better than Logistic Regression (AUC is similar, but rp80 is higher).

Table 8.4. Best static BERT performances.

Model	Weights	AUC bal	AUC unbal	rp80 bal	rp80 unbal
LR	Bio+Clin	**0.79**	**0.79**	0.03	0.03
LR	Bio+Disch	**0.79**	**0.79**	0.03	0.05
LR	Base	0.78	**0.79**	0.03	0.06
XGB	Bio+Clin	0.77	0.78	0.05	**0.07**
XGB	Bio+Disch	0.78	0.78	**0.06**	**0.07**
XGB	Base	0.77	0.77	0.04	0.06

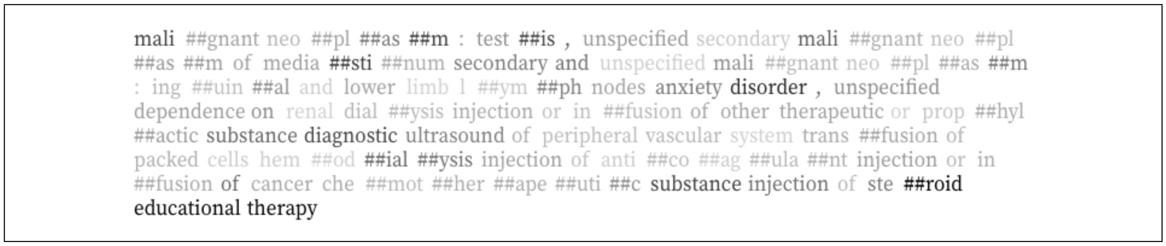
Adding demographic data also improved the results in these experiments. The best method of preparing the input was getting separate representations for the diagnoses and the procedures and then concatenating them as a final representation.

Regarding explainability, the BERT architecture has 12 multi-head attention mechanisms for each of the 12 transformer encoders. After training, each one of these mechanisms specializes in different patterns that are indicative of the task at hand (an NLP task in the case of the static experiments and Readmission in the case of dynamic experiments) (Huang et al., 2019). There are 144 attention values for every token and it is complicated to visualize and interpret this information. It is possible to stack these attentions by pooling (with mean or sum). It is not as interpretable as SHAP with one-hot encoding but at least it provides a visualization for what the model is paying attention to. (Parra et al., 2019) used design principles from information visualization to propose initial ideas for the visualization of these attentions. Figure 8.4 and 8.5 are examples of these visualizations, with the same patient whose prediction was analyzed using SHAP, we see that it seems that BERT Base distributes the attention along the whole text while Clinical BERT focuses on specific tokens.



ma ##li ##gnant neo ##p ##las ##m : test ##is , un ##sp ##ec ##ified secondary ma ##li
##gnant neo ##p ##las ##m of media ##st ##in ##um secondary and un ##sp ##ec ##ified
ma ##li ##gnant neo ##p ##las ##m : ing ##uin ##al and lower limb l ##ymph nodes
anxiety disorder , un ##sp ##ec ##ified dependence on re ##nal dial ##ys ##is injection or
in ##fusion of other therapeutic or prop ##hyl ##actic substance diagnostic ultra ##sound of
peripheral vascular system trans ##fusion of packed cells hem ##od ##ial ##ys ##is injection
of anti ##coa ##gu ##lant injection or in ##fusion of cancer ch ##em ##oth ##era ##pe
##uti ##c substance injection of s ##tero ##id educational therapy

Figure 8.4. Mean attention of cancer patient using BioClinical BERT.



mali ##gnant neo ##pl ##as ##m : test ##is , unspecified secondary mali ##gnant neo ##pl
##as ##m of media ##sti ##num secondary and unspecified mali ##gnant neo ##pl ##as ##m
: ing ##uin ##al and lower limb l ##ym ##ph nodes anxiety disorder , unspecified
dependence on renal dial ##ysis injection or in ##fusion of other therapeutic or prop ##hyl
##actic substance diagnostic ultrasound of peripheral vascular system trans ##fusion of
packed cells hem ##od ##ial ##ysis injection of anti ##co ##ag ##ula ##nt injection or in
##fusion of cancer che ##mot ##her ##ape ##uti ##c substance injection of ste ##roid
educational therapy

Figure 8.5. Mean attention of cancer patient using BERT Base.

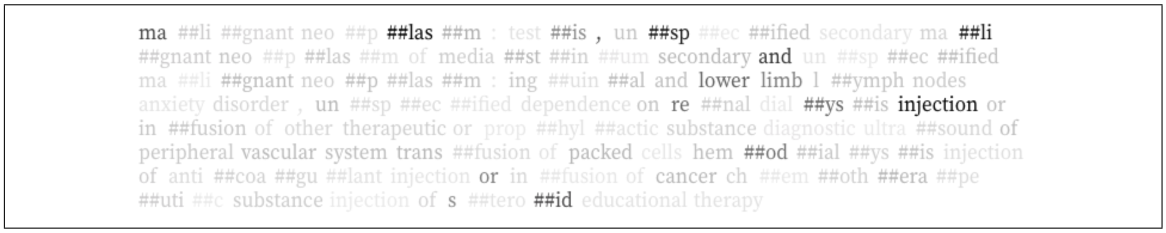
This experiment was the first attempt to use the codes' global context to solve the readmission problem. Since here episodes are represented as a concatenation of all codes, it did not perform as well as the baseline where different codes are easier to be individualized to predict based on their presence.

8.3. Finetuned BERT

Finetuning was performed feeding BioClinical BERT the concatenation of the ICD codes' text descriptions (per episode). These experiments took the longest to run (approximately 2 weeks). Using a balanced dataset (without special weights in the loss calculations), the best performing model obtained 0.08 rp80. This model had no frozen layers and batch size 10, so training is slower (three hours compared to a model with eleven frozen layers that takes to train less than an hour), so if training time is relevant, freezing some layers is worth considering since it did not considerably affect the results.

Similar to previous experiments, the unbalanced dataset worked better than the balanced dataset. The best result used a weight of 0.5 for both classes, with an AUC of 0.82 and rp80 0.11. This is the highest AUC in all experiments and is higher than all of the reviewed past works that performed classification on the global dataset (not by DRG). This shows that the text representation of an episode through ICD codes can achieve the state of the art results. The best results were achieved with an unbalanced dataset, this is surprising given the dataset's imbalance. It suggests that there is room for improvement, perhaps exploring other loss functions.

As expected, compared to the static experiments, results are better (almost double rp80). This highlights the importance of finetuning the models for the task at hand. Figure 8.6 shows the mean attention for the same patient as Figures 8.4 and 8.5. Like BioClinical BERT, it focuses on specific terms, but these terms are different after finetuning.



ma ##li ##gnant neo ##p ##las ##m : test ##is , un ##sp ##ec ##ified secondary ma ##li
##gnant neo ##p ##las ##m of media ##st ##in ##um secondary and un ##sp ##ec ##ified
ma ##li ##gnant neo ##p ##las ##m : ing ##uin ##al and lower limb l ##ymph nodes
anxiety disorder , un ##sp ##ec ##ified dependence on re ##nal dial ##ys ##is injection or
in ##fusion of other therapeutic or prop ##hyl ##actic substance diagnostic ultra ##sound of
peripheral vascular system trans ##fusion of packed cells hem ##od ##ial ##ys ##is injection
of anti ##coa ##gu ##lant injection or in ##fusion of cancer ch ##em ##oth ##era ##pe
##uti ##c substance injection of s ##tero ##id educational therapy

Figure 8.6. Mean attention of cancer patient using finetuned BioClinical BERT.

Figure 8.7 shows the average precision-curves for the three classification models (XGB and LR were trained with the one-hot representation); as previously discussed, Logistic regression has the worst performance, and it seems that Finetuned BERT and XGB have similar behavior. This indicates that the linear assumption taken by Logistic Regression is not enough to model the data.

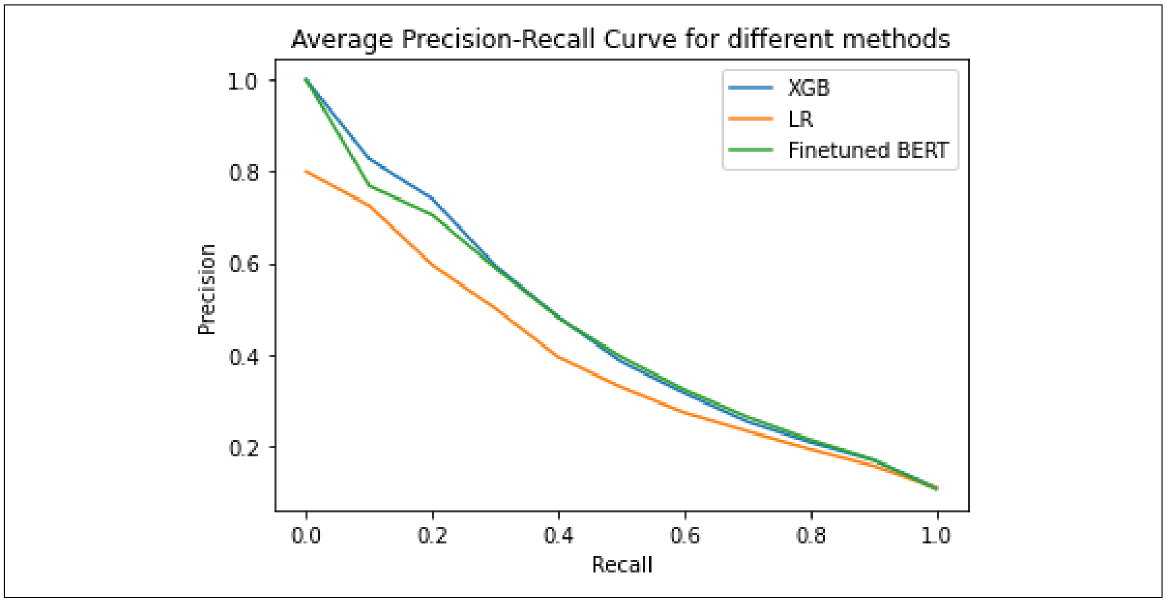


Figure 8.7. Average Precision-Recall curves for different classifiers. XGB and LR were trained using the one-hot encoding representation (with all the data), and BERT was finetuned for the readmission task.

8.4. Ontology encodings - Code level

Three different ontology-based encodings were evaluated at the code level: Node2Vec (with ICD ontology), PBG (with ICD ontology), and PBG (with the whole dataset represented as a graph).

In regards to the interpretability of the encodings, Node2Vec is a sure winner. This is probably because of how the algorithm works, performing random walks that can capture the ontology's structure better, versus PBG that works by learning a representation based on negative examples. This can be visually evaluated using the t-distributed stochastic neighbor embedding (t-SNE) algorithm, a non-linear dimensionality reduction technique used to embed high-dimensional vectors in a two-dimensional space (for plotting). Figure 8.8 shows the 2D projections of the Node2Vec embeddings for the diagnosis leaf codes in two specificity levels, while Figure 8.9 does the same for procedures. It is easy to spot several clusters; although separation is not perfect in the higher level (root chapters), it is very representative one level below (subchapter). This contrasts with Figure 8.10 which has the same plot for the diagnoses codes using PBG. At the higher level, all categories are mixed, and at a deeper level, the separation is not as good as with the Node2Vec representation. This also holds for the representation learned from the dataset represented as a graph and for the procedures embeddings.

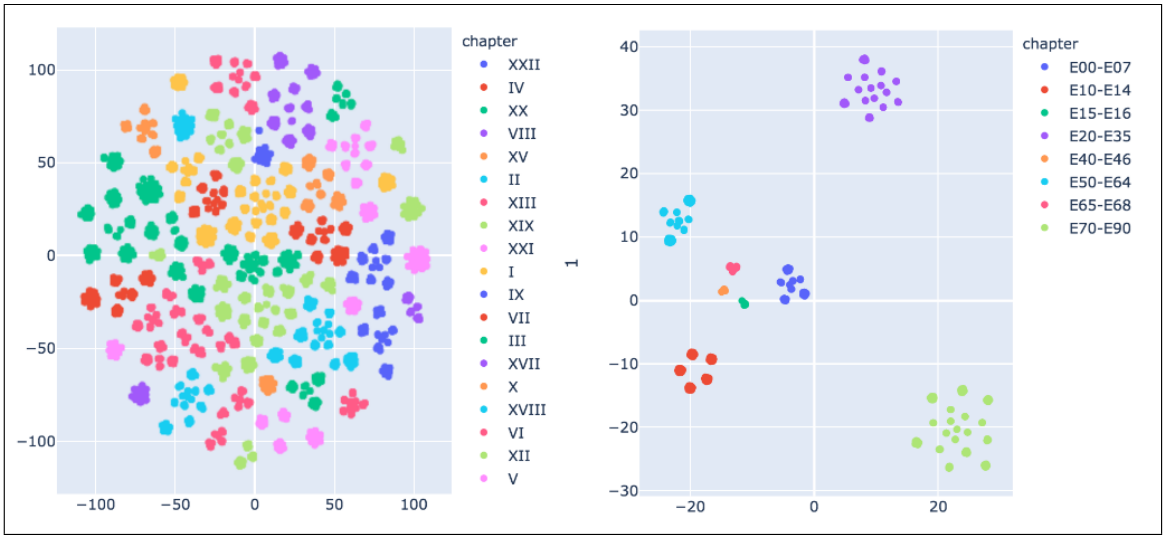


Figure 8.8. Node2Vec. Left Figure shows the t-SNE projections for leaf nodes categorized according to their root chapter. Right Figure shows leaf codes from chapter IV (Endocrine, nutritional and metabolic diseases) categorized according to their subchapters.

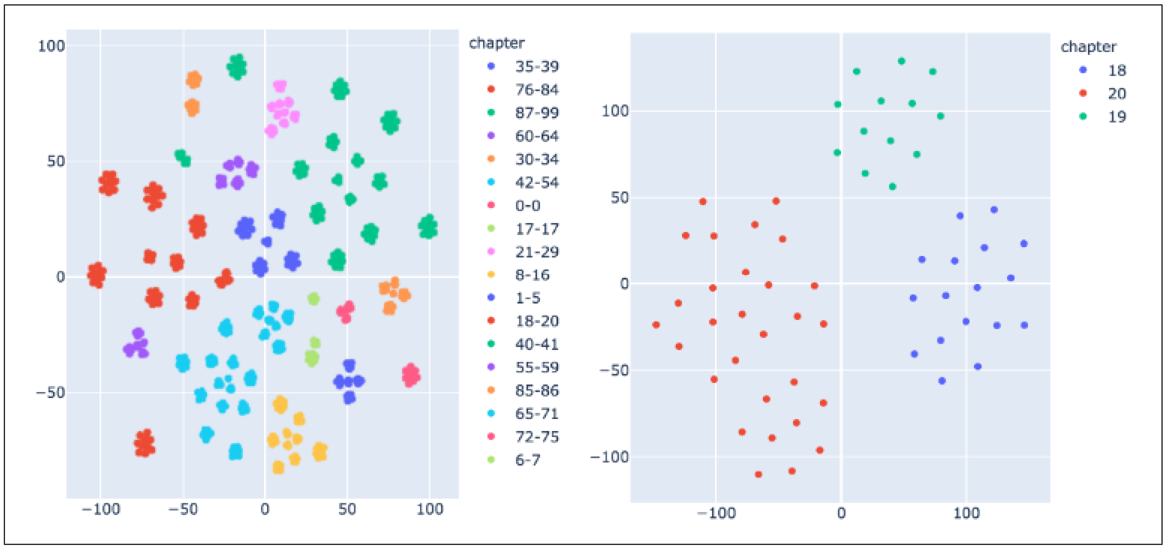


Figure 8.9. Node2Vec. Left Figure shows the t-SNE projections for leaf nodes categorized according to their root chapter. Right Figure shows leaf codes from chapter 4 (Operations on the Ear) categorized according to their subchapters.

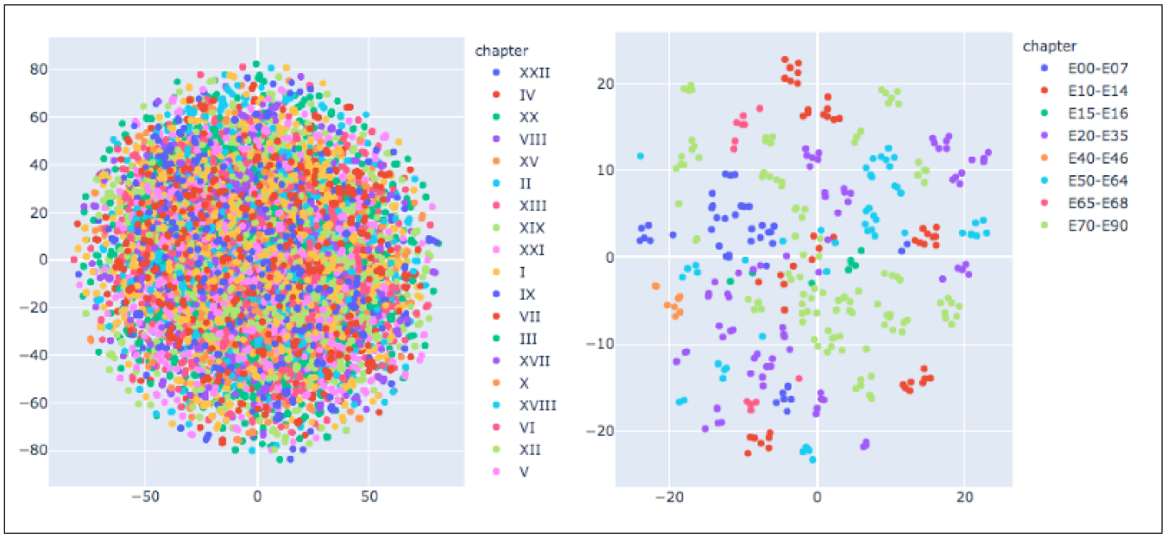


Figure 8.10. PBG. Left Figure shows the t-SNE projections for leaf nodes categorized according to their root chapter. Right Figure shows leaf codes from chapter 4 (Operations on the Ear) categorized according to their subchapters.

Regarding classification results, Table 8.5 has the maximum performances for each experiment, for both embedding sizes. There is no clear pattern for embedding size, so this is probably a parameter that would need to be finetuned depending on the task at hand. This is encouraging, and it aligns with the results of (Karimi et al., 2017), having low dimensional representations reduces computational and memory consumption.

Results are slightly worst than the baseline, but considering that a lot of specific information about the codes is lost when pooling, this was expected. Weighted mean pooling achieved the best results across the three experiments. This shows that including the sequentiality of the codes in the representation is an interesting step forward. Further research should be done to find better poolings since the ones we benchmarked are the simplest available. Also, as expected, concatenation of the diagnosis and procedure vector (instead of averaging them) performed better in general; when we average, we lose information.

Table 8.5. Best results for ontology encoding experiments. These algorithms were applied to generate encodings for every ICD code, and then these were pooled and classification was performed. Metrics are reported for two embedding sizes, size 30 and size 768.

Algorithm	Graph	Classifier	AUC (30)	AUC (768)	rp80 (30)	rp80(768)
Node2Vec	ICD	LR	0.74	0.77	0.02	0.06
PBG	ICD	LR	0.72	0.79	0.05	0.05
PBG	Hospital	LR	0.77	**0.8**	0.04	0.08
Node2Vec	ICD	XGB	0.79	0.79	0.07	0.09
PBG	ICD	XGB	0.77	0.78	0.08	**0.1**
PBG	Hospital	XGB	**0.8**	0.79	**0.1**	0.08

PBG trained on the ICD ontology underperformed in all data combinations (ignoring a few outliers) versus Node2Vec in the same ontology. As discussed in Section 6, PBG was not designed with small graphs in mind, and this shows in the results. On the other hand, PBG trained on the hospital dataset (with over 1M edges) performed better than both experiments trained exclusively on the ICD ontology. Mixing both the global information (ICD ontology) and the local context (hospital dataset) in the graph proved to be an interesting idea. This is a new representation that has not been discussed in the literature to the best of our knowledge.

8.5. Ontology encodings - Episode level

Using PBG, episode encodings were generated from their graph representation (including the ICD ontology). The graph did not have any readmission information. Specific data about the codes is lost in this representation since the input to the classifier is the encoding of the episode. This is also a novel representation.

Figure 8.11 shows the t-SNE projections of all episodes. There are some small clusters, but in general, there is not a clear separation. This is not surprising since, without additional supervision, the model can't know by which criteria to separate the vectors.

In this experiment, the highest recall at precision 80% was 0.03 (unbalanced, size 30, with demographic information). This is the worst result across all experiments.

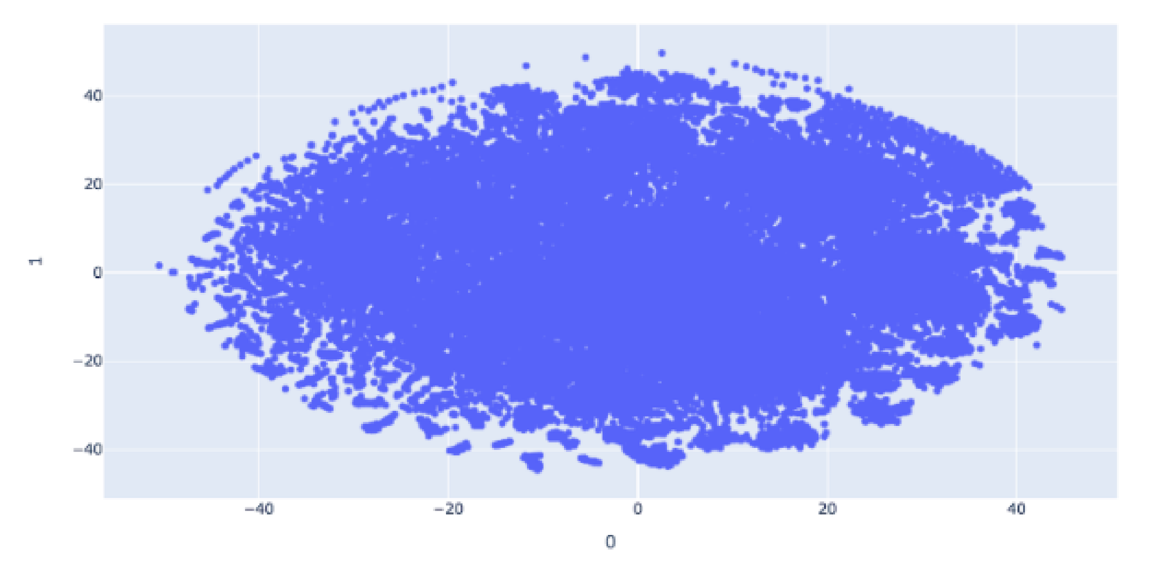


Figure 8.11. t-SNE projections for PBG embeddings of all episodes in the dataset.

9. CONCLUSIONS

We explored several ICD code representations and their variants: (1) One-hot encoding, which treats the codes as categorical variables and does not encode other known information about their medical context, (2) an original representation based on the text description of the codes, which leverages NLP pre-training efforts on medical corpora and (3) an ontology-based representation, and evaluated them in a 30-day hospital readmission prediction task.

We achieved state of the art results (AUC 0.82), but we also show that this is not enough to solve the problem. High AUC does not necessarily mean that the model is looking at the correct medical variables or that it will perform well for the specific task, which is detecting as many readmission cases as possible before they occur. The recall was low, which shows that it is essential to include other evaluation metrics such as precision-recall curves and recall at fixed precision and, also, that there is much improvement to be made to solve the readmission problem. We succeed at proving that it is possible to achieve good results with a simpler dataset, based on ICD codes.

Despite the accomplishment of our objectives, it is important to mention the limitations of this work. First of all, the dataset was highly preprocessed due to several problems. We have no information on the reliability of the ICD codes (some of them were likely the result of recording errors or misdiagnoses). Also, several factors are involved in hospital readmissions, and many are unpredictable. The dataset used in this work is not publicly available due to privacy concerns, limiting the reproducibility of our work.

It is clear that some classes of admissions are harder to classify; therefore, a cohort-specific approach should be taken. As future work, we propose doing a similar analysis separated by cohort, similar to (Futoma et al., 2015). Also, there is room to explore other solutions to solve the imbalance of the dataset (such as focal loss) since our experiments showed that the unbalanced dataset performed better. Finally, it would be interesting to explore a hybrid approach between these representations, similar to (Choi et al., 2017).

The results of our work are publicly available. We hope that researchers of other tasks that use ICD codes as input can benefit from them, as well as Chilean researchers who wish to work with this type of dataset.

REFERENCES

Alsentzer, E., Murphy, J., Boag, W., Weng, W.-H., Jindi, D., Naumann, T., & McDermott, M. (2019, June). Publicly available clinical BERT embeddings. In *Proceedings of the 2nd clinical natural language processing workshop* (pp. 72–78). Minneapolis, Minnesota, USA: Association for Computational Linguistics. Retrieved from `https://www.aclweb.org/anthology/W19-1909` doi: 10.18653/v1/W19-1909

A Miranda-Escalada, J. A.-E. M. K., A Gonzalez-Agirre. (2020). Overview of automatic clinical coding: annotations, guidelines, and solutions for non-english clinical cases at codiesp track of clef ehealth 2020. *Working Notes of Conference and Labs of the Evaluation (CLEF) Forum. CEUR Workshop Proceedings*, *2696*(263).

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., & Elhadad, N. (2015). Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining* (p. 1721–1730). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/2783258.2788613` doi: 10.1145/2783258.2788613

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (p. 785–794). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/2939672.2939785` doi: 10.1145/2939672.2939785

Ching, T., Himmelstein, D. S., Beaulieu-Jones, B. K., Kalinin, A. A., Do, B. T., Way,

G. P., ... Greene, C. S. (2018). Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, *15*(141), 20170387. Retrieved from https://royalsocietypublishing.org/doi/abs/10.1098/rsif.2017.0387 doi: 10.1098/rsif.2017.0387

Choi, E., Bahadori, M. T., Song, L., Stewart, W., & Sun, J. (2017). Gram: Graph-based attention model for healthcare representation learning. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/N19-1423 doi: 10.18653/v1/N19-1423

Fierro, C., Pérez, J., & Mora, J. (2020). *Predicting unplanned readmissions with highly unstructured data.*

Futoma, J., Morris, J., & Lucas, J. (2015). A comparison of models for predicting early hospital readmissions. *Journal of Biomedical Informatics*, *56*, 229 - 238. Retrieved from http://www.sciencedirect.com/science/article/pii/S1532046415000969 doi: https://doi.org/10.1016/j.jbi.2015.05.016

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. The MIT Press.

Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (p. 855–864). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2939672.2939754 doi: 10.1145/2939672.2939754

Gutiérrez, L., & Keith Norambuena, B. (2019, 01). A systematic literature review on word embeddings: Proceedings of the 7th international conference on software process improvement (cimps 2018). In (p. 132-141). doi: 10.1007/978-3-030-01171-0_12

Hema, N., & Justus, S. (2015). Conceptual graph representation framework for icd-10. *Procedia Computer Science*, *50*, 635 - 642. Retrieved from http://www.sciencedirect.com/science/article/pii/S1877050915005980 (Big Data, Cloud and Computing Challenges) doi: https://doi.org/10.1016/j.procs.2015.04.097

Huang, K., Altosaar, J., & Ranganath, R. (2019). Clinicalbert: Modeling clinical notes and predicting hospital readmission. *CoRR*, *abs/1904.05342*. Retrieved from http://arxiv.org/abs/1904.05342

Karimi, S., Dai, X., Hassanzadeh, H., & Nguyen, A. (2017, August). Automatic diagnosis coding of radiology reports: A comparison of deep learning and conventional classification methods. In *BioNLP 2017* (pp. 328–332). Vancouver, Canada,: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/W17-2342 doi: 10.18653/v1/W17-2342

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2019, Sep). Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*. Retrieved from http://dx.doi.org/10.1093/bioinformatics/btz682 doi: 10.1093/bioinformatics/btz682

Lerer, A., Wu, L., Shen, J., Lacroix, T., Wehrstedt, L., Bose, A., & Peysakhovich, A. (2019). PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of the 2nd sysml conference.* Palo Alto, CA, USA.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In I. Guyon et al. (Eds.), *Advances in neural information processing systems 30* (pp. 4765–4774). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval.* Cambridge, UK: Cambridge University Press. Retrieved from http://nlp.stanford.edu/IR-book/information-retrieval-book.html

Mihailovic, N., Kocic, S., & Jakovljevic, M. (2016). Review of diagnosis-related group-based financing of hospital care. *Health services research and managerial epidemiology*, *3*, 2333392816647892.

Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013, 01). Efficient estimation of word representations in vector space. In (p. 1-12).

Miotto, R., Li, L., & Kidd, B. (2016, 05). Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, *6*, 26094. doi: 10.1038/srep26094

MPAC. (2007). *June 2007 report to the congress: Promoting greater efficiency in medicare.* http://www.medpac.gov/docs/default-source/reports/Jun07_EntireReport.pdf. ((Accessed on 12/19/2020))

Nguyen, D., Luo, W., Venkatesh, S., & Phung, D. (2018). Effective identification of similar patients through sequential matching over icd code embedding. *Journal of Medical Systems*, *42*(5), 1–13.

Nguyen, P., Tran, T., Wickramasinghe, N., & Venkatesh, S. (2017). Deepr: A convolutional net for medical records. *IEEE Journal of Biomedical and Health Informatics*, *21*(1), 22-30.

Nori, H., Jenkins, S., Koch, P., & Caruana, R. (2019). Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*.

Parra, D., Valdivieso, H., Carvallo, A., Rada, G., Verbert, K., & Schreck, T. (2019, 8). Analyzing the design space for visualizing neural attention in text classification. In *Proc. ieee vis workshop on vis x ai: 2nd workshop on visualization for ai explainability (visxai).*

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pham, T., Tran, T., Phung, D. Q., & Venkatesh, S. (2016). Deepcare: A deep dynamic memory model for predictive medicine. In *Pakdd.*

Rajkomar, A., Oren, E., Chen, K., Dai, A., Hajaj, N., Liu, P., … Dean, J. (2018, 01). Scalable and accurate deep learning for electronic health records. *npj Digital Medicine*, *1*. doi: 10.1038/s41746-018-0029-1

Theodoridis, S. (2020). Chapter 3 - learning in parametric modeling: Basic concepts and directions. In S. Theodoridis (Ed.), *Machine learning (second edition)* (Second Edition ed., p. 67 - 120). Academic Press. Retrieved from `http://www.sciencedirect.com/science/article/pii/B978012818803300012X` doi: https://doi.org/10.1016/B978-0-12-818803-3.00012-X

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st international conference on neural information processing systems* (p. 6000–6010). Red Hook, NY, USA: Curran Associates Inc.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., … Rush, A. M. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, *abs/1910.03771*.

Zapata, M. (2018). Importancia del sistema grd para alcanzar la eficiencia hospitalaria. *Revista Médica Clínica Las Condes*, *29*(3), 347 - 352. Retrieved from http://www.sciencedirect.com/science/article/pii/S0716864018300592 (Tema central: Enfermería) doi: https://doi.org/10.1016/j.rmclc.2018.04.010

APPENDIX

A. DATASET COMPARISON (READMISSION PROBLEM)

Table A.1. Comparison of datasets used in literature to solve the Readmission Problem.

Research	Dataset size	Health data
Caruana, 2015	196K patients (train set) over 2 years, 4K features for each patient	Lab test results, summaries of doctor notes, details of previous hospitalizations
Futoma, 2015	3.3M hospital admissions over 6 years	Demographic and background data, ICD-10-AM codes (procedures and diagnoses), DRG code
Nguyen, 2017	300K patients and 600K admissions over 5 years	ICD codes (procedures and diagnoses)
Pham, 2017	53K admissions (diabetes cohort) over 12 years	ICD codes (procedures, diagnoses, and medications)
Rajkomar, 2018	100K patients and 200K admissions	Demographic data, provider orders, diagnoses, procedures, medications, Lab test results, vital signs, flowsheet data, free-text medical notes
Huang, 2019	60K admissions (MIMIC III)	Clinical notes
Fierro, 2020	186K admissions over 9 years	Demographic data, reason of visit, procedures, diagnoses, medications, clinical notes

B. PREPROCESSING PIPELINE DETAIL

B.1. Preparation

The original dataset came in two files, a .txt file for years 2014-2016 and a .xlsx file for years 2017-2018. In this step, both files are merged into a single file, and then the whole dataset is read, and all Spanish specific characters are replaced (i.e., é goes to e, Ú goes to u, ñ goes to n) to minimize further encoding issues.

The result of this stage is a single CSV file containing all the information provided by the hospital.

B.2. Data extraction

The Data extraction stage consisted of four other sub-stages: demographic data extraction, readmission tagging, codes extraction, and descriptions extraction.

B.2.1. Demographic data extraction

This stage's objective is to extract the columns that are not related to readmission, diagnosis, or procedure but might still be useful for predicting readmission. The first step is ensuring that every column is parsed to the correct data type, see B.1.

Table B.1. Demographic features extracted from the dataset.

Feature	dtype
Subject Id	str
Episode	str
Admission Date	datetime
Discharge Date	datetime
Sex	str
Age	int
Length of stay	int
DRG Weight	float

Afterward, all rows are processed and indexed by episode, with a total of 129816 episodes. Every episode is then reviewed, and episodes with conflicting information (more than one value for sex, age, or subject id) are deleted; 63 episodes are deleted in this stage resulting in 129753 episodes.

When there are multiple values in the other features per episodes, values are picked as follows:

- Admission date: Minimum Admission date
- Discharge date: Maximum Discharge date
- DRG Weight: Mean weight

Finally, the length of stay is calculated as the difference between the discharge date and admission date.

The maximum admission date in the dataset is 31st of December of 2018; this value is stored to set the date limit of data that will be considered. Since we are interested in 30-day readmission, data points with admissions after the 1st of December of 2018 (inclusive) will not be considered.

B.2.2. Readmission tagging

The original dataset comes with a readmission flag added according to the hospital's definition. Since in the previous step we fixed inconsistencies in the dataset (multiple entries for one episode), minimizing and maximizing the admission and discharge dates, respectively, the readmission tag needs to be slightly adjusted.

The algorithm used to tag readmission is:

(i) Sort rows by subject id as first criteria and admit date as second criteria

(ii) Iterate from the second row onwards

(iii) For every row, if the previous row has the same subject as the current row, there is possible readmission. The difference between the previous row discharge date and the current row admission date is calculated. If this difference is less than 30 days and considered readmission in the original dataset, then the previous row is tagged as a readmission source.

(iv) If the admission date is after or equal to 1st of December of 2018, the row is ignored

B.2.3. Codes extraction

The original dataset has a hard limit of 30 diagnoses and 30 procedures per row; coded columns are immediately followed by text description in a weaved format (i.e., Diagnosis code 1, Description Diagnosis 1, Diagnosis code 2, Description Diagnosis 2, ..., Procedure code 1, Procedure Description 1, Procedure code 2, Procedure Description 2). When a row has less than 30 procedures or diagnoses, then unused columns are filled with NaN.

Additionally, this type of dataset is sequential (i.e., the first diagnosis is more relevant than the second diagnosis); therefore, it is essential to maintain this sequence when extracting the codes. The data also includes the number of diagnoses and procedures per row, but as we know from the other stages, there might be more than one row per episode, so this number needs to be recalculated. It is important to note that not all values are filled, sometimes a pair has code and not description, and the other way around. Also, there is no guarantee that a code will have the same description across the dataset.

The output of this stage is a key-value data structure, indexed by episode. We iterate over all rows storing for every episode an array of arrays of codes (for both procedures and diagnosis) where each array represents the codes in a row. In this stage, we are focused exclusively on the codes and not the descriptions.

Our objective is to have a single array of codes for both diagnoses and procedures per episode instead of an array of arrays. The challenge is that we need to maintain the sequentiality of codes when there are multiple rows (there are up to ten rows per episode). To solve these challenges, a multiple queue merging algorithm was implemented and executed in every episode:

(i) Convert the array of arrays to a double-ended queue of double-ended queues (or to normal queues, depending on the implementation)

(ii) Initialize an empty set **seen** and an empty array **codes**

(iii) While the main queue is not empty: pop the leftmost element in the main queue, pop the leftmost code inside that secondary queue. If the code is not in **seen**, add to **codes**. If the secondary queue still has elements, add it to the end of the main queue.

B.2.4. Description extractions

We do another pass through the dataset to extract all code/description pairs within the dataset (for both diagnoses and procedures), including NaN values, to further analyze codes down the pipeline.

B.3. Code analysis

It is crucial for further steps to have a 1:1 mapping between codes and descriptions for both procedures and diagnoses. Also, to represent the codes using NLP, it would be ideal to have full descriptions in English for each code (the leaves in the ICD graph) and each intermediate node (chapter names). To get this mapping, intense manual engineering was required. After analyzing the code-descriptions pairs in the database, serious discrepancies were found. Here we aim to describe these discrepancies and the possible reasons for them, also the manual solutions used to extract as much information as possible from the dataset. We will explain the process that generates the final ground truth of the code-description mapping in English and Spanish and the final processing step that ends with a set of clean, truthful codes that are correctly mapped.

The final mappings were published in a Github repository[1]. This repository is a key contribution of this thesis. It provides mappings for both procedures and diagnoses with the current standards used in the Chilean healthcare systems in Spanish and English. This considerably facilitates further researchers that want to work with this data.

B.3.1. Procedure codes

The Chilean standard for procedure codes is the CIE-9-CM. Codes in this standard are floats with a root value of two numbers; specificity is given after the decimal point with up to two new numbers.

[1] https://github.com/tamycova/icd-cie-codes-chile

When analyzing the code-description pairs, we found 22 malformed codes (started with a letter instead of a number, similar to ICD-10 diagnosis codes). All of them appeared only in episodes during 2014-2016, so probably they were codes that could not be mapped appropriately to ICD-9-CM by the older electronic system.

From the universe of the other 2804 existing codes, two don't have an associated description, 1998 have a unique description, and 804 have more than one description. These 804 codes were manually checked, 598 were just slight modifications in the description, and 206 had a mix of unrelated descriptions.

In this step, a severe problem with the dataset was noticed. The original files had parsed the codes as floats, therefore, losing the leading and trailing zeros in codes. This is a problem because it generates code/descriptions collisions, and also, the codes in the dataset would not perfectly match a standardized mapping.

Further analysis was needed to decide what to do with the conflicting codes, and for this, it is necessary to have a ground truth of descriptions to compare with.

For Spanish, the mapping is available online[2], but it is not available in its raw version. Initially, the data was obtained from the official publication of the standard's translation[3]. The PDF file was parsed to HTML using an online tool and then scraped to find all the code-description lines.

After contacting the Spanish Ministry of Health, we were able to obtain an official file with the raw codes, and this one was used in the end as ground truth to guarantee correctness.

The file has all names for chapters and leaf nodes, and it corresponds to the CIE-9-MC version 2014.

[2] https://eciemaps.mscbs.gob.es/ecieMaps/browser/index_9_mc.html

[3] https://www.mscbs.gob.es/estadEstudios/estadisticas/docs/CIE9MC_2014_def_accesible.pdf

For English, the CMS (Centers for Medicare & Medicaid Services) has different versions of the ICD-9-CM Diagnosis and Procedure Codes (in abbreviated and full form) available for downloading online[4]. Unfortunately, the available codes are only the leaf nodes in the ICD graph (only four-digit codes), so we are still missing information about intermediate nodes' description.

The CDC (Centers for Disease Control and Prevention) has a rich text format (RTF) version of the 2011 edition (which corresponds to the CIE-9-MC of 2014 according to the Spanish Ministry of health) available online[5]. The file was converted to PDF format to work with this data, then to HTML format and parsed using the Beautiful Soup Python Library to finally have a TSV file with all the codes, both leaf and intermediate nodes. This is the version used from now on.

Both files (Spanish and English) were crossed into a single JSON file, where each code is the index and the value is a mapping of the English and Spanish version of the description of the code. Four codes lacked English translation, and these were manually added.

Finally, we determine the codes that are **invalid**. These codes do not have the ICD-9-CM structure (affects 140 rows) or a description (affects two rows).

Then, since the database came wrongly parsed, we need to modify every code at best convenience to match the original codes. The modification applied to each code depends on their configuration (i.e., configuration 2,2 means that the code has two elements before the decimal point and two elements after the decimal point, configuration 1,0 means that the code has one element before the decimal point and no element after the decimal point), being careful to only add specificity with a trailing zero when needed, as we can see in Figure B.2.

[4] https://www.cms.gov/Medicare/Coding/ICD9ProviderDiagnosticCodes/codes

[5] https://www.cdc.gov/nchs/icd/icd9cm.htm

Table B.2. Modification rules for Procedure codes.

Configuration	Modification
1,0	Add a leading 0, if the new code does not exist, then we add a trailing zero
1,1	Add a leading zero
1,2	Add a leading zero
2,0	Add trailing zero
2,1	If the code exists, leave as it is, if not, add a trailing zero
2,2	Leave as it is

Even though this method is not perfect, it is the best possible fix for the dataset, since if any trailing zero is lost, we only lose specificity but we still have the node's parent information.

Translations for all codes were manually inspected to guarantee the quality of the modifications to the original code and the Spanish/English mappings.

B.3.2. Diagnosis codes analysis

The standard used for diagnosis codes in Chile is ICD-10, published by the World Health Organization (WHO).

Like the procedure codes, some codes do not have the proper format of an ICD-10 diagnosis code (start with an alphabetical character). There were 4252 malformed codes, all of them used between 2014 and 2016, so this is probably due to the new coder's unmappable codes. From the 8196 valid codes, 196 do not have a description, 7806 have a unique description in the database, and 194 have a mix of descriptions. Of those, 194, 85 have similar descriptions, and 109 have a combination of unrelated descriptions.

Further analysis was needed because of the mixed descriptions, and also because within the codes that started with alphanumerical characters, there were codes like ”D49” (and ICD-10 only goes to D48), or ”E849.0” (when ICD-10 has only three characters before the decimal point). For this, a ground truth is needed (a set of codes that will be declared valid, with an associated description in Spanish and English).

For Spanish, the standard corresponds to the CIE 2009 8th edition, a translation made by PAHO (Panamerican Health Organization) and was extracted from the CIE documentation[6]. Some preprocessing was needed, removing the special Spanish characters and finally outputting a TSV file. A drawback of the data used is that particular specifications are described at the most specific level (for example, E10.0 is ”Insulin-dependent diabetes mellitus with coma” and in the Spanish dataset appears as ”con coma”), so special care is needed to use the Spanish descriptions in some instances (concatenating with the ancestor information).

For English, the first attempt was to scrape the HTML file of every chapter in WHO's website[7], and then extract the codes and descriptions. When comparing this result to the Spanish codes, it was noted that it misses some codes that are only present in the index and not in the main HTMLs, and other codes with special specifications (the same ones that do not have a full description in Spanish). In a second attempt, the WHO's JSON API was queried using the BFS algorithm. This was convenient since it also allowed to store the graph structure of ICD-10. This set of codes has more codes than the first attempt, but the codes with the specificity problem were still not present. There were 23399 missing codes. Since this number is quite large, to solve this problem, the most direct approach was to complete the set of codes using machine translation from the Spanish version, being careful to concatenate with the node's parent description to have the full description, but only the codes that we would need, that is, the codes that are used in the original dataset.

[6] https://eciemaps.mscbs.gob.es/ecieMaps/browser/index_10_2008.html

[7] https://icd.who.int/browse10/2008/en/

After checking the original dataset and inspecting the codes that were in the Spanish set but not in the English set, it was noted that there was a limited amount of suffixes that needed translation, both for wound location, physical location, activity, and status (i.e., forearm, house, while working, open). These were collected and translated using Google Translate; then, the English set was completed as Figure B.3 indicates.

Table B.3. Modification rules for Diagnosis codes.

Configuration	Modification
YXX.X	Concatenate Spanish translation of YXX.X with description of YXX, YXX is no longer leaf and now YXX.X is added to the graph as leaf.
YXX.XX when YXX.X in set	Concatenate Spanish translation of YXX.XX with description of YXX.X, YXX.X is no longer leaf and now YXX.XX is added to the graph as leaf.
YXX.XX when YXX.X not in set	Create two new nodes, YXX.XX and YXX.X, YXX is no longer leaf and YXX.XX is leaf. Translation of YXX.X is concatenated with description of YXX and assigned to YXX.X, then, translation of YXX.XX is concatenated with the new description of YXX.X and assigned to YXX.XX

The result of this stage is a JSON file with the ICD codes and their descriptions in Spanish and English. With the JSON file as the ground truth of valid codes, it is possible to identify the codes that will be considered **invalid** in the dataset.

The codes that do not have the ICD code format are invalid. The codes that did not have descriptions were checked. If the codes exist in the valid set of codes, then they are considered valid.

7494 of the codes considered in a suitable format and with a single description exist in the ground truth file, but 397 do not and are considered invalid. At this point, we manually compare the original descriptions of the dataset and the ground truths in Spanish and English to make sure translations and the standard used are correct.

The 109 codes that had a mixed description are manually compared to the ground truth. Only 92 are part of the ground truth file, and of those, only 14 have valid descriptions; the other 78 are considered invalid since their use along the original dataset is not consistent.

In summary, invalid codes are the codes that do not have the ICD-10 format, the ones that do not exist in the universe of codes that are considered ground truth, and the ones that are not consistently used in the original database (since they have different descriptions they might be typing errors). There are 4816 such codes.

B.4. Integration

The final dataset is created by integrating all the outputs of previous stages. For every episode, we have the demographic data, the readmission tag calculated in and a sequential list of codes for both diagnoses and procedures. We do an inner join on episode in the three files and only keep episodes with the information from the three stages.

Finally, we iterate over all episodes and calculate the portion of procedure and diagnoses codes that is valid (valid codes are those that are not in the invalid sets created in the codes analysis stage), if both procedures and diagnoses have over 0.75 percent of valid codes, then the episode is kept in the final dataset. 12134 episodes that had damaged rows were preserved, while 34357 were lost.

C. FEATURE ANALYSIS

Table C.1. Numeric Features Analysis.

	Age (Y)	Length of Stay (LOS)	Number of Diagnoses (ND)	Number of Diagnoses (ND)	DRG Weight
mean	44.13	4.89	4.88	9.32	5.28
std	24.54	12.08	3.35	5.19	12.3
min	0	0	1	0	0
25%	27	1	2	6	0.53
50%	44	2	4	8	0.81
75%	64	5	6	12	2.78
max	106	1417	31	31	222.66

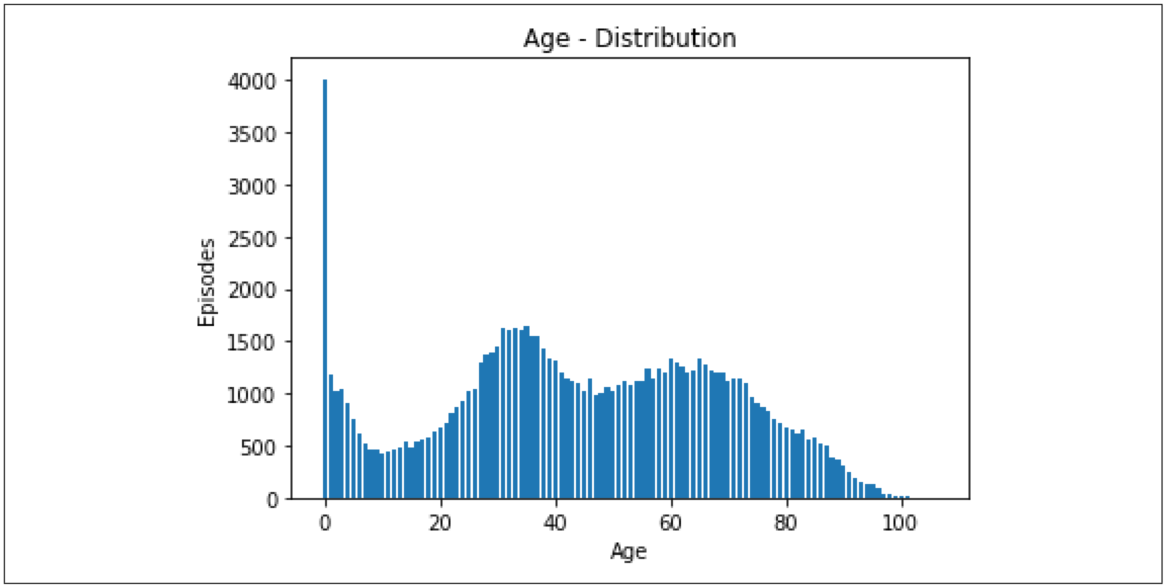


Figure C.1. Distribution of Age.

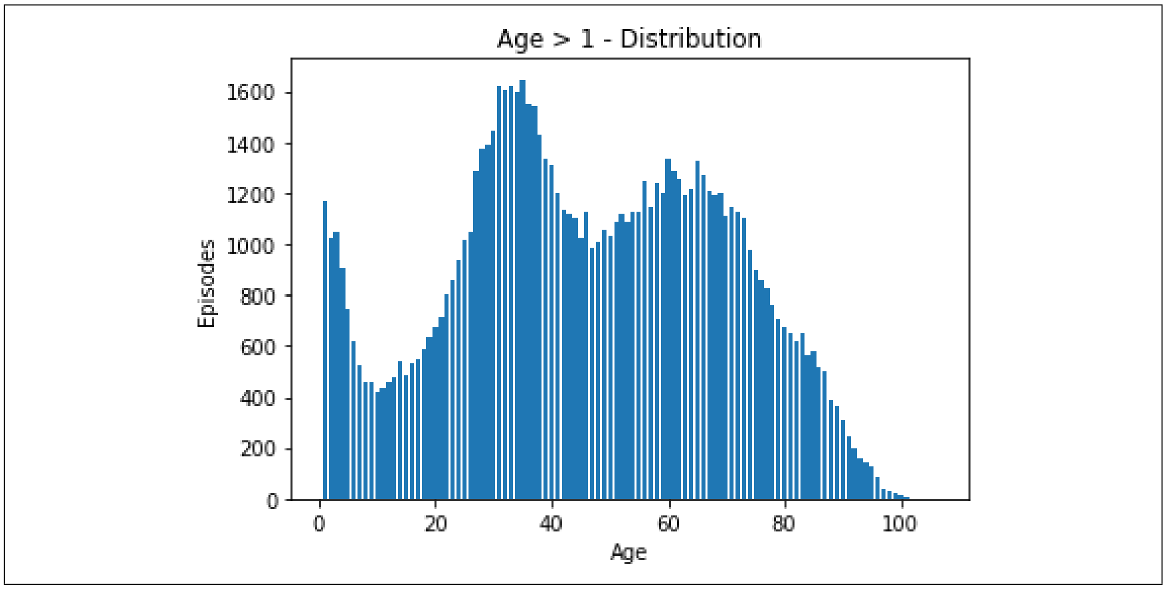


Figure C.2. Distribution of Age without age 0.

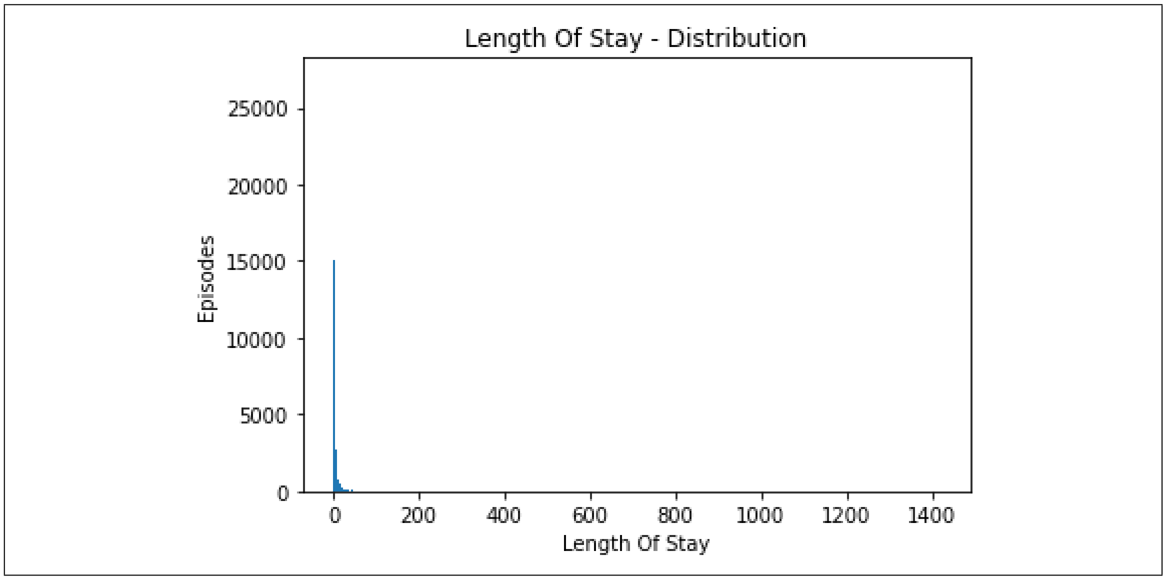


Figure C.3. Distribution of Length of Stay.

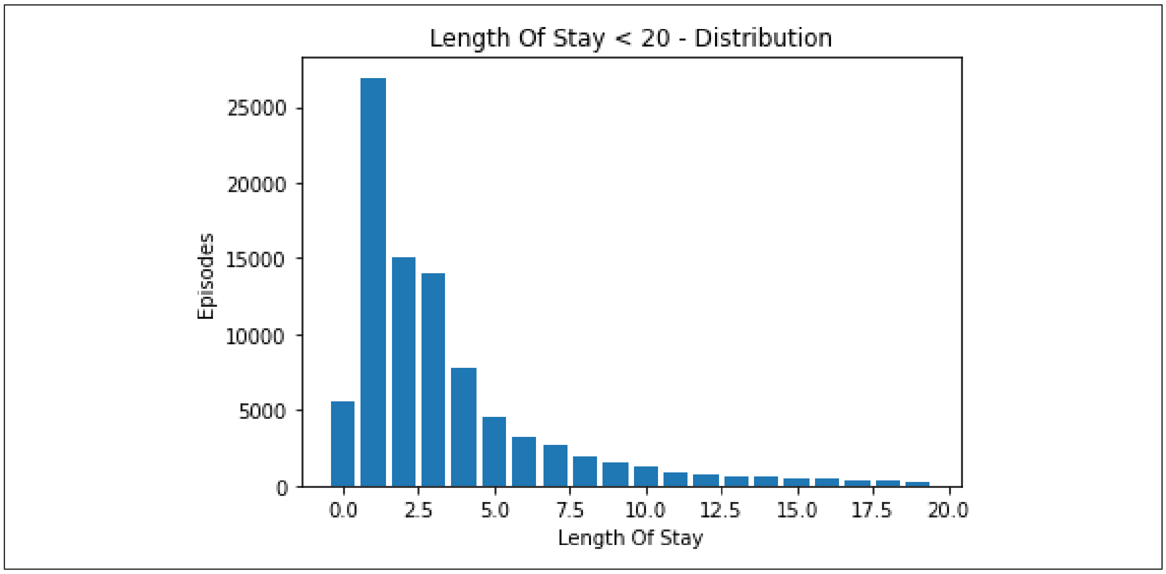


Figure C.4. Distribution of Length of Stay without outliers.

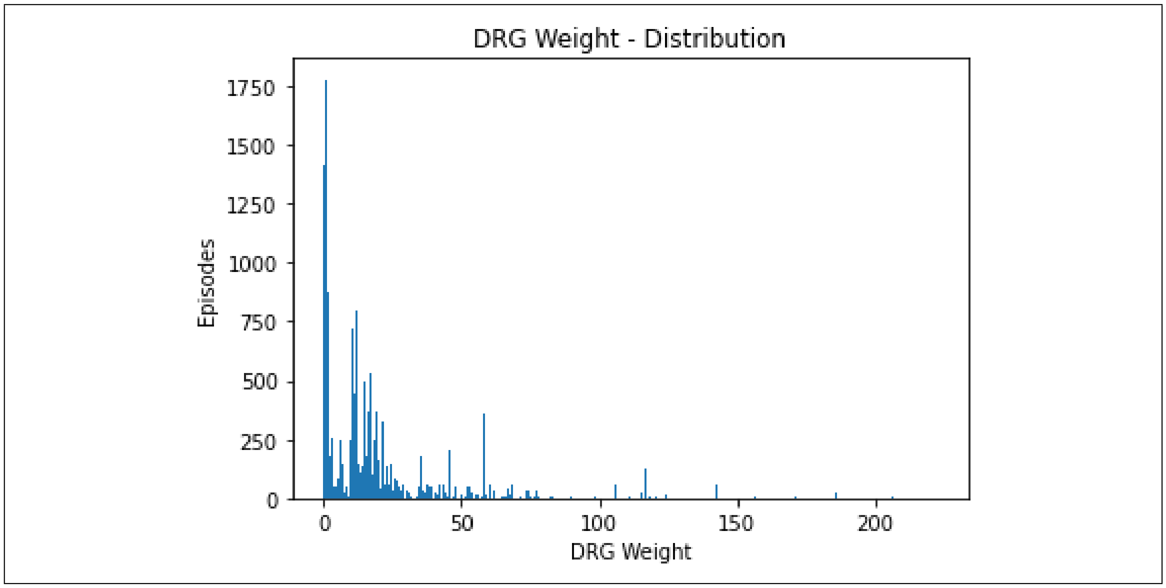


Figure C.5. Distribution of DRG weight.

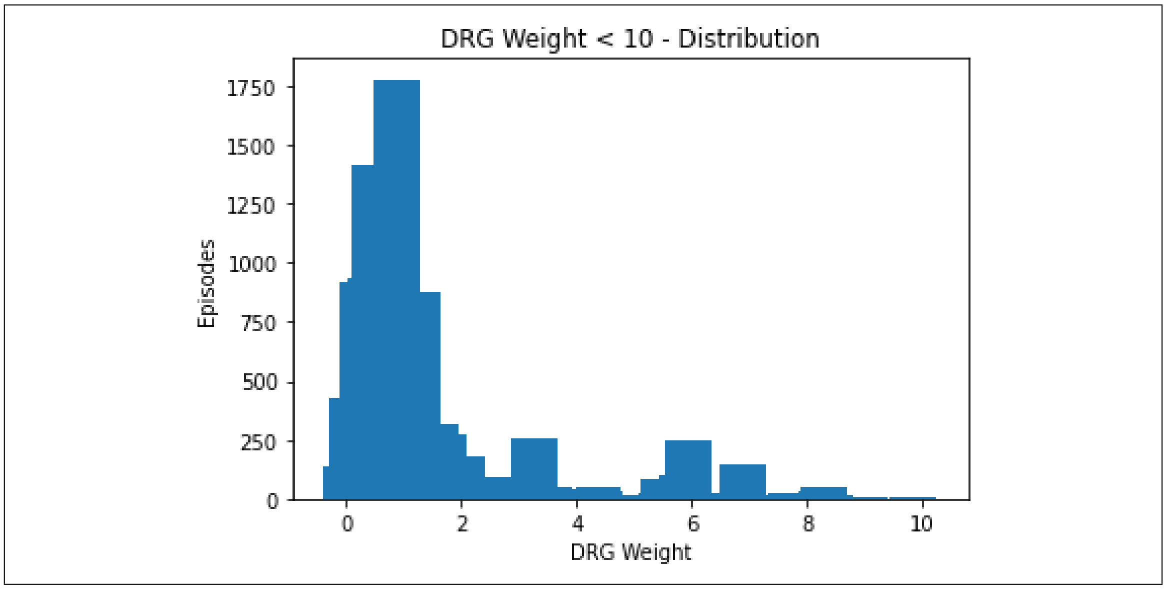


Figure C.6. Distribution of DRG weight with values less than 10.

D. DETAILED RESULTS

Table D.1. LR One-hot encoding results.

bal	demo	grd	codes	auc	rp60	rp80	acc
0	1	1	1	0.79	**0.2**	0.04	**0.9**
0	1	1	0	0.76	0.15	0.03	**0.9**
0	1	0	1	0.78	**0.2**	0.02	**0.9**
0	1	0	0	0.6	0.02	0.0	0.89
0	0	1	1	**0.8**	0.19	**0.07**	**0.9**
0	0	1	0	0.78	**0.2**	0.02	**0.9**
0	0	0	1	**0.8**	0.13	0.05	**0.9**
1	1	1	1	**0.8**	0.18	0.02	0.76
1	1	1	0	0.77	0.15	0.03	0.69
1	1	0	1	0.79	0.18	0.02	0.77
1	1	0	0	0.62	0.08	0.0	0.6
1	0	1	1	0.78	0.14	0.04	0.78
1	0	1	0	0.77	0.19	0.02	0.69
1	0	0	1	0.78	0.15	0.03	0.78

Table D.2. XGB One-hot encoding results.

bal	demo	drg	codes	auc	rp60	rp80	acc
1	1	1	1	**0.81**	**0.29**	**0.13**	0.78
1	1	1	0	0.78	0.2	0.09	0.71
1	1	0	1	**0.81**	**0.29**	0.09	0.79
1	1	0	0	0.76	0.17	0.07	0.74
1	0	1	1	0.8	0.27	0.09	0.8
1	0	1	0	0.77	0.19	0.06	0.6
1	0	0	1	0.8	0.27	0.09	0.81
0	1	1	1	**0.81**	**0.29**	0.12	**0.91**
0	1	1	0	0.79	0.22	0.09	0.9
0	1	0	1	**0.81**	**0.29**	**0.13**	**0.91**
0	1	0	0	0.77	0.18	0.07	0.9
0	0	1	1	**0.81**	0.28	0.11	**0.91**
0	0	1	0	0.77	0.19	0.03	0.9
0	0	0	1	0.8	0.27	0.11	**0.91**

Table D.3. LR Static BERT BioClinical results.

bal	demo	emb	auc	rp60	rp80	acc
1	1	both	0.76	0.08	0.02	0.7
1	1	both_int	0.77	0.08	0.02	0.69
1	1	conc	0.78	0.17	0.02	0.72
1	0	both	0.78	0.11	0.02	0.71
1	0	both_int	0.78	0.1	0.03	0.71
1	0	conc	**0.79**	0.15	0.03	0.73
0	1	both	0.76	0.12	0.02	**0.9**
0	1	both_int	0.76	0.1	0.03	**0.9**
0	1	conc	0.78	0.13	0.02	**0.9**
0	0	both	0.78	0.12	0.02	**0.9**
0	0	both_int	0.78	0.17	**0.04**	**0.9**
0	0	conc	**0.79**	**0.19**	0.03	**0.9**

Table D.4. XGB Static BERT BioClinical results.

bal	demo	emb	auc	rp60	rp80	acc
1	1	both	0.76	0.11	0.04	0.84
1	1	both_int	0.76	0.12	0.04	0.83
1	1	conc	0.77	**0.2**	0.05	0.84
1	0	both	0.75	0.11	0.03	0.83
1	0	both_int	0.74	0.1	0.04	0.83
1	0	conc	0.76	0.17	0.03	0.84
0	1	both	0.77	0.16	0.05	**0.9**
0	1	both_int	0.76	0.15	0.04	**0.9**
0	1	conc	**0.78**	0.19	**0.07**	**0.9**
0	0	both	0.75	0.14	0.03	**0.9**
0	0	both_int	0.75	0.11	0.03	**0.9**
0	0	conc	0.77	0.17	0.03	**0.9**

Table D.5. LR Static BERT Discharge results.

bal	demo	emb	auc	rp60	rp80	acc
1	1	both	0.77	0.11	0.02	0.71
1	1	both_int	0.77	0.11	0.02	0.71
1	1	conc	**0.79**	0.14	0.02	0.72
1	0	both	0.78	0.16	0.02	0.72
1	0	both_int	0.78	0.14	0.03	0.72
1	0	conc	**0.79**	0.17	0.03	0.73
0	1	both	0.77	0.17	0.04	**0.9**
0	1	both_int	0.77	0.16	0.02	**0.9**
0	1	conc	**0.79**	0.14	0.04	**0.9**
0	0	both	0.78	0.15	0.03	**0.9**
0	0	both_int	0.78	0.16	0.03	**0.9**
0	0	conc	**0.79**	**0.2**	**0.05**	**0.9**

Table D.6. XGB Static BERT Discharge results.

bal	demo	emb	auc	rp60	rp80	acc
1	1	both	0.76	0.17	0.05	0.83
1	1	both_int	0.76	0.13	0.04	0.83
1	1	conc	**0.78**	**0.2**	0.06	0.84
1	0	both	0.76	0.16	0.02	0.83
1	0	both_int	0.75	0.14	0.03	0.83
1	0	conc	0.77	0.18	0.04	0.84
0	1	both	0.77	0.18	0.06	**0.9**
0	1	both_int	0.77	0.18	0.04	**0.9**
0	1	conc	**0.78**	**0.2**	**0.07**	**0.9**
0	0	both	0.77	0.15	0.04	**0.9**
0	0	both_int	0.76	0.16	0.02	**0.9**
0	0	conc	0.77	0.18	0.04	**0.9**

Table D.7. LR Static BERT Base results.

bal	demo	emb	auc	rp60	rp80	acc
1	1	both	0.76	0.14	0.02	0.69
1	1	both_int	0.76	0.1	0.01	0.69
1	1	conc	0.78	0.12	0.04	0.71
1	0	both	0.78	0.12	0.02	0.72
1	0	both_int	0.78	0.13	0.01	0.71
1	0	conc	**0.79**	0.13	0.03	0.72
0	1	both	0.76	0.15	0.05	**0.9**
0	1	both_int	0.76	0.15	0.03	**0.9**
0	1	conc	0.78	0.17	**0.06**	**0.9**
0	0	both	0.78	0.18	0.04	**0.9**
0	0	both_int	0.78	0.14	0.02	**0.9**
0	0	conc	**0.79**	**0.19**	0.04	**0.9**

Table D.8. XGB Static BERT Base results.

bal	demo	emb	auc	rp60	rp80	acc
1	1	both	0.75	0.16	0.04	0.83
1	1	both_int	0.75	0.09	0.05	0.83
1	1	conc	**0.77**	**0.17**	0.04	0.83
1	0	both	0.74	0.15	0.04	0.83
1	0	both_int	0.74	0.09	0.02	0.83
1	0	conc	0.76	0.16	0.04	0.83
0	1	both	0.76	0.16	0.05	**0.9**
0	1	both_int	0.76	0.16	0.04	**0.9**
0	1	conc	**0.77**	**0.17**	**0.06**	**0.9**
0	0	both	0.75	0.13	0.03	**0.9**
0	0	both_int	0.75	0.12	0.03	**0.9**
0	0	conc	**0.77**	0.13	0.03	**0.9**

Table D.9. Dynamic Clinical BERT with balanced dataset results.

frozen layers/batch size	auc	rp60	rp80	acc
0/10	0.79	0.25	**0.08**	**0.8**
5/20	0.8	**0.28**	0.05	0.79
10/64	**0.81**	0.26	0.05	0.77
11/64	**0.81**	0.22	0.06	0.77

Table D.10. Dynamic Clinical BERT with unbalanced dataset results.

frozen layers/batch size	weight	auc	rp60	rp80	acc
0/10	0.1	**0.82**	0.28	0.06	0.81
0/10	0.15	**0.82**	0.29	0.07	0.85
0/10	0.3	**0.82**	**0.3**	0.1	0.9
0/10	0.5	**0.82**	0.29	**0.11**	**0.91**
5/20	0.1	**0.82**	0.28	0.05	0.79
5/20	0.15	**0.82**	0.28	0.07	0.84
5/20	0.3	**0.82**	0.28	0.07	0.89
5/20	0.5	**0.82**	0.29	0.09	**0.91**
10/64	0.1	0.81	0.26	0.02	0.75
10/64	0.15	0.81	0.16	0.04	0.82
10/64	0.3	**0.82**	0.26	0.02	0.89
10/64	0.5	0.81	0.27	0.06	0.9
11/64	0.1	0.81	0.21	0.02	0.74
11/64	0.15	0.81	0.21	0.02	0.82
11/64	0.3	0.81	0.24	0.03	0.89
11/64	0.5	0.81	0.24	0.03	0.9

Table D.11. Finetuned Clinical BERT with traditional classifier on one split.

model	demo	auc	rp60	rp80	acc
LR	1	**0.81**	**0.27**	**0.1**	**0.9**
LR	0	0.8	0.24	0.05	**0.9**
XGB	1	0.79	0.23	0.05	**0.9**
XGB	0	0.79	0.21	0.06	**0.9**

Table D.12. LR Node2Vec results (balanced dataset).

balanced	size	pool	conc	demo	auc	rp60	rp80	acc
1	30	mean	1	1	0.74	0.05	0.0	0.68
1	30	mean	1	0	0.73	0.03	0.0	0.67
1	30	mean	0	1	0.71	0.06	0.0	0.66
1	30	mean	0	0	0.69	0.04	0.0	0.64
1	30	sum	1	1	0.73	0.01	0.0	0.69
1	30	sum	1	0	0.72	0.01	0.0	0.69
1	30	sum	0	1	0.69	0.11	0.0	0.65
1	30	sum	0	0	0.68	0.0	0.0	0.64
1	30	w_mean	1	1	0.75	0.05	0.0	0.69
1	30	w_mean	1	0	0.74	0.02	0.0	0.68
1	30	w_mean	0	1	0.72	0.08	0.0	0.67
1	30	w_mean	0	0	0.7	0.04	0.0	0.66
1	768	mean	1	1	0.78	0.09	**0.05**	0.72
1	768	mean	1	0	**0.8**	0.1	0.01	0.72
1	768	mean	0	1	0.77	0.06	**0.05**	0.71
1	768	mean	0	0	0.79	0.12	0.02	0.72
1	768	sum	1	1	0.78	0.08	0.0	0.74
1	768	sum	1	0	0.79	0.03	0.02	**0.75**
1	768	sum	0	1	0.78	0.05	0.0	0.73
1	768	sum	0	0	0.79	0.08	0.02	0.74
1	768	w_mean	1	1	0.79	0.1	0.02	0.73
1	768	w_mean	1	0	**0.8**	**0.17**	0.01	0.73
1	768	w_mean	0	1	0.78	0.05	0.03	0.71
1	768	w_mean	0	0	**0.8**	0.1	0.03	0.73

Table D.13. LR Node2Vec results (unbalanced dataset).

balanced	size	pool	conc	demo	auc	rp60	rp80	acc
0	30	mean	1	1	0.73	0.02	0.0	0.89
0	30	mean	1	0	0.73	0.02	0.0	0.89
0	30	mean	0	1	0.69	0.03	0.0	0.89
0	30	mean	0	0	0.68	0.02	0.0	0.89
0	30	sum	1	1	0.72	0.01	0.0	0.89
0	30	sum	1	0	0.71	0.02	0.0	0.89
0	30	sum	0	1	0.69	0.0	0.0	0.89
0	30	sum	0	0	0.68	0.01	0.0	0.89
0	30	w_mean	1	1	0.74	0.02	0.02	**0.9**
0	30	w_mean	1	0	0.73	0.05	0.02	**0.9**
0	30	w_mean	0	1	0.71	0.03	0.01	0.89
0	30	w_mean	0	0	0.69	0.04	0.0	0.89
0	768	mean	1	1	0.77	0.12	0.05	**0.9**
0	768	mean	1	0	0.79	0.19	0.03	**0.9**
0	768	mean	0	1	0.76	0.05	0.03	**0.9**
0	768	mean	0	0	0.79	0.15	0.02	**0.9**
0	768	sum	1	1	0.78	0.13	0.04	**0.9**
0	768	sum	1	0	0.78	0.15	0.04	**0.9**
0	768	sum	0	1	0.77	0.11	0.04	**0.9**
0	768	sum	0	0	0.78	0.13	0.03	**0.9**
0	768	w_mean	1	1	0.78	0.17	0.03	**0.9**
0	768	w_mean	1	0	**0.8**	**0.22**	0.03	**0.9**
0	768	w_mean	0	1	0.77	0.12	**0.06**	**0.9**
0	768	w_mean	0	0	**0.8**	0.19	0.02	**0.9**

Table D.14. XGB Node2Vec results (balanced dataset).

balanced	size	pool	conc	demo	auc	rp60	rp80	acc
1	30	mean	1	1	0.78	0.21	0.05	0.81
1	30	mean	1	0	0.77	0.13	0.03	0.8
1	30	mean	0	1	0.77	0.19	0.06	0.79
1	30	mean	0	0	0.75	0.12	0.03	0.77
1	30	sum	1	1	0.77	0.18	0.04	0.81
1	30	sum	1	0	0.76	0.17	0.04	0.8
1	30	sum	0	1	0.76	0.12	0.04	0.79
1	30	sum	0	0	0.73	0.08	0.02	0.77
1	30	w_mean	1	1	**0.79**	0.22	0.07	0.81
1	30	w_mean	1	0	0.78	0.19	0.02	0.8
1	30	w_mean	0	1	0.78	0.22	0.05	0.8
1	30	w_mean	0	0	0.77	0.14	0.03	0.78
1	768	mean	1	1	0.78	0.22	0.08	**0.85**
1	768	mean	1	0	0.78	0.21	0.05	**0.85**
1	768	mean	0	1	0.78	0.21	0.07	0.84
1	768	mean	0	0	0.78	0.2	0.05	0.84
1	768	sum	1	1	0.78	**0.23**	0.08	0.84
1	768	sum	1	0	0.77	0.2	0.06	0.84
1	768	sum	0	1	0.77	0.2	0.05	0.83
1	768	sum	0	0	0.77	0.18	0.03	0.83
1	768	w_mean	1	1	**0.79**	0.19	0.08	**0.85**
1	768	w_mean	1	0	**0.79**	**0.23**	0.06	0.84
1	768	w_mean	0	1	**0.79**	**0.23**	**0.09**	0.84
1	768	w_mean	0	0	0.78	0.21	0.05	0.84

Table D.15. XGB Node2Vec results (unbalanced dataset).

balanced	size	pool	conc	demo	auc	rp60	rp80	acc
0	30	mean	1	1	**0.79**	0.21	0.05	**0.9**
0	30	mean	1	0	0.78	0.18	0.04	**0.9**
0	30	mean	0	1	0.77	0.2	0.06	**0.9**
0	30	mean	0	0	0.75	0.1	0.03	**0.9**
0	30	sum	1	1	0.78	0.19	0.07	**0.9**
0	30	sum	1	0	0.77	0.17	0.06	**0.9**
0	30	sum	0	1	0.77	0.16	0.05	**0.9**
0	30	sum	0	0	0.74	0.09	0.03	**0.9**
0	30	w_mean	1	1	**0.79**	0.22	0.06	**0.9**
0	30	w_mean	1	0	0.78	0.19	0.05	**0.9**
0	30	w_mean	0	1	0.78	0.2	0.05	**0.9**
0	30	w_mean	0	0	0.77	0.17	0.03	**0.9**
0	768	mean	1	1	**0.79**	**0.23**	0.03	**0.9**
0	768	mean	1	0	**0.79**	0.2	0.06	**0.9**
0	768	mean	0	1	0.78	0.21	0.04	**0.9**
0	768	mean	0	0	0.78	0.19	0.03	**0.9**
0	768	sum	1	1	**0.79**	**0.23**	**0.09**	**0.9**
0	768	sum	1	0	0.78	0.2	0.07	**0.9**
0	768	sum	0	1	0.78	0.22	0.06	**0.9**
0	768	sum	0	0	0.77	0.18	0.05	**0.9**
0	768	w_mean	1	1	**0.79**	**0.23**	0.08	**0.9**
0	768	w_mean	1	0	**0.79**	**0.23**	0.06	**0.9**
0	768	w_mean	0	1	**0.79**	**0.23**	0.06	**0.9**
0	768	w_mean	0	0	**0.79**	0.22	0.06	**0.9**

Table D.16. LR PBG ontologies results (balanced dataset).

balanced	size	pool	conc	demo	auc	rp60	rp80	acc
1	30	mean	1	1	0.71	0.06	0.01	0.66
1	30	mean	1	0	0.71	0.08	**0.05**	0.65
1	30	mean	0	1	0.69	0.06	0.0	0.64
1	30	mean	0	0	0.67	0.02	0.0	0.62
1	30	sum	1	1	0.72	0.07	0.0	0.68
1	30	sum	1	0	0.7	0.01	0.0	0.66
1	30	sum	0	1	0.69	0.05	0.0	0.66
1	30	sum	0	0	0.66	0.0	0.0	0.63
1	30	w_mean	1	1	0.72	0.1	**0.05**	0.67
1	30	w_mean	1	0	0.72	0.11	0.04	0.66
1	30	w_mean	0	1	0.7	0.09	0.0	0.65
1	30	w_mean	0	0	0.67	0.02	0.0	0.63
1	768	mean	1	1	0.78	0.16	0.04	0.7
1	768	mean	1	0	0.79	0.16	0.04	0.73
1	768	mean	0	1	0.77	0.15	0.02	0.69
1	768	mean	0	0	0.78	0.15	0.03	0.72
1	768	sum	1	1	0.79	0.14	0.04	0.74
1	768	sum	1	0	0.79	0.16	0.03	**0.75**
1	768	sum	0	1	0.78	0.1	0.03	0.73
1	768	sum	0	0	0.78	0.14	0.03	0.74
1	768	w_mean	1	1	0.79	0.18	0.03	0.71
1	768	w_mean	1	0	**0.8**	**0.19**	0.04	0.73
1	768	w_mean	0	1	0.77	0.14	0.03	0.7
1	768	w_mean	0	0	0.79	0.18	0.04	0.73

Table D.17. LR PBG ontologies results (unbalanced dataset).

balanced	size	pool	conc	demo	auc	rp60	rp80	acc
0	30	mean	1	1	0.71	0.1	0.02	**0.9**
0	30	mean	1	0	0.71	0.11	0.03	**0.9**
0	30	mean	0	1	0.68	0.02	0.0	0.89
0	30	mean	0	0	0.66	0.02	0.0	0.89
0	30	sum	1	1	0.71	0.07	0.01	0.89
0	30	sum	1	0	0.69	0.04	0.01	0.89
0	30	sum	0	1	0.68	0.0	0.0	0.89
0	30	sum	0	0	0.66	0.01	0.0	0.89
0	30	w_mean	1	1	0.71	0.13	0.03	**0.9**
0	30	w_mean	1	0	0.71	0.13	0.02	**0.9**
0	30	w_mean	0	1	0.68	0.06	0.01	**0.9**
0	30	w_mean	0	0	0.67	0.03	0.01	0.89
0	768	mean	1	1	0.77	0.18	0.03	**0.9**
0	768	mean	1	0	**0.8**	0.2	**0.05**	**0.9**
0	768	mean	0	1	0.76	0.14	0.04	**0.9**
0	768	mean	0	0	0.78	0.2	0.03	**0.9**
0	768	sum	1	1	0.78	0.2	**0.05**	**0.9**
0	768	sum	1	0	0.79	0.2	**0.05**	**0.9**
0	768	sum	0	1	0.77	0.18	0.04	**0.9**
0	768	sum	0	0	0.78	0.19	**0.05**	**0.9**
0	768	w_mean	1	1	0.78	0.16	0.04	**0.9**
0	768	w_mean	1	0	**0.8**	**0.23**	0.04	**0.9**
0	768	w_mean	0	1	0.77	0.12	0.03	**0.9**
0	768	w_mean	0	0	0.79	0.22	0.04	**0.9**

Table D.18. XGB PBG ontologies results (balanced dataset).

balanced	size	pool	conc	demo	auc	rp60	rp80	acc
1	30	mean	1	1	0.76	0.2	0.07	0.8
1	30	mean	1	0	0.75	0.18	0.03	0.8
1	30	mean	0	1	0.75	0.18	0.05	0.79
1	30	mean	0	0	0.71	0.12	0.04	0.76
1	30	sum	1	1	0.75	0.15	0.04	0.82
1	30	sum	1	0	0.73	0.17	0.03	0.81
1	30	sum	0	1	0.74	0.17	0.07	0.8
1	30	sum	0	0	0.7	0.11	0.03	0.79
1	30	w_mean	1	1	0.77	0.18	0.07	0.8
1	30	w_mean	1	0	0.76	0.2	0.03	0.8
1	30	w_mean	0	1	0.76	0.19	0.04	0.79
1	30	w_mean	0	0	0.73	0.16	0.04	0.77
1	768	mean	1	1	0.77	0.21	0.05	**0.86**
1	768	mean	1	0	0.76	0.2	0.05	0.85
1	768	mean	0	1	0.76	0.21	0.05	0.85
1	768	mean	0	0	0.76	0.18	0.06	0.85
1	768	sum	1	1	0.75	0.22	0.05	**0.86**
1	768	sum	1	0	0.75	0.21	0.03	**0.86**
1	768	sum	0	1	0.74	0.19	0.07	**0.86**
1	768	sum	0	0	0.74	0.18	0.06	**0.86**
1	768	w_mean	1	1	**0.78**	**0.24**	**0.1**	**0.86**
1	768	w_mean	1	0	0.77	0.21	0.07	**0.86**
1	768	w_mean	0	1	0.77	0.22	0.09	0.85
1	768	w_mean	0	0	0.77	0.21	0.07	0.85

Table D.19. XGB PBG ontologies results (unbalanced dataset).

balanced	size	pool	conc	demo	auc	rp60	rp80	acc
0	30	mean	1	1	0.77	0.16	0.04	**0.9**
0	30	mean	1	0	0.75	0.18	0.04	**0.9**
0	30	mean	0	1	0.76	0.18	0.06	**0.9**
0	30	mean	0	0	0.72	0.13	0.02	**0.9**
0	30	sum	1	1	0.77	0.21	0.05	**0.9**
0	30	sum	1	0	0.74	0.15	0.04	**0.9**
0	30	sum	0	1	0.76	0.14	0.06	**0.9**
0	30	sum	0	0	0.71	0.1	0.02	**0.9**
0	30	w_mean	1	1	0.77	0.22	**0.08**	**0.9**
0	30	w_mean	1	0	0.76	0.21	0.05	**0.9**
0	30	w_mean	0	1	0.77	0.19	0.06	**0.9**
0	30	w_mean	0	0	0.74	0.14	0.03	**0.9**
0	768	mean	1	1	0.78	0.22	0.06	**0.9**
0	768	mean	1	0	0.77	0.2	0.05	**0.9**
0	768	mean	0	1	0.77	0.21	0.05	**0.9**
0	768	mean	0	0	0.76	0.18	0.04	**0.9**
0	768	sum	1	1	0.77	0.2	0.06	**0.9**
0	768	sum	1	0	0.76	0.19	0.04	**0.9**
0	768	sum	0	1	0.76	0.2	0.07	**0.9**
0	768	sum	0	0	0.75	0.18	0.07	**0.9**
0	768	w_mean	1	1	**0.79**	**0.23**	0.05	**0.9**
0	768	w_mean	1	0	0.78	0.22	0.05	**0.9**
0	768	w_mean	0	1	0.78	0.22	0.05	**0.9**
0	768	w_mean	0	0	0.77	0.16	0.06	**0.9**

Table D.20. LR PBG hospital graph results (balanced dataset).

balanced	size	pool	conc	demo	auc	rp60	rp80	acc
1	30	mean	1	1	0.78	0.1	0.02	0.72
1	30	mean	1	0	0.78	0.11	0.01	0.71
1	30	mean	0	1	0.77	0.1	0.02	0.71
1	30	mean	0	0	0.77	0.09	0.02	0.71
1	30	sum	1	1	0.77	0.02	0.0	0.73
1	30	sum	1	0	0.77	0.02	0.0	0.74
1	30	sum	0	1	0.76	0.01	0.0	0.73
1	30	sum	0	0	0.76	0.01	0.01	0.74
1	30	w_mean	1	1	0.78	0.13	0.03	0.72
1	30	w_mean	1	0	0.78	0.13	0.02	0.72
1	30	w_mean	0	1	0.78	0.1	0.02	0.72
1	30	w_mean	0	0	0.77	0.11	0.03	0.71
1	768	mean	1	1	**0.8**	0.19	**0.06**	0.73
1	768	mean	1	0	**0.8**	0.2	0.04	0.74
1	768	mean	0	1	0.79	0.15	0.03	0.73
1	768	mean	0	0	**0.8**	0.18	0.03	0.73
1	768	sum	1	1	0.79	0.14	0.03	**0.75**
1	768	sum	1	0	0.79	0.15	0.03	**0.75**
1	768	sum	0	1	0.79	0.07	0.02	**0.75**
1	768	sum	0	0	0.79	0.14	0.03	**0.75**
1	768	w_mean	1	1	**0.8**	0.2	0.04	0.73
1	768	w_mean	1	0	**0.8**	**0.22**	**0.06**	0.74
1	768	w_mean	0	1	**0.8**	0.16	0.05	0.73
1	768	w_mean	0	0	**0.8**	0.21	0.05	0.73

Table D.21. LR PBG hospital graph results (unbalanced dataset).

balanced	size	pool	conc	demo	auc	rp60	rp80	acc
0	30	mean	1	1	0.77	0.12	0.03	0.9
0	30	mean	1	0	0.78	0.14	0.02	0.9
0	30	mean	0	1	0.77	0.12	0.02	0.9
0	30	mean	0	0	0.77	0.11	0.01	0.9
0	30	sum	1	1	0.76	0.07	0.02	0.9
0	30	sum	1	0	0.76	0.03	0.02	0.9
0	30	sum	0	1	0.75	0.03	0.0	0.89
0	30	sum	0	0	0.75	0.03	0.01	0.89
0	30	w_mean	1	1	0.77	0.16	0.04	0.9
0	30	w_mean	1	0	0.78	0.16	0.03	0.9
0	30	w_mean	0	1	0.77	0.16	0.03	0.9
0	30	w_mean	0	0	0.77	0.08	0.03	0.9
0	768	mean	1	1	**0.8**	0.2	0.07	0.9
0	768	mean	1	0	**0.8**	0.23	0.07	0.9
0	768	mean	0	1	0.79	0.16	0.04	0.9
0	768	mean	0	0	**0.8**	0.22	0.05	0.9
0	768	sum	1	1	0.79	0.22	0.06	0.9
0	768	sum	1	0	0.79	0.21	0.06	0.9
0	768	sum	0	1	0.78	0.16	0.05	0.9
0	768	sum	0	0	0.79	0.2	0.04	0.9
0	768	w_mean	1	1	**0.8**	0.19	0.06	0.9
0	768	w_mean	1	0	**0.8**	**0.24**	0.05	**0.91**
0	768	w_mean	0	1	0.79	0.17	0.05	0.9
0	768	w_mean	0	0	**0.8**	0.23	**0.08**	0.9

Table D.22. XGB PBG hospital graph results (balanced dataset).

balanced	size	pool	conc	demo	auc	rp60	rp80	acc
1	30	mean	1	1	0.79	0.24	**0.09**	0.82
1	30	mean	1	0	0.78	0.23	0.06	0.81
1	30	mean	0	1	0.79	0.23	0.02	0.8
1	30	mean	0	0	0.78	0.2	0.06	0.79
1	30	sum	1	1	0.78	0.17	0.05	0.82
1	30	sum	1	0	0.78	0.22	0.04	0.82
1	30	sum	0	1	0.78	0.24	0.08	0.81
1	30	sum	0	0	0.77	0.22	0.03	0.81
1	30	w_mean	1	1	**0.8**	**0.25**	0.06	0.81
1	30	w_mean	1	0	0.79	0.23	0.05	0.81
1	30	w_mean	0	1	0.79	0.24	0.06	0.8
1	30	w_mean	0	0	0.79	0.18	0.05	0.79
1	768	mean	1	1	0.78	0.24	0.05	**0.86**
1	768	mean	1	0	0.78	0.21	0.04	**0.86**
1	768	mean	0	1	0.78	0.21	0.06	**0.86**
1	768	mean	0	0	0.77	0.22	0.06	**0.86**
1	768	sum	1	1	0.76	**0.25**	0.08	**0.86**
1	768	sum	1	0	0.77	0.22	0.07	**0.86**
1	768	sum	0	1	0.77	0.22	0.05	**0.86**
1	768	sum	0	0	0.76	0.2	0.05	**0.86**
1	768	w_mean	1	1	0.79	**0.25**	0.07	**0.86**
1	768	w_mean	1	0	0.78	0.24	0.07	**0.86**
1	768	w_mean	0	1	0.78	**0.25**	0.07	0.85
1	768	w_mean	0	0	0.78	0.23	0.06	0.85

Table D.23. XGB PBG hospital graph results (unbalanced dataset).

balanced	size	pool	conc	demo	auc	rp60	rp80	acc
0	30	mean	1	1	**0.8**	**0.25**	0.09	**0.9**
0	30	mean	1	0	0.79	0.23	0.06	**0.9**
0	30	mean	0	1	**0.8**	0.23	**0.1**	**0.9**
0	30	mean	0	0	0.79	0.23	0.04	**0.9**
0	30	sum	1	1	**0.8**	0.23	0.07	**0.9**
0	30	sum	1	0	0.79	0.22	0.06	**0.9**
0	30	sum	0	1	0.79	0.24	**0.1**	**0.9**
0	30	sum	0	0	0.78	0.23	0.06	**0.9**
0	30	w_mean	1	1	**0.8**	**0.25**	**0.1**	**0.9**
0	30	w_mean	1	0	**0.8**	0.24	0.07	**0.9**
0	30	w_mean	0	1	**0.8**	0.24	0.07	**0.9**
0	30	w_mean	0	0	0.79	0.22	0.04	**0.9**
0	768	mean	1	1	0.79	0.24	0.04	**0.9**
0	768	mean	1	0	0.78	0.22	0.06	**0.9**
0	768	mean	0	1	0.78	0.23	0.07	**0.9**
0	768	mean	0	0	0.78	0.17	0.03	**0.9**
0	768	sum	1	1	0.78	0.23	0.08	**0.9**
0	768	sum	1	0	0.78	0.23	0.05	**0.9**
0	768	sum	0	1	0.78	0.18	0.08	**0.9**
0	768	sum	0	0	0.78	0.22	0.04	**0.9**
0	768	w_mean	1	1	0.79	0.24	0.07	**0.9**
0	768	w_mean	1	0	0.79	**0.25**	0.05	**0.9**
0	768	w_mean	0	1	0.79	0.24	0.06	**0.9**
0	768	w_mean	0	0	0.79	0.23	0.05	**0.9**

Table D.24. LR PBG hospital graph results (episodes).

balanced	size	demo	auc	rp60	rp80	acc
1	30	1	**0.75**	0.02	0.0	0.68
1	30	0	0.74	0.02	0.0	0.67
1	768	1	**0.75**	0.06	0.01	0.69
1	768	0	0.74	0.04	0.01	0.69
0	30	1	0.74	0.03	0.01	0.89
0	30	0	0.74	0.06	0.01	0.89
0	768	1	0.74	**0.13**	**0.03**	**0.9**
0	768	0	0.74	0.08	0.01	**0.9**

Table D.25. XGB PBG hospital graph results (episodes).

balanced	size	demo	auc	rp60	rp80	acc
1	30	1	**0.77**	**0.15**	0.02	0.79
1	30	0	0.75	0.05	0.01	0.78
1	768	1	0.72	0.1	**0.03**	0.84
1	768	0	0.7	0.05	0.01	0.83
0	30	1	**0.77**	**0.15**	**0.03**	**0.9**
0	30	0	0.75	0.08	0.01	**0.9**
0	768	1	0.74	0.1	**0.03**	**0.9**
0	768	0	0.72	0.04	0.02	**0.9**