Effective Primary Healthcare Differential Diagnosis

A Machine Learning Approach

Obinna S. Agba

Effective Primary Healthcare Differential Diagnosis

A Machine Learning Approach

by

Obinna S. Agba

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

at

Delft University of Technology

to be defended publicly on Wednesday August 26, 2020 at 3:00 PM.

Student number:	4793765	
Thesis committee:	Dr. ir. Z. Al-Ars,	TU Delft, supervisor
	Dr. M. Kitsak,	TU Delft
	Dr. T. Jaber,	Medvice

Abstract

Primary health care facilities are usually the first point of call for patients seeking medical help. However, mis-diagnosis at this stage of the clinical encounter is still quite prevalent. Mis-diagnosis can be potentially harmful to the patient and even when not the case, there is an increased financial cost of arriving at the correct diagnosis borne by the patient and an increased pressure on the capacity of the medical system. The focus of this thesis is an evaluation of machine learning models which can make a differential diagnosis of possible patient conditions from presented symptoms. In this project, a systematic approach to the acquisition and generation of data relevant to the task is presented. This approach sidesteps one of the major barriers to the application of artificial intelligence methods in the health care domain i.e. access to data. With a generated dataset of approximately 5 million records, containing 801 conditions and 376 symptoms, three machine learning models - Naive Bayes, Random Forest and Multilayer perceptron (MLP) - are evaluated and compared on the generated data using the accuracy, precision and Top-5 accuracy as evaluation metrics. The Naive Bayes model achieves a 58.8% accuracy score, 63.3% precision score and an 85.3% top-5 accuracy score. The Random Forest achieves 57.1% accuracy with a precision score of 61.2% and a top-5 accuracy of 84.5%. The MLP model achieves similar performance with Naive Bayes with an accuracy of 58.8%, a precision of 63.0% and a top-5 accuracy of 85.5%. The number of symptoms expressed per condition was shown to have a strong effect on the achieved metric scores. When evaluated on a generated dataset with at least 5 symptoms per condition, the accuracy score lay between 80.2% and 83.6%, the precision was within the range of 84.2% and 87.6% and the top-5 accuracy was between 95.7% and 96.6% across all evaluated models. For a better understanding of the potential efficacy of these models in a real world setting, a number of possible real world scenarios are proposed and new datasets are generated based on these scenarios. The trained models are then evaluated on these new datasets. It is shown that model performance is closely related to the relevance and number of observed symptoms for each condition - a higher number of symptoms expressed per condition results in higher performance by the models. It is also shown that model performance degrades considerably when the new datasets are very different from the original generated data. The models perform poorly especially in the case when symptoms not usually associated with a condition are presented even when the presentation probability is still low.

Preface

This thesis aims to outline a systematic approach to the development of a differential diagnostic tool suitable for application in the primary health care domain. The results of this project would not have been possible without the assistance and guidance of Dr Tareq Jaber and Dr Ziyad Jaber from Medvice. Their contributions and advice were invaluable throughout the entire course of the project.
A forest is made of many trees and over the course of this project I have received a great deal of help and pointers. My time at the TU Delft would never have been possible without the support of the TU Delft Global Initiative. I am eternally indebted to every one there for giving me this opportunity. I would also especially like to thank Dr Arsene Fansi Tchango for being open to collaborate on the data generation part of this project as well as Alex Smirnov co-founder of Dialogue.co for his suggestions in improving the data generation process.
To all my friends, Kat, Chukky, Grace, Paula, Francis and Rafa who in more ways than they might imagine have been extremely helpful during this process, I say a big thanks.
None of this would have been possible without the supervision and encouragement of my supervisor Dr. Zaid Al Ars whose contagious enthusiasm and belief proved a constant source of inspiration throughout this project.
In the end, we are nothing but pencils in the hand of the creator and I am thankful to him for an opportunity to make this contribution.

Obinna S. Agba
Delft, August 2020

List of Figures

List of Tables

Contents

1 Introduction

1.1. Motivating the Research

Primary health care facilities are usually the first point of call for patients seeking medical help. As a brief introduction to the main topic of this thesis consider the following scenario which typically occurs at primary health care facilities and almost all facets of medical practice that involve patient interaction. An ill patient arrives at a health care facility with complaints about one or more symptoms. In a conversation with a medical practitioner - e.g. a medical doctor, for the case of this illustration - the patient is typically asked some more questions regarding the nature of the presented symptoms, their location, how long the patient has been experiencing these symptoms, possible triggers, etc. At the end of this cross-examination of the patient, the doctor - based on the noted symptoms - makes a note of possible medical conditions, in order of *likelihood*, that could be responsible for the presented symptoms. As a next step, further medical tests might be recommended to either rule out or confirm the doctor's hypothesis. This provided illustration captures very broadly the components of what is known in the medical field as obtaining a differential diagnosis.

In many cases, this process works quite well and the proper diagnosis of the patient's condition is given. However, misdiagnosis is still quite prevalent. A study [1] has shown that in the United States, 1 in 20 outpatient visits are misdiagnosed. The same study also found that almost half of these misdiagnosed conditions could be potentially harmful.

Even when the patient's health is not immediately at risk due to a misdiagnosis there is an increase in the financial cost borne by the patient. This results from the additional tests and encounters with the health care practitioners in a bid to arrive at the actual cause of the patient's condition. This, in turn, increases the burden on the medical care system since it requires more contact moments to arrive at the right diagnosis. In what can now be described as a vicious cycle this added burden can also impair the doctor's ability to make an accurate diagnosis which further exacerbates the problem and keeps the cycle going.

Circling back to the doctor's initial list of possible conditions responsible for the patient's symptoms (i.e. the differential diagnosis for the patient), it is logical to assume that the chances of a misdiagnosis are higher if in the doctor's initial list of the actual cause of the patient's symptoms is not included. This suggests that a tool which is also able to make a differential diagnosis based on the patient's symptoms would be of great value in the diagnostic process. At the very least such a tool would confirm the doctor's own evaluation of the patient and might quite possibly also include conditions which the doctor might otherwise have overlooked. In collaboration with Medvice [1] - a Dutch Health Startup which focuses on automating as much of the patient-doctor interaction as is possible - this project explores the design of such a differential diagnostic tool.

With rapid advancements in machine learning techniques, it would be possible to utilize patient data showing presented symptoms and eventually diagnosed conditions to develop an automated differential diagnostic tool. It should be emphasized from the onset that it would be an outlandish claim to suggest that this tool would be a replacement for a medical expert. The main objective would be to assist the medical practitioner in arriving at an accurate diagnosis.

[1] https://www.medvice.io/

1.2. Problem Statement and Research Questions

Problem Statement and Main Research Question

The foregoing motivation directly leads to the following problem statement and main research question:

> What is the suitability of machine learning techniques to the task of predicting a differential diagnosis based solely on the patient's presented symptoms.

Research Questions

In addition to answering the main problem statement above, the following research questions are also relevant:

1. Another challenge that this thesis would attempt to answer is how can the accuracy of a predicted differential diagnosis be measured? For typical classification problems in machine learning, the goal of the model is to identify the target class (in this case actual causal condition) based on the available features (patient symptoms). However, for this project, the desired outcome is not just an accurate prediction of the causal condition but also an identification of other conditions that might also have been responsible for the presented symptoms.

2. Access to real medical data is difficult to obtain due to a myriad of reasons which will be touched on in subsequent chapters. To circumvent these difficulties synthetic patient data was used instead. A resulting research theme requires an investigation into how best to mimic real data behavior and what the effect would be on the trained models. More specifically, what metric scores are obtained in the mimicked real data scenarios and what do these results mean for the potential efficacy of the trained models in a real world setting.

1.3. Outline of the Thesis

The rest of the thesis is organized as follows: Chapter 2 gives a quick but sufficient introduction to differential diagnosis for medical conditions. It then proceeds to formulate the differential diagnosis problem as a machine learning task. To better understand the chapters to come, relevant machine learning theory is also discussed. In Chapter 3 an in-depth explanation is provided for the data generation process which allowed for the circumvention of restricted access to medical data. Chapter 4 explains the methodology and approach adopted for model training and evaluation on selected metrics. The results of this training and evaluation process are discussed in Chapter 5. Chapter 6 concludes this project with a summary of the results and suggestions for future research.

2
Differential Diagnosis and Machine Learning: An Introduction

2.1. Background

The Merriam Webster online dictionary defines differential diagnosis as:

"*The distinguishing of a disease or condition from others presenting with similar signs and symptoms*" [1]

Calling to mind the illustration of the patient-doctor encounter at a primary health care facility in the previous chapter, the reader can see how this definition ties in with the provided illustration. The initial set of possible conditions that the doctor deemed to be responsible for the presented symptoms is, in fact, a differential diagnosis. For readers who have ever had to consult a doctor for medical advice, this illustration would be very familiar and it can be agreed that based on anecdotal evidence the differential diagnosis component in the medical practice is very ubiquitous. One author [2] while highlighting the importance of this process to the task of the doctor referred to it as the "*bread and butter of the clinician's task*".

The ubiquity and relevance of obtaining a differential diagnosis in the diagnostic process does not, however, make it a trivial task. The ability of a doctor to make accurate differential diagnosis depends quite a lot on experience attained and familiarity with the symptoms being presented [3].

The Differential Diagnosis Process

As illustrated in figure 2.1 the differential diagnosis process can be broken down into the following steps:

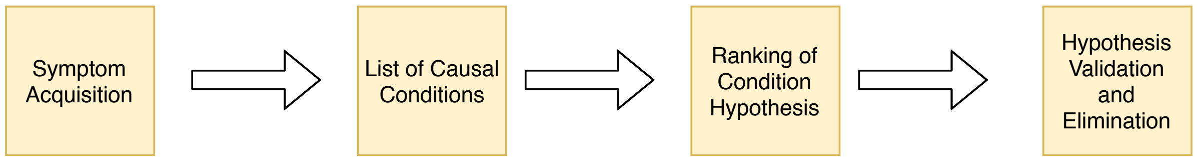

Symptom Acquisition
List of Causal Conditions
Ranking of Condition Hypothesis
Hypothesis Validation and Elimination

Figure 2.1: High-Level Overview of the Differential Diagnosis Process

Symptom Acquisition In this part of the process, the doctor examines the patient and identifies symptoms which the patient is experiencing. This process might also involve utilizing additional information about a patient's medical history. As an example of the relevance of medical history, the fact that a patient was once an active smoker for a prolonged period might provide additional context for a persistent cough even though the patient has since quit smoking.
Once the relevant patient data, symptoms and medical history have been gathered the doctor moves on to the next stage of the process.

List of Causal Conditions Based on the collected patient information and utilizing possessed medical knowledge, the doctor makes an initial list of conditions which might be responsible for the patients' symptoms. This initial list i.e. the hypothesis might be formed based on the organs where the symptoms manifest, the specific combination of the presented symptoms among other factors.

[1]`https://www.merriam-webster.com/dictionary/differential%20diagnosis`

Ranking of Conditions Hypothesis From the hypothesis of possible conditions, the doctor then ranks the conditions based on the likelihood of the condition being responsible for the symptoms. Many factors determine what the rank of an item on this initial list would be, some of which include: prevalence of the condition in the local population, current season - which might be relevant for seasonal infections, patient's age, sex, race and ethnicity, medical history of the patient, patterns observed by the doctor in previous encounters with other patients, etc.

Elimination of Conditions Once the ranking is done, the doctor then begins to eliminate or confirm the hypothesis. This process might involve requesting laboratory tests or other medical procedures. Treatment might even begin immediately with medication given to eliminate the more likely conditions first. The process of elimination is also dependent on several factors such as the severity of the symptoms, cost implications of the diagnostic tests required to confirm the hypothesis, etc.

It is worth noting that while the workflow has been presented - for sake of simplicity - as a linear top-down approach it is usually an iterative process with the flow repeated when more information becomes available. The doctor then begins an elimination of the conditions usually starting from the most likely condition first. This elimination process might involve laboratory tests that either confirm or debunk the hypothesis of the causing condition.

Given this initial list, the doctor constructs a ranking based on the probability of occurrence of each condition and the probability that the condition is indeed responsible for the presented symptoms.

2.2. Automated Differential Diagnosis

The health care industry has benefited greatly from the application of software solutions to many medical domains. This has resulted in rising adoption of electronic medical health record systems (E.M.R.s) - which enable hospitals to capture patient data, medical history, encounters with doctors etc -, systems for managing outpatient care, automating patient appointments and health care plan compliance checks just to name a few. It is safe to say that nearly all aspects of the medical domain have benefited from the digital revolution of the last few decades.

The process of generating differential diagnosis is no exception to this phenomenon. There are several tools available both to medical practitioners and patients as well which produce a differential diagnosis based on the patient's presented symptoms and sometimes other information such as medical history, patient demographics, etc. With increased global access to the Internet many of these tools, for example Symcat symptom checker [2] and WebMD [3], collectively referred to as symptom-checkers, allow a patient to enter in a web page the symptoms being experienced and receive a list of possible conditions and in some cases suggested courses of action regarding treatment.

There are also differential diagnostic tools which specifically target medical practitioners to aid their clinical reasoning during encounters with patient. One such tool - Isabel [4] - was used in a study [4] which suggested that the use of differential diagnosis tools early in the diagnostic process might help improve the accuracy of medical general practitioners. In medical lingo, automated differential diagnosis falls under a broader spectrum of tools collectively called: Clinical Decision Support Systems (C.D.S.S.). In a medical setting, such tools would typically receive as input patient data either entered manually or imported from an E.M.R. system in use by the health care facility.

Even outside active practice, such tools can be very helpful in the education process for clinical students and early-career practitioners. As has been suggested previously, the process at arriving at an accurate differential diagnosis can be complex and require a level of experience which takes time to acquire [3]. Accurate automated differential diagnosis tools also aid students in learning a framework for combining different aspects of patient information into a useful differential diagnosis.

[2]`http://www.symcat.com/`
[3]`https://symptoms.webmd.com/default.htm`
[4]`https://symptomchecker.isabelhealthcare.com/`

Automated differential diagnosis systems (and CDSS in general) would typically embed expert knowledge [5] and use this embedded information to suggest differential diagnoses for subsequent cases. This embedding process might take the form of a rule-based system that extracts diagnostic rules from a corpus of medical literature. Similar to this rule-based embedding is the use of decision trees which can capture expert knowledge. More recently, with advances in machine learning techniques, it is also possible to capture this expert knowledge via a trained model. Machine learning models can also utilize previous patient medical records with a confirmed diagnosis - which can be seen as an embedded form of expert knowledge - in arriving at a differential diagnosis for future patients.

2.3. Differential Diagnosis as a Machine Learning Problem

Recalling the differential diagnosis process shown in figure 2.1 it can be said in summary that the doctor collects information (symptoms, medical history, etc) from the patient, combines this information with obtained medical experience, environmental factors, clinical patterns, prevalence and generates a list of conditions ranked in order of the likelihood of responsibility for the presented symptoms.

If we denote all the patient information available as $p = P$ and each condition C_i in the set of all possible conditions (hypothesis) C where $C_i \in C \forall i$ then more formally we can state that the doctor ranks conditions using the probability:

$$Pr(c = C_i | p = P) \tag{2.1}$$

In colloquial terms, equation 2.1 gives the probability that the patient's condition is C_i given that P captures all available patient information e.g. demography (sex, age, race), symptoms, medical history, environmental conditions etc.

Using Bayes Law which can be stated as follows:

$$Pr(Y|X) = \frac{Pr(X|Y) \times Pr(Y)}{Pr(X)} \tag{2.2}$$

where $Pr(X)$ and $Pr(Y)$ represent the prior probability distribution of X and Y respectively , $Pr(Y|X)$ represents the posterior probability and $Pr(X|Y)$ represents the likelihood of X given Y, we can then write equation 2.1 as:

$$Pr(c = C_i | p = P) = \frac{Pr(p = P | c = C_i) \times Pr(c = C_i)}{Pr(p = P)} \tag{2.3}$$

This Bayesian formulation of the differential diagnosis task is an analog to the starting point for every machine learning problem.

2.4. Machine Learning Models

There are several machine learning models or techniques for obtaining $Pr(c = C_i | p = P)$ also known as the **posterior probability**. These techniques can be grouped into different categories e.g. linear/non-linear, parametric/non-parametric, etc. However, applying a very broad taxonomy, within the context of this project all the machine learning techniques will be grouped in two: Generative and Discriminative models.

2.4.1. Generative Models

The numerator of the R.H.S of equation 2.3 can more concisely be expressed as $Pr(P, C)$ which is known as the joint distribution of P and C i.e.

$$Pr(P, C) = Pr(C|P) \times Pr(P) = Pr(P|C) \times Pr(C) \tag{2.4}$$

This distribution can be thought of as the underlying function that maps the relationship between all patient information and all conditions. It is also sometimes referred to as the data generating distribution because once known it is possible to generate (fictitious) samples which would be identical to those found in the data set being evaluated.

Comparing equations 2.3 and 2.4 it can be seen that once the joint distribution is known the posterior probability $Pr(c = C_i|p = P)$ can easily be obtained. The joint distribution however is typically unknown and instead generative models attempt to estimate the likelihood $Pr(p = P|c = C_i)$ from the data and use this estimate to obtain the desired posterior probability using equation 2.3. The prior probabilities $Pr(p = P)$ and $Pr(c = C_i)$ are also usually estimated from the data.

Generative models typically utilize a density estimation technique where a probability density function is assumed for the likelihood $Pr(p = P|c = C_i)$ and parameters of the assumed function, which give the best fit of the density function to the available data, are estimated.

As a contrived example consider a very simple classification problem which determines if a patient has a flu based on the presence of cough. Since a patient can either have a cough or not the *feature* cough can take on the values 1 or 0. Given a dataset which contains patients with cough and the eventual flu diagnosis, the Bernoulli distribution which models a random variable which can take values 1 or 0 becomes an ideal assumption for the data generating function of this dataset.

The Bernoulli Distribution is given as:

$$Bern(x|\mu) = \mu^x(1-\mu)^{1-x} \tag{2.5}$$

Where:

- μ denotes the probability that the patient has the flu
- and x takes on the value 1 or 0 indicating the presence or absence of cough

In the estimation part of generative modeling, the parameter μ is estimated from the data such that $Bern(x|\mu)$ best fits the points in the given dataset.

Obtaining the joint probability distribution (and by extension an estimate for the likelihood) can be a difficult task. In the foregoing example, it was straight forward to select a probability distribution function that best fits the problem and also trivial to obtain an estimate of the parameters. In practice, this is not often the case. First, the space of possible probability distribution functions that might fit the data can be quite large and it is not often practical to explore this space completely. Nothing also excludes the possibility that the joint distribution is a combination of different functions. This fact is exploited by a sub-classification of generative models called mixture models. Even when the right approximating function (or functions) is selected for estimation, there is also the possibility that - depending on the size and nature of the problem being solved - estimating the parameters of the selected function might be too computationally expensive to be deemed feasible.

These considerations often lead to simplifying assumptions which make the problem easier to solve albeit at the loss of accuracy. An example of such a simplified model is the Naive Bayes model discussed below.

Naive Bayes

As an illustration, and continuing with our notation for the differential diagnosis problem, it is noted that the definition of $p = P$ encompasses all data available for the patient and includes also environmental conditions, etc. For the sake of this illustration (and indeed in practice) consider that P actually can be split into information regarding the patient's demography $d = D$, the symptoms presented $s = S$ and all other factors $o = O$. Equation 2.3 can then be re-written as:

$$Pr(c = C_i|d = D, s = S, o = O) = \frac{Pr(d = D, s = S, o = O|c = C_i) \times Pr(c = C_i)}{Pr(d = D, s = S, o = O)} \tag{2.6}$$

A density based estimation technique would aim to estimate $Pr(d = D, s = S, o = O|c = C_i)$ from the data. Naive Bayes, as an example of a simplifying technique, assumes that the parameters D, S and O are all conditionally independent given the patient's condition C_i i.e. C_i is enough to explain the relationship between these three variables. This allows for the distribution to be estimated as:

$$Pr(d = D, s = S, o = O|c = C_i) = Pr(d = D|c = C_i) \times Pr(s = S|c = C_i) \times Pr(o = O|c = C_i) \tag{2.7}$$

This is a much simpler formulation and easier to solve since each of the terms in the R.H.S of equation 2.7 can be estimated independently. As the reader might realize, the assumption of conditional independence might not hold and in such a case a reduction in accuracy becomes the price paid for this simplification.

It is however interesting to note that despite the strong assumption of conditional independence, Naive Bayes is known to perform reasonably well on many tasks [6] and research [6, 7] has shown that even when this assumption is wrong, Naive Bayes is still able to correctly classify the sample with acceptable accuracy.

2.4.2. Discriminative Models

Discriminative models on the other hand attempt to directly estimate the posterior probability $Pr(c = C_i|d = D, s = S, o = O)$ (or a proxy to this value) without the need of first obtaining the joint probability distribution. The aim of discriminative models - as the name suggests - is simply to distinguish between the different classes present in the data. As a result, any metric which provides such a distinction is a valid proxy to the target probability.

Consider an example dataset which contains presented symptoms $s = S$ and the eventual diagnosis $c = C_i$ where each symptom S_k can be considered a feature. Also assume for the sake of simplicity that there are only two conditions i.e. $i \in 0, 1$. A discriminative model might aim to define so-called decision boundaries that separate the conditions C_0 and C_1. New patients with a set of symptoms $s = S$ would then be classified according to what region of the decision boundary they lie on.

In the context of a differential diagnosis, and continuing with the example above, a discriminative model might identify multiple boundaries for each condition and the differential diagnosis for a new patient with symptoms $s = S$ would be a list of the conditions ranked according to the distance of the test sample to their respective decision boundaries.

Two such discriminative models of relevance to this project are the Random Forest algorithm and the Multilayer Perceptron. These are discussed below in further detail.

Random Forest

To give a proper explanation of the Random Forest model it is necessary that the reader is familiar with certain associated concepts:

1. **Decision Tree**

   The Decision tree is a popular algorithm used in supervised machine learning tasks both for classification and regression (where the desired output is a continuous value as opposed to the correct class). It uses a tree-like structure to infer classification rules (or functions) from the given data [8]. In a classification setting, the decision tree classifier uses a splitting-measure to rank features present in the data. A popular example of such a splitting measure is the information gain or entropy. The data is then repeatedly split based on features present - beginning with the highest ranked feature (according to the splitting measure) until all leaf nodes of the tree contain sample data points of the same class. At each split stage, the split measure is reevaluated to determine the feature that would give the next best split. In the case of the information gain, the feature which causes the highest drop in entropy at each level of the tree is used to split the data. Figure 2.2 shows a visual representation of the process.

2. **Ensemble Model**

   An ensemble model combines results from multiple models trained using the same learning algorithms. Two groups of such models are:

   - **Bagging**: where individual models are trained in parallel on randomly selected samples of the data set and the predicted class is selected based on a majority vote among the trained models
   - **Boosting**: where each model is also trained on a randomly selected sample but the training is sequential and each model takes into account the success of the previously trained model.

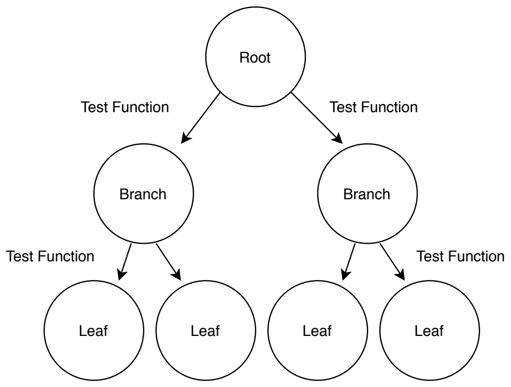


Figure 2.2: A Conceptual Decision Tree.
Ideally, for a fully grown tree, all samples in the leaf nodes would belong to the same class

The Random Forest model is an ensemble of decision trees combined using the Bagging approach. Each tree is trained using a different subset of the features present in the data. It has been shown that Random Forests are very performant and robust to noise [9]. It also does not suffer as much from overfitting the data - a problem often associated with Decision trees. Random Forest unlike many other machine learning models do not require feature normalization and can easily handle a combination of continuous, ordinal, and categorical data.

2.4.3. Multilayer Perceptrons

The multilayer perceptrons (MLP) deep learning model which is also discriminative and non-linear. Given that discriminative models seek to learn a function that maps from the input to the target label, an MLP seeks to learn the parameters that result in the best function fit [10]. More succinctly, given a classifier f which takes as input a vector x, an MLP seeks to learn θ such that $f(x;\theta)$ predicts the correct class labels associated with the input vector x. A typical architecture for an MLP is shown in figure 2.3. In the figure, the layers between the input layer (x) and the output layer (y) are called hidden layers. The figure shows only one hidden layer but in theory the MLP can have as many layers as necessary. This theoretically infinite depth is the reason for the *deep* in deep learning.
The MLP is typically trained using an iterative optimization technique called stochastic gradient descent (SGD). To apply SGD to an MLP, a loss function suited to the task at hand is selected for optimization. In each iteration of SGD the weights θ are updated so as to minimize the selected loss. The choice of the loss function is then very important as a loss which does not adequately capture learning task would result in a poorly performing model. For multi-class classification problems, the cross entropy loss is often selected for minimization.

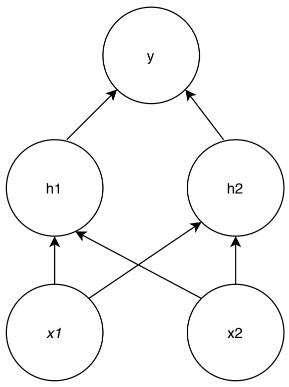


Figure 2.3: Simple MLP Architecture
Only one hidden layer is shown

2.5. Types of Data

The importance of data in the machine learning process cannot be overemphasized. Hence it would be important that the reader is familiar with the types of data typically encountered in machine learning tasks:

- **Continuous Data**: This type of data is quantitative and can take values from a possibly infinite continuous range. Examples include patient age, height measurements, etc. Depending on the algorithm being used, continuous features are usually normalized e.g. adjusted to have a unit mean and variance. This allows the algorithm to better utilize continuous features that might have different scales, e.g. patient height measurements in feet without normalization would be on a different scale (0-8) than a patient's age in years (0 - 100).
- **Ordinal Data**: This is also a quantitative data type but takes on an infinite but discrete range of values. Just like continuous data, there is an implicit ordering for ordinal features e.g., a patient's smoking history may be captured using the ordinal values - active (1), recently-stopped (2), and never smoked (3). The reader would recognize immediately an ordering in the assigned values.
- **Categorical Data**: Features of this data type take on a finite range of discrete values. There is no implicit ordering for categorical data. Examples include a patient's gender and race. There are two common methods of handling categorical data:
  1. **One-Hot Encoding**: In this method, each category is assigned a unique non-zero binary vector B with length equal to the number of categories. The *ith* column in this vector is assigned the value of 1 encoding the *ith* category. As an example given 4 race categories: White, Black, Asian and Hispanic, a one-hot encoding would assign the vector $B = [0 \quad 0 \quad 0 \quad 1]$ for White patients, $B = [0 \quad 0 \quad 1 \quad 0]$ for Black patients and so on. A downside of this method is an increase in the dimension of the feature space. In the above example, race - which is a single feature vector - has been replaced by a vector of length 4. The length of the one-hot encoded vector grows linearly with the number of categories.
  2. **Ordinal Encoding**: Due to dimensionality concerns associated with one-hot encoding, categorical variables are also sometimes encoded as ordinal variables. Continuing with the race example, a White patient would be encoded as 1, a Black patient as 2, and so on. This avoids an increase in the dimension of the feature space but it introduces an implicit *ordering* in the features which does not exist or is not desirable.

  There are other methods for handling categorical data but for the sake of brevity, the already mentioned methods would suffice.

2.6. Model Evaluation Metrics

An important component of training a machine learning model is the metric on which the model is evaluated. This determines how effective the trained model is and also serves as a means for comparing multiple models on the same task. There are several metrics on which machine learning models can be evaluated but for this project, only the following are discussed:

2.6.1. Classification Accuracy

Given N data samples, the classification accuracy is the ratio of correctly identified classes C to the total number of samples. It is the most common metric for evaluating models on a classification task.

2.6.2. Recall

In a classification task with only two labels (positive and negative classes) i.e. a binary classification task, Recall can be defined as the ratio:

$$recall = \frac{TP}{TP + FN} \tag{2.8}$$

where TP - the true positives - is the number of samples correctly identified as being part of the positive class and FN - the false negatives - is the number of samples incorrectly identified as being part of the negative class. Recall can take on values between 0 (worst case) and 1 (best case). It is a measure of the model's ability to correctly identify positive samples. This intuition becomes apparent when the reader considers that false negatives are in fact members of the positive class.

Consider this illustrative example which shows why recall can be an important metric. Given a classification task that aims to identify cancer in a patient. The outcome of such a classification would be YES (positive) or NO (negative). The reader would agree that the cost of wrongly assigning a NO-cancer label to a cancer patient outweighs that of wrongly assigning a YES label to a patient without cancer. In the latter case, subsequent tests would rule out the presence of cancer but in the former, the patient might, as a result, miss out on an opportunity to begin treatment early. Hence a high recall score for such a problem would be a better indicator of the model's effectiveness as opposed to just the classification accuracy.

When extending this to the multi-class case, there are some approaches [5] which can be taken:

1. **Micro Evaluation**: In this method the contribution of each class to the true positive and false negative count is calculated and the result is used in equation 2.8.
2. **Macro Evaluation**: In this method, the true positive and false negative count is calculated and recall values are obtained for each class using equation 2.8. The overall recall score is then calculated as the average of the obtained scores.
3. **Weighted Evaluation**: This method is similar to the macro evaluation approach with the difference being that the overall recall score is calculated as a weighted average of the scores for each class. The corresponding weights for each class indicate the prevalence of the class in the data. Hence, classes which occur less often in the data would contribute less to the overall score.

2.6.3. Precision

For a binary classification task, the precision is defined as the ratio:

$$precision = \frac{TP}{TP + FP} \tag{2.9}$$

where TP retains the definition from section 2.6.2 and FP - the false positives - indicates the number of samples incorrectly identified as being part of the positive class.
Recalling the illustrative example from section 2.6.2, it is clear that a model which immediately assigns the positive class to all samples would have the highest possible recall score. However, this would translate to a poor accuracy score and from a practical point of view (in the case of the cancer classification example) it is desirable to minimize the number of patients who need to undergo further examination when they do not have cancer. The precision score then acts as a counterbalance of sorts to the recall score as the reader would notice that assigning the positive class to all samples would result in a poor precision score. An accurate model would score high on both precision and recall. It should be noted that the methods for extension to the multi-class case discussed in 2.6.2 are also applicable to precision.

2.6.4. Top-N Accuracy

As mentioned, classifiers typically attempt to estimate the posterior probability (or some monotonically increasing/decreasing proxy of this value). The predicted class is taken as the label with the highest (or lowest - depending on the proxy selected) estimate. The Top-N accuracy also checks if the correct class is within the Top-N estimates.
This is especially relevant to the differential diagnosis task as the desired output does not necessarily require that the eventual diagnosis be labeled as the most likely but that it be in the list of suggested conditions.

2.7. Overview of the Machine Learning Pipeline

The diagram in figure 2.4 illustrates a typical machine learning pipeline showing major steps in the process of training a model and is representative of the steps taken in this project towards training models capable of making a differential diagnosis.

[5] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html`

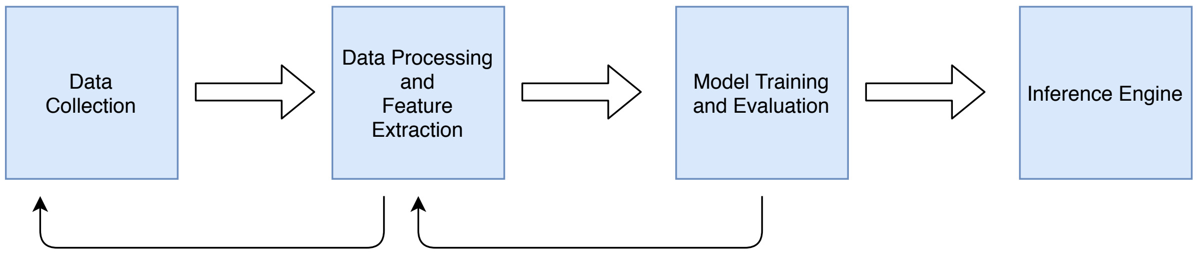

Data Collection
Data Processing and Feature Extraction
Model Training and Evaluation
Inference Engine

Figure 2.4: A Typical Machine Learning Pipeline

2.7.1. Data Collection

In the foregoing discussion *datasets* have been a recurring term hinting at the importance of data to the machine learning process. The first step of any machine learning task is indeed the collection of data. The eventual accuracy of the trained models depends just as much on the quality of the data with which the models are trained as the techniques employed in training the models.

2.7.2. Data Processing and Feature Extraction

After data collection, it is usually required to perform processing steps on the data. Such steps often include encoding or normalizing the data using schemes and techniques described in section 2.5 , handling class imbalance - a situation which arises when certain classes in the collected dataset are under-represented, dropping redundant features (feature selection), filling in missing values etc.
Feature extraction is sometimes also required and involves combining existing features in the dataset to yield new features which capture the information contained by its parent feature but help reduce the dimensionality of the data which in turn makes the training process less computationally expensive.

2.7.3. Model Training and Evaluation

With the data processed and relevant features selected and/or extracted, the next stage in the pipeline is the training and evaluation of machine learning models. In a typical setting several techniques - such as those mentioned earlier e.g. Naive Bayes, Random Forest - are trained on the data. Best practices in model training recommend that the data be split into 3: the training, validation and test set.
During training, models are trained using the train set and the validation set is used when optimizing hyperparameters of the model. The test set - which the model is never exposed to while training - is used to evaluate the performance of the model and is indicative of expected performance of the trained model on samples which it has never seen before (usually referred to as a measure of the model's ability to generalize).
In the evaluation process, the right metric is chosen as a measure of the effectiveness of the model. The concept of "*rightness*" of the selected metric depends largely on the application domain. In some domains, the accuracy of the model e.g. the ability to correctly classify a sample is the desired metric. For differential diagnosis monitoring the accuracy, precision, recall and top-N accuracy (where N defines how many conditions are expected in the differential diagnosis) would be good choices for evaluation metrics.

3 Data Generation

Medical data typically contains very sensitive information about the patients, this coupled with an increase in privacy rules surrounding access and utilization of data makes it especially difficult to access real patient electronic health records. There are often many requirements which need to be fulfilled both from legal and data security perspectives to obtain access and these requirements vary depending on the country and the institution providing the data.

Another difficulty associated with access to real medical data is the need to properly anonymize the data by eliminating any links to the patient from the data. There have been cases [11] where actual patient recovery from anonymized data was indeed possible. These factors have led many research efforts [12, 13] to resort to the use of synthetically generated patient records using data from publicly available medical databases such as PubMed [1] or other sources of such as medical literature, published records of disease symptom relationships [14]. There are also many electronic medical records generators [15, 16] which allow for a transparent, reproducible means of generating synthetic patient records etc.

To sidestep the difficulties mentioned above with regards to obtaining real patient data, use was made of one of such public disease-symptom data source in combination with a synthetic patient electronic medical record generator to generate the data used subsequently for the analysis in this project. The following sections provide more detail regarding the selected data source and generator.

3.1. Synthea: A Synthetic Patient Generator

The first component in the search for synthetic data is Synthea [16]. Written in Java, Synthea - a Synthetic Patient Population Simulator - allows for the generation of *realistic* patient medical records. To avoid privacy concerns, the generator was developed relying on publicly available medical information and health statistics. The source code for the project is also fully open-source with a permissive license which allows prospective users to modify the codebase to suit target applications. In its earliest version, Synthea modeled conditions ranked as the top 10 causes of visits to a primary health care provider and the top 10 conditions according to the "*years of life lost*" metric. Since this initial version, support has been added for more conditions with many of these contributed by its active community. This contribution is largely due to an expressive generic module framework which allows even non-technical users (with sufficient medical knowledge) to model any medical condition.

3.1.1. Synthea Architecture

The diagram in figure 3.1 gives a high level overview of Synthea's architecture. The following subsections give an overview of the main components of this architecture.

Incidence Prevalence Statistics

This contains information about the condition which is being modeled. It might include gender, age, and race prevalence, the incidence in a given population, etc. This information is eventually encoded in the disease/condition modules.

[1] https://pubmed.ncbi.nlm.nih.gov/

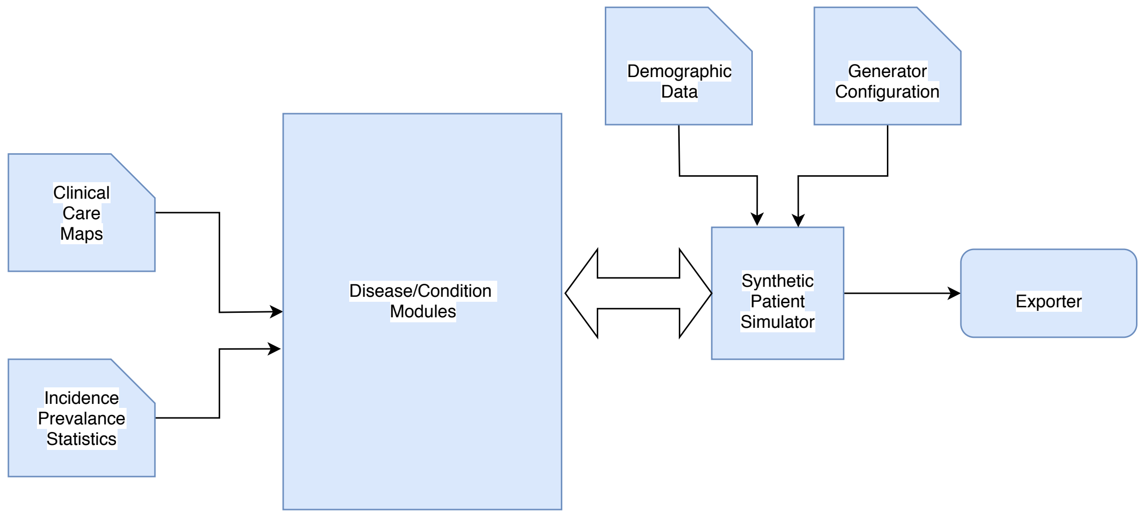

Demographic Data
Generator Configuration
Clinical Care Maps
Disease/Condition Modules
Synthetic Patient Simulator
Exporter
Incidence Prevalance Statistics

Figure 3.1: Synthea Architecture [16]

Clinical Care Maps
This component contains information about recommended care plans for patients suffering from the condition being modeled. It might include diagnostic procedures typically performed in the course of treatment, medication prescribed by the doctor, etc. Together with the incidence prevalence statistics, this component serves as input to the disease/condition modules.

Disease/Condition Modules
These are a core part of the Synthea generator and major contributing factor in its active user base. Using data from the prevalence statistics and clinical care maps, Synthea provides an expressive Generic Module Framework which allows for the proper definition of the progression of the modeled condition. These modules are designed as state machines which each state defining a step in the condition progression process. Synthea has two broad categories of states:

- Clinical States
- Control States

As the name suggests clinical states provide medically related information about the condition being modeled. They define encounters with the health care facilities or doctors, the condition which the patient develops, medications that might be prescribed, observations ranging from temperature readings associated with the condition to findings from more detailed laboratory investigation. Clinical states also encode the symptoms experienced by a patient suffering from the modeled condition.

Control states, on the other hand, are used to indicate the start and end of the disease module. They can also be used to introduce delays between other states e.g. introducing a time delay between the start of prescribed medication and the next encounter with the doctor. Some control states are also used to define special attributes on the patients or set a guard on a particular state e.g. preventing male patients from being pregnant.

All states - except the Terminal state which indicates the end of the module - must define a transition. This property determines which next state in the module the patient would move to. In combination with the clinical and control states, the state transition definitions and the prevalence statistics and clinical care maps, it is possible to completely model patient's interaction with a health caregiver while suffering from the modeled condition.

Synthea also provides a module builder [2] which allows users to graphically construct disease modules and to visualize constructed modules. A visualization of a fictitious deadly disease from the module builder

[2]https://synthetichealth.github.io/module-builder/

- *Examplitis* - is shown in figure 3.2 to give a better understanding of the disease modeling capabilities of Synthea.

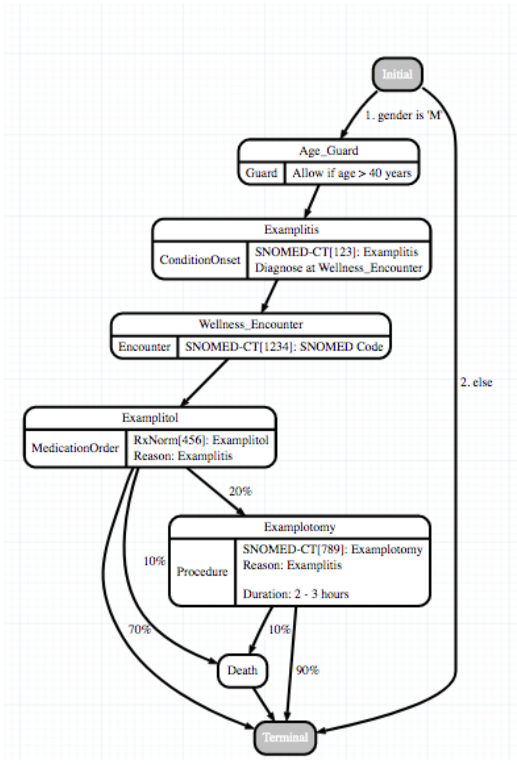


Figure 3.2: Synthea Generic Module for Examplitis (a fictitious terminal illness)
From the figure, the condition is only contracted by males older than 40 years old. All patients are first given *Examplitol*. 70% of patients on the medication end up recovering, 10% end up dying and 20% have to undergo an *Examplotomy*. 10% of those who undergo this procedure die while the rest make a full recovery.

Demographic Data
Synthea makes use of census data when generating its patient population. This data determines what the gender ratio of the generated population would be, as well as race, ethnicity, and age distribution.

Generator Configuration
The Synthea generator provides configuration options to customize the eventual output of the generation process. Configuration options include the number of patients to generate, census information to be used, specific disease modules to consider, etc.

Synthetic Patient Simulator
The patient simulator is another core component of the Synthea generator. Using the provided or specified demographic data, the simulator generates patients matching the demography. Over the lifetime of the patient i.e. from the assigned date of birth until death or the present time (whichever comes first), the patient interacts with the disease modules. The module definitions determine what encounter will be valid for each patient at each time step. Using the *Examplitis* module, a male patient would be prevented from contracting the disease until he is at least 40 years old and a female patient would never contract the disease. Once a patient reaches the terminal state for a module, the interaction between the module and the patient is stopped. If the terminal state is never reached then the patient can re-contract the disease at a future time-step. If a patient dies due to a condition (or natural causes) then the simulation for that patient ends.

Exporter
Synthea allows the generated records to be exported in different formats such as plain text, CSV files, Fast Healthcare Interoperability Resources (FHIR), and Consolidated Clinical Document Architecture (CCDA). The desired export format is specified in the Generator Configuration.

3.1.2. The Synthea Limitation

While Synthea does generate the patient records, more focus is placed on activities carried out during encounters with the health care providers e.g. laboratory procedures, payments, prescribed medication, etc. There is very little attention placed on the symptomatic expression of a particular condition. For many of the disease modules included in Synthea, symptoms were left out of the disease models and in the few cases where they were included, there was no statistical relationship between the condition being modeled and the symptoms presented. In such cases, once a patient contracts the disease, all symptoms associated with that disease would be expressed in the generated patient record. However, Synthea's expressive generic module framework provides a means to encode a disease-symptom source into Synthea compatible modules. For this project Symcat [14] was chosen as such a data source.

3.2. Symcat

Symcat is described by its creators as "A disease calculator that uses hundreds of thousands of patient records to estimate the probability of disease"[14]. It provides an interface where users can supply information about the symptoms being experienced and receive a differential diagnosis. Of greater importance to this project, however, is Symcat's conditions and symptoms directory. This knowledgebase which is publicly available on Symcat's website [3] provides probabilistic relationships between symptoms and conditions. It also includes the prevalence of diseases and symptoms by age, gender, race, and ethnicity.

Symcat Data Description

Symcat contains 801 conditions and 474 symptoms. For each condition, it gives a list of symptoms presented by patients with the condition and indicates the probability that patients will present with the specified symptom. Symcat data for each condition also contains the gender-based odds of contracting the disease. Also included are race-based odds for disease contraction. Symcat contains 4 race divisions: *White*, *Black*, *Hispanic* and *Others*. Finally, age based odds for contracting each condition are provided. Symcat has 8 age groups: *< 1 year*, *1-4 years*, *5-14 years*, *15-29 years*, *30-44 years*, *45-59 years*, *60-74 years* and *> 75 years*. It should be stated that of the 474 symptoms, only 376 of them were associated with a condition. Symptoms with no condition association were dropped.

Probabilistic Definition of Symcat Data A more formal description of Symcat's data is given below:

- A list of conditions C is provided.
- For each condition C_i, the age based odds $Pr(c = C_i | a = A_j)$ for each age group A_j is provided.
- Also, for each condition, the gender based odds $Pr(c = C_i | g = G_k)$ is provided given that $G_k \in \{male, female\}$.
- Race based odds $Pr(c = C_i | r = R_l)$ for each race group R_l, for each condition are also provided.
- For each condition C_i, a set of symptoms S^i which might be presented for that condition along with the probability that the symptom is presented $Pr(S^i_m | c = C_i)$ where $S^i_m \in S^i$ is also provided
- Additionally, for each symptom S_i, age based odds $Pr(s = S_i | a = A_j)$, race based odds $Pr(s = S_i | r = R_l)$ and gender based odds $Pr(s = S_i | g = G_k)$ are also provided

3.3. Combining Symcat and Synthea

As mentioned in section 3.1.2, the introduction of Symcat data replaces the poor symptomatic expressions in Synthea disease modules and also increases the number of modeled conditions. Symcat also has the advantage of having more modeled conditions. Incorporating Symcat data into the generation process requires the creation of Synthea-compatible modules based on Symcat data. Using the probabilistic definitions given in section 3.2, this section shows how the data from Symcat was used in generating Synthea modules.

As mentioned in section 3.1.1, a Synthea module is composed of states and each state defines a transition that determines the next state while progressing through the module. As was also mentioned, Synthea uses

[3]A scrapped CSV version of this data was provided by Alexis Smirnov of `https://www.dialogue.co/en` and is publicly available at `https://github.com/teliov/symcat-to-synthea`

demographic data to generate its patient population. Once a patient has been generated, for each disease module, the next step would require determining if the patient would contract the disease. This means formulating the probability that a patient contracts a disease based on the patients demography (age, gender and race) i.e. $Pr(c = C_i|a = A_j, g = G_k, r = R_l)$.

Applying Bayes Law (equation 2.3, this probability can be expressed as:

$$Pr(c = C_i|a = A_j, g = G_k, r = R_l) = \frac{Pr(a = A_i, g = G_k, r = R_l|c = C_i) \times Pr(c = C_i)}{Pr(a = A_i, g = G_k, r = R_l)} \tag{3.1}$$

Assuming a conditional independence for any condition, race, gender and age combination i.e assuming:

$$Pr(a = A_i, g = G_k, r = R_l|c = C_i) = Pr(c = C_i|a = A_j) \times Pr(c = C_i|g = G_k) \times Pr(c = C_i|r = R_l) \tag{3.2}$$

$$Pr(a = A_i, g = G_k, r = R_l) = Pr(a = A_i) \times Pr(g = G_k) \times Pr(r = R_l) \tag{3.3}$$

then equation 3.1 becomes:

$$Pr(c = C_i|a = A_j, g = G_k, r = R_l) = \frac{Pr(a = A_i|c = C_i) \times Pr(g = G_k|c = C_i) \times Pr(r = R_l|c = C_i) \times Pr(c = C_i)}{Pr(a = A_i) \times Pr(g = G_k) \times Pr(r = R_l)} \tag{3.4}$$

where with a repeated application of Bayes Law, we have:

$$Pr(a = A_i|c = C_i) = \frac{Pr(c = C_i|a = A_i) \times Pr(a = A_i)}{Pr(c = C_i)} \tag{3.5}$$

$$Pr(g = G_k|c = C_i) = \frac{Pr(c = C_i|g = G_k) \times Pr(g = G_k)}{Pr(c = C_i)} \tag{3.6}$$

$$Pr(r = R_k|c = C_i) = \frac{Pr(c = C_i|r = R_l) \times Pr(r = R_k)}{Pr(c = C_i)} \tag{3.7}$$

Equations 3.5 - 3.7 can then be substituted in 3.4 to obtain:

$$Pr(c = C_i|a = A_j, g = G_k, r = R_l) = \frac{Pr(c = C_i|a = A_i) \times Pr(c = C_i|g = G_k) \times Pr(c = C_i|r = R_l)}{Pr(c = C_i) \times Pr(c = C_i)} \tag{3.8}$$

It is also assumed that all conditions have an equal prior probability i.e. $Pr(c = C_i) = Pr(c = C_j) \forall i \neq j$.

We can then ignore the influence of the denominator in 3.8 since with respect to all conditions it only serves as a scaling factor. This results finally in:

$$Pr(c = C_i|a = A_j, g = G_k, r = R_l) = Pr(c = C_i|a = A_j) \times Pr(c = C_i|g = G_k) \times Pr(c = C_i|r = R_l) \tag{3.9}$$

Equation 3.9 allows for a direct formulation using Synthea's generic module framework and can be used to define transition probabilities for module states that would determine if a patient would contract a disease.

It should be noted that the conditional independence assumption implies that the condition $c = C_i$ captures any relationship between the age, gender, and race. This eliminates any possible relationship between a potential patient's age, race and gender except for the relationship based on the condition being considered.

This assumption does not always hold. As an example of a possible violation, consider the case of a hypothetical disease where a particular gender-race combination is more likely to develop the disease. This implies that $Pr(c = C_i|r = R_l, g = G_k) \neq Pr(c = C_i|r = R_l) \times Pr(c = C_i|g = G_k)$ or put otherwise $Pr(r = R_l, g = G_k) \neq Pr(r = R_l) \times Pr(g = G_k)$. Note that for simplicity sake in this example the influence of age has not been considered.

Violations aside and considering the contents of data supplied by Symcat, this is a reasonable simplifying assumption to make. The reader would recall that this is the same assumption employed by the Naive Bayes algorithm has stated in equation 2.6.

Once a patient contracts a disease, the presented symptoms are simply picked based on the probability $Pr(S^i_m|c = C_i)$ provided from Symcat's data.

Enforcing a Minimum Number of Symptoms Due to the probabilistic selection of the symptoms as described above it is possible that during the Synthea simulation process, a patient might contract a disease and not present any symptoms. While this is a perfectly valid medical scenario in practice, such data points will not provide any useful information in the training process and in a preprocessing step would be dropped. To avoid such a scenario, checks were also added - via control states - in the generated modules to make sure that a minimum number of symptoms are present for every condition. This minimum symptom count could then be configured when generating the modules and also presents an opportunity to observe how changes in this count would affect predictive performance.

A Python module [4] was developed in collaboration with Arsène Fansi Tchango [5] of the MILA [6] research institute in Quebec. This module performs two tasks: the first is parsing the Symcat data into JSON format which is easier to process while the second task incorporates the probabilistic formulations above in a Synthea module generating process. Synthea modules were generated for all 801 conditions present in the Symcat database. A sample generated module for appendicitis is shown in figure 3.3.

3.4. Generating Data

The generated modules were used as input to the Synthea generator and a patient population was generated. The CSV export format was chosen. For all datasets generated the demography was based on census data from the state of Massachusetts in the United States of America. Each dataset contained approximately 5 million records.

For the Massachusetts population, there was a slight difference in the race categorization scheme used as compared to the race categories described in section 3.2. For this demography the race categories were: *Black*, *White*, *Asian*, *Hispanic*, *Native* and *Others*. When generating Synthea modules from the Symcat data which is missing the *Asian* and *Native* race category, the probabilities associated with the *Others* race in Symcat was also applied to the *Asian* and *Native* race categories in Synthea. Table 3.1 shows sample data generated using this approach for the conditions: Appendicitis, Acute Sinusitis and Pharyngitis.

Patient ID	**Gender**	**Race**	**Ethnicity**	**Age**	**Pathology**	**Symptom Count**	**Symptoms**
05b9....	M	white	non-hispanic	7	Appendicitis	6	Vomiting;Chills;Lower abdominal pain;Nausea;Sharp abdominal pain;Side pain
b0f4...	F	white	non-hispanic	38	Acute-sinusitis	5	Cough;Coryza;Sore throat;Headache;Fever
4ce3...	M	white	non-hispanic	49	Pharyngitis	8	Hoarse voice;Cough;Coryza;Sore throat;Fever;Ear pain;Difficulty in swallowing;Wheezing

Table 3.1: Sample Generated Data for Patients with Appendicitis, Acute Sinusitis and Pharyngitis

[4] `https://github.com/teliov/symcat-to-synthea`
[5] `https://github.com/afansi`
[6] `https://mila.quebec/en/mila/`

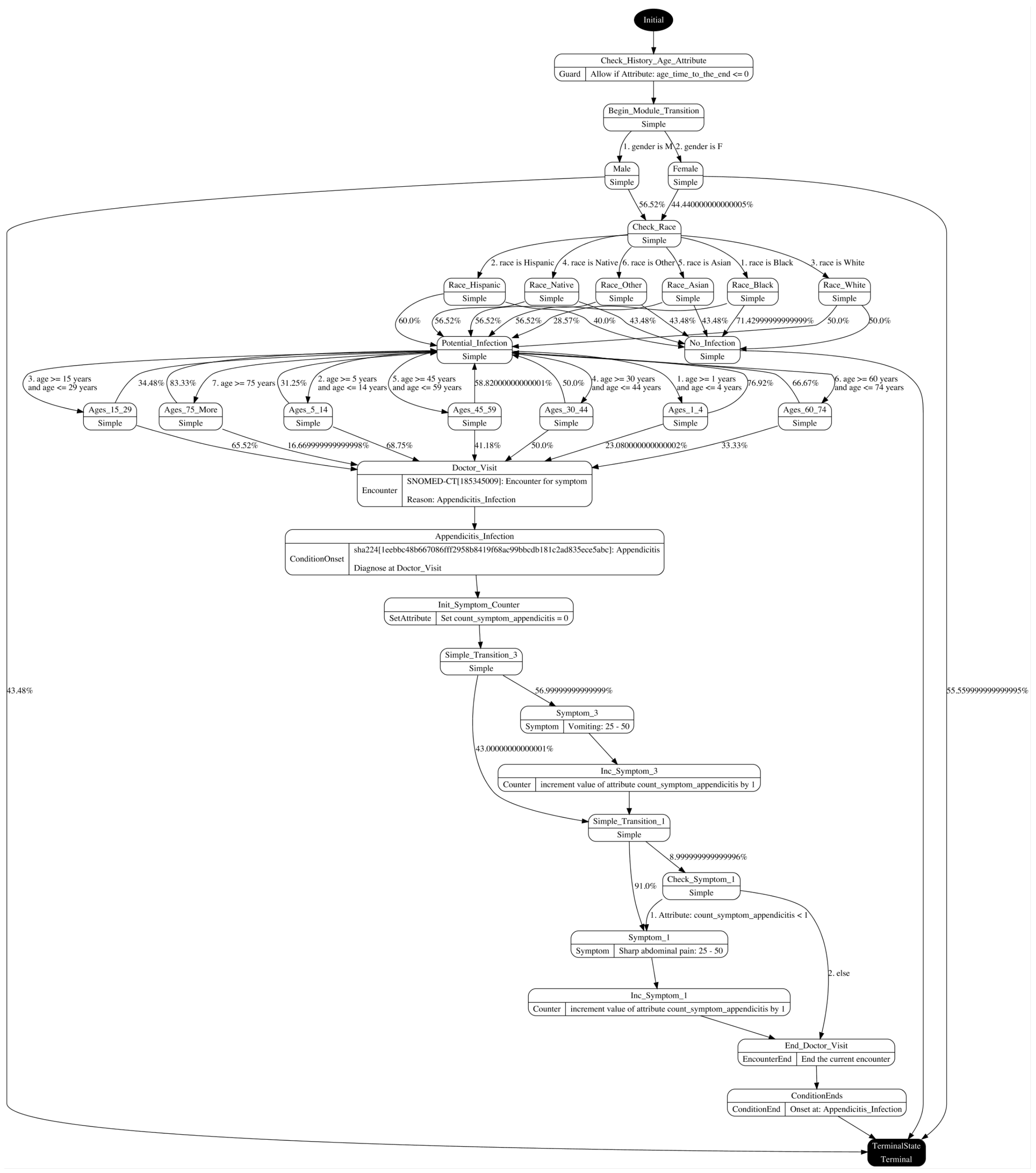


Figure 3.3: Generated Synthea Module for Appendicitis
Not all symptoms are shown to allow for an easier visualization

3.5. A Note on Synthetic Data Generation

The chosen approach of data generation sidesteps the problems stated with getting access to medical data at the start of this chapter. However, it should be re-iterated that there are downsides to using synthetic data. Already certain assumptions have been made while generating the data and it has been shown that there are scenarios where these assumptions do not necessarily hold. Also noteworthy is the fact that even for synthetic E.M.R. generators like Synthea, validation of the generated data is still an open question and so far the best attempts at validation have been comparisons with patterns observed in actual datasets.

Nonetheless, the use of synthetic data in machine learning research is very widespread and would continue to be especially in domains like the medical health sector where access to actual data is notoriously difficult and out of reach for many research teams.

4 Methodology and Approach

4.1. Data Processing Steps

Referring to the generated data presented in table 3.1, for each data point the following relevant information is available: the patient's Gender, Age and Race, the patient's Pathology and the presented Symptoms. The main processing step required the selection of proper encoding schemes for features which were not already in a numerical format. The selected encoding schemes are described below.

- **Gender**: A patient's gender is always either male or female. Hence a 0 or 1 encoding scheme was used for this feature.
- **Age**: The age feature was used without any processing for the Naive Bayes and Random Forest models. However, when training the Multilayer perceptron, this feature was normalized to have 0 mean and unit standard deviation.
- **Race**: As mentioned in sections 3.2 and 3.4, a patient's race can take discrete values. A One-Hot encoding scheme was selected. Since there is no implicit ordering present in race categories, an ordinal encoding scheme would not have been ideal. Also, the number of possible race values i.e. 5 was small enough to avoid an explosion in the dimensionality of the input feature vector to the models.
- **Pathology**: A patient's pathology i.e. the condition/disease which the patient is suffering from can also be considered as taking discrete values in the range from 0 to 800 (representing the 801 conditions available in the data). Each condition was assigned a number within this range as the class label for the condition.
- **Symptoms**: Since each patient, even for the same condition, can have a different set of symptoms the presented symptoms were considered as *yes* answers to a questionnaire containing all 376 valid symptoms. In other words, symptoms were encoded as a 376 length vector with each column corresponding to a particular symptom and taking the value of 0 except for symptoms which the patient reports as being present. It is worth noting that this results in a very sparse representation as all conditions have at most 12 possible symptoms associated with them in the Symcat data source.

4.2. Mimicking Real Data

One of the research questions put forward at the start of this project was to identify what sort of results we might expect if the trained models are evaluated on real data. To answer this question, datasets were generated in Synthea to approximate possible realistic scenarios. This section gives details about the reasoning behind these selected scenarios.

4.2.1. Baseline

This refers to data generated from Synthea using the Symcat generated modules with no modification to Symcat data or constraints imposed when generating the data.

4.2.2. Minimum Number of Symptoms

For this group of generated data sets, the minimum number of symptoms presented by patients in the data was varied from the default value of 1 to 5 which is the average number of symptoms per condition in Symcat.

From a medical point of view, it stands to reason that the higher the number of symptoms a patient presents the better the chance a doctor - and by extension, an automated diagnostic tool - has of making an accurate diagnosis. This is especially true when differentiating between conditions that share similar symptoms.

4.2.3. Perturbed Symptom Presentation Probability

While the Symcat data presents concrete statistical relationships between conditions and symptoms, it is not unlikely that in a real-world setting there would be a deviation from this relationship. Hence it might be instructive to observe how the models behave on a data set generated with a *perturbed* statistical relationship between a condition and its symptoms. To achieve this, the symptom presentation probability was randomly perturbed within a set limit ranging from 10% to 30%. This implies that assuming a fictional condition Examplitis for which the presentation probability of abdominal pain is given as 50%, then under a 10% perturbation range, the presentation probability might be as low as 45% or as high as 55%.

4.2.4. Symptom Injection

It is not uncommon in medical practice to have patients present with symptoms completely unrelated to the condition which is eventually diagnosed. This might be as a result of the fact that a different condition is responsible for the odd symptoms or that these oddities are a result of already expressed symptoms. To mimic this possibility the following approach was taken.

Likelihood Based Injection

In this method, "*likely*" symptoms were added as probable symptoms for each condition. To determine the likelihood of a symptom being presented by a condition even though it was not originally given as being related to that condition by Symcat, a graphical view of symptoms and conditions was taken. In this graph-based approach, each symptom represents a node and two nodes are connected if they are both the symptomatic expression list of a condition i.e. symptoms are connected if they can be presented for the same condition. With this approach, two nodes can have more than one connecting edge indicating that the represented symptoms can present together in more than one condition.

Hence, given a condition C_i with its set of symptoms S^i and given that S^m represents the set of symptoms such that $S^i \cap S^m = \emptyset$ then the likelihood of a symptom S^m_k where $S^m_k \in S^m$ being presented by the condition C_i is given as:

$$K = \sum_{j=1}^{i} E_{ki} \tag{4.1}$$

where E_{ki} is the edge count between symptom S^i_j and S^m_k given that $S^i_j \in S^i$ and $S^m_k \in S^m$.

In essence, this quantity measures how often the symptom being considered is presented alongside symptoms of the condition. A higher K value is taken to indicate a high *likelihood* that that symptom might present for the given condition even though it is not in the given symptomatic expression for the condition.

A maximum of 5 such symptoms were injected per condition. Once the symptoms to be injected were determined, three methods of assigning the probabilistic expression for a condition were explored. Symptoms were equally assigned a value equal to the most probable symptom, least probable symptom and the mean probabilistic expression value for the condition.

4.2.5. Data Augmentation

In many machine learning applications, it is often desired to make the output models more robust to possible deviations from the train data as is often experienced in a real world setting or to avoid overfitting. One popular method of achieving this robustness is Data Augmentation [17, 18]. In this method, samples which are different from those present in the training data are generated (or collected). Often, these samples are generated from populations where the original model typically performs poorly. This process of data augmentation is one of the more popular uses of synthetic data in machine learning problems.

This technique was also applied to this project. A dataset obtained by combining samples from all previously generated datasets was generated with care taken to maintain the existing class balance. Models would then be trained on this dataset and evaluated on other generated datasets.

4.3. Model Evaluation Metric Selection

As mentioned in section 2.7.3 selecting relevant metrics for evaluation is an important part of any machine learning task. For this project, the **accuracy** and **precision** metrics were selected. For the precision the weighted extension to the multiclass case was used as described in 2.6.2. For the multiclass weighted extension, the recall is the same as the accuracy score and hence is not explicitly reported. Also, because we are interested in a differential diagnosis and not just the exact diagnosis, the first five most likely conditions were taken as the required differential diagnosis. Hence, the Top-5 accuracy was also used when evaluating the models.

4.4. Selected Models

The space of machine learning models is very large and this project does not aim to make an exhaustive comparison of all possible choices. For this project the following models were selected and evaluated:

4.4.1. Naive Bayes

The Naive Bayes model was a logical choice following from the formulation of the data generation task of this project. The conditional independence assumption used in the data generation process is the same assumption made by the Naive Bayes algorithm. Also, this model allows for the most suitable probability distribution function to be used for each feature. This flexibility allows us to assume that the patient's age is distributed according to a Gaussian distribution, the patient's race assumes a categorical probability distribution and the gender along with all the symptoms (which take on values of 0 or 1) assume a Bernoulli probability distribution.

Hence, given that we want to determine the probability $Pr(c = C_i | a = A_i, g = G_j, r = R_l, S)$ for all conditions $C_i \in C$, it follows from the conditional independence assumption that:

$$Pr(c = C_i | a = A_i, g = G_j, r = R_l, s = S_k) = Pr(a = A_i | c = C_i) \times Pr(r = R_l | c = C_i) \times Pr(g = G_j | C) \times Pr(S | c = C_i) \tag{4.2}$$

Assuming a Gaussian distribution for the age, we can obtain $Pr(a = A_i | c = C_i)$ as:

$$Pr(a = A_i | c = C_i) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp^{-\frac{1}{2}(\frac{x-\mu_i}{\sigma_i})^2} \tag{4.3}$$

where $x = A_i$ and using maximum likelihood estimation we obtain μ_i and σi are the mean age and variance of patients with condition C_i.

With a categorical distribution assumed for the patient's race, we obtain $Pr(r = R_l | c = C_i)$ as:

$$Pr(r = R_l | c = C_i) = \prod_{l=1}^{k} p_l^{[r=l]} \tag{4.4}$$

where using the maximum likelihood estimate p_l is the ratio of patients of race $r = R_l$ who have condition C_i to the total number of patients who have that condition i.e. the probability that a patient suffering from condition C_i belongs to race $r = R_l$

With Gender taking on a Bernoulli distribution, we obtain $Pr(g = G_k | c = C_i)$ as:

$$Pr(g = G_k | c = C_i) = p^k (1-p)^{1-k} \quad \text{for} \quad k \in \{0, 1\} \tag{4.5}$$

where 0 represents females and 1 represents males and with a maximum likelihood estimate p is the ratio of males who have the condition C_i and conversely $1 - p$ is the ratio of females with the condition.

The explanation given for the Bernoulli distribution above is also valid for the symptoms presented by the patient. The only distinction is that since S is a set of symptoms then:

$$Pr(\mathbf{S} | c = C_i) = \prod_{k=1}^{K} Pr(S_k | c = C_i) \tag{4.6}$$

where each $Pr(S_k|c = C_i)$ is then estimated using a Bernoulli distribution.

The Naive Bayes formulation even though simplistic is nonetheless a very effective classifier and, as discussed, has been known to give decent results in very many applications, even when the conditional independence assumption does not hold [6]. However, while the Naive Bayes performs very well in identifying the correct class, it does not properly estimate the probabilities of other classes [19] which by extension suggests that its differential diagnosis output would not necessarily be accurate. It nonetheless provides a good baseline for comparing other models with better estimation capabilities.

4.4.2. Random Forest

The Random Forest model is also explored in this project. Its ability to handle a mix of quantitative and categorical data as well as robustness to noise and overfitting make it an attractive choice for exploration. It also provides better posterior probability estimates than the Naive Bayes making it a more ideal model for the problem. As discussed previously, fewer hyperparameters make model optimization an easier task from a computational perspective. The following section gives a summary of the Random Forest's hyperparameters selected for optimization.

Relevant Hyperparameters

The following are hyperparameters for the Random Forest which were selected to be optimized:

Maximum Tree Depth This parameter determines how deep each tree in the forest is allowed to grow when fitting the data. A larger tree depth usually means the generated forest is better able to fit the data set. It also translates to larger model size, and longer train time. This parameter can take on any integer value and can also be configured to allow trees in the forest to grow to their full depth.

Minimum Split Samples The minimum split samples determines the number of leaves i.e. data points that are allowed to be on a non-pure node (a node in the tree where all the samples are not of the same class). Smaller values result in a better fit of the data but like the maximum tree depth there is also an increase in the size of the tree, and the train time.

Minimum Leaf Samples This determines what the minimum number of leaves at each node in a tree in the forest should be. Splitting a node would only if the resulting children nodes have at least this many samples. Smaller values of this parameter follow a similar logic to the already discussed parameters i.e. increased fitting capability at the expense of model size, and train duration.

Maximum Number of Features As discussed previously, each tree in the Random Forest ensemble is trained on a subset of the features present in the dataset. This parameter then determines what the maximum size of this subset would be.

Number of Estimators Perhaps the most important parameter, the number of estimators determines how *large* the forest is i.e. the number of trees in the forest. Higher values translate to a larger tree and better fit on the data but also larger model size and train time.

In other to determine the best combination of parameters, a Grid Search approach was chosen. In this method, different combinations of the parameters are evaluated on a selected metric and the parameter combination with the best performance is selected. In an exhaustive Grid Search, all possible combinations would be evaluated. However, due to run time and memory constraints the exploratory space was reduced to make the search feasible.

Optimization Metric

When considering a suitable optimization metric, it was desired that the selected candidate performs well with regards to the validation score and that it has a reasonable model size (measured in megabytes) and train time (in seconds). Combining these requirements resulted in a multi-objective optimization problem. A simple linear scalarization approach was taken i.e.

$$\max_{p \in P} \sum_{i=1}^{N} \omega_i t_i(p) \tag{4.7}$$

where P is the hyper-parameter search space, t_i represents individual optimization objectives (validation score, training time, model size, p represents a particular combination of all hyper parameters and ω_i are weights which determine the importance of each objective to the final metric.

The following metric was formulated as the optimization target:

$$metric = \max_{p \in P} \quad (2 * validation\_score - 0.1 \times train\_time - 0.1 \times model\_size) \tag{4.8}$$

In equation 4.8 above, negative weights indicate that smaller values of the corresponding objectives would result in a better metric. To avoid a drastic loss in accuracy, an additional constraint was added which considered only parameter combinations that yielded a validation score within 1% of the maximum score in the search space. It should also be noted that all parameters in equation 4.8 are normalized.

Ideally, a grid search of the entire hyper-parameter space would yield the most optimal configuration, however due to computational and time constraints, the actual search space was restricted to allow for a feasible search. Within this restricted search space, the optimal hyper-parameters are shown in table 4.1. The optimization operation was carried out on the baseline dataset and these obtained configuration was used for all other generated datasets.

Hyper-parameter	Value
Maximum Tree Depth	380
Minimum Split Samples	2
Minimum Leaf Samples	2
Maximum Number of Features	$\log_2(num\_features)$
Number of Estimators	20

Table 4.1: Selected Hyper-parameters for Random Forest

4.4.3. Multilayer Perceptrons

The suitability of multilayer perceptrons to the differential diagnosis task is also evaluated in this project. The MLP is trained to minimize the cross entropy loss - a commonly selected loss function for multi-class classification tasks. The selected architecture of the MLP is shown in table 4.2. The rectified linear unit (ReLU) is used as the non-linear activation between fully connected layers. The weights of the layers are initialized using the so called He-Initialization [20] which has been shown to work well with ReLU activations. Due to time and computational constraints, an exhaustive hyper parameter optimization was not performed for the MLP. Also comparisons with the other explored models are restricted to those trained on the baseline data.

Layer Type	Activation	Input Dimension	Output Dimension
Linear	ReLU	383	1024
Linear	ReLU	1024	1024
Linear	None	1024	801

Table 4.2: Multilayer Perceptron Architecture
Note that the input dimension of the first layer is obtained as the sum of the available symptoms (376), the length of the one-hot encoded race vector (5) and the patient's age and gender. The output dimensions correspond to the number of conditions being predicted.

4.5. Computing Platform and Technology Stack

Computing Platform

Except where otherwise stated all experiments were carried out on the computing cluster of the Quantum and Computer Engineering (Q&CE) at the Delft University of Technology [21]. This project made use of four compute nodes on the cluster each equipped with 28 cores, 192GB of memory and a CPU speed of 2.4GHz. The Q&CE cluster is equipped with the SLURM [22] workload manager which was used to schedule training and evaluation jobs on the compute nodes.

Technology Stack

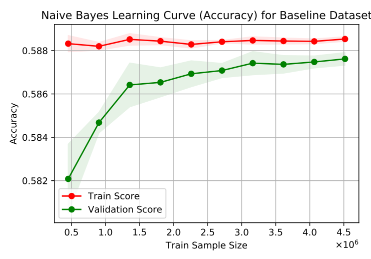
Data generation, as has been mentioned, was carried out using Synthea - a JAVA based application. The Q&CE cluster, as at the time of writing, does not support JAVA applications out of the box. To resolve this, a Singularity [23] container was built specifically for running the data generation process with Synthea on the cluster. All other components of this project were performed using Python libraries. NumPy (v1.18.2), SciPy (v1.4.1) [24] and Pandas [25] (v1.0.2) were used for data analysis and exploration. The scikit-learn (v0.22.2) [26] library provided the implementation for all the machine learning models trained and all plots where generated using Matplotlib (v3.2.1) [27]. MLFlow (v1.8.0) [28] was used to track results from different experiments.

5
Results and Discussion

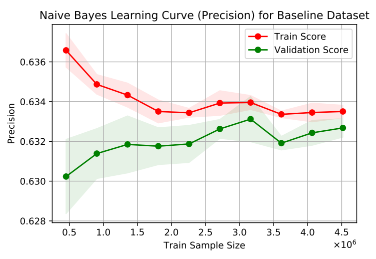
In this chapter, we discuss the results obtained using the models discussed previously with model performance evaluated using the aforementioned evaluation metrics. We first observe the performance of the selected models on the different generated data sets. This would provide some insight into how the models might behave in a real-world setting.

5.1. Performance on Baseline Data

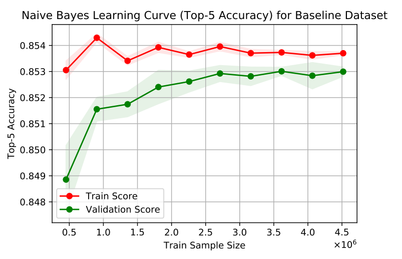
The baseline dataset was generated as described in section 4.2.1. For each selected metric a learning curve was generated with training size varying from 10-100% of the full training data. For each training sample size, 5-fold cross validation The learning curves for the trained Naive Bayes and Random Forest models are shown in the figures below.



(a) Accuracy

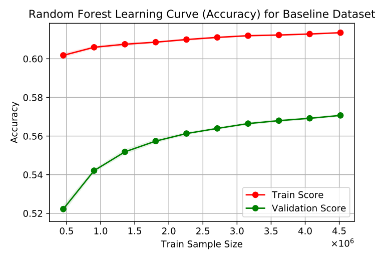


(b) Precision

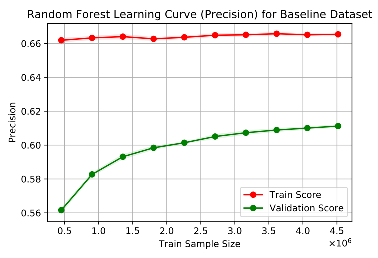


(c) Top-5 Accuracy

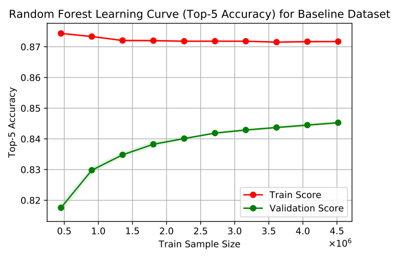
Figure 5.1: Learning Curves for Naive Bayes on Baseline Dataset
The shaded region around each plot indicates the standard deviation of the results.



(a) Accuracy



(b) Precision



(c) Top-5 Accuracy

Figure 5.2: Learning Curves for Random Forest on Baseline Dataset
The shaded region around each plot indicates the standard deviation of the results.

For both Naive Bayes and Random Forest, the learning curves show an increased performance on the validation set with an increase in the training sample size - which is typical behavior for learning curves. However, it can be seen that the performance gap between the training set and the validation set is much higher in the case of Random Forest than that of Naive Bayes.

Table 5.1 - which summarizes results from an evaluation on the full training set - shows that for all the metrics Naive Bayes performs better on the validation set than Random Forest while the reverse is the case for performance on the training set. This result indicates that there is perhaps a higher level of overfitting on the train set in the case of Random Forest than Naive Bayes. The almost nonexistent shaded region in the plots for the Random Forest model show that results - both in the train set and the validation set - are more stable than that of Naive Bayes for which there is a higher level of variation in performance especially on the validation set. Judging by these metrics, the Naive Bayes model is the better of the two though only by a small margin.

	Naive Bayes	Random Forest
Accuracy (Train)	0.589	0.614
Accuracy (Val.)	0.588	0.571
Precision (Train)	0.634	0.665
Precision (Val.)	0.633	0.611
Top-5 Accuracy (Train)	0.854	0.872
Top-5 Accuracy (Val.)	0.853	0.845

Table 5.1: Comparing Naive Bayes and Random Forest on Baseline Dataset
Results were obtained from an evaluation on the full training set with 5-Fold Stratified Cross Validation

5.1.1. Limits on the Accuracy of the Models

From the results discussed so far, the reader would notice that the accuracy obtained is quite low. For both models, this value is between 57-58%. One explanation for this low accuracy value is the difficulty distinguishing between conditions that present similar symptoms. The models simply would be unable to discriminate against conditions such as *Chronic Sinusitis* and *Acute Sinusitis* that both present near identical symptoms. This explanation is also in someway corroborated by the much higher Top-5 accuracy scores.

Another plausible explanation for the low values might be the number of symptoms presented by each condition. The reader would recall that when generating the baseline dataset, the only requirement enforced was that each condition should have at least one symptom. Figure 5.3 shows a histogram of symptoms per record in the data. Records with 3 or fewer symptoms account for about 57% of all generated records. Some symptoms can be very indicative of the condition e.g. presence of *Abdominal Pain* would strongly suggest a gastro-intestinal condition but many other symptoms - on their own - do not contain enough information to accurately determine a diagnosis.

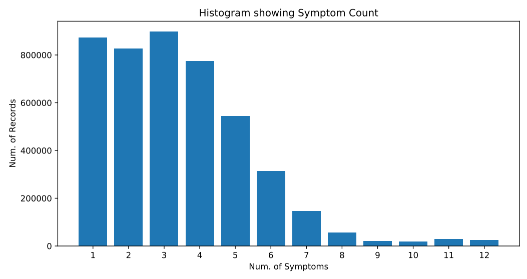


Figure 5.3: Histogram of Symptom Count in Baseline Data

5.1.2. Confusion Matrix

It would also be informative to investigate the confusion matrix for both models to gain more insights into the model behavior. However, the number of conditions is quite large - 801 - and this makes a presentation of the full confusion matrix cumbersome. For the sake of this discussion, a set of conditions were selected to make the presentation much easier. The underlying assumption with this approach is that the differences in performance between the models on these selected conditions can be extrapolated to the entire dataset.

Using the International Statistical Classification of Diseases and Related Health Problems (ICD) [1] system, the conditions are selected from three separate groups or blocks: diseases of the respiratory system, diseases of the digestive system and diseases of the genitourinary system. Conditions which fall within the same block all have codes beginning with the same letter of the English alphabet. The selected conditions are given in table 5.2.

Condition Group	ICD-10 Block Range	Condition Name	ICD-10 Code
Respiratory	J00–J99	Acute Bronchitis	J20.9
		Acute Sinusitis	J01.9
		Asthma	J45.9
Digestive	K00–K93	Appendicitis	K37
		Chronic Constipation	K59.00
		Acute Pancreatitis	K85.9
Genitourinary	N00–N99	Urethritis	N34.1
		Pyelonephritis	N12
		Cystitis	N30.9

Table 5.2: Selected Conditions for Confusion-Comparison

Tables B.2, B.3 and B.4 in the appendix show the top 5 wrongly assigned labels for each condition in the respective groups of table 5.2. Comparing the models show a high level of agreement in the predictions. Also, the top 5 wrongly assigned conditions, in nearly all cases, also belong to the same condition group as the correct condition. Conditions in the same group usually have similar symptomatic expressions and this backs up the explanation that the models need more information in such cases to make a proper distinction.

5.1.3. Comparing the Top-5 Predictions

Using the same conditions as listed in table 5.2, the top-5 co-occurring predictions are also compared across the models. In other words, given that the prediction is Top-5 accurate, the tables B.5, B.6 and B.7 show 5 other conditions which are most commonly included in the Top-5 along with the selected conditions. Again, we observe a strong similarity between the predictions of the models and these top co-occurring conditions in all cases also fall under the same condition group as the conditions being evaluated.

For a differential diagnostic tool, however, the Top-5 accuracy and the condition group similarity between the top 5 wrong labels and the target condition as well as the similarity between the top co-occurring conditions suggests that this approach is indeed feasible. In a practical setting the doctor would then request for additional tests to begin the elimination/confirmation part of the differential diagnosis flow.

5.1.4. Posterior Estimate Comparisons

In addition to the qualitative comparisons made in the preceding discussion, it is also interesting to observe the posterior estimates for each of the models. Since the differential diagnosis - in the case of this study - is taken as the first 5 conditions, and this selection is made based on the posterior estimates from each model, we can also say that the posterior estimates tell how *confident* the models are about the diagnosis. We observe this confidence for both models in the following scenarios:

1. Top-1 Accurate
2. Top-5 Accurate
3. Non-Top-1 Accurate
4. Non-Top-5 Accurate

Top-1 Accurate Figure 5.4 below compares the posterior probability estimate for both models when they are accurate in predicting the correct diagnosis as the most likely condition. What is particularly interesting is Naive Bayes' seemingly very high probability estimates (median of 0.97) in this case when compared

[1] `https://en.wikipedia.org/wiki/ICD-10`

to the more moderate Random Forest (median of 0.57). Another interesting consequence of this *confident* classification is that the next top 4 predictions of Naive Bayes receive much lower (and almost similar - by the median) estimates. This behavior of Naive Bayes is in line with its ability to discriminate the correct class without having accurate probability estimates [19].

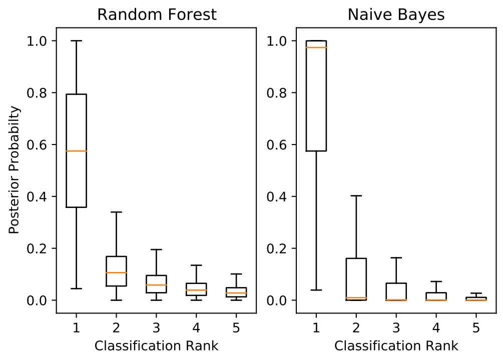


Figure 5.4: Comparing Posterior Probability Estimates for Top-1 Accurate Classification

Top-5 Accurate When comparing the posterior probability estimates for the Top-5 accurate case, again the Naive Bayes selects the most likely condition with higher confidence (median of 0.68) than the Random Forest (median of 0.43). When these values are compared with the Top-1 accurate case above, the median confidence level of Naive Bayes was reduced by 30.52% compared to a 25.28% reduction for the Random Forest.

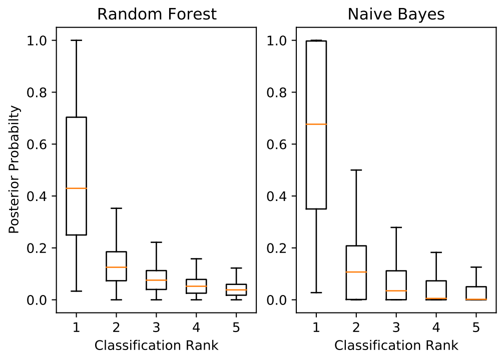


Figure 5.5: Comparing Posterior Probability Estimates for Top-5 Accurate Classification

Non-Top-1 Accurate In this case, the probability estimates for both models are much lower compared to Top-1 accurate as seen in figure 5.6. This behaviour is ideal since it would be desirable that the models are not very confident when they are wrong. This also allows for the use of a confidence threshold in practice. In such a setting, predicted diagnoses would be presented only if the most accurate prediction exceeds the set confidence threshold. The median posterior probability estimate or the 25th percentile estimate for the most likely condition are possible values for such a confidence threshold.

Non-Top-5 Accurate Figure 5.7 shows the posterior estimates for both models when the top 5 predictions do not contain the actual diagnosis. The low posterior estimates for both models reinforce the argument for a confidence threshold when presenting the predicted diagnoses to a user.

Confidence Threshold Evaluation To better understand the effects of a confidence threshold applied to exclude possible wrong predictions, different threshold values were evaluated on predictions made by the

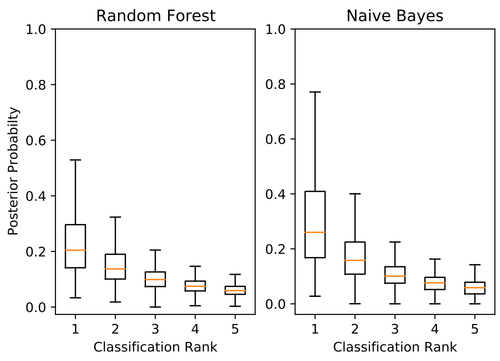


Figure 5.6: Comparing Posterior Probability Estimates for Non-Top-1 Accurate Classification

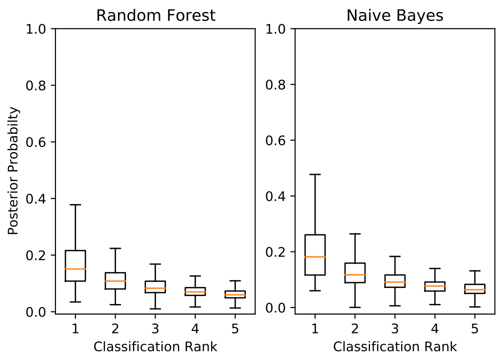


Figure 5.7: Comparing Posterior Probability Estimates for Non-Top-5 Accurate Classification

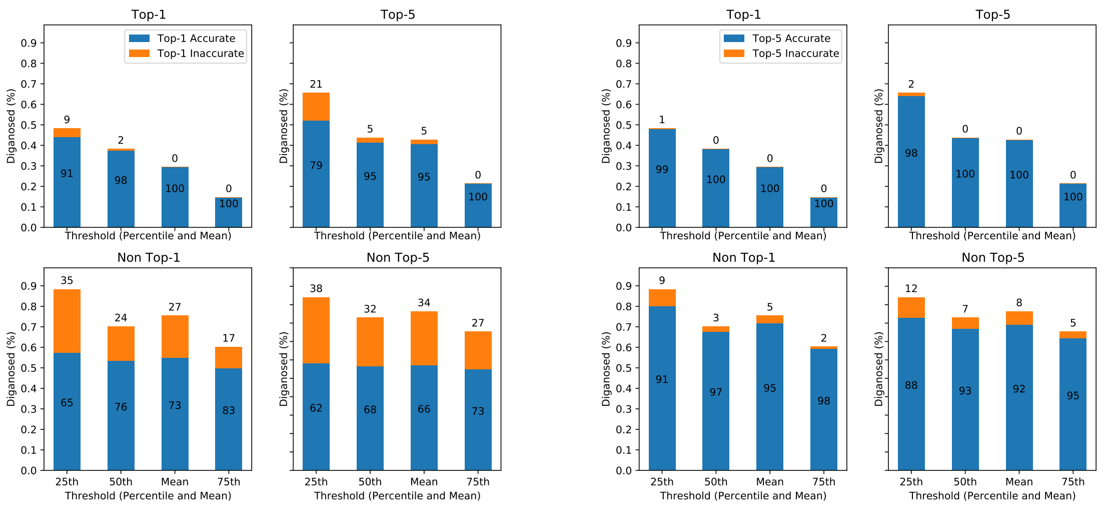
Naive Bayes model on the test set. The selected threshold values were the 25th percentile, the median, the mean and the 75th percentile of the posterior probability estimates for the top-1 predictions on the train set. Figures 5.8a and 5.8b show the effects of these thresholds on the Top-1 and Top-5 accuracy respectively. From the figures, we see - as expected - that increasing the threshold values increases the accuracy (both top-1 and top-5) obtained but also reduces the number of predictions which are considered "*confident*". Taking the 25th percentile of the Non Top-1 accurate case, the model obtains a 65% Top-1 accuracy and a 91% Top-5 accuracy and it admits 88% of the predictions. In a deployed environment, such a threshold value might be considered ideal. Predictions below the threshold would not typically be presented to the doctor.

5.2. Non-Baseline Data

In this section, we train and evaluate both the Naive Bayes and Random Forest on datasets generated as described in section 4.2. The reader would recall that these different datasets were generated to observe model behavior in possible real-life scenarios. Ultimately, models trained on the baseline data would be evaluated on these datasets but as a precursor we compare performance of models trained on these modified datasets with the performance obtained in section 5.1. This might give some insight as to how the baseline model would perform on this data eventually.

5.2.1. Varying Minimum Number of Symptoms

Recall that for these datasets, the minimum number of symptoms was varied from 2 to 5. Following, from the comments on the obtained accuracies in the baseline case, we expect that if the number of symptoms presented increases then the models would do a much better job at identifying the actual condition. Figure 5.9 shows the accuracy, precision, and Top-5 accuracy plots for each minimum number of symptoms. The baseline case i.e. minimum of 1 symptom was also included for comparison. From the figure, we do indeed see an almost linear increase in metric scores with an increase in the minimum number of symptoms. This highlights the importance of symptom discovery/acquisition in a general differential diagnosis.



(a) Top-1 Accuracy

(b) Top-5 Accuracy

Figure 5.8: Naive Bayes Top-1 and Top-5 Accuracy and Percentage of Diagnosed Conditions for Different Posterior Probability Threshold Values

5.2.2. Perturbed Symptom Presentation Probability

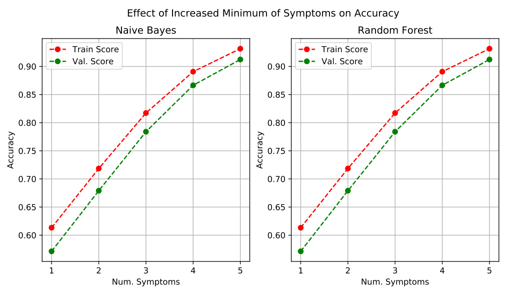
Models were also trained on a dataset in which the probabilistic relationship between symptoms and conditions were perturbed as described in section 4.2.3. The results for both Naive Bayes and Random Forest are shown in figure 5.10

Effectively, perturbing the condition-symptom probabilistic relationship increases the probabilities of certain symptoms being presented while reducing that for other symptoms. Since these perturbations were randomly applied, in some cases, this led to an increase in the number of symptoms expressed per condition which - as has been previously established - increases the models' ability to distinguish between conditions.

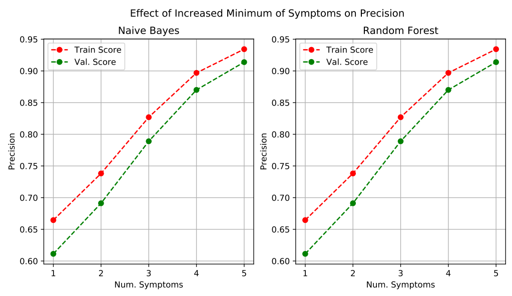
Figure 5.11 shows the average number of expressed symptoms per condition. We see that in the perturbed-50% case, the average symptom count is much higher than the others and this correlates with the recorded performance. However, solely by this reasoning, we should have obtained much worse performance from the perturbed-70% case. One possible explanation for this irregularity is that in the perturbed-70% case, there is a widened gap between symptoms whose expressive probabilities were increased and those for which the values were reduced. This widened range would have increased the overall predictive potential of the positively increased symptoms and made them more relevant to the models when making distinctions between the conditions. This explanation is made stronger by the empirical observation that for the perturbed-70% data, which had 49.49% of its samples present at most two symptoms, a trained Naive Bayes model got an accuracy score of 44.14% on conditions with at most two symptoms compared to a 20.60% accuracy score with the same setting on the baseline dataset which had 37.58% of its samples present with at most two symptoms. This increase was also the case for the Random Forest model. This theory aligns with medical reasoning where for certain cases the presence of a particular symptom becomes a strong indicator for the actual causal condition.

5.2.3. Symptom Injection

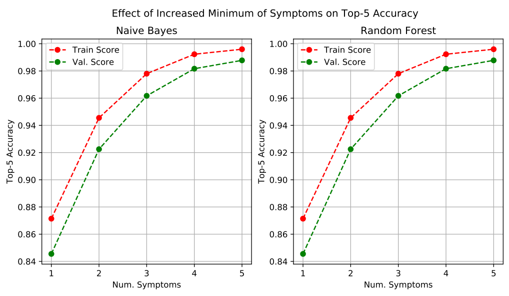
The last realistic scenario was the injection of so-called *similar symptoms* as described in section 4.2.4. Recall that once the similar symptoms were injected, three methods were used to assign their probability of expression: the largest, smallest, and mean probability value of existing symptoms for each condition. The results for models trained on these different datasets are shown in figure 5.12. From this figure, we observe a significant difference in the max-injected based model compared to the baseline based model. This might be explained by the fact that for the symptoms injected with maximum probability, they increase both the relevance and number of the symptoms present per condition - also suggesting a significant change in the data distribution. There is almost no change when comparing the min-injected based Naive Bayes model



(a) Accuracy



(b) Precision



(c) Top-5 Accuracy

Figure 5.9: Effect of Increased Minimum Number of Symptoms on Evaluation Metrics

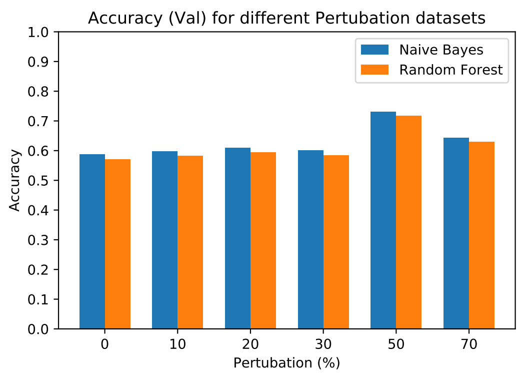
(58.766%) to the baseline based model's accuracy (58.762%). This might be explained by the fact that injecting the additional symptoms with so little probability of being expressed does not significantly change the data distribution when compared to the baseline. For the mean-injected model, there was a slight drop in accuracy (56.993%) compared to the baseline. A logical explanation might be that the additional symptoms injected have the effect of increasing the symptom space without providing much discriminatory information hence making the task of the models slightly more difficult.

In summary, observing the performance of the models on these modified datasets have given a strong intuition and insights into the metric scores obtained. More specifically, it has been established that both the number of symptoms presented by a patient and the relevance of these symptoms aid the model in being able to properly diagnose the condition.

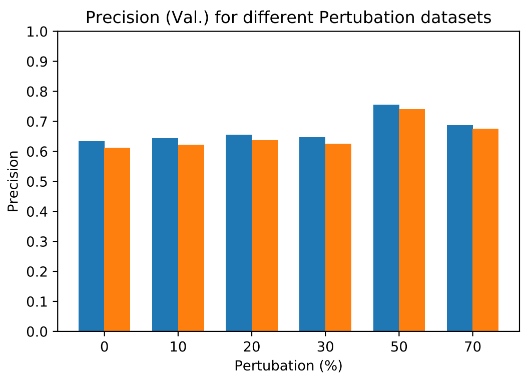
5.3. Baseline Model Performance on Modified Datasets

While the analysis of section 5.2 provided possible explanation for model performance under different scenarios, the main aim of generating these different datasets was to evaluate the performance of the baseline model under these different possible real-world scenarios. To accomplish this, both the baseline Naive Bayes and Random Forest models were evaluated on the test sets of the different cases and comparisons are made with the performance of models trained on specifically on the different non-baseline datasets. In the following discussion, we evaluate the performance of the baseline-trained models on the test sets of the different datasets mimicking real world scenarios. As a benchmark, we use the performance of models trained specifically in these real-world settings. This gives a sort of upper limit on achievable performance.

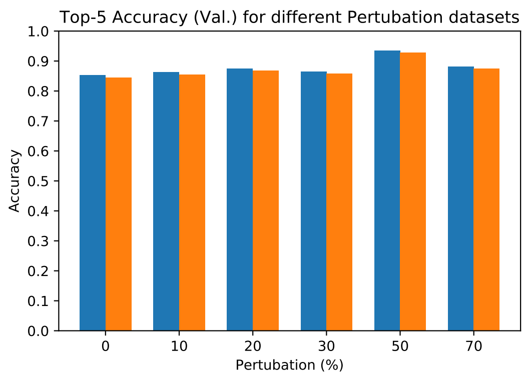
Tables A.1 and A.2 show the performance of the baseline models for Naive Bayes and Random Forest respectively on all the different generated datasets. We see that the baseline models still perform comparatively well compared to the data-specific models in the case of datasets generated with increased minimum number of symptoms. The performance of the baseline based models also increases with an increase in the minimum number of symptoms as show in figure 5.13. The near linear relationship observed in figure 5.9 is also present. The models perform much better in this case with Naive Bayes model achieving 83.6% top-1 accuracy, 87.6% precision and 96.6% top-5 accuracy with a minimum of 5 symptoms. Similarly Random Forest achieves 80.2% top-1 accuracy, 84.2% precision and 95.7% top-5 accuracy. This is despite having being trained on a dataset which contained conditions presenting only one symptom. Again this underscores the importance of identifying as many symptoms as possible in the differential diagnosis flow. When observing the results on the datasets generated by perturbing the condition-symptom probabilistic relationship, it can be seen that for



(a) Accuracy



(b) Precision



(c) Top-5 Accuracy

Figure 5.10: Evaluation Metrics for Models trained on Perturbed Datasets
Note that the perturbed-0% case corresponds to the baseline data i.e. no perturbation

small perturbations the difference between the baseline and the data-specific models is almost negligible. As the perturbations increase, the difference becomes even more noticeable. At 70% perturbation, the baseline based models fall below their performance on the baseline dataset. In the 50% perturbation case, the baseline model still performs reasonably well in comparison. The higher number of symptoms explanation given in section 5.2.2 might be a reason for this relatively good performance despite a high level of perturbation. In the case of the *similarly* injected symptoms and for all three modes of assigning probabilities to the injected symptoms, the performance of the baseline based models was well below their performance on the baseline dataset and also much lower than the performance of the data specific models. The performance on the symptom injected datasets got worse with an increase in the probability with which the similar symptoms were injected. In the max-injected case, the baseline based model achieved an accuracy of 0.099 and 0.1 for Naive Bayes and Random Forest respectively. The Top-5 accuracy was also very low (0.234 and 0.271) for both models. It is fair to reiterate that the data modeling process did not capture a possible similarity between symptoms, therefore injecting the new symptoms presented the models with patterns they had never encountered during training and the higher the probability of the injected symptoms, the greater the divergence of the resulting data from the baseline data, hence the results are not entirely surprising.

5.4. Augmented Model Performance on Modified Datasets

Both the Random Forest and Naive Bayes model were trained on the augmented data which was generated according to section 4.2.5. The performance comparison of these augmented based Naive Bayes and Random Forest models compared to that of their baseline based counterparts on the different generated datasets are shown in the tables A.3 and A.4.

In both tables A.3 and A.4, it is observed that for most of the datasets where the baseline performed relatively good compared to the data specific models there was a drop in performance for the augmented model but this drop was never more than 5%. We also observe significant improvements over the baseline in the

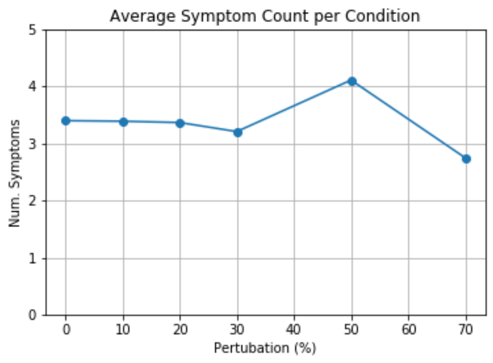
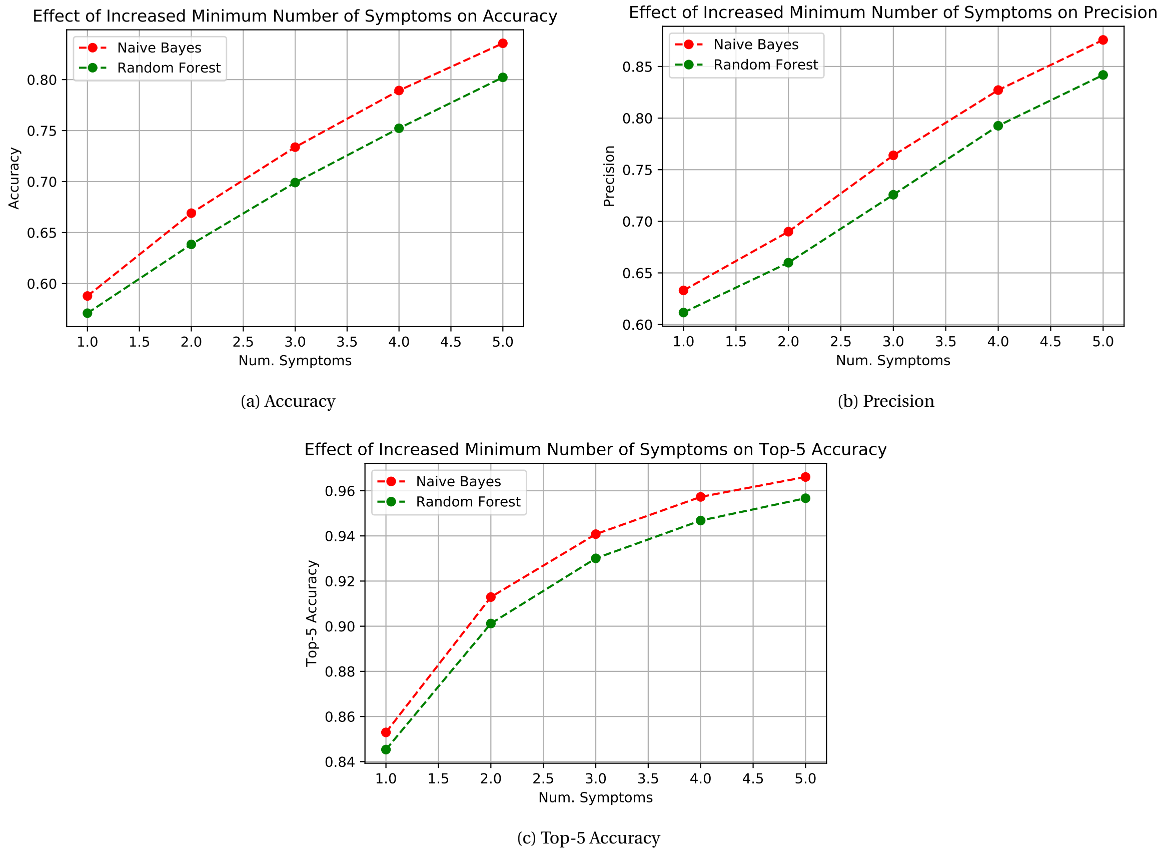


Figure 5.11: Average Expressed Symptom Count per Condition in Perturbed Datasets

case of the symptom injected datasets although the performance, especially in the max-injected case, is still far off from that of the data-specific model. Nonetheless, training on this sort of augmented data does provide some level of robustness.



(a) Accuracy

(b) Precision

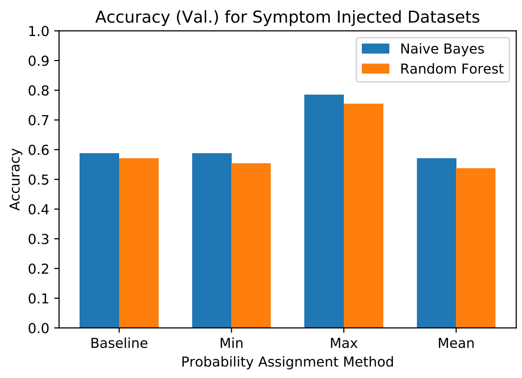
(c) Top-5 Accuracy

Figure 5.13: Performance of Baseline Based Models on Datasets With Increasing Minimum Number of Symptoms

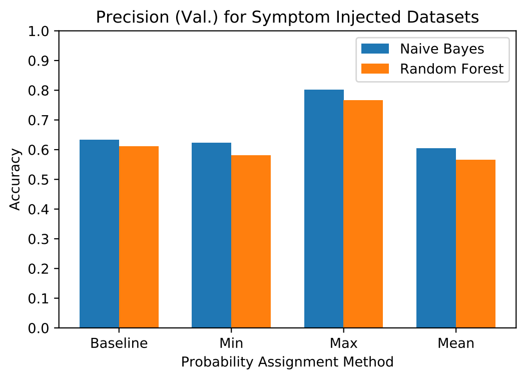
5.5. Evaluation of the Multilayer Perceptron

The multilayer perceptron was trained on the baseline dataset. This trained model was then evaluated on the different generated datasets. The comparison of the MLP with the Naive Bayes and Random Forest models are shown in table A.5

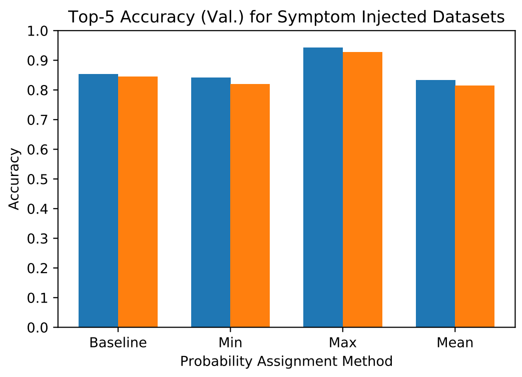
We see from table A.5 that the MLP has a very similar performance to the Naive Bayes model across all metrics. Just like the Naive Bayes model it also consistently out performs the Random Forest model. Previous studies [29] have shown that the MLP approximates a Bayes optimal discriminant function and recalling that



(a) Accuracy



(b) Precision



(c) Top-5 Accuracy

Figure 5.12: Evaluation Metrics for Models trained on Symptom-Injected Datasets

the data generation process as explained in chapter 3 closely resembles a Naive Bayes formulation, this would explain the similarity in performance of the Naive Bayes and the MLP.
Comparing the posterior probability estimates of the MLP model also reveals near identical performance with the Naive Bayes model as might be observed from figure 5.14.

5.6. Summary and Discussion

From the foregoing analysis, it has been shown that both the number of symptoms available per condition and the significance of the presented symptoms make the task of obtaining an accurate differential diagnosis easier for trained models. Also from the results obtained on the baseline dataset, it was observed that a posterior probability estimate threshold could be used in practice to filter out most wrong and weakly-correct predictions.

The models compared showed similar performance. The Naive Bayes and the Multilayer Perceptron produced nearly identical results with respect to the evaluation metrics selected. They also showed similar posterior probability estimates for the Top-1 accurate, Top-5 accurate, Non Top-1 accurate and Non Top-5 accurate cases. The Random Forest in most scenarios performed slightly worse than both the Naive Bayes and the MLP model. The difference in metric scores however were not very significant.
Computationally, the Naive Bayes is a much simpler model to train but the Random Forest produces better posterior probability estimates which perhaps makes it a more suitable candidate for the task of differential diagnosis.

While mimicking possible real-world scenarios, it was also observed that the performance of the models was significantly reduced when symptoms not usually associated in a condition were presented and when the symptom-condition probabilistic relationship was significantly different than that on which the models were trained on.

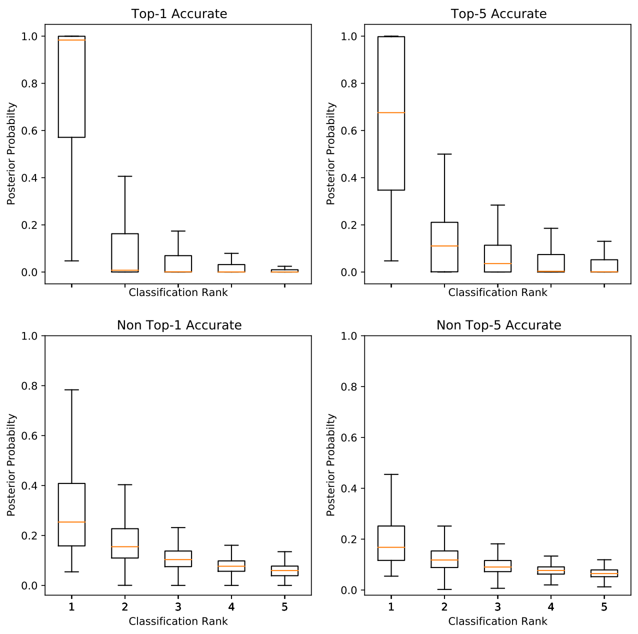


Figure 5.14: Comparing Posterior Probability Estimates for MLP

When the Naive Bayes and Random Forest models were trained on an augmented dataset, there was an improvement in the performance on the real-world scenario datasets although the obtained performance was still less than the performance of models trained specifically on generated datasets. The MLP model was not trained on the augmented dataset but judging by the similarity in performance to the Naive Bayes model, it is logical to expect similar results. These result highlights the important role of the data (or in this particular case the probabilistic relationship between patient's demography, symptoms and conditions) in the success of these models in a real-world setting.

An interesting discussion ensues from the results of the augmented based model, one which might strengthen the case for synthetic data in this type of medical application. All the datasets which were generated were based - as mentioned earlier - on the population distribution of Massachusetts in the United States of America and all modifications made were solely targeted at modifying the probabilistic relationship between symptoms and conditions. From a medical perspective, there is known relationship between patient's race and the prevalence of certain conditions e.g. certain races [30] being more likely to have diabetes and its resulting symptoms or a dependence on a combination of race and age etc. The nature of these dependencies also varies depending on the population being studied. In many cases, real data obtained for similar studies are gotten from a specific country in a specific hospital/settings and might not be representative of the *general* case. However, medical academic literature contains several studies across different populations and hospitals which often summarize the important relationships between a patient's demography, symptoms, and conditions. An argument can then possibly be made for a potential superiority in results of an augmented data approach i.e. combining datasets generated using these summary statistics when compared to real data obtained from just one location. Also, more sophisticated probabilistic models can be developed using data from these different studies and sources to further improve expected performance in a real world setting.

6 Conclusion and Recommendations

To conclude, we revisit the problem statement and research questions posed in chapter 1. Recommendations for possible future research are also discussed.

6.1. Problem Statement and Research Questions Revisited

Suitability of Machine Learning Techniques to the Differential Diagnosis Problem From the results in chapter 5, it was shown that the models trained were able to capture the probabilistic relationship between conditions, symptoms, and patient's demography making them suitable to the task of obtaining a differential diagnoses. The Naive Bayes model achieved a top-1 accuracy of 58.8%, a precision of 63.3% and a top-5 accuracy of 85.3%. The MLP model achieved a 58.8% top-1 accuracy, 63.0% precision and a top-5 accuracy of 85.5%. For the Random Forest, the top-1 accuracy was 57.1%, a precision of 61.2% and a top-5 accuracy of 85.5%. Since the most important task in a differential diagnoses is obtaining a valid list of possible symptoms, the top-5 accuracy is a good measure of the suitability of these models to the task and judging by the relatively high scores on this metric, it is logical to conclude that the models are up to the task. It was also shown that the number of symptoms presented per condition had an almost linear relationship with the achieved metric scores. When the models were evaluated on data with at least 5 symptoms per condition, the accuracy was between 80.2 and 83.3%, precision lay between 84.2 and 87.4% while top-5 accuracy was between 95.7 and 96.6% across all evaluated models.
A qualitative comparison of the top 5 co-occurring predictions for selected conditions showed an agreement between the models. These co-occurring conditions also belonged to the same condition group as the selected conditions i.e they affect related organs and also present similar symptoms.
A qualitative analysis of the top 5 misclassification for selected conditions also revealed that these top 5 conditions were usually very similar to the actual condition and as such would present similar symptoms highlighting the difficulty the models face in distinguishing similar conditions based on symptoms alone.

Model Performance on Possible Real World Scenarios

In this project, especially owing to the use of synthetic data, it was desired to also synthesize possible real-world scenarios and observe how the trained models behave in these cases. Results from chapter 5 show that the performance of the models start to degrade when the synthesized scenarios result in datasets which deviate significantly from the baseline dataset. As observed earlier, this was especially true when the models were confronted with cases that had symptoms not associated with the causal condition (as captured in the baseline dataset). Even when these *new* symptoms were injected with minimal probability, there was a reduction in accuracy of about 16% in the performance when compared with models trained specifically on the modified data.
When the probabilistic relationship between symptoms and conditions were *perturbed* - mimicking a change in the relevance of the symptoms it was seen that the greater the perturbation, the larger the difference in performance between the baseline based models and the data specific models. When the probabilities of these perturbations were restricted to within 10% of the baseline values, there was hardly any change in accuracy (less than 1%) while for a 70% perturbation there was a reduction in accuracy of about 31% for the Naive Bayes model and 35% for the Random Forest.
These results indicate that the performance of the models in a real world setting depends a lot on the similarity between real world data and the synthetically generated data. The larger the deviation, the larger the difference in expected performance.

It was also shown that training models on data augmented to account for these deviations produced much better results than the baseline - even though less than ideal. In the case of the 70% perturbed data, the augmented model had a reduction in performance of 28.6% for Naive Bayes and 20.67% for Random Forest - which are better than the baseline models. This suggests that if possible deviations from a real world setting can be anticipated, data augmentation techniques can help improve the robustness of the trained models.

6.2. Recommendations

Improving Symptom Information

As mentioned, distinguishing conditions based on just the symptoms presented is a difficult task for the trained models and understandably so - similar conditions have similar symptoms. Enriching the data with extra information such as the nature of the symptom, its duration, what situations or activities excite the symptom, etc might also provide additional discriminatory information and possibly yield better results. Some work has already been done in this regard on a small set of conditions and symptoms and initial results suggest that these additional features do make it easier for models to distinguish between conditions.

Investigating More Metrics for Differential Diagnosis Accuracy

In this project, the top-5 accuracy was selected as the main metric of interest when evaluating a differential diagnosis prediction. A qualitative analysis also showed that predicted top 5 conditions - when accurate - typically affect the same body parts. It is not inconceivable that two conditions might affect different body parts and present a subset of similar conditions.Hence a body-part based similarity test would not be a good evaluation metric in such a case. To the best of our knowledge, there is no gold standard for evaluating a differential diagnosis - even in practice. This provides an interesting angle for further research.

Model Evaluation in a Real World Setting

The ultimate test of any machine learning model is its performance in a real-world setting. In the planning of this project, Medvice - our industry collaborators - were running data gathering pilot programs with partner hospitals and the trained models would be evaluated on the data. However, due to the corona-virus pandemic, those efforts had to be suspended. At the time of writing, the lock-down and restrictions placed on movement and business activities are being lifted and the plan is to resume the data gathering pilots and use the data obtained for a more realistic evaluation of the model's efficacy.

Further Exploration of Deep Learning Techniques

In this project, more focus was placed on the Naive Bayes and Random Forest models. Results from the MLP model were very similar to that of the Naive Bayes but there is still more room for optimization e.g. hyperparameter optimization, improved architecture, e.t.c. An interesting direction is the use of Autoencoders [31, 32] to learn latent representations in the dataset and use this learned representation as input to other models. This might also help improve the robustness of the trained models.

A
Model Comparison Data

The table below compares the Naive Bayes and Random Forest for selected conditions and shows the top 5 mis-classification for each of these conditions.

A.1. Baseline Based Model Performance on Generated Datasets

Dataset	Accuracy		Precision		Top-5	
	Base Model	Data Model	Base Model	Data Model	Base Model	Data Model
Baseline	0.588	0.588	0.633	0.633	0.853	0.853
Min. 2 Symptoms	0.669	0.690	0.690	0.706	0.913	0.925
Min. 3 Symptoms	0.734	0.789	0.764	0.798	0.941	0.964
Min. 4 Symptoms	0.789	0.869	0.827	0.876	0.957	0.983
Min. 5 Symptoms	0.836	0.914	0.876	0.921	0.966	0.989
Mean Injected	0.312	0.588	0.380	0.623	0.563	0.842
Max Injected	0.099	0.784	0.207	0.800	0.234	0.942
Min Injected	0.480	0.571	0.515	0.605	0.754	0.834
Perturbed-10%	0.594	0.597	0.642	0.641	0.861	0.863
Perturbed-20%	0.600	0.610	0.653	0.656	0.867	0.875
Perturbed-30%	0.549	0.601	0.599	0.647	0.826	0.866
Perturbed-50%	0.696	0.731	0.724	0.756	0.919	0.935
Perturbed-70%	0.441	0.643	0.507	0.687	0.726	0.882

Table A.1: Naive Bayes Model Performance on Test Set of Different Generated Datasets
Baseline Model is compared with performance of Model trained specifically on each generated dataset.

Dataset	Accuracy		Precision		Top-5	
	Base Model	Data Model	Base Model	Data Model	Base Model	Data Model
Baseline	0.571	0.571	0.612	0.612	0.845	0.845
Min. 2 Symptoms	0.638	0.679	0.660	0.691	0.901	0.922
Min. 3 Symptoms	0.699	0.785	0.726	0.790	0.930	0.961
Min. 4 Symptoms	0.752	0.866	0.793	0.870	0.947	0.981
Min. 5 Symptoms	0.802	0.912	0.842	0.914	0.957	0.988
Mean Injected	0.286	0.554	0.340	0.582	0.560	0.820
Max Injected	0.100	0.754	0.177	0.766	0.271	0.928
Min Injected	0.451	0.538	0.478	0.567	0.743	0.815
Perturbed-10%	0.580	0.581	0.623	0.620	0.855	0.855
Perturbed-20%	0.589	0.596	0.636	0.637	0.865	0.868
Perturbed-30%	0.533	0.585	0.578	0.626	0.818	0.858
Perturbed-50%	0.679	0.718	0.705	0.741	0.915	0.929
Perturbed-70%	0.407	0.629	0.465	0.673	0.709	0.874

Table A.2: Random Forest Model Performance on Test Set of Different Generated Datasets
Baseline Model is compared with performance of Model trained specifically on each generated dataset.

A.2. Effect of Data Augmentation on Model Performance on Generated Datasets

Dataset	Accuracy		Precision		Top-5	
	Base Model	Aug. Model	Base Model	Aug. Model	Base Model	Aug. Model
Baseline	0.588	0.562	0.633	0.618	0.853	0.835
Min. 2 Symptoms	0.669	0.656	0.690	0.680	0.913	0.902
Min. 3 Symptoms	0.734	0.752	0.764	0.765	0.941	0.944
Min. 4 Symptoms	0.789	0.821	0.827	0.835	0.957	0.966
Min. 5 Symptoms	0.836	0.866	0.876	0.887	0.966	0.977
Mean Injected	0.312	0.531	0.380	0.556	0.563	0.796
Max Injected	0.099	0.488	0.207	0.550	0.234	0.735
Min. Injected	0.480	0.553	0.515	0.588	0.754	0.822
Perturbed-10%	0.594	0.570	0.642	0.628	0.861	0.844
Perturbed-20%	0.600	0.578	0.653	0.640	0.867	0.854
Perturbed-30%	0.549	0.538	0.599	0.601	0.826	0.823
Perturbed-50%	0.696	0.687	0.724	0.719	0.919	0.916
Perturbed-70%	0.441	0.459	0.507	0.536	0.726	0.742

Table A.3: Comparing Naive Bayes Baseline and Augmented Model Performance on Different Generated Datasets
As before, comparisons are made on the test set.

Dataset	Accuracy		Precision		Top-5	
	Base Model	Aug. Model	Base Model	Aug. Model	Base Model	Aug. Model
Baseline	0.571	0.557	0.612	0.602	0.845	0.835
Min. 2 Symptoms	0.638	0.654	0.660	0.669	0.901	0.906
Min. 3 Symptoms	0.699	0.749	0.726	0.757	0.930	0.946
Min. 4 Symptoms	0.752	0.830	0.793	0.838	0.947	0.970
Min. 5 Symptoms	0.802	0.888	0.842	0.895	0.957	0.983
Mean Injected	0.286	0.454	0.340	0.469	0.560	0.735
Max Injected	0.100	0.574	0.177	0.584	0.271	0.794
Min. Injected	0.451	0.507	0.478	0.528	0.743	0.789
Perturbed-10%	0.580	0.569	0.623	0.616	0.855	0.847
Perturbed-20%	0.589	0.580	0.636	0.631	0.865	0.858
Perturbed-30%	0.533	0.535	0.578	0.584	0.818	0.819
Perturbed-50%	0.679	0.683	0.705	0.710	0.915	0.916
Perturbed-70%	0.407	0.499	0.465	0.563	0.709	0.784

Table A.4: Comparing Random Forest Baseline and Augmented Model Performance on Different Generated Datasets

A.3. Performance of MLP on Generated Datasets

	Accuracy			Precision			Top-5		
Dataset	NB	RF	MLP	NB	RF	MLP	NB	RF	MLP
Baseline	0.588	0.571	0.588	0.633	0.612	0.630	0.853	0.845	0.855
Min. 2 Symptoms	0.669	0.638	0.662	0.690	0.660	0.691	0.913	0.901	0.911
Min. 3 Symptoms	0.734	0.699	0.729	0.764	0.726	0.763	0.941	0.930	0.941
Min. 4 Symptoms	0.789	0.752	0.785	0.827	0.793	0.827	0.957	0.947	0.955
Min. 5 Symptoms	0.836	0.802	0.833	0.876	0.842	0.874	0.966	0.957	0.966
Mean Injected	0.312	0.286	0.333	0.380	0.340	0.392	0.563	0.560	0.585
Max Injected	0.099	0.100	0.127	0.207	0.177	0.228	0.234	0.271	0.275
Min Injected	0.480	0.451	0.492	0.515	0.478	0.521	0.754	0.743	0.766
Perturbed-10%	0.594	0.580	0.595	0.642	0.623	0.639	0.861	0.855	0.863
Perturbed-20%	0.600	0.589	0.601	0.653	0.636	0.650	0.867	0.865	0.870
Perturbed-30%	0.549	0.533	0.551	0.599	0.578	0.598	0.826	0.818	0.828
Perturbed-50%	0.696	0.679	0.695	0.724	0.705	0.721	0.919	0.915	0.922
Perturbed-70%	0.441	0.407	0.450	0.507	0.465	0.509	0.726	0.709	0.738

Table A.5: Comparing Baseline MLP, Naive Bayes and Random Forest on Different Generated Datasets

B

Qualitative Evaluation of Model Predictions

B.1. ICD-10 Code Blocks

The table below shows the grouping for ICD-10 classification codes.

Chapter	Block	Title
I	A00–B99	Certain infectious and parasitic diseases
II	C00–D48	Neoplasms
III	D50–D89	Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism
IV	E00–E90	Endocrine, nutritional and metabolic diseases
V	F00–F99	Mental and behavioural disorders
VI	G00–G99	Diseases of the nervous system
VII	H00–H59	Diseases of the eye and adnexa
VIII	H60–H95	Diseases of the ear and mastoid process
IX	I00–I99	Diseases of the circulatory system
X	J00–J99	Diseases of the respiratory system
XI	K00–K93	Diseases of the digestive system
XII	L00–L99	Diseases of the skin and subcutaneous tissue
XIII	M00–M99	Diseases of the musculoskeletal system and connective tissue
XIV	N00–N99	Diseases of the genitourinary system
XV	O00–O99	Pregnancy, childbirth and the puerperium
XVI	P00–P96	Certain conditions originating in the perinatal period
XVII	Q00–Q99	Congenital malformations, deformations and chromosomal abnormalities
XVIII	R00–R99	Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified
XIX	S00–T98	Injury, poisoning and certain other consequences of external causes
XX	V01–Y98	External causes of morbidity and mortality
XXI	Z00–Z99	Factors influencing health status and contact with health services
XXII	U00–U99	Codes for special purposes

Table B.1: ICD-10 Code Blocks

B.2. Confusion Comparison for Selected Models

The table below compares the Naive Bayes Random Forest and MLP for selected conditions and shows the top 5 mis-classifications for each of the selected conditions.

Condition	Naive Bayes	Random Forest	MLP
Acute Bronchitis (J20.9)	COPD (J44.9) Acute Bronchospasm (J98.01) Common Cold (J00) Asthma (J45.9) Croup (J05.0)	COPD (J44.9) Acute Bronchospasm (J98.01) Common Cold (J00) Bronchiectasis (J47) Asthma (J45.9)	COPD (J44.9) Acute Bronchospasm (J98.01) Common Cold (J00) Asthma (J45.9) Bronchiectasis (J47)
Acute Sinusitis (J01.9)	Chronic Sinusitis (J32.9) Nose Disorder (J34) Common Cold (J00) Abscess of nose (J34.0) Strep. Throat (J02.0)	Chronic Sinusitis (J32.9) Nose Disorder (J34) Common Cold (J00) Abscess of nose (J34.0) Nasal polyp (J33.9)	Chronic sinusitis (J32.9) Nose disorder (J34) Common Cold (J00) Abscess of nose (J34.0) Abscess of the pharynx (J39.1)
Asthma (J45.9)	COPD (J44.9) Acute bronchospasm (J98.01) ARDS (J80) Croup (J05.0) Pulmonary fibrosis (J84.10)	ARDS (J80) Pulmonary congestion (J81) Acute bronchospasm (J98.01) Acute bronchiolitis (J21.9) Pulmonary fibrosis (J84.10)	COPD (J44.9) ARDS (J80) Acute bronchospasm (J98.01) Croup (J05.0) Pulmonary congestion (J81)

Table B.2: Confusion Comparison for Selected Respiratory Infections
ICD-10 codes are shown as well.

Condition	Naive Bayes	Random Forest	MLP
Acute pancreatitis (K85.9)	Cholecystitis (K81.9) Gallstone (K80) Crohn disease (K50.9) Gallbladder disease (K82.9) Persistent vomiting (R11)	Cholecystitis (K81.9) Gallstone (K80) Chronic pancreatitis (K86.1) Crohn disease (K50.9) Gallbladder disease (K82.9)	Cholecystitis (K81.9) Gallstone (K80) Persistent Vomiting (R11) Crohn disease (K50.9) Gallbladder disease (K82.9)
Appendicitis (K37)	Diverticulitis (K57.3) Peritonitis (K65.9) Crohn disease (K50.9) Noninfectious gastroenteritis (K52.9) Persistent vomiting (R11)	Diverticulitis (K57.3) Crohn disease (K50.9) Persistent Vomiting (R11) Noninfectious gastroenteritis (K52.9) Peritonitis (K65.9)	Diverticulitis (K57.3) Ovarian cyst (N83.2) Peritonitis (K65.9) Crohn disease (K50.9) Persistent Vomiting (R11)
Chronic constipation (K59.00)	Intestinal obstruction (K56.6) Hirschsprung disease (Q43.1) Crohn disease (K50.9) Ileus (K56.7) Diverticulosis (K57)	Hirschsprung disease (Q43.1) Intestinal obstruction (K56.6) Diverticulosis (K57) Hashimoto thyroiditis (E06.3) Crohn disease (K50.9)	Hirschsprung disease (Q43.1) Intestinal obstruction (K56.6) Crohn disease (K50.9) Gallbladder cancer (C23) Diverticulosis (K57)

Table B.3: Confusion Comparison for Selected Gastro-Intestinal Infections

Condition	Naive Bayes	Random Forest	MLP
Cystitis (N30.9)	Benign blood in urine (R31) Urethral disorder (N36.9) Bladder disorder (N32.9) Urge incontinence (N39.4) Urethral stricture (N35.9)	Benign blood in urine (R31) Urethral disorder (N36.9) Bladder disorder (N32.9) Urge incontinence (N39.4) Urethral stricture (N35.9)	Benign blood in urine (R31) Urethral disorder (N36.9) Bladder disorder (N32.9) Urge incontinence (N39.4) Urethral stricture (N35.9)
Pyelonephritis (N12)	Kidney stone (N20.0) Urinary tract infection (N39.0) Peritonitis (K65.9) Hydronephrosis (N13.3) Benign kidney cyst (N28.1)	Kidney stone (N20.0) Urinary tract infection (N39.0) Hydronephrosis (N13.3) Peritonitis (K65.9) Benign kidney cyst (N28.1)	Kidney stone (N20.0) Urinary tract infection (N39.0) Peritonitis (K65.9) Hydronephrosis (N13.3) Benign kidney cyst (N28.1)
Urethritis (N34.1)	Urethral stricture (N35.9) Gonorrhea (A54.9) Chlamydia (A56) Benign blood in urine (R31) Urethral disorder (N36.9)	Urethral stricture (N35.9) Cystitis (N30.9) Gonorrhea (A54.9) Phimosis (N47) Benign blood in urine (R31)	Urethral stricture (N35.9) Gonorrhea (A54.9) Benign blood in urine (R31) Phimosis (N47) Prostatitis (N41.9)

Table B.4: Confusion Comparison for Selected Urinary Tract Infections
ICD-10 codes are shown as well

B.3. Comparing Top-5 Model Predictions

For the same set of conditions as above, this table shows the top 5 most commonly occurring conditions given that the prediction is Top-5 accurate. In other words, if the actual condition is in the top-5 predictions, the tables below show the top 5 other conditions which are included in the differential diagnosis.

Condition	Naive Bayes	Random Forest	MLP
Acute Bronchitis (J20.9)	COPD (J44.9) Acute bronchospasm (J98.01) Asthma (J45.9) Pneumonia (J18.9) Interstitial lung disease (J84.9)	COPD (J44.9) Acute bronchospasm (J98.01) Asthma (J45.9) Pneumonia (J18.9) Poisoning due to gas (Y17.9)	COPD (J44.9) Acute bronchospasm (J98.01) Asthma (J45.9) Pneumonia (J18.9) Interstitial lung disease (J84.9)
Acute sinusitis (J01.9)	Chronic sinusitis (J32.9) Nose disorder (J34) Abscess of nose (J34.0) Hay fever (J30.1) Common cold (J00)	Chronic sinusitis (J32.9) Nose disorder (J34) Hay fever (J30.1) Abscess of nose (J34.0) Common cold (J00)	Chronic sinusitis (J32.9) Nose disorder (J34) Hay fever (J30.1) Abscess of nose (J34.0) Abscess of the pharynx (J39.1)
Asthma (J45.9)	Acute bronchospasm (J98.01) COPD (J44.9) ARDS (J80) Croup (J05.0) Acute bronchiolitis (J21.9)	COPD (J44.9) Acute bronchospasm (J98.01) ARDS (J80) Croup (J05.0) Acute bronchiolitis (J21.9)	COPD (J44.9) Acute bronchospasm (J98.01) ARDS (J80) Croup (J05.0) Acute bronchitis (J21.9)

Table B.5: Top 5 Commonly Included Predictions for Selected Respiratory Infections
ICD-10 codes included

Condition	Naive Bayes	Random Forest	MLP
Acute pancreatitis (K85.9)	Cholecystitis (K81.9) Gallstone (K80) Choledocholithiasis (K80.5) Gallbladder disease (K82.9) Chronic pancreatitis (K86.1)	Choledocholithiasis (K80.5) Cholecystitis (K81.9) Gallstone (K80) Gallbladder disease (K82.9) Chronic pancreatitis (K86.1)	Gallstone (K80) Choledocholithiasis (K80.5) Cholecystitis (K81.9) Gallbladder disease (K82.9) Chronic pancreatitis (K86.1)
Appendicitis (K37)	Diverticulitis (K57.3) Peritonitis (K65.9) Ileus (K56.7) Noninfectious gastroenteritis (K52.9) Infectious gastroenteritis (A09.0)	Diverticulitis (K57.3) Peritonitis (K65.9) Ileus (K56.7) Ovarian cyst (N83.2) Ischemia of the bowel (K55.9)	Diverticulitis (K57.3) Peritonitis (K65.9) Ileus (K56.7) Pyelonephritis (N12) Infectious gastroenteritis (A09.0)
Chronic constipation (K59.00)	Diverticulosis (K57) Intestinal obstruction (K56.6) Ileus (K56.7) Ischemia of the bowel (K55.9) Hirschsprung disease (Q43.1)	Intestinal obstruction (K56.6) Diverticulosis (K57) Ileus (K56.7) Hirschsprung disease (Q43.1) Ischemia of the bowel (K55.9)	Diverticulosis (K57) Intestinal obstruction (K56.6) Ischemia of the bowel (K55.9) Ileus (K56.7) Colonic polyp (K63.5)

Table B.6: Top 5 Commonly Included Predictions for Selected Digestive Infections

Condition	Naive Bayes	Random Forest	MLP
Cystitis (N30.9)	Benign blood in urine (R31) Urinary tract infection (N39.0) Urethral disorder (N36.9) Bladder disorder (N32.9) Urinary tract obstruction (N13.9)	Benign blood in urine (R31) Urethral disorder (N36.9) Urinary tract infection (N39.0) Bladder disorder (N32.9) Urge incontinence (N39.4)	Benign blood in urine (R31) Urinary tract infection (N39.0) Urethral disorder (N36.9) Bladder disorder (N32.9) Urinary tract obstruction (N13.9)
Pyelonephritis (N12)	Urinary tract infection (N39.0) Kidney stone (N20.0) Peritonitis (K65.9) Hydronephrosis (N13.3) Polycystic kidney disease (Q61.3)	Urinary tract infection (N39.0) Kidney stone (N20.0) Peritonitis (K65.9) Hydronephrosis (N13.3) Benign kidney cyst (N28.1)	Urinary tract infection (N39.0) Kidney stone (N20.0) Hydronephrosis (N13.3) Peritonitis (K65.9) Pelvic inflammatory disease (N73.9)
Urethritis (N34.1)	Chlamydia (A56) Gonorrhea (A54.9) Cystitis (N30.9) Prostatitis (N41.9) Urethral disorder (N36.9)	Chlamydia (A56) Gonorrhea (A54.9) Cystitis (N30.9) Prostatitis (N41.9) Benign blood in urine (R11)	Chlamydia (A56) Gonorrhea (A54.9) Prostatitis (N41.9) Balanitis (N48.1) Phimosis (N47)

Table B.7: Top 5 Commonly Included Predictions for Selected Genitourinary Infections

Bibliography

[1] Hardeep Singh, Ashley ND Meyer, and Eric J Thomas. The frequency of diagnostic errors in outpatient care: estimations from three large observational studies involving us adult populations. *BMJ Qual Saf*, 23(9):727–731, 2014.

[2] Micheal B. First. *DSM-5: A Handbook of Differential Diagnosis*. American Psychiatric Publishing, 1 edition, 2013.

[3] Milford Fulop. Teaching differential diagnosis to beginning clinical students. *American Journal of Medicing*, 1995.

[4] Olga Kostopoulou. Early diagnostic suggestions improve accuracy of gps. *British Journal of General Practice*, pages 49–54, January 2015.

[5] Eta Berner. Clinical decision support systems: State of the art. *Agency for Healthcare Research and Quality, U.S. Department of Health and Human Services*, 2009.

[6] Michael Pazzani Pedro Domingos. Beyond independence:conditions for the optimality of the simple bayesian classifier. *Machine Learning*, 29:103–130, 1997.

[7] Harry Zhang. The optimality of naive bayes. In *FLAIRS Conference*, 2004.

[8] Philip H. Swain and Hans Hauska. The decision tree classifier: Design and potential. *IEEE TRANSACTIONS ON GEOSCIENCE ELECTRONICS*, GE-15(3):142–148, July 1997.

[9] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[11] Julia Winn Latanya Sweeney, Akua Abu. Identifying participants in the personal genome project by name. `https://dataprivacylab.org/projects/pgp/1021-1.pdf`, April 2013.

[12] Edward Y. Chang Hao-Cheng Kao, Kai-Fu Tang. Context-aware symptom checking for disease diagnosisusing hierarchical reinforcement learning. *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 2305–2313, 2018.

[13] Markus Zlabinger, Sebastian Hofstätter, Navid Rekabsaz, and Allan Hanbury. DSR: A Collection for the evaluation of graded disease-symptom relations. *Advances in Information Retrieval*, page 433–440, 2020.

[14] AHEAD Research Inc. Symcat: Symptom-based,computer assisted triage. `http://www.symcat.com/`, 2020.

[15] Uri Kartoun. Advancing informatics with electronic medical records bots (emrbots). *Software Impacts*, 2019.

[16] Jason Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association*, 25(3):230–238, 2018.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[18] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks, 2017.

[19] Paul N. Bennet. Assessing the calibration of naive bayes' posterior estimates. Technical report, Carnegie Mellon University, September 2000.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015.

[21] Quantum and Computer Engineering TU Delft. QCE Cluster. `http://qce-it-infra.ewi.tudelft.nl/qce_cluster.html`, 2020. Accessed: 2020-08-05.

[22] SchedMD. Slurm. `https://slurm.schedmd.com/overview.html`, 2020. Accessed: 2020-08-05.

[23] Syslabs.io. Singularity. `https://sylabs.io/guides/3.6/user-guide/introduction.html`, 2020. Accessed: 2020-08-05.

[24] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[25] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

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

[27] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9(3):90–95, 2007.

[28] LF Projects LLC. Mlflow. `https://mlflow.org/`. Accessed: 2020-08-05.

[29] Dennis W Ruck, Steven K Rogers, Matthew Kabrisky, Mark E Oxley, and Bruce W Suter. The multilayer perceptron as an approximation to a bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1(4):296–298, 1990.

[30] Yiling J. Cheng, Alka M. Kanaya, Maria Rosario G. Araneta, Sharon H. Saydah, Henry S. Kahn, Edward W. Gregg, Wilfred Y. Fujimoto, and Giuseppina Imperatore. Prevalence of Diabetes by Race and Ethnicity in the United States, 2011-2016. *JAMA*, 322(24):2389–2398, 12 2019.

[31] Riccardo Miotto, Li Li, and Joel T. Dudley. Deep learning to predict patient future diseases from the electronic health records. In Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello, editors, *Advances in Information Retrieval*, pages 768–774, Cham, 2016. Springer International Publishing.

[32] Weinan Zhang, Tianming Du, and Jun Wang. Deep learning over multi-field categorical data. *Advances in Information Retrieval*, page 45–57, 2016.