Predicting Diabetes Readmission

Sailesh Kaveti
University of Rochester
Rochester, NY
skaveti@u.rochester.edu

Julian Travis
University of Rochester
Rochester, NY
jtravis3@u.rochester.edu

ABSTRACT

This paper provides an analysis of a diabetes dataset in order to find a model that accurately predicts whether a diabetic individual will be readmitted to a hospital given their condition. With respect to the dataset, we chose the Diabetes 130-US Hospitals for years 1999-2008 Data Set from the UCI Machine Learning Repository. After investigation, we determined that the current state of the dataset would not be very valuable in our classification problem, so multiple data cleaning, transformation, and selection tasks needed to be used in order to better apply our classifications models. For example, some attributes were simply not valuable due to the number of missing values, while others were too categorical for our models. For the classification models, we chose a Naive Bayes Classifier to provide a baseline model, followed by Logistic Regression, as well as Random Forest, AdaBoost, and Neural Network classification for more accurate results. This project was completed for CSC 240: Data Mining, at the University of Rochester in the Spring of 2019. [3]

KEYWORDS

Data Mining, Data Cleaning, Diabetes Prediction, Classification, Random Forest, Naive Bayes, Neural Network

ACM Reference Format:
Sailesh Kaveti and Julian Travis. 2019. Predicting Diabetes Readmission. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages.

1 INTRODUCTION

In 2017, 425 million people had diabetes worldwide. More specifically, in the United States, more than 100 million people suffer from diabetes or prediabetes, the latter of which is a condition that can lead to type 2 diabetes within five years if not treated properly. In addition, the CDC reports that diabetes is the seventh leading cause of death in the U.S. Diabetes becomes more common is older demographics, as 25 percent of people older than 65 suffer from diabetes. In addition, people with diabetes are at increased risk of various different serious health complications including vision loss, heart disease, stroke, kidney failure, amputation, or even premature death. As a result, it becomes clear why we would be interested in trying to predict whether someone would be readmitted into the hospital for a health issue given their condition. More importantly, if we could even predict the time frame within which the patient would be readmitted, doctors can take precautions when scheduling appointments or monitoring their patients. [1]

Our dataset contains information about thousands of patients, with multiple attributes that can provide valuable information such as whether they have been prescribed a certain medication, their levels for a certain test, or even the number of times that they have been to the hospital before. All of these metrics, as well as many others, will be explored in a later section to determine their value in our classification problem. It is important to have a very thorough understanding of these attributes in order to study their effects on diabetic readmission. As implied earlier, this dataset encourages a classification solution in order to predict the class that a row in the dataset in a part of. The three classes that represent readmission are '<30', '>30', and 'No', which represent readmitted within 30 days, readmitted after 30 days, and not readmitted at all respectively. We wanted to find a model that could properly process clean data to correctly predict the class. With this in mind, we followed the following process to arrive at a solution to accurately predict the class of a row: [3]

(1) Data Cleaning
(2) Data Transformation
(3) Feature Selection
(4) Apply the Naive Bayes Classifier
(5) Apply Logistic Regression
(6) Apply Random Forest Classification
(7) Apply AdaBoost Classification
(8) Apply Neural Network Classification
(9) Select the model with the highest accuracy

For reference, the libraries used over the course of the course of the project include seaborn, numpy, pandas, sklearn and matplotlib.

2 DEFINING THE CLASSIFICATION PROBLEM

As mentioned earlier, the Diabetes 130-US Hospitals for years 1999-2008 Data Set has three classes: '<30', '>30', 'No', which highlight the time frame where the patient is readmitted. However, it is important to also note that the members of the '<30' class are at the most risk. This is because their condition is likely serious enough that they revisit the hospital within 30 days of their initial visit. As a result, it will be of interest to reclassify the dataset into '<30' and 'No' where '<30' is the initial '<30' and 'No' is composed of both '>30' and 'No' in the original classification. For the sake of brevity, we will refer to the original assignment of class as *readmitted_1* and the new binary classification as *readmitted_2*. The five models discussed earlier will be applied and explored on both *readmitted_1* and *readmitted_2* [3].

3 UNDERSTANDING AND PROCESSING THE DIABETES DATA SET

This section will discuss the steps that we had to take in order to better understand and process the Diabetes 130-US Hospitals for years 1999-2008 Data Set so that our models could be applied intelligently and accurately. The dataset represents 10 years of data of clinical care at 130 US Hospitals and integrated delivery networks.

The dataset was extracted from the database for encounters that satisfied the following criteria:

(1) It is an inpatient encounter (hospital admission)
(2) It is a diabetic encounter, that is, one during which any kind of diabetes was entered to the system as a diagnosis
(3) The length of stay was at least one day and at most fourteen days
(4) Labratory tests must have been performed during the encounter
(5) Medications were administered at some point during the encounter

With that in mind, we can go on to better understand the attributes [**?**].

3.1 Attributes and Classes

The Diabetes 130-US Hospitals for years 1999-2008 Data Set is a mutlivariate dataset that initially contains 101765 instances as well as 50 total attributes, one of which represents the class. Even though the dataset was extracted from a database, it is not entirely clean and processed, meaning that there are many missing values. In addition, there is a lot of variation in the types of data that exist in dataset, ranging from unranked and ranked categorical data to continuous data. The attributes in the dataset are as follows [2]:

Attribute	Description
Encounter ID	Unique identifier of an encounter
Patient Number	Unique identifier of a patient
Race	Caucasian, Asian, African American, Hispanic, and other
Gender	male, female, and unknown/invalid
Age	Grouped in 10-year intervals: [0, 10), [10, 20), …, [90, 100)
Weight	Weight in pounds
Admission Type	Integer identifier corresponding to 9 distinct values, for example, emergency, urgent, elective, newborn, and not available
Discharge disposition	Integer identifier corresponding to 29 distinct values, for example, discharged to home, expired, and not available
Admission source	Integer identifier corresponding to 21 distinct values, for example, physician referral, emergency room, and transfer from a hospital
Time in hospital	Integer number of days between admission and discharge
Payer code	Integer identifier corresponding to 23 distinct values, for example, Blue Cross/Blue Shield, Medicare, and self-pay
Medical Specialty	Integer identifier of a specialty of the admitting physician, corresponding to 84 distinct values, for example, cardiology, internal medicine, family/general practice, and surgeon
Number of lab procedures	Number of lab tests performed during the encounter
Number of procedures	Number of procedures (other than lab tests) performed during the encounter
Number of medications	Number of distinct generic names administered during the encounter
Number of outpatient visits	Number of outpatient visits of the patient in the year preceding the encounter
Number of emergency visits	Number of emergency visits of the patient in the year preceding the encounter
Number of inpatient visits	Number of inpatient visits of the patient in the year preceding the encounter
Diagnosis 1	The primary diagnosis (coded as first three digits of ICD9); 848 distinct values
Diagnosis 2	Secondary diagnosis (coded as first three digits of ICD9); 923 distinct values
Diagnosis 3	Additional secondary diagnosis (coded as first three digits of ICD9); 954 distinct values
Number of Diagnoses	Number of diagnoses entered to the system
Glucose serum test result	Indicates the range of the result or if the test was not taken. Values: “>200’,’ “>300”, “normal”, and “none” if not measured
A1c test result	Indicates the range of the result or if the test was not taken. Values: “>8” if the result was greater than 8%, “>” if the result was greater than 7% but less than 8%, “normal” if the result was less than 7%, and “none” if not measured.
Change of medications	Indicates if there was a change in diabetic medications (either dosage or generic name). Values: “change” and “no change”
Diabetes medications	Indicates if there was any diabetic medication prescribed. Values: “yes” and “no”
24 features for medications	For the generic names: metformin, repaglinide, nateglinide, chlorpropamide, glimepiride, acetohexamide, glipizide, glyburide, tolbutamide, pioglitazone, rosiglitazone, acarbose, miglitol, troglitazone, tolazamide, examide, sitagliptin, insulin, glyburide-metformin, glipizide-metformin, glimepiride-pioglitazone, metformin-rosiglitazone, and metformin-pioglitazone, the feature indicates whether the drug was prescribed or there was a change in the dosage. Values: “up” if the dosage was increased during the encounter, “down” if the dosage was decreased, “steady” if the dosage did not change, and “no” if the drug was not prescribed

Readmitted	Days to inpatient readmission. Values: "<30" if the patient was readmitted in less than 30 days, ">30" if the patient was readmitted in more than 30 days, and "No" for no record of readmission.

With this mind, we can more intelligently modify our data set in order to prepare them for the models[2].

3.2 Class Distribution

For our classification task, we need to predict both the *readmission_1* and *readmission_2* attributes which are composed of '<30', '>30', and 'No' and '<30' and 'No' respectively. Consider the following pie graph of these three classes:

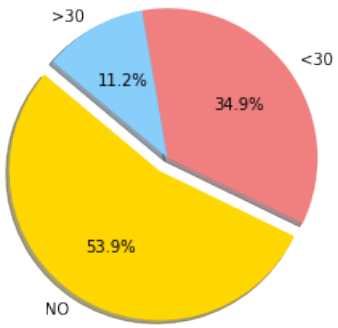

Figure 1: The distribution of the classes

This highlights something very important. In our *readmission_2* classification, we combine the '<30' and 'No' classes, meaning that the '<30' class would be in '11.2%' of our data points while the 'No' class would be in '88.8%' of our data points. In this instance, it is very important to not over-fit the models to only select for the 'No' class. Doing this would yield an accuracy rate of around '88%', which may seem right on the surface, but doesn't yield any valuable results because it will never actually predict the '<30' class, which we have established as the people who are at the most risk. We will address this in a future section using the SMOTE algorithm.

3.3 Missing Attributes

As with the majority of datasets, we have to deal with missing data, often times in different forms. In our dataset, we had missing data either in the form of NaN or '?' both of which we had to take into account. Combining both of them, we get the following table for missing data:[2]

Attribute	% missing
Encounter ID	0%
Patient Number	0%
Race	2%
Gender	0%
Age	0%
Weight	97%
Admission Type	0%
Discharge disposition	0%
Admission source	0%
Time in hospital	0%
Payer code	52%
Medical Specialty	53%
Number of lab procedures	0%
Number of procedures	0%
Number of medications	0%
Number of outpatient visits	0%
Number of emergency visits	0%
Number of inpatient visits	0%
Diagnosis 1	0%
Diagnosis 2	0%
Diagnosis 3	1%
Number of Diagnoses	0%
Glucose serum test result	0%
A1c test result	0%
Change of medications	0%
Diabetes medications	0%
24 features for medications	0%
Readmitted	0%

With this in mind, we made the decision to replace all instances of NaN and '?' with 'missing'. Given that 97% of all patients had their weights missing, we made the decision to drop that column from our analysis. We also decides to drop payer code from the analysis because the way that the patient chooses to pay will usually have no impact on the type of care they received. In addition, we chose to keep medical specialty as an attribute mostly because we believed that the kind of doctored that referred to the hospitality could be a hint to the gravity of their situation and thus the odds that they would be readmitted to the hospital. Every other attribute remained unchanged at this step [2].

3.4 Different Types of Attributes

As mentioned earlier there are many types of data, ranging from quantitative data points such as continuous and discrete data to qualitative data points such as binomial data, nominal data, and ordinal data. For example, in this case, there are many ordinal, discrete data categories such as the medications, which has 'steady', 'up', and 'down', where there is a clear order or ranking between the different terms. In addition, we had additional attributes left that we potentially needed to drop because of their lack of impact on the final class. The next few subsections will further discuss how we dealt with any remaining attributes.

3.4.1 One hot Encoding. In short, one hot encoding is way to easily turn categorical variables into a form that could help the machine learning models improve in prediction. It works particularly well with non-ordinal categorical data. Consider the following sample of the dataset:

Patient Number	Race	Gender
8222157	Caucasian	Female
86047875	African American	Female
82442376	Caucasian	Male
139505341	Other	Female

The problem with regular ordinal attribute encoding is that it assumes some kind of rank between the items. This is obviously not the case for categories such as race, admission type, or the medical specialty of the doctor that recommended them. No value of the attribute inherently has a higher significance than another value. One hot encoding converts any non-ordinal category into a binary category to be used as a feature to train the model. The above sample, once one-hot encoding is used on the Race attribute, looks like the following:

Patient Number	Caucasian	African American	Other	Gender
8222157	1.0	0.0	0.0	Female
86047875	0.0	1.0	0.0	Female
82442376	1.0	0.0	0.0	Male
139505341	0.0	0.0	1.0	Female

As a result, race is properly transformed into something more productive for the Machine Learning models that will be used. One hot encoding drastically increases the dimensionality of the dataset, which will be addressed in section 3.5 [5].

3.4.2 Binning Continuous Attributes. Continuous attributes are not particularly useful to classification algorithms when they are not transformed. This is because almost all classification algorithms needs every attribute to discrete in order to properly classify the final data. For example, if age was left untransformed, an algorithm such as the Naive Bayes Classifier would calculate the conditional probability of each age, which would not very productive as well as very time consuming. With that being said, we can bin continuous attributes in order convert them to categorical categories with rank. The original dataset has age binned, leaving us to bin attributes such as time_in_hospital. Consider the following table, where time_in_hospital_binned represents the final binned category. The binning was done in intervals of 3 [5].

Patient Number	time_in_hospital	time_in_hospital_binned
8222157	1	[1-3]
55629189	4	[4-6]
86047875	6	[4-6]
82442376	8	[7-9]

At this point, all of our attributes are ordinal, categorical attributes that we can further clean in the next step.

3.4.3 Ordinal Attribute Encoding. For this project, we decided to use existing Python libraries in order to ensure that we have the best implementation of each of these complex algorithms. With this in mind, sci-kit learn's implementation for various ML algorithms require that each attribute be a numeric value. As a result, we needed convert categories such as age, which was binned, to a numerical equivalent. What this meant was converting the value [0-10) to 0, [10-20) to 1, [20-30) to 2 and so on. This ensured that any rank in a certain attribute was preserved. Just as [10-20) means that this individual is older than an individual that is [0-10), 2 is greater than 1. This process, combined with the binning process, shrinks each continuous attribute into a number between 0 and the total number bins. At this point, all of our attributes are numeric and discrete and thus ready feature selection[5].

3.5 Feature Selection

At this point in the cleaning process, we were left with 149 attributes, far more than the original amount of 50. This was mostly because of the one-hot encoding that we performed to deal with non-ordinal categorical data. As a result, it becomes important to reduce the total number of attributes so that the time that the algorithms take to run is less. In addition, if these attributes are selected properly, we can both prevent over-fitting and increase our accuracy rate. We can perform feature selection in order to reduce the total number of attributes in our dataset. The two forms of feature selection that we perform are LASSO algorithm and Correlation Analysis.

3.5.1 LASSO Algorithm. The LASSO algorithm is a shrinkage and selection method for the attributes in a dataset. With the high dimensionality of the dataset, we needed a way of decreasing the dimensions was used. The feature reduction algorithm that was used was LASSO. LASSO performs variable reduction and normalization and is good for sparse data. Lasso fits a linear model to the data using the least sum of squared residuals with a bound. Therefore, it can remove features that are poor predictors. As a result of the LASSO algorithm, we reduced the number of attributes from 149 to 61.

3.5.2 Correlation Analysis. Correlation analysis is a process by which we can remove attributes that are highly correlated with other attributes. This is an important step in feature selection because it can remove redundant attributes that are not significant to the data mining process. For example, if we had a data set that included kilometers per hour and miles per hour, these two statistics are highly correlated so we can simply remove one of them because one of the attributes can be directly predicted using the other. In order to address this, we can generate a heatmap of all the Pearson correlation coefficients in the dataset, which we can then use to select attributes. For reference, the formula for the Pearson correlation for two attributes is as follows[5]:

$$\frac{n\sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{\sqrt{(n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2)}}$$

Below is the correlation heatmap for the dataset using the Pearson correlation:

The attribute labels are omitted in order to prevent clutter. After the correlation was found, I removed any attributes that were highly correlated (Pearson correlation greater than 0.7) with at least one other attribute [5].

3.6 The Synthetic Minority Oversampling Technique

The Synthetic Minority Oversampling Technique (SMOTE) is an algorithm that essentially helps balance the dataset that we run the model on. For example, consider a case where there is an algorithm

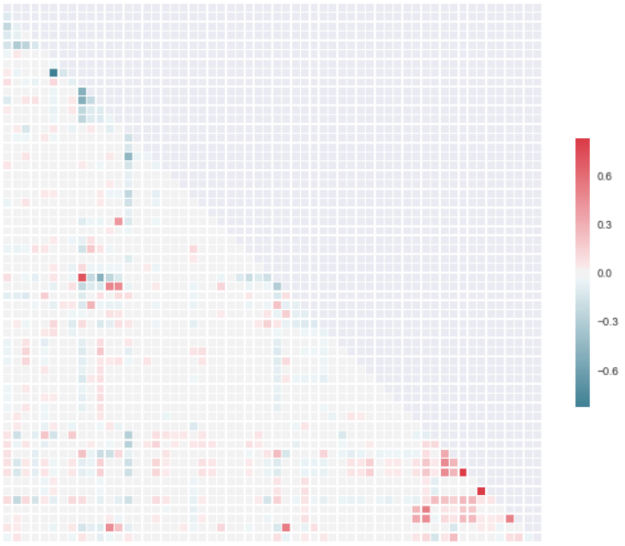

Figure 2: Correlation Heatmap

to predict whether a transaction is fraudulent. Relative to the total number of transactions, very few transactions are fraudulent. As a result, it makes sense that we wouldn't want to always predict that a transaction is not fraudulent even though this would result in a very high accuracy rate. SMOTE deal with this issue by generating artificial samples of the rare class and then over sampling the rare class. This allows more equality between the classes in our classification problems and will help with our *readmission_2* classification. This was the last step of our data processing step, leaving only the application of the models and evaluation of the results [5].

4 CLASSIFICATION MODELS

Classification aims to predict the categorical class labels, which can be either discrete or nominal. All classification models classify data based on a training set, which comprises of the attributes as well as the classifying attribute. Outside of diabetic readmission, classification problems exist in credit/loan approval as well as web page and video categorization. Once the model is constructed, the model is then tested on a test set where we can arrive at an accuracy that represents how good our model is.

In this section, we will take a closer look at the following five models:

(1) Naive Bayes Classifier
(2) Logistic Regression
(3) Random Forest Classifier
(4) AdaBoost Classifier
(5) Neural Network Classifier

For each of these models, we will discuss its implementation, as well as its strengths and weaknesses. It is important to note that the Naive Bayes Classifier will serve as the baseline for performance. With that being said, we performed a 75%-25% split on our data, meaning that our models were trained on 75% of our dataset, and then tested our models on 25% of the dataset to arrive at an accuracy rate.

It is important to note that these models will be run a total of 10 times. Each of the five models will be run on both the *readmitted_1* and *readmitted_2* classes.

All the implementations for these algorithms are from sklearn.

4.1 The Naive Bayes Classifier

The Naive Bayes Classifier is a basic classification algorithm that uses conditional probabilities to predict the class of a certain row in the dataset. Mathematically speaking, in order to predict the class label of X, where X is tuple in the test dataset, we must pick the class C_i if and only if

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) for 1 \leq j \leq m.$$

Equivalently, the predicted class label is the class C_i for which $P(X|C_i)P(C_i)$ is the maximum. We can compute the values for $P(X|C_i)$ using the following formula:

$$P(X|C_i) = \prod_{x_j \in X} P(x_j|C_i)$$

For the implementation of the algorithm, we used the Gaussian Naive Bayesian Classifier provided in the sklearn library. The Gaussian Naive Bayes Classifier is slightly more robust than the regular Naive Bayes Classifier because it is also able to place any continuous-valued attribute on a Gaussian distribution. However, this is not applicable for our project because all of our attributes are purely categorical.

This raises the question of how effective Bayesian classifiers really are. Various studies have found it to be comparable in some domains to both decision trees and neural networks. However, this is not always the case. As mentioned in the previous equation, we find each conditional probability of a tuple by multiplying the conditional probabilities of each attribute in the tuple. This makes a dangerous assumption that we cannot always guarantee at this step in the process. This assumes class-conditional independence, which does not always hold true. In addition, the lack of probability data for rare classes can force us to use the Laplacian correction, which can lead to poor results. Despite its faults, the Naive Bayes Classifier remains a good algorithm for finding a baseline performance for the rest of our algorithms. It is easy to implement and is computationally quick, meaning that it can handle large datasets well with proper hashing [5].

Consider the following sample of conditional probabilities from our Naive Bayes Classifier:

Conditional Term	Probability
$P(race = Caucasian \mid < 30)$	0.7401
$P(race = Caucasian \mid NO)$	0.7211
$P(race = Caucasian \mid > 30)$	0.7480
$P(age = [60-70) \mid < 30)$	0.2168
$P(age = [60-70) \mid NO)$	0.2169
$P(age = [60-70) \mid > 30)$	0.2221

As you can see, the individual conditional probabilities are often similar, but when multiplied over the course of all attributes in a tuple, and then multiplied by the probability of a certain class, a clear likely class hopefully arises [5].

4.2 Logistic Regression

Logistic Regression is a strong algorithm that is particularly good with binary classification problems. With that being said, it is still a good algorithm for a dataset with multiple classes, such as *readmitted_1*. At the core of the Logistic Regression algorithm is the logistic function, which can also be called the sigmoid function. The logistic function is as follows:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Plotting this function where $x \in [-5, 5]$, we get the following result:

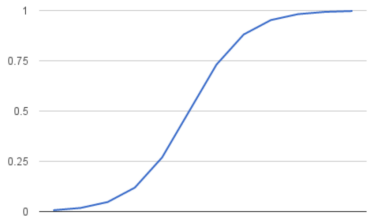

Figure 3: Logistic Function

The logistic function will always return a value between 0 and 1, so it makes sense to treat it as a classifying probability based on the value. For example, in a binomial classification problem, it makes sense to use a threshold of 0.5, and if the function is greater than 0.5, it predicts one class, and if below 0.5, predicts the other class.

Logistic regression allows us to analyze the coefficients to understand which attributes are more likely to impact the final result. In addition, Logistic Regression is particularly strong for binary classification problems because we can form a threshold for our logistic function. With that being said, Logistic Regression struggles with there are multiple highly correlated categories, which we needed to address in section 3.5.2. Nonetheless, Logistic Regression is an effective algorithm with a low complexity overhead that allows us to accurately predict the class of a tuple [5].

4.3 Random Forest Classifier

The basis for the Random Forest Classifier is Decision Tree induction, which is the learning of decision trees from class-labeled tuples. A decision tree basically functions like a flow chart, where each internal node denotes a test of an attribute, and each branch off of the node represents an outcome of the test. Consider the following example of a decision tree:

Decision Tree classifiers are an extremely popular classification method. They do not require any domain knowledge and can easily handle multidimensional data. They are also strong algorithms when dealing with multiple classes such as *readmitted_1*. In general, the learning and classification steps of decision tree induction are both simple in implementation as well as complexity. As a result, decision tree algorithms have good accuracy rates. With that being said, success is often highly dependent on the data set. There are

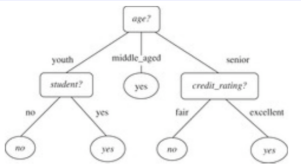

Figure 4: Example of Decision Tree

multiple existing methods to help us generate the Decision Trees. These include **ID3** (Iterative Dichotomiser), **C4.5** (a successor of ID3), and **CART** (Classification and Regression Trees). All of these algorithms use a greedy approach where the decision trees are constructed in a top-down recursive divide-and-conquer algorithm.

The critical step of the algorithm is the find the splitting criterion, also known as the variable at which the dataset should be partitioned on at a single node. Each of these algorithms use a different splitting criterion. For example, the ID3 algorithm uses **information gain** as its attribute selection measure for a dataset D.

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D)$$

Thus, the information gain is:

$$Gain(D) = Info(D) - Info_A(D)$$

In addition, the CART algorithm uses a different splitting criterion, the Gini index, in order to measure the impurity of a tuple D. The Gini index is as follows:

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

For example, when considering a binary split, we compute the weighted sum of the impurity of the partition. If there is a split on a subset A of an attribute into partitions D_1 and D_2, then the Gini index of D on A is:

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

In this case, we choose the attribute subset that results in the maximum reduction of impurity.

With that in mind, consider an ensemble of decision tree classifiers. This is called a random forest classifier. In this case, individual decision trees are generated using a random selection of attributes at each notes to determine the split. In order to determine the classification, each tree votes for the class of the tuple and then the most popular tree is returned. Random forests are built using bagging along with random attribute selection. In order to grow each tree, the CART methodology is used [5].

4.4 AdaBoost Classifier

Like the random forest algorithm, the AdaBoost classifier is an ensemble method. In a boosting algorithm such as AdaBoost, weights are assigned to each training tuple and a series of k classifiers are iteratively learned. Once a certain classifier M_i is learned, the weights are then updated to allow the next classifier, M_{i+1} to pay more attention to the training tuples that were inaccurately classified by M_i.

AdaBoost (Adaptive Boosting) is a popular boosting algorithm. Given a dataset D, where we have a like of class-labeled tuples $(X_1, y_1), (X_2, y_2), (X_3, y_3), ..., (X_d, y_d)$, where y_i is the class label for tuple X_i and $d = |D|$. Each training tuple is assigned a weight of $\frac{1}{d}$. We are able to generate the k classifiers for the ensemble by running the rest of the algorithm k times. In each round, we sample a portion of the initial dataset D known as D_i. A classifier model M_i is derived using the training tuples of D_i and the error rate is calculated using D_i as a test set. After this, these weights that have been generated are then adjusted when compared to the next model after M_i.

For reference, the method to calculate the error of a certain model M_i is

$$error(M_i) = \sum_{j=1}^{d} w_j \times err(X_j)$$

where $err(X_j)$ is the misclassification error of tuple X_j.

Boosting algorithms often run the risk of overfitting the model to fit the training data. What this means is that the "boosted" model is often times less accurate than the single model derived from the data. In addition, AdaBoost is computationally intensive model. Despite this, "boosting" models can be very accurate if the weights for the tuples are accurately monitored [5].

4.5 Neural Network Classifier

At a high level, a **neural network** is a set of connected input and output units where each connection between an input and output unit has a weight associated with it. Every node it put into a layer, where each layer can contain any amount of nodes and each neural netowork can contain as many layers as necessary. The effects of layers and nodes per layer will be further studied in our results section. There is an input layer, that takes in all of the tuples, and an output layer that results in our classification prediction.

The disadvantages for neural networks are clear. They often require long training times and are more suitable for applications where the model does not need to be tested repeatedly. In addition, neural networks require additional information involving network topology as well as general structural issues that require testing. Even though neural networks are just a set of nodes and weights that are updated via backpropagation, neural networks are nonetheless hard to interpret. There is no real intuitive meaning behind the weights between nodes.

With that being said, one of the big advantages of neural networks is the fact that they are highly tolerant of noisy data and are able to classify patterns on information that the neural network has not even been trained on. Even though our data is highly categorical, neural networks are nonetheless well suited for continuous inputs and outputs. Neural networks are regularly used on many real-world datasets, including character recognition as well as natural language processing [5].

5 RESULTS

As mentioned earlier, each of these models were run twice, once for *readmitted_1* and once for *readmitted_2*. Just for reference, the SMOTE algorithm was run for the classification of *readmitted_2*. Below are the accuracy rates for both of these classifications of the data:

Model	readmitted_1 accuracy
Naive Bayes Classifier	40.6%
Logistic Regression	55.9%
Random Forest Classifier	56.1%
AdaBoost Classifier	56.2%
Neural Network Classifier	55.7%

Model	readmitted_2 accuracy
Naive Bayes Classifier	23.7%
Logistic Regression	88.7%
Random Forest Classifier	88.8%
AdaBoost Classifier	88.8%
Neural Network Classifier	88.7%

6 DISCUSSION

Over the course of this project, we developed a system for cleaning our messy, highly categorical data and analyzed our cleaned dataset using five different models, each with their own benefits and disadvantages. The two tables in the results section properly summarize our accuracy rates for both of our classification problems. Look at the results, we see that the AdaBoost classifier had the highest accuracy for both our *readmitted_1* and *readmitted_2* classification problems. In our testing of our AdaBoost algorithm, we made sure that the model was indeed not overfitting to the dataset, something that we had established as a clear fault of the algorithm. Thus, it was expected that the AdaBoost algorithm would give us the best results. In addition, as expected, the Naive Bayes Classifier indeed gave us the worst accuracy. The Naive Bayes Classifier served as a baseline that all the other algorithms must outperform. However, we did not anticipate that the binary classification problem of *readmitted_2* would have a lower accuracy rate than the regular classification. We hypothesize that this is because the conditional probabilities were different than the regular classification, and with the lack of minority data, forced our program to revert to the Laplacian correction. In terms of time complexities, the models in increasing runtimes were as follows:

(1) Naive Bayes Classifier
(2) Logistic Regression
(3) Random Forest Classifier
(4) AdaBoost Classifier
(5) Neural Network Classifier

Thus, if we include runtime as a constraint in model selection, it makes sense to pick Logistic Regress or Random Forest Classification for a far faster algorithm with still comparable results [5].

REFERENCES

[1] Basics | Diabetes | CDC. Retrieved May 1, 2019 from https://www.cdc.gov/diabetes/basics/diabetes.html

[2] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore, "*Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records*, BioMed Research International, vol. 2014, Article ID 781670, 11 pages, 2014.

[3] Diabetes Data Set. Retrieved May 1, 2019 from https://archive.ics.uci.edu/ml/datasets/diabetes

[4] International Diabetes Federation (2017). I*DF Diabetes Atlas, 8th Ed.* Brussels, Belgium: International Diabetes Federation.

[5] Jian Pei, Jiawei Han, Micheline Kemper. 1993. *Data Mining: Concepts and Techniques (3rd Ed.).* Morgan Kaufmann; (July 6, 2011)