# CLINICAL DATA BASED PATIENT READMISSION PREDICTOR

**BY**
SHRUTI PATANKAR
VIJET BADIGANNAVAR
RUTA GADGIL
VISHAL SANJIV KOTAK
SWAPNIL MAHAJAN

# Table of Contents

# Introduction

## Motivation

This project has been implemented as a part of Northeastern University course CS6220 (Data Mining Techniques). This is a two-class classifier that can handle datasets with many numbers of features and instances. For details of data, please visit the https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008# website.

## Overview

The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. This project exactly uses this concept for a hospital data, to predict, if a patient will return to the hospital in next thirty days. The data contains attributes such as patient number, race, gender, age, admission type, time in hospital, medical specialty of admitting physician, number of lab test performed, HbA1c test result, diagnosis, number of medication, diabetic medications, and emergency visits in the year before the hospitalization, etc. Based on this data, the predictor built by this project classifies the instance as "1" (the patient will return) or "0" (the patient will not return). Identifying factors that highly affect the outcome i.e. if a patient will return or not, and factors that affect the other factors can help in analyzing and predicting if a patient will return. This may help in taking extra care to improve patient safety.

# Analysis

## Problem domain

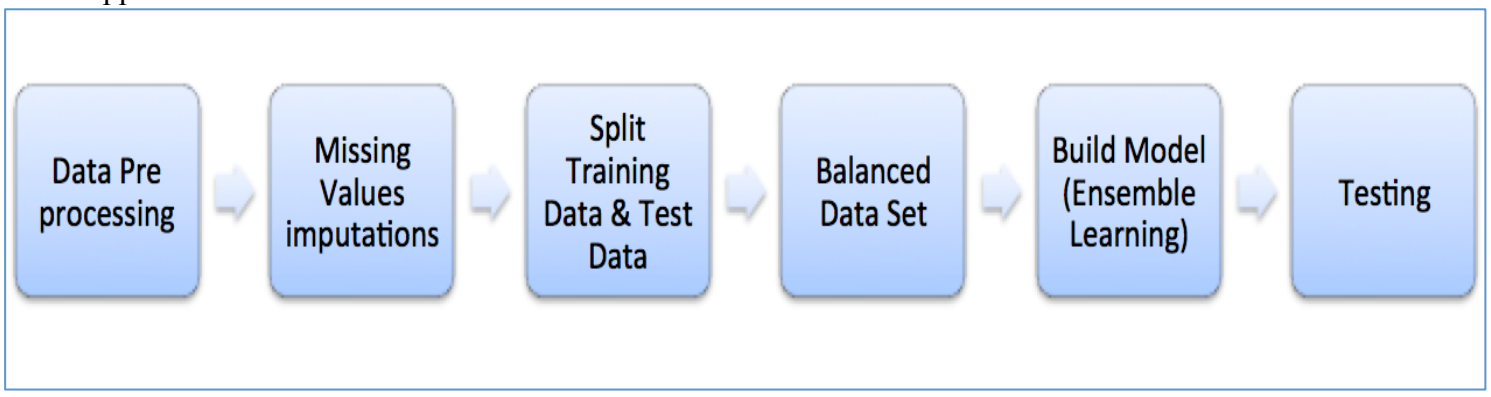
This is a medical domain problem. The data used, was submitted on behalf of the Center for Clinical and Translational Research, Virginia Commonwealth University.

## Application domain

This project is a returning patient classifier and can be used in medical/clinical domain to predict if a patient will return given his medical data.

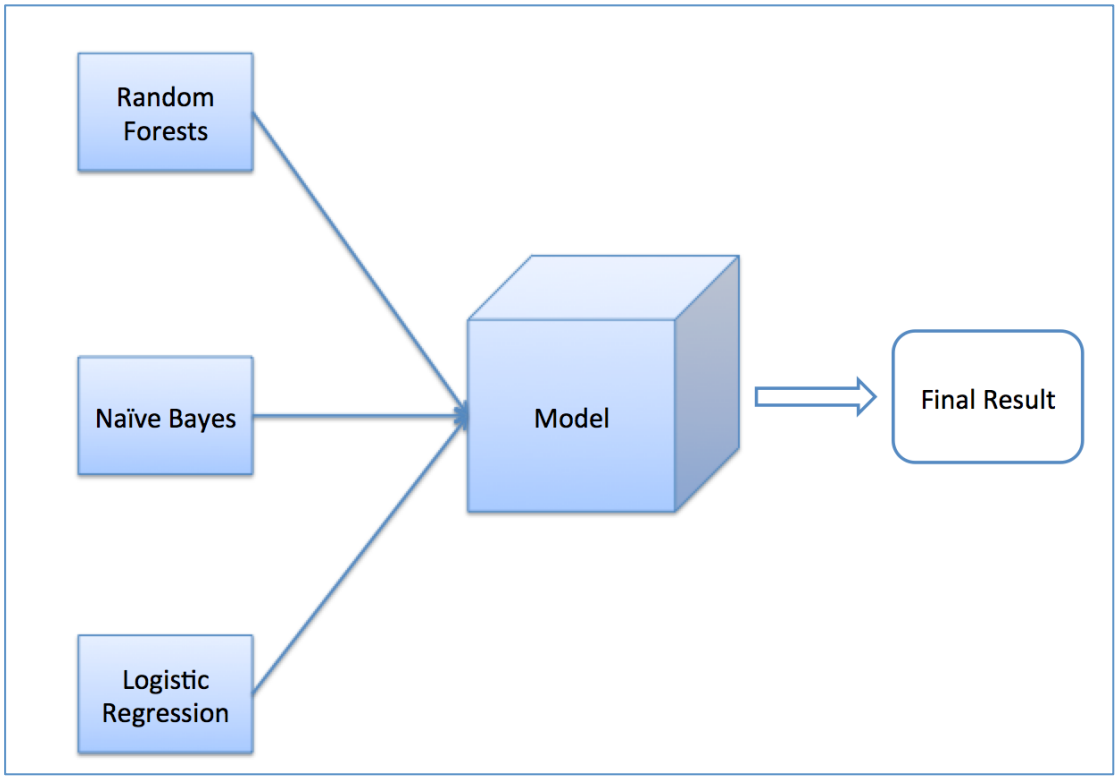
# Design

## Basic Approach used:




## Algorithm:

Once the data is cleaned and missing values are imputed, for predicting the most accurate outcome of the instances, we have used three algorithms and we predict the final outcome based on the majority outcome of the algorithms.




We will describe the choices of these algorithms going ahead. We observed that the highest accuracy we achieved is for random forests algorithm. But, we avoid using only one algorithm. This is because, even though the predictor gives higher accuracy for training data, it may not work well with test data or unseen data in general. Thereby, the use of ensemble predictor consisting of the above three algorithms is used for learning to enhance the future predictions.

Technology

1. Java 1.8
2. Eclipse Luna
3. Java Weka API 3.8.0
4. SMOTE 1.0.2
5. R Studio for basic analysis (not used for final project implementation)
6. Weka GUI for basic analysis (not used for final project implementation)

## Implementation

Data Preprocessing

Data Cleaning

*Column: Readmitted*

The data set initially had three labels for the final outcome column, i.e “No”, “>30”, “<30”. We are treating “No” and “>30” as same class and we label it as zero. Whereas, for instances having outcome column with value “<30”, we label it as one. After doing the above conversion these are the numbers we get for class labelled 1 and 0 respectively.

```
>30 or none : 90409
<30 :11357
```

*Columns: diag_1, diag_2, diag_3*
The columns, diag_1, diag_2, and diag_3 are in fact the ICD-9 codes of the international statistical classification of diseases and related health problems. These values were converted to an ICD-9 category based on its value. There are 19 categories of ICD-9 codes, in which the current data has been classified. This is anticipating the fact that a certain type of disease may have a high impact on a patient getting readmitted. The ICD-9 codes have been set in the data as follows:

ICD-9-type1 - codes 001–139: infectious and parasitic diseases
ICD-9-type2 - codes 140–239: neoplasms
ICD-9-type3 - codes 240–279: endocrine, nutritional and metabolic diseases, and immunity disorders
ICD-9-type4 - codes 280–289: diseases of the blood and blood-forming organs
ICD-9-type5 - codes 290–319: mental disorders
ICD-9-type6 - codes 320–359: diseases of the nervous system
ICD-9-type7 - codes 360–389: diseases of the sense organs
ICD-9-type8 - codes 390–459: diseases of the circulatory system
ICD-9-type9 - codes 460–519: diseases of the respiratory system
ICD-9-type10 - codes 520–579: diseases of the digestive system
ICD-9-type11 - codes 580–629: diseases of the genitourinary system
ICD-9-type12 - codes 630–679: complications of pregnancy, childbirth, and the puerperium
ICD-9-type13 - codes 680–709: diseases of the skin and subcutaneous tissue
ICD-9-type14 - codes 710–739: diseases of the musculoskeletal system and connective tissue
ICD-9-type15 - codes 740–759: congenital anomalies
ICD-9-type16 - codes 760–779: certain conditions originating in the perinatal period
ICD-9-type17 - codes 780–799: symptoms, signs, and ill-defined conditions
ICD-9-type18 - codes 800–999: injury and poisoning
ICD-9-type19 - codes E and V codes: external causes of injury and supplemental classification

*Column: admission_type_id*
This column was replaced with the name of the admission type corresponding to each id provided in the file IDs_mapping.csv. This was done to avoid giving importance to a category of admission type over the other (Having integers instead of categories would have meant that category with value 1 is less important than category with value 2). Also, the "NULL" and "Not Available", "Not Mapped" was set to be a "?" since it does not provide any information about the category, but it simply means that the value is missing. These two values should not be considered as a category since it will make the data corrupt.

*Column: discharge_disposition_id*
This column was replaced with the name of the discharge disposition id corresponding to each id provided in the file IDs_mapping.csv. This was done to avoid giving importance to a category of discharge disposition type over the other. For this column, the ids corresponding to "NULL", "Not Mapped" and "Unknown/Invalid" were replaced with "?" to denote missing value.

*Column: admission_source_id*
This column was replaced with the name of the admission source id corresponding to each id provided in the file IDs_mapping.csv. This was done to avoid giving importance to a category of admission source over the other. For this

column, the ids corresponding to "NULL", "Not Mapped", "Not Available" and "Unknown/Invalid" were replaced with "?" to denote missing value.

*Column: age*
For the column age, each of the age values was replaced with an integer. For example, [0-10) is set to value 1, whereas, [10-20) is set to value 2. Here, since age [0-10) is less than [10-20), replacing [0-10) with 1 and [10-20) with 2, makes [10-20) a higher value than [0-10), which is in fact, the case. Likewise, the rest of the age categories are replaced with integers.

*Column: weight*
For the column weight, each of the weight was replaced with an integer. For example, [0-25) is set to value 1, whereas, [25-50) is set to value 2. Here, since weight [0-25) is less than [25-50), replacing [0-25) with 1 and [25-50) with 2, makes [25-50) a higher value than [0-25), which is in fact, the case. Likewise, the rest of the weight categories are replaced with integers.

Missing values imputation

| Total number of observations | After removing all the missing values |
|---|---|
| 101766 | **25846** |

Given 101766 number of instances and choosing only 25846 out of them may not be a good idea. This is because 25,846 observations are only **25%** of all the data that is available. Also, only few feature values of observations may be missing. Throwing out the complete observation, just because few feature values are missing is not wise. Thereby, we decide to impute the missing values.

Missing values of weight was replaced with mean value. However, a genuine observation of weight column shows that since lot of values are missing, imputations will not help much. This column needs to be discarded altogether. For columns, payer code, medical specialty, diag_1, diag_2, diag_3, and race, the missing values were replaced using mode, since these are categorical values. Also, discharge_disposition_id, admission_type_id and admission_source_id values were changed to string values during data cleaning and as discussed, the values like "NULL", were changed to "?". These values being categorical are imputed using mode as well.

Discarding irrelevant features:
While feature selection helps in achieving higher accuracy, whereas, few columns which do not contribute anything to the predictor, can be dropped. This helps in discarding values that are completely irrelevant to the problem. For example, we dropped columns encounter_id, patient_nbr, examide and citoglipton. Even before starting with feature selection, this step helps in considering only those features, which are not completely irrelevant.

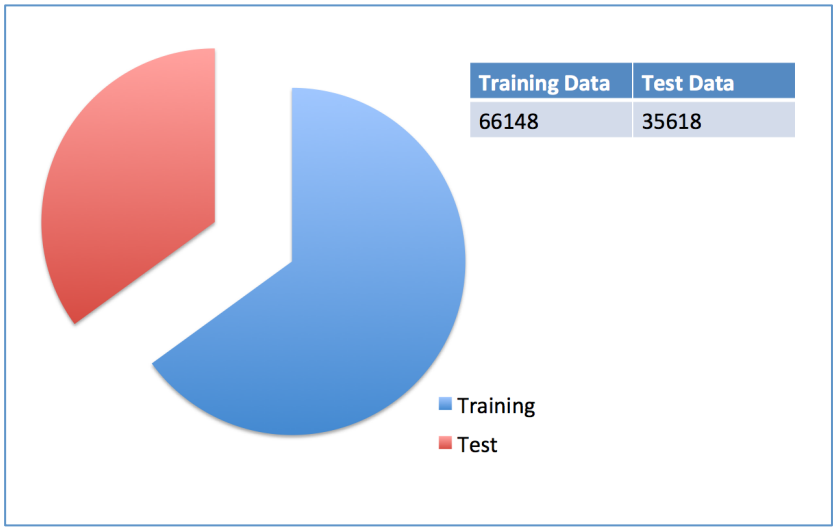
Feature Selection:
We have used Wrapper method for feature selection using Weka's BestFirst Class. BestFirst may start with the empty set of attributes and search forward, or start with the full set of attributes and search backward, or start at any point and search in both directions (by considering all possible single attribute additions and deletions at a given point). For the purpose of this project we have used forward selection method. Using forward selection method, we get following features:

| discharge_disposition_id, number_inpatient, chlorpropamide, acetohexamide, tolbutamide, troglitazone, glipizide-metformin, metformin-rosiglitazone, metformin-pioglitazone, diabetesMed |
|---|

| Number of Features | Accuracy |
|---|---|
| 10 features with feature selection method | 84.06% |
| 45 features without feature selection | 84.06% |

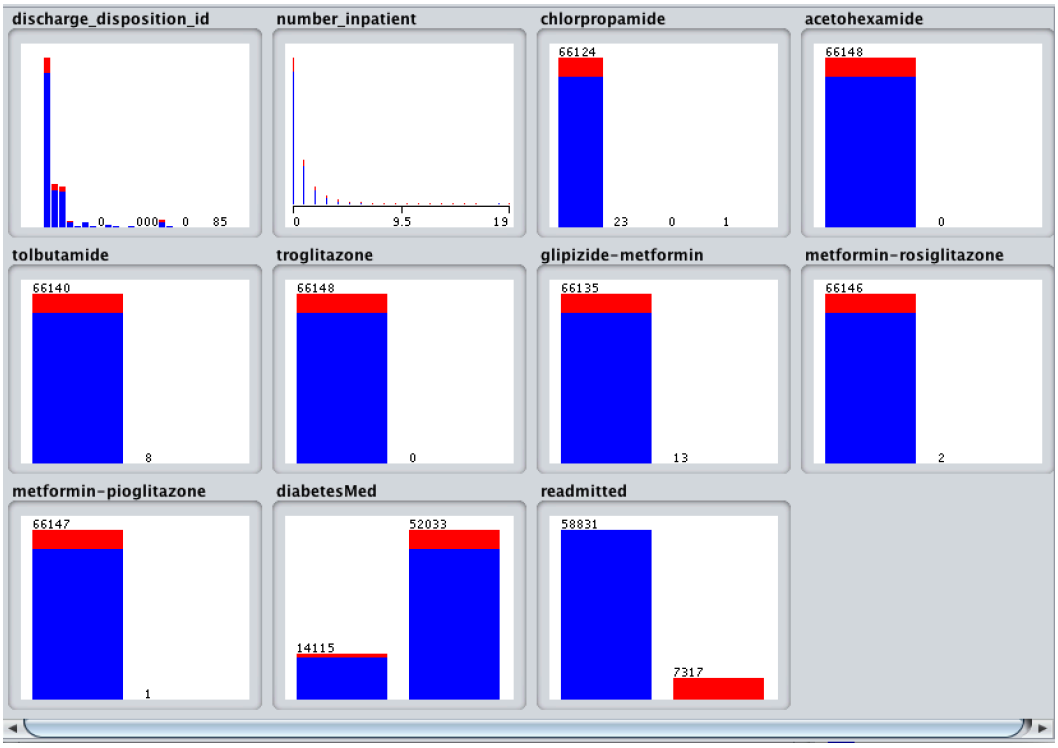
As these above experiments suggest we observe, using selected features or all the features gives same accuracy. Thereby, we have used selected features considering the fact that a model with lesser features will run faster. Also, if few features can correctly predict the outcome, we can save the costs of collecting the values of rest of the features in the future.

Division of data into training and test set:
We decided to keep a test set aside, so that the accuracies obtained are for the data that is not seen by the model at all. We divide the data into 65% for training data and 35% for test data. Total number of instances is 101766. Accuracies based on 35% unseen data seems to be a good measure given that the training was done on a larger part i.e. 65% of the data.

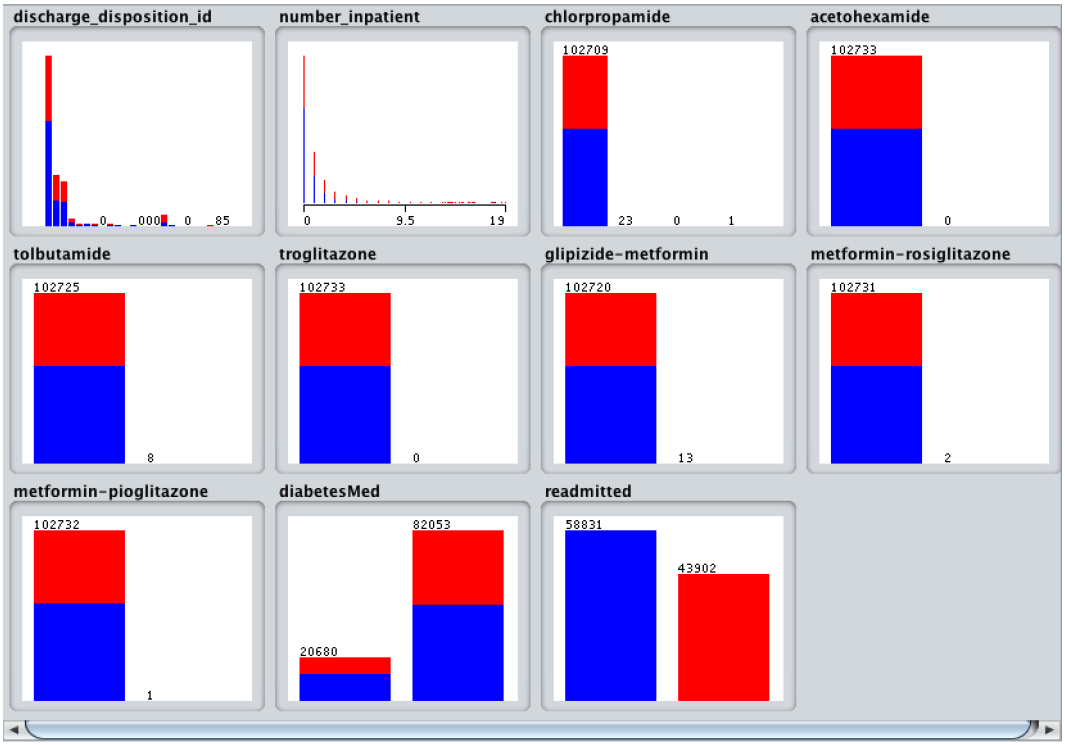



Remedies for imbalanced sets:
SMOTE - (Synthetic Minority Over-sampling Technique) is an over-sampling approach in which the minority class is over-sampled by creating “synthetic” examples rather than by over-sampling with replacement. The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen. Our implementation currently uses k nearest neighbors depending on Weka. For instance, if the amount of over-sampling needed is 200%, only two neighbors from the five nearest neighbors are chosen and one sample is generated in the direction of each. Synthetic samples are generated in the following way: Take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general.

BEFORE




AFTER



Number of instances of training data after balancing data is 102733. The split is as shown in the above diagram. Note that we have split the data into training and test set before balancing the data. This ensures that the test data does not contain any synthetically created examples. The test data is part of the real data collected and should give better results.

We implemented 500% increase for minority class after exploring the following options for data balancing:

| **Percentage increase in minority set** | **Confusion Matrix** | **Percent of Correctly classified instances** | **Percentage of incorrectly classified instances** |
|---|---|---|---|
| 700% | (Classified as)<br>class 0 class 1<br>(True Values)<br>class 0 26832 4746<br>class 1 2667 1373 | 79.19% | 20.81 |
| 600% | (Classified as)<br>class 0 class 1<br>(True Values)<br>class 0 28069 3509<br>class 1 2954 1086 | 81.85% | 18.15% |
| 500% | (Classified as)<br>class 0 class 1<br>(True Values)<br>class 0 29248 2330<br>class 1 3226 814 | 84.40% | 15.60% |

We also tried to do oversampling for balanced data, however the accuracies obtained for each of the algorithms were not acceptable and we observed that SMOTE gives a better result.

| | |
|---|---|
| Balanced data using Oversampling | Ensembled predictor accuracy ~ 73% |
| Balanced data using SMOTE | Ensembled predictor accuracy ~ 84% |

## The Classifier

Design:
This qualified data set is fed to each of the classifier we have chosen, in order to predict the value of each of the testing instance. The majority vote of these three classifiers is used as a final prediction of that testing instance.

Choice of algorithms:
We experimented with 8 different algorithms for choosing 3 algorithms to be finally used and then try to fit an ensemble model by taking the majority vote. We chose 3 algorithms so that the majority vote is never even and is always odd. Here is the list of algorithms we used:

a. J48 Classifier:

This is an improvement of ID3 algorithm used for Decision Trees. However, random forest algorithm uses an ensemble method in itself. This enhances the decision tree performance and even though we explored this option, we did not include it as one of the algorithms of final predictor model since it would simply be a repetition of random forests and in fact random forests work better. See explored options section to know more analysis of this algorithm.

b. Naive Bayes
Naive Bayes is one of the best algorithms for classification. It also adds weights to features based on the correlation of feature with the final outcome. This in itself is a huge advantage to predict the correct outcome. Also, Naive Bayes being probabilistic classifier, it should give a good indication clubbed with the decision from random forests, which are not probabilistic and logistic regression.

c. Random Forests
Random forests give state of the art performance since they pick random samples, pick random features and give a majority based final decision. We have included random forests as one of the algorithm for the final predictor.

d. Logistic Regression:
Logistic Regression is used a lot in the industry for classification problems. Since it gives a probabilistic estimate, we thought it is a good idea to use Logistic Regression along with Random Forests and Naive Bayes to give the final output.

e. ID3
ID3 is one of the methods used for implementing decision trees, however since we are using Random forests, we did not use ID3 as it would have been repetition.

f. PART
PART is also another way of implementing decision trees, however since we are using random forests, we did not use PART for predicting the final outcome.

g. SVM
SVM takes a very long time to run. Thereby, we have decided to not use SVM as one of our final classifiers.

h. KNN:
KNN doesn't know which attributes are more important. Whereas, using Naive Bayes or logistic makes more sense. KNN is a lazy classifier and takes a very long time to finish since it has to compute distance between each point with every other point. Also, deciding K is one of the open research area. We have also used SMOTE for balancing the imbalanced data and we think that KNN classifier will get negatively affected because of the synthetically created examples. Thereby, we have not used KNN.

Random Forests do give state of the art performance. We did consider the option to only consider random forests to predict the outcome of out built model. However, we do not want to rely on the output of only one algorithm. Also, small changes in training data can affect random forests a lot. Moreover, random forests never use likelihood estimates for predicting final outcome. Thereby, we also wanted to club Naive Bayes and Logistic Regression with Random Forests.

## Explored options discussions

a. J48:
Class for generating a pruned or unpruned C4.5 decision tree. Decision tree algorithms begin with a set of cases, or examples, and create a tree data structure that can be used to classify new cases. Each case is described by a set of attributes (or features) which can have numeric or symbolic values. Associated with each training case is a label

representing the name of a class. Each internal node of a decision tree contains a test, the result of which is used to decide what branch to follow from that node. For example, a test might ask "is x > 4 for attribute x?" If the test is true, then the case will proceed down the left branch, and if not then it will follow the right branch. The leaf nodes contain class labels instead of tests. In classification mode, when a test case (which has no label) reaches a leaf node, C4.5 classifies it using the label stored there.

b. Decision Tree using Part classifier:
Class for generating a PART decision list. Uses separate-and-conquer. Builds a partial C4.5 decision tree in each iteration and makes the "best" leaf into a rule. More info can be found in the references

c. Decision Table classifier (ID3):
This class is for building and using a simple decision table majority classifier. More info can be found in the references.

d. IBK classifier (KNN Classifier):
This is a function that maps instances to categories: given an instance drawn from the instance space, it yields a classification, which is the predicted value for this instance's category attribute. An instance-based concept description includes a set of stored instances. This set of instances can change after each training instance is processed. However, IBL/IBK algorithms do not construct extensional concept descriptions. Instead, concept descriptions are determined by how the IBL/IBK algorithm's selected similarity and classification functions use the current set of saved instances.

e. LibSVM (SVM)
LibSVM runs faster than SMO since it uses LibSVM to build the SVM classifier. LibSVM allows users to experiment with One-class SVM, Regressing SVM, and nu-SVM supported by LibSVM tool. LibSVM reports many useful statistics about LibSVM classifier (e.g., confusion matrix, precision, recall, ROC score, etc.).More info can be found in the references.

## Tests

Confusion matrix, percentages of correctly/incorrectly classified instances:

| **Algorithm** | **Confusion Matrix** | | | **Percent of Correctly classified instances** | **Percentage of incorrectly classified instances** |
|---|---|---|---|---|---|
| Naive Bayes | | (Classified as) | | 85.11% | 14.89% |
| | | class 0 | class 1 | | |
| | (True Values) | | | | |
| | class 0 | 29591 | 1987 | | |
| | class 1 | 3317 | 732 | | |

| | | | |
|---|---|---|---|
| Logistic Regression | (Classified as)<br>class 0 class 1<br>(True Values)<br>class 0 28520 3058<br>class 1 3036 1004 | 82.89% | 17.11% |
| Random Forests | (Classified as)<br>class 0 class 1<br>(True Values)<br>class 0 30424 1154<br>class 1 3662 378 | 86.48% | 13.52% |

| **Algorithm** | **Confusion Matrix** | **Percent of Correctly classified instances** | **Percentage of incorrectly classified instances** |
|---|---|---|---|
| Final ensembled predictor (Model) | (Classified as)<br>class 0 class 1<br>(True Values)<br>class 0 29248 2330<br>class 1 3226 814 | 84.40% | 15.6% |

## Result

The results obtained show a good accuracy value overall, however, the confusion matrix still shows that in spite of using balanced data to build the model, the class 1 in test data was not quite correctly classified after all. Even after using ensembled predictor, the class 1 is not yet classified up to the mark. More options can be explored to achieve this.

## Conclusion

We were able to create a classifier for predicting if a patient will be readmitted in a hospital based on 10 features out of 49 features. We cleaned the data, preprocessed it to remove irrelevant features, selected features, built a model using three different algorithms and gave a final prediction.

## Future Work

Additional experimental results can be used to validate the results obtained in this analysis. Cost and runtime of using KNN classifier and SVM can be explored.

## References

https://en.wikipedia.org/wiki/Data_mining
https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008#
https://www.hindawi.com/journals/bmri/2014/781670/
https://en.wikipedia.org/wiki/List_of_ICD-9_codes
https://www.jair.org/media/953/live-953-2037-jair.pdf
https://www.jair.org/media/953/live-953-2037-jair.pdf
Springer - J48
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
Springer - IBK

https://en.wikipedia.org/wiki/Random_forest
http://web.cs.iastate.edu/~honavar/bayes-continuous.pdf
http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/PART.html