Predicting Readmission of Diabetic Patients Using Various Classification Techniques

Abhinay Sarvayyagari, Ashwin Venkatesh Prabhu, Febin Zachariah, Rujuta Mahindrakar

University of North Carolina at Charlotte

Author Note

Under the guidance of Dr. Xi Niu, Assistant Professor, Department of Software and Information Systems, University of North Carolina at Charlotte

Table of Contents

Abstract

Diabetes is a chronic condition prevalent among more than 25 million people across the world, affecting people of all ages. It can be said as a condition of the body where it cannot produce enough insulin to break down the sugar or it cannot use the insulin produced by the body. It can also be considered as a “slow poison”, which does not show its entire effects immediately but destroys the body step by step, rather slowly. Despite major advances in science and technology, diabetes continues to be a chronic disease, with a thirty-day readmission rate of around 20%, as compared to an average of 12% for the rest of the diseases. Additionally, readmissions cost hospitals a fair amount of money, so the end goal is to identify and reduce the possibility of a readmission. Prevention of patient readmission has been given a greater importance due to large cost involvement.

Introduction

Diabetes is a chronic condition which serves as a forerunner for many other health conditions in the future. It has been estimated that over a quarter of a billion people across the world are suffering from diabetes right now.

We first referred to the research paper "Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records" consisted of a similar case of hospital readmission data. It explained about the number of people suffering from diabetes and the importance of predicting the readmission of the patient, which will further assist in reducing the readmission rate of patients, and the cost of inpatient care. The detailed description on various features present in the dataset gave us a better understanding of the data which we are dealing with.

Objectives

The primary objective of this project is to predict whether the patient will be readmitted to the hospital or not. We have used different classification methods for this purpose. Detailed explanation on the same is present in the below sections. Additionally, we created models to predict, 1) the time a patient is likely to spend in the hospital based on the preliminary diagnoses (data available on day 0), and, 2) future diagnoses (diagnoses 2 and 3).

Data

The diabetes dataset we have used consists of 10,000 records and 52 features. Below given is the detailed description of some of the important features in the dataset.

1. Row ID: Categorical values. Provides a unique identifier for each observation. Missing data 0%
2. Race: Categorical values. Values: African-American Asian Caucasian Hispanic Other. Missing data 2.21%
3. Gender: Categorical values. Values: Male Female. Missing data 0%
4. Age: Categorical values. Values: [0-10) [10-20) [20-30) [30-40) [40-50) [50-60) [60-70) [70-80) [80-90) [90-100). Missing data 0%
5. Weight: Numerical values. Values: Weight of a patient. Missing data 95.92%
6. Admission type ID: Categorical values. Values: Elective Emergency Newborn Not Available Not Mapped Urgent. Missing data 7.21%
7. Discharge disposition ID: Categorical values. Values: 22 levels - Admitted as an inpatient to this hospital ... Not Mapped. Missing data 4.69%
8. Admission source ID: Categorical values. Values: 11 levels - Clinic Referral Court/Law Enforcement Emergency Room HMO Referral Not Available ... Transfer from another health care facility. Missing data 9.36%
9. Time in hospital: Numerical values. Values: Days spent in the hospital by the patient. Missing data 0%
10. Payer code: Categorical values. Values: BC CH CM CP DM HM MC MD OG OT PO SI SP UN WC. Missing data 53.41%
11. Medical specialty: Categorical values. Values: 52 Levels - Anesthesiology-Pediatric Cardiology Cardiology-Pediatric Emergency/Trauma ... Urology. Missing data 41%

12. Number lab procedures: Numerical values. Values: Number of lab procedures done. Missing data 0%
13. Number procedures: Numerical values. Values: Number of procedures done. Missing data 0%
14. Number medications: Numerical values. Values: Number of medications prescribed. Missing data 0%
15. Number outpatient: Numerical values. Values: Number of outpatient visits by the patient. Missing data 0%
16. Number emergency: Numerical values. Values: Number of emergency visits by the patient. Missing data 0%
17. Number inpatient: Numerical values. Values: Number of inpatient visits by the patient. Missing data 0%
18. Diagnoses 1: Categorical values. Values: Different types of diagnoses referred from the ICD-9 codes 457 Levels: 11 110 112 141 150 151 153 154 155 156 157 158 160 161 162 164 171 174 180 182 183 184 185 187 188 ... V71. Missing data 0.02%
19. Diagnoses 2: Categorical values. Values: Different types of diagnoses referred from the ICD-9 codes 429 Levels: 11 110 112 131 135 136 138 150 151 153 154 155 156 157 162 171 173 174 185 188 189 193 196 197 198 ... V85. Missing data 0.59%
20. Diagnoses 3: Categorical values. Values: Different types of diagnoses referred from the ICD-9 codes 460 Levels: 110 112 117 135 138 150 151 153 154 155 162 170 172 174 179 185 188 189 196 197 198 199 200 201 202 ... V85. Missing data 2.08%
21. Number diagnoses: Numerical values. Values: Number of diagnoses done by the patient. Missing data 0%
22. Max glu serum: Categorical values. Indicates the range of the result or if the test was not taken. Values: >200, >300, Norm and None if not measured. Missing data 0%
23. A1Cresult: Categorical values. Indicates the range of the result or if the test was not taken. Values: >7 (if result was greater than 7\%), >8 (if result was greater than 8\%), None (if test not taken), and Norm (if result is normal). Missing data 0%
24. 23 features of medications: Metformin, Repaglinide, Nateglinide, Chlorpropamide, Glimepiride, Acetohexamide, Glipizide, Glyburide, Tolbutamide, Pioglitazone, Rosiglitazone, Acarbose, Miglitol, Troglitazone, Tolazamide, Examide, Citoglipton, Insulin, Glyburide metformin, Glipizide metformin, Glimepiride pioglitazone, Metformin rosiglitazone, Metformin pioglitazone. Values: Down (if the dosage is reduce), No (if the medicine is not given), Steady (if the dosage is steady), Up (if the dosage is increased). Missing data 0%
25. Change: Categorical values. Values: Ch (if change in medicine), No (if no change in medicine). Missing data 0%
26. Diabetes medicine: Categorical values. Values: No Yes. Missing data 0%
27. Readmitted: Categorical values. Values: FALSE TRUE. Missing data 0%

Data Preprocessing

The initial dataset had 52 features and 10,000 observations. The dataset had undergone intense cleaning procedure to be ready for modeling. Few steps taken in the data cleaning process are given below:

1. Features, diagnoses description 1, diagnoses description 2, diagnoses description 3, did not play any role in the classification of readmitted or not, predicting time in hospital, or predicting future diagnoses. Hence, we removed this feature from the dataset.
2. Weight is a very important feature for this dataset, but 95.92% of records did not have the weight of the patient recorded. On the account of so many missing values for weight, we removed this feature from the dataset.
3. Like Weight, Medical specialty and Payer code were two features which had a lot of missing values. So, we removed this feature from the dataset.
4. Left with 46 features and significant amount of missing values, the next step was to impute the data and guess the missing values. We used kNN imputation method for this purpose.
5. Our dataset had a lot of categorical values. We converted the categorical values for various features into numerical values to conduct kNN imputation on the dataset. The categorical values converted are given as below:
    a. Race has five levels and the numerical values given ranged from 1 to 5
    b. Gender has two levels and the numerical values given are 1 and 2
    c. Age has 10 levels (values ranging from [0-10) to [90-100)] and the numerical values given are 5, 15, 25, ..., 95
    d. Admission type ID has five levels and the numerical values given are 1 to 5
    e. Discharge disposition ID has 22 levels. We first categorized these 22 levels in 6 different categories, namely, discharged, expired, not mapped, left AMA, hospice, and admitted. The numerical values for these 6 categories were 1 to 6, respectively.
    f. Admission source ID has 11 levels. We first categorized these 11 levels in 5 different categories, namely, physical referral, transfer, court/law enforcement, emergency, and not mapped. The numerical values for these 5 categories were 1 to 5, respectively.
    g. Diagnoses 1, 2 and 3 are categorical variables based on the ICD-9 codes. For example, if the value of a diagnoses is 010, then that refers to "Tuberculosis", which falls under the "Infectious and parasitic diseases" among the ICD-9 codes. Based on the diagnoses value of each record, we reduced the number of levels for diagnoses to 19 per the below given table. The values ranged from 1 to 19. A complete categorization of ICD-9 codes is given below:

Table 1: Description of ICD-9 codes

Disease Category	ICD-9 code
Infectious and parasitic diseases	001-139
Neoplasms	140-239
Endocrine, nutritional and metabolic diseases, and immunity disorders	240-279
Diseases of the blood and blood-forming organs	280-289
Mental disorders	290-319
Diseases of nervous system	320-359

Diseases of sense organs	360-389
Diseases of circulatory system	390-459
Diseases of respiratory system	460-519
Diseases of digestive system	520-579
Diseases of genitourinary system	580-629
Complications of pregnancy, childbirth, and the puerperium	630-679
Diseases of the skin and subcutaneous tissue	680-709
Diseases of the musculoskeletal system and connective tissue	710-739
congenital anomalies	740-759
Certain conditions originating in the perinatal period	760-779
symptoms, signs, and ill-defined conditions	780-799
injury and poisoning	800-999
External causes of injury and supplemental classification	E and V codes

   h. Max glu serum has four levels and the numerical values given are 1 to 4
   i. A1Cresult has four levels and the numerical values given are 1 to 4
   j. 23 features of medication mentioned in the above "Data" section has four levels, and the numerical values given are 1 to 4
6. After converting all the categorical values into numerical values, we imputed the dataset to get the missing values.

The dataset we have now is cleaned, imputed, but we have 46 features available for modeling. We need analyze the significance of the features first before creating the models.

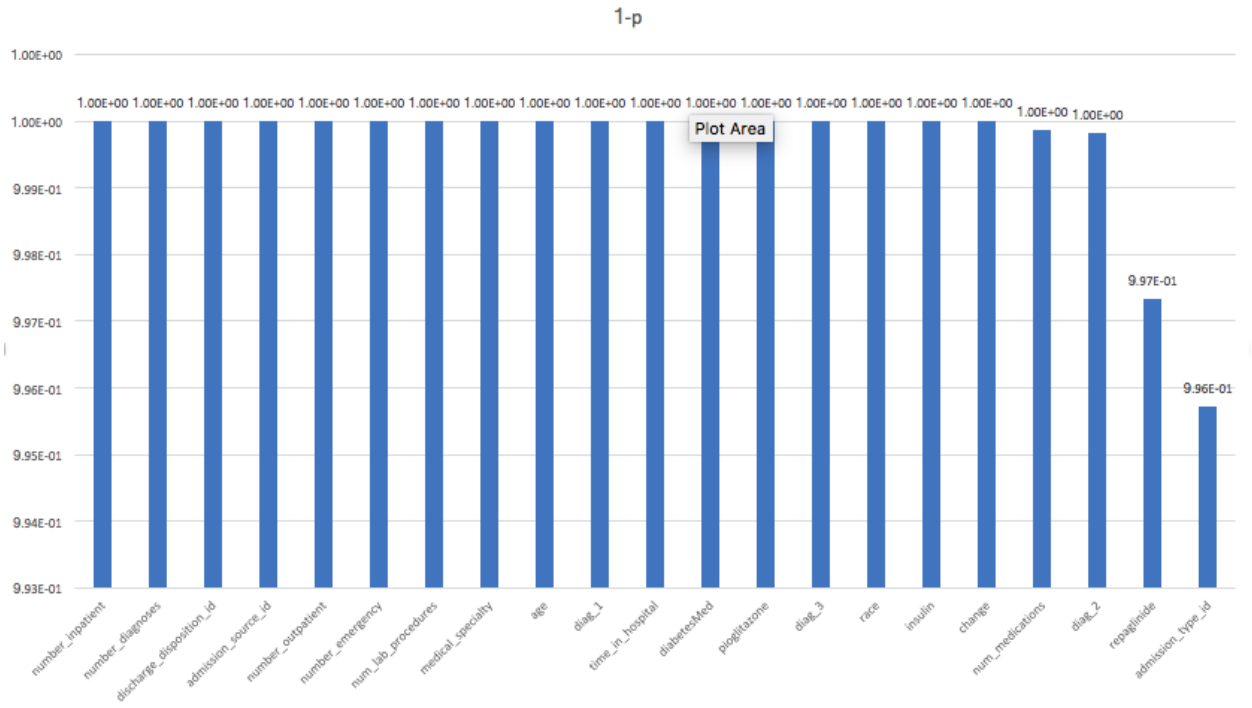
Dimensionality Reduction

Correlation analysis

We conducted the correlation analysis for the main task (readmitted or not), and the two other subtasks on the imputed dataset to identify the significant features. For categorical data, we used, Chi square test, and for numerical data, we used Pearson's coefficient. We used the metric p-value to determine the significance of a feature for a task. If the p value is below a threshold, which is 0.05, then the feature is considered significant.

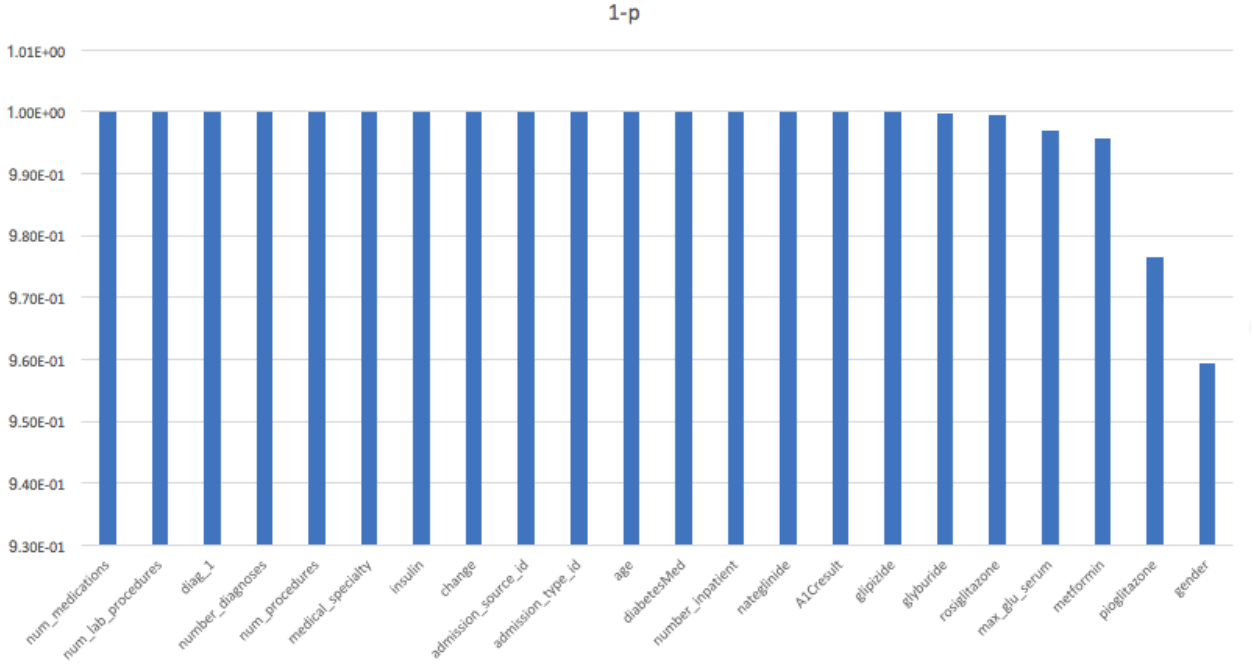
1. For the main task (readmitted or not), the correlation analysis resulted in 21 features which were significant. They are represented in figure 1.

Figure 1: Significant features for main task using correlation analysis



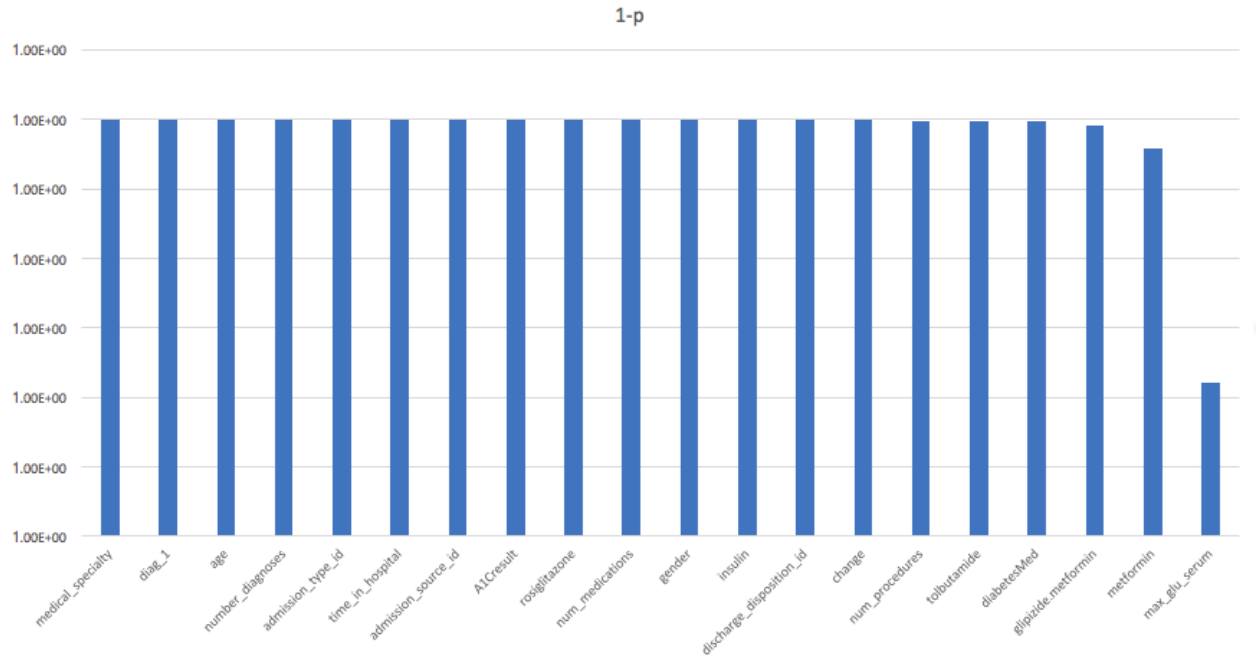
2. For task 1 (time in hospital), the correlation analysis resulted in 22 significant features. They are represented in figure 2

Figure 2: Significant features for task 1 (time in hospital) using correlation analysis



3. For task 2 (future diagnoses), the correlation analysis resulted in 20 significant features. They are represented in figure 3.

Figure 3: Significant features for task 2 (future diagnoses) using correlation analysis



Note: We performed regression analysis for all the three tasks using only the significant features, but the accuracy achieved was not that great, averaging at just about 50%. One of the important reasons to do dimensionality reduction is to reduce the number of features, or to better our feature selection, so that we can create models using as less number of features as possible, so that it is computationally efficient, and makes accurate predictions at the same time.

Class imbalance

To improve our feature selection, we considered the class imbalance issue in our dataset. *Class imbalance is a problem which occurs in the machine learning techniques, where the total number of a class of data (positive) is far less than the total number of another class of data (negative).* After analyzing the different features in the dataset, we eliminated the features which had class imbalance. All the eliminated features are listed below:

Table 2: Features with class imbalance

Feature name	Class distribution
Repaglinide	(”No”: 9870, “Steady”: 112, “Up”:13, “Down”:5)
Nateglinide	(”No”: 9949, “Steady”: 49, “Up”:1, “Down”:1)
Chlorpropamide	(”No”: 9987, “Steady”: 12, “Up”:1, “Down”:0)
Tolbutamide	(”No”: 9997, “Steady”: 3, “Up”:0, “Down”:0)
Acarbose	(”No”: 9968, “Steady”: 31, “Up”:1, “Down”:0)
Miglitol	(”No”: 9995, “Steady”: 3, “Up”:1, “Down”:1)
Tolazamide	(”No”: 9996, “Steady”: 4, “Up”:0, “Down”:0)

glyburide.metformin	(”No”: 9944, “Steady”: 53, “Up”:2, “Down”:1)
glipizide.metformin	(”No”: 9998, “Steady”: 2, “Up”:0, “Down”:0)
Acetohexamide	Only 1 factor level
troglitazone	Only 1 factor level
examide	Only 1 factor level
citoglipton	Only 1 factor level
glimperide.pioglitazone	Only 1 factor level
metformin.rosiglitazone	Only 1 factor level
metformin.pioglitazone	Only 1 factor level

After removing all these features, we are left with 30 features in total in our dataset. They are as follows:

rowID, race, gender, age, admission_type_id, discharge_disposition_id, admission_source_id, time_in_hospital, num_lab_procedures, num_procedures, num_medications, number_outpatient, number_emergency, number_inpatient, diag_1, diag_2, diag_3, number_diagnoses, max_glu_serum, A1Cresult, metformin, glimepiride, glipizide, glyburide, pioglitazone, rosiglitazone, insulin, change

Model Creation

Main task: Readmitted or Not

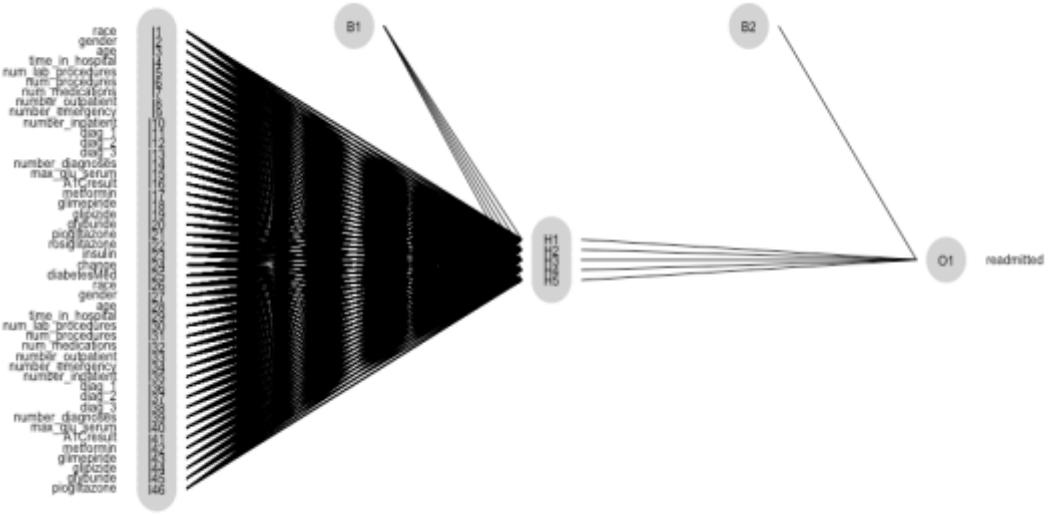
This is a classification problem. To predict whether a patient will be readmitted or not, we created six different classifiers, namely, Support Vector Machines, Generalized logistic regression, Artificial Neural Networks, Random Forest Classifier, Naïve Bayes Classifier, Decision Trees. The features used to create these models are listed in the above section. All the features, except for “rowID” is used in the classification. That makes it 29 features in total.

1. Support Vector Machines is a discriminative classifier formally defined by a separating hyperplane. It is a supervised machine learning algorithm mainly used for classification problems. In this algorithm, we plot each data item as a point in the n-dimensional space, where n is the number of features, with the value of each feature being the value of the predictor coordinate. Then we perform classification by finding the hyperplane that differentiate the two classes very well. We used 10-fold cross validation technique to assess the predictive performance of the model generated. We used “e1071” library in R to generate an SVM model. This classifier achieved an accuracy of 63.95% (compared to the regression model which we used along with the features we obtained from the correlation analysis, this performed much better). Not only is the accuracy of this classifier significantly higher, this classifier was the fastest.

2. Generalized logistic regression is the appropriate regression analysis when the dependent variable is categorical and binary. The goal of the logistic regression is to find the best fitting model to describe the dependent variable and a set if independent variables. We used “glm” method in R to generate this model. The classifier achieved an accuracy of 64.6%. This is the highest accuracy achieved among the models which we have built.

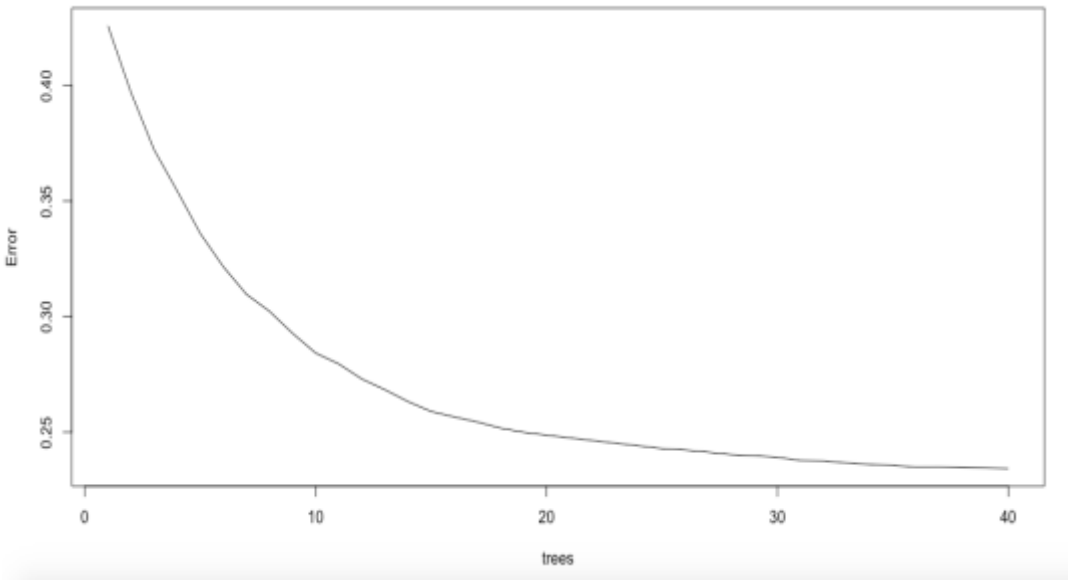
3. Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of many highly-interconnected processing elements (neurons) working in unison to solve specific problems. Artificial neural networks (ANNs) provide a general, practical method for learning real-valued, discrete-valued, and vector-valued functions from examples. Algorithms such as Backpropagation use gradient descent to tune network parameters to best fit a training set of input-output pairs. We used "nnet" library to generate an Artificial Neural Network model. This classifier achieved an accuracy of 63.35%, but this was by far the slowest classifier among the rest. The neural net model build is shown in the below figure:

Figure 4: Neural network model for main task (readmitted or not)



4. Random forest is an ensemble learning method which relies on simple averaging of models and prevents the problem of overfitting. It is a collection of decision trees and the final output is an averaged value of the respective trees. The number of variables are selected in each tree at random, and so is the depth of the tree. These are the parameters to tune in a forest. Random Forest for continuous variables is called a regression tree, and it also gives us the important variables in predicting the target variable. We used the "randomForest" library in R to generate a random forest. This classifier achieved an accuracy of 63.55%. The number of trees we used in this classifier is 40. A graph representing the misclassification error as the more number of trees get generated is shown below in figure 5. As we can see, as the number of trees increase, the error rate of the model keeps decreasing.

Figure 5: Number of trees vs misclassification error



5. Naive Bayes classifier is based on the Bayesian theorem, and is suitable particularly when the dimensionality of the input is high. The Bayesian classification represents a supervised learning method as well as a statistical method for classification. Assumes an underlying probabilistic model and it allows us to capture uncertainty about the model in a principled way by determining probabilities of the outcomes. It can solve diagnostic and predictive problems. Despite its simplicity, this classifier can outperform more sophisticated classifiers. We used "naivebayes" library in R to generate a Naïve Bayes classifier. This classifier achieved an accuracy of 62.15%.

6. Decision tree uses recursive partitioning algorithm, which works by splitting the dataset recursively, where the subset which arise from the split are further split till a predetermined termination occurs. At each step, the split is made based on the independent variable that results in the largest possible reduction in heterogeneity of the predicted variables. We used Information Gain (Entropy) to determine the split at each level. An inherent problem with the decision tree is that, it tends to over fit the data, and to overcome this issue, we have pruned the decision tree to get a generalized model. We used "tree" library in R to generate a decision tree. This classifier achieved an accuracy of 50% with this classifier, which is the worst among the other classifiers.

A table consolidating the results achieved by the above-mentioned model is given below:

Table 3: Main task (Readmitted or not) results

Classifier	Accuracy	Running time (1-Fastest, 6-Slowest)
Support Vector Machines	63.95	1
Generalized Logistic Regression	64.6	3
Artificial Neural Networks	63.35	6
Random Forest	63.55	5
Naïve Bayes	52.15	2
Decision Tree	50	4

Task 1: Time in hospital

This is a regression problem. Here we are trying to determine the duration of stay of a patient on day 0, when the patient gets readmitted, based on the data available at that point of time. Here, we do not consider the features "rowID", "diag_2", "diag_3" for predicting the duration, because we assume that on day 0, the patient is diagnosed with the disease, which is "diag_1". We are trying to predict the duration of stay of the patient after the first diagnoses only.

We created four regression models to predict the duration of stay of a patient, namely, *Step-wise linear regression, Artificial Neural Network, Decision Tree, and Random Forest Regressor.* We used Root mean squared error, as well as, Root mean squared logarithmic error metric to measure the accuracy of these models.

Root mean squared error (RMSE): RMSE is a standard deviation of the residuals. Residuals are a measure of how far from the regression line data points are. RMSE is a measure of how far the data points are spread out. RMSE is just the square root of variance, which is standard deviation. The formula for RMSE is given as below:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Root mean squared logarithmic error (RMSLE): In RMSLE, we take the log of predicted and actual values. So, what changes here is the variance that we are measuring. RMSLE is used when we don't want to penalize huge differences in actual and predicted values when both these values are huge numbers. If both the predicted and actual values are small, then RMSE and RMSLE is same. If either or both predicted and actual values are big, then RMSE is greater than RMSLE. RMSLE becomes almost negligible in this case. The formula for RMSLE is given as below:

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

Our assumption on the data is that, diagnoses 1 is the first diagnoses done on the patient on day 0, when the patient visits the office. Diagnoses 2 and 3 may or may not happen at the later point of time. For the models created below, we do not take diagnoses 2 and diagnoses 3 into consideration while building the model, because we are trying to predict the time spent by the patient based on the diagnoses 1, which is done on day 1, and the other relevant data available. Our dataset now has 30 features (refer class imbalance section), out of which we will not consider "rowID", "diag_2", "diag_3". Now, 27 features remain for this task.

1. Step-wise linear regression is a model that assumes a linear relationship between the target variable and the predictor variables. The target variable is thus written as a linear sum of the predictor variables, along with their individual coefficients. The goal is to estimate these coefficients, also called parameters, by minimizing the sum of squares of errors. Error is the discrepancy between the predicted value and the actual value, of the target variable. In our data, using this technique, we predict the estimated time spent by the patient in the

hospital using the independent variables. Additionally, the step-wise approach has been used in linear regression to see which set of variables are the best fit. *This is a feature selection method in linear regression based on the Akaike information criterion (The Akaike information criterion (AIC) is a relative quality of statistical models for a given set of data. Given a collection of models for the data, AIC value estimates the quality of each model, relative to each of the other models. Hence, AIC provides a means for model selection. Lesser the AIC value, lesser will be the information loss).* Step-wise linear regression gives us a set of different models with different features, and their AIC value. We select the model with the least AIC value.

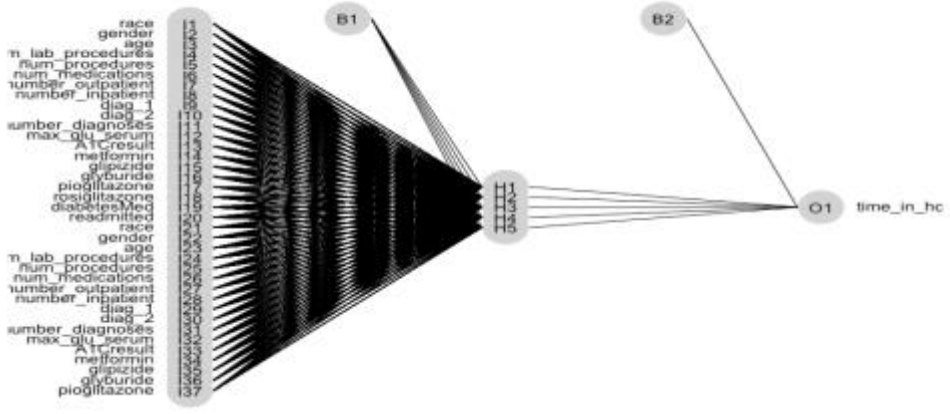
We use the "step" method in R to perform feature selection and get the optimal set of features. Out of the 27 features available, this technique eliminated 10, and we are left with 17 features for model creation. The features which are obtained using this technique are given as follows:

Race, age, gender, number lab procedures, number procedures, number inpatient, number outpatient, number diagnoses, max glu serum, A1C result, metformin, glipizide, glyburide, pioglitazone, rosiglitazone, diabetesMed, readmitted.

We used "lm" method to perform linear regression on the above-mentioned features. The RMSE value for this model is 2.198489, and the RMSLE value is 0.4379768. This is a good rate of prediction by this model for this task.

2. We also built a neural network model for this task. The set of features used were the same as ones we obtained using stepwise linear regression.
We used "nnet" library in R to build this model. The RMSE value obtained for this model is 4.06, and the RMSLE value obtained is 0.96. Compared to linear regression, this is not a good rate of prediction for this model. Poor performance of a neural network is usually the result of high bias in the model. Training with more data point may fix this issue and help this model to predict better. The neural network model built is shown in the below figure 6:

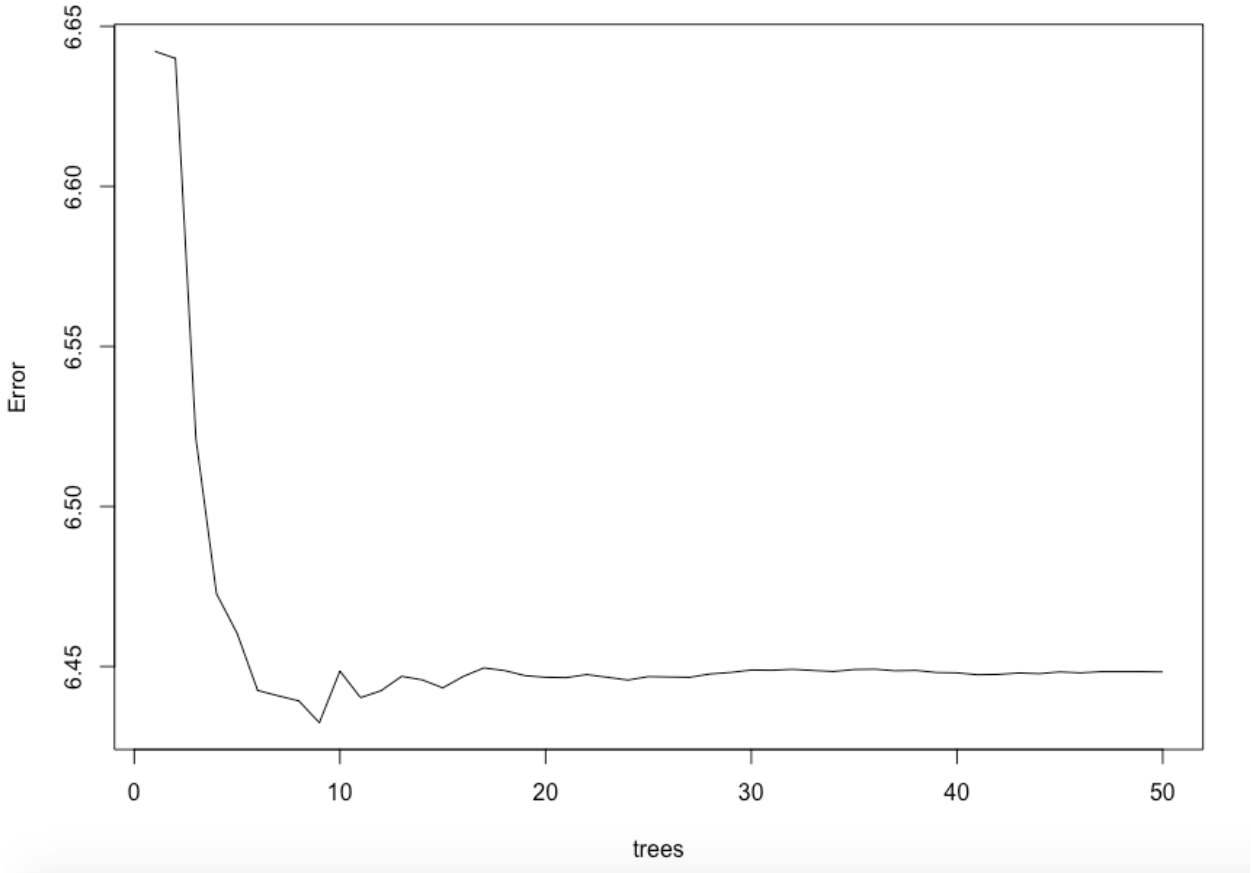
Figure 6: Neural network model for time in hospital



3. We also built a Decision tree model for this task. Again, we used the same set of features which obtained from the step wise linear regression. We used the Information Gain metric to determine the split at each level.
   We used "tree" library in R to generate a decision tree. The RMSE value obtained for this model is 2.28, and the RMSLE value is 0.45. These values are close to the ones we got for linear regression, and hence the rate of prediction is close to that of linear regression.

4. We built a Random Forest Regressor for this task. We used the same set of features for this model too.
   We used the "randomForest" library in R to generate this model. The RMSE value for this model was 2.56, and the RMSLE value was 0.51. The rate of prediction here is good, but it's not better than the Decision tree model. Random forest in basically a collection of decision trees and the final output is the averaged output of all the decision trees generated by the model. Even though the rate of prediction here is good, the expectation was that this model would perform better than the decision tree model. The number of trees we used in this classifier is 50. A graph representing the misclassification error as the more number of trees get generated is shown below in figure 7. As we can see, as the number of trees increase, the error rate of the model keeps decreasing. The error rate was pretty constant after the ntree = 10

Figure 7: Number of trees vs misclassification error



A table consolidating the results achieved by the above-mentioned model is given below:

Table 4: Task 1 (Time in hospital) results

Regression Model	RMSE	RMSLE	Running time (1-Fastest, 6-Slowest)
Stepwise Linear Regression	2.19	0.44	3
Artificial Neural Networks	4.06	0.96	4
Random Forest Regressor	2.56	0.51	2
Decision Tree	2.28	0.45	1

Task: Predict future diagnoses (Diagnoses 2 and 3)

This task can be further divided into two sub problems. 1) Predicting diagnoses 2, after diagnoses 1 is diagnosed, using other relevant data available (Note that diagnoses 3 is not used here), 2) Predicting diagnoses 3, after diagnoses 1 and 2 are diagnosed, using other relevant data available (All the data are used to this prediction). This is classification problem and we built Support Vector Machines, Random Forest, and Artificial Neural Network, models to predict the future diagnoses. In this case, we are considering the dataset with 30 features which we obtained after eliminating the features with class imbalance (check the class imbalance section). Out of these, for predicting diagnoses 2, we will not consider the features, diagnoses 3 and rowID (left with 28 features). And, for predicting diagnoses 3, we will not consider the features, rowID (left with 29 features).

1. We used "e1071" library in R to generate an SVM model for these two sub problems. This classifier achieved an accuracy of 38.67% while predicting diagnoses 2, and 39.44% accuracy while predicting the diagnoses 3. One major issue which we found was that, this model classified the test data into either level 3 (metabolic diseases), or level 8 (circulatory diseases). These two levels are very prominent in the training data.

2. We built a Random Forest classifier for this task. We used the "randomForest" library from R to create this model. We achieved an accuracy of 40.68% while predicting diagnoses 2, and 38.15% while predicting diagnoses 3. Even though the accuracy achieved was like that of an SVM model, Random Forest classifier performed better as it was not biased towards the major classes and tried to classify all the classes available.

3. We also build an Artificial Neural Network model for this task. We used "nnet" library for to create this model. We achieved an accuracy of 39.19% while predicting diagnoses 2, and 37.43% while predicting diagnoses 3. Even though the accuracy achieved is almost at par with the accuracies of before mentioned models, Neural Network was significantly slower to build than the other models.

A table consolidating the results achieved by the above-mentioned model is given below:

Table 4: Task 2 (Future diagnoses) results

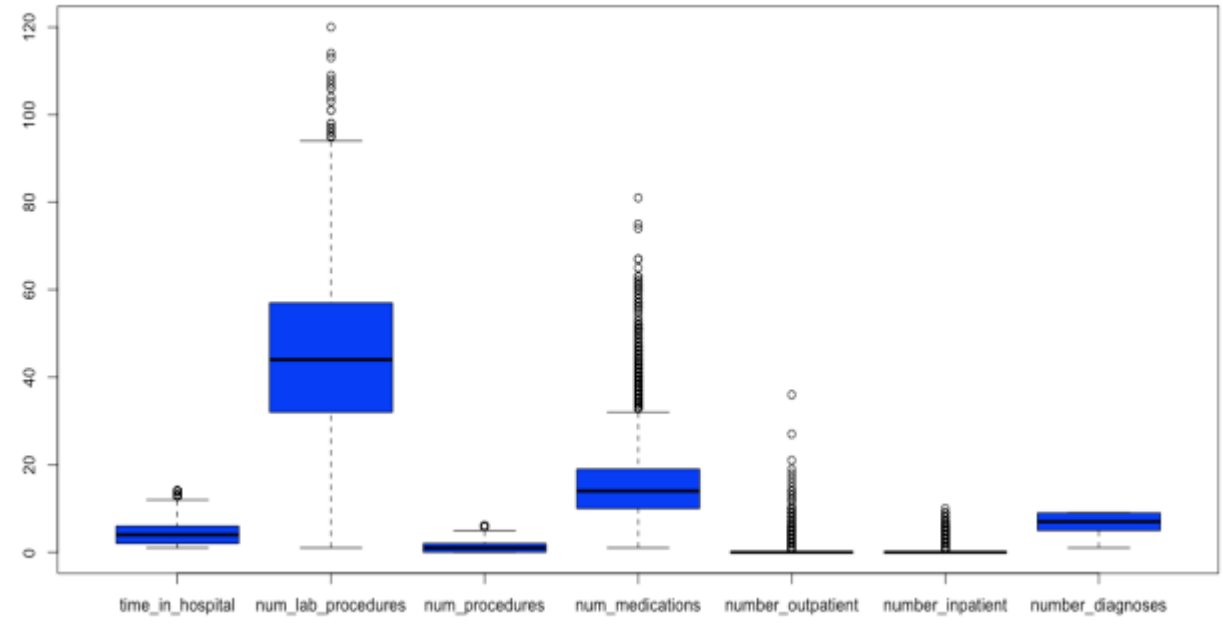
Classifier	Diagnoses 2 Accuracy	Diagnoses 3 Accuracy	Running time (1-Fastest, 6-Slowest)
Support Vector Machine	38.67	39.44	1
Random Forest Regressor	40.68	38.15	2
Artificial neural Network	39.19	37.43	3

Outlier Detection

In data mining, anomaly detection or outlier detection is the process of identification of items, events or observations which do not conform to an expected pattern or other items in the dataset. An outlier is a data object that deviates significantly from the rest of the objects, as it is generated by a different mechanism. Outliers are interesting because they are suspected of not being generated by the same mechanisms as the rest of the data. Therefore, it is important that we explain why the outliers detected are generated by some other mechanism. We can explain the outliers by making some assumptions on the existing data and showing that the outliers move away from those assumptions significantly. Even if the outliers are found, it is not acceptable to drop an observation just because it is an outlier. It can be a legitimate observation, and sometimes even interesting one. It is important to investigate the nature of an outlier before dropping it. Some points we need to look for before deciding to drop an outlier is as follows: 1) If it is obvious that the outlier is due to incorrectly entered or measure data, then drop it, 2) If the outlier does not change the results but does affect some assumptions, the drop the outlier, and 3) If the outlier creates a significant association, then drop the outlier.

The dataset which we have right now (after cleaning and removing irrelevant features), has 30 features. Out of these, many features are categorical. Categorical variables are discrete-valued and do not have any outliers present in it. We tried to detect some outliers among the numerical variables, namely, *time in hospital, number lab procedures, number procedures, number outpatient, number medications, number inpatient, number diagnoses*. Outliers found on these features is represent below:

Figure 8: Outliers for time in hospital, number lab procedures, number procedures, number medications, number outpatient, number inpatient, number diagnoses



As we can see in the above figure, number lab procedures, number medications, number outpatient, and number inpatient. We found a couple of outliers in number lab procedures. They are listed as below:

1. One observation was for a Caucasian male, aged 40 to 50, time spent in hospital is 2 days, and is diagnosed with infectious and parasitic diseases. This patient is not readmitted. But the number of lab procedures done for this patient was 120 (in 2 days spent in the hospital). We believe that this data was entered incorrectly, and the actual number of lab procedures done by the patient may have been 12, but is entered as 120.
2. Another observation was for a Caucasian male, aged 20 to 30, time spent in hospital is 7 days, and is diagnosed with diseases in respiratory system and blood forming diseases. This patient is not readmitted. But the number of lab procedures done for this patient was 113 (in 7 days spent in the hospital). We believe that it is unlikely that a patient did 113 procedures in 7 days.

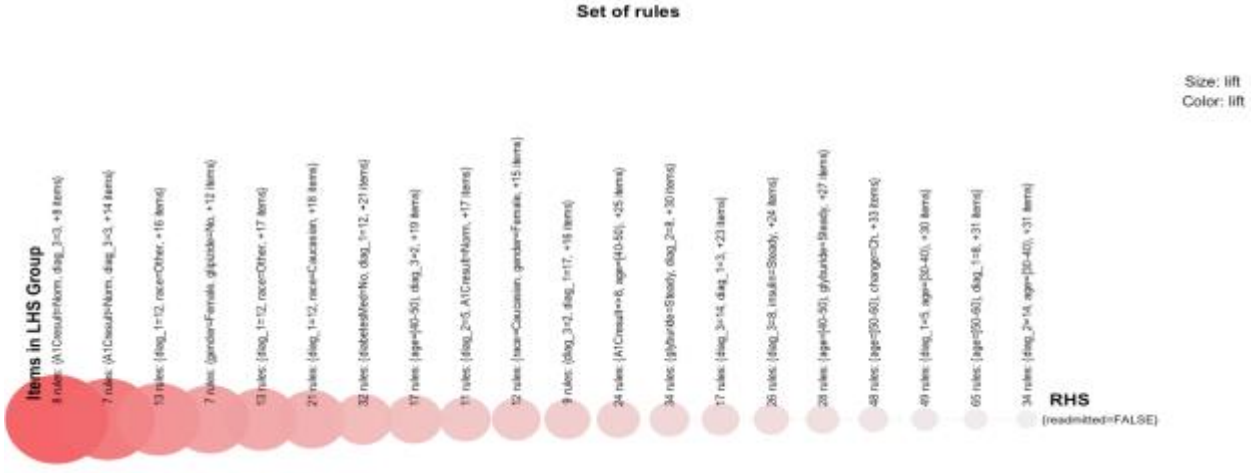
Even after dropping these outliers, we could not observe any significant different in the accuracies of the models, which we built. Hence, we can conclude that these outliers did not create any significant association in the data.

Pattern Mining

Pattern mining uses data mining algorithms to discover interesting, useful and unexpected patterns in the dataset. We define an interesting pattern as a pattern that appears frequently in the database. We have used the Apriori algorithm to mine the patterns in our data. Apriori algorithm is an influential algorithm for mining frequent itemsets for Boolean association rules. Apriori uses a bottom-up approach, where the frequent itemsets are extended one item at a time.

We used “apriori” method from “arules” library in R to mine the patterns. The dataset used was the one which we obtained after removing the irrelevant features (refer the class imbalance section). The support was set at 0.005 and the confidence was set at 0.8. We mined the patterns for readmitted or not, and got 2244 rules. After removing the duplicate rules, the number of rules came down to 475 rules. These rules are visualized as below:

Figure 9: Patterns found for readmitted or not



Some of the patterns which we found are listed below:

1. Readmission rate is higher for a male than for a female by a factor of 4

Figure 10: Rules for readmission for male

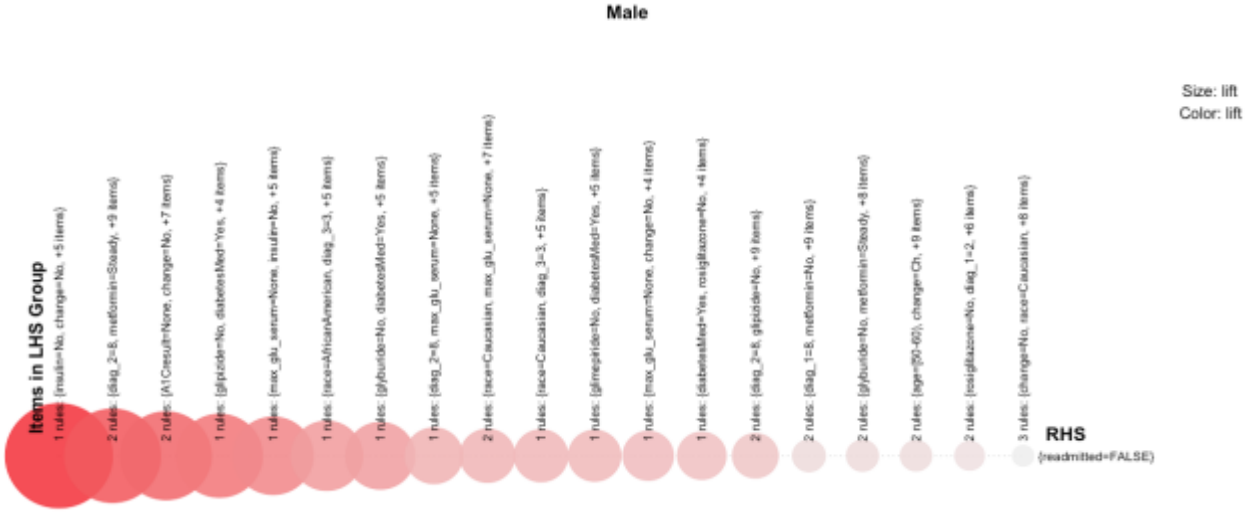
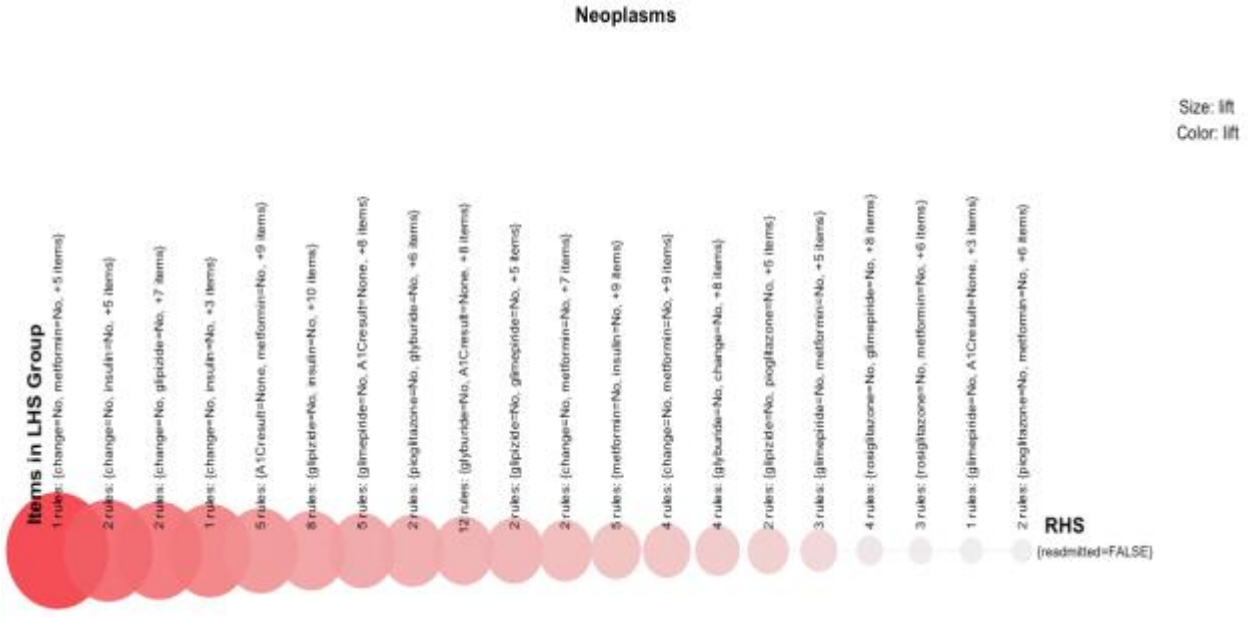


Figure 11: Rules for readmission for female

We can see that the rules for female is much more than that for male.

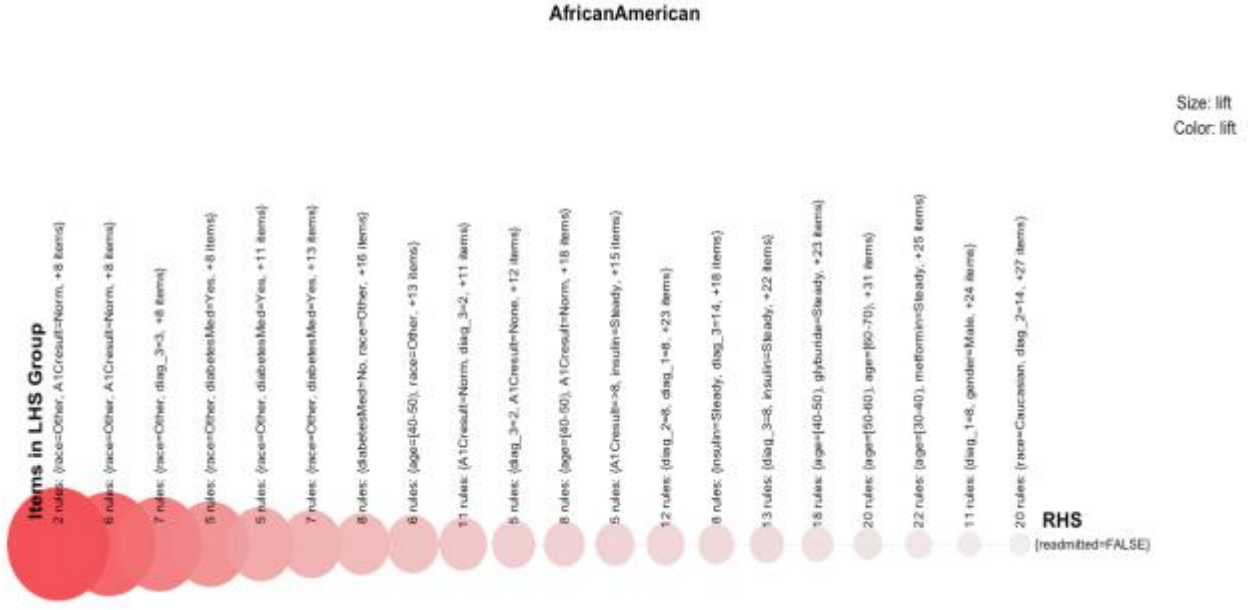
2. Readmission rate is low for a Caucasian female diagnosed with malignant neoplasms

Figure 12: Rules for Caucasian female diagnosed with malignant neoplasms



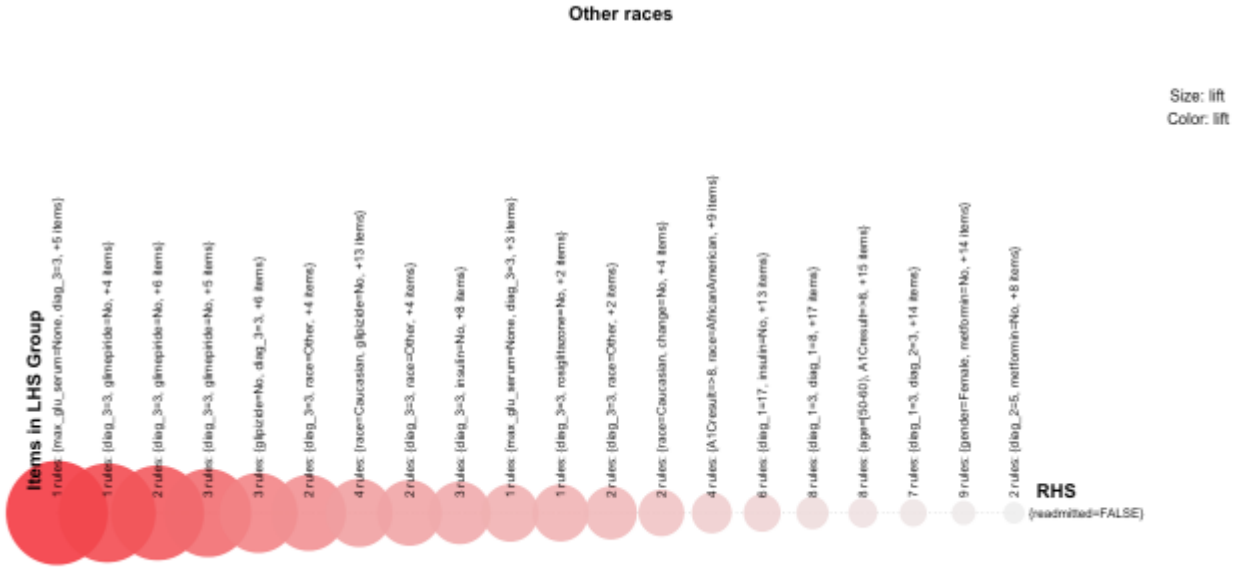
3. Readmission is low among African-Americans, who are diagnosed with either “endocrine, nutritional and metabolic diseases” or “symptoms, signs, and ill-defined conditions”

Figure 13: Readmission for African-American with metabolic diseases or ill-defined conditions



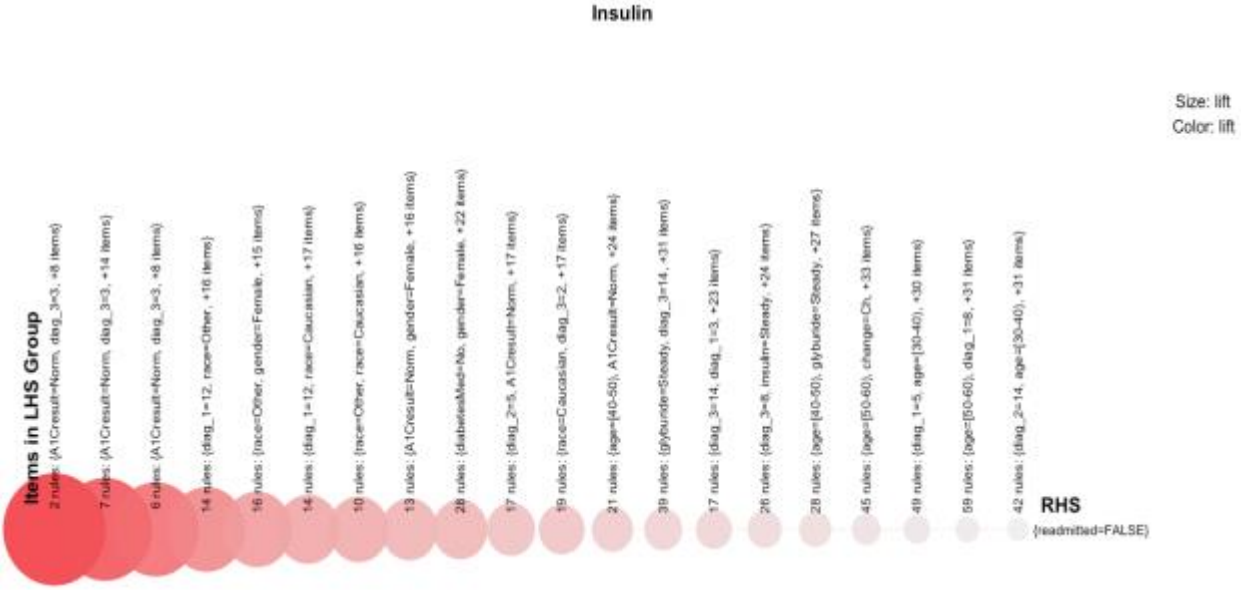
4. Readmission is low for other races, when the patient consumes the diabetes medicine

Figure 14: Readmission for "Other" races who consume diabetes medicine



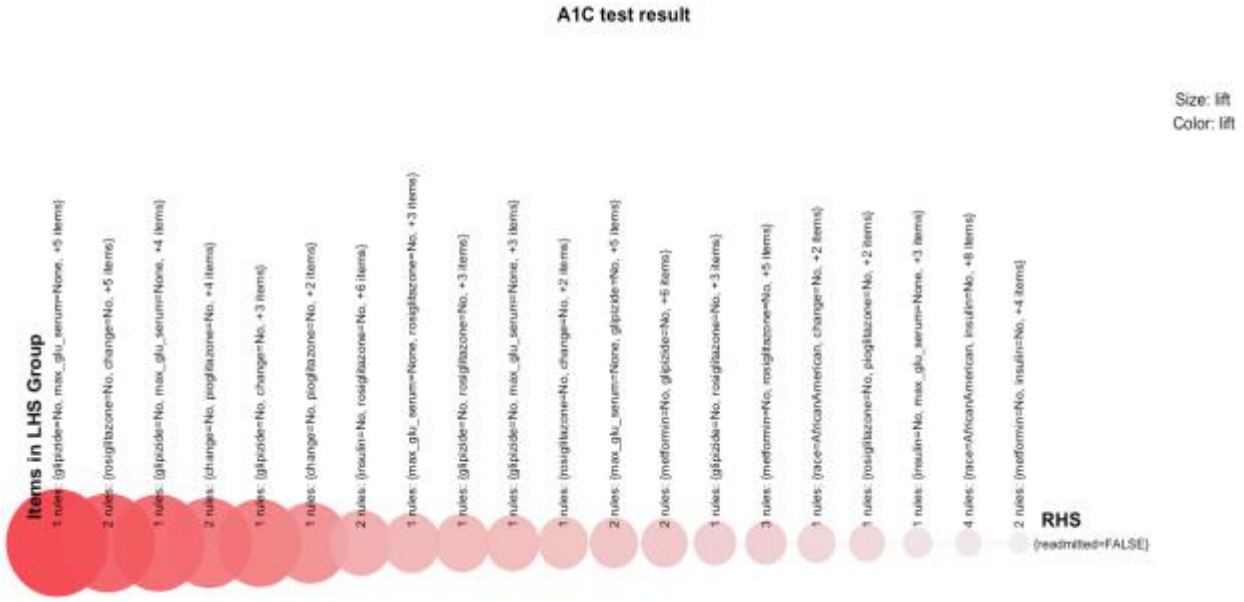
5. Readmission is low when the insulin administered is either Steady (means there is no increase or decrease in the insulin dosage), or if no insulin is administered

Figure 15: Readmission for patients for whom insulin administered is either steady or no insulin is administered



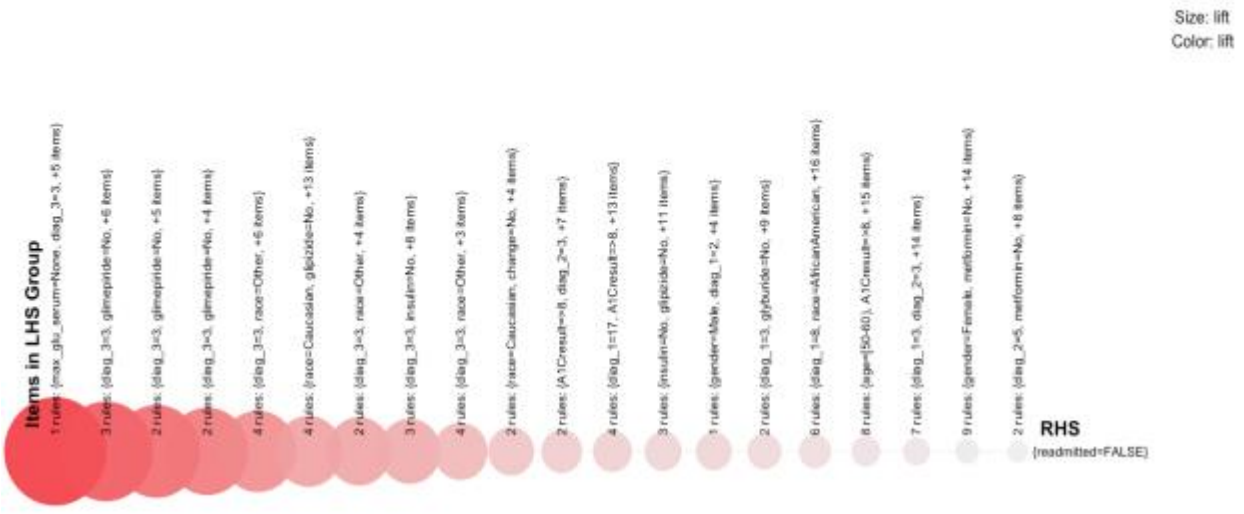
6. Readmission is low when the A1C test result is Normal

Figure 16: Readmission for patients for whom A1C test result is normal



7. Readmission is low in general for patients who take diabetes medicine

Figure 17: Readmission for patients for take diabetes medicine



Cluster Analysis

Cluster Analysis is a multivariate method which aims to classify a sample of data objects based on a set of measured variables into several different groups such that similar data objects are placed in the same group. We used a hierarchical cluster analysis on the dataset using “hclust” method from “stats” library in R. Distance metric used was Euclidean distance, and the agglomeration method used is ward.D2. We formed 10 clusters on features like diagnoses 1, diagnoses 2, diagnoses 3, number of lab procedures, number diagnoses. Cluster analysis on this data did not heed any interesting results for us. Visualizing the clusters was difficult due to the huge amount of data available. The cluster obtained is visualized using the below in figure 18. We also tried to visualize the different clusters using tableau to understand what kind of values are present in these clusters. This visualization is show in figure 19. As we can see in figure 19, one of the biggest clusters is present at the left most top corner represents, cluster 3, which consists of patients who are Caucasian female, 489 of whom are readmitted. Another big cluster consists of Caucasian male, 423 of whom are readmitted. Clusters representing African-American and other races are small.

Figure 18: Cluster dendogram on the dataset

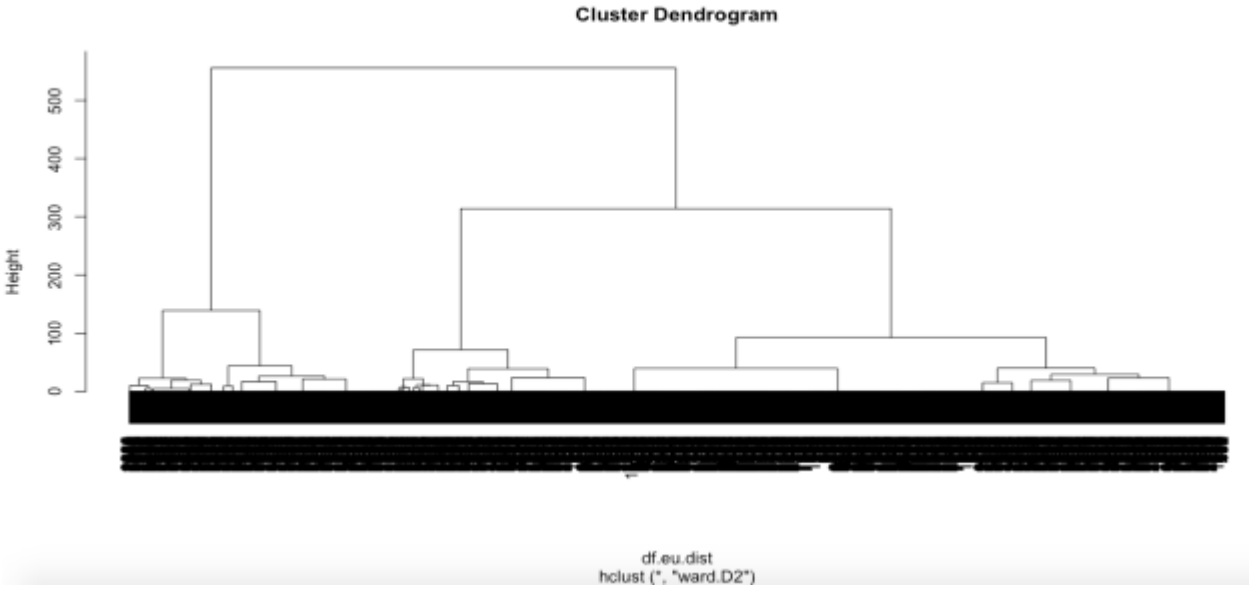
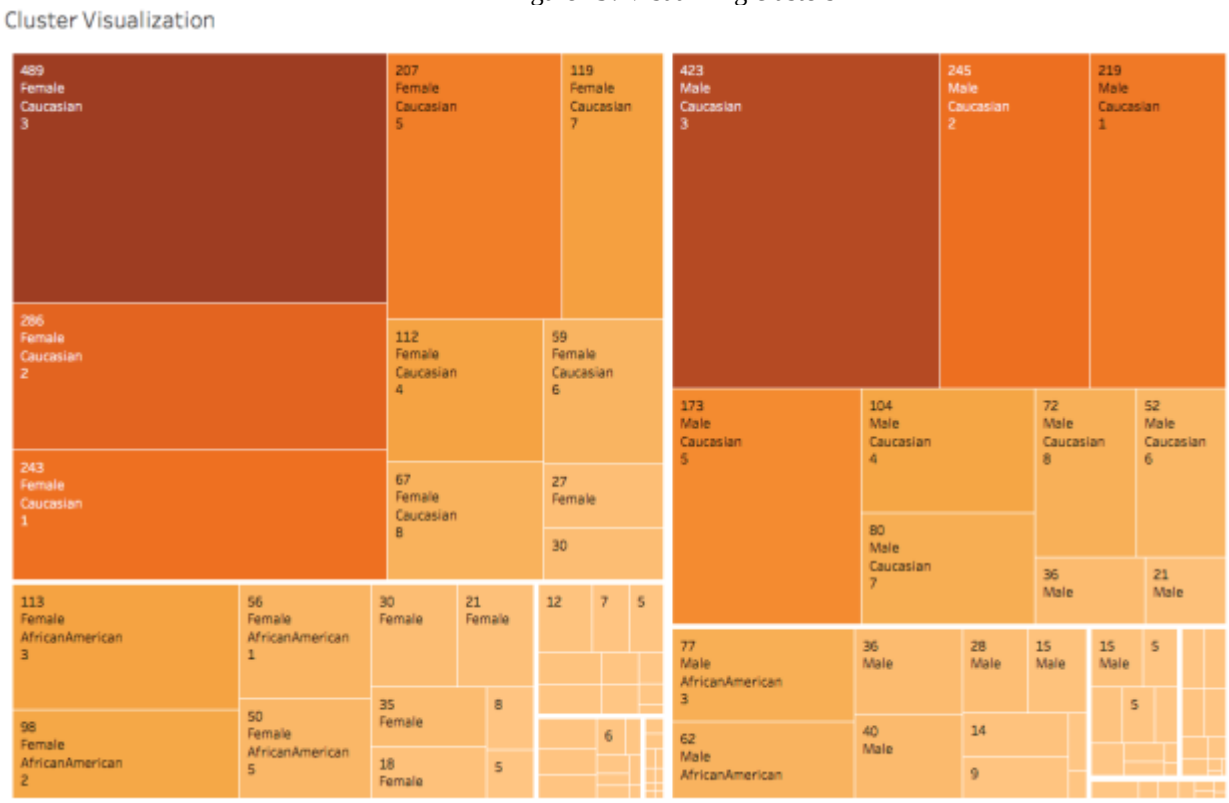


Figure 19: Visualizing clusters



Conclusion

From the above analysis, we can conclude that the readmission rate is high among the Caucasians. The result of A1C test is another good predictor of readmission rate for diabetes patients, which may come in handy in developing strategies to reduce the readmission rates and costs incurred due to the readmission of a patient. If the result of A1C test is Normal, then the readmission rate is low. Readmission rate is low when the insulin administered is a Steady dosage, or if no dosage of insulin is administered. While the readmission rate remained high for male than female, readmission was low for patients who were consuming diabetes medicine.

References

1. https://cran.r-project.org/web/packages/e1071/index.html
2. http://vassarstats.net/matrix2.html
3. https://escience.rpi.edu/data/DA/svmbasic_notes.pdf
4. http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html
5. https://cran.r-project.org/web/packages/nnet/nnet.pdf
6. https://www.linkedin.com/pulse/neural-networks-using-r-jeffrey-strickland-ph-d-cmsp-asep
7. https://cran.r-project.org/web/packages/naivebayes/naivebayes.pdf
8. http://software.ucv.ro/~cmihaescu/ro/teaching/AIR/docs/Lab4-NaiveBayes.pdf
9. Data Mining Concepts and Techniques By Jiawei Han, Micheline Kamber, Jian Pei, Third edition
10. http://www.chioka.in/class-imbalance-problem/
11. https://en.wikipedia.org/wiki/List_of_ICD-9_codes
12. https://www.cs.ucsb.edu/~xyan/papers/dmkd07_frequentpattern.pdf
13. https://www.rroij.com/open-access/an-analytical-study-on-sequential-pattern-miningwith-progressive-database.pdf
14. https://core.ac.uk/download/pdf/11779693.pdf
15. Discovery statistics using R by Andy Field, Jeremy Miles, Zoe Field

Appendix

1. Cleaned and imputed dataset, after removing irrelevant features –

2. Code base for our project (Code is pasted below. But for better visualization, attached the github link https://github.com/fzachariah/Machine-Learning/blob/master/Code%20for%20Predictive%20analysis%20on%20Hospital%20Readmission%20data.pdf) –

```
library(randomForest)
library(nnet)
library(SDMTools)
library(tree)
library(naivebayes)
library(e1071)
library(Metrics)
library(neuralnet)
library(class)
library(stats)
library(arulesViz)

classify <- function(x) {
	value = -1
	if (startsWith(x, "E"))
	{
		value = 19
	}
	else if (startsWith(x, "V"))
	{
		value = 20
	}
	else
	{
		x = as.numeric(x)
		if (x < 140) {
			value = 1
		}
		else if (x >= 140 && x < 240) {
			value = 2
		}
		else if (x >= 240 && x < 280) {
			value = 3
		}
		else if (x >= 280 && x < 290) {
			value = 4
		}
		else if (x >= 290 && x < 320) {
			value = 5
		}
		else if (x >= 320 && x < 360) {
			value = 6
		}
		else if (x >= 360 && x < 390) {
			value = 7
		}
		else if (x >= 390 && x < 460) {
```

```
                value = 8
            }
            else if (x >= 460 && x < 520) {
                value = 9
            }
            else if (x >= 520 && x < 580) {
                value = 10
            }
            else if (x >= 580 && x < 630) {
                value = 11
            }
            else if (x >= 630 && x < 680) {
                value = 12
            }
            else if (x >= 680 && x < 710) {
                value = 13
            }
            else if (x >= 710 && x < 740) {
                value = 14
            }
            else if (x >= 740 && x < 760) {
                value = 15
            }
            else if (x >= 760 && x < 780) {
                value = 16
            }
            else if (x >= 780 && x < 800) {
                value = 17
            }
            else if (x >= 800 && x < 1000) {
                value = 18
            }
        }
        value
}

maxidx <- function(arr) {
        return(  which(arr == max(arr)) )
}

df <- read.csv("diabetes_clean.csv", header = TRUE, strip.white = TRUE, na.strings = c("NA", "?"," ","."))
df$readmitted <- as.factor(df$readmitted)

train <- sample(1:nrow(df), 0.8*nrow(df)) # Split the data into 80:20 ratio for cross validation
train_data <- df[train,] # Training data
test_data <- df[-train,] # Test data

model.logit <- glm(readmitted~., data=train_data[,-1], family=binomial(link='logit'))
pred.logit <- predict(model.logit,test_data, type = "response")
pred.logit <- ifelse(pred.logit > 0.5, 1, 0)
pred.logit.2 <- ifelse(pred.logit == 1, "TRUE", "FALSE")
mean(pred.logit==test_data$readmitted) # Accuracy: 64.6%

# Model 2: Random forest
df$readmitted <- ifelse(df$readmitted == 1, "TRUE", "FALSE")
df$readmitted <- as.factor(df$readmitted)
model.rf1 <-randomForest(readmitted~., data=train_data[,-1], ntree=10, na.action=na.exclude,
importance=T,proximity=T)
print(model.rf1) #error rate: 42.08%

model.rf2 <-randomForest(readmitted~., data=train_data, ntree=20, na.action=na.exclude, importance=T,proximity=T)
print(model.rf2) #error rate: 40.21%
```

```
model.rf3 <-randomForest(readmitted~., data=train_data, ntree=30, na.action=na.exclude, importance=T,proximity=T)
print(model.rf3) #error rate: 39.21%

model.rf4 <-randomForest(readmitted~., data=train_data, ntree=40, na.action=na.exclude, importance=T,proximity=T)
print(model.rf4) #error rate: 38.32%

model.rf5 <-randomForest(readmitted~., data=train_data, ntree=50, na.action=na.exclude, importance=T,proximity=T)
print(model.rf5) #error rate: 38.57%

model.rf <- randomForest(readmitted~., data=train_data, ntree=40, mtry = 8, na.action=na.exclude,
importance=T,proximity=T)
pred.rf <- predict(model.rf, test_data)
mean(pred.rf==test_data$readmitted) # Accuracy: 63.55%
plot(model.rf)

model.nn <- nnet(readmitted ~., data=train_data[,-1], size=5, maxit=1000)
pred.nn <- predict(model.nn, test_data,type= "raw")
pred.nn <- ifelse(pred.nn > 0.5, "TRUE", "FALSE")
mean(pred.nn==test_data$readmitted) # Accuracy: 63.35%

#plot(model.nn, cex.val = 0.6, circle.cex = 2, nid = F)

df$readmitted <- as.factor(df$readmitted)
model.tree <- tree(readmitted~., data = train_data[,-1])
pred.tree <- predict(model.tree, test_data)
pred.response <- ifelse(pred.tree > 0.5, "FALSE", "TRUE")
mean(test_data$readmitted != pred.response) # Accuracy: 50%

plot(model.tree)
text(model.tree)

# Model 5: Naive Bayes
train_data$readmitted <- as.factor(train_data$readmitted)
test_data$readmitted <- as.factor(test_data$readmitted)
model.nb <- naive_bayes(train_data[,-1], train_data$readmitted, laplace = 1, usekernel = T, prior = NULL)
pred.nb <- predict(model.nb, test_data, type = "prob", threshold = 0.01, eps = 0.1)
idx.nb <- apply(pred.nb, c(1), maxidx)
actual_result <- ifelse(df$readmitted == df$readmitted[1], 0,1)
mean(idx.nb-1 == actual_result) # Accuracy: 52.15%

# Model 6: SVM
model.svm <- svm(readmitted~., data = train_data, kernel = "linear",
            type = "C-classification", cross = 10, cost = 0.01, gamma = 1000)
pred.svm <- predict(model.svm, test_data, decision.values = F)
mean(pred.svm == test_data$readmitted) # Accuracy: 63.95%

####### Task 1: Time in hospital #######
model.lm.steps <- step(lm(time_in_hospital~., data=train_data), direction = "both")
model.task1.lm <- lm(time_in_hospital ~ race + gender + age + num_lab_procedures +
                       num_procedures + num_medications + number_outpatient + number_inpatient +
                       diag_1 + number_diagnoses + max_glu_serum + A1Cresult +
                       metformin + glipizide + glyburide + pioglitazone + rosiglitazone +
                       diabetesMed + readmitted, data = train_data)
pred.task1.lm <- predict(model.task1.lm,test_data)
rmse(test_data$time_in_hospital,pred.task1.lm) #RMSE: 2.198489
rmsle(test_data$time_in_hospital,pred.task1.lm) #RMSLE: 0.4379768

# Model 2: Neural networks
model.task1.nn <- nnet(time_in_hospital ~ race + gender + age + num_lab_procedures +
                         num_procedures + num_medications + number_outpatient + number_inpatient +
                         diag_1 + number_diagnoses + max_glu_serum + A1Cresult +
```

```
                    metformin + glipizide + glyburide + pioglitazone + rosiglitazone +
                    diabetesMed + readmitted, data=train_data, size=5, maxit=1000)
pred.task1.nn <- predict(model.task1.nn, test_data)
rmse(test_data$time_in_hospital, pred.task1.nn) #4.061443
rmsle(test_data$time_in_hospital, pred.task1.nn) #0.9621453

plot(model.task1.nn)

# Model 3: Decision tree
model.task1.tree <- tree(time_in_hospital ~ race + gender + age + num_lab_procedures +
                     num_procedures + num_medications + number_outpatient + number_inpatient +
                     diag_1 + diag_2 + number_diagnoses + max_glu_serum + A1Cresult +
                     metformin + glipizide + glyburide + pioglitazone + rosiglitazone +
                     diabetesMed + readmitted, data = train_data)
pred.task1.tree <- predict(model.task1.tree, test_data)
rmse(test_data$time_in_hospital,pred.task1.tree) #2.276108
rmsle(test_data$time_in_hospital,pred.task1.tree) #0.4512615

plot(model.task1.tree)
text(model.task1.tree)

#Model 4: Random Forest
model.task1.rf1 <-randomForest(time_in_hospital ~ race + gender + age + num_lab_procedures +
                         num_procedures + num_medications + number_outpatient + number_inpatient +
                         diag_1 + number_diagnoses + max_glu_serum + A1Cresult +
                         metformin + glipizide + glyburide + pioglitazone + rosiglitazone +
                         diabetesMed + readmitted,
                       data=train_data,ntree=10, na.action=na.exclude, importance=T,proximity=T)
print(model.task1.rf1) # % Var explained: 13.57%

model.task1.rf2 <-randomForest(time_in_hospital ~ race + gender + age + num_lab_procedures +
                         num_procedures + num_medications + number_outpatient + number_inpatient +
                         diag_1 + number_diagnoses + max_glu_serum + A1Cresult +
                         metformin + glipizide + glyburide + pioglitazone + rosiglitazone +
                         diabetesMed + readmitted,
                       data=train_data,ntree=20, na.action=na.exclude, importance=T,proximity=T)
print(model.task1.rf2) # % Var explained: 24.02%

model.task1.rf3 <-randomForest(time_in_hospital ~ race + gender + age + num_lab_procedures +
                         num_procedures + num_medications + number_outpatient + number_inpatient +
                         diag_1 + number_diagnoses + max_glu_serum + A1Cresult +
                         metformin + glipizide + glyburide + pioglitazone + rosiglitazone +
                         diabetesMed + readmitted,
                       data=train_data,ntree=30, na.action=na.exclude, importance=T,proximity=T)
print(model.task1.rf3) # % Var explained: 28.21%

model.task1.rf4 <-randomForest(time_in_hospital ~ race + gender + age + num_lab_procedures +
                         num_procedures + num_medications + number_outpatient + number_inpatient +
                         diag_1 + number_diagnoses + max_glu_serum + A1Cresult +
                         metformin + glipizide + glyburide + pioglitazone + rosiglitazone +
                         diabetesMed + readmitted,
                       data=train_data,ntree=40, na.action=na.exclude, importance=T,proximity=T)
print(model.task1.rf4) # % Var explained: 30.35%

model.task1.rf5 <-randomForest(time_in_hospital ~ race + gender + age + num_lab_procedures +
                         num_procedures + num_medications + number_outpatient + number_inpatient +
                         diag_1 + number_diagnoses + max_glu_serum + A1Cresult +
                         metformin + glipizide + glyburide + pioglitazone + rosiglitazone +
                         diabetesMed + readmitted,
                       data=train_data,ntree=50, na.action=na.exclude, importance=T,proximity=T)
print(model.task1.rf5) # % Var explained: 31.43%
```

```
model.task1.rf <- randomForest(time_in_hospital~gender+readmitted+change+diabetesMed+glimepiride,
                    data=train_data, ntree=50,mtry = 27, na.action=na.exclude, importance=T,
                    proximity=T)
pred.task1.rf <- predict(model.task1.rf, test_data)
rmse(test_data$time_in_hospital,pred.task1.rf)#2.558702
rmsle(test_data$time_in_hospital,pred.task1.rf)#0.5073352

plot(model.task1.rf)

# Model 1: SVM
test_data$diag_3 <- as.factor(test_data$diag_3)

model.task2.svm.1 <- svm(diag_2~., train_data[,-c(1,14)])
pred.task2.svm.1 <- predict(model.task2.svm.1, newdata = test_data)
mean(round(pred.task2.svm.1)==test_data$diag_2)

model.task2.svm.2 <- svm(diag_3~., train_data[,-c(1)])
pred.task2.svm.2 <- predict(model.task2.svm.2, newdata = test_data)
mean(round(pred.task2.svm.2)==test_data$diag_3)

# Model 2: Random forest

model.task2.rf.1 <-randomForest(x=train_data[,-c(1,14)], y=train_data$diag_2, ntree=10, na.action=na.exclude,
importance=T,
            proximity=T)
print(model.task2.rf.1)#23.17% error

model.task2.rf.2 <-randomForest(x=train_data[,-c(1,14)], y=train_data$diag_2, ntree=20, na.action=na.exclude,
importance=T,
            proximity=T)
print(model.task2.rf.2)#14.69% error

model.task2.rf.3 <-randomForest(x=train_data[,-c(1,14)], y=train_data$diag_2, ntree=30, na.action=na.exclude,
importance=T,
            proximity=T)
print(model.task2.rf.3)#10.43% error

model.task2.rf.4 <-randomForest(x=train_data[,-c(1,14)], y=train_data$diag_2, ntree=40, na.action=na.exclude,
importance=T,
            proximity=T)
print(model.task2.rf.4)#9.88% error

model.task2.rf.5 <-randomForest(x=train_data[,-c(1,14)], y=train_data$diag_2, ntree=50, na.action=na.exclude,
importance=T,
            proximity=T)
print(model.task2.rf.5)#6.1% error

model.task2.rf.6 <-randomForest(x=train_data[,-c(1,14)], y=train_data$diag_2, ntree=60, na.action=na.exclude,
importance=T,
            proximity=T)
print(model.task2.rf.6)#6.13% error

#mtry.2 <- tuneRF(train_data, train_data$diag_2, ntreeTry=50, stepFactor=1.5,
#           improve=0.01, trace=TRUE, plot=TRUE)
model.task2.rf <- randomForest(diag_2~., data=train_data, ntree=50,mtry = 10, na.action=na.exclude, importance=T,
                    proximity=T)

pred.task2.rf <- predict(model.task2.rf, test_data)
mean(as.character(pred.task2.rf)== as.character(test_data$diag_2)) # 40.68%

#mtry.3 <- tuneRF(train_data, train_data$diag_3, ntreeTry=50, stepFactor=1.5,
#            improve=0.01, trace=TRUE, plot=TRUE)
```

```
model.task2.rf.2 <- randomForest(diag_3~., data=train_data, ntree=50,mtry = 5, na.action=na.exclude, importance=T,
                        proximity=T)
pred.task2.rf.2 <- predict(model.task2.rf.2, test_data)
mean(as.character(pred.task2.rf.2)== as.character(test_data$diag_3)) # 38.15%
train_data$diag_2 <- as.factor(train_data$diag_2)
test_data$diag_2 <- as.factor(test_data$diag_2)
train_data$diag_3 <- as.factor(train_data$diag_3)
test_data$diag_3 <- as.factor(test_data$diag_3)
model.task2.nn.1 <- nnet(train_data$diag_2 ~ ., data=train_data[,-c(1,14)], size=5, maxit=1000)
pred.task2.nn.1 <- predict(model.task2.nn.1,newdata = test_data[,-1], type = "class")
mean(as.character(pred.task2.nn.1)==as.character(test_data$diag_2)) # 39.19%

model.task2.nn.2 <- nnet(train_data$diag_3 ~ ., data=train_data[,-c(1)], size=5, maxit=1000)
pred.task2.nn.2 <- as.factor(predict(model.task2.nn.2,newdata = test_data, type = "class"))
mean(as.character(pred.task2.nn.2)==as.character(test_data$diag_3)) # 37.43%

df$readmitted <- ifelse(df$readmitted == "TRUE", 1, 0)
df.eu.dist <- dist(df[,c(12,27)], method = "euclidean")
hClust1 <- hclust(df.eu.dist, method = "ward.D2")

plot(hClust1)

####### Outlier detection #######
boxplot(df[,c(5,6,7,8,9,11,15)], col = "blue")

####### Pattern mining #######
df.initial <- read.csv("10kDiabetes.csv", header = TRUE, strip.white = TRUE, na.strings = c("NA", "?"," ","."))
mining <- df
mining$age <- df.initial$age
mining$diag_1 <- as.factor(mining$diag_1)
mining$diag_2 <- as.factor(mining$diag_2)
mining$diag_3 <- as.factor(mining$diag_3)
mining[,c(1, 5, 6, 7, 8, 9, 10, 11, 15)] <- NULL
mining$readmitted <- df.initial$readmitted
mining$readmitted <- as.factor(mining$readmitted)
rules1 <- apriori(mining, parameter=list(minlen=2,supp=0.005,conf=0.8),
            appearance=list (rhs=c("readmitted=FALSE","readmitted=TRUE"), default="lhs"))
rules1.sort <- sort(rules1, by="lift")
subset.matrix<-is.subset(rules1.sort,rules1.sort)
subset.matrix[lower.tri(subset.matrix,diag=T)] <- 0
redudant<-colSums(subset.matrix) >= 1
rules1.pruned <- rules1.sort[!redudant]
rules.sub <- subset(rules1.pruned, subset = lhs %pin% "Male" & rhs %pin% "FALSE")

plot(rules.sub,method="grouped",measure = "lift", control = list(main="Diabetes
medicine",cex=.8,itemLabels=T,arrowSize=0))
```