**DS6372 – Project 2**

# **Introduction**

Unplanned patients’ readmission often builds up patient stress, hospital resources, and expenditure. Although it’s not uncommon for diabetic patients to get readmitted within 30 days, some of those readmissions may be predictable and thus may be avoided. In this project, we’ll be using “*Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records*” [1] as a reference, in which we’ll be studying the details of each diabetic hospital readmission record ([dataset](https://github.com/chiawang/DS6372_Project2_Group3/blob/master/diabetic_data.csv)) to gain better insight into the likelihood of hospital readmission after the patient’s 1st discharge

For the hospital, understanding the probability of each diabetic patient’s readmission will help the hospital to have better planning and save costs. Diabetic patients will also be benefited as they know where to look for early signs of hospital readmission. In addition, this helps the patients to make better the decision on time to spend in the hospital. Furthermore, doctors can utilize multiple factors related to hospital readmission to determine the best treatment plan for the patients. These same factors can be built into a model for the hospital readmission prediction that assists in taking some of the guesswork out of treatment plan.

The purpose of this paper is to provide a predictive statistical analysis of diabetic patients' hospital readmission. The analysis has two main objectives with the dataset. The first objective is to build a regression model using the dataset and interpret the regression model. The second objective is to use the model from objective one as a baseline and perform additional competing models to demonstrate and improve the prediction on hospital readmission.

## **Data Description**

The data used for this analysis comes from the *UCI Machine Learning Repository*. (<https://archive.ics.uci.edu/ml/datasets/diabetes+130-us+hospitals+for+years+1999-2008>)

The dataset contains clinical patient records at 130 US hospitals from 1999 through 2008. The total data set contains 10,1766 observations with 50 features or variables. All the clinical datasets were collected from diabetic patients only.

These 50 features contain diabetic patients’ information that may contribute to the likelihood of hospital readmission. For example, the datasets include information such as race, gender, age, readmission within 30 days, time in hospital just to name a few.

The dataset also contains a mix of categorical variables and numeric variables. The categorical variables indicate provides the patient details such as race, gender, age, readmission within 30 days just to name a few. The numerical variables provide information such as time in the hospital, diag 1, diag2, diag3, number of procedures, etc. Here’s the link that provides a dictionary of elements used in the dataset.  
<https://www.hindawi.com/journals/bmri/2014/781670/tab1/>

## **Exploratory Data Analysis (EDA)**

For the EDA, we first go through the dataset and replace ‘?’ with ‘NA’, and check variables with missing values.   
  
After that, we review the EDA to determine which features, if any, could be removed from the dataset due to the frequency of missing data. It was determined that the following features could be removed:

* Weight – actual weight of each diabetic patient (*nearly 96% of the data were missing*)
* Medical Specialty – medical practice that was used on the patients (*49% of the data were missing*)
* Payer code – this is the insurance label used to identify pay rate for each patient (*nearly 40% of the data were missing*)

Next, we remove observations where the discharge disposition is related to hospice or death since these will not add to the possibility of being readmitted.

* 11 – Expired
* 13 – Hospice/Home
* 14 – Hospice/Medical Facility
* 19 – Expired at Home
* 20 – Expired at Medical Facility
* 21 – Expired at Unknown

In addition, we notice many patients have multiple admissions in the dataset. The paper “*Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records*” [1] suggests using only the first one to satisfy the independence assumption. Therefore, we filter and only use the first encounter (the lowest for a given member) for the analysis.

To further help with analysis, we create buckets for the ICD-9 diagnosis codes. We use the mapping defined based on the following (<https://www.hindawi.com/journals/bmri/2014/781670/tab2/>).   
  
Additionally, since age is a factor with 10 levels. We decided to bucket age into 3 different groups, (0-30 yr, 30-60 yr, 60-100 yr) in a new feature, and remove the old feature. There are 2 medications that have only 1 factor level, “examide” and “citogliption”. We thus remove those two from the dataset. Since encounter\_id and patient\_nbr are identifiers, we remove those as well.

At the end, we clean up race features NAs and label them as missing. Lastly, we create an outcome column that is Yes if admitted within 30 days, and No for everything else. Then remove the readmitted column from the original dataset.

The final dataset consists of 69,990 observations with 43 features. The EDA summary statistics (Table 1) now shows that there were no missing values within the data set The data set was clean and contained no missing values.

Additional analysis was performed to investigate feature correlation. Through a correlation plot (Figure 4) it was determined that the features: Age, Time in Hospital, Number of Diagnoses, and race are factors with a higher correlation to hospital readmission when compared to other factors.

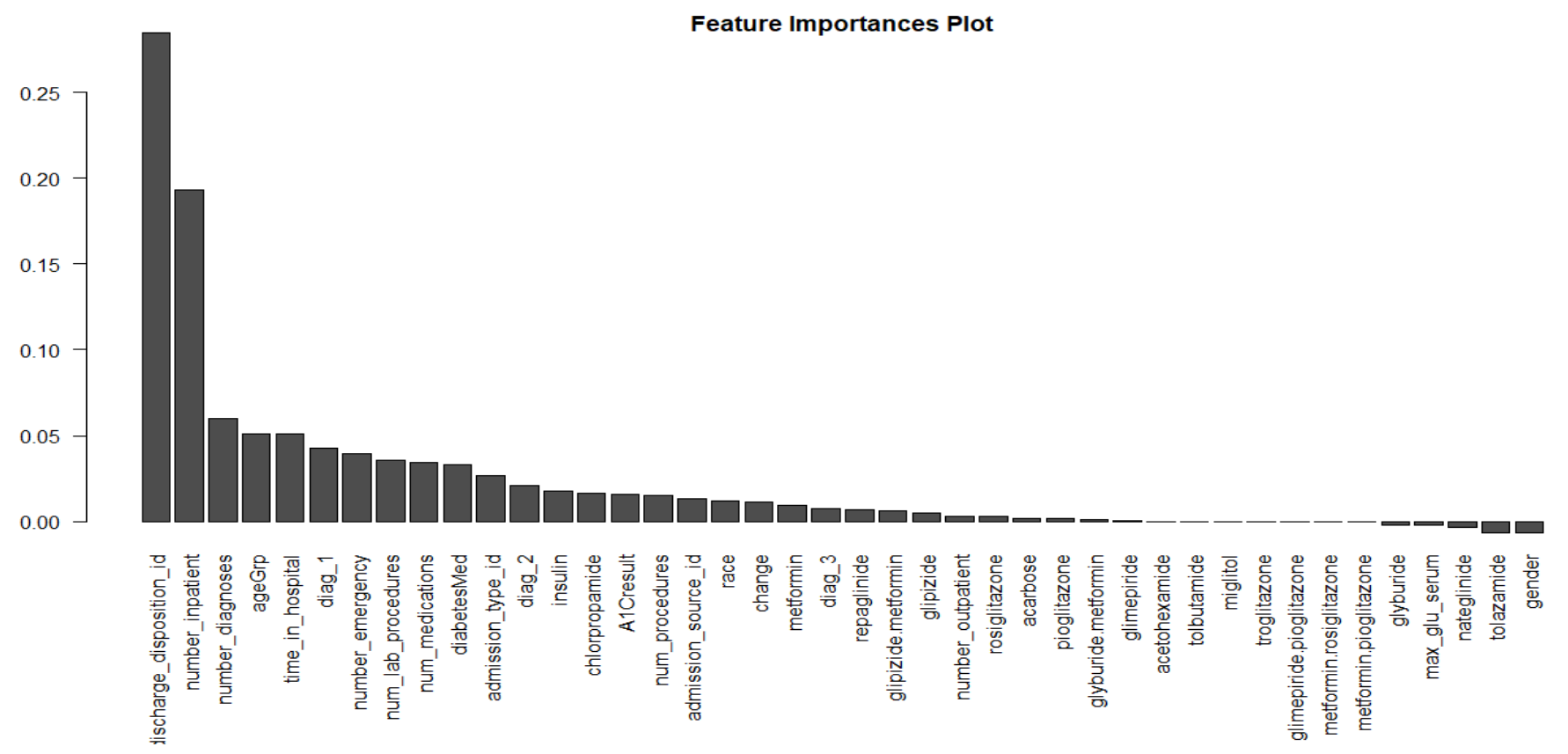
|  |  |  |
| --- | --- | --- |
|  | Observations | Figure |
| Age vs Outcome | The bulk of the patients are in the age range of 40 - 90 years. The proportion of readmits increases from the lowest bucket up to the 80s bucket. | As indicated in Figure 1 |
| Time in Hospital VS Outcome | This is the number of days a patient spent in the hospital. The distribution is at max at 3 days, then tails off at the time increases. The maximum observed value is 14 days. | As indicated in Figure 2 |
| Number of Diagnoses VS Outcome | There seems to be a correlation between readmittance and the number of diagnoses. | As indicated in Figure 3 |

## **Interpretative Modeling**

## **Problem Statement**

The goal of analysis Objective 1 is to use the EDA above to build a logistic regression model. Here’s we’ll include an interpretation of the regression model coefficients, confidence intervals, and hypothesis testing.

## **Feature Selection**

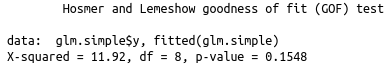
We used two methods to assist in feature selection: LASSO regression and random forest importances. A plot of the random forest importances shows that discharge\_disposition\_id and number\_inpatient features are assigned the highest importance with the other features tailing off. These two features also receive the highest regression coefficients when fit with LASSO regression. These two features where selected because of the agreement of the two feature selection techniques.  


In addition to the two features noted previously, number\_diagnoses and diag\_1 were also included. These values scored lower in terms of feature importance, but exhibited correlation with the outcome. The correlation between outcome and number\_diagnoses is shown visually in Figure 3.

A logistic model trained based on these 4 features received an AUC score of 0.640 when predicting on a test set. For reference, a logistic model trained on the top 12 parameters from LASSO received an AUC score of 0.646. Thus, much of the explainable variation appears to be captured in the model with 4 predictors. This model will be used for interpretation.

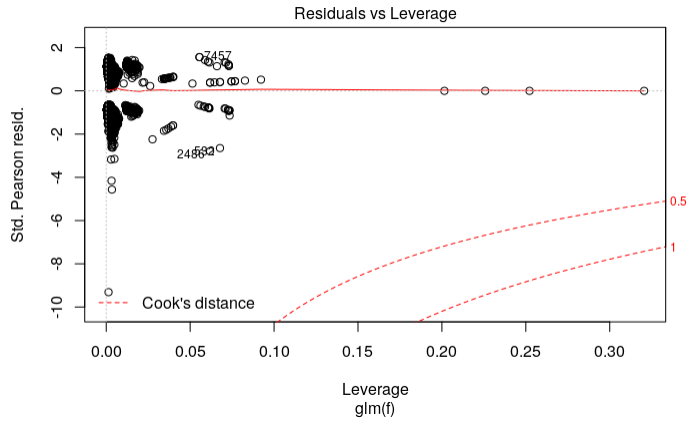
## **Assumptions and Goodness of Fit**

From a Hosmer-Lemeshow test, we fail to reject the null hypothesis that the model fit the data (p-value = 0.1548).



From the plot of residuals vs. Leverage:

* There are a few high leverage points, but the cook’s distance for these points is not high. These points are likely not cause for concern.
* There is one outlier, but it is a low leverage point. This point should not cause influence on the fit.

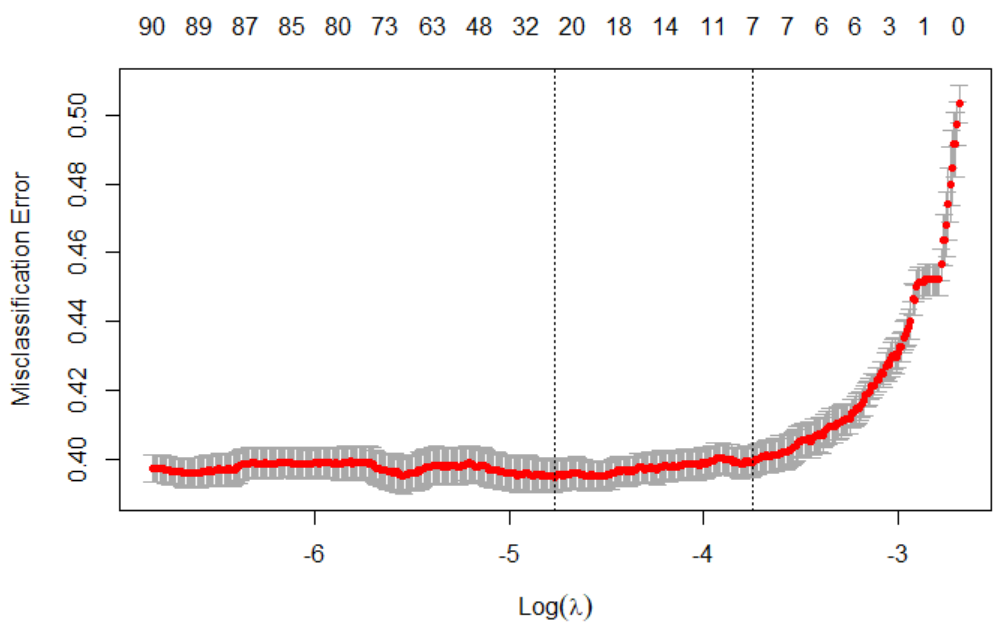


## **Predictive Modeling**

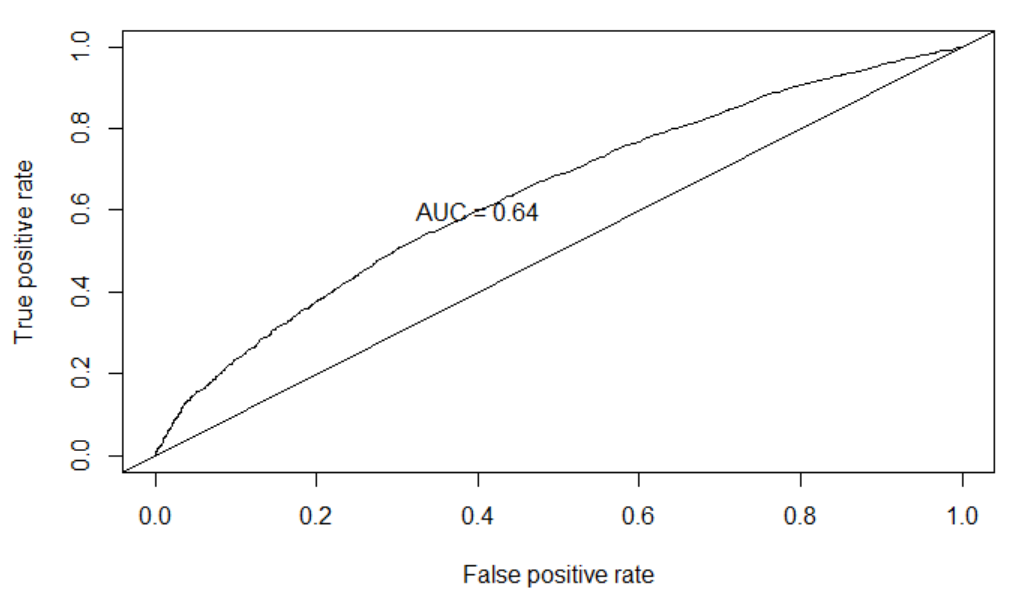
## 

## In the second objective of this project, we aim to provide the most predictive model possible. Several model types including logistic regression, random forests, and k-NN are used for prediction. In cases where a hyperparameter was used, the hyperparameter was tuned using cross validation on the training set. Then the models were scored using predictions on a test set.

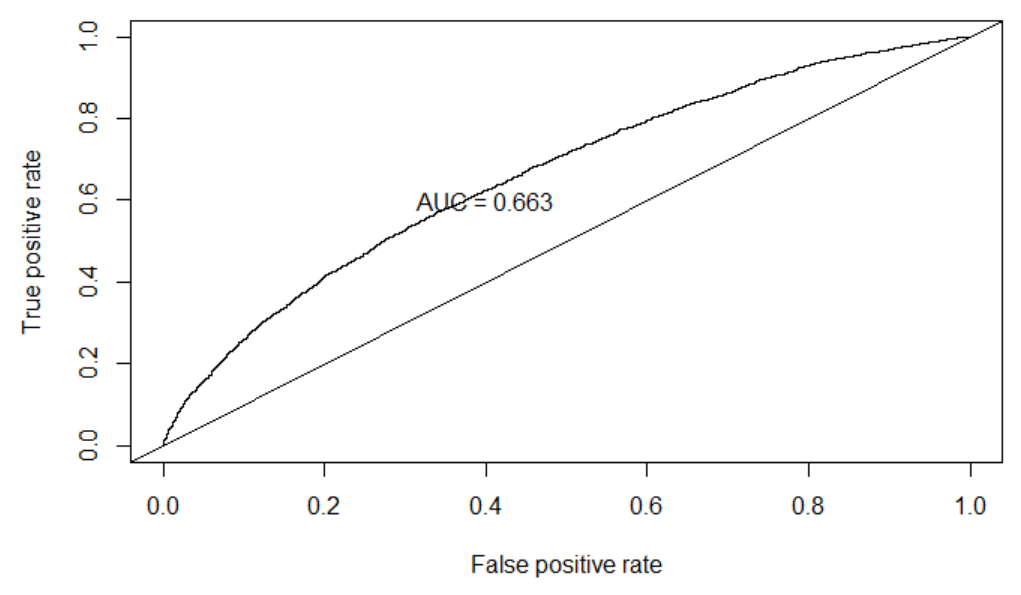
## **Model Selection – Logistic Approach**

As the team’s first initial attempt, we first create a matrix using a simple logistic approach [Figure 8].   


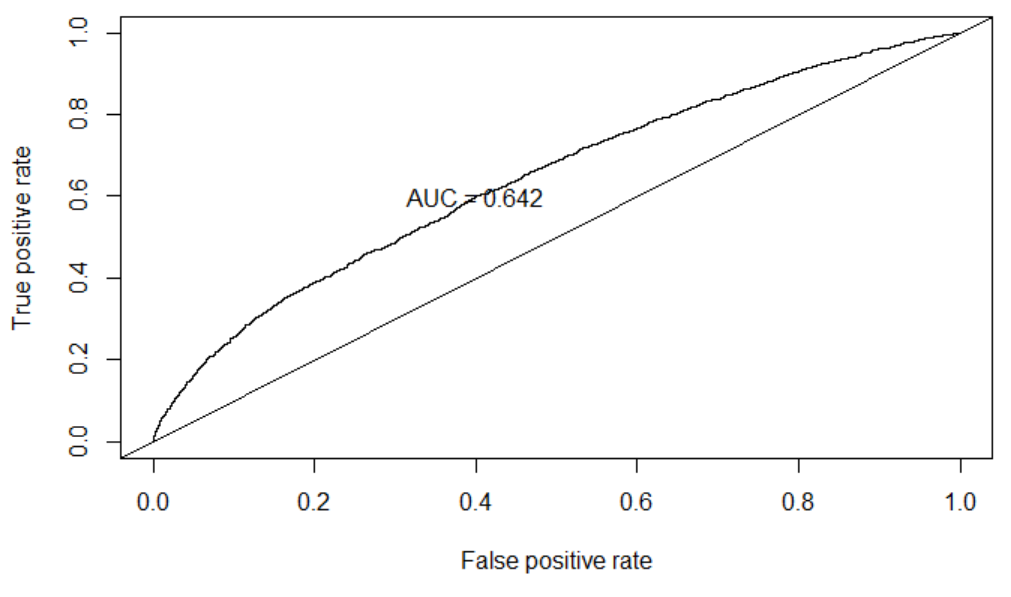
Upon initial inspection, we decided to try some different modeling options because we know we have an imbalanced data set. We started with Simple Logistic. Below we are getting the training set predictions for simple Logistic. We know this is biased but we went ahead and used it to create our first ROC’s.



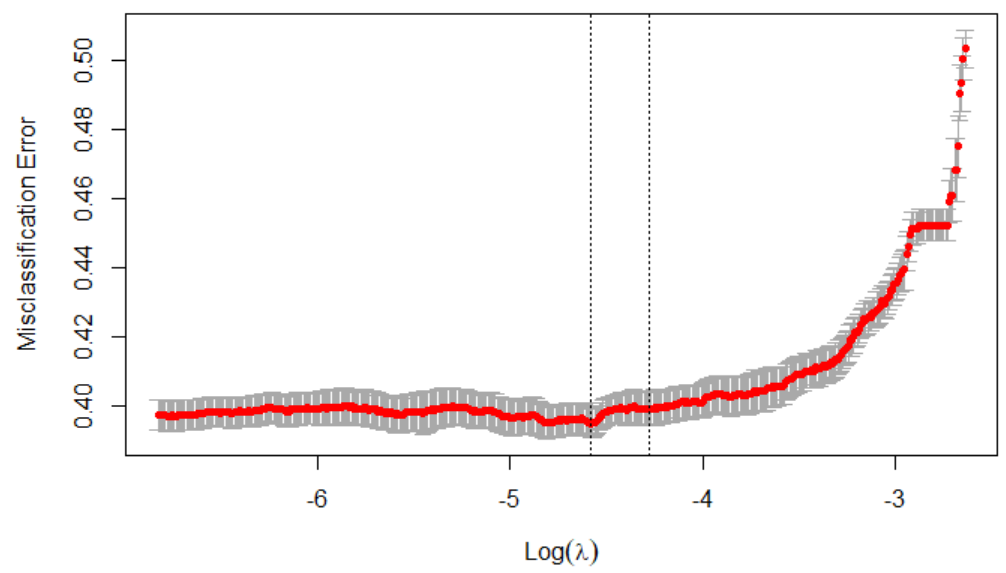
We took the coefficients from the above Logistic model and create a glm model using them since LASSO coefficients will be biased toward zero.

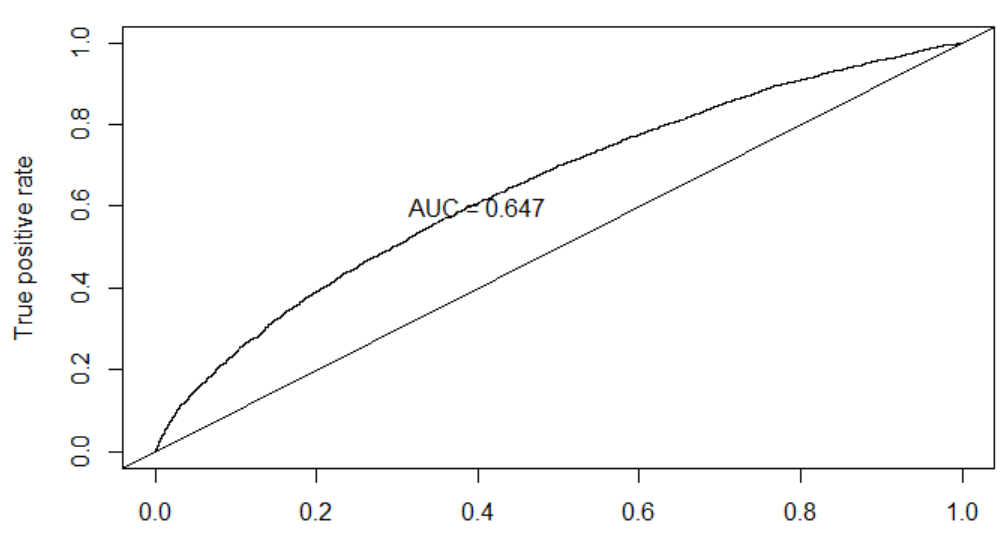


We run against test data set and remove missing levels.



For complex Logistic example, we also included sqrt of number\_inpatients and got AUC =0.647.





## **Model Selection – Random Forest Approach**

The random forest model requires turning the hyperparameter mtry. This parameter is the number of predictors sampled for splitting the trees at each node. Mtry was tuned by cross validation on the training set with tuning values between 1 and 48. Based on OOB error, mtry was set to 3 for the random forest model. As indicated in [Figure 9], the AUC for the Random Forest training set is 1, and 0.643 for the test set.

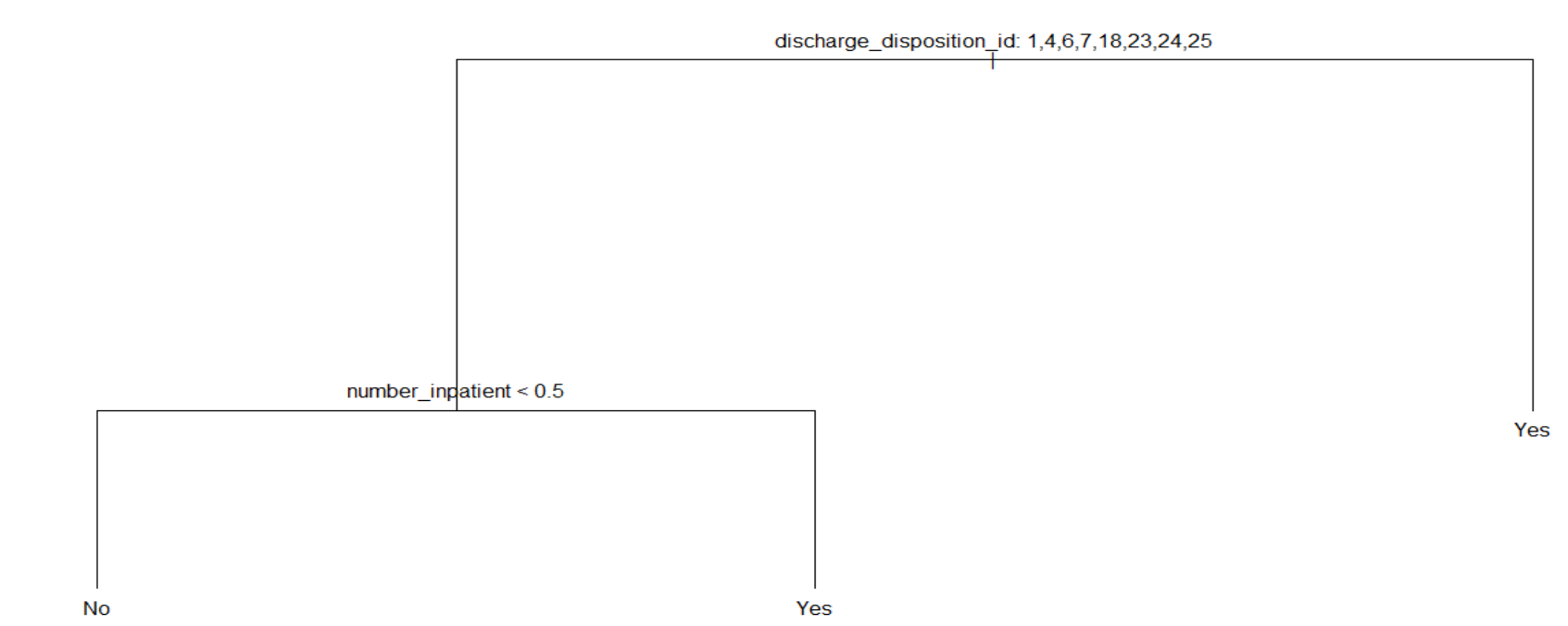
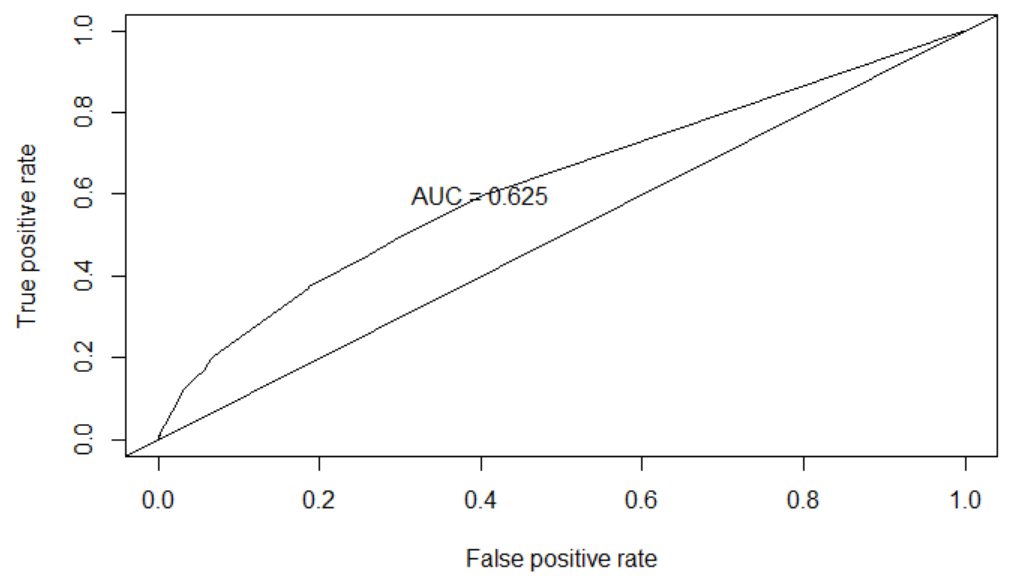


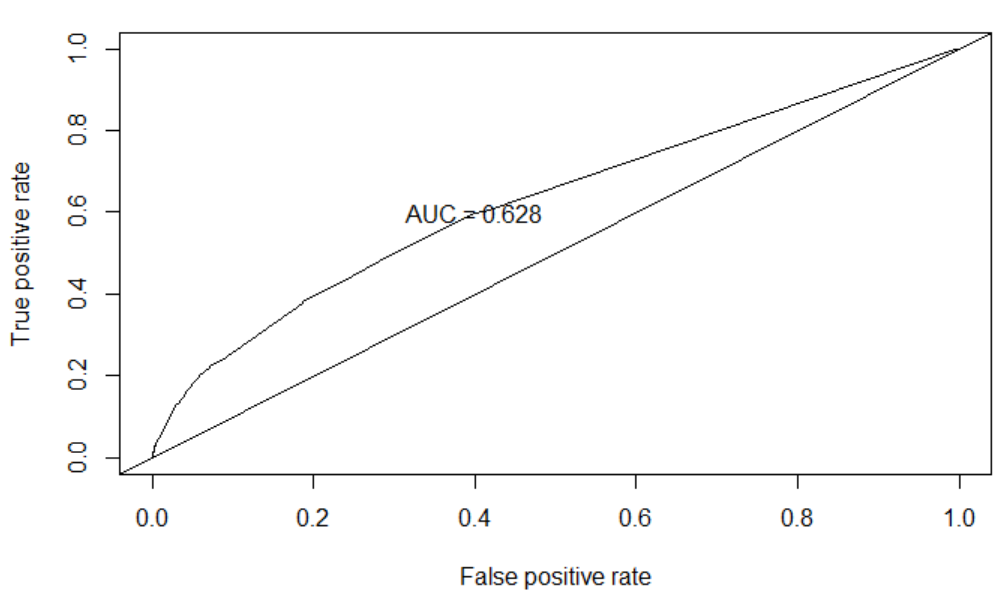
We then perform the Sampling Turning test [Figure 7] to check the OOB Error.

## 

## 

## **Model Selection – Decision Tree Approach**

Here, we’ll need to use a more balanced data set for a decision tree. We tried running against the original data set which returns nothing. And thus here we used the train set created by down sampling. As indicated in [Figure 10], we create a glm model using just the predictors above from the decision tree to how it compares. The AUC we got here is 0.625.  
  


We again apply it against test set.  


## **Model Selection – KNN**

## The KNN model requires tuning of the hyperparameter k. This hyper parameter is the number of neighboring data points that are considered when the prediction is made. We used cross validation on the training set to create a tuning grid for k. The values of k used in tuning ranged from 0 to 30. The tuning results are shown below. The performance of the model is only slightly better than chance and increase very little as k increases. Thus, we determine that this model will not work well for this task. We choose to provide the results of the KNN model with k = 30 for comparison with the other models.

## 

## **Conclusion**

|  |  |
| --- | --- |
| Model | AUC |
| Decision Tree | 0.628 |
| Random Forest | 0.64 |
| Simple Logistic | 0.663 |
| Knn | 0.56 |

In summary, we can see the Simple Logistic and Random Forest performed better than the rest. Overall, we observe that the simple logistic performs better than the rest of the different models. At its baseline, the simple logistic model we created can achieve a accuracy of 66.3%. This result is okay, however, we were not able to improve this test accuracy any further.

## **Reference**

[1] \_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.\_

## **Appendix**

## **List of Tables**

**Table 1 – EDA Summary Statistics**

encounter\_id patient\_nbr race gender admission\_type\_id

Min. : 12522 Min. : 135 ? : 0 Female :37239 1 :35480

1st Qu.: 81307114 1st Qu.: 23342362 AfricanAmerican:12627 Male :32748 3 :13787

Median :143869560 Median : 47985084 Asian : 488 Unknown/Invalid: 3 2 :12803

Mean :156673162 Mean : 54947105 Caucasian :52305 6 : 4516

3rd Qu.:215382384 3rd Qu.: 87498490 Hispanic : 1501 5 : 3086

Max. :443867222 Max. :189502619 Other : 1150 8 : 291

NA's : 1919 (Other): 27

discharge\_disposition\_id admission\_source\_id time\_in\_hospital num\_lab\_procedures num\_procedures num\_medications

1 :44322 7 :37273 Min. : 1.000 Min. : 1.00 Min. :0.000 Min. : 1.00

3 : 8790 1 :21749 1st Qu.: 2.000 1st Qu.: 31.00 1st Qu.:0.000 1st Qu.:10.00

6 : 8291 17 : 4821 Median : 3.000 Median : 44.00 Median :1.000 Median :14.00

18 : 2474 4 : 2530 Mean : 4.273 Mean : 42.88 Mean :1.425 Mean :15.67

2 : 1541 6 : 1785 3rd Qu.: 6.000 3rd Qu.: 57.00 3rd Qu.:2.000 3rd Qu.:20.00

22 : 1411 2 : 908 Max. :14.000 Max. :132.00 Max. :6.000 Max. :81.00

(Other): 3161 (Other): 924

number\_outpatient number\_emergency number\_inpatient diag\_1 diag\_2 diag\_3

Min. : 0.0000 Min. : 0.0000 Min. : 0.0000 414 : 5210 250 : 4996 250 : 8981

1st Qu.: 0.0000 1st Qu.: 0.0000 1st Qu.: 0.0000 428 : 3879 276 : 4496 401 : 6549

Median : 0.0000 Median : 0.0000 Median : 0.0000 786 : 3040 428 : 4220 276 : 3302

Mean : 0.2796 Mean : 0.1039 Mean : 0.1763 410 : 2775 427 : 3445 428 : 2754

3rd Qu.: 0.0000 3rd Qu.: 0.0000 3rd Qu.: 0.0000 486 : 2364 401 : 3077 414 : 2647

Max. :42.0000 Max. :42.0000 Max. :12.0000 (Other):52712 (Other):49463 (Other):44533

NA's : 10 NA's : 293 NA's : 1224

number\_diagnoses max\_glu\_serum A1Cresult metformin repaglinide nateglinide chlorpropamide

Min. : 1.000 >200: 936 >7 : 2866 Down : 435 Down : 28 Down : 8 Down : 1

1st Qu.: 6.000 >300: 712 >8 : 6239 No :55085 No :69073 No :69499 No :69919

Median : 8.000 None:66641 None:57144 Steady:13636 Steady: 818 Steady: 467 Steady: 66

Mean : 7.224 Norm: 1701 Norm: 3741 Up : 834 Up : 71 Up : 16 Up : 4

3rd Qu.: 9.000

Max. :16.000

glimepiride acetohexamide glipizide glyburide tolbutamide pioglitazone rosiglitazone

Down : 136 No :69989 Down : 371 Down : 418 No :69973 Down : 81 Down : 74

No :66292 Steady: 1 No :60980 No :62214 Steady: 17 No :64724 No :65329

Steady: 3332 Steady: 8066 Steady: 6745 Steady: 5007 Steady: 4455

Up : 230 Up : 573 Up : 613 Up : 178 Up : 132

acarbose miglitol troglitazone tolazamide examide citoglipton insulin

Down : 0 Down : 1 No :69987 No :69960 No:69990 No:69990 Down : 7324

No :69790 No :69970 Steady: 3 Steady: 30 No :34268

Steady: 190 Steady: 18 Up : 0 Steady:21621

Up : 10 Up : 1 Up : 6777

glyburide.metformin glipizide.metformin glimepiride.pioglitazone metformin.rosiglitazone metformin.pioglitazone

Down : 4 No :69983 No :69990 No :69988 No :69989

No :69494 Steady: 7 Steady: 0 Steady: 2 Steady: 1

Steady: 485

Up : 7

change diabetesMed ageGrp outcome

Ch:31497 No :16686 [0-30] : 1808 No :63705

No:38493 Yes:53304 [30-60] :21871 Yes: 6285

[60-100]:46311

**Table 2 – Gini Importances Table**

No Yes MeanDecreaseAccuracy MeanDecreaseGini

race 2.3166998 0.2431839 1.88392655 125.08964962

gender 0.2581688 -1.7201287 -1.08876370 78.11033274

admission\_type\_id 4.2240826 1.4645976 4.17251130 181.75860201

discharge\_disposition\_id 41.1686233 21.7023153 44.64468652 305.76153425

admission\_source\_id 3.1839399 -0.4365326 2.09687373 108.30500922

time\_in\_hospital 10.9428419 -1.3638747 7.98254930 276.40750946

num\_lab\_procedures 8.6814185 -1.4953307 5.54967618 426.81281916

num\_procedures 1.8833704 1.2708087 2.41453922 188.87550251

num\_medications 7.5134345 -0.6738950 5.42218548 371.33285746

number\_outpatient 3.2718194 -2.3736210 0.49938294 86.23454726

number\_emergency 5.3303260 3.2659062 6.18206273 58.01641338

number\_inpatient 24.0192313 18.8191766 30.31021333 117.22979136

diag\_1 4.7073685 3.9471989 6.65245450 342.65928079

diag\_2 3.4511510 0.8760012 3.26525934 334.03448064

diag\_3 5.7357227 -4.2772364 1.16134402 346.41677523

number\_diagnoses 9.9494503 2.0783318 9.41240922 198.36945348

max\_glu\_serum 2.1662668 -2.6239457 -0.36292977 36.98429396

A1Cresult 2.3792820 1.1011525 2.48358652 106.96960356

metformin 2.3925329 -0.2304288 1.51285708 73.13800093

repaglinide 3.7636730 -2.2157969 1.09162160 13.25996618

nateglinide -0.8729613 0.2563365 -0.50013350 6.58846637

chlorpropamide 2.7793297 0.5439778 2.57174396 1.25219697

glimepiride 1.8014007 -1.6330008 0.06706097 34.28720326

acetohexamide 0.0000000 0.0000000 0.00000000 0.00000000

glipizide 0.8464160 0.1776770 0.75463800 64.26408109

glyburide -1.9644601 1.5633491 -0.29776045 58.98497937

tolbutamide 0.0000000 0.0000000 0.00000000 0.05003254

pioglitazone 1.7029229 -1.3051835 0.27436434 40.04184851

rosiglitazone -1.1140403 1.9244691 0.49916794 40.96473071

acarbose 0.5628665 -0.1938199 0.27618642 2.84145495

miglitol 0.0000000 0.0000000 0.00000000 0.00000000

troglitazone 0.0000000 0.0000000 0.00000000 0.00000000

tolazamide 0.0000000 -1.0010015 -1.00100150 0.18516347

insulin 5.8940980 -2.3602948 2.76356204 152.83609033

glyburide.metformin 1.7903285 -1.6577318 0.16404513 6.23534837

glipizide.metformin 0.0000000 1.0010015 1.00100150 0.25605584

glimepiride.pioglitazone 0.0000000 0.0000000 0.00000000 0.00000000

metformin.rosiglitazone 0.0000000 0.0000000 0.00000000 0.00000000

metformin.pioglitazone 0.0000000 0.0000000 0.00000000 0.00000000

change 5.9886494 -3.9681160 1.75407172 60.05036455

diabetesMed 7.3859298 -1.1887178 5.15335580 47.79040703

ageGrp 10.7335337 -0.6332057 8.01427331 73.21851673

[1] 305.76153425 117.22979136 198.36945348 73.21851673 276.40750946 342.65928079 58.01641338

[8] 426.81281916 371.33285746 47.79040703 181.75860201 334.03448064 152.83609033 1.25219697

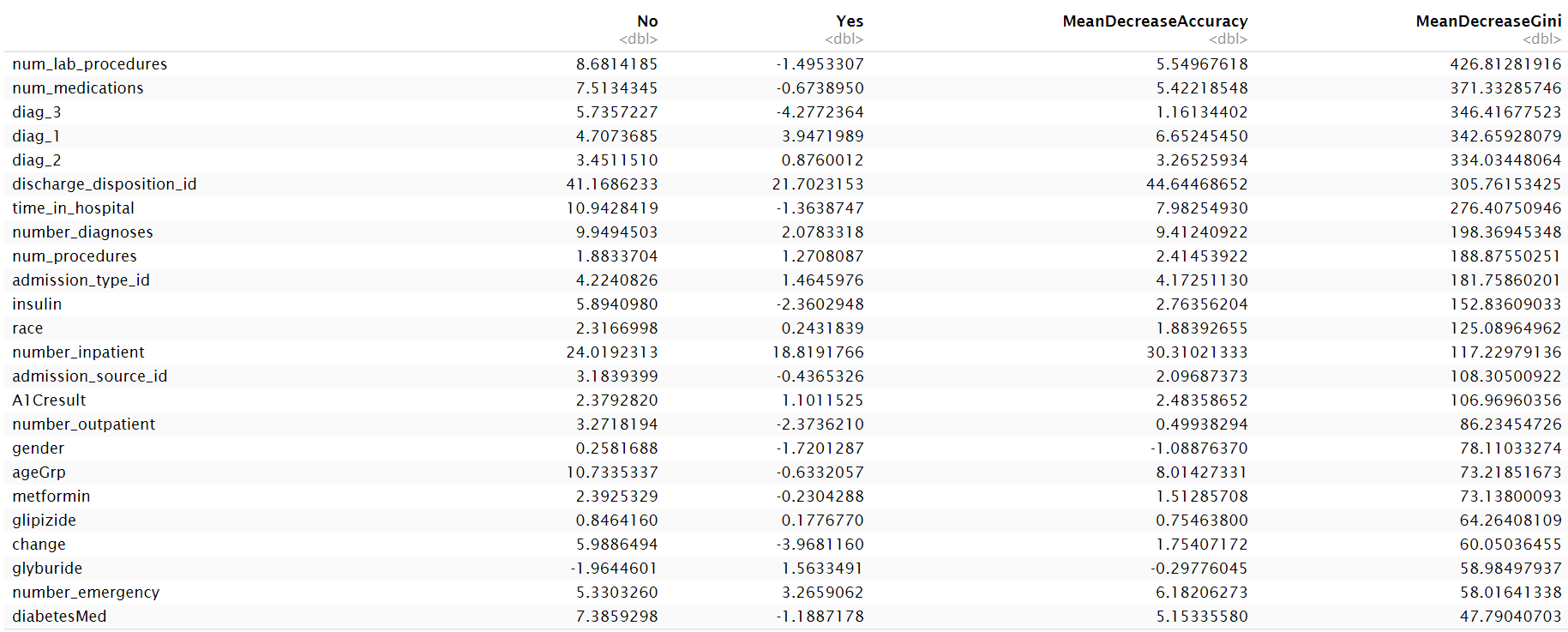
[15] 106.96960356 188.87550251 108.30500922 125.08964962 60.05036455 73.13800093 346.41677523

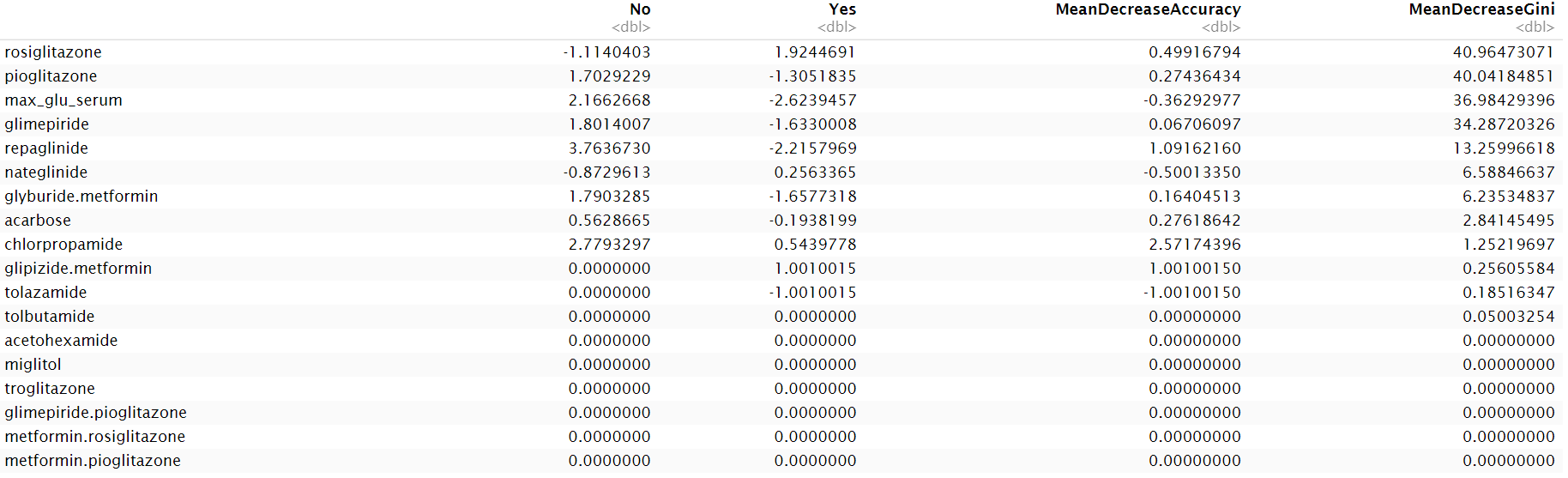
[22] 13.25996618 0.25605584 64.26408109 86.23454726 40.96473071 2.84145495 40.04184851

[29] 6.23534837 34.28720326 0.00000000 0.05003254 0.00000000 0.00000000 0.00000000

[36] 0.00000000 0.00000000 58.98497937 36.98429396 6.58846637 0.18516347 78.11033274

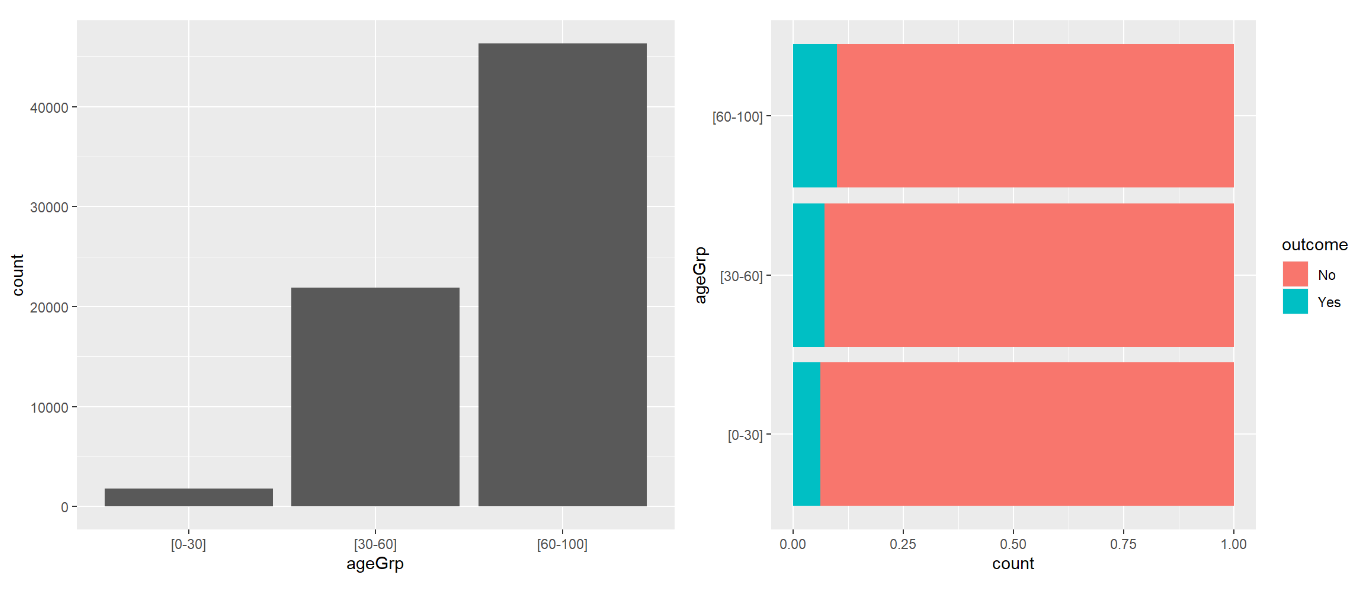
**Table 3 – Gini Table**



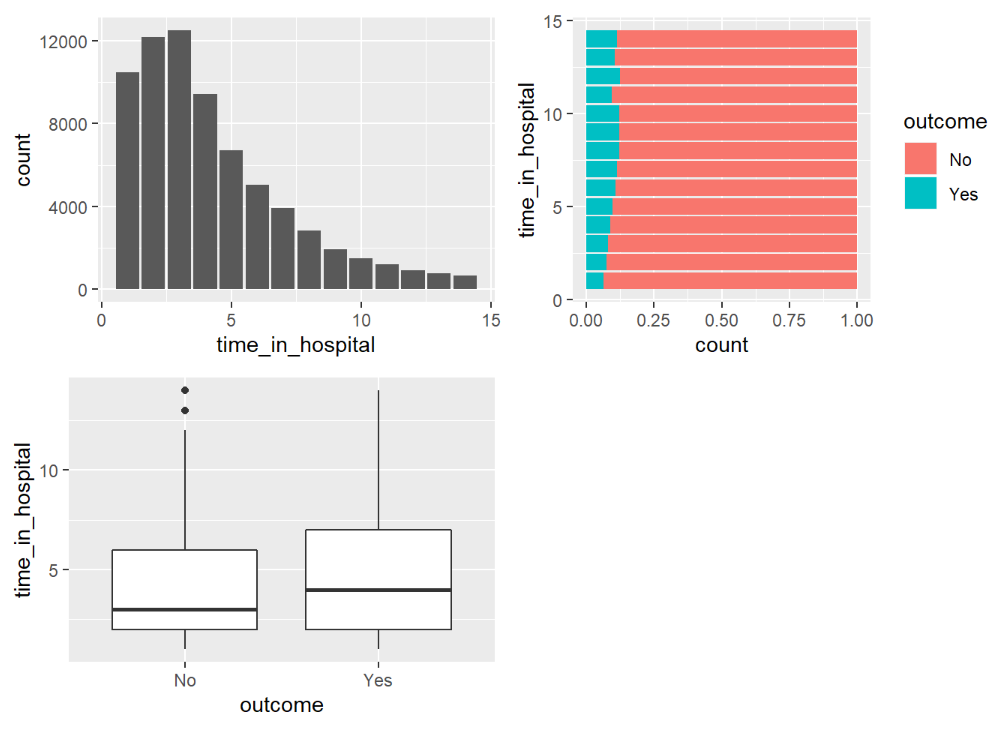


## List of Figures

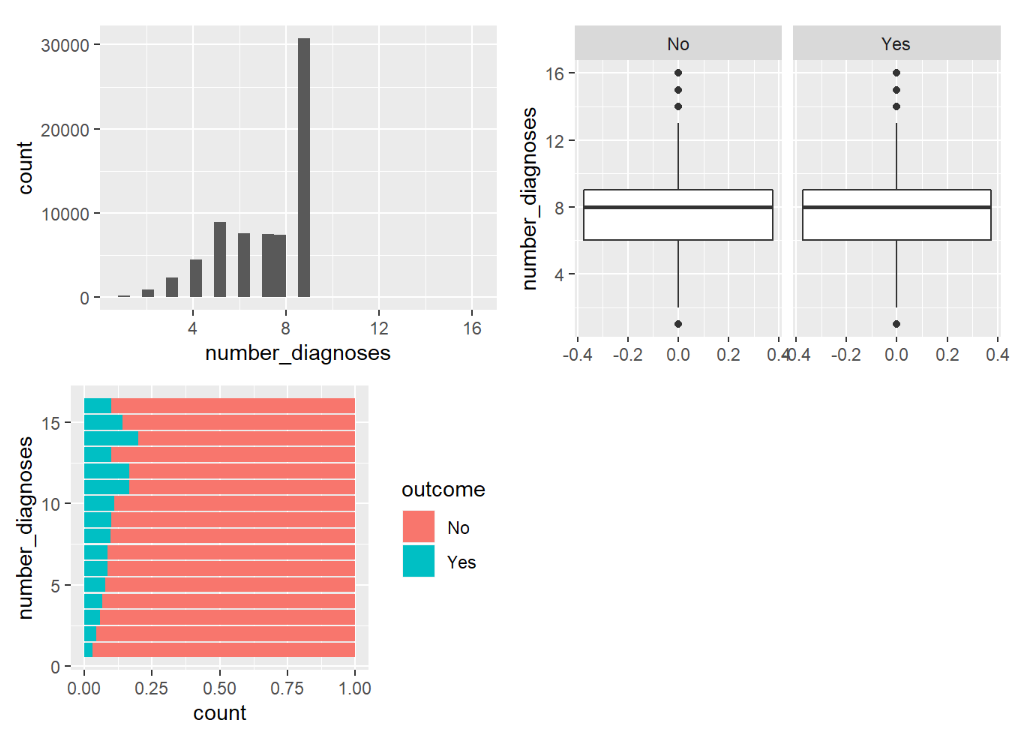
**Figure 1 – Age vs Outcome (Readmission within 30 days)**



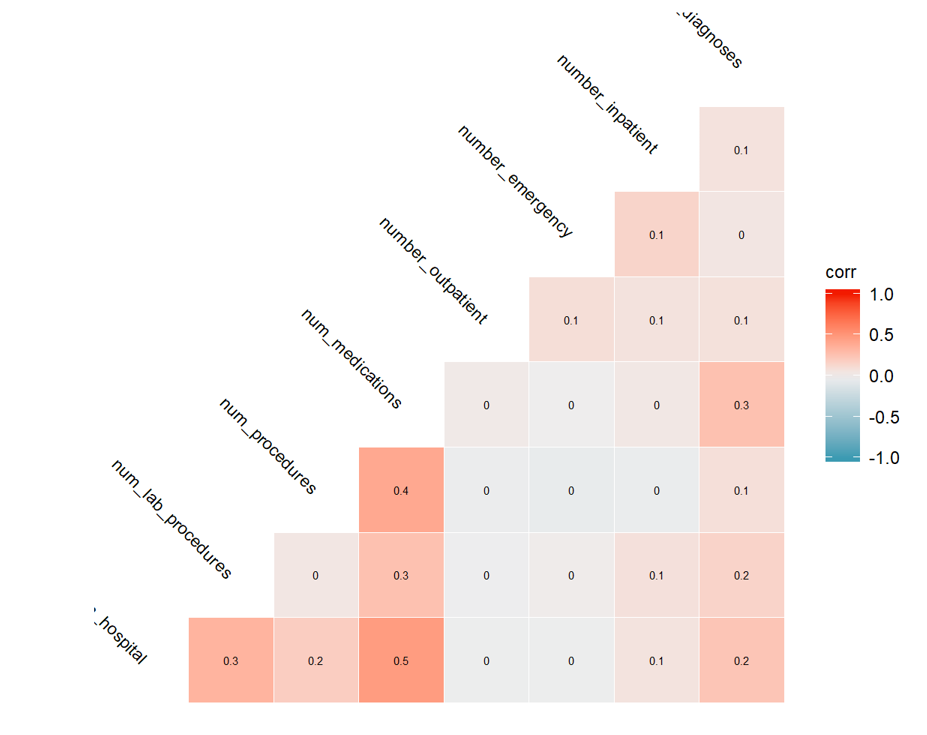
**Figure 2 – Time in Hosptial vs Outcome (Readmission within 30 days)**

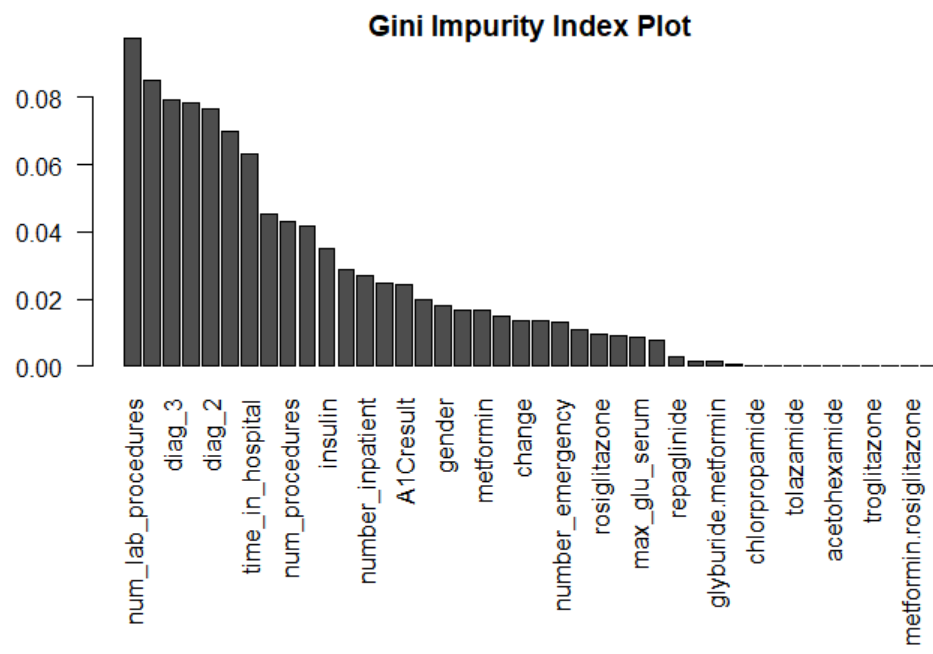
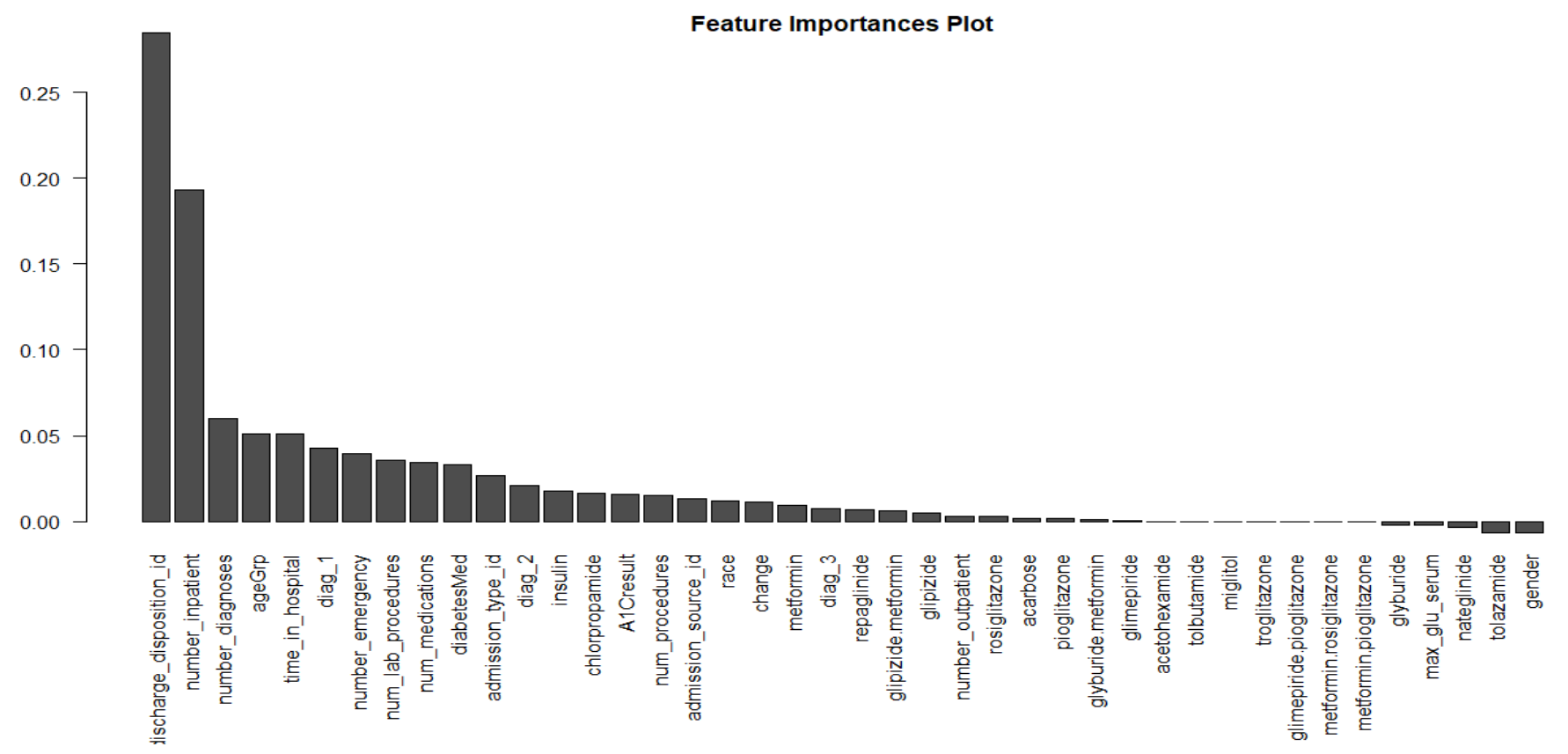


**Figure 3 – Number of Diagnoses vs Outcome (Readmission within 30 days)**

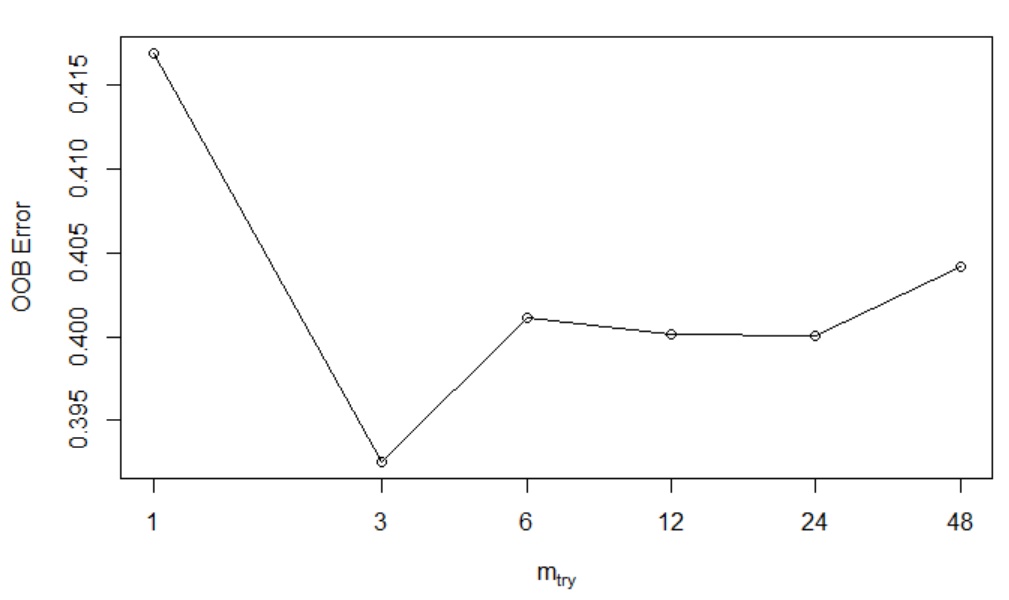


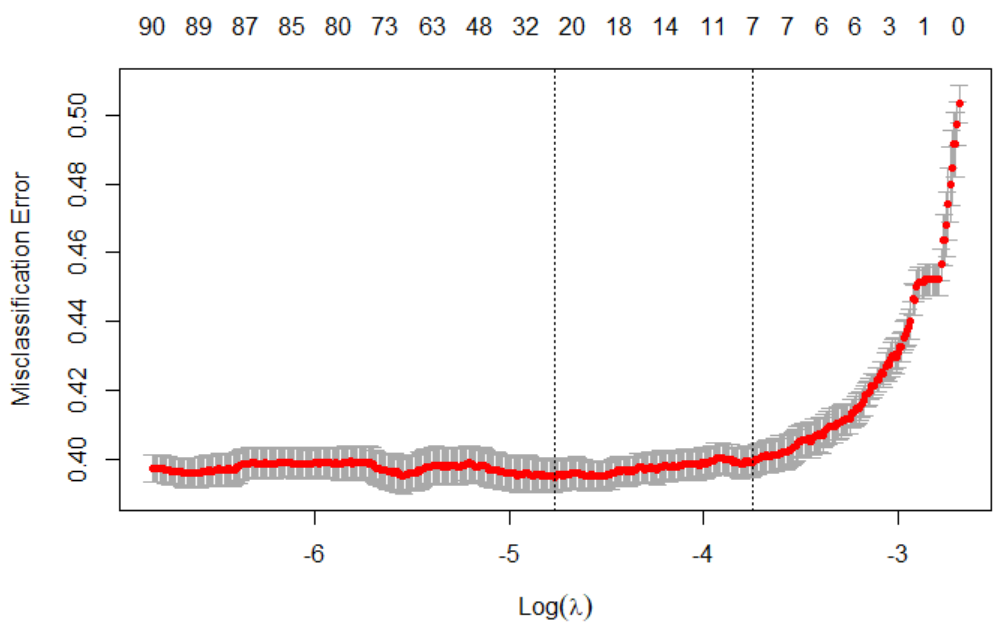
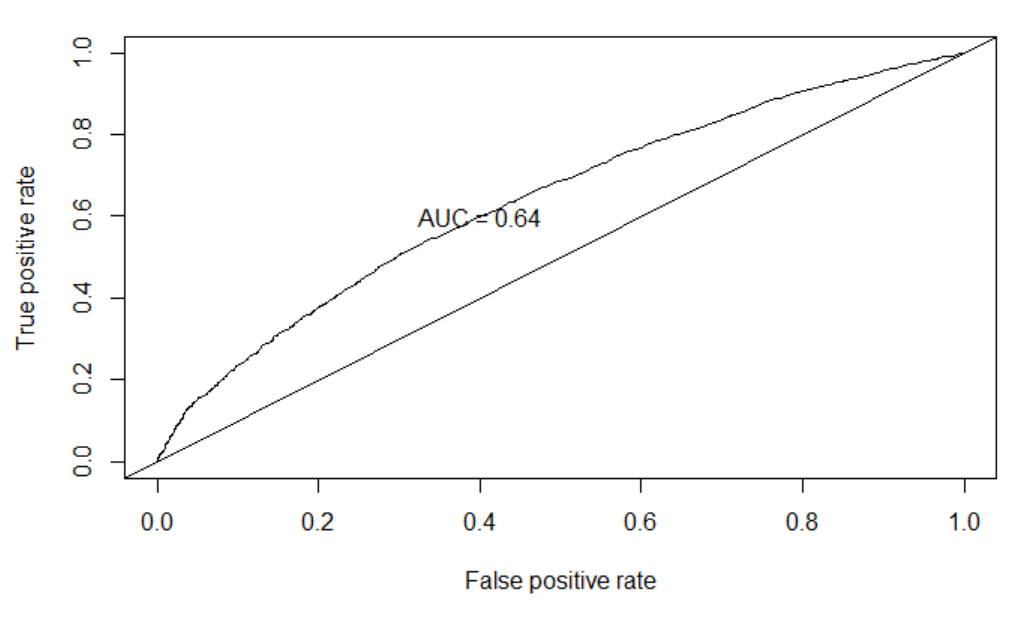
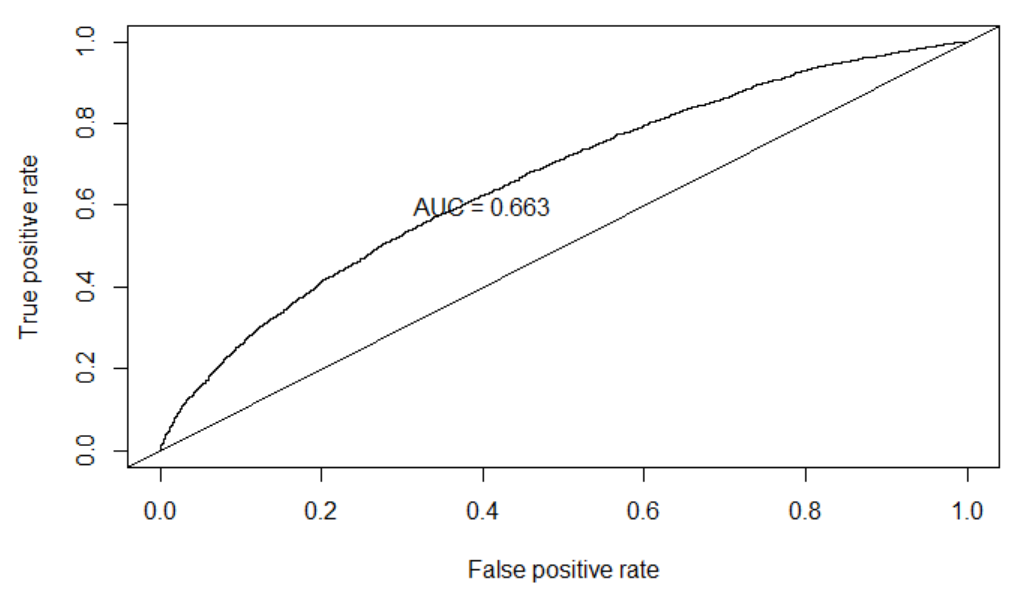
**Figure 4 – Correlation Matrix of Features**

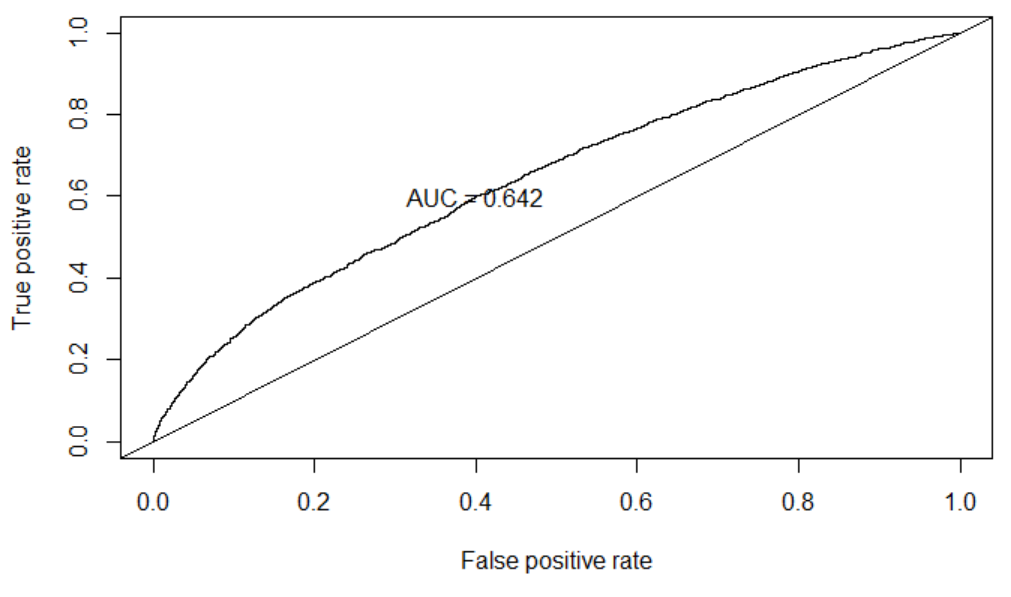


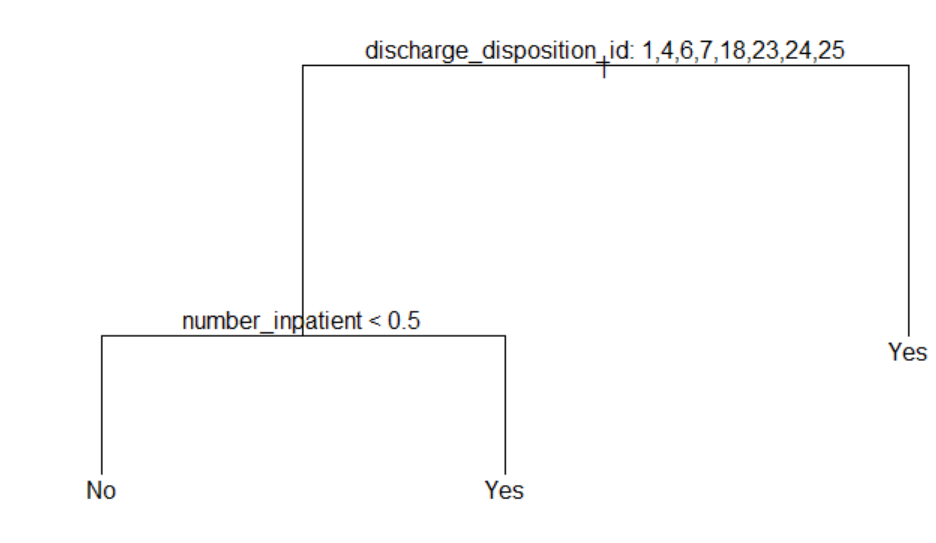
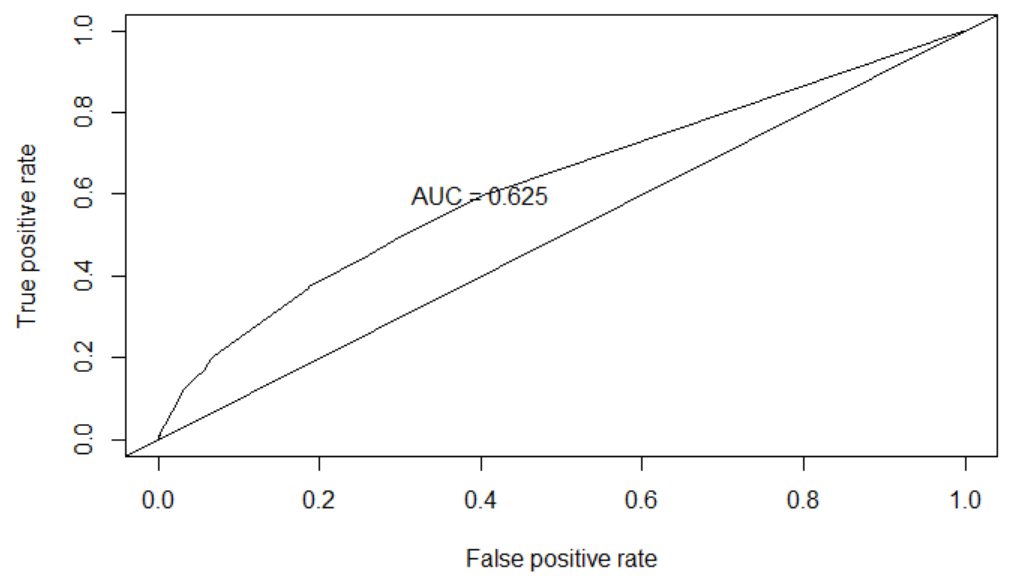
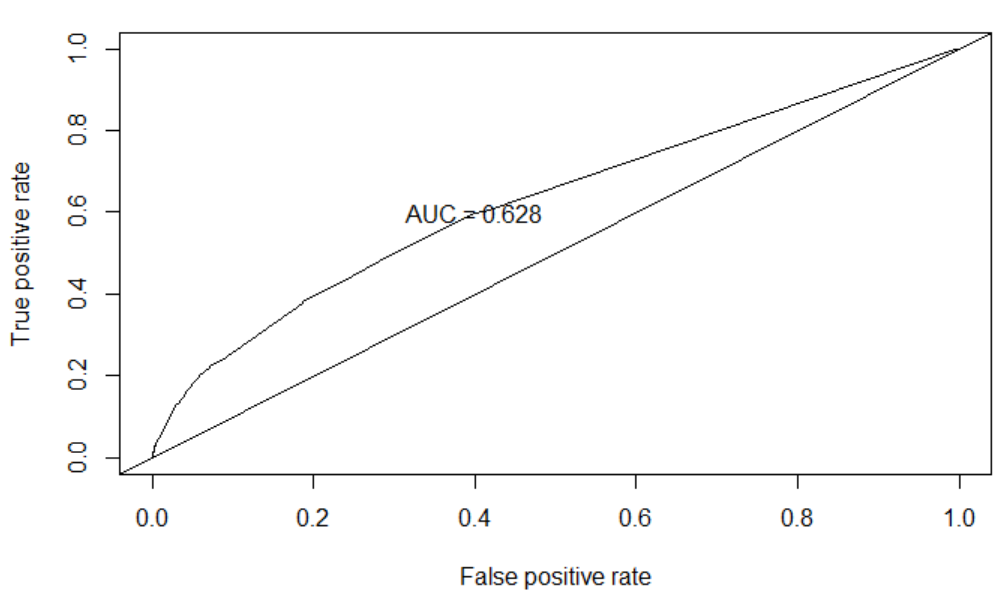
**Figure 5 – Gini Impurity Index Plot**  
  
  
**Figure 6 – Gini Importance Plot**  
  


**Figure 7 – Sampling Tuning Test**

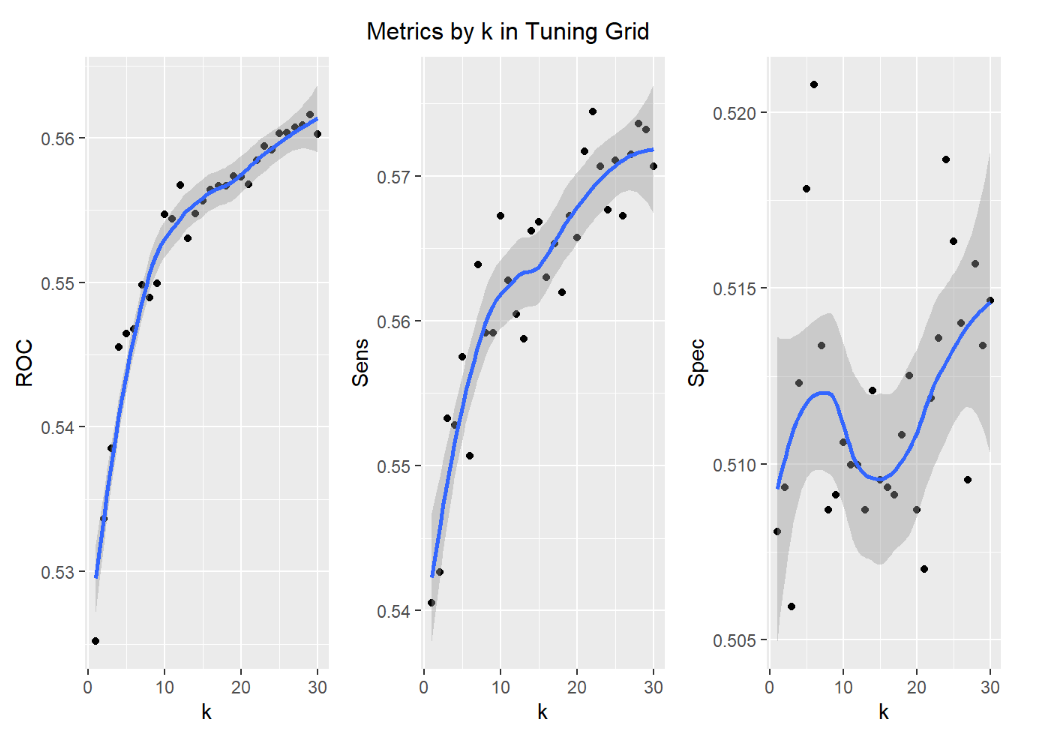


**Figure 8 – Simple Logistic**   
  
  


 **Figure 9 – Random Forest**  
**Figure 10 – Decision Tree**

**Figure 11 – KNN**



## Completed Code

GitHub repository for all the source code and materials:

<https://github.com/chiawang/DS6372_Project2_Group3>

A copy of the source is listed below.

|  |
| --- |
| ---  title: "Prelim EDA"  author: "Stats 2 Project Team"  date: "`r Sys.Date()`"  output: html\_document  ---  ```{r setup, include=FALSE}  knitr::opts\_chunk$set(echo = TRUE)  library(knitr)  library(tidyverse)  library(naniar)  library(questionr)  library(rebus)  library(GGally)  library(gridExtra)  library(caret)  library(randomForest)  library(ROCR)  library(glmnet)  library(car)  library(class)  library(psych)  library(tree)  ```  # Background  https://www.hindawi.com/journals/bmri/2014/781670/  http://downloads.hindawi.com/journals/bmri/2014/781670.pdf  To Do: Provide summary of paper, site as reference.  \_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.\_  # Data Set  https://archive.ics.uci.edu/ml/datasets/diabetes+130-us+hospitals+for+years+1999-2008  The following is taken verbatim from above.  The dataset represents 10 years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks. It includes over 50 features representing patient and hospital outcomes. Information was extracted from the database for encounters that satisfied the following criteria.  \* It is an inpatient encounter (a hospital admission).  \* It is a diabetic encounter, that is, one during which any kind of diabetes was entered to the system as a diagnosis.  \* The length of stay was at least 1 day and at most 14 days.  \* Laboratory tests were performed during the encounter.  \* Medications were administered during the encounter.  The data contains such attributes 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, number of outpatient, inpatient, and emergency visits in the year before the hospitalization, etc.  ## Dictionary  To Do: Provide dictionary of elements. This is provided in the paper's pdf. Let's copy it here and reference page 3 of Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records pdf document. The table can also be found here,  https://www.hindawi.com/journals/bmri/2014/781670/tab1/  ## Import Data  ```{r import}  data <- read.csv("./diabetic\_data.csv")  head(data)  ```  Notice above that we having missing values represented by '?'. Let's replace '?' with NA.  ```{r replace\_missing}  data <- data %>% mutate\_all(~na\_if(., '?'))  ```  Check variables with missing values.  ```{r}  freq.na(data)  ```  Due to the frequency of missing data, let's remove 'weight', 'medical\_specialty', and 'payer\_code'.  ```{r}  data <- select(data, select=-c("weight","medical\_specialty","payer\_code"))  ```  Next, let's remove observations where the discharge disposition is related to hospice or death, since these will not add to the possibility of being readmitted.  \* 11 - Expired  \* 13 - Hospice/Home  \* 14 - Hospice/Medical Facility  \* 19 - Expired at Home  \* 20 - Expired at Medical Facility  \* 21 - Expired at Unknown  ```{r discharge}  data <- data %>% filter(!data$discharge\_disposition\_id %in% c('11','13','14','19','20','21'))  ```  In the data set, we find that patients have multiple admissions. The paper suggests using only the first one to satisfy the independence assumption. So we will filter and only use the first encounter (the lowest for a given member). I wonder if this could be challenged. If the diagnosis or reason for visit is different than a previous admission, I think it could be considered statisically independent.  ```{r oneEncounter}  data <- data %>% group\_by(patient\_nbr) %>% filter(encounter\_id == min(encounter\_id))  data <- ungroup(data)  ```  Next, let's create buckets for the ICD-9 diagnosis codes. We'll use the mapping defined based on the following.  https://www.hindawi.com/journals/bmri/2014/781670/tab2/  ```{r helper}  replaceDX <- function(df = data, rx, replaceValue) {  df$diag\_1[grep(rx,df$diag\_1)] <- replaceValue  df$diag\_2[grep(rx,df$diag\_2)] <- replaceValue  df$diag\_3[grep(rx,df$diag\_3)] <- replaceValue    return(df)  }  ```  ```{r}  # Create temporary data frame of bucketed DX codes  dx <- c("diag\_1","diag\_2","diag\_3")  dataDX <- data[dx]  dataDX[] <- lapply(data[dx], as.character)  # Circulatory  label <- "Circulatory"  rx <- number\_range(390,459)  dataDX <- replaceDX(dataDX, rx, label)  dataDX <- replaceDX(dataDX, "785", label)  # Respiratory  label <- "Respiratory"  rx <- number\_range(460,519)  dataDX <- replaceDX(dataDX, rx, label)  dataDX <- replaceDX(dataDX, "786", label)  # Digestive  label <- "Digestive"  rx <- number\_range(520,579)  dataDX <- replaceDX(dataDX, rx, label)  dataDX <- replaceDX(dataDX, "787", label)  # Diabetes  label <- "Diabetes"  rx <- number\_range(250,250.99)  dataDX <- replaceDX(dataDX, rx, label)  # Injury  label <- "Injury"  rx <- number\_range(800,999)  dataDX <- replaceDX(dataDX, rx, label)  # Musculoskeletal  label <- "Musculoskeletal"  rx <- number\_range(710,739)  dataDX <- replaceDX(dataDX, rx, label)  # Genitourinary  label <- "Genitourinary"  rx <- number\_range(580,629)  dataDX <- replaceDX(dataDX, rx, label)  dataDX <- replaceDX(dataDX, "788", label)  # Neoplasms  label <- "Neoplasms"  rx <- number\_range(140,239)  dataDX <- replaceDX(dataDX, rx, label)  rx <- number\_range(780,782)  dataDX <- replaceDX(dataDX, rx, label)  dataDX <- replaceDX(dataDX, "784", label)  rx <- number\_range(790,799)  dataDX <- replaceDX(dataDX, rx, label)  rx <- number\_range(240,279)  dataDX <- replaceDX(dataDX, rx, label)  rx <- number\_range(680,709)  dataDX <- replaceDX(dataDX, rx, label)  # Number range here doesn't work as I would expect, probably because it spans 1 to 3 digits  # So I'll just create my own regEx object  # rx <- number\_range(1,139)  rx <- regex("(?:^([1-9]|[1-8][0-9]|9[0-9]|1[0-2][0-9]|13[0-9])$)")  dataDX <- replaceDX(dataDX, rx, label)  # Other  label = "Other"  rx = "^E.\*"  dataDX <- replaceDX(dataDX, rx, label)  rx = "^V.\*"  dataDX <- replaceDX(dataDX, rx, label)  rx <- number\_range(290,319)  dataDX <- replaceDX(dataDX, rx, label)  rx <- number\_range(280,289)  dataDX <- replaceDX(dataDX, rx, label)  rx <- number\_range(320,359)  dataDX <- replaceDX(dataDX, rx, label)  rx <- number\_range(630,679)  dataDX <- replaceDX(dataDX, rx, label)  rx <- number\_range(360,389)  dataDX <- replaceDX(dataDX, rx, label)  rx <- number\_range(740,759)  dataDX <- replaceDX(dataDX, rx, label)  dataDX <- replaceDX(dataDX, "783", label)  dataDX <- replaceDX(dataDX, "789", label)  #If value is "NA", we'll replace with "None"  #dataDX <- replace\_na(dataDX,"None")  # Now update main data table and set DX buckets as factors. Replace NA with None  dataDX <- lapply(dataDX,replace\_na,"None")  data[dx] <- lapply(dataDX, as.factor)  ```  ```{r}  # Use plyr here, it looks better  plyr::count(data$diag\_1)  ```  Age is a factor with 10 levels. We are going to bucket these into 3 different groups in a new feature, and remove the old feature.  \* 0 - 30  \* 30 - 60  \* 60 - 100  ```{r}  data[which(data$age=="[0-10)"),c("ageGrp")] <- "[0-30]"  data[which(data$age=="[10-20)"),c("ageGrp")] <- "[0-30]"  data[which(data$age=="[20-30)"),c("ageGrp")] <- "[0-30]"  data[which(data$age=="[30-40)"),c("ageGrp")] <- "[30-60]"  data[which(data$age=="[40-50)"),c("ageGrp")] <- "[30-60]"  data[which(data$age=="[50-60)"),c("ageGrp")] <- "[30-60]"  data[is.na(data$ageGrp),c("ageGrp")] <- "[60-100]"  data$ageGrp <- as.factor(data$ageGrp)  # Remove old feature  data <- select(data, select=-c("age"))  ```  Convert features with numerals as factors. Perhaps it would be better to convert these to their actual names.  ```{r}  data$admission\_type\_id <- as.factor(data$admission\_type\_id)  data$discharge\_disposition\_id <- as.factor(data$discharge\_disposition\_id)  data$admission\_source\_id <- as.factor(data$admission\_source\_id)  ```  There are two medications that have only 1 factor level, "examide" and "citoglipton". We will remove these from the data set.  ```{r}  data <- select(data, select=-c("examide","citoglipton"))  ```  Since encounter\_id and patient\_nbr are identifiers, we will remove these as well.  ```{r}  # We need to ungroup by patient\_nbr since we grouped by it earlier.  data <- select(data, select=-c("encounter\_id","patient\_nbr"))  ```  Clean up Race feature NAs and label as "Missing"  ```{r}  levels(data$race) <- c(levels(data$race), "Missing")  data$race <- replace\_na(data$race, "Missing")  ```  There are some levels for discharge dispostion that have very few values.  ```{r}  count(data,data$discharge\_disposition\_id)  ```  Let's bucket those less than 10 into the "Not Mapped" bucket, which is id 25  ```{r}  data$discharge\_disposition\_id[data$discharge\_disposition\_id %in% c(9,10,12,16,17,27)] <- 25  count(data,data$discharge\_disposition\_id)  ```  Similarly, let's bucket admission\_source\_id since the distribution of it isn't very even. This can create issues when splitting the data set up and not having the same levels in both the train and test sets.  ```{r}  count(data,data$admission\_source\_id)  ```  Levels 1, 2, and 3 will be bucketed into 1 (Admitted because of Referral). Level 7 will remain as is, admitted by Emergency Room. Everything else will be 4 (Otherwise)  ```{r}  `%!in%` = Negate(`%in%`)  data$admission\_source\_id[data$admission\_source\_id %in% c(1,2,3)] <- 1  data$admission\_source\_id[data$admission\_source\_id %!in% c(1,2,3,7)] <- 4  data$admission\_source\_id <- as.factor(data$admission\_source\_id)  count(data,data$admission\_source\_id)  ```  Lastly, let's create an outcome column that is Yes if admitted within 30 days, and No for everything else. Then remove the readmitted column. We'll use "outcome" as the response variable.  ```{r}  data$outcome <- factor(ifelse(data$readmitted == "<30","Yes","No"))  data <- select(data, select=-c("readmitted"))  plyr::count(data$outcome)  ```  Above we see that about 10% of our data shows as readmitted, so we have an inbalanced data set. Need to consider that in our analysis. See  https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18  The final dataset consists of 69,990 observations with 43 features.  ### Proportion Plots  #### Age  \* The bulk of the patients are in the age range of 40 - 90 years.  \* The proportion of readimts increase from the lowest bucket up to the 80s bucket.  ```{r, fig.width=12}  p1 <- data %>%  ggplot(aes(x = ageGrp)) +  geom\_bar()  p2 <- data %>%  ggplot(aes(x = ageGrp, fill = outcome)) +  geom\_bar(position = 'fill') +  coord\_flip()  grid.arrange(p1, p2, ncol = 2)  ```  #### Time in Hospital  This is the number of days a patient spent in the hospital.  The distribution is at max at 3 days, then tails off at the time increases.  Maximum observed value is 14 days.  ```{r}  p1 <- data %>%  ggplot(aes(x = time\_in\_hospital)) +  geom\_bar()  p2 <- data %>%  ggplot(aes(x = time\_in\_hospital, fill = outcome)) +  geom\_bar(position = 'fill') +  coord\_flip()  p3 <- data %>%  ggplot(aes(x = outcome, y = time\_in\_hospital)) +  geom\_boxplot()  grid.arrange(p1, p2, p3, ncol = 2)  ```  #### Number of Diagnoses  There seem to be a correlation between readmittance and number of diagnoses.  ```{r}  p1 <- data %>%  ggplot(aes(x = number\_diagnoses)) +  geom\_histogram(bins = 30)  p2 <- data %>%  ggplot(aes(y = number\_diagnoses)) +  geom\_boxplot() +  facet\_wrap(. ~ outcome)  p3 <- data %>%  ggplot(aes(x = number\_diagnoses, fill = outcome)) +  geom\_bar(position = 'fill') +  coord\_flip()  grid.arrange(p1, p2, p3, ncol = 2)  ```  #### Heatmap  correlation between each factors.  ```{r}  ggcorr(data %>% select(-outcome, -ageGrp, -admission\_source\_id, -discharge\_disposition\_id, -admission\_type\_id, -race, -gender, -diag\_1, -diag\_2, -diag\_3, -max\_glu\_serum, -A1Cresult, -metformin, -repaglinide, -nateglinide, -chlorpropamide, -glimepiride, -acetohexamide, -glipizide, -glyburide, -tolbutamide, -pioglitazone, -rosiglitazone, -acarbose, -miglitol, -troglitazone, -tolazamide,-insulin, -glyburide.metformin, -glipizide.metformin, -glimepiride.pioglitazone, -metformin.rosiglitazone, -metformin.pioglitazone, -change, -diabetesMed), name = "corr", label = TRUE, hjust = 1, label\_size = 2, angle = -45, size = 3)  ```  ## Modeling  Let's try some different modeling options. Since we have an imbalanced data set, there are different things we can try. Let's start with down sampling.  ### Down Sample  ```{r}  set.seed(9560)  trainIdx <- createDataPartition(data$outcome, p =.75, list = F, times = 1)  train <- data[trainIdx, ]  test <- data[-trainIdx, ]  set.seed(9560)  train <- downSample(x = train[, -ncol(train)],  y = train$outcome, yname="outcome")  train.y = train$outcome  train.x = train[,-ncol(train)]  test.y = test$outcome  test.x = test[,-ncol(test)]  ```  ### Random Forest Example  https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/  Training Set  ```{r}  train.rf<-randomForest(outcome~.,data=train,importance=T)  fit.pred<-predict(train.rf,newdata=train,type="prob")  pred <- prediction(fit.pred[,2], train$outcome)  roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")  auc.train <- performance(pred, measure = "auc")  auc.train <- auc.train@y.values  plot(roc.perf)  abline(a=0, b= 1)  text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))  ```  Test Set  ```{r}  pred.val1<-predict(train.rf,newdata=test,type="prob")  pred <- prediction(pred.val1[,2], test$outcome)  roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")  auc.test <- performance(pred, measure = "auc")  auc.test <- auc.test@y.values  plot(roc.perf)  abline(a=0, b=1)  text(x = .4, y = .6,paste("AUC = ", round(auc.test[[1]],3), sep = ""))  ```  \*\*Gini importances plot\*\*  Gini has to do with the cross-entroy of the trees from a split. Decreases in Gini index are desirable.  https://towardsdatascience.com/seeing-the-random-forest-from-the-decision-trees-an-intuitive-explanation-of-random-forest-beaa2d6a0d80  ISL p.312  https://stats.stackexchange.com/questions/92419/relative-importance-of-a-set-of-predictors-in-a-random-forests-classification-in  ```{r}  importance(train.rf)  rf.feature.importance <- data.frame(importance(train.rf))  varimp1 <- rf.feature.importance[order(rf.feature.importance$MeanDecreaseGini,decreasing = T),]  varimp1  par(mar=c(10,5,1,1))  giniplot <- barplot(t(varimp1[-2]['MeanDecreaseGini']/sum(varimp1[-2]['MeanDecreaseGini'])),  las=2, cex.names=1,  main="Gini Impurity Index Plot")  varimp2 <- rf.feature.importance[order(rf.feature.importance$MeanDecreaseAccuracy,decreasing = T),]  varimp2$MeanDecreaseGini  par(mar=c(10,5,1,1))  giniplot <- barplot(t(varimp2[-2]['MeanDecreaseAccuracy']/sum(varimp2[-2]['MeanDecreaseAccuracy'])),  las=2, cex.names=1,  main="Feature Importances Plot")  ```  Sampling Tuning Test  ```{r}  bestmtry <- tuneRF(train.x, train.y, stepFactor=.5, improve=1e-5, ntree=500)  print(bestmtry)  ```  ### Simple Logistic Example  ```{r}  #glmnet requires a matrix  set.seed(9560)  f <- as.formula(outcome ~ .)  train.x <- model.matrix(f, train)  #train.x <- data.matrix(train.x)  cvfit <- cv.glmnet(train.x,train.y, family = "binomial", type.measure = "class", nlambda = 1000)  plot(cvfit)  coef(cvfit, s = "lambda.min")  #rownames(coef(cvfit, s = 'lambda.min'))[coef(cvfit, s = 'lambda.min')[,1]!=0]  #Get training set predictions...We know they are biased but lets create ROC's.  #These are predicted probabilities from logistic model exp(b)/(1+exp(b))  fit.pred <- predict(cvfit, newx = train.x, type = "response")  #Compare the prediction to the real outcome  head(fit.pred)  head(train.y)  #Create ROC curves  pred <- prediction(fit.pred[,1], train.y)  roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")  auc.train <- performance(pred, measure = "auc")  auc.train <- auc.train@y.values  #Plot ROC  plot(roc.perf)  abline(a=0, b= 1) #Ref line indicating poor performance  text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))  ```  Take the coefficients from above, and create a glm model using them since glmnet/lasso will bias the coefficients towards zero. Note, each time I restart R, I get a different set of coefficients for the same seed value??  ```{r}  f <- as.formula(outcome ~ race + discharge\_disposition\_id + admission\_type\_id + time\_in\_hospital + num\_lab\_procedures + num\_medications + number\_emergency + number\_inpatient + diag\_1 + diag\_2 + diag\_3 + number\_diagnoses + A1Cresult + metformin + acarbose + diabetesMed + ageGrp)  #f <- as.formula(outcome ~ discharge\_disposition\_id + number\_inpatient + diag\_1)  glm.simple <- glm(formula = f, family = "binomial", data = train)  coef(glm.simple)  #Get training set predictions...We know they are biased but lets create ROC's.  #These are predicted probabilities from logistic model exp(b)/(1+exp(b))  fit.pred <- predict(glm.simple, newx = train.x, type = "response")  #Compare the prediction to the real outcome  head(fit.pred)  head(train.y)  #Create ROC curves  pred <- prediction(fit.pred, train.y)  roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")  auc.train <- performance(pred, measure = "auc")  auc.train <- auc.train@y.values  #Plot ROC  plot(roc.perf)  abline(a=0, b= 1) #Ref line indicating poor performance  text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))  ```  Update! This function is not needed now since we went back and rebucketed those disposition values with a small frequency into a different bucket. But keep here for now in case we need it.  Stole the function from stack overflow. I think in general this will always be an issue. Missing factor levels between test and train sets.  https://stackoverflow.com/questions/4285214/predict-lm-with-an-unknown-factor-level-in-test-data  ```{r}  remove\_missing\_levels <- function(fit, test\_data) {  # https://stackoverflow.com/a/39495480/4185785  # drop empty factor levels in test data  test\_data %>%  droplevels() %>%  as.data.frame() -> test\_data  # 'fit' object structure of 'lm' and 'glmmPQL' is different so we need to  # account for it  if (any(class(fit) == "glmmPQL")) {  # Obtain factor predictors in the model and their levels  factors <- (gsub("[-^0-9]|as.factor|\\(|\\)", "",  names(unlist(fit$contrasts))))  # do nothing if no factors are present  if (length(factors) == 0) {  return(test\_data)  }  map(fit$contrasts, function(x) names(unmatrix(x))) %>%  unlist() -> factor\_levels  factor\_levels %>% str\_split(":", simplify = TRUE) %>%  extract(, 1) -> factor\_levels  model\_factors <- as.data.frame(cbind(factors, factor\_levels))  } else {  # Obtain factor predictors in the model and their levels  factors <- (gsub("[-^0-9]|as.factor|\\(|\\)", "",  names(unlist(fit$xlevels))))  # do nothing if no factors are present  if (length(factors) == 0) {  return(test\_data)  }  factor\_levels <- unname(unlist(fit$xlevels))  model\_factors <- as.data.frame(cbind(factors, factor\_levels))  }  # Select column names in test data that are factor predictors in  # trained model  predictors <- names(test\_data[names(test\_data) %in% factors])  # For each factor predictor in your data, if the level is not in the model,  # set the value to NA  for (i in 1:length(predictors)) {  found <- test\_data[, predictors[i]] %in% model\_factors[  model\_factors$factors == predictors[i], ]$factor\_levels  if (any(!found)) {  # track which variable  var <- predictors[i]  # set to NA  test\_data[!found, predictors[i]] <- NA  # drop empty factor levels in test data  test\_data %>%  droplevels() -> test\_data  # issue warning to console  message(sprintf(paste0("Setting missing levels in '%s', only present",  " in test data but missing in train data,",  " to 'NA'."),  var))  }  }  return(test\_data)  }  ```  Run model against test set, using simple fit, no interaction. Use the model created above. Run against test data set and remove missing levels, otherwise, predict will fail if there are levels in test that weren't encountered in train.  ```{r}  #Run model from training set on test set  set.seed(9560)  #fit.pred1 <- predict(glm.simple, newdata = remove\_missing\_levels(fit=glm.simple, test\_data = test), type = "response")  fit.pred1 <- predict(glm.simple, newdata = test, type = "response")  #ROC curves  pred1 <- prediction(fit.pred1, test.y)  roc.perf1 = performance(pred1, measure = "tpr", x.measure = "fpr")  auc.val1 <- performance(pred1, measure = "auc")  auc.val1 <- auc.val1@y.values  plot(roc.perf1)  abline(a=0, b= 1)  text(x = .40, y = .6,paste("AUC = ", round(auc.val1[[1]],3), sep = ""))  ```  ### Desiscion Tree  We'll need to use a more balanced data set for a decsion tree. Running against the original data set returns nothing. So used the train set created by down sampling.  ```{r}  par(mfrow=c(1,1))  thetree <- tree(outcome ~ ., data = train, mincut=5)  summary(thetree)  plot(thetree)  text(thetree,pretty=0)  ```  Let's create a glm model using just the predictors above from decision tree, and see how it compares.  ```{r}  f <- as.formula(outcome ~ discharge\_disposition\_id + number\_inpatient)  glm.tree <- glm(formula = f, family = "binomial", data = train)  coef(glm.tree)  #Get training set predictions...We know they are biased but lets create ROC's.  #These are predicted probabilities from logistic model exp(b)/(1+exp(b))  fit.pred <- predict(glm.tree, newx = train.x, type = "response")  #Compare the prediction to the real outcome  head(fit.pred)  head(train.y)  #Create ROC curves  pred <- prediction(fit.pred, train.y)  roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")  auc.train <- performance(pred, measure = "auc")  auc.train <- auc.train@y.values  #Plot ROC  plot(roc.perf)  abline(a=0, b= 1) #Ref line indicating poor performance  text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))  ```  Now apply it against test set  ```{r}  #Run model from descision tree predictors  set.seed(9560)  #fit.pred.tree <- predict(glm.tree, newdata = remove\_missing\_levels(fit=glm.tree, test\_data = test), type = "response")  fit.pred.tree <- predict(glm.tree, newdata = test, type = "response")  #ROC curves  pred.tree <- prediction(fit.pred.tree, test.y)  roc.perf.tree = performance(pred.tree, measure = "tpr", x.measure = "fpr")  auc.val.tree <- performance(pred.tree, measure = "auc")  auc.val.tree <- auc.val.tree@y.values  plot(roc.perf.tree)  abline(a=0, b= 1)  text(x = .40, y = .6,paste("AUC = ", round(auc.val.tree[[1]],3), sep = ""))  ```  ### Complex Logistic Example, Let's include sqrt of number\_inpatient  ```{r}  #glmnet requires a matrix  set.seed(9560)  f <- as.formula(outcome ~ . + I(number\_inpatient^.5))  train.x <- model.matrix(f, train)  cvfit <- cv.glmnet(train.x,train.y, family = "binomial", type.measure = "class", nlambda = 1000)  plot(cvfit)  coef(cvfit, s = "lambda.min")  #rownames(coef(cvfit, s = 'lambda.min'))[coef(cvfit, s = 'lambda.min')[,1]!=0]  #Get training set predictions...We know they are biased but lets create ROC's.  #These are predicted probabilities from logistic model exp(b)/(1+exp(b))  fit.pred <- predict(cvfit, newx = train.x, type = "response")  #Compare the prediction to the real outcome  head(fit.pred)  head(train.y)  #Create ROC curves  pred <- prediction(fit.pred[,1], train.y)  roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")  auc.train <- performance(pred, measure = "auc")  auc.train <- auc.train@y.values  #Plot ROC  plot(roc.perf)  abline(a=0, b= 1) #Ref line indicating poor performance  text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))  ```  Now use coefficients determined above, create a new formula and create complex model  ```{r}  f <- as.formula(outcome ~ discharge\_disposition\_id + time\_in\_hospital + num\_lab\_procedures + num\_medications + number\_emergency + number\_inpatient + diag\_1 + diag\_2 + number\_diagnoses + metformin + diabetesMed + ageGrp + I(number\_inpatient^.5))  glm.complex <- glm(formula = f, family = "binomial", data = train)  coef(glm.complex)  #Get training set predictions...We know they are biased but lets create ROC's.  #These are predicted probabilities from logistic model exp(b)/(1+exp(b))  fit.pred <- predict(glm.complex, newx = train.x, type = "response")  #Compare the prediction to the real outcome  head(fit.pred)  head(train.y)  #Create ROC curves  pred <- prediction(fit.pred, train.y)  roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")  auc.train <- performance(pred, measure = "auc")  auc.train <- auc.train@y.values  #Plot ROC  plot(roc.perf)  abline(a=0, b= 1) #Ref line indicating poor performance  text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))  ```  ```{r}  #Run model from complex train on test  set.seed(9560)  #fit.pred.complex <- predict(glm.complex, newdata = remove\_missing\_levels(fit=glm.complex, test\_data = test), type = "response")  fit.pred.complex <- predict(glm.complex, newdata = test, type = "response")  #ROC curves  pred1 <- prediction(fit.pred.complex, test.y)  roc.perf.complex = performance(pred1, measure = "tpr", x.measure = "fpr")  auc.val.complex <- performance(pred1, measure = "auc")  auc.val.complex <- auc.val.complex@y.values  plot(roc.perf.complex)  abline(a=0, b= 1)  text(x = .40, y = .6,paste("AUC = ", round(auc.val1[[1]],3), sep = ""))  ```  ### K-NN  ```{r}  # do lots of 'dummy' transforms for knn because it cannot take factor vars  train2 <- train  test2 <- test  # training data  gender <- as.data.frame(dummy.code(train2$gender))  race <- as.data.frame(dummy.code(train2$race))  admission\_type\_id <- as.data.frame(dummy.code(train2$admission\_type\_id))  names(admission\_type\_id) <- paste('admission\_type\_id.', names(admission\_type\_id), sep = '')  discharge\_disposition\_id <- as.data.frame(dummy.code(train2$discharge\_disposition\_id))  names(discharge\_disposition\_id) <- paste('discharge\_disposition\_id.', names(discharge\_disposition\_id), sep = '')  admission\_source\_id <- as.data.frame(dummy.code(train2$admission\_source\_id))  names(admission\_source\_id) <- paste('admission\_source\_id.', names(admission\_source\_id), sep = '')  diag\_1 <- as.data.frame(dummy.code(train2$diag\_1))  names(diag\_1) <- paste('diag\_1.', names(diag\_1), sep = '')  diag\_2 <- as.data.frame(dummy.code(train2$diag\_2))  names(diag\_2) <- paste('diag\_2.', names(diag\_2), sep = '')  diag\_3 <- as.data.frame(dummy.code(train2$diag\_3))  names(diag\_3) <- paste('diag\_3.', names(diag\_3), sep = '')  max\_glu\_serum <- as.data.frame(dummy.code(train2$max\_glu\_serum))  names(max\_glu\_serum) <- paste('max\_glu\_serum.', names(max\_glu\_serum), sep = '')  A1Cresult <- as.data.frame(dummy.code(train2$A1Cresult))  names(A1Cresult) <- paste('A1Cresult.', names(A1Cresult), sep = '')  metformin <- as.data.frame(dummy.code(train2$metformin))  names(metformin) <- paste('metformin.', names(metformin), sep = '')  repaglinide <- as.data.frame(dummy.code(train2$repaglinide))  names(repaglinide) <- paste('repaglinide.', names(repaglinide), sep = '')  nateglinide <- as.data.frame(dummy.code(train2$nateglinide))  names(nateglinide) <- paste('nateglinide.', names(nateglinide), sep = '')  chlorpropamide <- as.data.frame(dummy.code(train2$chlorpropamide))  names(chlorpropamide) <- paste('chlorpropamide.', names(chlorpropamide), sep = '')  glimepiride <- as.data.frame(dummy.code(train2$glimepiride))  names(glimepiride) <- paste('glimepiride.', names(glimepiride), sep = '')  acetohexamide <- as.data.frame(dummy.code(train2$acetohexamide))  names(acetohexamide) <- paste('acetohexamide.', names(acetohexamide), sep = '')  glipizide<- as.data.frame(dummy.code(train2$glipizide))  names(glipizide) <- paste('glipizide.', names(glipizide), sep = '')  glyburide <- as.data.frame(dummy.code(train2$glyburide))  names(glyburide) <- paste('glyburide.', names(glyburide), sep = '')  tolbutamide <- as.data.frame(dummy.code(train2$tolbutamide))  names(tolbutamide) <- paste('tolbutamide.', names(tolbutamide), sep = '')  pioglitazone <- as.data.frame(dummy.code(train2$pioglitazone))  names(pioglitazone) <- paste('pioglitazone.', names(pioglitazone), sep = '')  rosiglitazone <- as.data.frame(dummy.code(train2$rosiglitazone))  names(rosiglitazone) <- paste('rosiglitazone.', names(rosiglitazone), sep = '')  acarbose <- as.data.frame(dummy.code(train2$acarbose))  names(acarbose) <- paste('acarbose.', names(acarbose), sep = '')  miglitol <- as.data.frame(dummy.code(train2$miglitol))  names(miglitol) <- paste('miglitol.', names(miglitol), sep = '')  troglitazone <- as.data.frame(dummy.code(train2$troglitazone))  names(troglitazone) <- paste('troglitazone.', names(troglitazone), sep = '')  tolazamide <- as.data.frame(dummy.code(train2$tolazamide))  names(tolazamide) <- paste('tolazamide.', names(tolazamide), sep = '')  insulin <- as.data.frame(dummy.code(train2$insulin))  names(insulin) <- paste('insulin.', names(insulin), sep = '')  glyburide.metformin <- as.data.frame(dummy.code(train2$glyburide.metformin))  names(glyburide.metformin) <- paste('glyburide.metformin.', names(glyburide.metformin), sep = '')  glipizide.metformin <- as.data.frame(dummy.code(train2$glipizide.metformin))  names(glipizide.metformin) <- paste('glipizide.metformin.', names(glipizide.metformin), sep = '')  metformin.rosiglitazone <- as.data.frame(dummy.code(train2$metformin.rosiglitazone))  names(metformin.rosiglitazone) <- paste('metformin.rosiglitazone.', names(metformin.rosiglitazone), sep = '')  glimepiride.pioglitazone <- as.data.frame(dummy.code(train2$glimepiride.pioglitazone))  names(glimepiride.pioglitazone) <- paste('glimepiride.pioglitazone.', names(glimepiride.pioglitazone), sep = '')  metformin.pioglitazone <- as.data.frame(dummy.code(train2$metformin.pioglitazone))  names(metformin.pioglitazone) <- paste('metformin.pioglitazone', names(metformin.pioglitazone), sep = '')  change <- as.data.frame(dummy.code(train2$change))  names(change) <- paste('change.', names(change), sep = '')  ageGrp <- as.data.frame(dummy.code(train2$ageGrp))  names(ageGrp) <- paste('ageGrp.', names(ageGrp), sep = '')  diabetesMed <- as.data.frame(dummy.code(train2$diabetesMed))  names(diabetesMed) <- paste('diabetesMed.', names(diabetesMed), sep = '')  train2 <- cbind(train2, gender, race, admission\_type\_id, discharge\_disposition\_id, admission\_source\_id, diag\_1, diag\_2, diag\_3,  max\_glu\_serum, A1Cresult, metformin, repaglinide, nateglinide, chlorpropamide, glimepiride, acetohexamide,  glipizide, glyburide, tolbutamide, pioglitazone, rosiglitazone, acarbose, miglitol, troglitazone, tolazamide,  insulin, glyburide.metformin, glipizide.metformin, metformin.rosiglitazone, glimepiride.pioglitazone,  metformin.pioglitazone, change, ageGrp, diabetesMed)  train2 <- train2 %>%  select(-one\_of(c('gender', 'race', 'admission\_type\_id', 'discharge\_disposition\_id', 'admission\_source\_id',  'diag\_1', 'diag\_2', 'diag\_3',  'max\_glu\_serum', 'A1Cresult', 'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',  'glimepiride', 'acetohexamide',  'glipizide', 'glyburide', 'tolbutamide', 'pioglitazone', 'rosiglitazone', 'acarbose',  'miglitol', 'troglitazone', 'tolazamide',  'insulin', 'glyburide.metformin', 'glipizide.metformin', 'metformin.rosiglitazone',  'glimepiride.pioglitazone', 'metformin.pioglitazone', 'change',  'ageGrp', 'diabetesMed', 'outcome')))  # testing data  gender <- as.data.frame(dummy.code(test2$gender))  race <- as.data.frame(dummy.code(test2$race))  admission\_type\_id <- as.data.frame(dummy.code(test2$admission\_type\_id))  names(admission\_type\_id) <- paste('admission\_type\_id.', names(admission\_type\_id), sep = '')  discharge\_disposition\_id <- as.data.frame(dummy.code(test2$discharge\_disposition\_id))  names(discharge\_disposition\_id) <- paste('discharge\_disposition\_id.', names(discharge\_disposition\_id), sep = '')  admission\_source\_id <- as.data.frame(dummy.code(test2$admission\_source\_id))  names(admission\_source\_id) <- paste('admission\_source\_id.', names(admission\_source\_id), sep = '')  diag\_1 <- as.data.frame(dummy.code(test2$diag\_1))  names(diag\_1) <- paste('diag\_1.', names(diag\_1), sep = '')  diag\_2 <- as.data.frame(dummy.code(test2$diag\_2))  names(diag\_2) <- paste('diag\_2.', names(diag\_2), sep = '')  diag\_3 <- as.data.frame(dummy.code(test2$diag\_3))  names(diag\_3) <- paste('diag\_3.', names(diag\_3), sep = '')  max\_glu\_serum <- as.data.frame(dummy.code(test2$max\_glu\_serum))  names(max\_glu\_serum) <- paste('max\_glu\_serum.', names(max\_glu\_serum), sep = '')  A1Cresult <- as.data.frame(dummy.code(test2$A1Cresult))  names(A1Cresult) <- paste('A1Cresult.', names(A1Cresult), sep = '')  metformin <- as.data.frame(dummy.code(test2$metformin))  names(metformin) <- paste('metformin.', names(metformin), sep = '')  repaglinide <- as.data.frame(dummy.code(test2$repaglinide))  names(repaglinide) <- paste('repaglinide.', names(repaglinide), sep = '')  nateglinide <- as.data.frame(dummy.code(test2$nateglinide))  names(nateglinide) <- paste('nateglinide.', names(nateglinide), sep = '')  chlorpropamide <- as.data.frame(dummy.code(test2$chlorpropamide))  names(chlorpropamide) <- paste('chlorpropamide.', names(chlorpropamide), sep = '')  glimepiride <- as.data.frame(dummy.code(test2$glimepiride))  names(glimepiride) <- paste('glimepiride.', names(glimepiride), sep = '')  acetohexamide <- as.data.frame(dummy.code(test2$acetohexamide))  names(acetohexamide) <- paste('acetohexamide.', names(acetohexamide), sep = '')  glipizide<- as.data.frame(dummy.code(test2$glipizide))  names(glipizide) <- paste('glipizide.', names(glipizide), sep = '')  glyburide <- as.data.frame(dummy.code(test2$glyburide))  names(glyburide) <- paste('glyburide.', names(glyburide), sep = '')  tolbutamide <- as.data.frame(dummy.code(test2$tolbutamide))  names(tolbutamide) <- paste('tolbutamide.', names(tolbutamide), sep = '')  pioglitazone <- as.data.frame(dummy.code(test2$pioglitazone))  names(pioglitazone) <- paste('pioglitazone.', names(pioglitazone), sep = '')  rosiglitazone <- as.data.frame(dummy.code(test2$rosiglitazone))  names(rosiglitazone) <- paste('rosiglitazone.', names(rosiglitazone), sep = '')  acarbose <- as.data.frame(dummy.code(test2$acarbose))  names(acarbose) <- paste('acarbose.', names(acarbose), sep = '')  miglitol <- as.data.frame(dummy.code(test2$miglitol))  names(miglitol) <- paste('miglitol.', names(miglitol), sep = '')  troglitazone <- as.data.frame(dummy.code(test2$troglitazone))  names(troglitazone) <- paste('troglitazone.', names(troglitazone), sep = '')  tolazamide <- as.data.frame(dummy.code(test2$tolazamide))  names(tolazamide) <- paste('tolazamide.', names(tolazamide), sep = '')  insulin <- as.data.frame(dummy.code(test2$insulin))  names(insulin) <- paste('insulin.', names(insulin), sep = '')  glyburide.metformin <- as.data.frame(dummy.code(test2$glyburide.metformin))  names(glyburide.metformin) <- paste('glyburide.metformin.', names(glyburide.metformin), sep = '')  glipizide.metformin <- as.data.frame(dummy.code(test2$glipizide.metformin))  names(glipizide.metformin) <- paste('glipizide.metformin.', names(glipizide.metformin), sep = '')  metformin.rosiglitazone <- as.data.frame(dummy.code(test2$metformin.rosiglitazone))  names(metformin.rosiglitazone) <- paste('metformin.rosiglitazone.', names(metformin.rosiglitazone), sep = '')  glimepiride.pioglitazone <- as.data.frame(dummy.code(test2$glimepiride.pioglitazone))  names(glimepiride.pioglitazone) <- paste('glimepiride.pioglitazone.', names(glimepiride.pioglitazone), sep = '')  metformin.pioglitazone <- as.data.frame(dummy.code(test2$metformin.pioglitazone))  names(metformin.pioglitazone) <- paste('metformin.pioglitazone', names(metformin.pioglitazone), sep = '')  change <- as.data.frame(dummy.code(test2$change))  names(change) <- paste('change.', names(change), sep = '')  ageGrp <- as.data.frame(dummy.code(test2$ageGrp))  names(ageGrp) <- paste('ageGrp.', names(ageGrp), sep = '')  diabetesMed <- as.data.frame(dummy.code(test2$diabetesMed))  names(diabetesMed) <- paste('diabetesMed.', names(diabetesMed), sep = '')  test2 <- cbind(test2, gender, race, admission\_type\_id, discharge\_disposition\_id, admission\_source\_id, diag\_1, diag\_2, diag\_3,  max\_glu\_serum, A1Cresult, metformin, repaglinide, nateglinide, chlorpropamide, glimepiride, acetohexamide,  glipizide, glyburide, tolbutamide, pioglitazone, rosiglitazone, acarbose, miglitol, troglitazone, tolazamide,  insulin, glyburide.metformin, glipizide.metformin, metformin.rosiglitazone, glimepiride.pioglitazone,  metformin.pioglitazone, change, ageGrp, diabetesMed)  test2 <- test2 %>%  select(-one\_of(c('gender', 'race', 'admission\_type\_id', 'discharge\_disposition\_id', 'admission\_source\_id',  'diag\_1', 'diag\_2', 'diag\_3',  'max\_glu\_serum', 'A1Cresult', 'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',  'glimepiride', 'acetohexamide',  'glipizide', 'glyburide', 'tolbutamide', 'pioglitazone', 'rosiglitazone', 'acarbose',  'miglitol', 'troglitazone', 'tolazamide',  'insulin', 'glyburide.metformin', 'glipizide.metformin', 'metformin.rosiglitazone',  'glimepiride.pioglitazone', 'metformin.pioglitazone', 'change',  'ageGrp', 'diabetesMed')))  # And train!  # make a large tunning grid  knn.tuningGrid <- expand.grid(k = seq(1:30))  train.control <-trainControl(method = "cv",  number = 2,  summaryFunction = twoClassSummary,  classProbs = TRUE,  savePredictions = TRUE  )  model.cv <-train(y = train$outcome,  x = train2,  method = 'knn',  trControl = train.control,  tuneGrid = knn.tuningGrid  )  model.cv  # plot metrics by k tuning grid  knn.metrics <-data.frame(  'k' = model.cv$results$k,  'ROC' = model.cv$results$ROC,  'Sens' = model.cv$results$Sens,  'Spec' = model.cv$results$Spec)  p1 <- knn.metrics %>%  ggplot(aes(x = k, y = ROC)) +  geom\_point() +  geom\_smooth(method = 'loess')  p2 <- knn.metrics %>%  ggplot(aes(x = k, y = Sens)) +  geom\_point() +  geom\_smooth(method = 'loess')  p3 <- knn.metrics %>%  ggplot(aes(x = k, y = Spec)) +  geom\_point() +  geom\_smooth(method = 'loess')  grid.arrange(p1, p2, p3, ncol = 3,  top = 'Metrics by k in Tuning Grid')  preds <- predict(model.cv, test2)  confusionMatrix(preds, test2$outcome)  #ROC curves  pred1 <- prediction(as.numeric(preds), test.y)  roc.perf1 = performance(pred1, measure = "tpr", x.measure = "fpr")  auc.val1 <- performance(pred1, measure = "auc")  auc.val1 <- auc.val1@y.values  plot(roc.perf1)  abline(a=0, b= 1)  text(x = .40, y = .6, paste("AUC = ", round(auc.val1[[1]], 3), sep = ""))  ```  ### LDA  Train Set  ```{r}  fit.lda <- lda(outcome ~ time\_in\_hospital + num\_lab\_procedures + num\_procedures + num\_medications + number\_outpatient + number\_emergency + number\_inpatient, data = train)  pred.lda <- predict(fit.lda, newdata = train)    preds <- pred.lda$posterior  preds <- as.data.frame(preds)    pred <- prediction(preds[,2],train.y)  roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")  auc.train <- performance(pred, measure = "auc")  auc.train <- auc.train@y.values  plot(roc.perf)  abline(a=0, b= 1)  text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))  ```    Test Set  ```{r}  pred.lda1 <- predict(fit.lda, newdata = test)    preds1 <- pred.lda1$posterior  preds1 <- as.data.frame(preds1)    pred1 <- prediction(preds1[,2],test.y)  roc.perf = performance(pred1, measure = "tpr", x.measure = "fpr")  auc.train <- performance(pred1, measure = "auc")  auc.train <- auc.train@y.values  plot(roc.perf)  abline(a=0, b= 1)  text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))  ```  ### PCA  ```{r}  #Another option here would be to do PCA among the continous predictors to see  #if they seperate out. Or a heatmap.  pc.result<-prcomp(data[,c("time\_in\_hospital","num\_lab\_procedures","number\_inpatient","num\_medications","number\_diagnoses","number\_emergency")],scale.=TRUE)  pc.scores<-pc.result$x  pc.scores<-data.frame(pc.scores)  pc.scores$outcome<-data$outcome  #Loadings for interpretation  pc.result$rotation  #Scree plot  pc.eigen<-(pc.result$sdev)^2  pc.prop<-pc.eigen/sum(pc.eigen)  pc.cumprop<-cumsum(pc.prop)  plot(1:6,pc.prop,type="l",main="Scree Plot",ylim=c(0,1),xlab="PC #",ylab="Proportion of Variation")  lines(1:6,pc.cumprop,lty=3)  #Use ggplot2 to plot the first few pc's  ggplot(data = pc.scores, aes(x = PC1, y = PC2)) +  geom\_point(aes(col=outcome), size=1)+  geom\_hline(yintercept = 0, colour = "gray65") +  geom\_vline(xintercept = 0, colour = "gray65") +  ggtitle("PCA plot ...")  ggplot(data = pc.scores, aes(x = PC1, y = PC3)) +  geom\_point(aes(col=outcome), size=1)+  geom\_hline(yintercept = 0, colour = "gray65") +  geom\_vline(xintercept = 0, colour = "gray65") +  ggtitle("PCA plot ...")  ggplot(data = pc.scores, aes(x = PC3, y = PC6)) +  geom\_point(aes(col=outcome), size=1)+  geom\_hline(yintercept = 0, colour = "gray65") +  geom\_vline(xintercept = 0, colour = "gray65") +  ggtitle("PCA plot ...")  ggplot(data = pc.scores, aes(x = PC5, y = PC6)) +  geom\_point(aes(col=outcome), size=1)+  geom\_hline(yintercept = 0, colour = "gray65") +  geom\_vline(xintercept = 0, colour = "gray65") +  ggtitle("PCA plot ...")  ``` |