Healthcare Analytics: Predicting Length of Stay

Hanmaro Song, Tyler Wolff, Andrew Zazueta

University of San Diego

Master of Science, Applied Data Science

503

Section 1

June 28, 2021

Author Note

We have no known conflict of interest to disclose.

Correspondence concerning this article should be addressed to Hanmaro Song, Tyler Wolff, or Andrew Zazueta, at University of San Diego, 5998 Alcala Park, San Diego, CA 92110.

Emails: hanmarosong@sandiego.edu, tylerwolff@sandiego.edu, azazueta@sandiego.edu

Abstract

In this paper, the objective is to predict how long a patient is going to stay in a room at a hospital using different machine learning algorithms. Predicting length of stay (LOS) is important for hospitals so they can provide the best care possible to their patients. With the COVID-19 pandemic, hospital room space has been scarce. Knowing if a patient is a high LOS risk can help hospital staff plan their treatments better and allocate their bed situation easier. The methodology for conducting these predictions examines factors like the amount of bed space the hospital has, the age of the patient, the amount deposited when they entered the hospital, the department the patient was admitted to, the severity of their issues, and more to accurately predict how long their stay would possibly be. The results gave accuracies above 35% for multiple models that were trained, with the best achieving an accuracy of 40%. The machine learning algorithms that made these predictions were Random Forests, Naïve Bayes, Flexible Discriminant Analysis (FDA), Multiple Discriminant Analysis (MDA), KNN, Elastic Net, Support Vector Machines (SVM), CART, C5.0, and Neural Networks (NN).

Keywords: Length of stay, machine learning, hospital, modeling

Contents

Data Set and the Objective of Study

The data set for this project was taken from a Kaggle competition page. There are 18 variables within the data set and each row corresponds to a different patient admitted to a certain hospital. The dataset contains multiple hospital codes and city codes from where the data was collected, and in total, there are 318,438 rows (not including the header) and 18 columns. The question presented for this study was: Can you "accurately predict the Length of Stay for each patient on a case by case basis so that the Hospitals can use this information for optimal resource allocation and better functioning" (Prabhavalkar, 2020). The columns that were helpful in making these predictions were 'hospital code,' 'hospital region code,' 'available extra rooms in hospital,' 'department,' 'ward type,' 'bed grade,' 'city code patient,' 'type of admission,' 'severity of illness,' 'visitors with patient,' 'age,' and 'admission deposit.'

The response variable 'stay' is split up into 11 classes. These classes are each 10-day spans, with the shortest stay class being 1-10 days and longest stay class being 'more than 100 days.' Most of the prediction variables were categorical in nature. The 'hospital code' feature was a number ranging from 1-31 which was a unique code for the hospital. The 'department' feature told which department overlooked the case. For example, the department could be radiotherapy or anesthesia to name a few. The 'bed grade' feature ranked the condition of the bed with a number 1-4. The 'type of admission' would take the values 'trauma,' emergency,' or 'urgent' depending on what the hospital registered it as. Lastly, the 'severity of the illness' variable would measure how sick the patient was at the time of admission. Other features not described in this section are self-explanatory. The objective of this study is to use different data cleaning methods, data preprocessing methods, and predictive modeling strategies learned this semester and apply them to this data set using R.

Overview and Plan of Project

Before machine learning algorithms could be applied to the data set, it needed to be cleaned and prepared using exploratory data analysis. The relationships between features and their distributions were examined first to ensure our models could perform optimally. Due to the amount of data given, some models needed to have a data set which was reduced in size so that the models could be made in a reasonable amount of time. Since 'Stay' is a categorical variable, machine learning algorithms like Random Forests, Naive Bayes, KNN, FDA, etc. can be used to predict which class each patient would most likely take.

The route taken with this project was to look at multiple algorithms learned this semester and apply them. Then, the performance of these models was examined to find the one which gave the highest accuracy. Accuracy is the best statistic to use to measure the effectiveness of the models due to the number of classes that were needed to be predicted. Using sensitivity or specificity was not as helpful due to each individual class having a widely different answer from each other. For example, the specificity of the class 20-31 days could be 63% and the sensitivity 65%, but for the 61-70 days class the specificity could be 95% and sensitivity 30%. Since each class has wildly different statistics, it was easiest to look at overall model accuracy for each class.

This approach to the project was taken due to the complexity of the data set used. Predicting 11 separate classes is much more difficult than predicting two or three. Because of this complexity, it was known that getting a high accuracy was unlikely. To increase the odds of finding the optimal model, multiple algorithms were selected to find the one that fit the data best. Then, once the model with the highest accuracy was found, it could then be tuned to improve it further. The algorithms that were chosen to predict our response variable were Random Forests,

Naïve Bayes (NB), Flexible Discriminant Analysis (FDA), Multiple Discriminant Analysis (MDA), KNN, Elastic Net, Support Vector Machines (SVM), CART, C5.0, and Neural Networks (NN).

Exploratory Data Analysis

As mentioned above, among 318438 data samples there are two columns that contain NA values which are “Bed.Grade” and “City_Code_Patient”. They each contain 113 and 4532 missing values. Because compared to the whole dataset, their number is relatively small, dropping them instead of imputing those missing values will not do much harm to the training and predicting of a model. The following shows the brief structure of the dataset.

Table 1

Sample Structure of the Healthcare Analytics Dataset

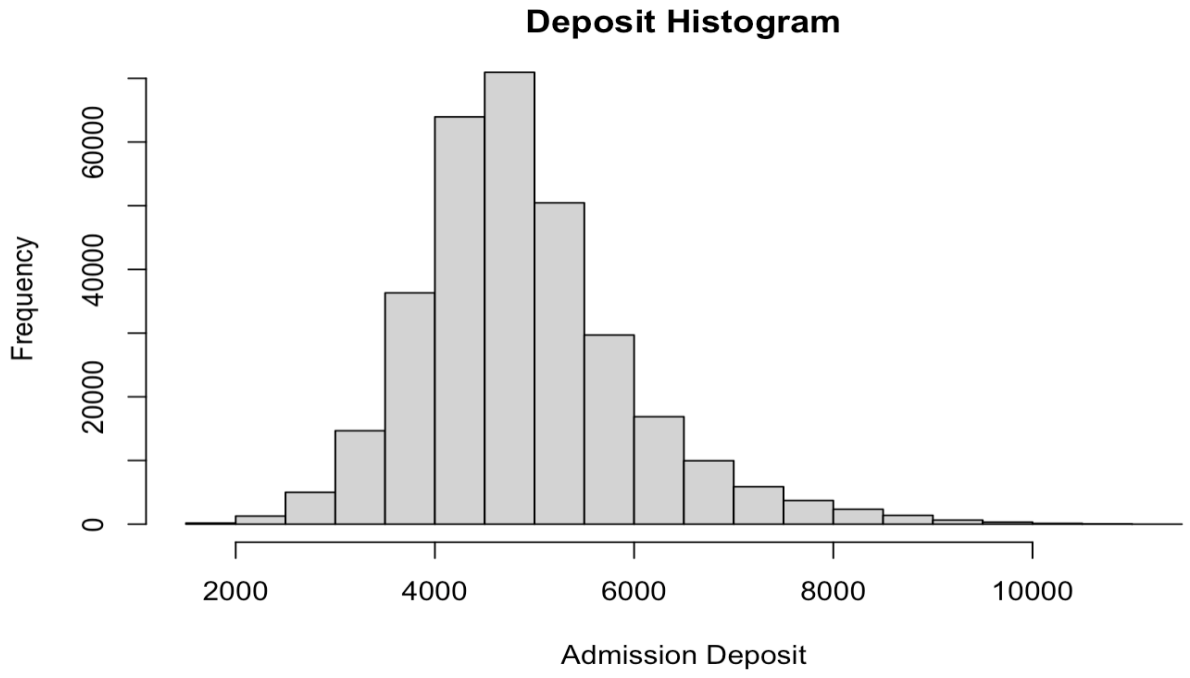
```
'data.frame':   313793 obs. of  17 variables:
 $ Hospital_code                   : int  8 2 10 26 26 23 32 23 1 10 ...
 $ Hospital_type_code              : chr  "c" "c" "e" "b" ...
 $ City_Code_Hospital              : int  3 5 1 2 2 6 9 6 10 1 ...
 $ Hospital_region_code            : chr  "Z" "Z" "X" "Y" ...
 $ Available.Extra.Rooms.in.Hospital: int  3 2 2 2 2 2 1 4 2 2 ...
 $ Department                      : chr  "radiotherapy" "radiotherapy" "anesthesia" "radiotherapy" ...
 $ Ward_Type                       : chr  "R" "S" "S" "R" ...
 $ Ward_Facility_Code              : chr  "F" "F" "E" "D" ...
 $ Bed.Grade                       : num  2 2 2 2 2 2 3 3 4 3 ...
 $ patientid                       : int  31397 31397 31397 31397 31397 31397 31397 31397 31397 31397 ...
 $ City_Code_Patient               : num  7 7 7 7 7 7 7 7 7 7 ...
 $ Type.of.Admission               : chr  "Emergency" "Trauma" "Trauma" "Trauma" ...
 $ Severity.of.Illness             : chr  "Extreme" "Extreme" "Extreme" "Extreme" ...
 $ Visitors.with.Patient           : int  2 2 2 2 2 2 2 2 2 2 ...
 $ Age                             : chr  "51-60" "51-60" "51-60" "51-60" ...
 $ Admission_Deposit               : num  4911 5954 4745 7272 5558 ...
 $ Stay                            : chr  "0-10" "41-50" "31-40" "41-50" ...
```

One thing to note here is that there originally was a feature named case_id but this was removed at the beginning because it was used as an index number as a unique identifier which doesn’t affect anything to the prediction or the “Stay” feature. In the table above, many of them are in char type. To this matter, we converted chr types into numerical categories by assigning specific

numbers representing a subgroup so that we can check out distribution of each unique value as well as feed them into a model later. For example with the “Age” group, there are 10 different sub-groups such as “0-10”, “11-20”, “21-30”, and all the way to “91-100”. These strings of range were converted so that they are now in 1 to 10 numerical values. We applied the same thing to the “Hospital_type_code”, “Hospital_region_code”, “Ward_Type”, “Ward_Facility_Code”, and “Stay” predictors so that they only contain Integer type values. After that, we took a look at the “Admission_Deposit” which was the only numerical feature among the dataset and the following shows the distribution of it.

Figure 1

Histogram of Admission Deposit

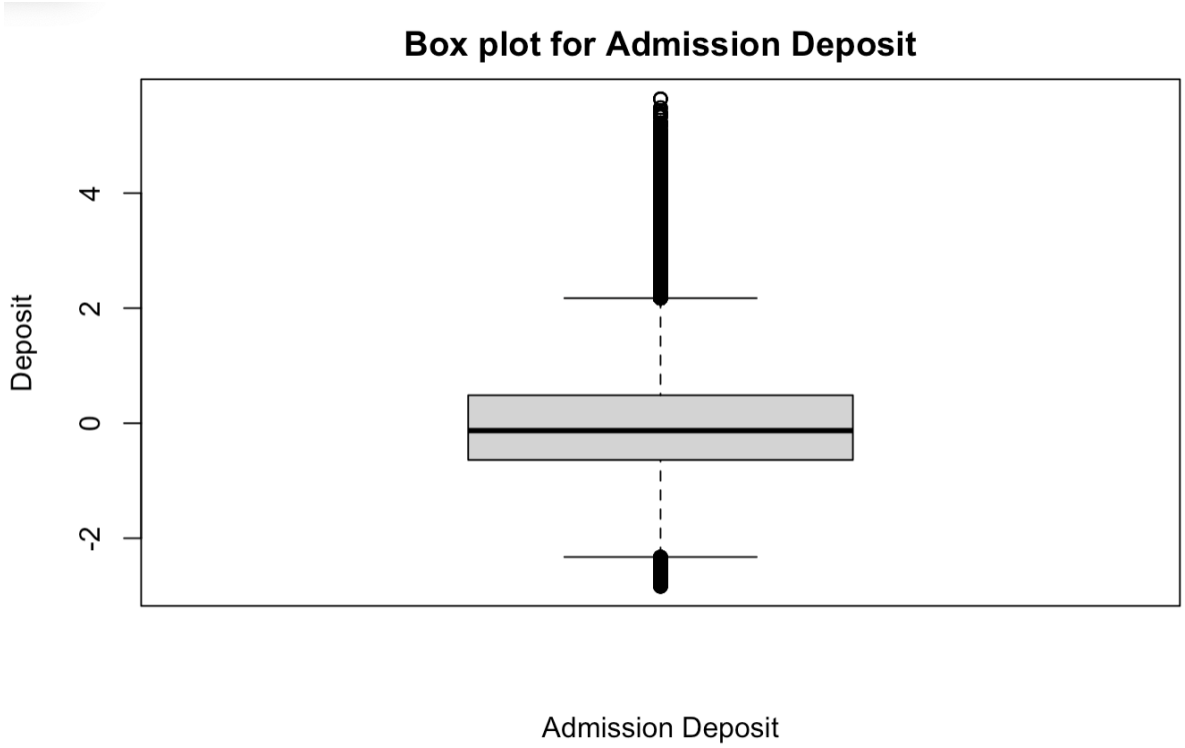


The plot above shows quite like a normal distribution although the right tail is a bit longer than the left. Because it’s not skewed much to have an impact on the training, we did not apply log

transformation on this feature but still standardized them so that they are centered at 0 with the variance of 0. After that using boxplot on the feature, we generated the following plot to review.

Figure 2

Boxplot of Admission Deposit



There seems to be quite many samples outside the Q1 and Q3 so using the 95% confidence interval, we removed any outliers and samples that are below 2.5% (value of -1.6625) and above 97.5% (value of 2.4765) of the data.

After all the steps mentioned above, now we have the dataframe with only numeric values and the following **Figure 3** shows the transformed dataset.

Figure 3

First 6 rows of the Dataset after Transformation

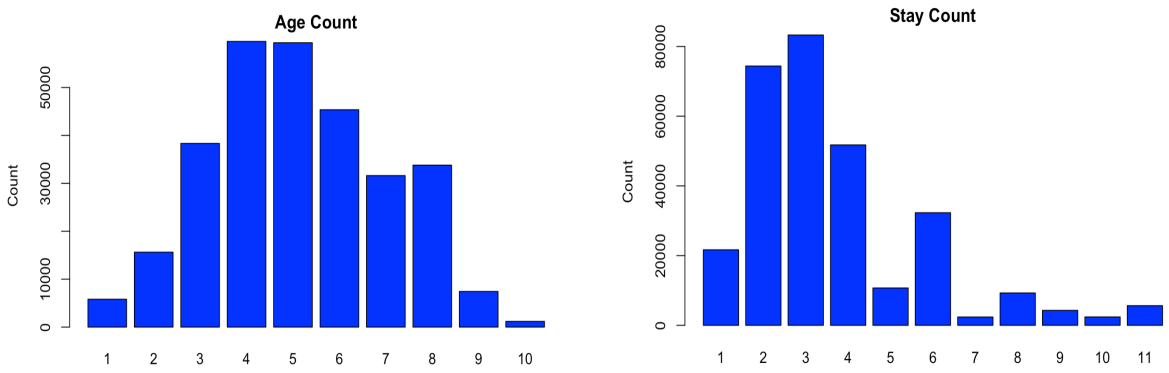
Description: df [6 × 25]

Bed.Grade <dbl>	patientid <int>	City_Code_Patient <dbl>	Visitors.with.Patient <int>	Age <int>	Admission_Deposit <dbl>	Stay <int>	Departmentanesthesia <dbl>	Departmentgynecology <dbl>
2	31397	7	2	6	4911	1	0	0
2	31397	7	2	6	5954	5	0	0
2	31397	7	2	6	4745	4	1	0
2	31397	7	2	6	7272	5	0	0
2	31397	7	2	6	5558	5	0	0
2	31397	7	2	6	4449	2	1	0

Now we checked the frequency distribution of "Age" and "Stay" features to see if there is any imbalance within the data.

Figure 4

Frequency Distribution of "Age" (left) and "Stay" (right)



The count of groups in 1, 9, and 10 are relatively small than other groups which quite makes sense because they represent the age in 0-10, 81-90, and 91-100. But when we look at the "Stay" frequency, it is easy to say that the problem is heavily class-imbalanced. Their actual numbers are as follows.

Figure 5

Number of each Class in Dataset

1	2	3	4	5	6	7	8	9	10	11
21669	74390	83294	51752	10715	32305	2383	9285	4286	2406	5625

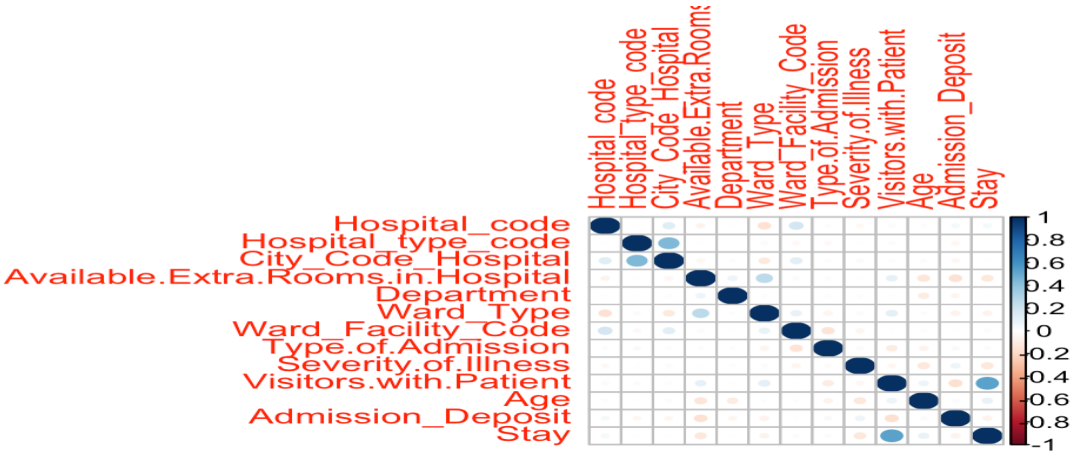
The class with the lowest count is class 7 (61-70 days) with only 2383 samples. There are two ways of dealing with this problem, oversampling and undersampling, which we did both.

The next thing we did is to figure out features that have near-zero-variance so we could drop them. Using *nearZeroVar* function, we concluded there are no columns that we have to drop as degenerate predictors.

The **Figure 6** in the next page shows the correlation between each feature of the dataset. It is a little hard to see because of the names of columns but we can see that there are some predictors that have positive and negative correlation such as among the Severity group and Department group. A note here is it makes sense that there exists correlation among the Severity group and Department group because if a patient is heavily injured (Severity.of.Illness.Extreme), it means other values of the Severity group will just be 0. Also it's most likely that each department has to work with each other to treat a patient so that it's possible to have multiple different departments at the same time for a given record.

Figure 6

Correlation Plot between each Feature



Data Preparation

There are a few columns that we removed before feeding them into a model which are "case_id", "patientid", "City_Code_Hospital", "City_Code_Patient", "Bed.Grade", "Hospital_region_code", and "Hospital_type_code". One reason for removing the first two columns is that they are unique to each patient and do not provide any information on how long a specific patient will stay at a hospital. In a similar sense, "City_Code_Hospital" and "Hospital_type_code" were also removed from the training dataset because as shown in **Figure 6**, they have positive correlations among them that we decided to use only one of them. For example, when the hospital code is **c**, the city code value is either 3 or 5. And when it is **d**, the city code can be 5, 10, or 15. Because their relationship is one to many without any overlap between different hospital code values, we can drop the other two columns and only keep one.

After dropping unnecessary features and converting some into numeric values, we tried a few different approaches on which subset of the remaining dataset to use for different models. For example with a K-nearest neighbor algorithm, only numeric predictors such as "Hospital_code", "Available Extra Rooms in Hospital", and a few more others, are used which can be viewed under Appendix (code written by Andrew Zazueta). Other non-numeric features were also converted into numeric values so that each number represents a sub-group within that feature for other models such as Random Forest, NN, and SVM.

Model Implementation and Evaluation

The models we decided to use are as follows: Naive Bayes, FDA, MDA, KNN, SVM, NN, ElasticNet, C5.0, and CART.

Because there are 314,438 rows and among them are the class of 3 (“21-30 days”) with 83294 samples, simply predicting all values as 3 will give us an accuracy of around 26.489% which we set as a baseline model. Training all the models were done only using CPU as we only tried to use what we learned from the course (so no GPU usage).

The overall best model performance of the 9 different models tested was delivered by the Random Forest model in which we received a 40.19% of accuracy. The worst performance was KNN with an accuracy of 32.9%. The average of the 9 models was 32.9% of accuracy. **Table 2** depicts each of the 9 models and the resulting accuracy. Default tuning was used on the models to help with computing time, this was a limitation noted while completing the predictions. If time had allotted for advanced tuning of each model there is a chance that higher levels of accuracy would have been produced.

The following table shows briefly about the accuracy of the models we trained.

Table 2

Model Accuracy Table

Model	NB	FDA	MDA	KNN	RF	CART	C5.0	GLM	NN
Accuracy	0.3693	0.3667	0.3517	0.329	0.4019	0.35	0.36	0.3300	0.37

Conclusion

There are still rooms for improvement on each model. It is possible that after more cleaning and pre-processing of the dataset, we could create a model that could predict with close to perfect accuracy, sensitivity, and specificity. Mending the group of stay into month-long

categories (0-30,31-60,61-90,90+) could have helped processing time, and increased accuracy. Not only that, if we utilize the usage of GPU for faster computation for the training, we can build a more complex and complicated deep learning model that could potentially outperform all the other models. The scope of the course limited us on which models to use. For the given problem, we concluded that the Random Forest algorithm outperformed the others with the accuracy of 40.19%. In conclusion, we noted that more pre-processing, and an advanced deep learning model is needed to increase the accuracy of predicting length of stay for patients at a hospital.

References

Prabhavalkar, A. (2020, August). *AV: Healthcare Analytics II*. Kaggle. https://www.kaggle.com/nehaprabhavalkar/av-healthcare-analytics-ii

Appendix A

Hanmaro Song Tyler Wolff Andrew Zazueta

6/28/2021

Obtaining Data and Setting up Libraries

```
setwd("C:/Users/mzazu/OneDrive/Documents/USD papers/503/503_Project")
library("tidyverse")
library("caret")
library("e1071")
library("mda")
library("earth")
library("nnet")
library("pROC")
library("rpart")
library("C50")
healthcare <- read_csv("train_data.csv")
```

Cleaning and Preporation Phase

Part 1: Exploratory Data Analysis and Handling Missing Values

```
# Finding missing values
dim(healthcare[!complete.cases(healthcare),])
```

```
## [1] 4645   18
```

```
# All of the rows with missing values are from the variables "Bed Grade" and "City Code Patient".
# 0.04% of the bed grades are missing and 1.4% of the city Code Patients are missing from the
# data set.
sum(is.na(healthcare$`Bed Grade`))
```

```
## [1] 113
```

```
sum(is.na(healthcare$`Bed Grade`)) / dim(healthcare)[1]
```

```
## [1] 0.0003548571
```

```r
sum(is.na(healthcare$City_Code_Patient))
```

```
## [1] 4532
```

```r
sum(is.na(healthcare$City_Code_Patient)) / dim(healthcare)[1]
```

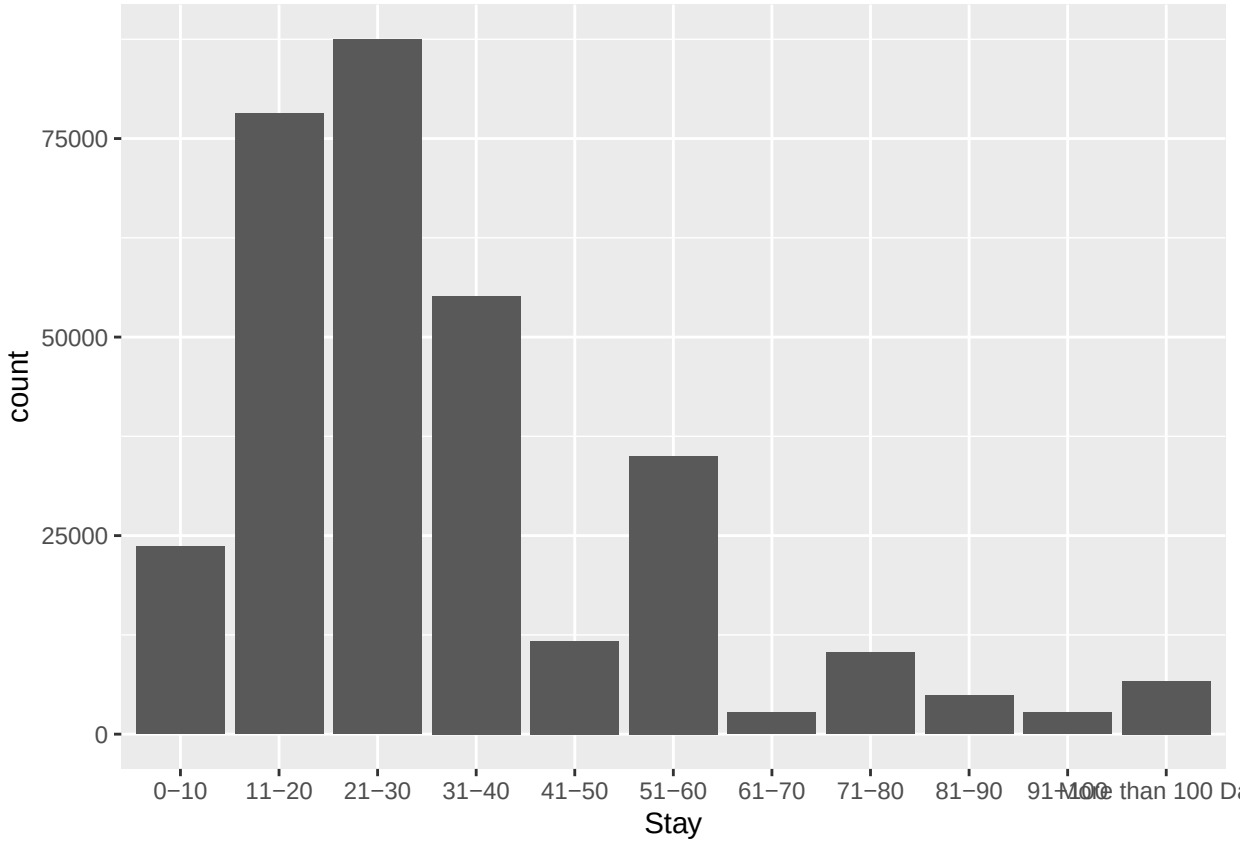
```
## [1] 0.01423197
```

```r
# The average bed grade is 2.6, so missing bed grades will be replaced with 3.
val <- ceiling(mean(healthcare$`Bed Grade`, na.rm=TRUE))
for(i in 1:nrow(healthcare)){
  if(is.na(healthcare[i, "Bed Grade"])==TRUE){
    healthcare[i, "Bed Grade"] <- val
  }
}

# Examining the city code patient values, the codes are all grouped together in a series. For
# example, the first 14 rows are 7's and the following 11 rows are 8's, and next the following
# rows are 2's. The NA's follow a similar pattern, where there will be a series of them
# sandwiched between another series of numbers. It is unclear to whether the missing NA's are
# numbers that they are between or they are a completely different number than the numbers they
# are between. The best guess is that it is the later, so all NA's will be replaced with a dummy
# value of 0 as its own unique city code for patients.

for(i in 1:nrow(healthcare)){
  if(is.na(healthcare[i, "City_Code_Patient"])==TRUE){
    healthcare[i, "City_Code_Patient"] <- 0
  }
}

# There are no more missing values
dim(healthcare[!complete.cases(healthcare),])
```

```
## [1]  0 18
```

```r
# Checking most prevalent response class. "21-30 days" is the most common class in the response
# variable "Stay" at 27.5%, so our models must have an accuracy better than this so we can be
# better than an "all positive" baseline model. Also, there is some class imbalance within the
# set, so that will have an effect on our model building effectiveness.
healthcare %>% ggplot(aes(Stay)) +
  geom_histogram(stat= "count")
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```




```
length(which(healthcare$Stay == "21-30"))/length(healthcare$Stay)
```

```
## [1] 0.2747505
```

```
# Checking for near zero variance columns; returned none
nearZeroVar(healthcare)
```

```
## integer(0)
```

```
# Removing features that will not be helpful to our modeling. Since case id and patient id are
# specific to a person, it will not help in our modeling
hc_removed <- healthcare %>%
  select(-c(case_id, patientid))

# There is a strong relationship between hospital code, hospital type code, and city code
# hospital, so only one of these columns is needed. For example, when the hospital type code is
# 'c', the city code hospital value is either 3 or 5. When the hospital type code is 'd', the
# city code hospital values are 5, 10, or 13. The same occurrences happened between hospital code
# and hospital type code, so we will keep "Hospital Code."
hc_removed <- hc_removed %>%
  select(-c(City_Code_Hospital, Hospital_type_code))
```

Part 2: Data Splitting

```
# Moving the response variable out of data frame
stay <- hc_removed$Stay
hc_no_stay <- hc_removed %>%
  select(-Stay)

# Splitting the data into training and test set
set.seed(1)
split <- createDataPartition(stay, p = .80, times = 1, list = FALSE)
trainPredictors <- hc_no_stay[split, ]
testPredictors <- hc_no_stay[-split,]
trainClasses <- stay[split]
testClasses <- stay[-split]

# Having data sets with predictors and response combined

trainCombo <- tibble(trainPredictors, trainClasses)
testCombo <- tibble(testPredictors, testClasses)

# Data sets with only numeric predictors

trainNum <- trainCombo %>%
  select(Hospital_code, `Available Extra Rooms in Hospital`, `Bed Grade`, City_Code_Patient,
         `Visitors with Patient`, Admission_Deposit)

testNum <- testCombo %>%
  select(Hospital_code, `Available Extra Rooms in Hospital`, `Bed Grade`, City_Code_Patient,
         `Visitors with Patient`, Admission_Deposit)

# Data set that is reduced for computational purposes. This data set will be used
# for KNN, so we will also only have numeric predictors. The data frame was reduced to
# 20% of what it used to be. Once this was completed, we split the training the test data 80:20.

split2 <- createDataPartition(stay, p = .2, times = 1, list = FALSE)
numReducedPred <- hc_no_stay[split2, ]
numReducedClass <- stay[split2]
split3 <- createDataPartition(numReducedClass, p = .80, times = 1, list = FALSE)

trainReducedPredictors <- numReducedPred[split3, ]
testReducedPredictors <- numReducedPred[-split3,]
trainReducedClasses <- numReducedClass[split3]
testReducedClasses <- numReducedClass[-split3]

trainReducedNum <- trainReducedPredictors %>%
  select(Hospital_code, `Available Extra Rooms in Hospital`, `Bed Grade`, City_Code_Patient,
         `Visitors with Patient`, Admission_Deposit)

testReducedNum <- testReducedPredictors %>%
  select(Hospital_code, `Available Extra Rooms in Hospital`, `Bed Grade`, City_Code_Patient,
         `Visitors with Patient`, Admission_Deposit)
```

Model Building Phase

```
# Naive Bayes
nb <- naiveBayes(x = trainPredictors, y = trainClasses)
confusionMatrix(predict(nb, testPredictors), as.factor(testClasses))
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction            0-10 11-20 21-30 31-40 41-50 51-60 61-70 71-80 81-90
##   0-10                 426   432   311   196    71    83     9    18     7
##   11-20               2187  6899  5106  2699   580   949    80   254    43
##   21-30               1986  7182 11421  4765  1377  1574   297   373    58
##   31-40                106   600   340  1597   154  1513    63   467    91
##   41-50                  4    29    61    23     8     6     2     3     1
##   51-60                  9   459   215  1661   131  2631    70   722   610
##   61-70                  1     1     3     6     1     5     2     3     0
##   71-80                  0     7     5    15     1    59     2    16    21
##   81-90                  0     4     4    11     4    29     4    35    16
##   91-100                 0     0     0     0     0     0     0     0     0
##   More than 100 Days     1    14    32    58    21   154    19   159   120
##                     Reference
## Prediction           91-100 More than 100 Days
##   0-10                    7                 15
##   11-20                  41                 65
##   21-30                 103                100
##   31-40                 124                 89
##   41-50                   2                  3
##   51-60                 183                495
##   61-70                   1                  3
##   71-80                   4                 35
##   81-90                   6                 32
##   91-100                  0                  0
##   More than 100 Days     82                499
##
## Overall Statistics
##
##                Accuracy : 0.3693
##                  95% CI : (0.3655, 0.373)
##     No Information Rate : 0.2748
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1832
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: 0-10 Class: 11-20 Class: 21-30 Class: 31-40
## Sensitivity              0.09025       0.4415       0.6527      0.14477
## Specificity              0.98051       0.7502       0.6143      0.93263
## Pos Pred Value           0.27048       0.3650       0.3906      0.31046
## Neg Pred Value           0.93086       0.8051       0.8236      0.83884
```

```
## Prevalence               0.07412       0.2454       0.2748      0.17322
## Detection Rate           0.00669       0.1083       0.1793      0.02508
## Detection Prevalence     0.02473       0.2968       0.4591      0.08078
## Balanced Accuracy        0.53538       0.5958       0.6335      0.53870
##                      Class: 41-50 Class: 51-60 Class: 61-70 Class: 71-80
## Sensitivity             0.0034072      0.37570    3.650e-03    0.0078049
## Specificity             0.9978152      0.91963    9.996e-01    0.9975824
## Pos Pred Value          0.0563380      0.36613    7.692e-02    0.0969697
## Neg Pred Value          0.9631722      0.92261    9.914e-01    0.9679766
## Prevalence              0.0368713      0.10997    8.605e-03    0.0321917
## Detection Rate          0.0001256      0.04132    3.141e-05    0.0002513
## Detection Prevalence    0.0022299      0.11284    4.083e-04    0.0025910
## Balanced Accuracy       0.5006112      0.64766    5.016e-01    0.5026936
##                      Class: 81-90 Class: 91-100 Class: More than 100 Days
## Sensitivity             0.0165460      0.000000                  0.373503
## Specificity             0.9979430      1.000000                  0.989414
## Pos Pred Value          0.1103448           NaN                  0.430544
## Neg Pred Value          0.9850321      0.991316                  0.986613
## Prevalence              0.0151851      0.008684                  0.020980
## Detection Rate          0.0002513      0.000000                  0.007836
## Detection Prevalence    0.0022770      0.000000                  0.018200
## Balanced Accuracy       0.5072445      0.500000                  0.681458
```

```r
# FDA
fda <- fda(trainClasses ~ .,
                  data = trainCombo,
                  method = earth)
confusionMatrix(predict(fda, testCombo), as.factor(testCombo$testClasses))
```

```
## Confusion Matrix and Statistics
##
##                      Reference
## Prediction            0-10 11-20 21-30 31-40 41-50 51-60 61-70 71-80 81-90
##   0-10                 381   216   227   159    69    59    16    22    14
##   11-20               2124  6563  5000  2073   541   582    95   131    38
##   21-30               2094  7385 11376  4310  1341   951   254   195    43
##   31-40                 91   660   478  1744   167  1529    56   506    29
##   41-50                  0     0     0     0     0     0     0     0     0
##   51-60                 10   457   138  1644   111  2125    34   573   144
##   61-70                  0     0     0     0     0     0     0     0     0
##   71-80                  0     0     0     0     0     0     0     0     0
##   81-90                 20   308   231   899    88  1388    63   348   507
##   91-100                 0     0     0     0     0     0     0     0     0
##   More than 100 Days     0    38    48   202    31   369    30   275   192
##                      Reference
## Prediction           91-100 More than 100 Days
##   0-10                   10                 18
##   11-20                  33                 54
##   21-30                  47                 60
##   31-40                 119                 88
##   41-50                   0                  0
##   51-60                 158                125
##   61-70                   0                  0
##   71-80                   0                  0
```

```
##   81-90                  78               335
##   91-100                  0                 0
##   More than 100 Days    108               656
##
## Overall Statistics
##
##                Accuracy : 0.3667
##                  95% CI : (0.363, 0.3705)
##     No Information Rate : 0.2748
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1933
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0-10 Class: 11-20 Class: 21-30 Class: 31-40
## Sensitivity             0.080720       0.4200       0.6501      0.15810
## Specificity             0.986262       0.7779       0.6388      0.92929
## Pos Pred Value          0.319899       0.3808       0.4055      0.31900
## Neg Pred Value          0.930565       0.8049       0.8282      0.84047
## Prevalence              0.074119       0.2454       0.2748      0.17322
## Detection Rate          0.005983       0.1031       0.1786      0.02739
## Detection Prevalence    0.018703       0.2706       0.4406      0.08585
## Balanced Accuracy       0.533491       0.5990       0.6445      0.54369
##                      Class: 41-50 Class: 51-60 Class: 61-70 Class: 71-80
## Sensitivity               0.00000      0.30344     0.000000      0.00000
## Specificity               1.00000      0.94012     1.000000      1.00000
## Pos Pred Value                NaN      0.38503          NaN          NaN
## Neg Pred Value            0.96313      0.91613     0.991395      0.96781
## Prevalence                0.03687      0.10997     0.008605      0.03219
## Detection Rate            0.00000      0.03337     0.000000      0.00000
## Detection Prevalence      0.00000      0.08667     0.000000      0.00000
## Balanced Accuracy         0.50000      0.62178     0.500000      0.50000
##                      Class: 81-90 Class: 91-100 Class: More than 100 Days
## Sensitivity              0.524302      0.000000                    0.49102
## Specificity              0.940077      1.000000                    0.97926
## Pos Pred Value           0.118875           NaN                    0.33658
## Neg Pred Value           0.992258      0.991316                    0.98898
## Prevalence               0.015185      0.008684                    0.02098
## Detection Rate           0.007962      0.000000                    0.01030
## Detection Prevalence     0.066974      0.000000                    0.03061
## Balanced Accuracy        0.732190      0.500000                    0.73514
```

```
# MDA
mda <- mda(trainClasses ~ .,
           data = trainNum)
confusionMatrix(predict(mda, testNum), as.factor(testCombo$testClasses))
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction            0-10 11-20 21-30 31-40 41-50 51-60 61-70 71-80 81-90
```

```
##   0-10                155    95    80    54    35    40     9     8     9
##   11-20              1726  5733  4680  2670   597   935   108   267    53
##   21-30              2655  8568 11997  4952  1396  1789   274   472    64
##   31-40               153   658   377  1493   145  1406    43   395    63
##   41-50                 0     0     0     0     0     0     0     0     0
##   51-60                31   531   309  1626   139  2418    77   607   554
##   61-70                 0     0     0     0     0     0     0     0     0
##   71-80                 0    38    26   158    13   220    19   134    72
##   81-90                 0     4     7    63     5    94    10    64    68
##   91-100                0     0     0     0     2     0     0     0     0
##   More than 100 Days    0     0    22    15    16   101     8   103    84
##                     Reference
## Prediction          91-100 More than 100 Days
##   0-10                   3                 14
##   11-20                 65                 86
##   21-30                111                 99
##   31-40                105                 74
##   41-50                  0                  0
##   51-60                157                379
##   61-70                  0                  0
##   71-80                 23                169
##   81-90                 14                114
##   91-100                 0                  0
##   More than 100 Days    75                401
##
## Overall Statistics
##
##                Accuracy : 0.3517
##                  95% CI : (0.348, 0.3555)
##     No Information Rate : 0.2748
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1565
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0-10 Class: 11-20 Class: 21-30 Class: 31-40
## Sensitivity             0.032839      0.36687       0.6856      0.13535
## Specificity             0.994115      0.76720       0.5587      0.93506
## Pos Pred Value          0.308765      0.33883       0.3705      0.30395
## Neg Pred Value          0.927745      0.78841       0.8243      0.83770
## Prevalence              0.074119      0.24540       0.2748      0.17322
## Detection Rate          0.002434      0.09003       0.1884      0.02344
## Detection Prevalence    0.007883      0.26570       0.5084      0.07713
## Balanced Accuracy       0.513477      0.56703       0.6222      0.53520
##                      Class: 41-50 Class: 51-60 Class: 61-70 Class: 71-80
## Sensitivity               0.00000      0.34528     0.000000     0.065366
## Specificity               1.00000      0.92219     1.000000     0.988026
## Pos Pred Value                NaN      0.35413          NaN     0.153670
## Neg Pred Value            0.96313      0.91935     0.991395     0.969495
## Prevalence                0.03687      0.10997     0.008605     0.032192
## Detection Rate            0.00000      0.03797     0.000000     0.002104
```

```
## Detection Prevalence      0.00000      0.10722     0.000000     0.013693
## Balanced Accuracy         0.50000      0.63374     0.500000     0.526696
##                      Class: 81-90 Class: 91-100 Class: More than 100 Days
## Sensitivity              0.070321     0.000e+00                   0.300150
## Specificity              0.994020     1.000e+00                   0.993199
## Pos Pred Value           0.153499     0.000e+00                   0.486061
## Neg Pred Value           0.985784     9.913e-01                   0.985125
## Prevalence               0.015185     8.684e-03                   0.020980
## Detection Rate           0.001068     0.000e+00                   0.006297
## Detection Prevalence     0.006957     3.141e-05                   0.012955
## Balanced Accuracy        0.532171     5.000e-01                   0.646674
```

```
# KNN
ctrl <- trainControl(method = "cv", number = 10)
knnFit <- train(trainReducedNum, trainReducedClasses,
                method = "knn",
                preProc = c("center", "scale"),
                trControl = ctrl)
confusionMatrix(predict(knnFit, testReducedNum), as.factor(testReducedClasses))
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction           0-10 11-20 21-30 31-40 41-50 51-60 61-70 71-80 81-90
##   0-10                 77   100    90    45    19    29     4     1     5
##   11-20               359  1221  1056   547   150   197    21    46     7
##   21-30               397  1277  1839   776   196   240    44    55    20
##   31-40                87   371   370   456    63   353    16   114    38
##   41-50                 4    12    24    19     4     6     1     1     0
##   51-60                19   130    96   314    28   460    18   132    75
##   61-70                 0     0     1     0     0     1     0     2     0
##   71-80                 1     9     8    30     2    40     1    24     5
##   81-90                 0     3     7     5     4    31     1     8    11
##   91-100                0     0     1     2     0     4     0     1     2
##   More than 100 Days    0     2     7    12     3    39     3    26    30
##                     Reference
## Prediction           91-100 More than 100 Days
##   0-10                    1                  6
##   11-20                  16                 15
##   21-30                  13                 12
##   31-40                  23                 25
##   41-50                   1                  0
##   51-60                  37                 87
##   61-70                   0                  0
##   71-80                   2                  8
##   81-90                   5                 15
##   91-100                  0                  2
##   More than 100 Days     12                 97
##
## Overall Statistics
##
##                Accuracy : 0.329
##                  95% CI : (0.3209, 0.3373)
##     No Information Rate : 0.2748
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1436
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0-10 Class: 11-20 Class: 21-30 Class: 31-40
## Sensitivity             0.081568       0.3907       0.5256      0.20671
## Specificity             0.974550       0.7487       0.6718      0.86130
## Pos Pred Value          0.204244       0.3359       0.3777      0.23800
## Neg Pred Value          0.929826       0.7907       0.7889      0.83820
## Prevalence              0.074144       0.2454       0.2748      0.17326
## Detection Rate          0.006048       0.0959       0.1444      0.03582
## Detection Prevalence    0.029610       0.2855       0.3824      0.15049
## Balanced Accuracy       0.528059       0.5697       0.5987      0.53400
##                      Class: 41-50 Class: 51-60 Class: 61-70 Class: 71-80
## Sensitivity             0.0085288      0.32857    0.0000000     0.058537
## Specificity             0.9944549      0.91740    0.9996831     0.991398
## Pos Pred Value          0.0555556      0.32951    0.0000000     0.184615
## Neg Pred Value          0.9632701      0.91708    0.9914362     0.969370
## Prevalence              0.0368363      0.10996    0.0085611     0.032202
## Detection Rate          0.0003142      0.03613    0.0000000     0.001885
## Detection Prevalence    0.0056550      0.10964    0.0003142     0.010210
## Balanced Accuracy       0.5014918      0.62299    0.4998416     0.524967
##                      Class: 81-90 Class: 91-100 Class: More than 100 Days
## Sensitivity              0.056995     0.0000000                  0.363296
## Specificity              0.993700     0.9990493                  0.989250
## Pos Pred Value           0.122222     0.0000000                  0.419913
## Neg Pred Value           0.985604     0.9913522                  0.986401
## Prevalence               0.015159     0.0086396                  0.020971
## Detection Rate           0.000864     0.0000000                  0.007619
## Detection Prevalence     0.007069     0.0009425                  0.018143
## Balanced Accuracy        0.525347     0.4995246                  0.676273
```

```
#CART
cart <- rpart(trainClasses ~ ., method = "class", data = trainNum)
confusionMatrix(predict(cart, testNum, "class"), as.factor(testClasses))
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction           0-10 11-20 21-30 31-40 41-50 51-60 61-70 71-80 81-90
##   0-10                  0     0     0     0     0     0     0     0     0
##   11-20              2022  7273  6181  2780   764   704   141   185    50
##   21-30              2510  6778 10207  4206  1207  1487   237   371    60
##   31-40                 0     0     0     0     0     0     0     0     0
##   41-50                 0     0     0     0     0     0     0     0     0
##   51-60               188  1576  1110  4045   377  4812   170  1494   857
##   61-70                 0     0     0     0     0     0     0     0     0
##   71-80                 0     0     0     0     0     0     0     0     0
##   81-90                 0     0     0     0     0     0     0     0     0
##   91-100                0     0     0     0     0     0     0     0     0
```

```
##   More than 100 Days     0     0     0     0     0     0     0     0     0
##                     Reference
## Prediction           91-100 More than 100 Days
##   0-10                    0                   0
##   11-20                  41                  82
##   21-30                  91                  90
##   31-40                   0                   0
##   41-50                   0                   0
##   51-60                 421                1164
##   61-70                   0                   0
##   71-80                   0                   0
##   81-90                   0                   0
##   91-100                  0                   0
##   More than 100 Days      0                   0
##
## Overall Statistics
##
##                Accuracy : 0.3501
##                  95% CI : (0.3464, 0.3538)
##     No Information Rate : 0.2748
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.163
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0-10 Class: 11-20 Class: 21-30 Class: 31-40
## Sensitivity              0.00000       0.4654       0.5833       0.0000
## Specificity              1.00000       0.7305       0.6311       1.0000
## Pos Pred Value               NaN       0.3596       0.3747          NaN
## Neg Pred Value           0.92588       0.8078       0.7999       0.8268
## Prevalence               0.07412       0.2454       0.2748       0.1732
## Detection Rate           0.00000       0.1142       0.1603       0.0000
## Detection Prevalence     0.00000       0.3176       0.4278       0.0000
## Balanced Accuracy        0.50000       0.5980       0.6072       0.5000
##                      Class: 41-50 Class: 51-60 Class: 61-70 Class: 71-80
## Sensitivity               0.00000      0.68713     0.000000      0.00000
## Specificity               1.00000      0.79883     1.000000      1.00000
## Pos Pred Value                NaN      0.29678          NaN          NaN
## Neg Pred Value            0.96313      0.95384     0.991395      0.96781
## Prevalence                0.03687      0.10997     0.008605      0.03219
## Detection Rate            0.00000      0.07556     0.000000      0.00000
## Detection Prevalence      0.00000      0.25461     0.000000      0.00000
## Balanced Accuracy         0.50000      0.74298     0.500000      0.50000
##                      Class: 81-90 Class: 91-100 Class: More than 100 Days
## Sensitivity               0.00000      0.000000                    0.00000
## Specificity               1.00000      1.000000                    1.00000
## Pos Pred Value                NaN           NaN                        NaN
## Neg Pred Value            0.98481      0.991316                    0.97902
## Prevalence                0.01519      0.008684                    0.02098
## Detection Rate            0.00000      0.000000                    0.00000
## Detection Prevalence      0.00000      0.000000                    0.00000
```

```
## Balanced Accuracy         0.50000      0.500000                  0.50000
```

```
# C50
C50 <- C5.0(x = trainPredictors, y = as.factor(trainClasses))
p <- predict(C50, testPredictors, type = "class")
confusionMatrix(p, as.factor(testClasses))
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction           0-10 11-20 21-30 31-40 41-50 51-60 61-70 71-80 81-90
##   0-10                922  1107   993   457   137   163    22    43    19
##   11-20              1749  6386  4927  2059   470   704    93   180    48
##   21-30              1555  5804  9288  3604  1146   870   211   177    85
##   31-40               321  1473  1458  2685   311  1689    87   453   126
##   41-50                61   188   272   203    64    91    16    32     7
##   51-60                79   508   390  1678   152  2750    66   751   315
##   61-70                10    19    30    22     7    30    11    10     9
##   71-80                12    94    57   192    27   291    17   215    46
##   81-90                 5    16    43    63    10   222     8    49   221
##   91-100                3    14     8    31    10    52     4    29     3
##   More than 100 Days    3    18    32    37    14   141    13   111    88
##                     Reference
## Prediction           91-100 More than 100 Days
##   0-10                   13                31
##   11-20                  39                63
##   21-30                  53                63
##   31-40                  95               117
##   41-50                   7                19
##   51-60                 190               250
##   61-70                   4                 8
##   71-80                  55               103
##   81-90                  12               124
##   91-100                 33                36
##   More than 100 Days     52               522
##
## Overall Statistics
##
##                Accuracy : 0.3627
##                  95% CI : (0.359, 0.3664)
##     No Information Rate : 0.2748
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1978
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: 0-10 Class: 11-20 Class: 21-30 Class: 31-40
## Sensitivity              0.19534       0.4087       0.5308      0.24340
## Specificity              0.94937       0.7850       0.7062      0.88357
## Pos Pred Value           0.23599       0.3820       0.4064      0.30459
## Neg Pred Value           0.93646       0.8032       0.7989      0.84788
```

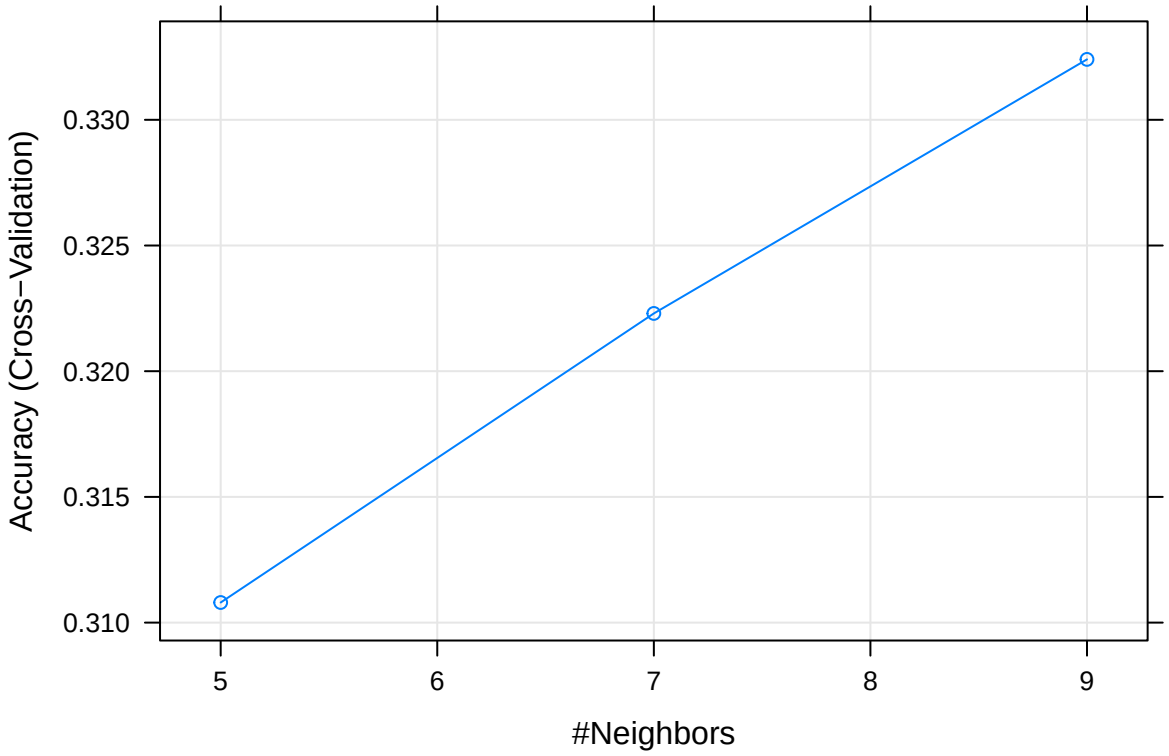
```
## Prevalence               0.07412       0.2454       0.2748      0.17322
## Detection Rate           0.01448       0.1003       0.1459      0.04216
## Detection Prevalence     0.06135       0.2625       0.3589      0.13842
## Balanced Accuracy        0.57236       0.5968       0.6185      0.56349
##                      Class: 41-50 Class: 51-60 Class: 61-70 Class: 71-80
## Sensitivity              0.027257      0.39269    0.0200730     0.104878
## Specificity              0.985391      0.92274    0.9976399     0.985494
## Pos Pred Value           0.066667      0.38575    0.0687500     0.193868
## Neg Pred Value           0.963585      0.92479    0.9915461     0.970674
## Prevalence               0.036871      0.10997    0.0086054     0.032192
## Detection Rate           0.001005      0.04318    0.0001727     0.003376
## Detection Prevalence     0.015075      0.11195    0.0025125     0.017415
## Balanced Accuracy        0.506324      0.65771    0.5088564     0.545186
##                      Class: 81-90 Class: 91-100 Class: More than 100 Days
## Sensitivity               0.22854     0.0596745                  0.390719
## Specificity               0.99120     0.9969902                  0.991836
## Pos Pred Value            0.28590     0.1479821                  0.506305
## Neg Pred Value            0.98814     0.9918056                  0.987007
## Prevalence                0.01519     0.0086839                  0.020980
## Detection Rate            0.00347     0.0005182                  0.008197
## Detection Prevalence      0.01214     0.0035018                  0.016190
## Balanced Accuracy         0.60987     0.5283324                  0.691277
```

Model Evaluation Phase

```
# Top predictors for FDA model
imp <- varImp(fda)
imp
```

```
##                                     Overall
## `Visitors with Patient`            100.00000
## Ward_TypeS                          54.45296
## `Type of Admission`Trauma           45.83576
## Ward_TypeR                          34.66234
## `Bed Grade`                         28.79172
## Admission_Deposit                   24.67205
## Ward_Facility_CodeC                 19.66470
## `Available Extra Rooms in Hospital` 14.50104
```

```
# Accuracy for KNN
plot(knnFit)
```


Accuracy (Cross−Validation)
0.330
0.325
0.320
0.315
0.310
5
6
7
8
9
#Neighbors

Appendix

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✓ ggplot2 3.3.3     ✓ purrr   0.3.4
## ✓ tibble  3.1.2     ✓ dplyr   1.0.6
## ✓ tidyr   1.1.3     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
train_health = read.csv("healthcare/train_data.csv")
```

Cleaning and Preporation Phase

Part 1: Exploratory Data Analysis and Handling Missing Values

```
summary(train_health)
```

```
##     case_id       Hospital_code   Hospital_type_code City_Code_Hospital
##  Min.   :     1   Min.   : 1.00   Length:318438      Min.   : 1.000
##  1st Qu.: 79610   1st Qu.:11.00   Class :character   1st Qu.: 2.000
##  Median :159220   Median :19.00   Mode  :character   Median : 5.000
##  Mean   :159220   Mean   :18.32                      Mean   : 4.772
##  3rd Qu.:238829   3rd Qu.:26.00                      3rd Qu.: 7.000
##  Max.   :318438   Max.   :32.00                      Max.   :13.000
##
##  Hospital_region_code Available.Extra.Rooms.in.Hospital  Department
##  Length:318438        Min.   : 0.000                   Length:318438
##  Class :character     1st Qu.: 2.000                   Class :character
##  Mode  :character     Median : 3.000                   Mode  :character
##                       Mean   : 3.198
##                       3rd Qu.: 4.000
##                       Max.   :24.000
##
##   Ward_Type          Ward_Facility_Code   Bed.Grade       patientid
##  Length:318438      Length:318438      Min.   :1.000   Min.   :     1
##  Class :character   Class :character   1st Qu.:2.000   1st Qu.: 32847
##  Mode  :character   Mode  :character   Median :3.000   Median : 65724
##                                        Mean   :2.626   Mean   : 65748
##                                        3rd Qu.:3.000   3rd Qu.: 98470
##                                        Max.   :4.000   Max.   :131624
##                                        NA's   :113
##  City_Code_Patient Type.of.Admission  Severity.of.Illness Visitors.with.Patient
##  Min.   : 1.000    Length:318438      Length:318438       Min.   : 0.000
##  1st Qu.: 4.000    Class :character   Class :character    1st Qu.: 2.000
##  Median : 8.000    Mode  :character   Mode  :character    Median : 3.000
##  Mean   : 7.252                                           Mean   : 3.284
##  3rd Qu.: 8.000                                           3rd Qu.: 4.000
##  Max.   :38.000                                           Max.   :32.000
##  NA's   :4532
##      Age            Admission_Deposit     Stay
##  Length:318438      Min.   : 1800     Length:318438
##  Class :character   1st Qu.: 4186     Class :character
##  Mode  :character   Median : 4741     Mode  :character
##                     Mean   : 4881
##                     3rd Qu.: 5409
##                     Max.   :11008
##
```

Because among 318438 training data there are only 4532 missing values for City_Code_Patient and only 113 for Bed.Grade, simply dropping those rows will not affect the performance of a model.

```
non_na_train_health = train_health[complete.cases(train_health), ]
dim(non_na_train_health)
```

```
## [1] 313793     18
```

```
summary(non_na_train_health)
```

```
##     case_id       Hospital_code   Hospital_type_code City_Code_Hospital
##  Min.   :     1   Min.   : 1.00   Length:313793      Min.   : 1.000
##  1st Qu.: 79271   1st Qu.:11.00   Class :character   1st Qu.: 2.000
##  Median :158950   Median :19.00   Mode  :character   Median : 5.000
##  Mean   :158938   Mean   :18.33                      Mean   : 4.778
##  3rd Qu.:238399   3rd Qu.:26.00                      3rd Qu.: 7.000
##  Max.   :318438   Max.   :32.00                      Max.   :13.000
##  Hospital_region_code Available.Extra.Rooms.in.Hospital  Department
##  Length:313793        Min.   : 0.000                    Length:313793
##  Class :character     1st Qu.: 2.000                    Class :character
##  Mode  :character     Median : 3.000                    Mode  :character
##                       Mean   : 3.196
##                       3rd Qu.: 4.000
##                       Max.   :24.000
##   Ward_Type          Ward_Facility_Code   Bed.Grade       patientid
##  Length:313793      Length:313793      Min.   :1.000   Min.   :     1
##  Class :character   Class :character   1st Qu.:2.000   1st Qu.: 32833
##  Mode  :character   Mode  :character   Median :3.000   Median : 65735
##                                        Mean   :2.623   Mean   : 65743
##                                        3rd Qu.:3.000   3rd Qu.: 98472
##                                        Max.   :4.000   Max.   :131624
##  City_Code_Patient Type.of.Admission  Severity.of.Illness Visitors.with.Patient
##  Min.   : 1.000    Length:313793      Length:313793       Min.   : 0.000
##  1st Qu.: 4.000    Class :character   Class :character    1st Qu.: 2.000
##  Median : 8.000    Mode  :character   Mode  :character    Median : 3.000
##  Mean   : 7.252                                           Mean   : 3.281
##  3rd Qu.: 8.000                                           3rd Qu.: 4.000
##  Max.   :38.000                                           Max.   :32.000
##      Age            Admission_Deposit     Stay
##  Length:313793      Min.   : 1800     Length:313793
##  Class :character   1st Qu.: 4188     Class :character
##  Mode  :character   Median : 4742     Mode  :character
##                     Mean   : 4882
##                     3rd Qu.: 5410
##                     Max.   :11008
```

```
head(non_na_train_health)
```

```
##   case_id Hospital_code Hospital_type_code City_Code_Hospital
## 1       1             8                  c                  3
## 2       2             2                  c                  5
## 3       3            10                  e                  1
## 4       4            26                  b                  2
## 5       5            26                  b                  2
## 6       6            23                  a                  6
##   Hospital_region_code Available.Extra.Rooms.in.Hospital   Department Ward_Type
## 1                    Z                                3 radiotherapy         R
## 2                    Z                                2 radiotherapy         S
## 3                    X                                2   anesthesia         S
## 4                    Y                                2 radiotherapy         R
## 5                    Y                                2 radiotherapy         S
## 6                    X                                2   anesthesia         S
##   Ward_Facility_Code Bed.Grade patientid City_Code_Patient Type.of.Admission
## 1                  F         2     31397                 7         Emergency
## 2                  F         2     31397                 7            Trauma
## 3                  E         2     31397                 7            Trauma
## 4                  D         2     31397                 7            Trauma
## 5                  D         2     31397                 7            Trauma
## 6                  F         2     31397                 7            Trauma
##   Severity.of.Illness Visitors.with.Patient   Age Admission_Deposit  Stay
## 1             Extreme                     2 51-60              4911  0-10
## 2             Extreme                     2 51-60              5954 41-50
## 3             Extreme                     2 51-60              4745 31-40
## 4             Extreme                     2 51-60              7272 41-50
## 5             Extreme                     2 51-60              5558 41-50
## 6             Extreme                     2 51-60              4449 11-20
```

```
# case_id is unique for each row that we can drop them
length(unique(non_na_train_health$case_id))
```

```
## [1] 313793
```

```
non_na_train_health = subset(non_na_train_health, select=-case_id)
```

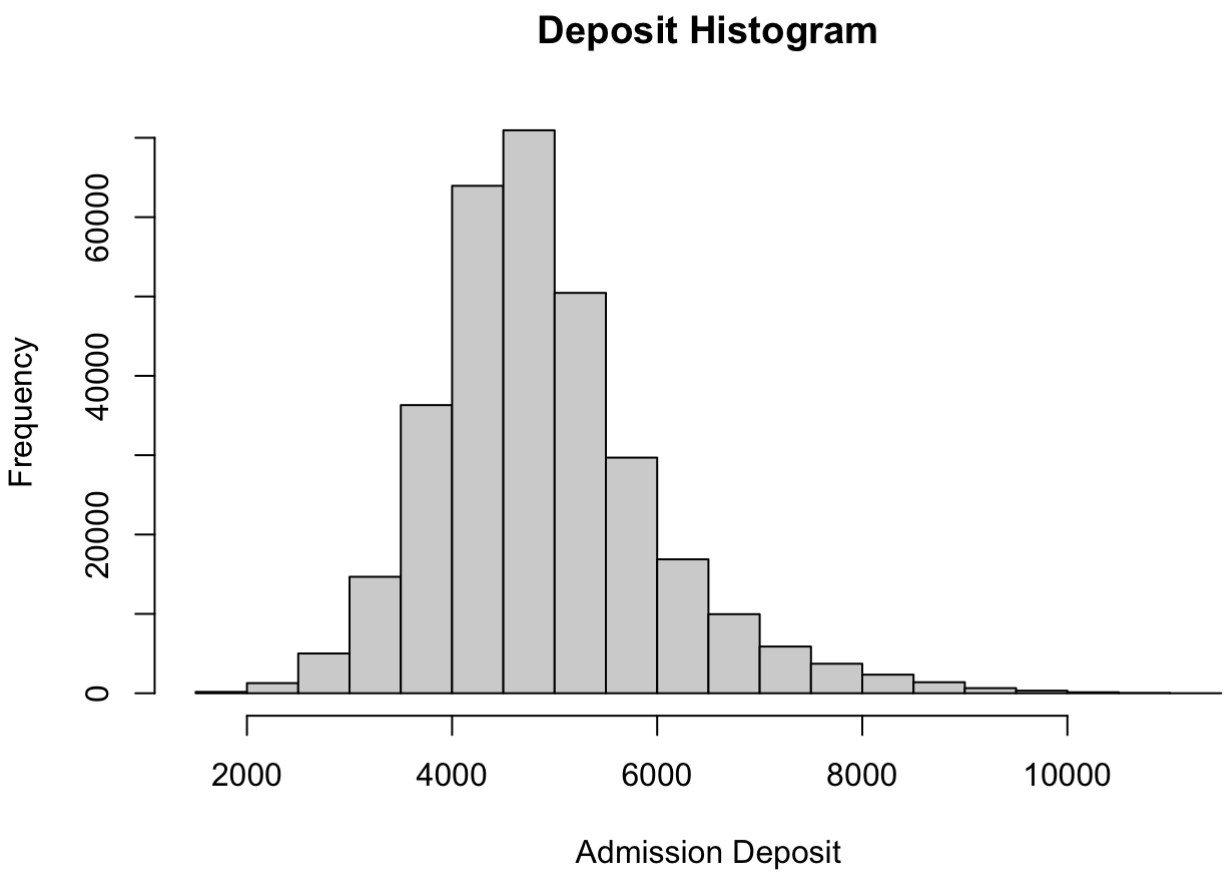
Things to consider 1. Does patientid affect the length of the stay? 2. Does City_Code_Patient affect the stay? 3. Does Visitors.with.Patient affect the stay?

```
str(non_na_train_health)
```

```
## 'data.frame':    313793 obs. of  17 variables:
##  $ Hospital_code                    : int  8 2 10 26 26 23 32 23 1 10 ...
##  $ Hospital_type_code               : chr  "c" "c" "e" "b" ...
##  $ City_Code_Hospital               : int  3 5 1 2 2 6 9 6 10 1 ...
##  $ Hospital_region_code             : chr  "Z" "Z" "X" "Y" ...
##  $ Available.Extra.Rooms.in.Hospital: int  3 2 2 2 2 2 1 4 2 2 ...
##  $ Department                       : chr  "radiotherapy" "radiotherapy" "anesthesia"
"radiotherapy" ...
##  $ Ward_Type                        : chr  "R" "S" "S" "R" ...
##  $ Ward_Facility_Code               : chr  "F" "F" "E" "D" ...
##  $ Bed.Grade                        : num  2 2 2 2 2 2 3 3 4 3 ...
##  $ patientid                        : int  31397 31397 31397 31397 31397 31397 31397
31397 31397 31397 ...
##  $ City_Code_Patient                : num  7 7 7 7 7 7 7 7 7 7 ...
##  $ Type.of.Admission                : chr  "Emergency" "Trauma" "Trauma" "Trauma" ...
##  $ Severity.of.Illness              : chr  "Extreme" "Extreme" "Extreme" "Extreme"
...
##  $ Visitors.with.Patient            : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ Age                              : chr  "51-60" "51-60" "51-60" "51-60" ...
##  $ Admission_Deposit                : num  4911 5954 4745 7272 5558 ...
##  $ Stay                             : chr  "0-10" "41-50" "31-40" "41-50" ...
```

One can see that “Admission_Deposit” is the only column with continuous feature. Others are categorical. Among them, Hospitcal_type_code, Hospital_region_code, Ward_Type, and Ward_Facility_Code can be converted into int type since it is easier to use for visualization and see any correlation.

```
# The histogram seems to be quite like normal distribution with a bit longer right-tail.
# Since not much skewness, skip standardization
hist(non_na_train_health$Admission_Deposit, xlab='Admission Deposit', main='Deposit Hist
ogram')
```




```
head(train_health, 1)
```

```
##   case_id Hospital_code Hospital_type_code City_Code_Hospital
## 1       1             8                   c                  3
##   Hospital_region_code Available.Extra.Rooms.in.Hospital   Department Ward_Type
## 1                    Z                                3 radiotherapy         R
##   Ward_Facility_Code Bed.Grade patientid City_Code_Patient Type.of.Admission
## 1                  F         2     31397                 7         Emergency
##   Severity.of.Illness Visitors.with.Patient   Age Admission_Deposit Stay
## 1             Extreme                     2 51-60              4911 0-10
```

Convert Hospital_type_code, Hospital_region_code, Ward_Type, and Ward_Facility_Code into INT values.

```
library(caret)

unique(non_na_train_health$Hospital_region_code)
```

```
## [1] "Z" "X" "Y"
```

```
unique(non_na_train_health$Hospital_type_code)
```

```
## [1] "c" "e" "b" "a" "f" "d" "g"
```

```
unique(non_na_train_health$Ward_Type)
```

```
## [1] "R" "S" "Q" "P" "T" "U"
```

```
unique(non_na_train_health$Ward_Facility_Code)
```

```
## [1] "F" "E" "D" "B" "A" "C"
```

```
convert_to_int = function(df, col_name) {

  unique_val = unique(df[, col_name])
  num_unique = length(unique_val)

  for (i in 1:num_unique) {
    df[df[col_name]==unique_val[i], col_name] = i
  }

  df[, col_name] = as.integer(df[, col_name])
  return(df)
}

preprocessed_health = subset(non_na_train_health, select=-c(Hospital_region_code, Bed.Gr
ade, patientid, City_Code_Patient))

preprocessed_health = convert_to_int(preprocessed_health, 'Department')
preprocessed_health = convert_to_int(preprocessed_health, 'Hospital_type_code')
preprocessed_health = convert_to_int(preprocessed_health, 'Ward_Type')
preprocessed_health = convert_to_int(preprocessed_health, 'Ward_Facility_Code')
preprocessed_health = convert_to_int(preprocessed_health, 'Type.of.Admission')
preprocessed_health = convert_to_int(preprocessed_health, 'Severity.of.Illness')


# Now convert Age into ordinal categorical feature
# 0-10 will be 1, 11-20 will be 2 and so on until 90-100 is 10
preprocessed_health[preprocessed_health$Age=='0-10', 'Age'] = 1
preprocessed_health[preprocessed_health$Age=='11-20', 'Age'] = 2
preprocessed_health[preprocessed_health$Age=='21-30', 'Age'] = 3
preprocessed_health[preprocessed_health$Age=='31-40', 'Age'] = 4
preprocessed_health[preprocessed_health$Age=='41-50', 'Age'] = 5
preprocessed_health[preprocessed_health$Age=='51-60', 'Age'] = 6
preprocessed_health[preprocessed_health$Age=='61-70', 'Age'] = 7
preprocessed_health[preprocessed_health$Age=='71-80', 'Age'] = 8
preprocessed_health[preprocessed_health$Age=='81-90', 'Age'] = 9
preprocessed_health[preprocessed_health$Age=='91-100', 'Age'] = 10

preprocessed_health$Age = as.integer(preprocessed_health$Age)

# Do the same thing for Stay. We have to convert it into number so that we can plot corr
elation later
unique(preprocessed_health$Stay)
```

```
##  [1] "0-10"              "41-50"             "31-40"
##  [4] "11-20"             "51-60"             "21-30"
##  [7] "71-80"             "More than 100 Days" "81-90"
## [10] "61-70"             "91-100"
```

```
preprocessed_health[preprocessed_health$Stay=='0-10', 'Stay'] = 1
preprocessed_health[preprocessed_health$Stay=='11-20', 'Stay'] = 2
preprocessed_health[preprocessed_health$Stay=='21-30', 'Stay'] = 3
preprocessed_health[preprocessed_health$Stay=='31-40', 'Stay'] = 4
preprocessed_health[preprocessed_health$Stay=='41-50', 'Stay'] = 5
preprocessed_health[preprocessed_health$Stay=='51-60', 'Stay'] = 6
preprocessed_health[preprocessed_health$Stay=='61-70', 'Stay'] = 7
preprocessed_health[preprocessed_health$Stay=='71-80', 'Stay'] = 8
preprocessed_health[preprocessed_health$Stay=='81-90', 'Stay'] = 9
preprocessed_health[preprocessed_health$Stay=='91-100', 'Stay'] = 10
preprocessed_health[preprocessed_health$Stay=='More than 100 Days', 'Stay'] = 11

preprocessed_health$Stay = as.integer(preprocessed_health$Stay)

head(preprocessed_health)
```

```
##   Hospital_code Hospital_type_code City_Code_Hospital
## 1             8                  1                  3
## 2             2                  1                  5
## 3            10                  2                  1
## 4            26                  3                  2
## 5            26                  3                  2
## 6            23                  4                  6
##   Available.Extra.Rooms.in.Hospital Department Ward_Type Ward_Facility_Code
## 1                                 3          1         1                  1
## 2                                 2          1         2                  1
## 3                                 2          2         2                  2
## 4                                 2          1         1                  3
## 5                                 2          1         2                  3
## 6                                 2          2         2                  1
##   Type.of.Admission Severity.of.Illness Visitors.with.Patient Age
## 1                 1                   1                     2   6
## 2                 2                   1                     2   6
## 3                 2                   1                     2   6
## 4                 2                   1                     2   6
## 5                 2                   1                     2   6
## 6                 2                   1                     2   6
##   Admission_Deposit Stay
## 1              4911    1
## 2              5954    5
## 3              4745    4
## 4              7272    5
## 5              5558    5
## 6              4449    2
```

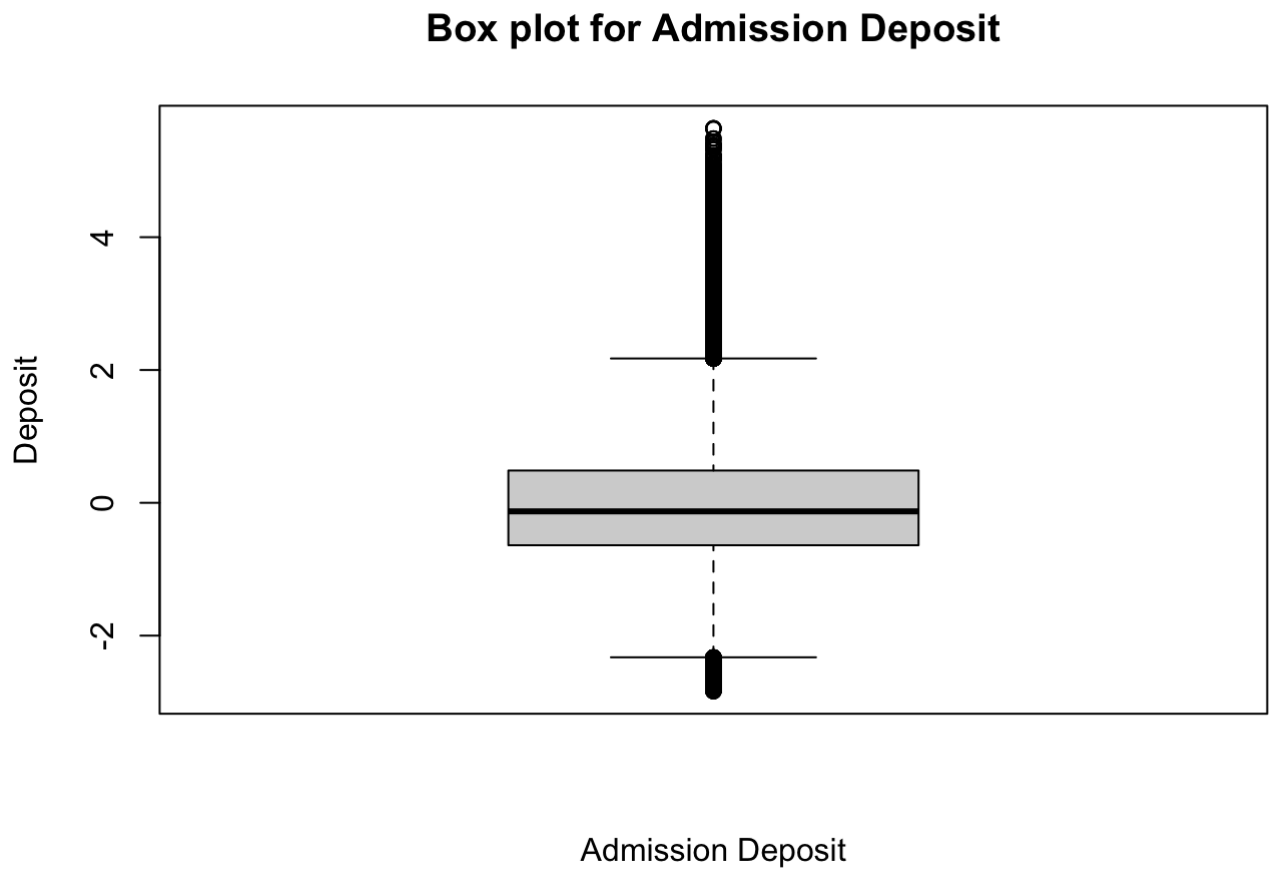
```
# Now the only chr columns are Department, Type.of.Admission, and Severity.of.Illness wh
ich can be one-hot-encoded
#ohe = dummyVars(' ~ Department +  Type.of.Admission + Severity.of.Illness', data=prepro
cessed_health)
#preprocessed_health = data.frame(subset(preprocessed_health, select=-c(Department, Typ
e.of.Admission, Severity.of.Illness)), predict(ohe, preprocessed_health))
```

Standardize columns

```
preprocessed_health$Type.of.Admission = scale(preprocessed_health[, 'Type.of.Admission'
])
preprocessed_health$Available.Extra.Rooms.in.Hospital = scale(preprocessed_health[, 'Ava
ilable.Extra.Rooms.in.Hospital'])
preprocessed_health$Visitors.with.Patient = scale(preprocessed_health[, 'Visitors.with.P
atient'])
preprocessed_health$Admission_Deposit = scale(preprocessed_health$Admission_Deposit)
```

Plots

```
# Using box for Admission Deposit, see if there is any outlier
boxplot(preprocessed_health$Admission_Deposit, xlab='Admission Deposit', ylab='Deposit',
main='Box plot for Admission Deposit')
```



There are quite some values outside the range of 1Q and 3Q. Using 95% confidence interval, drop possible outliers if there are not many samples.

```
print('2.5% and 97.5% values are')
```

```
## [1] "2.5% and 97.5% values are"
```

```
quantile(preprocessed_health$Admission_Deposit, c(0.025, 0.975))
```

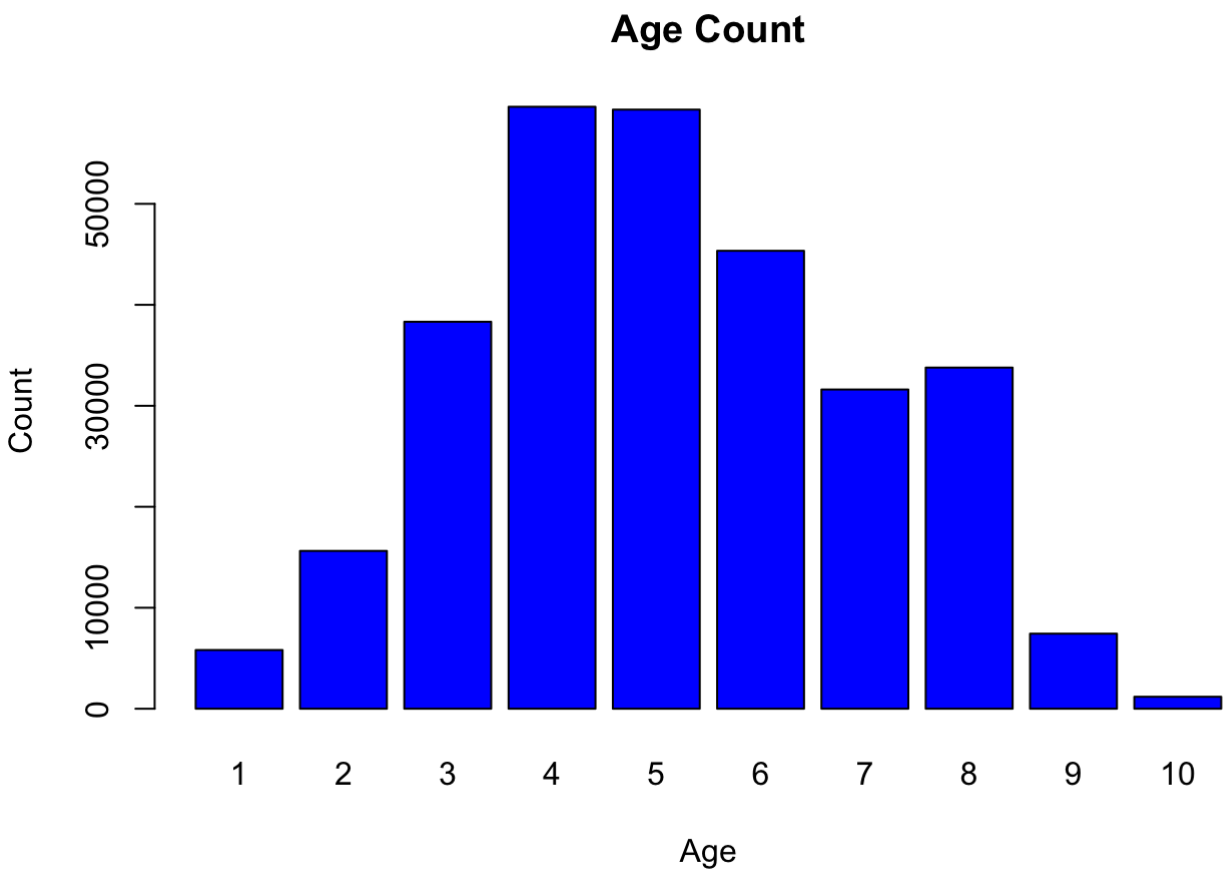
```
##      2.5%     97.5%
## -1.662512  2.476522
```

```
preprocessed_health = preprocessed_health[preprocessed_health$Admission_Deposit>=-1.6625
& preprocessed_health$Admission_Deposit <=2.4765, ]

head(preprocessed_health)
```

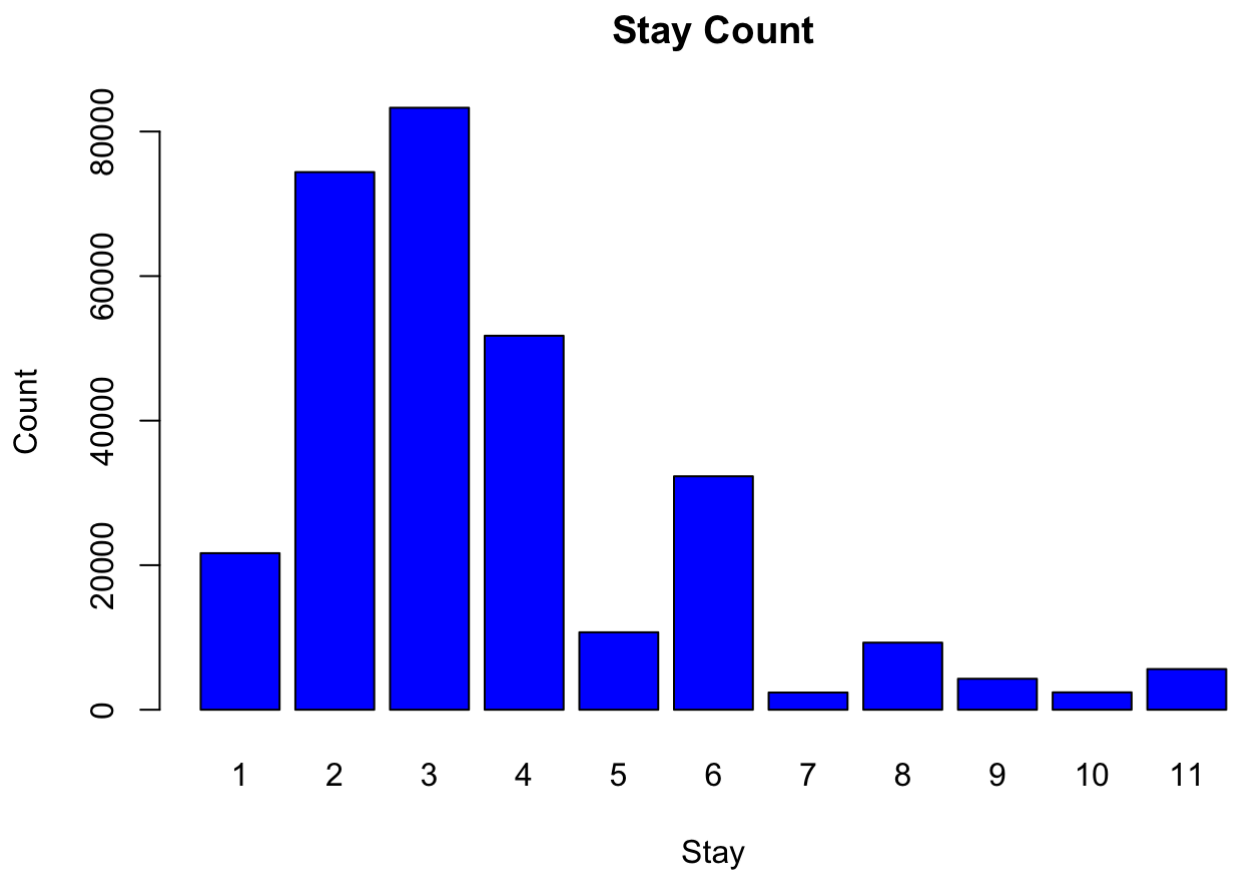
```
##   Hospital_code Hospital_type_code City_Code_Hospital
## 1             8                  1                  3
## 2             2                  1                  5
## 3            10                  2                  1
## 4            26                  3                  2
## 5            26                  3                  2
## 6            23                  4                  6
##   Available.Extra.Rooms.in.Hospital Department Ward_Type Ward_Facility_Code
## 1                        -0.1681782          1         1                  1
## 2                        -1.0243985          1         2                  1
## 3                        -1.0243985          2         2                  2
## 4                        -1.0243985          1         1                  3
## 5                        -1.0243985          1         2                  3
## 6                        -1.0243985          2         2                  1
##   Type.of.Admission Severity.of.Illness Visitors.with.Patient Age
## 1        -1.1386940                   1            -0.7270336   6
## 2         0.3125875                   1            -0.7270336   6
## 3         0.3125875                   1            -0.7270336   6
## 4         0.3125875                   1            -0.7270336   6
## 5         0.3125875                   1            -0.7270336   6
## 6         0.3125875                   1            -0.7270336   6
##   Admission_Deposit Stay
## 1        0.02679585    1
## 2        0.98698533    5
## 3       -0.12602433    4
## 4        2.20034077    5
## 5        0.62242633    5
## 6       -0.39852297    2
```

Count Values for Stay

```
# Age Frequency
barplot(table(preprocessed_health$Age),
  main="Age Count",
  xlab="Age",
  ylab="Count",
  col="blue"
)
```




```
# Stay Frequency
barplot(table(preprocessed_health$Stay),
  main="Stay Count",
  xlab="Stay",
  ylab="Count",
  col="blue"
)
```




```
table(preprocessed_health$Stay)
```

```
##
##     1     2     3     4     5     6     7     8     9    10    11
## 21661 74387 83290 51748 10714 32300  2381  9283  4285  2406  5625
```

Remove NZV values

```
# Now let's remove NZV values
dim(preprocessed_health)
```

```
## [1] 298080     13
```

```
health_nzv = nearZeroVar(subset(preprocessed_health, select=-Stay))
health_nzv
```

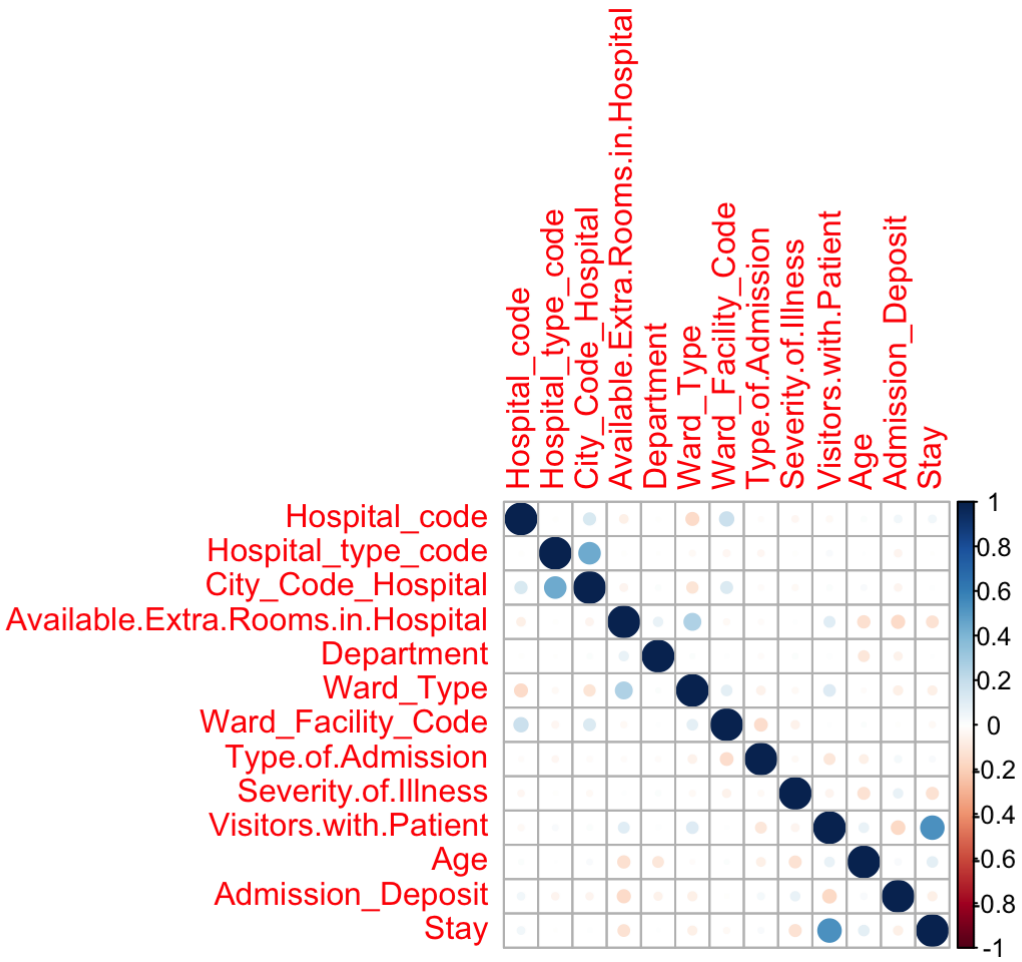
```
## integer(0)
```

Plot Correlation

```
library(corrplot)
```

```
## corrplot 0.88 loaded
```

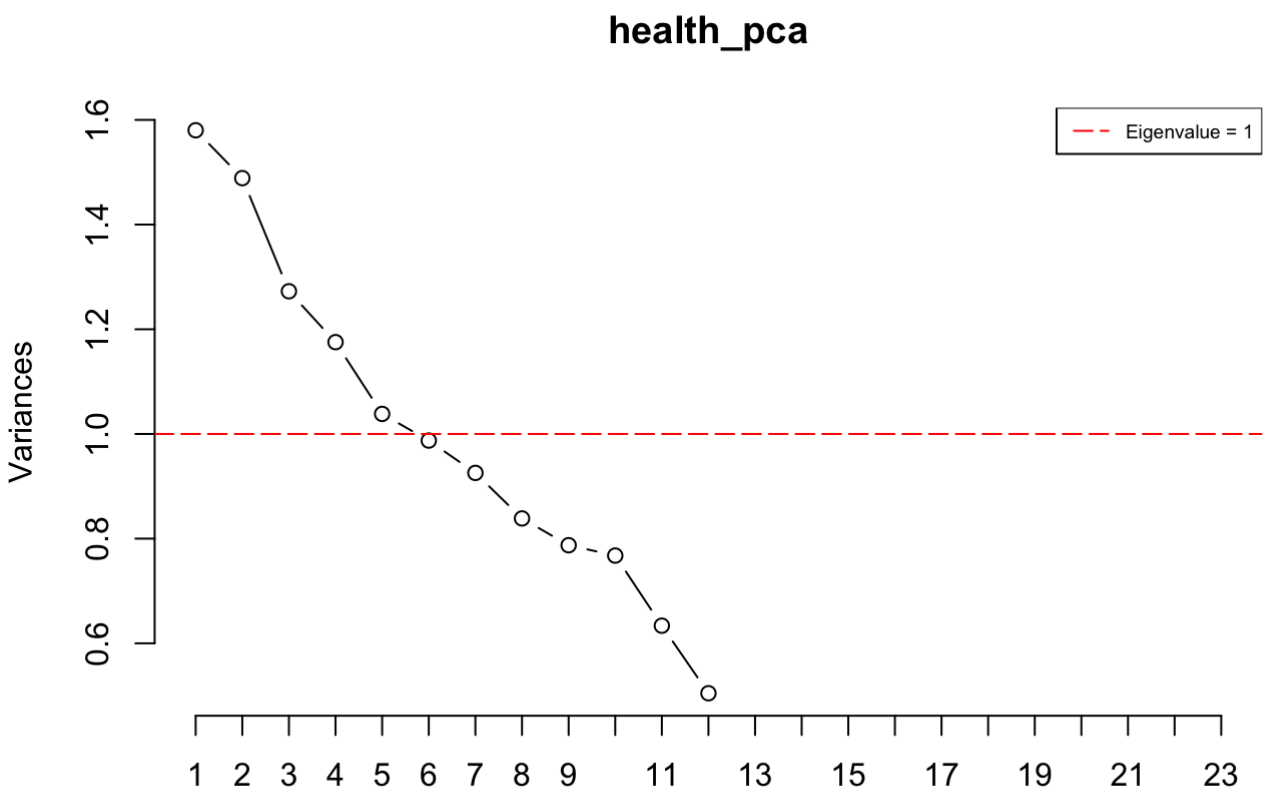
```
# First let's see any correlation among features
corrplot(cor(preprocessed_health))
```



From the plot, we can see that there is one column strongly correlated with Stay predictor which is Visitors.with.Patient. Also other features show slight correlation between each other although they are weak.

Apply PCA

```
health_pca = prcomp(subset(preprocessed_health, select=-Stay), center=TRUE, scale=TRUE)
processed_health = predict(health_pca, preprocessed_health)

screeplot(health_pca, type='l', npcs=23)
abline(h = 1, col="red", lty=5)
legend("topright", legend=c("Eigenvalue = 1"),
       col=c("red"), lty=5, cex=0.6)
```



Data Split

```
dim(preprocessed_health)
```

```
## [1] 298080     13
```

```
X = subset(preprocessed_health, select=-Stay)
y = preprocessed_health$Stay
y = factor(y, levels=c(1,2,3,4,5,6,7,8,9,10,11))

train_idx = createDataPartition(y, p=.75, list=FALSE)
train_X = X[train_idx, ]
train_y = y[train_idx]

test_X = X[-train_idx, ]
test_y = y[-train_idx]

dim(train_X)
```

```
## [1] 223564     12
```

```
dim(test_X)
```

```
## [1] 74516    12
```

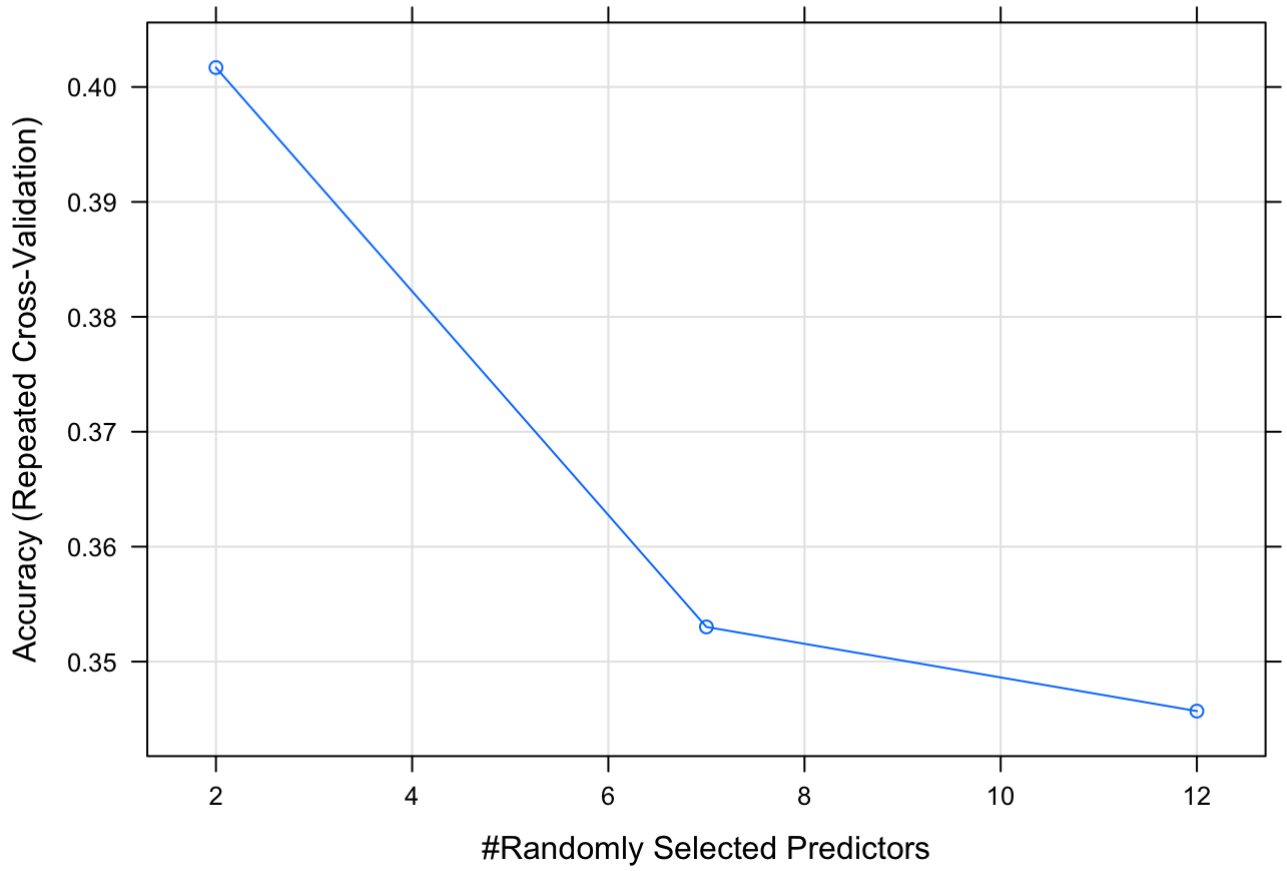
```
head(train_X, 5)
```

```
##   Hospital_code Hospital_type_code City_Code_Hospital
## 1             8                  1                  3
## 2             2                  1                  5
## 3            10                  2                  1
## 4            26                  3                  2
## 5            26                  3                  2
##   Available.Extra.Rooms.in.Hospital Department Ward_Type Ward_Facility_Code
## 1                        -0.1681782          1         1                  1
## 2                        -1.0243985          1         2                  1
## 3                        -1.0243985          2         2                  2
## 4                        -1.0243985          1         1                  3
## 5                        -1.0243985          1         2                  3
##   Type.of.Admission Severity.of.Illness Visitors.with.Patient Age
## 1        -1.1386940                   1            -0.7270336   6
## 2         0.3125875                   1            -0.7270336   6
## 3         0.3125875                   1            -0.7270336   6
## 4         0.3125875                   1            -0.7270336   6
## 5         0.3125875                   1            -0.7270336   6
##   Admission_Deposit
## 1        0.02679585
## 2        0.98698533
## 3       -0.12602433
## 4        2.20034077
## 5        0.62242633
```

Model Implementation

Random Forest

```
set.seed(123)

trControl = trainControl(method='repeatedcv', repeats=1)

rfc = train(x=train_X,
            y=train_y,
            method='rf',
            ntree=100,
            trControl=trControl)

plot(rfc)
```

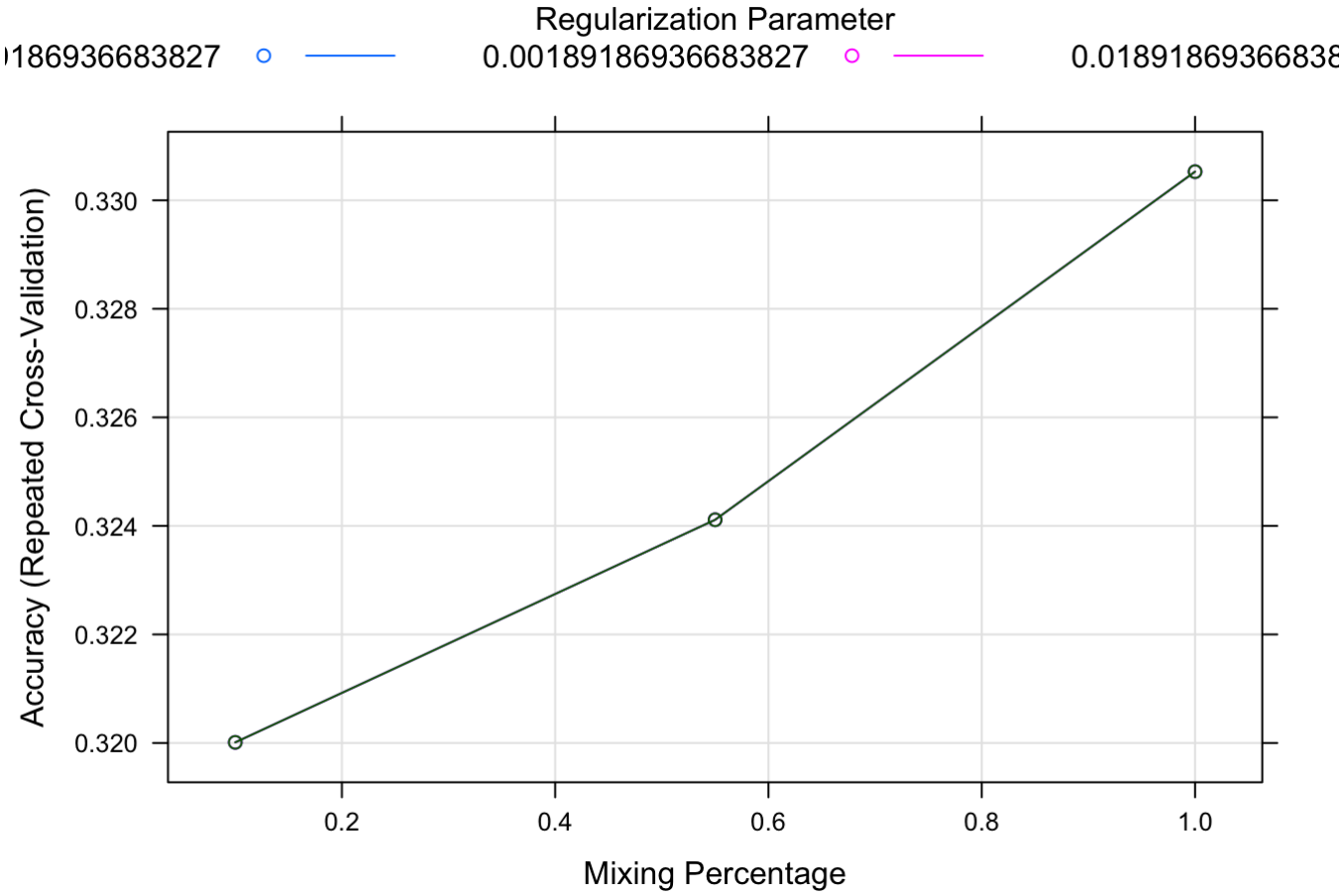



```
rfc
```

```
## Random Forest
##
## 223564 samples
##     12 predictor
##     11 classes: '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 201206, 201209, 201209, 201207, 201208, 201209, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.4016881  0.2209656
##    7    0.3530175  0.1791318
##   12    0.3456952  0.1719272
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Elastic Net

```
set.seed(123)

glmnet = train(x=train_X,
           y=train_y,
           method='glmnet',
#           tuneGrid=expand.grid(.alpha=seq(0,1,length=20), .lambda=10^seq(2,-2,length=1
00)),
           maxit=1000,
           trControl=trControl)

plot(glmnet)
```




```
head(glmnet$results[order(-glmnet$results$Accuracy),], 5)
```

```
##   alpha       lambda  Accuracy      Kappa  AccuracySD     KappaSD
## 7  1.00 0.0001891869 0.3305273 0.10308927 0.003114584 0.004474795
## 8  1.00 0.0018918694 0.3305273 0.10308927 0.003114584 0.004474795
## 9  1.00 0.0189186937 0.3305273 0.10308927 0.003114584 0.004474795
## 4  0.55 0.0001891869 0.3241130 0.09075387 0.005704473 0.009380298
## 5  0.55 0.0018918694 0.3241130 0.09075387 0.005704473 0.009380298
```

```
library(e1071)
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
set.seed(123)


nnet = train(x=train_X,
             y=train_y,
             method='nnet',
             trace=FALSE,
             maxit=1000)

plot(nnet)
```