Predictive Modeling for determining patients' length of stay

Department of Information Systems, San Diego State University

Dr. Aaron C. Elkins

Course ID: MIS 720

Course Name: Electronic Business And Big Data Analytics

From

Akshaya Viswanathan (823720372)

Max Gueniau (828607891)

Najah Izquierdo (129718339)

Naveen Reddy Sama (828541383)

Tobias Kleinhansl (828616315)

Dec 15, 2022

Table Of Contents

Executive Summary

The following report provides a predictive model to predict a given patient's length of stay based on different healthcare parameters. The relevance of optimizing resources at hospitals saw new heights during the covid pandemic as hospitals were faced with shortages of hospital beds. It was a severe problem and a huge ethical dilemma as it forced the hospitals to be selective between which patients they handle and when which in the worst case could be a matter of life or death. The lessons from covid prove that there is an urgent need for healthcare professionals to learn from each other on how to efficiently allocate resources to limit the risk of operating with scarce resources if a similar incident were to happen.

This report addresses this problem by building and testing various classification models such as Logistic Regression, Naive Bayes, K-Nearest Neighbor, and Decision Drees. The best performer was Logistic Regression, which scored the highest accuracy rate with 79.7%.

Consequently, the KNN model is the outcome of this paper. It can be used as a supportive tool to empower decision-makers to better plan and optimize resources at a given hospital. The model's predictions can be generalized across multiple hospitals and are fairly robust as it is based on a dataset with more than 300k unique instances from a variety of hospitals across multiple cities. Yet, it should be noted that the model input is not better than the input. While some inputs such as the age of the patient, and the number of visitors for the patient are easy to determine and base a prediction on. Other inputs such as the severity of the illness, may be more difficult to assess

properly, or subject to the examiner's bias or subjective opinions, which may be wrong. Finally, given the ethical concerns about the predictions and potential consequences, the model is not strong enough to be used autonomously, rather the predictions should be vetted by a human being and used to help augment decision-making.

Data Preparation

In the Data Analytics Lifecycle, Data Preparation is a crucial step and heavily influences the quality of the results of the project. The main focus is to preprocess the data in a way that it becomes a sufficient basis for the Model Building. Multiple steps may be performed to clean, normalize and transform the data, which is also known as “Data Conditioning” (EMC Educational Service 2015, p. 57-58). Moreover, the goal is to develop a comprehensive understanding of the data and gain as many insights as possible.

After reading in the dataset and loading the necessary packages, a first look was taken at the data. First of the row count was determined by using the nrow() command, returning a total of 318438 rows. In addition to displaying the first rows of the dataset, followed by opening a complete view of the data, the summary() function enabled us to get a statistical overview. To ensure compatibility with the models the data types of the individual columns were displayed using the str() function and characters were changed to factors using the as.factor() command. The columns “case_id” and “patientid” were dropped since they are unique identifiers of data entries and therefore hold now value for predicting the stay. Furthermore “Hospital_region_code” was dropped since it caused errors in the modeling part of the project. A

constraint for the project was to use datasets with a maximum of 100,000 rows in order to ensure that computational limitations would not restrict the project's progress. Therefore the dataset was reduced by randomly dropping rows using the sample() command with the instruction to keep only 100,000 rows. To guarantee the reproducibility of results, a seed was set.

The objective of this project was to perform a binary classification in order to predict whether a hospital patient will stay longer than 30 days or not. However, the target variable “stay” has multiple values and therefore has to be transformed into binary. This was done by doing the following operations.

- “0-10” was converted to “0”
- “11-20” was converted to “0”
- “21-30” was converted to “0”
- “31-40” was converted to “1”
- “41-50” was converted to “1”
- “51-60” was converted to “1”
- “61-70” was converted to “1”
- “71-80” was converted to “1”
- “81-90” was converted to “1”
- “91-100” was converted to “1”
- “More than 100 days” was converted to “1”

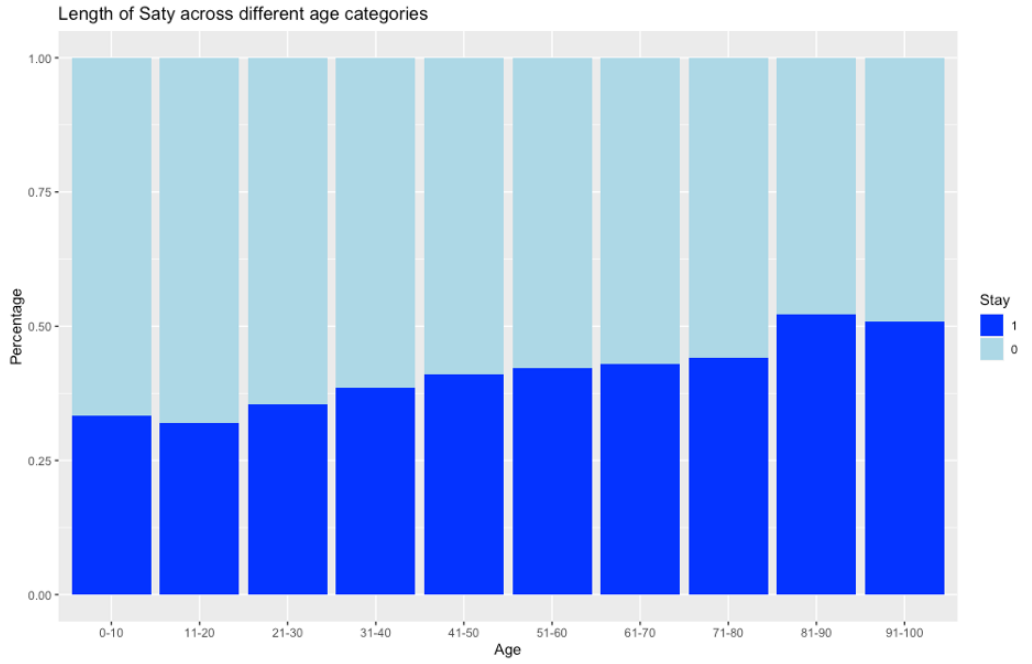
Dealing with skewed data is another challenge of the Data Preparation step of the Data Analytics Lifecycle. To do so, the non-categorical predictors “Available.Extra.Rooms.in.Hospital”, “Visitors.with.Patient” and “Admission_Deposit” were examined by using the summary() function and by displaying a histogram in order to get visual clues of skewness. The data of

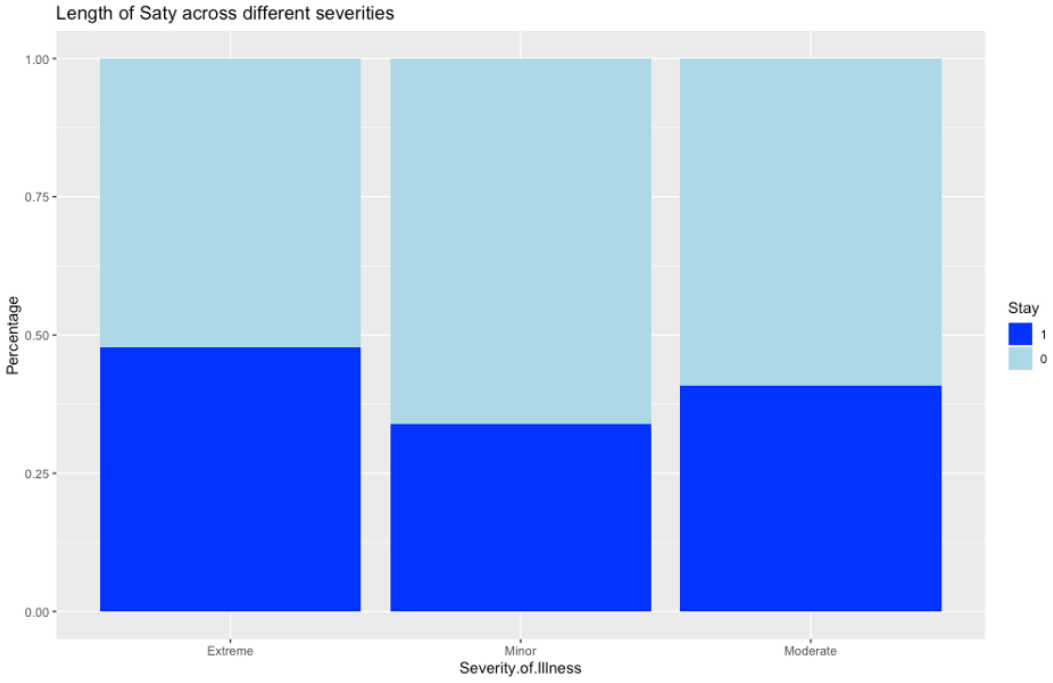
“Available.Extra.Rooms.in.Hospital”, was skewed to the right, so a cutoff of values above eight was put in place. A similar situation was found for “Visitors.with.Patient”, so a cutoff at 11 was found to be reasonable. For “Admission_Deposit” the skewness was less dramatic, however, there was still a need to deal with it. Therefore a cutoff at 9000 was done to reduce the skewness. For each predictor, a statistical summary and a histogram were invoked to evaluate the changes made to the data and to ensure that the skewness was reduced to an acceptable level without impairing the data too much. By using the nrow() function it was identified that the row count was reduced from 100,000 to 98,894. This means that 1106 rows were dropped, which equals roughly 1 percent of the data.

Using categorical predictors can lead to problems in the modeling part of the project since some models can not handle categorical input values. For certain models like the logistic regression, the categorical predictors had to be transformed into binary. This was done by using one-hot encoding. This effectively splits categorical predictors into multiple binary columns. The number of rows was thereby significantly increased, since the transformation turns a predictor with x different values, into x-1 binary columns. The dummyVars() function from the caret package was used to do so.

Hereafter, an excerpt of the preprocessed data can be seen. The one-hot encoding is not shown here, since it only applies to certain models.

Some plots to understand the data, about how the stay is distributed along the classes of the columns





Model Planning

To solve a classification problem there are multiple different techniques. By testing and deploying different kinds of parametric and non-parametric models our research will get a good grasp of the data. Where some of the models suffer, other models may thrive, making it interesting to see which model will provide the best and most accurate predictions. Specifically, this research seeks to evaluate and compare the strengths and weaknesses of 1) Logistic Regression (LR), 2) Decision Trees, 3) Naïve Bayes (NB), and 4) K-Nearest Neighbor (KNN) on our dataset. Each of these methods will be elaborated on in the following section.

Logistic regression

LR is widely adopted for solving classification problems when the target variable is discrete. It provides a probability of a certain outcome e.g., the probability that patient x, would have to stay 0-10 days or 11-20 days. A decision threshold must be determined such as e.g., 0.3 probability, in order for it to provide a final prediction.

The strength of LR is that it provides insights into which variables affect the stay length and therefore can create some additional insights into hospital operations. For instance, it would be interesting if the length of the stay was affected by the hospital id, as it would mean that there could be potential areas of improvement. Moreover, we worry that some features may correlate with each other e.g., # visitors and age, as more visitors could be expected to decrease with elderly people. In addition, the dataset has both integers and characters, which is something LR is good at handling.

The weakness is that it can be rather cynical to use numbers to try to optimize stays, as some hospitals may tend to keep patients for too short periods to discover the real diagnosis – which in the worst case may be a matter of life and death. Moreover, it may be that some of the inputs in the model are irrelevant, which is something LR can have trouble controlling. Finally, with 11 potential outcomes, and a set probabilistic threshold it may be very difficult for the model to come up with any meaningful predictions, as opposed to a prediction model with binary outcomes.

Naïve Bayes

Similarly, to LR, NB is a probability-based classification. Yet, it has some other strengths and weaknesses, which is why it is an interesting model to test and compare with the other models. For instance, LR predicts based on a naïve assumption that features are conditionally independent of each other. Although we hypothesize that this is not the case, it is still something that could be plausible until proven otherwise.

Another area where NB contributes to the paper's research design is its strengths in dealing with irrelevant inputs, which is something we suspect might be relevant to our study. In addition, the model leverages the fact that our target variable has 11 discrete outcomes, which we consider to be a rather large number of levels for a categorical variable.

Yet, a weakness is that the model similar to LR is sensitive to multicollinearity and has some trouble with estimating probabilities.

K-nearest neighbor

KNN uses a given input value to make a class prediction based on proximity to other input values in the training set, also known as neighbors. Specifically, KNN makes its prediction based on a majority vote of the nearest neighbor's length of stay. The number of nearest neighbors is defined by the number K. The higher the K, the higher the accuracy typically is, yet also requires more computational power. One of the main differences of KNN compared to LR, is that it is a non-parametric model and thus has no assumption concerning the underlying data.

The strengths of KNN are that it thrives when n is large, which we with 318,438 observations consider our data to be. Yet, one of KNNs issues is that it suffers from the curse of dimensionality, which our data is likely to exhibit with both large p (17) and large n, which can result in slow and inaccurate predictions. Moreover, the interpretability of the model may be a bit more difficult compared to parametric models such as LR. Another weakness of KNN is that it only provides the label of its prediction and not the level of confidence.

Decision trees

Like LR, decision trees provide probability scores of different outcomes and make its prediction based on the class that scored the highest. It outputs a tree that explains the decision flow, where each leaf returns a probability score.

The strengths of decision trees are that it is very flexible in terms of data types and is able to handle variables with non-linear effects on the outcome variable. In addition, it is robust against multicollinearity and can provide insights into significant variables. Yet, it also exhibits some shortcomings. For one, it is prone to overfitting, especially when it is dependent on many variables. Finally, it can be very difficult to design the decision rules.

Model Building and Evaluation

In order to build the planned classification models, additional adjustments to the data preparation phase were required. Data preprocessing provided a subdataset of 100,000 observations and 16 variables to meet the data criteria. To perform the modeling process, our dependent variable was converted to a binary response: 1, representing any patient with a stay of over 30 days, and 0 representing patients with a stay that was less than 30 days. As stated in the previous section, we decided to apply Logistic Regression, Naive Bayes, KNN, and a Decision Tree. We then performed the models and selected the most efficient based on the accuracy score.

To determine the effectiveness and performance of the models, we utilized the Caret, class, and e1071 libraries. The first step was to split the data using the split function, in which we split at 75% and then split that into training and testing datasets. In order to check the accuracy of the split, we checked the distribution of the response variable from the original dataset and split datasets. These all matched up at 59.31% for less than a 30-day stay and 40.69% for a stay longer than 30 days. The data created with these functions were primarily used for Naive Bayes, KNN, and the Decision Tree models. In order to run our Logistic Regression model we utilized Caret's dummyVars function which will apply one hot encoding to the data, which allows text data to be presented as numerical data. We also removed the null values in both training and testing data.

To calculate the accuracies of each model, we used the roc function and created plots for each model for both training and testing data, which we then combined.

Training and Testing Each Model

Logistic Regression:

Given we are dealing with a binary classification problem, an additional response variable was created using the as.numeric() function to utilize 1 and 0s. We used the dummy technique to perform one hot encoding on the data because logistic regression cannot handle character variables. This resulted in the creation of several columns in our data based on the classes present in each character variable column. The character variable columns were then removed from the data.

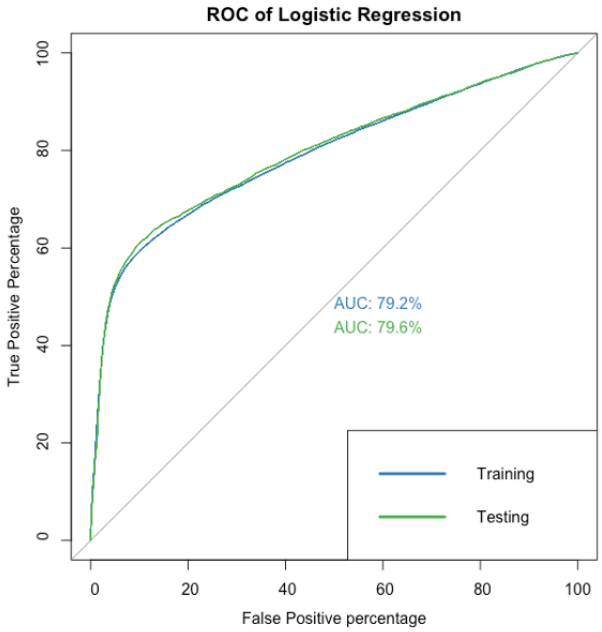
The **glm** function utilizes train_dataset$Stay as its dependent variable and the dummy_train as our data, this then produces a probability score for each predictor. In the summary of the model many of the variables are significant. Then we observed the variable importance ranking of logistic regression model, the results are as follows,

Variable	**Importance**	Age.41.50	3.30631686
Visitors.with.Patient	104.220601	Age.21.30	3.23900733
Available.Extra.Rooms.in.Hospital	25.9226581	Ward_Facility_Code.A	3.06086314
Ward_Facility_Code.C	17.7645713	Hospital_type_code.f	2.86597486
Severity.of.Illness.Minor	14.2848531	Age.71.80	2.72569608
Hospital_code	10.5877609	Age.31.40	2.28511171
Type.of.Admission.Emergency	10.2373087	Hospital_type_code.c	1.78214131
Severity.of.Illness.Extreme	9.61075294	Bed.Grade	1.72830553
Ward_Facility_Code.D	8.975311	patientid	1.52897369
Type.of.Admission.Trauma	7.63432499	Admission_Deposit	1.25738764
Department.gynecology	7.29221141	City_Code_Patient	1.14281103
City_Code_Hospital	6.54587723	Hospital_type_code.d	1.07441098
Ward_Facility_Code.B	6.01345063	Ward_Type.S	0.69294935
Age.11.20	5.74982535	Age.81.90	0.58086446
Age.0.10	5.71322165	Ward_Type.T	0.55471597
Hospital_type_code.b	5.44213569	Hospital_type_code.a	0.53281628
Age.51.60	4.69019693	Ward_Type.P	0.47478257
Age.61.70	4.66247941	Ward_Type.R	0.39314677
Department.surgery	4.56472413	Department.anesthesia	0.22872184
Department.radiotherapy	3.75823393	Hospital_type_code.e	0.17562172
Ward_Facility_Code.E	3.72257657	Ward_Type.Q	0.08943505

The model is then tested on 24377 testing data samples and below is the confusion matrix

Accuracy (0.78)	Reference		
Prediction		0	1
	0	13313	4019
	1	1270	5775

According to the findings, visitors with the patient is the best predictor, while ward type Q is the least predictor for the logistic regression model.



ROC of logistic regression

We predicted the response of both training and testing data using the model and compared it to the true values. Then, as shown in the figure, we plotted a ROC curve and calculated the accuracy. We achieved **78% training accuracy** and **78% testing accuracy**.

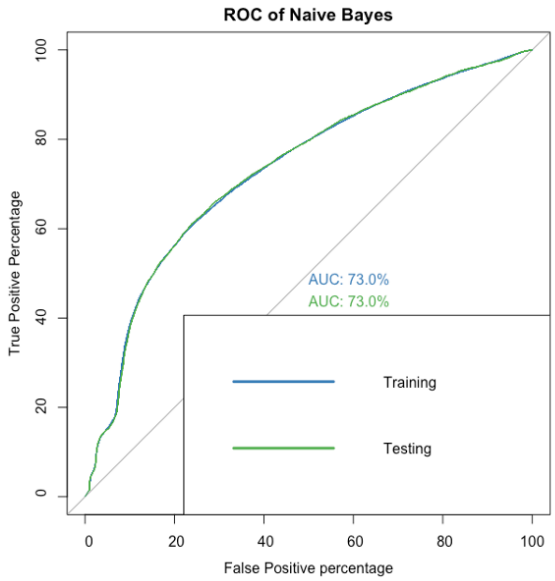
Naive Bayes:

Naive Bayes model was set up using the test_data and Stay dependent variable, this required little to no alterations within the dataset. This model used 73080 samples, 15 predictors and 2 classes, being '0' and '1'. Then the model is tested on 24377 samples. The following is the model confusion matrix on testing data:

Accuracy (0.7)	Reference		
Prediction		0	1
	0	13313	4019
	1	1270	5775

Confusion Matrix results

We predicted the response of both training and testing data using the model and compared it to the true values. Then, as shown in the figure, we plotted a ROC curve and calculated the accuracy. We achieved **73.0%** training accuracy and **73.0%** testing accuracy.



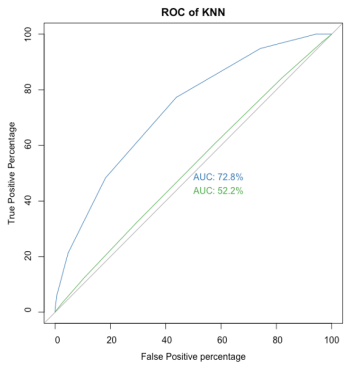
K-nearest neighbor:

Since we are analyzing a larger dataset with a large n and p, a KNN model is very long to process and run. Using Caret's train function, we implemented a control variable, tuneLength of 2, and used the train_data. Given this model functions best with a larger n, utilizing the training data with ~73,000 observations would maximize the efficiency of the model. Our K-nearest Neighbors model ran 73,924 samples, 16 predictors, and 2 classes, giving us an accuracy score of 68% with a value of k=7. Unfortunately, the Kappa score for the model was extremely low. Summary and confusion matrix of the model when fitted on testing data is as follows:

Accuracy (0.78)	Reference		
Prediction		0	1
	0	10302	6641
	1	4264	3134

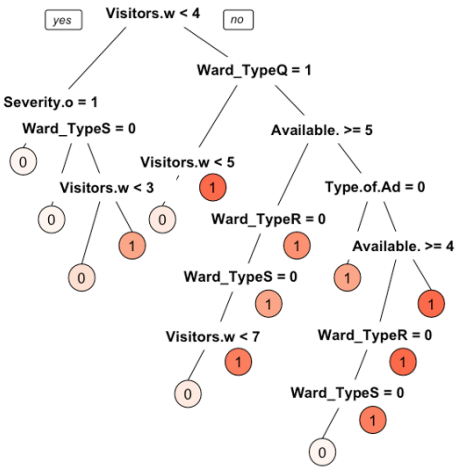
KNN Confusion Matrix

We predicted the response of both training and testing data using the model and compared it to the true values. Then, as shown in the figure, we plotted a ROC curve and calculated the accuracy. We achieved **68.8%** training accuracy and **68%** testing accuracy.

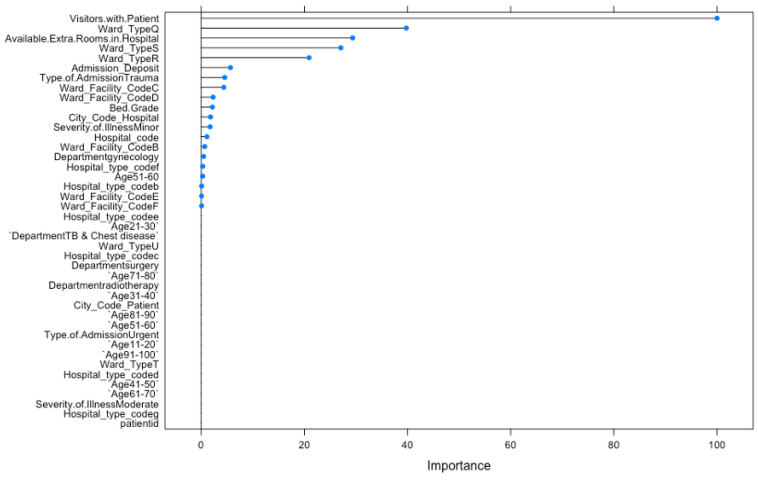


Decision trees:

The Decision tree seemed to perform well. It utilizes the test_dataset which has 73080 samples, 15 predictors, and 2 classes. Training this model was relatively easy and ran very quickly and efficiently, returning an accuracy score of 78.89% and a kappa score of 53% with cp value of 0.00054. Running the original dataset through the model resulted in similar accuracy and kappa scores. We plotted the tree as given below:



The variable importance ranking of random forest model is given below:

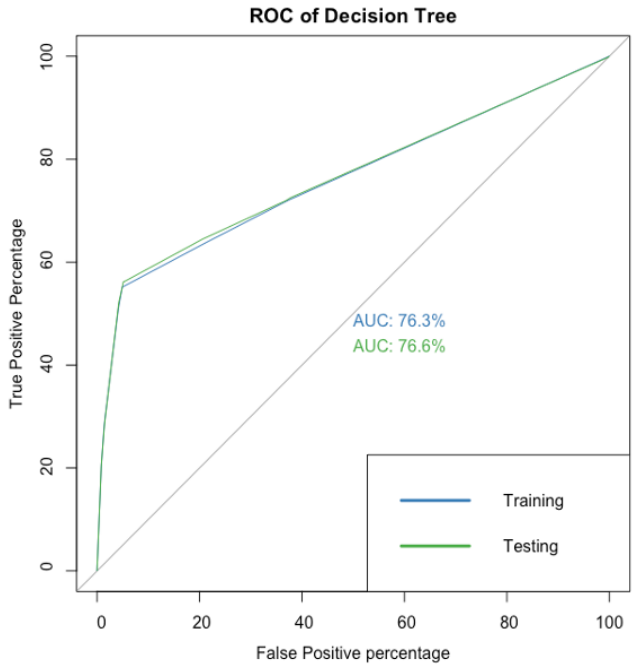


Then the model is tested on test_data and achieved an accuracy of 79%

Accuracy (0.79)	Reference		
Prediction		0	1
	0	13897	4364
	1	686	5430

Decision Tree Confusion Matrix

We predicted the response of both training and testing data using the model and compared it to the true values. Then, as shown in the figure, we plotted a ROC curve and calculated the accuracy. We achieved **79.9%** training accuracy and **79%** testing accuracy.



Results and Conclusion

Ultimately, the decision tree and logistic regression outperformed the preceding models exponentially. The overall accuracies along with the area under the curves are as given below table:

Model	Accuracy	AUC (Training)	AUC(Testing)
Logistic Regression	78	79.2	79.6
Naive Bayes	70	73.2	73.2
K-Nearest Neighbor (k=7)	68	72.8	52.2
Decision Tree	79	76.3	76.6

Accuracy Results

But if observing the area under the curve logistic regression has a more area under the curve when compared to the decision tree, but as the difference of AUC is only around 2% for both models we can consider the **Decision Tree** as the best model out of all four models.

Bibliography

EMC Educational Services (2015). *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data.* 1st Edition. John Wiley & Sons.

Appendix

Rcode

GitHub Repository: https://github.com/najahizquierdo/MIS720Project-

Data Preprocessing:

health_care_analytics.R

```
#San Diego State University
#MIS 720 - E-Business and Big Data Infrastructures
#Term Project
#Predicting Length of stay in hospital
#Team: Aztec Data Analytics
#Max Gueniau, Akshaya Viswanathan, Najah Izquierdo, Naveen Reddy Sama, Tobias Kleinhansl

#---------------------------------

#1 Packages
```

```
library(ggplot2)
library(caret)
library(ROCR)
library(pROC)
require(caret)
library(rpart.plot)

#2 Data Preparation

#Import an first look
data_raw <- read.csv('/Users/naveenreddysama/Desktop/Naveen/BDA/Semester1/MIS
720/Project/train.csv')
nrow(data_raw)
head(data_raw)
View(data_raw)
summary(data_raw)



#Data types
str(data_raw)

# changing characters to factors
data_raw$Hospital_type_code <- as.factor(data_raw$Hospital_type_code)
```

```
data_raw$Department <- as.factor(data_raw$Department)
data_raw$Ward_Type <- as.factor(data_raw$Ward_Type)
data_raw$Ward_Facility_Code <- as.factor(data_raw$Ward_Facility_Code)
data_raw$Type.of.Admission <- as.factor(data_raw$Type.of.Admission)
data_raw$Severity.of.Illness <- as.factor(data_raw$Severity.of.Illness)
data_raw$Age <- as.factor(data_raw$Age)

#predictors with data type character should be transformed using one hot encoding at a later
point


#Unique values for target variable
table(data_raw$Stay)

#Unique values for categorical predictors

summary(data_raw)
table(data_raw$Hospital_code)
table(data_raw$Hospital_type_code)
table(data_raw$City_Code_Hospital)
table(data_raw$Hospital_region_code)
table(data_raw$Department)
table(data_raw$Ward_Type)
```

```
table(data_raw$Ward_Facility_Code)

table(data_raw$Bed.Grade)

table(data_raw$City_Code_Patient)

table(data_raw$Type.of.Admission)

table(data_raw$Severity.of.Illness)

table(data_raw$Age)



#Drop row case_id since its just there for numbering the rows and holds no value for predicting
the stay

data_red1 <- subset(data_raw, select = -c(case_id))

data_red2 <- subset(data_red1, select = -c(patientid))

data_red2 <- subset(data_red1, select = -c(Hospital_region_code))

head(data_red2)



#A constraint for the project was to use datasets with a maximum of 100,000 rows.

#Therefore 218438 rows are dropped randomly

#For reproducability, a seed is set

set.seed(12345)

data_red3 <- data_red2[sample(1:nrow(data_red1), 100000),]

nrow(data_red3)

#Check the first lines of data_red2 vs data_red3
```

```
head(data_red2)

head(data_red3)



#The hospital wants to be able to predict if a patient stays longer than 30 days

#The target variable 'Stay' has therefore to be transformed into binary, where 0 = less or equal

than 30 days and 1= more than 30 days

data_red3$Stay <- gsub('0-10', 0, data_red3$Stay)

data_red3$Stay <- gsub('11-20', 0, data_red3$Stay)

data_red3$Stay <- gsub('21-30', 0, data_red3$Stay)

data_red3$Stay <- gsub('31-40', 1, data_red3$Stay)

data_red3$Stay <- gsub('41-50', 1, data_red3$Stay)

data_red3$Stay <- gsub('51-60', 1, data_red3$Stay)

data_red3$Stay <- gsub('61-70', 1, data_red3$Stay)

data_red3$Stay <- gsub('71-80', 1, data_red3$Stay)

data_red3$Stay <- gsub('81-90', 1, data_red3$Stay)

data_red3$Stay <- gsub('91-100', 1, data_red3$Stay)

data_red3$Stay <- gsub('More than 100 Days', 1, data_red3$Stay)

View(data_red3)



#Check if non-categorical data (Available Extra Rooms in Hospital, Visitors with Patient,

Admission_Deposit) is skewed and transform accordingly
```

```
summary(data_red3$Available.Extra.Rooms.in.Hospital)

hist(data_red3$Available.Extra.Rooms.in.Hospital)

#Skewed, a cutoff at 8 seems reasonable

data_skew1 <- subset(data_red3, data_red3$Available.Extra.Rooms.in.Hospital<=8)

View(data_skew1)

hist(data_skew1$Available.Extra.Rooms.in.Hospital)

summary(data_skew1$Available.Extra.Rooms.in.Hospital)


summary(data_skew1$Visitors.with.Patient)

hist(data_skew1$Visitors.with.Patient)

#Skewed, a cutoff at 11 seems reasonable

data_skew2 <- subset(data_skew1, data_skew1$Visitors.with.Patient<=11)

hist(data_skew2$Visitors.with.Patient)

summary(data_skew2$Visitors.with.Patient)


summary(data_skew2$Admission_Deposit)

hist(data_skew2$Admission_Deposit)

#A little skewed, a cutoff at 9000 seems reasonable

data_skew3 <- subset(data_skew2, data_skew2$Admission_Deposit<=9000)

hist(data_skew3$Admission_Deposit)

summary(data_skew3$Admission_Deposit)


#Check how many rows were dropped
```

```
nrow(data_red3)

nrow(data_skew3)

#1106 rows were dropped, so about 1%


# Age with respect to length of stay Bar Plot


ggplot(data_skew3, aes(x=Age, fill=Stay))+
  scale_fill_manual(breaks = c("1", "0"),
                    values=c("blue", "light blue"))+
  geom_bar(position="fill")+
  ggtitle("Length of Saty across different age categories")+
  ylab('Percentage')

# Hospital_type_code with respect to length of stay Bar Plot


ggplot(data_skew3, aes(x=Hospital_type_code, fill=Stay))+
  scale_fill_manual(breaks = c("1", "0"),
                    values=c("blue", "light blue"))+
  geom_bar(position="fill")+
  ggtitle("Length of Saty across different Hospital_type_codes")+
  ylab('Percentage')


# Department with respect to length of stay Bar Plot
```

```
ggplot(data_skew3, aes(x=Department, fill=Stay))+
  scale_fill_manual(breaks = c("1", "0"),
                    values=c("blue", "light blue"))+
  geom_bar(position="fill")+
  ggtitle("Length of Saty across different Departments")+
  ylab('Percentage')

# Severity.of.Illness with respect to length of stay Bar Plot

ggplot(data_skew3, aes(x=Severity.of.Illness, fill=Stay))+
  scale_fill_manual(breaks = c("1", "0"),
                    values=c("blue", "light blue"))+
  geom_bar(position="fill")+
  ggtitle("Length of Saty across different severities")+
  ylab('Percentage')


# splitting the data into 75/25

sample <- sample.split(data_skew3$Stay, SplitRatio = 0.75)
train_data  <- subset(data_skew3, sample == TRUE)
test_data <- subset(data_skew3, sample == FALSE)
```

```
#check distribution of test train split
prop.table(table(train_data$Stay)) * 100
prop.table(table(test_data$Stay)) * 100
prop.table(table(data_skew3$Stay)) * 100

#Preprocessing for Logistic Reg

# splitting the data to X and Y

training_dataX <- train_data[,names(train_data) != "Stay"]

dummy <- dummyVars( ~ ., data=training_dataX)

# Applying one hot encoding to data for logistic regression
dummy_train <- data.frame(predict(dummy, newdata = train_data))

# splitting the data to test and train for logistic regression
dummy_test <- data.frame(predict(dummy, newdata = test_data))
dummy_test <- na.omit(dummy_test)
dummy_train <- na.omit(dummy_train)
```

```
# eliminating the null values
train_data <- na.omit(train_data)
test_data <- na.omit(test_data)

# converting the response to a factor of 0 and 1
train_data$Stay <- as.factor(as.numeric(train_data$Stay))
test_data$Stay <- as.factor(test_data$Stay)
train_data$Stay <- as.factor(train_data$Stay)
set.seed(400)

# Logistic Regression


logistic_model <- glm(train_data$Stay ~ .,
                      data = dummy_train,
                      family = "binomial")

summary(logistic_model)

# observing the best variables in the model
(varImp(logistic_model))

par(pty= "s")
```

```
# confusion matrix and accuracy

# training

p_train_log <- predict(logistic_model, newdata = dummy_train, type = "response")
# converting probability to labels
p_train_log.cat <- ifelse(p_train_log > 0.5,"1","0")
p_train_log.cat <- as.factor(p_train_log.cat)
cm_log_train<-confusionMatrix(data=p_train_log.cat,reference=train_data$Stay)
cm_log_train

Accuracy_log_test<-round(cm_log_train$overall[1],2)
Accuracy_log_test

# testing

p_test_log <- predict(logistic_model, newdata = dummy_test, type = "response")

p_test_log.cat <- ifelse(p_test_log > 0.5,"1","0")
p_test_log.cat<-as.factor(p_test_log.cat)

cm_log_test<-confusionMatrix(data=p_test_log.cat,reference=test_data$Stay)
```

```
cm_log_test

Accuracy_log_test<-round(cm_log_test$overall[1],2)
Accuracy_log_test

#roc

# taining
roc(train_data$Stay, p_train_log, plot = TRUE, legacy.axes = TRUE, percent = TRUE,
    xlab = "False Positive percentage", ylab = "True Positive Percentage",
    main= "ROC of Logistic Regression",
    col="#377eb8", lwd =1, print.auc = TRUE)



# testing
plot.roc(test_data$Stay, p_test_log, percent = TRUE,lwd = 1,
         col="#4daf4a", print.auc.y = 45, print.auc = TRUE, add = TRUE)
legend("bottomright",c("Training", "Testing"), col = c("#377eb8","#4daf4a"), lwd =3)



#Decision Trees

ctrl_dtree <- trainControl(method="repeatedcv",repeats = 3)
```

```
dtree_model <- train(Stay ~.,
                     data = train_data,
                     method = "rpart",
                     parms = list(split = "information"),
                     trControl=ctrl_dtree,
                     tuneLength = 10,
                     na.action = na.pass)

dtree_model # model summary

# plotting the tree
prp(dtree_model$finalModel, box.palette = "Reds", tweak = 1.2)
plot(dtree_model, col="red")



# observing the best performing variables in the model
plot(varImp(dtree_model))

# confusion matrix and accuracy

# training data
p_train_dt <- predict(dtree_model, newdata = train_data, type = "raw")
p_train_dt.cat <- as.factor(p_train_dt)
```

```
cm_dt_train<-confusionMatrix(data=p_train_dt.cat,reference=train_data$Stay)
cm_dt_train

Accuracy_dt_test<-round(cm_dt_train$overall[1],2)
Accuracy_dt_test

# testing data
p_test_dt <- predict(dtree_model, newdata = test_data, type = "raw")
p_test_dt.cat<-as.factor(p_test_dt)

cm_dt_test<-confusionMatrix(data=p_test_dt.cat,reference=test_data$Stay)
cm_dt_test

Accuracy_dt_test<-round(cm_dt_test$overall[1],2)
Accuracy_dt_test

# roc

# training data

p_train_dt_roc <- predict(dtree_model, train_data, type = "prob")
```

```
roc(train_data$Stay, p_train_dt_roc[ ,2], plot = TRUE, legacy.axes = TRUE, percent = TRUE,
    xlab = "False Positive percentage", ylab = "True Positive Percentage",
    main= "ROC of Decision Tree",
    col="#377eb8", lwd =1, print.auc = TRUE)

# testing data

p_test_dt_roc <- predict(dtree_model, test_data, type = "prob")

plot.roc(test_data$Stay, p_test_dt_roc[ ,2], percent = TRUE,lwd = 1,
        col="#4daf4a", print.auc.y = 45, print.auc = TRUE, add = TRUE)
legend("bottomright",c("Training", "Testing"), col = c("#377eb8","#4daf4a"), lwd =3)




# KNN

ctrl_knn <- trainControl(method = "cv",
                         summaryFunction = defaultSummary,
                         number = 5)
set.seed(2)
knn <- train(Stay ~ .,
             data = train_data,
```

```
        method = "knn",
        trControl=ctrl_knn,
        metric = "Accuracy",
        tuneLength = 2,
        na.action = na.pass
)


print(knn)

# confusion matrix and accuracy

# training data
p_train_knn <- predict(knn, newdata = train_data, type = "raw")
p_train_knn.cat <- as.factor(p_train_knn)
cm_knn_train<-confusionMatrix(data=p_train_knn.cat,reference=train_data$Stay)
cm_knn_train

Accuracy_knn_test<-round(cm_knn_train$overall[1],2)
Accuracy_knn_test

# testing data
p_test_knn <- predict(knn, newdata = test_data, type = "raw")
```

```
p_test_knn.cat<-as.factor(p_test_knn)

cm_knn_test<-confusionMatrix(data=p_test_knn.cat,reference=test_data$Stay)

cm_knn_test


Accuracy_knn_test<-round(cm_knn_test$overall[1],2)

Accuracy_knn_test


# roc


# training data


p_train_knn_roc <- predict(knn, train_data, type = "prob")


roc(train_data$Stay, p_train_knn_roc[ ,2], plot = TRUE, legacy.axes = TRUE, percent = TRUE,

    xlab = "False Positive percentage", ylab = "True Positive Percentage",

    main= "ROC of KNN",

    col="#377eb8", lwd =1, print.auc = TRUE)


# testing data


p_test_knn_roc <- predict(knn, test_data, type = "prob")


plot.roc(test_data$Stay, p_test_knn_roc[ ,2], percent = TRUE,lwd = 1,
```

```
        col="#4daf4a", print.auc.y = 45, print.auc = TRUE, add = TRUE)
legend("bottomright",c("Training", "Testing"), col = c("#377eb8","#4daf4a"), lwd =3)
# naive bayes

naive_bayes <- train(Stay ~ .,
                     data = train_data,
                     method = "naive_bayes",
                     usepoisson = TRUE,
                     na.action = na.pass)

# Viewing the model
naive_bayes

# confusion matrix and accuracy

# training data
p_train_nb <- predict(naive_bayes, newdata = train_data, type = "raw")
p_train_nb.cat <- as.factor(p_train_nb)
cm_nb_train<-confusionMatrix(data=p_train_nb.cat,reference=train_data$Stay)
cm_nb_train

Accuracy_nb_test<-round(cm_nb_train$overall[1],2)
Accuracy_nb_test
```

```
# testing data
p_test_nb <- predict(naive_bayes, newdata = test_data, type = "raw")
p_test_nb.cat<-as.factor(p_test_nb)

cm_nb_test<-confusionMatrix(data=p_test_nb.cat,reference=test_data$Stay)
cm_nb_test

Accuracy_nb_test<-round(cm_nb_test$overall[1],2)
Accuracy_nb_test

# roc

# training data

p_train_nb_roc <- predict(naive_bayes, train_data, type = "prob")

roc(train_data$Stay, p_train_nb_roc[ ,2], plot = TRUE, legacy.axes = TRUE, percent = TRUE,
    xlab = "False Positive percentage", ylab = "True Positive Percentage",
    main= "ROC of Naive Bayes",
    col="#377eb8", lwd =1, print.auc = TRUE)

# testing data
```

```
p_test_nb_roc <- predict(naive_bayes, test_data, type = "prob")


plot.roc(test_data$Stay, p_test_nb_roc[ ,2], percent = TRUE,lwd = 1,
        col="#4daf4a", print.auc.y = 45, print.auc = TRUE, add = TRUE)
legend("bottomright",c("Training", "Testing"), col = c("#377eb8","#4daf4a"), lwd =3)


# out of all models decision tree has the highest accuracy of 79%
```