The Sinking of the *Titanic* (1912) – Evaluating Survival Using Classification Methods

Joel Cabrera

Rutgers University – New Brunswick

jc2135@scarletmail.rutgers.edu

Statistical Models and Computing (16:954:567)

Master of Science in Data Science (MSDS) Program

Professor Wang

May 09, 2021

I. Introduction

The sinking of the RMS *Titanic* (1912) was one of the most infamous shipwrecks in human history. This event occurred on April 15, 1912, where the *Titanic* sank after colliding with an iceberg. Of the 2224 passengers and crew that were aboard the ship, about 1502 of them died. While such a disaster occurred over a century ago, the tragedy is still discussed and studied today. Indeed, some people still wonder how exactly it sank, despite its complex infrastructure **[2]**. Others still tell interesting stories about the *Titanic* and its passengers, expanding on its legacy and the impact it has had on ship building and wreckages **[3]**. But, as a result of the disaster, there are those who fear of embarking on ships, as they believe they may be casualties themselves **[1]**. Thus, from these stories, we can derive two problems: How many people survived the sinking of the *Titanic*? And could we extend our finding such that we lessen people's fears of voyaging on cruise ships?

The goal of this project is to develop a machine learning model that makes the most accurate predictions possible for who survived the sinking of the *Titanic*, given certain factors (e.g. age, sex, number of relatives aboard, etc.). Should we accomplish this goal, we can determine who was most likely to have survived back. Such a finding would not only resolve the debate of who survived the *Titanic*'s wreckage, but also can lessen people's fears of embarking on cruise ships like the *Titanic*.

II. Data & Description

The dataset that will be used for our data analysis is the "Titanic – Machine Learning from Disaster" dataset from the Kaggle website **[5]**. For the sake of this paper, it will be referred to as simply the "Titanic" dataset from now on. The data is originally formatted as three separate CSV files called "test", "train", and "gender_submission", respectively. The "gender_submission" file contains a set of predictions that assume all and only female survive. This is a subset of the "test"

dataset; its split is intended to help those who want to have “gender” as the response variable. For our project, we combined all three of these files into one dataset. We did this because we wanted to have an easier time with cleaning and wrangling the data in R.

Having combined the data into one, we examine it in its entirety in R first. We find that it contains 1309 observations (rows) and 12 variables (columns). Of these 12 variables, 4 are continuous and 8 are categorical. The names of these variables are found below.

```
> names(data)
 [1] "PassengerId" "Survived"    "Pclass"      "Name"        "Sex"         "Age"
 [7] "SibSp"       "Parch"       "Ticket"      "Fare"        "Cabin"       "Embarked"
```

However, it should be noted that we are no using all 12 of these variables. Four of these variables are categorical with unique IDs as their values. These variables are “PassengerId”, “Name”, “Ticket”, and “Cabin”. They also have missing values. Replacing them with the means of each variable would be implausible. Thus, we drop them from our upcoming data analyses.

Given the information of all 12 variables, we decide to use the remaining 8 variables – with 7 as predictors and 1, “Survived”, as the response (binary) variable for our statistical models. Examining these variables even further, we find that some of their values seem arcane. While these values can be understood by the data dictionary provided by the Kaggle website below, some of them are missing and are qualitative/multi-level. Both of these problems, however, are accounted for in the upcoming sections.

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

III. Quantitative Methodologies

Given the properties of the "Titanic" dataset, we will conduct some data cleaning, exploratory data analysis (EDA), and data splitting to prepare the data for model building and statistical analysis. After this, we will then analyze the data using five statistical models to process and analyze said dataset. These five statistical methods are: logistic regression, classification trees, generalized additive models (GAMs), random forests, and k-nearest neighbors (KNN). We use these models to analyze the dataset because they are ideal for classification analysis – being that "Survived", the response variable, is binary. The performance of all five models will be evaluated via cross-validation. This will be done in order to choose the optimal tuning parameters for their respective models. The model that will be considered the "best" among them will be based on determining which model's prediction accuracy for "Survived" is the highest. We will also obtain the AUC values and ROC plots for each model, in order to see how well each one predicts.

Data Cleaning

The data currently contain 263 NAs and 1 NA in the "age" and "fare" variables, respectively. We replace the missing values with the means of each variable. We do this because dropping the missing values would also result in about 20% of the data being dropped. Such a drop

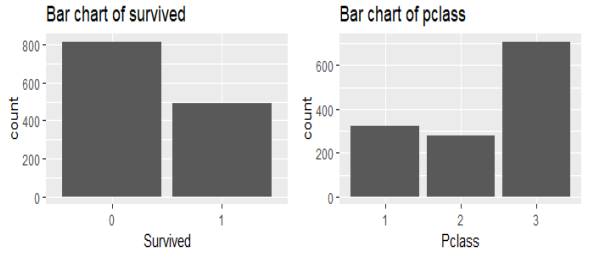
would spell disaster for our statistical models and their results. Upon doing this, the variables no longer have any NAs. However, checking for missing values in the variables again in R, we find that "Embarked" has 2 NAs. As "Embarked" is a qualitative unordered variable, and that it is senseless to replace them with the mean of the variable, we drop its NAs from the data. The data is now free of any missing values – now having dimensions of 1307 x 12.

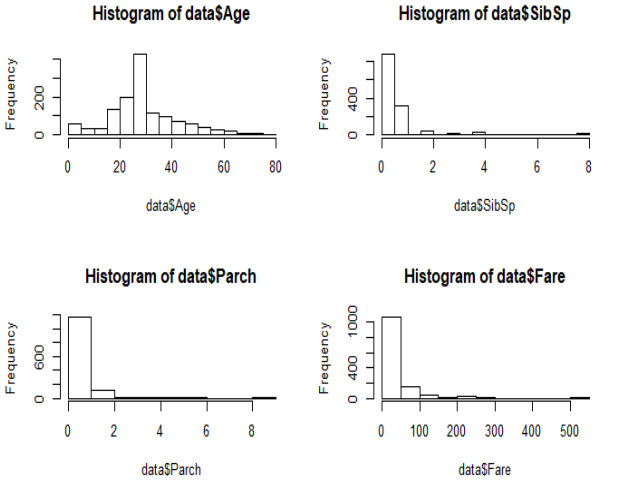
Some of the categorical variables in the dataset still need cleaning, as they are either qualitative or are unordered/ordered. Thus, in R, we do the following: Convert the values of "Embarked" to numeric (1 = "Cherbourg", 2 = "Queenstown", 3 = "Southampton"), convert "gender" values to numeric (1 = "male", 0 = "female"), defined "Pclass" as an ordered variable, defined "Embarked" as an unordered variable (locations are not ordered, after all). After doing this and checking the data once more in R, we are assured that the data is now ready for data analysis.

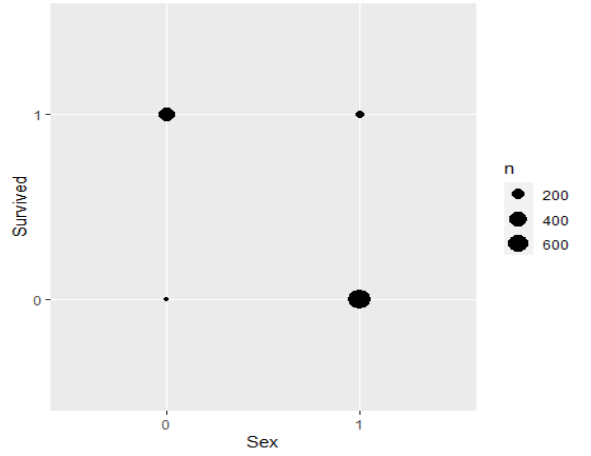
EDA

In terms of checking and exploring the data, we focused on three kinds of figures. The first were bar charts, which depicted the number of levels for each categorical variable. They indicated that the data is balanced, especially for the "Survived" response variable. Such an observation is good news for model building, as we can assured that our results are accurate and valid. The second were histograms, which depicted the frequencies of each continuous variable. They indicated that the variables are right-skewed. While one may say this may be a problem for our models, we say that this is not too much of a concern. This is because the data is cleaned and balanced, as demonstrated earlier. The third were scatterplots, which depicted the relationships between age and sex against survival. The first plot indicated that males were more likely to not survive the *Titanic* wreckage than females, while the second plot indicated an even distribution of the ages of

passengers/crewmates for survival. We keep these observations in mind when building our models and interpreting their results. Below are the three kinds of figures discussed previously.



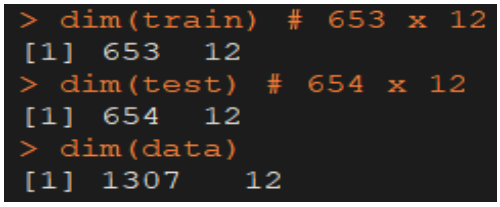


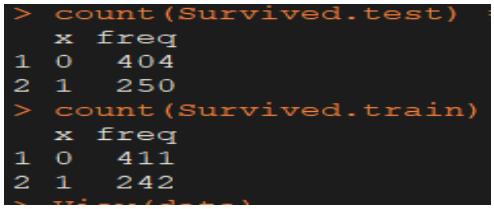


Data Splitting

After cleaning the data and conducting some EDA, we did a 50-50 split of the data into training and test sets, respectively. Both sets have dimensions of 653 x 12 and 653 x 12, also respectively. While one may argue that such a split is not optimal, given the sample size of the data is 1307, we believe that this split is actually suited for this data. The data contain features of the passengers and crew of the *Titanic*. While the cruise ship was quite large, it was only large

enough to hold around 2000 people. Furthermore, due to its wreckage, not all people or their features were documented, since they were unrecovered from the sea. Hence, we have the (cleaned) 1307 observations for the dataset. Since we want to predict how many people survived the *Titanic* sinking - given certain features, and that about half the people actually survived the disaster (data could be larger if missing people were found), a 50-50 data split is therefore reasonable. Also, the 50-50 split allows for balanced values in the response variable (shown below), further solidifying more accurate results for our models.





IV. Results & Discussion

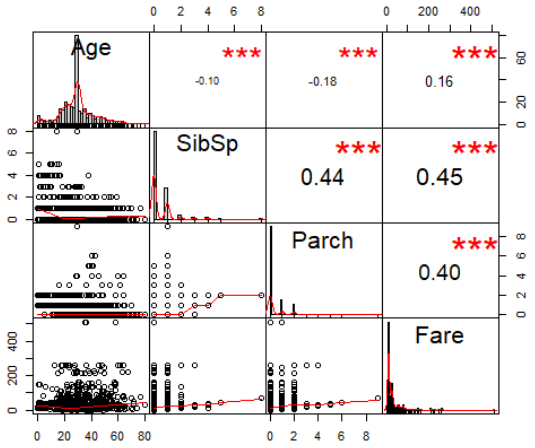
Logistic Regression

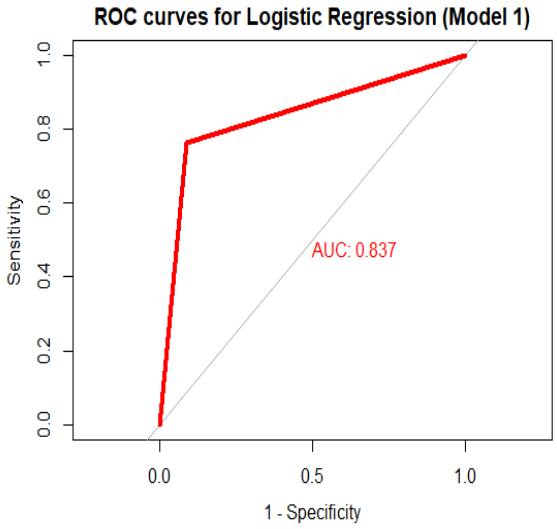
Since "Survived" is a binary response variable, we can fit logistic regression to the data. However, before doing so, we should check the linearity among the continuous variables. This is because the model assumes that all continuous variables are linearly independent. Producing and checking the scatter matrix of these variables in R, we can see that there is no linearity among these variables. This assumption is satisfied and thus we can safely fit logistic regression to this data.

We construct three logistic regression models to the data. We do this for a couple of reasons. The first is that we can use them to estimate the probabilities for "Survived". The second is that we choose the best model among the three, such that it has the highest prediction accuracy. We use the training data to estimate the models' regression coefficients and evaluate said models via cross-validation.

Each candidate logit model contains a number of predictors. The first model contains all 7 predictors. After fitting the model, based on their p-values, some predictors were dropped. This resulted in the second model, which contains 5 predictors. Again, after fitting this model, some of its predictors were dropped due to their statistically significant p-values. This resulted in the third and final model, which contains only 3 predictors.

Given our three logit models – each progressively have a reduced number of predictors, we then evaluate their performance on the test set via cross-validation. Doing this in R resulted in the following: Models 1, 2, and 3 correctly predicted about 85.47%, 62.39%, and 64.07% of observations in the test set, respectively. Their leave-one-out cross-validation (LOOCV) errors are 0.1117, 0.2045, and 0.2153, also respectively. Since the first model has the highest prediction accuracy and the lowest CV error, it can be said that it has the best predictive performance, and thus is the best model among the three. Its AUC value is 0.837, as shown below.

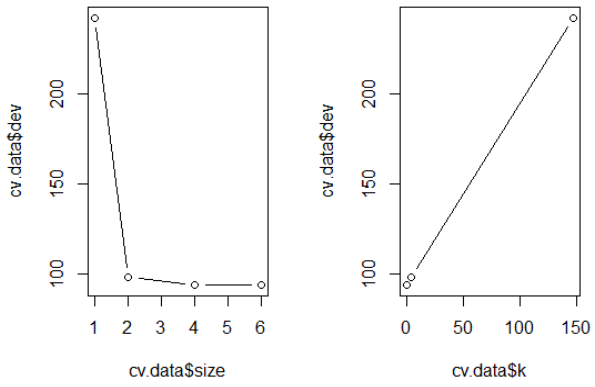


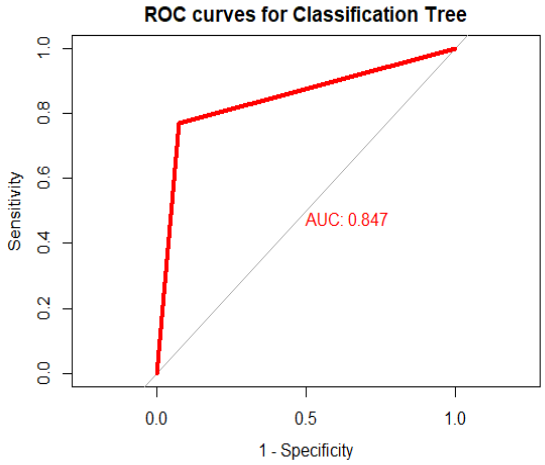


```
> table(glm.pred1, Survived.test)
         Survived.test
glm.pred1   0   1
        0 368  59
        1  36 191
> 100*mean(glm.pred1 == Survived.test)
[1] 85.47401
```

Classification Tree

We now fit a classification tree to this data in R. In doing so, we find that 100% of the test observations for "Survived" are correctly classified. This is a very dubious result, as this would mean that this model can perfectly predict all of the observations in the test set. Thus, to obtain a more accurate classification rate, we prune the tree for the optimal number of nodes. Through cross-validation, we found that 4 nodes are the optimal number (as shown by the cv.data$size graph). Pruning the classification tree based on 4 nodes, we find that it correctly classified 100% of the observations in the test set. Thus, pruning the tree based on the optimal number of nodes yields no improvement of the original classification rate of 100%. Again, like with the original tree, this is a very questionable result, as our 4th model, a random forest, was designed to improve the accuracy of classification trees. We also ensured that the R code for the classification tree is correct, as it was confirmed by multiple source (e.g. classmates, professors, and the internet). Thus, we keep this result under extreme suspicion, but mindful of comparing it to the results of the upcoming models. The AUC for the classification tree, however, is 0.847, which is a much more reasonable result.



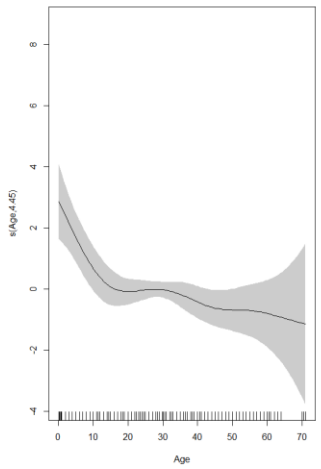


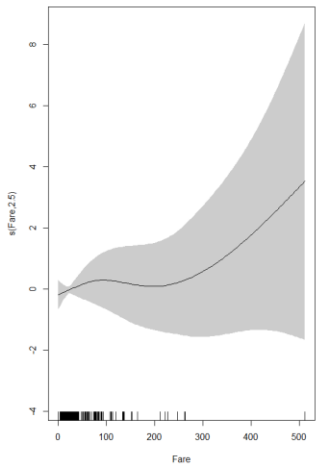

```
> table(tree.preds, Survived.test)
           Survived.test
tree.preds   0   1
         0 374  58
         1  30 192
> 100*round((374 + 192)/566, 4) #
[1] 100
```

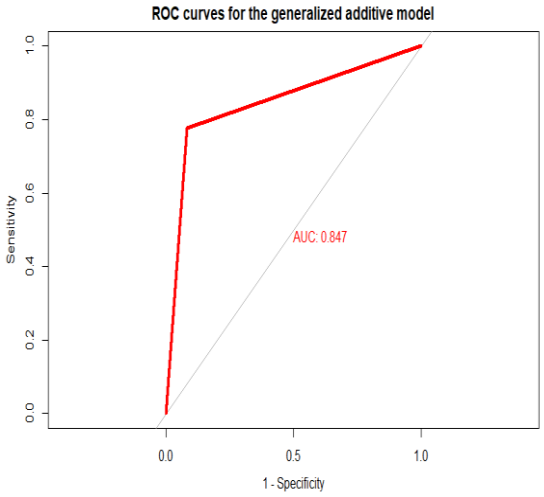
GAM

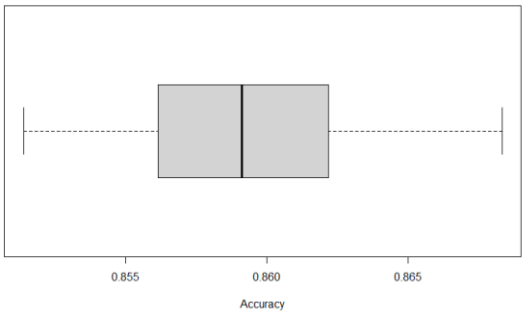
Given that we were able to fit a logistic regression model to the data, it would make sense to also fit a logistic GAM to the data. It would also make sense fit multiple GAMs and choose the best one among them based on its predictive accuracy and CV error, like what we have done for the three candidate logit models. However, attempting to fit multiple GAMs in R results in errors, so we have decided to fit only one GAM. Despite this shortcoming, we believe that it is still fit for comparison against other models, as its results are comparable to those of the other models.

Fitting a GAM with a degree of freedom (DF) of 1 for the smoothing terms (e.g. “age” and “fare”, since they are continuous variables), we obtains its test accuracy of 86.39%. Evaluating this model via 5-fold CV 50 times in R, we find that it results in a mean predictive accuracy of 86.91%. Thus, it can be said that the GAM can predict approximately 86.9% of the observations in the test set. The AUC value of this model is 0.847.









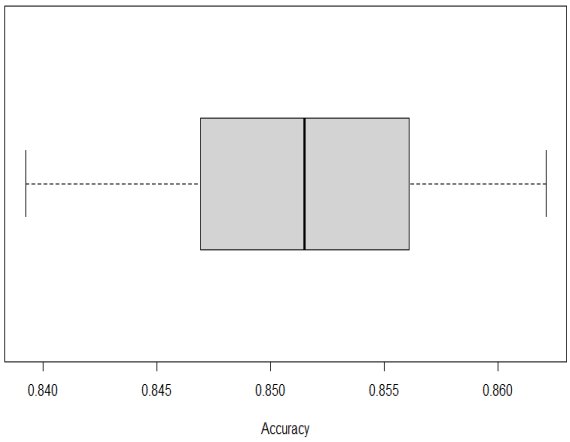

```
> table(gam_pred, Survived.test) # evaluatin
        Survived.test
gam_pred   0   1
   FALSE 371  56
   TRUE   33 194
> mean(gam_pred == as.numeric(Survived.test)
[1] 0.8639144
```

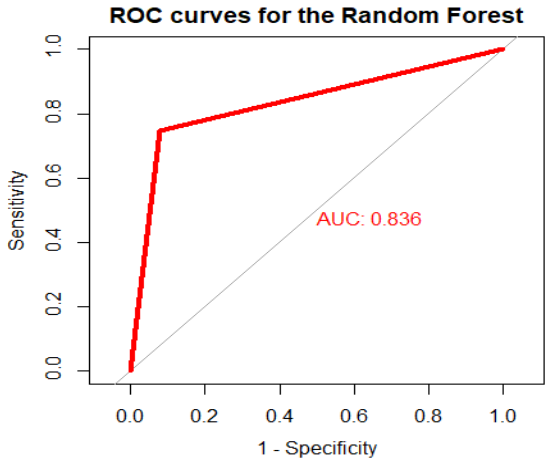
Random Forest

We now fit a random forest to the “Titanic” training data. Our random forest contains 500 trees and randomly samples 3 variables as candidates at each split of the tree. These parameters are implemented so that we can ensure the most randomization of the model possible, allowing for more accurate results.

Fitting this in R, we find that the model correctly predicts 85.63% of the observations in the test set. Evaluating this model via 5-fold CV 50 times in R, we find that it results in a mean predictive accuracy of 85.22%. Thus, it can be said that the GAM can predict approximately 85.2% of the observations in the test set. The AUC value of this model is 0.836. Also, an interesting note to mention is that the “Pclass” variable is considered the most important predictor in the random

forest, as shown in the bottom-left figure. This would suggest that one's ticket class (e.g. high, middle, and low classes) had the biggest impact of surviving the *Titanic* sinking in the model.





	0	1	MeanDecreaseAccuracy	MeanDecreaseGini
Pclass	0.013285218	0.0610927329	0.031153507	16.09316
Sex	0.164223978	0.2838360415	0.208056545	126.10142
Age	0.003414229	0.0396888186	0.016930931	47.48576
SibSp	0.012677544	0.0045629681	0.009699534	12.19347
Parch	0.003504983	0.0007254028	0.002472198	9.33236
Fare	0.016843078	0.0212237849	0.018440685	50.73510
Embarked	0.006034964	0.0208031207	0.011488860	8.01419

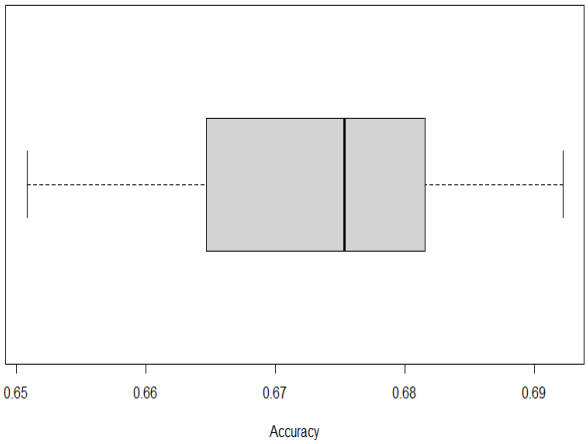
```
> table(rf_pred, Survived.test)
       Survived.test
rf_pred   0   1
      0 373  63
      1  31 187
> mean(rf_pred == Survived.test)
[1] 0.8562691
```

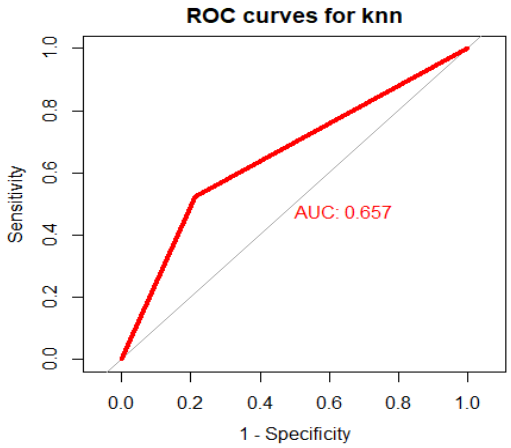
KNN

We now fit our final model, KNN, to the training data. Our KNN model considers 3 neighbors. We chose this k-value for a couple reasons. The first is that having a small value means less computational power in R, allowing us to obtain faster results. The second is that "Survived", the response variable, has 2 classes: 1 means that one survived the *Titanic* sinking, and 0 means otherwise. Because it has 2 classes, we chose an odd number of neighbors, which is a common approach employed by data scientists **[4]**.

Fitting KNN in R, we find that the model correctly predicts 68.81% of the observations in the test set. Evaluating this model via 5-fold CV 50 times in R, we find that it results in a mean

predictive accuracy of 67.25%. Thus, it can be said that KNN can predict approximately 67.3% of the observations in the test set. The AUC value of this model is 0.657.






```
> table(knn_pred, Survived.test)
        Survived.test
knn_pred   0   1
       0 319 119
       1  85 131
> mean(knn_pred == Survived.test)
[1] 0.6880734
```

V. Summary of Results

Now that we were able to obtain the best models for each statistical method, we summarize their prediction accuracy rates and AUC values in the table below. This will be useful in making our final conclusions about our models.

Model	Accuracy %	AUC
Logistic Regression	85.47	0.837
Classification Tree	100	0.847
GAM	86.9	0.847

Random Forest	85.2	0.836
KNN	67.3	0.657

VI. Conclusion

Considering all five models used to predict "Survived" using all other 7 variables, we find that the most optimal logistic regression, classification tree, GAM, random forest, and KNN models have predictive accuracy rates of 85.47%, 100%, 86.9%, 85.2%, and 67.3%, respectively. Additionally, their corresponding AUC values are 0.837, 0.847, 0.847, 0.836, and 0.657, also respectively. Given both kinds of criteria, one would believe that the classification tree has the best predictive performance at first. But, this can be called under severe question for the reason below.

Our random forest model has lower predictive performance. Random forests are intended to be an improvement over classification trees. Thus, for our purposes, the tree's results can be discarded. Aside from the tree, GAM has the best predictive performance. It has the highest prediction accuracy rate and AUC value. Thus, GAM is the best model of the five models used – being able to predict who would had survived the *Titanic* sinking accurately, given certain features. While there are other models that can used for this classification problem (e.g. boosting, XG Boosting, logistic regularized regression, etc.), and that our models can be further optimized, we hope that these ideas can be expanded further in the future (perhaps as a side project). The finding from our best model is useful for those who want to know the approximate number of deaths in the *Titanic* sinking, and for those who would want to predict their own chances of surviving a similar event.

References

[1] Editor, et al. “Cruise Fears and Why You Shouldn't Worry about Them.” *ETurboNews*, 12 Apr. 2010, eturbonews.com/30160/cruise-fears-and-why-you-shouldnt-worry-about-them/.

[2] “The Secret of How the Titanic Sank.” *U.S. News & World Report*, U.S. News & World Report, www.usnews.com/news/national/articles/2008/09/25/the-secret-of-how-the-titanic-sunk.

[3] Staff, JED REINERT | Digital. “7 Stories about the Titanic That Sank 109 Years Ago Today, Including the Story of a Local Notable Passenger.” *LancasterOnline*, 15 Apr. 2021, lancasteronline.com/features/yesteryear/history/7-stories-about-the-titanic-that-sank-109-years-ago-today-including-the-story-of/article_c47310bc-8322-11ea-8451-ff0ece775d7c.html.

[4] Subramanian, Dhilip. “A Simple Introduction to K-Nearest Neighbors Algorithm.” *Medium*, Towards Data Science, 3 Jan. 2020, towardsdatascience.com/a-simple-introduction-to-k-nearest-neighbors-algorithm-b3519ed98e.

[5] “Titanic - Machine Learning from Disaster.” *Kaggle*, www.kaggle.com/c/titanic/overview

Appendix A – R Code

```
# Joel, Lichen
# Statistical Models & Computing (01:954:567)
# Professor Wang
# April 20, 2021

### Preliminaries
library(ggplot2)
library(caret)
library(tidyverse)
library(boot)
library(broom)
library(tidytext)
library(rvest)
library(purrr)
library(gridExtra)
library(grid)
library(lattice)
library(bnpa) # check.na
library(plyr)
library(tree)
library(pROC)
library(mgcv)
library(cvTools)
library(randomForest)
library(gam)
library(gbm)
library(glmnet)
library(rpart)

### Checking & cleaning data
## loading data
data = read.csv(file = 'Titanic_Data.csv') # choose working directory first
temp_data = data # used to compare original data
head(data)
dim(data) # 1309 rows x 12 columns
names(data)
## cleaning data
# imputing missing values for Age & Fare with mean (reference: https://www.guru99.com/r-
replace-missing-values.html)
check.na(data) # missing approx. 20% of data, so cannot drop observations/rows
age_mean = mean(data$Age, na.rm = TRUE)
fare_mean = mean(data$Fare, na.rm = TRUE)
data$Age[is.na(data$Age)] = age_mean # note: don't do data$Age[data$Age == NA] =
age_mean
```

```
data$Fare[is.na(data$Fare)] = fare_mean
check.na(data) # should be no more missing values! (note: there are 2 more left...)
# dropping NA's for Embarked
levels(data$Embarked)# "" = implies missing values, but not marked with NA
data$Embarked[data$Embarked == ""] = NA # marks them as such; doing
count(data$Embarked) shows 2 NA's
check.na(data) # still 2 left, let's drop them (can't impute means, since is multinomial variable)
data = na.omit(data)
count(data$Embarked)
data$Embarked = factor(data$Embarked) # eliminates "" level
levels(data$Embarked) # no more NAs/"" level
# Converting all values to numeric
# data = map_df(data,as.numeric) %>% as.data.frame() ### not using this, since converts all
numeric values
data[, c(6, 7, 8, 10)] = map_df(data[, c(6, 7, 8, 10)], as.numeric)
# Converting nominal and ordinal categorical variables
data[, c(2, 5)] = map_df(data[, c(2, 5)], as.factor)
data$Pclass = factor(data$Pclass, ordered = TRUE) # is 3 column
# checking categorical variables (which will be used)
levels(data$Survived) # binary
levels(data$Pclass) # ordinal
levels(data$Sex) # binary
levels(data$Embarked) # multinomial
# imputing 0'& 1's for sex
levels(data$Sex)
levels(data$Sex)[1] = 0 # female = 0
levels(data$Sex)[2] = 1 # male = 1
levels(data$Sex)
# imputing 1's, 2's, & 3's for Embarked
levels(data$Embarked)
levels(data$Embarked)[1] = 1 # C = 1
levels(data$Embarked)[2] = 2 # Q = 2
levels(data$Embarked)[3] = 3 # S = 3
levels(data$Embarked)
data$Embarked = factor(data$Embarked, ordered = FALSE) # is 12 column; NEEDED to let R
recognize it as multinomial

### Descriptive Statistics (EDA)
## Summaries
dim(data)
head(data)
summary(data)
## Plots
# Bar charts (since variables are categorical)
p1 = ggplot(data, aes(x = Survived)) +
  geom_bar() + ggtitle("Bar chart of survived")
```

```
p2 = ggplot(data, aes(x = Pclass)) +
  geom_bar() + ggtitle("Bar chart of pclass")
p3 = ggplot(data, aes(x = Sex)) +
  geom_bar() + ggtitle("Bar chart of sex")
p4 = ggplot(data, aes(x = Embarked)) +
  geom_bar() + ggtitle("Bar chart of embarked")
grid.arrange(p1, p2, p3, p4, nrow = 2, ncol = 2)
# Histograms (since variables are continuous)
par(mfrow = c(2, 2))
hist(data$Age)
hist(data$SibSp)
hist(data$Parch)
hist(data$Fare)
par(mfrow = c(1, 1)) # must run, resets par()
# Plots (reference: https://r4ds.had.co.nz/exploratory-data-analysis.html)
ggplot(data = data) + # plot b/w 2 categorical variables
  geom_count(mapping = aes(x = Sex, y = Survived))
plot(data$Age, data$Survived, main = "Scatterplot of Age & Survived", # plot cont. variable
against categorical
    xlab = "Age", ylab = "Survived", pch=19)

### Model building
## splitting and check data
set.seed(1)
set = sample(1:nrow(data), nrow(data)*0.5) # 0.5 = 1/2
train = data[set, ] # training set
test = data[-set, ] # test set
Survived.train = train$Survived
Survived.test = test$Survived
dim(train) # 653 x 12
dim(test) # 654 x 12
dim(data)
count(Survived.test) # load plyr library first before running this
count(Survived.train) # seems to be a nice ratio of both 0's and 1's for both sets

## Logistic regression
# logit models (2nd & 3rd models = reduced based on most statistically significant p-values)
glm.fits1 = glm(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked, family =
binomial, data = train)
summary(glm.fits1)
glm.fits2 = glm(Survived ~ Pclass + SibSp + Parch + Fare + Embarked, family = binomial, data
= train)
summary(glm.fits2)
glm.fits3 = glm(Survived ~ SibSp + Parch + Fare, family = binomial, data = train)
summary(glm.fits3) # note that AIC values = decreasing
# predicting 1st model
```

```
nrow(test)
glm.probs1 = predict(glm.fits1, test, type = "response")
glm.pred1 = rep(0, 654) # 654 = comes from nrow(test)
glm.pred1[glm.probs1 > 0.5] = 1
table(glm.pred1, Survived.test)
100*mean(glm.pred1 == Survived.test) # gives 85.47 test accuracy
# predicting 2nd model
glm.probs2 = predict(glm.fits2, test, type = "response")
glm.pred2 = rep(0, 654) # 654 = comes from nrow(test)
glm.pred2[glm.probs2 > 0.5] = 1
table(glm.pred2, Survived.test)
100*mean(glm.pred2 == Survived.test) # gives 62.39 test accuracy
# predicting 3rd model
glm.probs3 = predict(glm.fits3, test, type = "response")
glm.pred3 = rep(0, 654) # 654 = comes from nrow(test)
glm.pred3[glm.probs3 > 0.5] = 1
table(glm.pred3, Survived.test)
100*mean(glm.pred3 == Survived.test) # gives 64.07 test accuracy
# LOOCV
cv.err1 = cv.glm(train, glm.fits1)
cv.err1$delta
cv.err2 = cv.glm(train, glm.fits2)
cv.err2$delta
cv.err3 = cv.glm(train, glm.fits3)
cv.err3$delta
# Given the fact that the 1st model has the lowest AIC value, highest
# prediction accuracy, and lowest CV error, we choose model 1
# AUC
glm.fits1_roc = roc((as.integer(Survived.test)-1),glm.pred1, plot=T,print.auc=T,
                    col="red",lwd=4,legacy.axes=T, main="ROC curves for Logistic Regression (Model
1)")
# check multicollinearity b/w continous variables
chart.Correlation(data[,c(6, 7, 8, 10)] ,method="spearman")

## Classification tree
set.seed(1)
tree.data = tree(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked, data = train)
# tree.data = R gives output corresponding to each branch of tree
tree.pred = predict(tree.data, test, type = "class")
table(tree.pred, Survived.test) # evaluating performance on test data
100*round((374 + 192)/566, 4) # 100% test observations are correctly classified
# performing CV on tree
set.seed(1)
cv.data = cv.tree(tree.data, FUN = prune.misclass) # let's see if pruning tree leads = improved
results
names(cv.data) # cv.tree = determines optimal level of tree complexity
```

```
cv.data # dev = corresponds to CV error rate; tree with 2 terminal nodes = has lowest CV error
# plot error rate as function of size & k
par(mfrow = c(1, 2))
plot(cv.data$size, cv.data$dev, type = "b")
plot(cv.data$k, cv.data$dev, type = "b")
par(mfrow = c(1, 1)) # reset par
# pruning & plotting tree of 5 nodes
prune.data = prune.misclass(tree.data, best = 4) # based on cv.data, 4 or 6 nodes is best
plot(prune.data)
text(prune.data, pretty = 0)
tree.preds = predict(prune.data, test, type = "class") # check how well pruned tree performs on
test set
table(tree.preds, Survived.test)
100*round((374 + 192)/566, 4) # 100% of test observations are correctly classified
# AUC
tree_roc = roc((as.integer(Survived.test)-1), (as.integer(tree.preds)-1), plot=T,print.auc=T,
            col="red",lwd=4,legacy.axes=T, main="ROC curves for Classification Tree")

## Generalized Additive Model (GAM)
# fit the train data using the generalized additive model
# s is the smooth term
# Pclass, Sex and Embarked are facors, we do not need to use the smooth terms
# SibSp and Parch are discrete variables, we do not have enough degrees to fit the data so we do
not use the smooth terms
# 1st GAM
set.seed(1)
gam_fit <- gam(Survived ~ Pclass + Sex + s(Age) + SibSp + Parch + s(Fare) + Embarked,
family = binomial, data = train)
summary(gam_fit)
layout(matrix(1:2, nrow = 1))
plot(gam_fit, shade = T)
# predict function returns the linear prediction
gam_pred <- 1 - 1 / (1 + exp(predict(gam_fit, test))) > 0.5
table(gam_pred, Survived.test) # evaluating performance on test data
mean(gam_pred == as.numeric(Survived.test) - 1) # gives 86.39% test accuracy
##### Failed attempt for 2nd & 3rd GAMs ######
# 2nd GAM
#gam_fit1 <- gam(Survived ~ Pclass + Sex + s(Age,2) + SibSp + Parch + s(Fare) + Embarked,
family = binomial, data = train)
#summary(gam_fit1)
#layout(matrix(1:2, nrow = 1))
#plot(gam_fit1, shade = T)
# predict function returns the linear prediction
#gam_pred1 <- 1 - 1 / (1 + exp(predict(gam_fit1, test))) > 0.5
#table(gam_pred1, Survived.test) # evaluating performance on test data
#mean(gam_pred1 == as.numeric(Survived.test) - 1) # gives 86.39% test accuracy
```

```
# 3rd GAM
#gam_fit2 <- gam(Survived ~ Pclass + Sex + s(Age) + SibSp + Parch + s(Fare, 2) + Embarked,
family = binomial, data = train)
#summary(gam_fit2)
#layout(matrix(1:2, nrow = 1))
#plot(gam_fit2, shade = T)
# predict function returns the linear prediction
#gam_pred2 <- 1 - 1 / (1 + exp(predict(gam_fit2, test))) > 0.5
#table(gam_pred2, Survived.test) # evaluating performance on test data
#mean(gam_pred2 == as.numeric(Survived.test) - 1) # gives 86.39% test accuracy
##### End of failed attempt #####
# use cross validation to evaluate the model
# we perform 50 times CV
cvK <- 5  # number of CV folds
cv_gam_res <- cv_acc <- c()
for (i in 1:50) {
  cvSets <- cvTools::cvFolds(nrow(train), cvK)  # permute all the data, into cvK folds
  cv_acc <- NA  # initialise results vector
  for (j in 1:cvK) {
    test_id <- cvSets$subsets[cvSets$which == j]
    df_test <- train[test_id, ]
    df_train <- train[-test_id, ]
    fit <- gam(Survived ~ Pclass + Sex + s(Age) + SibSp + Parch + s(Fare) + Embarked, family =
binomial, data = df_train)
    pred <- 1 - 1 / (1 + exp(predict(fit, df_test))) > 0.5
    cv_acc[j] <- mean(pred == as.numeric(df_test$Survived) - 1)
  }
  cv_gam_res <- append(cv_gam_res, mean(cv_acc))
}
par(mfrow = c(1, 1))
boxplot(cv_gam_res, horizontal = TRUE, xlab = "Accuracy")
mean(cv_gam_res)*100
# AUC
par(mfrow = c(1, 1))
gam_roc = roc((as.numeric(Survived.test)-1),as.numeric(gam_pred), plot=T,print.auc=T,
            col="red",lwd=4,legacy.axes=T, main="ROC curves for the generalized additive
model")

## Random Forest
# fit the train data using Random Forest
# ntree: Number of trees to grow
# mtry: Number of variables randomly sampled as candidates at each split
rf_fit <- randomForest(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked, data
= train,
                       ntree = 500, mtry = 3, proximity = TRUE, importance = TRUE)
# the importance of variables
```

```
# 0: the influence of variable replacement on the data classified as 0
# 1: Represents the impact of variable replacement on data classified as 0
# Mean decrease accuracy: the decrease of accuracy after variable replacement
# Mean decrease Gini: the decrease of Gini coefficient after variable replacement.
# The larger the value, the more important the variable is.
rf_fit$importance
# predict the test data
rf_pred <- predict(rf_fit, test)
table(rf_pred, Survived.test) # evaluating performance on test data
mean(rf_pred == Survived.test) # give test accuracy
# use cross validation to evaluate the model
# we perform 50 times CV
cvK <- 5  # number of CV folds
cv_rf_res <- cv_acc <- c()
for (i in 1:50) {
  cvSets <- cvTools::cvFolds(nrow(train), cvK)  # permute all the data, into cvK folds
  cv_acc <- NA  # initialise results vector
  for (j in 1:cvK) {
    test_id <- cvSets$subsets[cvSets$which == j]
    df_test <- train[test_id, ]
    df_train <- train[-test_id, ]
    fit <- randomForest(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked, data
= df_train,
                        ntree = 500, mtry = 3, proximity = TRUE, importance = TRUE)
    pred <- predict(fit, df_test)
    cv_acc[j] <- mean(pred == df_test$Survived)
  }
  cv_rf_res <- append(cv_rf_res, mean(cv_acc))
}
par(mfrow = c(1, 1))
boxplot(cv_rf_res, horizontal = TRUE, xlab = "Accuracy")
mean(cv_rf_res)*100
# AUC
rf_roc = roc((as.numeric(Survived.test)-1),as.numeric(rf_pred), plot=T,print.auc=T,
             col="red",lwd=4,legacy.axes=T, main="ROC curves for the Random Forest")

## KNN
# fit the train data using knn
# consider 3 neighbours
idx <- c(3, 5:8, 10, 12)
knn_pred <- class::knn(train[, idx], test[, idx], cl = train[, 2], k = 3)
table(knn_pred, Survived.test) # evaluating performance on test data
mean(knn_pred == Survived.test) # give test accuracy
# use cross validation to evaluate the model
# we perform 50 times CV
cvK <- 5  # number of CV folds
```

```
cv_knn_res <- cv_acc <- c()
for (i in 1:50) {
  cvSets <- cvTools::cvFolds(nrow(train), cvK)  # permute all the data, into cvK folds
  cv_acc <- NA  # initialise results vector
  for (j in 1:cvK) {
    test_id <- cvSets$subsets[cvSets$which == j]
    df_test <- train[test_id, ]
    df_train <- train[-test_id, ]
    pred <- class::knn(df_train[, idx], df_test[, idx], cl = df_train[, 2], k = 3)
    cv_acc[j] <- mean(pred == df_test$Survived)
  }
  cv_knn_res <- append(cv_knn_res, mean(cv_acc))
}
par(mfrow = c(1, 1))
boxplot(cv_knn_res, horizontal = TRUE, xlab = "Accuracy")
mean(cv_knn_res)*100
# AUC
knn_roc = roc((as.numeric(Survived.test)-1),as.numeric(knn_pred), plot=T,print.auc=T,
              col="red",lwd=4,legacy.axes=T, main="ROC curves for knn")
```