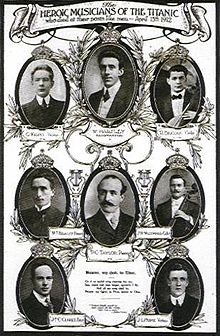


The Titanic, the “unsinkable” ship!

A British luxury passenger liner that sank on April 14–15, 1912, during its maiden voyage, enroute to New York City from Southampton, England. One of the most famous tragedies in modern history, it has inspired numerous stories, several films, and a musical and has been the subject of much scholarship and scientific speculation. In the end, more than 1,500 perished. Aside from the crew, which had about 700 fatalities, third class suffered the greatest loss: of approximately 710, only some 174 survived.  Source <http://www.britannica.com/topic/Titanic>

*Lesser known facts, did you know….*

* *The ship’s band played on for hours after the ship hit the iceberg knowing the ship was sinking….*

**

* *Milton Hershey, a.k.a. the man who's behind one of the greatest chocolate bars of all time, had tickets to be on the Titanic, but changed his plans and didn’t board the ship.*
* More than half of the lifeboats on the ship were not filled to capacity during the evacuation.
* The infamous iceberg that brought the Titanic to her knees has been floating around since around 1,000 B.C
* The budget for the James Cameron film Titanic was actually higher than the budget spent on building the ship in real life.

Source <http://www.omgfacts.com/motors/15969/12-Haunting-Facts-About-The-Titanic-That-You-ve-Never-Heard-Before-7-Blew-My-Mind>

**Introduction**

I have chosen to do my Capstone project on the Titanic. I chose this very early on based on preliminary research which suggested this was a good “starter” project for someone completely new to Data Science and Machine Learning. As the *Titanic survivors story* has become well documented especially with the Data science community due in large part to the Kaggle competition of the same name “Kaggle Data Project: Titanic – Machine Learning From Disaster.”

Two datasets are made available via Kaggle and CRAN (Comprehensive R Archive Network). The objective is to predict whether a passenger survived the Titanic sinking. Given the two datasets Train and Test, each of which include predictor variables such as Age, Passenger Class, Sex, etc. upon these two data sets the following was conducted:

1. **Create a model which will predict whether a passenger survived using only the Train data set.** A subset of the Train data set was created, split into Train.2 and Test.2. This was done so that the model can learn on train.2 and predict on test.2. A confusion matrix was created to “manually” examine the True Positives plus the True Negatives divided by the Total for an overall model Accuracy. \*This split also helps ensure that for the final prediction the model is being applied only once to the Kaggle Test set, in order to gauge the model’s true prediction capabilities against an “unknown” dataset.
2. **Predict whether the passengers survived in the Kaggle Test data set based on the model created.** This final prediction is applied against the Test data and submitted to Kaggle for scoring. This is done by submitting a spreadsheet with predictions for which passengers in the Test data set survived. The spreadsheet has **only two** columns: a column for the Passenger ID and another column which indicates whether they survived (0 for death, 1 for survival).

**Data Exploration:** The key idea here, is to take a look at the data and try to determine which of the variables are related to the target variable we are trying to predict: Survived.

|  |  |
| --- | --- |
|  | The **first tasks** are to read in and examine the datasets “Train” and “Test” using some R commands and functions such as:  The dim () function used to review the dimensions of the dataset.  The str () function used to review the structure of the dataset.  The getwd () function used to get the working directory  The setwd () function used to set the working directory  The head () function used to review the first 6 rows of data  The tail () function used to review the last 6 rows of data  The names () function used to review the variable (column/features) of the dataset  The plot () function used to create some basic graphs  Its very important to understand the dataset you are working with before embarking on coding and modeling.   1. *Understand what each of the column titles represent.* 2. *Understand what each row represents.* 3. *Understand the variables and data types.* 4. *Determine how much data is missing.* |
|  | I also applied some basic plots to get a visual of the data. A few simple generic x-y plots, plotting the density to see general center of if there is a skew? Does is generally take higher values? Where are most of the values concentrated? |
|  |  |
|  | plot (density (train$Age, na.rm = TRUE))  plot (density (train$Fare, na.rm = TRUE))  plot (density (train$Pclass, na.rm =TRUE)) |

\*na.rm = TRUE means ignore the NA’s in the data set.

\*\*density plots are an effective way to view the distribution of a variable

***Survival rate by gender, bar plot***

Taking a look at survival rate filtered by gender. As the story goes, during this era there was a general motto of saving “Women and Children first” in crisis situations, and therefore lies the theory why women and children were the first to board the lifeboats. I created a table and called it “counts”. Then using R’s bar plot () function added an x-axis, y-axis, and main titles. Then calculated the male to female survival rates from the table by indexing the table “counts”. Where counts [1] returns the top left value of the table, counts [2] the bottom left, counts [3] top right, counts [4] bottom right.

*This shows a 74% females over 18% male survival rate.*

R code:

|  |  |
| --- | --- |
|  | counts <- table(train$Survived, train$Sex)  barplot(counts, xlab = "Gender", ylab = "Number of People", main = "Survived vs. Non Survived Males vs. Females")  counts[2] / (counts[1] + counts[2])  counts[4] / (counts[3] + counts[4]) |

***Survival Rate by Passenger Class, bar plot***

Take a look at survival rate filtered by passenger class.

|  |  |
| --- | --- |
|  | R code:  Pclass\_survival <- table (train$Survived, train$Pclass)  barplot (Pclass\_survival, xlab = "Cabin Class", ylab = "Number of People",  main = " Survived vs. Non Survived Males vs. Females")  Pclass\_survival[2] / (Pclass\_survival[1] + Pclass\_survival[2])  Pclass\_survival[4] / (Pclass\_survival[3] + Pclass\_survival[4])  Pclass\_survival[6] / (Pclass\_survival[5] + Pclass\_survival[6]) |

*The results from the above shows 63% of passengers in first class survived over 47% and 24 % in second and third class respectively.*

**Cleaning the datasets:** Manipulating the data to prepare it to be “plugged” into a model. After the exploratory analysis of the data, the next step is to “clean” the data. This additional step is necessary in to further prepare the data for the model. The exploration of the data enables one to examine the elements that need to be cleaned. Commonly, missing data needs to be addressed as was the case with the Train and Test sets. Many variables were missing data, including a feature significant variable Age.

Some basic cleaning tasks were applied to the original Train and Test datasets.

1. Converting categorical variable to dummy variables
2. Imputing missing data
3. Removing some rows
4. Deciding which columns (variables) to omit

**Training a Model:** We first feed the training data into a model, and the model will optimize itself to give you the best explanation for your variables and outcome. The idea is to build a model for predicting survival using the Train dataset. Then input the observations from the Test dataset to predict their survival.

However, to ensure that the model can be generalized to work on “unknown” data. We first split the original train dataset into train.2 and test.2. Then the model is trained the train.2 dataset then “tested” on the test.2 dataset. The confusion matrix results our calculated which shows the overall accuracy of the model using True Positives + True Negatives divided by the Total. Then the final test of the model is applied to the original Test dataset which the model has not seen. The predictions are captured and uploaded to Kaggle to score.

Three models were built, a Decision tree, a General Linear (GLM) and a Random Forest model.

A logistic regression model is a generalized linear model which is used to try to predict something that is binary. Since whether a passenger survived or not is binary, we can use logistic regression GLM.

Tree methods such as CART (classification and regression trees) can be used as alternatives to logistic regression. It is a way that can be used to show the probability of being in any hierarchical group.

Below we’ll explore the code for all three models. Note the # (hash /pound sign is used for comments R will ignore the text aside the # signs.

**## using the Decision Tree model in R:**

set.seed (1234) # set seed so random generated numbers can be reproduced.

## Explore the datasets Train and Test

train <- read.csv ('a-train.csv’, header=T, na. strings=c (" ")) # read in both datasets

test <-read.csv ('a-test.csv’, header=T, na. strings=c (""))

train$Pclass <- ordered (train$Pclass, levels=c ("3","2","1")) # convert the Passenger Class to ordered factor

test$Pclass <- ordered (test$Pclass, levels=c ("3","2","1"))

# R will treat factors as nominal variables and ordered factors as ordinal variables in statistical procedures and graphical analyses.

data <- subset(train,select = c(2,3,5:8)) # cleaning and formatting both datasets Train and Test

data$Age[is.na(data$Age)] <- median(data$Age,na.rm=T) # impute Age where NA, use median of Age

data <- data[!is.na(data$Embarked),] # remove the two rows with missing values in Embarked

rownames(data) <- NULL

## Split the data into a subset of Train (train.2 and test.2) datasets for training and testing the model

# This supports the holdout validation by splitting the training data into two parts,

# a training set and a validation set, building a model with the training set and then

# assessing performance with the validation set.

data = titanic\_train

dim(titanic\_train) # 891 rows/obs 12 var/features

indexes = sample (1: nrow(titanic\_train), size=0.3\*nrow(titanic\_train)) ## sample indexes (randomize)

test.2 = titanic\_train[indexes,] # split data 70% Train 30% Test

dim(test.2) # 267 rows/obs (30%)

train.2 = data[-indexes,]

dim(train.2) # 624 rows/obs (70%)

## install and load the "rpart" package to used generate decision tree models and the "rpart.plot" package to print plots of the trees

#install.packages("rpart") # uncomment to install package(s)

#install.packages("rpart.plot")

library (rpart) # load package(s)

library(rpart.plot)

# Grow the tree & predict survival based on gender and passenger class

decision.tree1 <- rpart (Survived ~ Sex + Pclass, data=train.2)

# plot and don’t abbreviate, length of factor level names in splits, add the question(?) mark to splits

prp (decision.tree1, faclen=0,split.suffix="?",facsep=" or “)

## Interpreting the chart of a "simple" decision tree (decisiontree1)

# The plot shows use that within each gender, the model assigns a lower survival probability to passenger

# with lower passenger classes: men of class 3 and 2 only have a 14% chance of survival while women of classes 2 and 1 have a 95% chance # of survival.

## below adding more features to the decision tree two

decision\_tree2 <- rpart (Survived ~ Sex + Pclass + Age + SibSp + Fare + Embarked,

cp = 0.001, # set complexity parameter

data = train.2)

## The complexity parameter (cp) governs model complexity. A smaller complexity parameter will allow for more complex models.

# cp adjusts the improvement of the model fit necessary for it to create a new branch

# As well, the maximum depth of the tree and the minimum number of observations at each leaf node to limit model complexity

cols <- ifelse(decision\_tree2$frame$yval == 1, "darkred", "green4") # adjusting the colors of the "leaves" green if survived for plotting

prp (decision\_tree2, col=cols,nn.box.col=3) # plot the decision tree two (decisiontree2)

decision\_tree3 <- rpart(Survived ~ Sex + Pclass + Age + SibSp +Fare+Embarked, ## prune the tree to reduce complexity

cp = 0.001, # set complexity parameter

maxdepth = 5, # set maximum tree depth

minbucket = 2, # set min number of obs in leaf nodes

method = "class", # return classifications instead of probs

data = train.2) # use the titanic train.2 dataset

## adjusting the colors of the "leaves" to green if survived for plotting

cols <- ifelse(decision\_tree3$frame$yval == 1, "darkred", "green4")

## plot the "pruned" decision tree

prp (decision\_tree3, col=cols, nn.box.col=3, faclen=0, split. suffix=“?”, facsep=" or “)

## plot tree and don’t abbreviate length of factors names, add? to decision splits

## Make predictions

train\_preds <- predict (decision\_tree3,

newdata=test.2,

type="class”) # return class predictions

## using confusionMatrix () function from caret package, calculates a cross-tabulation of observed and predicted classes with associated statistics.

## Install.packages("caret", dependencies = TRUE) # uncomment to install

library (caret) # load the caret package

confusionMatrix (train\_preds, test.2$Survived)

## Confusion Matrix Accuracy of 0.794

## Manual Confusion Matrix

table (train\_preds, test.2$Survived)

## Accuracy is 0.7940075

## train\_preds 0 1

## 0 153 45

## 1 10 59

## total true positives + total true negatives)/total

## 212/267= 0.7940075

## Submit to Kaggle for score

Prediction <- predict (decision\_tree3, newdata=test, type = "class")

submit <- data.frame(PassengerId = test$PassengerId, Survived = Prediction)

write.csv (submit, file = "kaggle\_decisontree3.csv", row.names = FALSE)

## Kaggle Score# 0.75598

**## using a GLM (General Logistics model) in R**

set.seed(1234) # set seed so random generated numbers can be reproduced.

## Explore the datasets Train and Test

train <- read.csv('a-train.csv',header=T,na.strings=c(""))

test <-read.csv('a-test.csv',header=T,na.strings=c(""))

## Check for missing values and look how many unique values there are for each variable

# using the sapply() function which applies the function passed as argument to each column of the dataframe.

sapply(train,function(x) sum(is.na(x))) # examine and show where there are NAs (missing data). Age has 177, Cabin has 687 and Embarked has 2

sapply (train, function(x) length(unique(x))) # examine and show the unique values e.g. four unique values for Embarked, 89 unique ages etc.

## bellow uncomment to install the Amelia package to use the plotting function missmap() to get a visual of the visual of the missing data

## install. packages ("Amelia")

library(Amelia) # use library () function to load package(s)

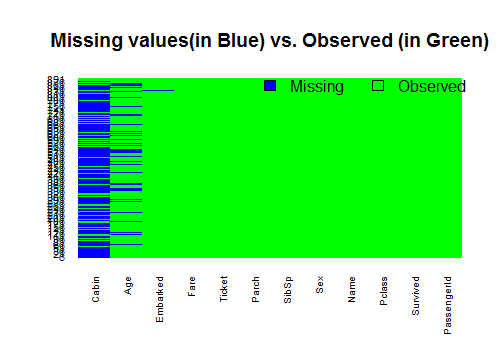
missmap(train, legend = TRUE, col = c("blue","green"), main = "Missing values(in Blue) vs. Observed (in Green)", y.cex = 0.6, x.cex = 0.6 )

## Interpreting the chart:

## The variable cabin has too many missing values, so excluding it.

## In addition excluding passengerId (as it shouldn’t have any relevance to survival)

## Using the subset() function to subset the original dataset selecting the relevant columns only.



data <- subset(train,select=c(2,3,5:8,10,12)) # cleaning and formatting both datasets Train and Test

data$Age[is.na(data$Age)] <- median(data$Age,na.rm=T) # impute Age where NA, use median of Age

data <- data[!is.na(data$Embarked),] # remove the two rows with missing values in Embarked

rownames(data) <- NULL

age.median<- median(test$Age, na.rm=TRUE) # remove NA for Age in Test

test$Age[is.na(test$Age)] = age.median

age.median

mean (test$Age)

## Split original Train set into two subsets train.2 and test.2

# The train.2 set will be used to fit a model which we will apply to the test.2 for prediction.

# Holdout validation involves splitting the original training data into two parts, a sub-training set and a test set.

# then building a model and training it on the sub-train set and then assessing performance on the test set.

data = train

dim(train) # 891 rows/obs 12 var/features

indexes = sample(1:nrow(train), size=0.3\*nrow(train)) #sample Indexes (randomize)

# split data 70% Train 30% Test

test.2 = train[indexes,]

dim(test.2) # 267 rows (30%)

train.2 = data[-indexes,]

dim(train.2) # 624 rows (70%)

names(train.2) # checking Columns Variables

## Building model

glmmodel <- glm (Survived ~ Pclass + Sex + Age + SibSp + Parch , data = train.2, family = binomial ('logit'), maxit = 100)

summary (glmmodel) # review summary function results

## Summary explained:

# we can see that SibSp, Fare and Embarked are not statistically significant.

# As for the statistically significant variables, sex has the lowest p-value suggesting a strong association of the sex of the passenger with the

# probability of having survived.

# The negative coefficient for this predictor suggests that all other variables being equal, the male passenger is less likely to have survived.

# In the logit model the response variable is log odds: ln(odds) = ln(p/(1-p)) = a\*x1 + b\*x2 + . + z\*xn.

# Since male is a dummy variable, being male reduces the log odds by 2.56, and a unit increase in Age reduces the log odds by 0.044

## Make Predictions:

p <- predict(glmmodel, newdata=subset(test.2,select=c(2,3,4,5,6,7,8,10,12)), type="response")

pr <- prediction(p, test.2$Survived)

print (p)

## Confusion Matrix for different thresholds

table (test.2$Survived,p>0.7)

table (test.2$Survived,p>0.7)

## Accuracy using p>0.7

## FALSE TRUE

## 0 117 5

## 1 45 42

## 117 + 42/209= 0.7607656

table (test.2$Survived,p>0.5) # best score

## Accuracy using p>0.5

## FALSE TRUE

## 0 101 21

## 1 23 64

## 101 + 64/209 = 0.7894737

table (test.2$Survived,p>0.3)

##Accuracy using p>0.3

## FALSE TRUE

## 0 76 46

## 1 13 74

## 76 + 74 /209 =0.7177033

## Plot

barchart <- ggplot(test.2, aes(as.factor(Pclass), fill=as.factor(Survived)))+geom\_bar()

barchart+xlab("Passenger Class")+ylab("Number of Passengers")+ggtitle("Survival by Passenger Class")+scale\_fill\_discrete(name = "", labels = c("Died", "Survived"))

ggsave("titanic\_barchart\_submit.png", width = 5, height = 5)

## Submit to Kaggle for score

predict.glm <-predict(glmmodel, newdata=test, type="response")

test$Survived <- as.numeric(as.numeric(predict.glm)>0.7)

write.csv(test[,c("PassengerId", "Survived")],"Kaggle\_glm\_r\_submission4.csv", row.names=F)

## Kaggle GLM Score 0.76555

**## using a Random Forest model in R**

## using a RF (Random Forest) model in R

set.seed(1234) # set seed so random generated numbers can be reproduced.

# install.packages("randomForest") # uncomment to install the random forest package

library(randomForest) # use the library() function to load a package(s)

library(caret)

## Explore the datasets Train and Test

train <- read.csv('a-train.csv',header=T,na.strings=c("")) # read in train dataset

train$Pclass <- ordered(train$Pclass, # convert to ordered factor

levels=c("3","2","1"))

names(train) # checking columns

train <- subset(train,select = c(2,3,5:8)) # cleaning and formatting both datasets Train and Test

train$Age[is.na(train$Age)] <- median(train$Age,na.rm=T) # impute Age where NA, use median of Age

train$Survived <- as.factor(train$Survived) # convert Survived variable to factor

## Split the data into a train.2 and test.2 datasets for training and testing the model

######################################################################################

## This supports the holdout validation by splitting the training data into two parts,

## a training set and a validation set, building a model with the training set and then

## assessing performance with the validation set.

data = train

dim (train) # 891 rows/obs 12 var/features

indexes = sample(1:nrow(train), size=0.3\*nrow(train)) # sample indexes (randomize)

test.2 = train[indexes,] # split data 70% Train 30% Test

dim(test.2) # 267 rows/obs (30%)

train.2 = data[-indexes,]

dim(train.2) # 624 rows/obs (70%)

## Build model on train.2

rf1 <- randomForest (Survived ~ Pclass + Sex + Age + SibSp + Parch,

data= train.2, # data set

ntree=1000, # number of trees to grow

mtry=2) # number of branch variables

rf1 # view model summary

## Output from rf1 model summary () function ran on train.2

## Call:

## randomForest(formula = Survived ~ Pclass + Sex + Age + SibSp + Parch, data = train.2, ntree = 1000, mtry = 2)

## Type of random forest: classification

## Number of trees: 1000

## No. of variables tried at each split: 2

## OOB estimate of error rate: 16.83%

## Confusion matrix:

## Accuracy is 0.8317308

## 0 1 class.error

## 0 355 31 0.08031088

## 1 74 164 0.31092437

## Run rf1 model against test.2

rf2 <- randomForest(Survived ~ Sex + Pclass + Age + SibSp + Parch,

data= test.2, # data set

ntree=1000, # number of trees to grow

mtry=2) # number of branch variables

rf2 # view model summary

varImpPlot(rf2) # plot variable importance

# top 3 vars for Survival are Sex, Age, Pclass

rf2$importance # show important variables

## Output from rf2 model summary () function ran on test.2

## Call:

## randomForest(formula = Survived ~ Sex + Pclass + Age + SibSp + Parch, data = test.2, ntree = 1000, mtry = 2)

## Type of random forest: classification

## Number of trees: 1000

## No. of variables tried at each split: 2

## OOB estimate of error rate: 21.35%

## Confusion matrix:

## Accuracy is 0.7865169

## 0 1 class.error

## 0 140 23 0.1411043

## 1 34 70 0.3269231

# Load and prepare the original test dataset to run model on "unknown" data

test <- read.csv('a-test.csv',header=T,na.strings=c("")) # read in original test dataset

names(test)

test <- subset(test,select = c(1,2,4,5:7)) # cleaning and formatting both datasets Train and Test

test$Pclass <- ordered(test$Pclass, # convert to ordered factor

levels=c("3","2","1"))

test$Age[is.na(test$Age)] <- median(test$Age,na.rm=T) # impute Age where NA, use median of Age

test\_pred <- predict(rf1,newdata = test, type = "class")

prediction\_rf1 <- data.frame(PassengerId=test$PassengerId, Survived = test\_pred)

write.csv (prediction\_rf1, "kaggle\_rf\_submit.csv", row. names=FALSE)

##Kaggle Score 0.75120