Titanic: Machine Learning from Diaster

AC

July 19, 2016

Introduction

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

In this project, I have completed the analysis of what sorts of people were likely to survive. In particular, I have applied the tools of machine learning to predict which passengers survived the tragedy.

The dataset looks like the following:

```r
suppressWarnings(suppressMessages(library(DMwR)))
library(DMwR)
train <- read.csv("train.csv")
test <- read.csv("test.csv")

#Training data set
head(train[,1:3],2)

##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1

head(train[,4:7],2)

##                                                   Name    Sex Age SibSp
## 1                              Braund, Mr. Owen Harris   male  22     1
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1

head(train[,8:12],2)

##   Parch    Ticket    Fare Cabin Embarked
## 1     0 A/5 21171  7.2500              S
## 2     0  PC 17599 71.2833   C85        C
```

```
#Test data set - No "Survived" column
head(test[,1:3],2)

##   PassengerId Pclass                             Name
## 1         892      3                 Kelly, Mr. James
## 2         893      3 Wilkes, Mrs. James (Ellen Needs)

head(test[,4:7],2)

##      Sex  Age SibSp Parch
## 1   male 34.5     0     0
## 2 female 47.0     1     0

head(test[,8:11],2)

##   Ticket   Fare Cabin Embarked
## 1 330911 7.8292              Q
## 2 363272 7.0000              S
```

On closely observing the data, we see that there are a few missing values in the dataset.

```
train[c(6,18,20),4:7]

##                             Name    Sex Age SibSp
## 6              Moran, Mr. James   male  NA     0
## 18 Williams, Mr. Charles Eugene   male  NA     0
## 20      Masselmani, Mrs. Fatima female  NA     0
```

To deal with the missing values, we use the *K-Nearest Neighbours (KNN)* algorithm and replace the missing values with the approximated values. Before we apply the *KNN* algorithm it is important to first merge the training and test datasets.

The merged dataset is obtained by binding of the rows of the two sets. Before binding, we need to make sure that the column titles are the same. In our case, test data set does not have the "Survived" column. So we create one and merge.

```
test$Survived <- NA
merge_data <-  rbind(train,test)
#Number of rows in each data-set
nrow(train)       # 891

## [1] 891
```

```
nrow(test)        # 418
```

```
## [1] 418
```

```
nrow(merge_data)  #1309
```

```
## [1] 1309
```

```
#Applying KNN algorithm
knnOutput <- knnImputation(merge_data[, !names(merge_data) %in% "Survived"])
merge_data <- cbind.data.frame(knnOutput,Survived = merge_data$Survived)
merge_data$Age <- as.integer(merge_data$Age)
merge_data[c(6,18,20),4:7]      #Filled NA values
```

```
##        Sex Age SibSp Parch
## 6     male  25     0     0
## 18    male  33     0     0
## 20 female  21     0     0
```

Feature Addition

Now since the data is in order, we will try to extract more features from the data. The names of the people have titles. So data can be aggregated based on titles. Similarly, the Surnames of the people can help identify family members. These features have been added in the following code snippet:

```
#Extracting meaning from name of the person in the data
name <- as.character(merge_data$Name)

title <-sapply(name, FUN =  function(x) {(strsplit(x,split =
'[,.]'))[[1]][2]})
title <- sub(' ','',title)

surname <- sapply(name, FUN =  function(x) {(strsplit(x,split =
'[,.]'))[[1]][1]})

table(title)  #Original titles
```

```
## title
##         Capt          Col          Don         Dona           Dr
##            1            4            1            1            8
##     Jonkheer         Lady        Major       Master         Miss
##            1            1            2           61          260
##         Mlle          Mme           Mr          Mrs           Ms
##            2            1          757          197            2
##          Rev          Sir the Countess
##            8            1            1
```

```
title <- as.vector(title)

title[title %in% c('Capt','Don','Major', 'Sir')] <- 'Sir'
title[title %in% c('Dona','Lady','the Countess', 'Jonkheer','Mlle','Mme')] <-
'Lady'
title[title %in% 'Ms'] <- 'Miss'
title <- as.factor(title);

table(title)  #Grouped Titles

## title
##    Col     Dr   Lady Master   Miss     Mr    Mrs    Rev    Sir
##      4      8      7     61    262    757    197      8      5

merge_data$Title <- title

#Including family size

merge_data$familySize <- NA
merge_data$familySize <- merge_data$SibSp + merge_data$Parch + 1

merge_data$family_set <- paste(as.character(merge_data$familySize), surname,
sep = "")
merge_data$family_set[merge_data$familySize <3] <- 'small'
char <- merge_data$family_set


famIssue <- data.frame(table(merge_data$family_set))
famIssue <- famIssue[famIssue$Freq<3,]

char1 <- famIssue$Var1
char[char %in% char1] <- 'small'

merge_data$family_set <- char
head(merge_data$family_set)

## [1] "small" "small" "small" "small" "small" "small"
```

Splitting the Dataset

We performed all the feature addition operations on the data and now its time to separate the two data sets into two separate files. It can be done with the following simple command:

```
merge_data$family_set<- as.factor(merge_data$family_set)
train <-  merge_data[1:nrow(train),]
```

```
test <- merge_data[(nrow(train) + 1):nrow(merge_data),]
nrow(train)

## [1] 891

nrow(test)

## [1] 418
```

Training a Conditional Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Conditional forest is an implementation of the random forest and bagging ensemble algorithms utilizing conditional inference trees as base learners.

We train our forest based on the Sex, Age, Pclass, SibSp, Parch, Fare and Title of the train set data. This can be done in R using the following code:

```
#Conditional Random Forest
suppressWarnings(suppressMessages(library(party)))
library(party)
set.seed(755)

fit <- cforest(as.factor(Survived) ~ Sex + Age + Pclass + SibSp + Parch + 
Fare +
                      Title + Embarked + familySize + family_set,
                    data = train, controls = cforest_unbiased(ntree = 2000, 
mtry=3))
```

Predicting Survivals from Test Data

The number of survivals in the test data can be predicted using the *predict* function in R.

```
pre <- predict(fit, test, OOB=TRUE, type = "response")
```

Creating an Excel Result File

Once we get the predicted values of survivals, we can create an excel file of the result in whatever format we want. For sake of convenience, we will be creating an csv file with two columns *Passenger ID* and *Survival Output*. It can be prepared using the following lines of code:

```r
predictions <- data.frame(PassengerId = test$PassengerId, Survived = pre)
write.csv(predictions, file = "myoutput.csv", row.names = FALSE)
```

References

1. Dataset and instructions: Kaggle https://www.kaggle.com/c/titanic

2. General methodolgy: http://trevorstephens.com/kaggle-titanic-tutorial/getting-started-with-r/