Survival on the Titanic Estimation

1. Introduction

This paper attempts to describe the creation of a model which predicts whether a specific passenger on the RMS Titanic is expected to have survived or not, based on a group of passenger characteristics.

The data for this project come from the "Titanic: Machine Learning from Disaster" competition on the Kaggle web site.

Due to Kaggle policy, the original data used are only available to Kaggle users. If you would like to reproduce this research, please sign in to Kaggle to download the data to a local folder and make sure to set the working directory accordingly.

We have been provided with a comma separated value file named 'train.csv', which contains a list of 891 records of the RMS Titanic passengers, each record consisting of 12 fields (variables).

VARIABLE DESCRIPTIONS:

1. **PassengerId** : Passenger's ID, integer number.
2. **Survived** : Survival status (0 = No; 1 = Yes)
3. **Pclass** : Passenger's traveling Class (1 = 1st; 2 = 2nd; 3 = 3rd)
4. **Name** : Passenger's full name
5. **Sex** : Passenger's sex (male/female)
6. **Age** : Passenger's age (in years)
7. **SibSp** : Passenger's number of Siblings/Spouses Aboard
8. **Parch** : Passenger's number of Parents/Children Aboard
9. **Ticket** : Passenger's Ticket Number
10. **Fare** : Passenger's Fare
11. **Cabin** : Passenger's Cabin
12. **Embarked** : Passenger's Port of Embarkation (C=Cherbourg; Q=Queenstown; S=Southampton)

In order to measure the success of our chosen model, the outcome of the prediction is going to be compared against a test set and compute the percentage of correct answers (model's accuracy).

2. Tools and methodology

In order to make a prediction, we will be using a classification model. The solution will be developed using ***R***, a well-known programming language for data analysis.

Various classifications algorithms have been implemented in ***R*** . The "caret" package alone, includes almost 150 different models. We will test a number of models, decide which offers the best accuracy and use that model to make our final prediction and submit for evaluation to Kaggle. Our choice contains the following models:

1. GLM - Generalized Linear model
2. RPART - Decision Trees
3. RF - Random Forest model
4. GBM - Generalized Boosted Regression Model
5. LDA - Linear Discriminant Analysis

6. NB - Naive Bayes model

```
# Load the appropriate packages, set the working directory and read
the data
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
library(rpart)
library(randomForest)
```

```
## randomForest 4.6-7
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(gbm)
```

```
## Loading required package: survival
## Loading required package: splines
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: parallel
## Loaded gbm 2.1
```

```
library(plyr)
library(MASS)
library(klaR)
setwd("E:/Titanic")
titanic <- read.csv("train.csv", header = T)
```

Before proceeding with the analysis, we must first choose which variables we are going to use for our prediction. Opening the 'train.csv' file with MS EXCEL gives us a first opportunity to inspect the data. Here we use the 'head()' R function to display only the first few rows of the data.

```
# Display the first 6 rows of data
head(titanic)
```

```
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
## 5           5        0      3
## 6           6        0      3
##                                                  Name    Sex Age
SibSp
## 1                             Braund, Mr. Owen Harris   male  22
1
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38
1
## 3                              Heikkinen, Miss. Laina female  26
0
## 4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35
1
## 5                            Allen, Mr. William Henry   male  35
0
## 6                                    Moran, Mr. James   male  NA
0
##   Parch           Ticket   Fare Cabin Embarked
## 1     0        A/5 21171  7.250              S
## 2     0         PC 17599 71.283   C85        C
## 3     0 STON/O2. 3101282  7.925              S
## 4     0           113803 53.100  C123        S
## 5     0           373450  8.050              S
## 6     0           330877  8.458              Q
```

It is obvious that the **"Passgengerld"** and the **"Name"** variables are not of any use in the outcome estimation, so we are going to leave them out. The **"Ticket"** attribute also falls in the same category, so we are going to leave it out as well. The **"Cabin"** attribute might be of relative importance, but the "Cabin" column has too many missing values, so it may not help us.

From the remaining attributes, the **"Embarked"** attribute does not seem to have any logical connection to a passenger's chances of survival, except perhaps from the fact that passengers who embarked first might have better knowledge of the ship's layout and thus better chances to find their way to a lifeboat.

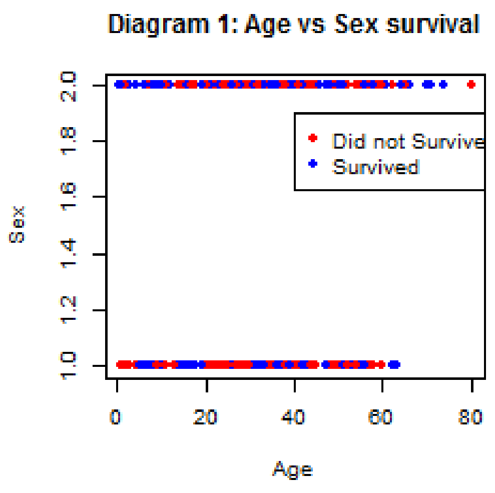
The **"SibSp"** and **"Parch"** variables may be related to the survival of a passenger, as a parent might choose to save his/her children first or as children might not be able to find their way to the lifeboats on their own. Moreover, the probability of all the members of a family surviving should be lower for large families. However, as the **"SibSp"** and the **"Parch"** variables do not define family relations, they may be of less additional value to our model. Therefore, we should test to see whether their inclusion in the prediction model improves the model's accuracy.

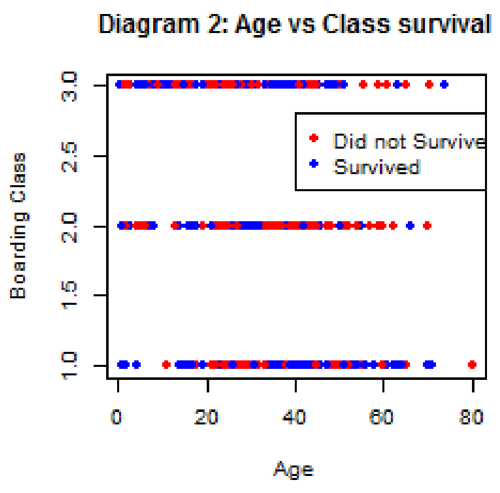
All the remaining variables seem to have direct connection to the survival rate, as depicted in the following diagram:

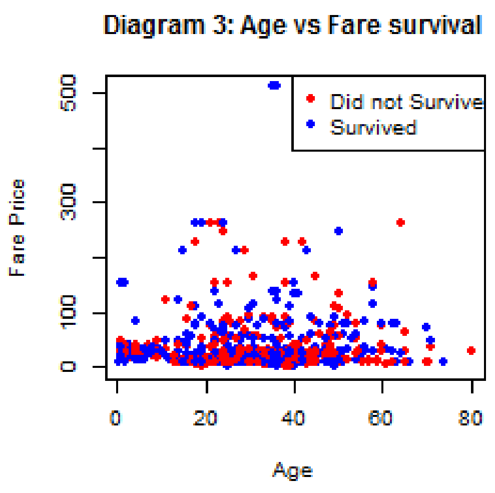
```
par(mfrow = c(2, 2))
plot(titanic$Age, titanic$Sex, type = "p", col = c("red", "blue"),
pch = 19,
    main = "Diagram 1: Age vs Sex survival", xlab = "Age", ylab =
"Sex")
legend(x = 40, y = 1.9, legend = c("Did not Survive", "Survived"),
pch = 19,
    col = c("red", "blue"))

plot(titanic$Age, titanic$Pclass, type = "p", col = c("red",
"blue"), pch = 19,
    main = "Diagram 2: Age vs Class survival", xlab = "Age", ylab =
"Boarding Class")
legend(x = 40, y = 2.8, legend = c("Did not Survive", "Survived"),
pch = 19,
    col = c("red", "blue"))

plot(titanic$Age, titanic$Fare, type = "p", col = c("red", "blue"),
pch = 19,
    main = "Diagram 3: Age vs Fare survival", xlab = "Age", ylab =
"Fare Price")
legend("topright", legend = c("Did not Survive", "Survived"), pch =
19, col = c("red",
    "blue"))
```







From the first diagram we may see that there were more female survivors (Sex value=2) than male ones, and that for both sexes, children present a higher survival rate than adults.

In the second diagram, it is obvious that the ratio of survivors is higher for the 1st class passengers, while the fewest survivors were among the 2nd class passengers.

Finally, in the last diagram we notice that as the fare price goes up, the ratio between those who survived and those who did not, changes in favor of those who managed to survive.

3. Cleaning Data.

One of the most important aspects of data analysis is the data cleaning. In our case, data have come in a very “clean” form of comma separated values, however, by inspecting the data we can see that there are many missing values in the **“Cabin”** column as well as some missing values in the **“Age”** column. Since we have opted not to consider **“Cabin”** as a variable in our model, we shall leave the column out altogether. On the other hand, **“Age”** is a significant variable, so we shall use those rows of the data that have values in the **“Age”** field and leave out only the rows for which **“Age”** is empty.

We are also going to convert the values of the **“Survived”**, **“Sex”** and **“Pclass”** variables from integers (continuous variables) to 'levels' (categorical variables), using the *factor* data type of **R**.

```
# Leave the 11th column (Cabin) out
traindata <- titanic[, -11]

# Create a logical vector of rows which have values in all fields
compcases <- complete.cases(traindata)

# Keep only the complete rows
traindata <- traindata[compcases, ]
traindata$Survived <- as.factor(traindata$Survived)
traindata$Sex <- as.factor(traindata$Sex)
traindata$Pclass <- as.factor(traindata$Pclass)
```

4. Splitting data into Training and Test sets

Before submitting our results for evaluation to Kaggle, we should be able to do some testing in order to choose the best model and the best variable combination. Unfortunately, the test data provided by Kaggle do not include the survivor variable to compare with, so we are going to split the originally available training data into one train and one test set, with a ratio of 70% - 30%. So, we shall use the 70% of the original data to train our models and the 30% to test and evaluate our models.

```
# Use the same seed number in order to reproduce this analysis
set.seed(1234)

trainidx <- createDataPartition(y = traindata$PassengerId, p = 0.7,
list = F)
ttrn <- traindata[trainidx, ]
ttst <- traindata[-trainidx, ]
```

5. Creating and choosing classifiers

In this part we are going to create our models, train them on the training set, evaluate them on the test set and choose the best model based on the model's accuracy.

```
modellist <- data.frame(Accuracy = c(0, 0, 0, 0, 0, 0), row.names =
c("GLM",
    "RPART", "RF", "GBM", "LDA", "NB"))
predmod <- train(Survived ~ Sex + Age + Fare + Parch + SibSp +
Embarked, data = ttrn,
    method = "glm")
pred <- predict(predmod, ttst)
predcomp <- confusionMatrix(ttst$Survived, pred)
modellist[1, 1] <- predcomp$overall[1]

predmod <- train(Survived ~ Sex + Age + Fare + Parch + SibSp +
Embarked, data = ttrn,
    method = "rpart")
pred <- predict(predmod, ttst)
predcomp <- confusionMatrix(ttst$Survived, pred)
modellist[2, 1] <- predcomp$overall[1]

predmod <- train(Survived ~ Sex + Age + Fare + Parch + SibSp +
Embarked, data = ttrn,
    method = "rf")
pred <- predict(predmod, ttst)
predcomp <- confusionMatrix(ttst$Survived, pred)
modellist[3, 1] <- predcomp$overall[1]

predmod <- train(Survived ~ Sex + Age + Fare + Parch + SibSp +
Embarked, data = ttrn,
    method = "gbm", verbose = F)
pred <- predict(predmod, ttst)
predcomp <- confusionMatrix(ttst$Survived, pred)
modellist[4, 1] <- predcomp$overall[1]

predmod <- train(Survived ~ Sex + Age + Fare + Parch + SibSp +
Embarked, data = ttrn,
    method = "lda")
pred <- predict(predmod, ttst)
predcomp <- confusionMatrix(ttst$Survived, pred)
modellist[5, 1] <- predcomp$overall[1]

predmod <- train(Survived ~ Sex + Age + Fare + Parch + SibSp +
Embarked, data = ttrn,
    method = "nb")
pred <- predict(predmod, ttst)
predcomp <- confusionMatrix(ttst$Survived, pred)
modellist[6, 1] <- predcomp$overall[1]

print(modellist)
```

```
##         Accuracy
## GLM       0.7594
## RPART     0.7642
## RF        0.8066
## GBM       0.8019
## LDA       0.7594
## NB        0.7642
```

As we see, the **RF** model achieves the highest accuracy (slightly better than the **GBM**), so we will opt for **Random Forest** as a classifier.

6. Submitting prediction for evaluation

It is now time to submit our prediction for evaluation at Kaggle. In order to do that, we are going to run our model on the original training data, apply it to the original test data offered by Kaggle, produce a prediction and submit it.

However, we should notice that there are missing values in the test dataset. And if with the training dataset we could omit the rows with null values, here we have to make a prediction for EVERY passenger. In order to achieve that, we must fill in the missing values. We opt to use the ***mean*** value of every variable for all the missing values of this variable.

```
test <- read.csv("test.csv", header = T)
meanage = mean(test$Age, na.rm = TRUE)
meanfare = mean(test$Fare, na.rm = TRUE)
for (i in 1:length(test$Age)) {
    if (is.na(test$Age[i])) {
        test$Age[i] = meanage
    }
    if (is.na(test$Fare[i])) {
        test$Fare[i] = meanfare
    }
}
predmod <- train(Survived ~ Sex + Age + Fare + Parch + SibSp +
Embarked, data = traindata,
    method = "rf")
pred <- predict(predmod, test)
prediction <- data.frame(PassengerId = test$PassengerId, Survived =
pred)
write.table(prediction, "prediction.csv", row.names = FALSE, sep =
",", col.names = TRUE)
```

The accuracy achieved was **0.78469** and got a rank of **667**. This is well above the average accuracy (0.77512 / 961), but it cannot be considered adequate. We will have to tune our model to achieve a better ranking.

7. Model tuning.

There are a few simple tweaks we could do to achieve better accuracy.

7.1 Filling in empty values.

In our first attempt, we left out the rows with empty values either in the **“Age”** or in the **“Fare”** fields. However, we had to fill in the empty values in the test dataset. So, perhaps we could use the same method to fill in the training dataset as well, and see how we will do.

7.2 Variable combination

As mentioned earlier, the **“Embarked”**, **“Parch”** and **“SibSp”** variables seem not to have great relation to the survival status of a passenger, so we may omit them and see what happens to the model's accuracy.

We also consider the fact that the fare rate may be closely related to the **“Pclass”** variable, so when creating our prediction model, we should not only consider the fare value by itself but take into account the relation between the fare and the class as well.

Applying these tweaks, changed our code as follows:

```
library(caret)
library(randomForest)
library(rpart)
library(gbm)
library(klaR)
setwd("E:/Titanic")
titanic <- read.csv("train.csv", header = T)

meanage = mean(titanic$Age, na.rm = TRUE)
meanfare = mean(titanic$Fare, na.rm = TRUE)

for (i in 1:length(titanic$Age)) {
    if (is.na(titanic$Age[i])) {
        titanic$Age[i] = meanage
    }
    if (is.na(titanic$Fare[i])) {
        titanic$Fare[i] = meanfare
    }
}

titanic$Survived <- as.factor(titanic$Survived)
titanic$Sex <- as.factor(titanic$Sex)
titanic$Pclass <- as.factor(titanic$Pclass)

predmod <- train(Survived ~ Sex + Age + Fare + Fare:Pclass, data =
titanic,
    method = "rf", verbose = F)


test <- read.csv("test.csv", header = T)

meanage = mean(test$Age, na.rm = TRUE)
meanfare = mean(test$Fare, na.rm = TRUE)

for (i in 1:length(test$Age)) {
    if (is.na(test$Age[i])) {
        test$Age[i] = meanage
    }
    if (is.na(test$Fare[i])) {
        test$Fare[i] = meanfare
    }
}

test$Sex <- as.factor(test$Sex)
test$Pclass <- as.factor(test$Pclass)

pred <- predict(predmod, test)

prediction <- data.frame(PassengerId = test$PassengerId, Survived =
pred)
write.table(prediction, "prediction.csv", row.names = FALSE, sep =
",", col.names = TRUE)
```

The improvement was lower than expected, as the accuracy rose by only 0.00478 to **0.78947**. Still, the ranking went up by 180 places, to **470**.

This may indicate that the model is overfitting the training data, and although it does well on the training dataset, it does not perform as well on the test dataset.

Obviously there is room for further tuning, and other classifiers could also be tested in our effort to improve the model's accuracy.

Created by **Themos Patrikios** as part of the Coursera's 'Introduction to Data Science' course, Aug 2014.