Titanic sruvivors prediction model

Mohammed Amine BOUSSETTA

2019-04-12

Contents

Introduction

This report is part of the second project in the Professional data science certification from Harvardx. We chose the Titanic dataset from Kaggle as a machine learning project because it is a story that remained in our head after we saw the movie when we were young. This choice was also based on the quality of the dataset: it is understandable and has few variables which will make us work with the visualization tools. It also contains many NAs and outliers that need some cleaning. We already worked in a bigger kaggle competitions "Santander Customer Transaction Prediction" with 200 variables where logistic regression, LDA, QDA, KNN and random forest had average accuracy comparing to other algorithm like Gradient Boosting Machines. This data set was already cleaned with no serious correlation between variables and the column names were hidden. There was simply no room for data cleaning and exploration which are appreciated sections in a report, especially for this projet. Those were the reasons why we chose this wonderful dataset that can help beginners like us understand data science. But first, let's talk about the Titanic disaster.

It took 3 years to design the most beautiful boat ever designed: the Titanic! Construction began in 1909 and was completed in 1912 on the Harland & Wolff shipyard in Belfast. On April 10, 1912, the Titanic leaves the port of Southampton (England) to meet the fate we know so well … Two days later, the liner receives the first signal of icebergs and dense fog off the Atlantic coast. On the night of 14 to 15 April 1912, the Titanic hits an iceberg and shipwreck leaving behind 1500 dead and 700 survivors … Many facts were called into question at the time: the number of lifeboats, communication on board , the signals sent to the boats, the evacuation … That was the biggest maritime disaster of all times.

They say that money can't buy your life but it may save it! Many articles published at the time showed for example that more first class passengers survived than those in third class in term of percentages. Well, let's prove this with science and try to make predictions on whether a passenger survived or not using the provided information in the dataset.

In the next sections, we will first discover the dataset, prepare it and then make some exploration analysis to make first thoughts about the modeling approach. Then we will present the two models used in this project to predict the survivals (Random forest and K nearest neighbors). Finally, we will show and discuss our results and talk about further improvements to this work.

1- Data preparation

1-1 Overview

After donwloading the files from the Kaggle data in the Titanic competition, we read the file to local variable **fulldataset** and create two data sets, **train** and **test** using createDataPartition from Caret package:

```
fulldataset <- read.csv("./train.csv")
set.seed(1)
test_index <- createDataPartition(y = fulldataset$Survived, times = 1,
                                  p = 0.2, list = FALSE)
trainset <- fulldataset[-test_index,]
testset <- fulldataset[test_index,]

dim(trainset)
```

```
## [1] 712  12
```

```
dim(testset)
```

```
## [1] 179  12
```

Let's see now how our data looks:

```
head(fulldataset)
```

```
##   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  7.2500              S
## 2     0         PC 17599 71.2833   C85        C
## 3     0 STON/O2. 3101282  7.9250              S
## 4     0           113803 53.1000  C123        S
## 5     0           373450  8.0500              S
## 6     0           330877  8.4583              Q
```

This dataset has 12 variables. Some of them are clear enough. Let's describe the others:

- pclass: A proxy for socio-economic status (SES):
- 1st = Upper
- 2nd = Middle

- 3rd = Lower
- sibsp: The dataset defines family relations in this way:
- Sibling = brother, sister, stepbrother, stepsister
- Spouse = husband, wife (mistresses and fiancés were ignored)
- parch: The dataset defines family relations in this way:
- Parent = mother, father
- Child = daughter, son, stepdaughter, stepson

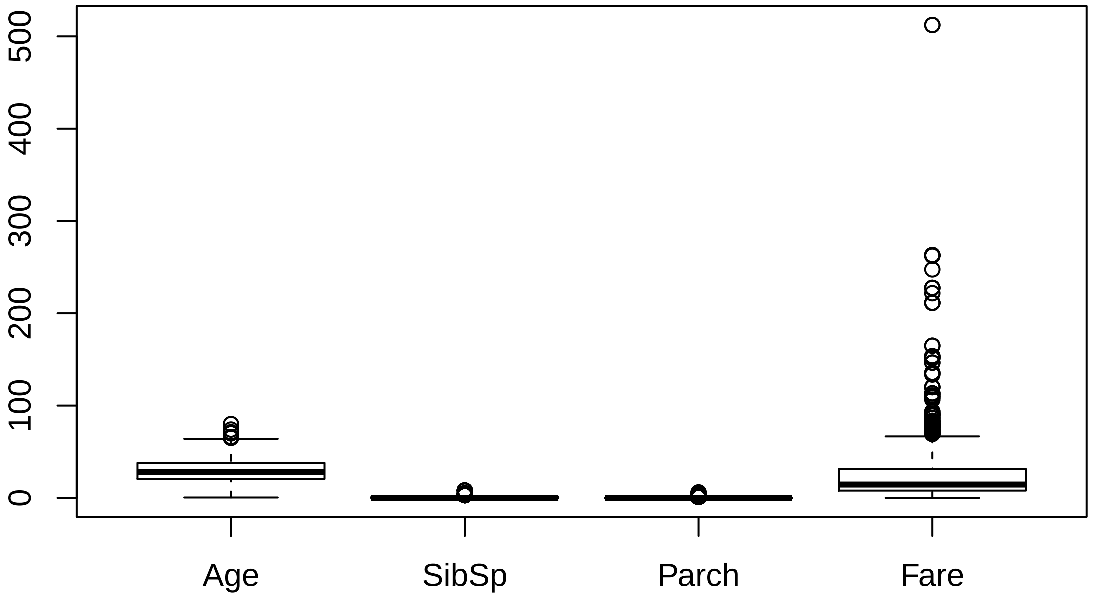
Let's check now how many NAs we got in the trainset (first 891 rows from the full data):

```
colSums(is.na(trainset))
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0           0           0           0           0         143
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           0           0           0
```

The only columns that has NAs is the Age with 177 missed entry. Lets check now for outliers with this simple boxplot of the AGe, number of family members and the ticket price:

```
boxplot(trainset[,c(6,7,8,10)])
```



We can see that the Fare variable has a distinct value of 512.33 very far from the average fare. Let's check this again by making a summary of this data:

```
summary(trainset[c(6,7,8,10)])
```

```
##       Age            SibSp            Parch            Fare
##  Min.   : 0.42   Min.   :0.0000   Min.   :0.000   Min.   :  0.000
##  1st Qu.:20.50   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:  7.896
##  Median :28.00   Median :0.0000   Median :0.000   Median : 14.454
##  Mean   :29.59   Mean   :0.5478   Mean   :0.375   Mean   : 32.379
##  3rd Qu.:38.00   3rd Qu.:1.0000   3rd Qu.:0.000   3rd Qu.: 31.387
##  Max.   :80.00   Max.   :8.0000   Max.   :6.000   Max.   :512.329
##  NA's   :143
```

Three people from first class bought this very expensive ticket. Maybe this is not an outlier. It would be if the class of those people was 3rd class our 2nd.

Finally, let's transform some non numeric variables into factors to optimise the RAM and make summaries more interesting.

```
vars <- c('PassengerId','Pclass','Sex','Embarked', 'Survived')
fulldataset[vars] <- lapply(fulldataset[vars], function(x) as.factor(x))
```

1-2 Cleaning and feature engineering

Sometimes, we can extract valuable information from raw variables. For example, if we sum the variable Parch and SibSp, we get the family size. We can also get from the variable Name the title of the person, probably if you have a high social rank you will be served first like a doctor, a countess, a queen or princess… Let's add those columns to our dataset:

```
# Add the age stage, the family size and the title
fulldataset <- mutate(fulldataset, Title = gsub('(.*, )|(\\..*)', '', fulldataset$Name),
                      Agestage = ifelse(Age<18, "Child", "Adult"),
                      Familysize = SibSp + Parch + 1)
```

Let's check now the count of those titles by sex and age stage:

```
table(fulldataset$Sex, fulldataset$Title)
```

```
##
##          Capt Col Don  Dr Jonkheer Lady Major Master Miss Mlle Mme  Mr Mrs
##   female    0   0   0   1        0    1     0      0  182    2   1   0 125
##   male      1   2   1   6        1    0     2     40    0    0   0 517   0
##
##           Ms Rev Sir the Countess
##   female   1   0   0            1
##   male     0   6   1            0
```

```
table(fulldataset$Agestage, fulldataset$Title)
```

```
##
##         Capt Col Don  Dr Jonkheer Lady Major Master Miss Mlle Mme  Mr Mrs
##   Adult    1   2   1   6        1    1     2      0   95    2   1 376 104
##   Child    0   0   0   0        0    0     0     36   51    0   0  22   4
##
##          Ms Rev Sir the Countess
##   Adult   1   6   1            1
##   Child   0   0   0            0
```

We seen the the Title **"Master"** only contains children. Some titles are also redundant like Mlle (French title for Miss) and Mme is French for Mrs. We also have 2 people with the title "Ms". Then, we have a lot of rare titles like Major, countess, Dr, …

Let's reassign those redundant titles and group the rare ones:

```
# Titles with very low cell counts that should be grouped
rare <- c('Capt', 'Col', 'Don',
                'Dona','Dr', 'Jonkheer', 'Lady', 'Major','Rev', 'Sir','the Countess')

# Also reassign mlle, ms, and mme accordingly
fulldataset$Title[fulldataset$Title == 'Mlle'] <- 'Miss'
fulldataset$Title[fulldataset$Title == 'Ms'] <- 'Miss'
fulldataset$Title[fulldataset$Title == 'Mme'] <- 'Mrs'
fulldataset$Title[fulldataset$Title %in% rare] <- 'Rare Title'
```

```
# Show title counts again
table(fulldataset$Title)
```

```
##
##     Master       Miss         Mr        Mrs Rare Title
##         40        185        517        126         23
```

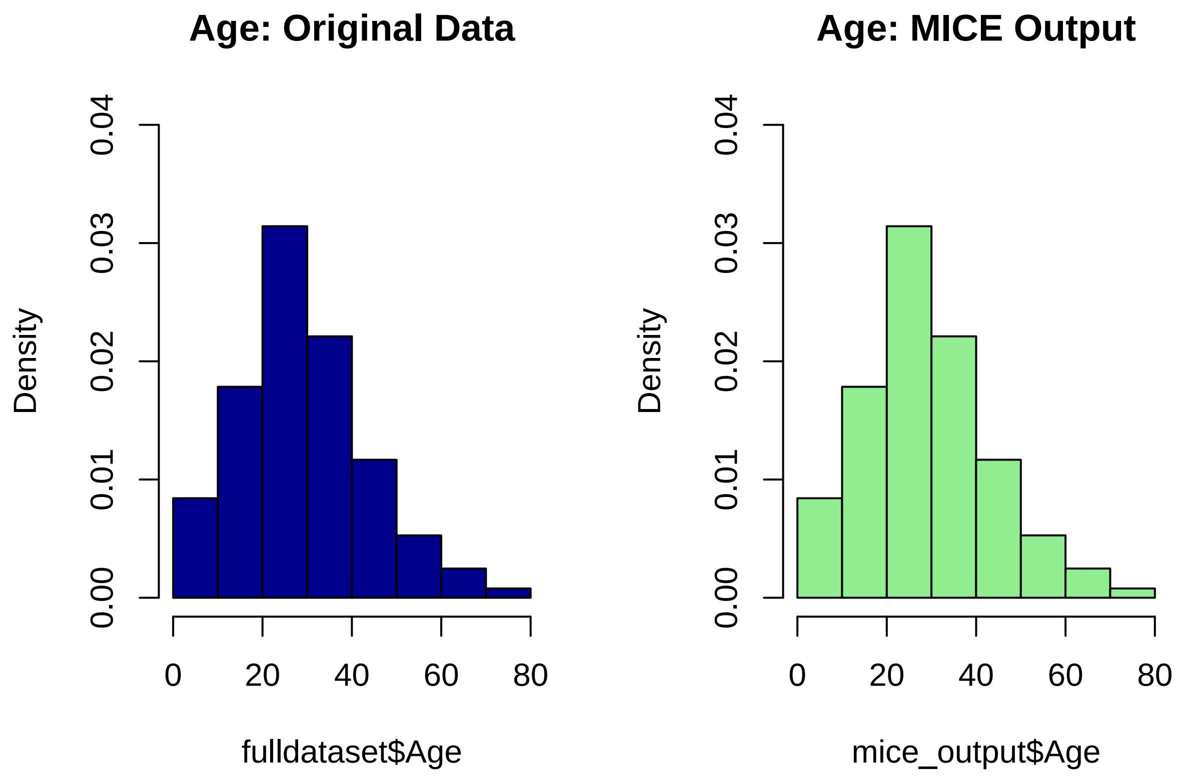
Now let's deal with the Age NAs. The mice package implements a method to deal with missing data. The abreviation stands for Multivariate Imputation By Chained Equations. Let's do this and compare the output with the original data:

```
#Fill the age NAs with Mice:
mice <- mice(fulldataset[, !names(fulldataset) %in% c('PassengerId','Name',
                                                       'Ticket','Cabin','Familisize',
                                                       'Survived')], method='rf')
```

```
##
##  iter imp variable
##   1   1  Age
##   1   2  Age
##   1   3  Age
##   1   4  Age
##   1   5  Age
##   2   1  Age
##   2   2  Age
##   2   3  Age
##   2   4  Age
##   2   5  Age
##   3   1  Age
##   3   2  Age
##   3   3  Age
##   3   4  Age
##   3   5  Age
##   4   1  Age
##   4   2  Age
##   4   3  Age
##   4   4  Age
##   4   5  Age
##   5   1  Age
##   5   2  Age
##   5   3  Age
##   5   4  Age
##   5   5  Age
```

```
mice_output <- complete(mice)
fulldataset$Age <- mice_output$Age

# Plot the two age distributions
par(mfrow=c(1,2))
hist(fulldataset$Age, freq=F, main='Age: Original Data',
     col='darkblue', ylim=c(0,0.04))
hist(mice_output$Age, freq=F, main='Age: MICE Output',
     col='lightgreen', ylim=c(0,0.04))
```



Finally, let's add an important piece of information: Mother or not mother from the variables Age, Title, Parch and Sex:

```
fulldataset$Mother <- 'Not Mother'
fulldataset$Mother[fulldataset$Sex == 'female' & fulldataset$Parch > 0 &
              fulldataset$Age > 18 & fulldataset$Title != 'Miss'] <- 'Mother'
```

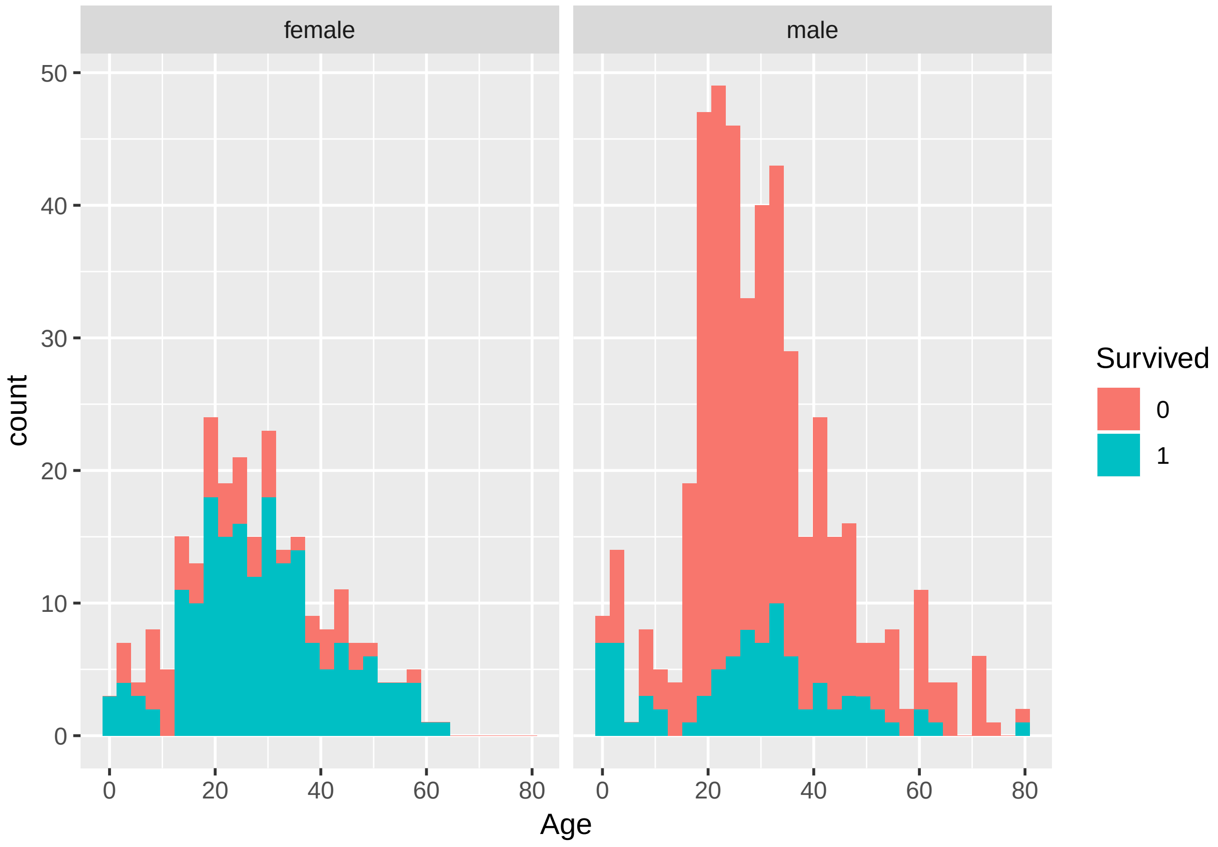
The dataset is now cleaned and ready to be explored. Let's split it back into train and set:

```
trainset <- fulldataset[-test_index,]
testset <- fulldataset[test_index,]
```

2- Data exploration and analysis

With no results yet, we can guess that children and mothers were saved first before men and this is natural in disasters. We can also assume that people with high titles and people with high class tickets will make it. Let's confirm those assumptions using ggplot:

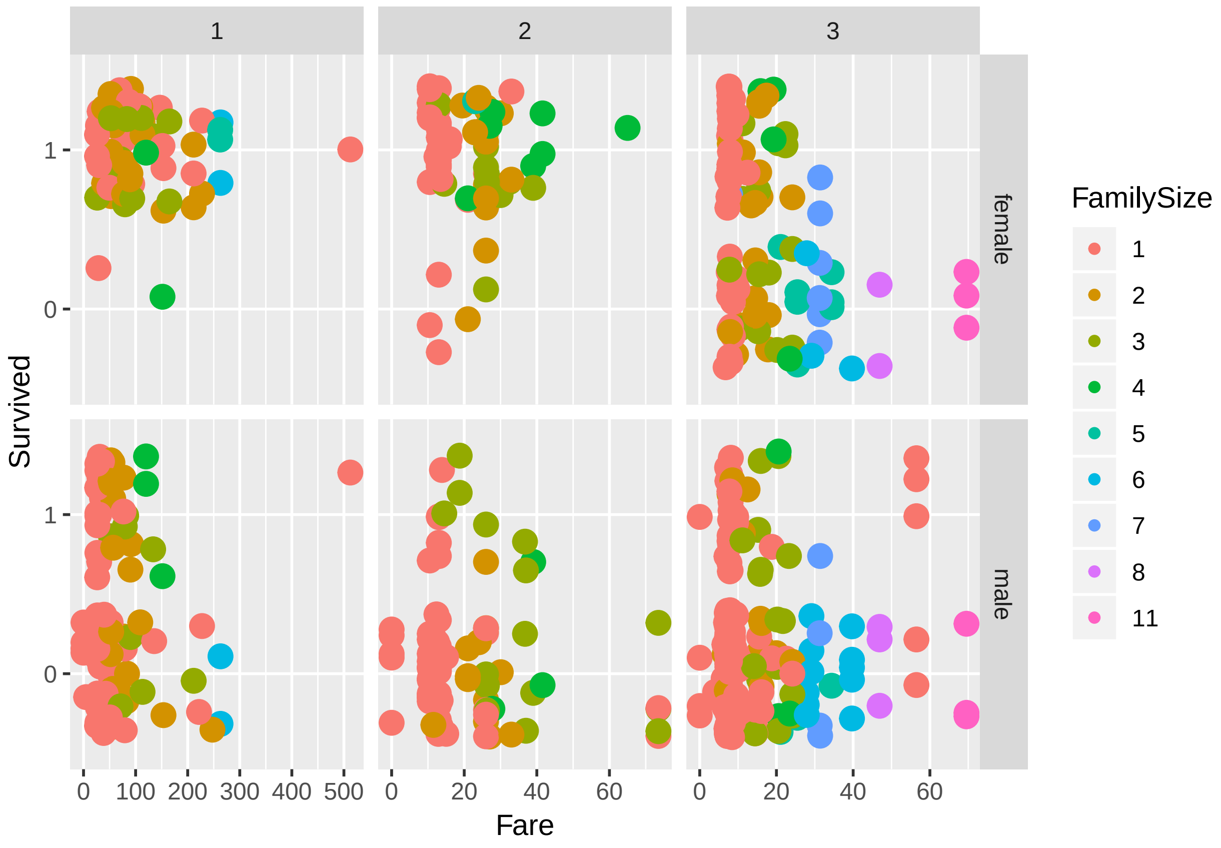
```
ggplot(trainset, aes(Age, fill = Survived)) +
  geom_histogram() +
  facet_grid(.~Sex)
```



We can see from the histogram that the propotion of people who survived is greater for females vs males and for children vs adults. Many people who did not survive are aged between 18 and 60.

Now let's plot a more serious graph combining the family size, the fare, the class and the survival:

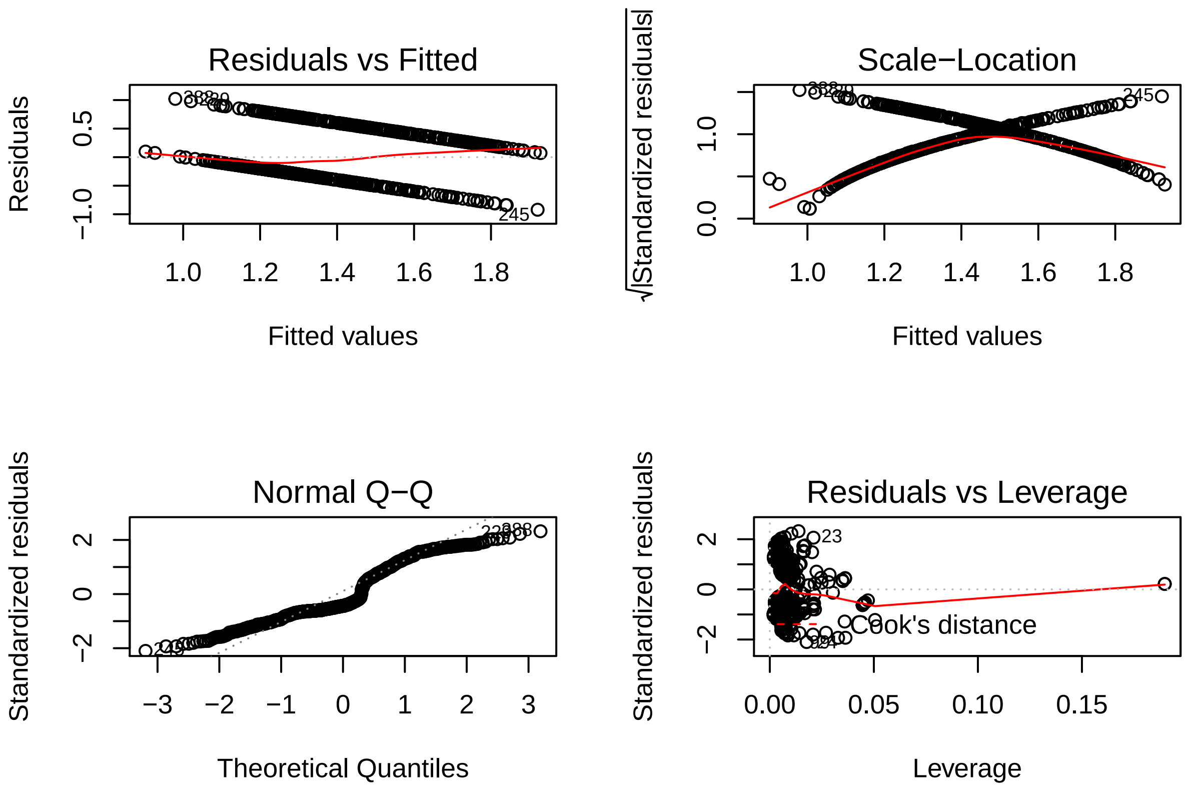
```
trainset %>%
  ggplot(aes(x = Fare, y = Survived)) +
  geom_jitter(aes_string(color = as.factor(trainset$Familysize)),
              size = as.factor(trainset$Familysize)) +
  facet_grid(Sex~Pclass, scales = "free") +
  scale_color_discrete("FamilySize") +
  scale_size_discrete("FamilySize")
```



This plot has plenty to say: first, large families didn't survive in most cases. This could be explained by the fact that members of a family try to search for each other during a disaster so they have less chance to get to the survival boats in time. We also remark that people with first class tickets survived most of the time especially if they are females (up-left corner). The fare is highly correletad to the class so the same reasoning applies.

Now let's see the correlation between some chosen variables and Survived using anova analysis:

```r
df <- trainset %>% select(Survived,Age,Fare, Pclass, Familysize)
df <- as.data.frame(lapply(df, function(x) as.numeric(x)))
fit <- aov(Survived ~ Age + Fare + Familysize + Pclass, data = df)
layout(matrix(c(1,2,3,4),2,2)) # optional layout
plot(fit)
```



```r
summary(fit)
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)    
## Age           1   0.33   0.330   1.685   0.1946    
## Fare          1  11.37  11.372  57.994 8.45e-14 ***
## Familysize    1   0.97   0.967   4.931   0.0267 *  
## Pclass        1  16.31  16.310  83.180  < 2e-16 ***
## Residuals   707 138.63   0.196                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the summary, we can see that **Fare** and **Pclass** are the most significant followed by **Familysize** and **Age**.

3- Building the models:

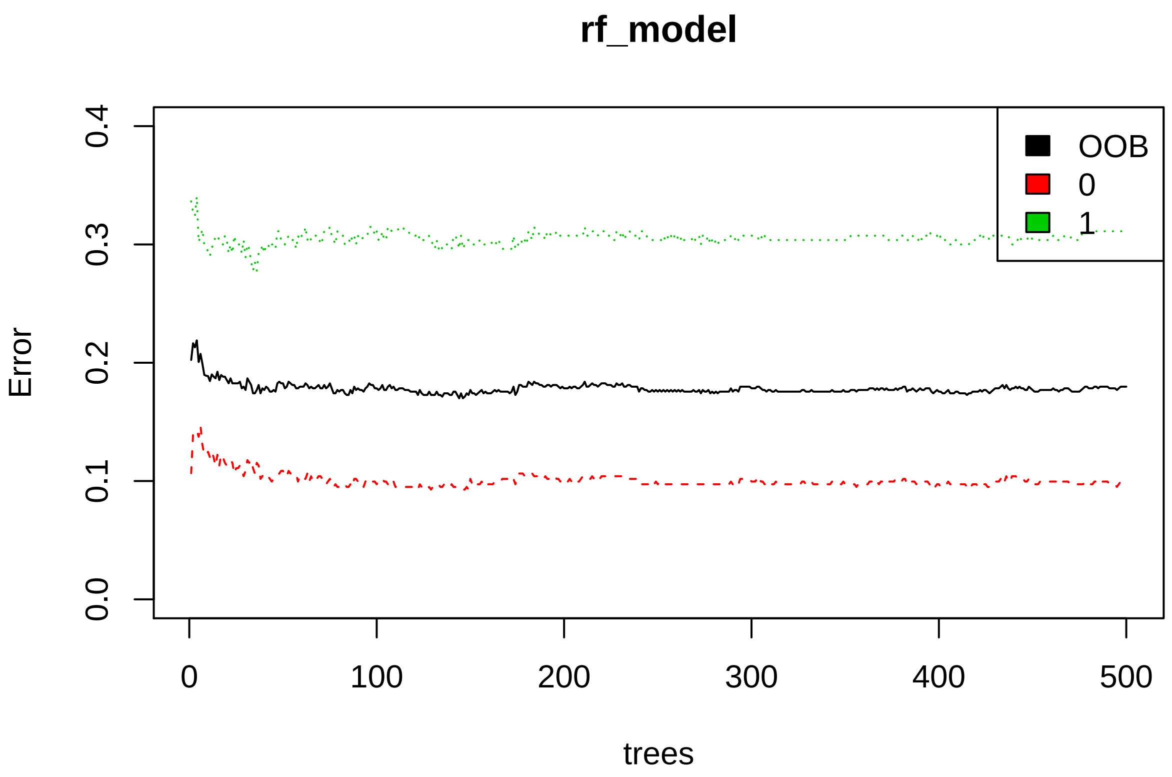
3-1 Random forest:

Let's build our first model using random forest algorithm. We will choose the variables that seemed to impact the classification in the previous section:

```r
# Training the random forest model
trainset <- trainset %>% mutate_if(is.character, as.factor)
testset <- testset %>% mutate_if(is.character, as.factor)

rf_model <- randomForest(Survived ~ Sex + Age + SibSp + Parch +
                           Pclass + Fare + Mother + Embarked +
                           Title + Familysize, data = trainset,
                           na.action = na.exclude)
```

```r
# Plot the erro
plot(rf_model, ylim=c(0,0.4))
legend('topright', colnames(rf_model$err.rate), col=1:3, fill=1:3)
```

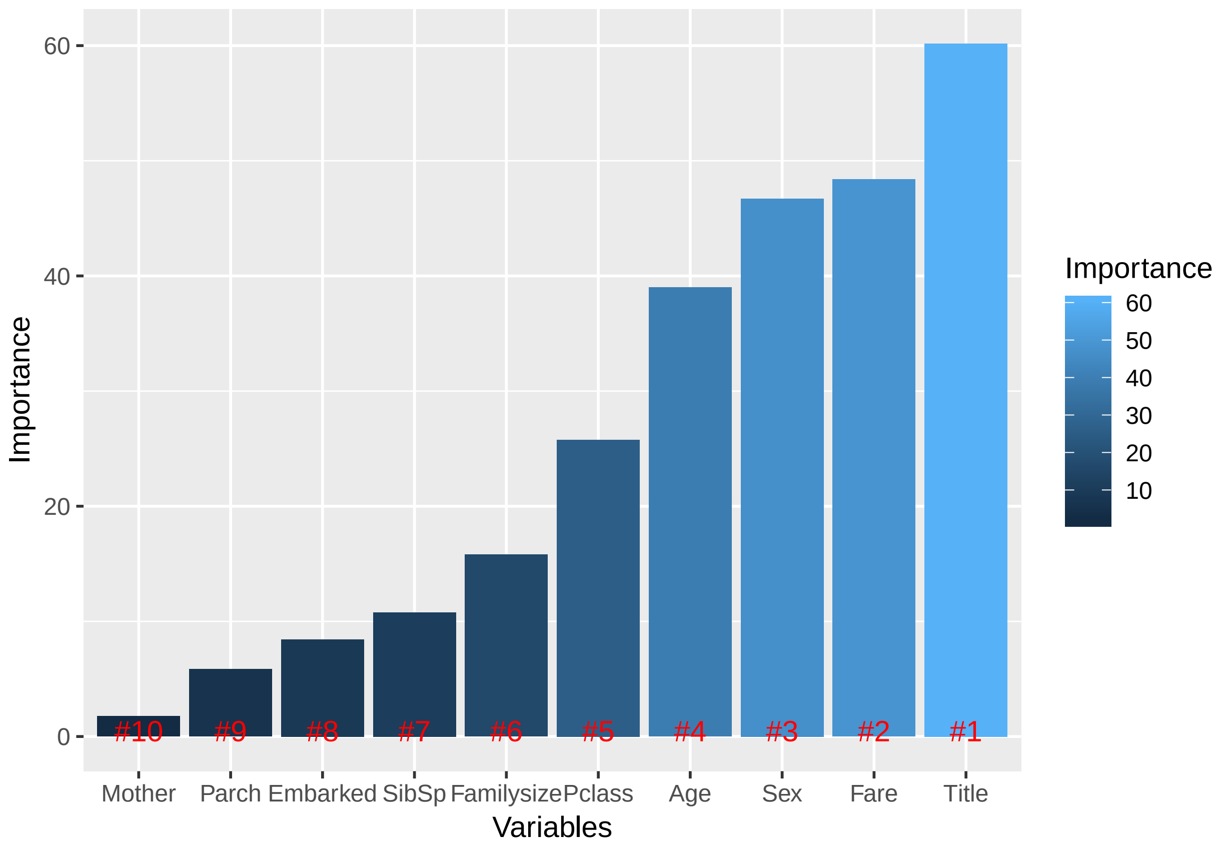


In this plot, we see that the global error is around 20%. The green line represents the error of people who survived (30%) and the red one represents those who died (10%). Now let's get the relative importance of all those variables:

```r
# get the variable importance
rf_importance<- importance(rf_model)
var_importance <- data.frame(Variables = row.names(rf_importance),
                            Importance = round(rf_importance[ ,'MeanDecreaseGini'],2))

# Create a rank variable based on importance
rank_importance <- var_importance %>%
  mutate(Rank = paste0('#',dense_rank(desc(Importance))))

# Use ggplot2 to visualize the relative importance of variables
ggplot(rank_importance, aes(x = reorder(Variables, Importance),
                           y = Importance, fill = Importance)) +
  geom_bar(stat='identity') +
  geom_text(aes(x = Variables, y = 0.5, label = Rank),
            colour = 'red') +
  labs(x = 'Variables')
```



And now it's time for predictions:

```
prediction <- predict(rf_model, testset)
RF_acc <- confusionMatrix(prediction,testset$Survived)
RF_acc
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 92 18
##          1 15 54
##
##                Accuracy : 0.8156
##                  95% CI : (0.751, 0.8696)
##     No Information Rate : 0.5978
##     P-Value [Acc > NIR] : 3.41e-10
##
##                   Kappa : 0.614
##  Mcnemar's Test P-Value : 0.7277
##
##             Sensitivity : 0.8598
##             Specificity : 0.7500
##          Pos Pred Value : 0.8364
##          Neg Pred Value : 0.7826
##              Prevalence : 0.5978
##          Detection Rate : 0.5140
##    Detection Prevalence : 0.6145
##        Balanced Accuracy : 0.8049
##
##         'Positive' Class : 0
##
```

```
# calculating rmse
prediction <- as.numeric(prediction)
testsetpred<- as.numeric(testset$Survived)
RF_rmse <- RMSE(prediction,testsetpred)
RF_rmse
```

```
## [1] 0.4293688
```

3-2 K-nearest neighbors:

In the second model, we will use KNN and crossvalidation using the caret package to train it and then make predictions:

```
trainset <- fulldataset[-test_index,]
testset <- fulldataset[test_index,]

# Controlling the parameters of the training phase
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(1)

#train the model
knn_fit <- train(Survived ~ Sex + Age + Familysize + Fare + Title + Mother,
                 data = trainset, method = "knn",
                 trControl=trctrl,
                 preProcess = c("center", "scale"),
                 tuneLength = 10)

# Show the sumary
knn_fit
```

```
## k-Nearest Neighbors
##
## 712 samples
##   6 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (9), scaled (9)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 641, 641, 641, 640, 641, 641, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.8090050  0.5872844
##    7  0.8118349  0.5928045
##    9  0.8235589  0.6192615
##   11  0.8249478  0.6224131
##   13  0.8310316  0.6351605
##   15  0.8324335  0.6379354
##   17  0.8366523  0.6473975
##   19  0.8319770  0.6373708
##   21  0.8301056  0.6335431
##   23  0.8230960  0.6199895
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 17.
```

Using 10 folds and repeated 3 times, the model that gave the optimal result used a value of 17 for the parameter k. The accuracy on the training set was 0.8366.

Now let's make the predictions and compare the accuracy of this model with the Random forest:

```r
# Make predictions and get accuracy results
results<- predict(knn_fit, newdata=testset)
Knn_acc<- confusionMatrix(results, testset$Survived)
Knn_acc
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 85 18
##          1 22 54
##
##                Accuracy : 0.7765
##                  95% CI : (0.7084, 0.8353)
##     No Information Rate : 0.5978
##     P-Value [Acc > NIR] : 3.185e-07
##
##                   Kappa : 0.5395
##  Mcnemar's Test P-Value : 0.6353
##
##             Sensitivity : 0.7944
##             Specificity : 0.7500
##          Pos Pred Value : 0.8252
##          Neg Pred Value : 0.7105
##              Prevalence : 0.5978
##          Detection Rate : 0.4749
##    Detection Prevalence : 0.5754
##       Balanced Accuracy : 0.7722
##
##        'Positive' Class : 0
##
```

```r
prediction <- as.numeric(results)
testsetpred<- as.numeric(testset$Survived)
Knn_rmse <- RMSE(prediction,testsetpred)
Knn_rmse
```

```
## [1] 0.4727195
```

4- Presenting the results:

This is the final table for the models prediction results:

```r
res <- data.frame(RandomForest = RF_acc$overall, Knn = Knn_acc$overall)
res["rmse", ] <- matrix(c(RF_rmse,Knn_rmse), ncol = 2)
kable(format(res, scientific = FALSE), caption = "Summary of the two models")
```

Table 1: Summary of the two models

	RandomForest	Knn
Accuracy	0.8156424581005587	0.7765363128492
Kappa	0.6139972554401099	0.5394906097247
AccuracyLower	0.7509528811304463	0.7083811287792
AccuracyUpper	0.8695679559101372	0.8352982062316
AccuracyNull	0.5977653631284916	0.5977653631285
AccuracyPValue	0.0000000003410293	0.0000003185356
McnemarPValue	0.7277235466695501	0.6352562959972
rmse	0.4293687714534458	0.4727194592471

The RF algorithm performs better than Knn when evaluated on accuracy and on rmse.

5- Conclusion and future work

This report comes to the end. We have shown using data exploration that some variables in the Titanic dataset were very important when predicting the survival of people. We then built two models and test them on accuracy and rmse and the random forest algorithm won. As future work, we can use neural networks and gradient boosting machines and make further feature engineering to improve those metrics.

We participated in the Titanic competitions and used lightgbm in python to score a 0.82 accuracy making us 364 out of 10974 participants in this moment.

We would like to thank professor Rafael for this wonderful course and the students for making the forums a nice place to complete our learning.