Getting Started with R: Titanic Competition in Kaggle

Armand Ruiz - IBM

Thursday, March 26, 2015

This R Markdown document is based on the excellent tutorial of Trevor Stephens that you can find here: http://trevorstephens.com/post/72916401642/titanic-getting-started-with-r (http://trevorstephens.com/post/72916401642/titanic-getting-started-with-r).

This Markdown document is part of the IBM DeveloperWorks article: Using DashDB and Bluemix to solve a Kaggle Competition (http://example.com)

Kaggle is a platform for predictive modelling and analytics competitions on which companies and researchers post their data and statisticians and data miners from all over the world compete to produce the best models.

kaggle (https://www.kaggle.com/c/titanic-gettingStarted/details/new-getting-started-with-r) (https://www.kaggle.com/c/titanic-gettingStarted/details/new-getting-started-with-r) (https://www.kaggle.com/c/titanic-gettingStarted/details/new-getting-started-with-r)

In this article we are going to show some of the approaches that you can take to participate in the competition **Titanic: Machine Learning from Disaster**. The goal is to complete the analysis of what sorts of people were likely to survive.

(https://www.kaggle.com/c/titanic-gettingStarted/details/new-getting-started-with-r) (https://www.kaggle.com/c/titanic-gettingStarted/details/new-getting-started-with-r) (https://www.kaggle.com/c/titanic-gettingStarted/details/new-getting-started-with-r)

As most of Kaggle competitions, there are two datasets:

- training set: complete with the outcome for a group of passengers
- test set: you must predict the now unkown target variable based on the passenger attributes

We previously loaded our data in dashDB and now we can just load this data into data frames in R. Let's take a look at the structure of the dataframe:

```
#library(ibmdbR)
#con <- idaConnect("BLUDB","","")
#idaInit(con)
#query1<-paste('select * from train)
#train <- idaQuery(query1,as.is=F)
train <- read.csv("~/GitHub/TitanicShinyApplication/data/train.csv")
test <- read.csv("~/GitHub/TitanicShinyApplication/data/test.csv")
str(train)
```

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 358 277 16 559 520 629
 417 581 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 681 levels "110152","110413",..: 524 597 670 50 473 276 86 396 345
133 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 1 57 1 1 131 1 1 1 ...
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
```

Using the **table** command we will get a vector that will count the occurrence of each value. Let's run this in order to get the number of passengers that survived and then number of passengers that died.

```
table(train$Survived)
```

```
##
##   0   1
## 549 342
```

We can see that 342 passengers survived and 549 died. If instead you want to get the proportion, we can use the function **prop.table**

```
prop.table(table(train$Survived))
```

```
##
##         0         1
## 0.6161616 0.3838384
```

Regarding this dataset, 38% of the passengers.

Prediction 1: Everybody dies

In this first prediction to test the set dataframe is going to be assumed that everybody dies. Using the command **rep** we will have something repeated by the number of times we tell it too.

```
test$Survived <- rep(0, 418)
```

Since everybody is dead in the dataframe, it will repeat 0's prediction 418 times, the number of rows we have in the test dataset.

To summit in Kaggle.com we need to prepare a CSV file with the PassengerID as well as our Survived prediction.

```
submit <- data.frame(PassengerId = test$PassengerId, Survived = test$Survived)
write.csv(submit, file = "Prediction1.csv", row.names = FALSE)
```

After submitting we get a score of 0.62679 and a pretty low possition of 2006. But with such a simple approach we have a 62% of accuracy.

2006	new	Armand Ruis	0.62679	1

Prediction 2: The Gender-Class Model

In this new approach we will take a look at the Sex and Age variable to see if any patterns are evident.

```
summary(train$Sex)
```

```
## female   male
##    314    577
```

Majority of the passengers were male. Let's see a comparison on the number of males and females that survived:

```
prop.table(table(train$Sex, train$Survived),1)
```

```
##
##                  0         1
##   female 0.2579618 0.7420382
##   male   0.8110919 0.1889081
```

We can see that the majority of females aboard survived, and a very low percentagle of males did. In this new prediction we will make all the females survive and all the males die.

```
test$Survived <- 0
test$Survived[test$Sex == 'female'] <- 1
#Creating Submit File
submit <- data.frame(PassengerId = test$PassengerId, Survived = test$Survived)
write.csv(submit, file = "Prediction2.csv", row.names = FALSE)
```

Submitting again…and..Awesome! Now we are in position 1697, we are 309 higher and now the accuracy is 76.55%.

1697	new	Armand Ruis	0.76555	2

Let start checking the age variable:

```
summary(train$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.42   20.12   28.00   29.70   38.00   80.00     177
```

The main issue we see in the age variable is that we have 177 missing values. There are many ways to face this issue, in this case is going to be assumed that the missing values are the average age of the rest of the passengers, late twenties.

In order to use proportion tables we have to have categorical variables. Let's create a new variable, Child, to indicate wheter the passenger is below the age of 18:

```
train$Child <- 0
train$Child[train$Age < 18] <- 1
```

Let’s create a table with both gender and age to see the survival proportions for different subsets. For that we use the **aggregate** command that takes a formula with the target variable on the left hand side of the tilde symbol and the variables to subset over on the right. We then tell it which dataframe to look at with the data argument, and finally what function to apply to these subsets.

```
 aggregate(Survived ~ Child + Sex, data=train, FUN=length)
```

```
##   Child    Sex Survived
## 1     0 female      259
## 2     1 female       55
## 3     0   male      519
## 4     1   male       58
```

This looked at the length of the Survived vector for each subset and output the result. Let’s check now the proportion.

```
aggregate(Survived ~ Child + Sex, data=train, FUN=function(x) {sum(x)/length(x)})
```

```
##   Child    Sex  Survived
## 1     0 female 0.7528958
## 2     1 female 0.6909091
## 3     0   male 0.1657033
## 4     1   male 0.3965517
```

If the passenger is female most survive and if they were male most don’t, regardless of wheter they were a child or not. So we are not going to change anything in our predictions here.

Next we will check the class variable, that is limited to a manageable 3 values, the fare is again a continuous variable that need to be reduced to something that can be easily tabulated.

```
train$Fare2 <- '30+'
train$Fare2[train$Fare < 30 & train$Fare >= 20] <- '20-30'
train$Fare2[train$Fare < 20 & train$Fare >= 10] <- '10-20'
train$Fare2[train$Fare < 10] <- '<10'
```

Noe let’s run a longer aggregate function to see if there’s anything interesting to work with:

```
aggregate(Survived ~ Fare2 + Pclass + Sex, data=train, FUN=function(x) {sum(x)/length(x)})
```

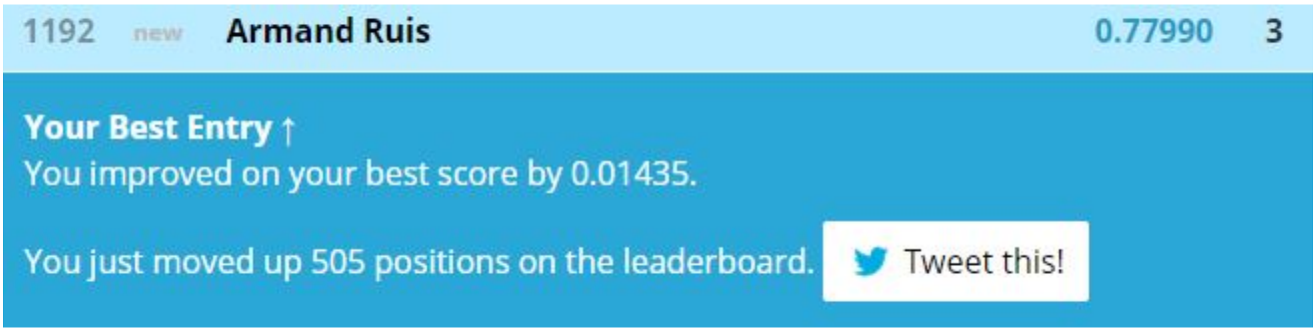
```
##    Fare2 Pclass    Sex  Survived
## 1  20-30      1 female 0.8333333
## 2    30+      1 female 0.9772727
## 3  10-20      2 female 0.9142857
## 4  20-30      2 female 0.9000000
## 5    30+      2 female 1.0000000
## 6    <10      3 female 0.5937500
## 7  10-20      3 female 0.5813953
## 8  20-30      3 female 0.3333333
## 9    30+      3 female 0.1250000
## 10   <10      1   male 0.0000000
## 11 20-30      1   male 0.4000000
## 12   30+      1   male 0.3837209
## 13   <10      2   male 0.0000000
## 14 10-20      2   male 0.1587302
## 15 20-30      2   male 0.1600000
## 16   30+      2   male 0.2142857
## 17   <10      3   male 0.1115385
## 18 10-20      3   male 0.2368421
## 19 20-30      3   male 0.1250000
## 20   30+      3   male 0.2400000
```

While the majority of males, regardless of class or fare still don't do so well, we notice that most of the class 3 women who paid more than $20 for their ticket actually also miss out on a lifeboat.

Let's make a new prediction based on this new insight.

```
test$Survived <- 0
test$Survived[test$Sex == 'female'] <- 1
test$Survived[test$Sex == 'female' & test$Pclass == 3 & test$Fare >= 20] <- 0
#Creating Submit File
submit <- data.frame(PassengerId = test$PassengerId, Survived = test$Survived)
write.csv(submit, file = "Prediction3.csv", row.names = FALSE)
```

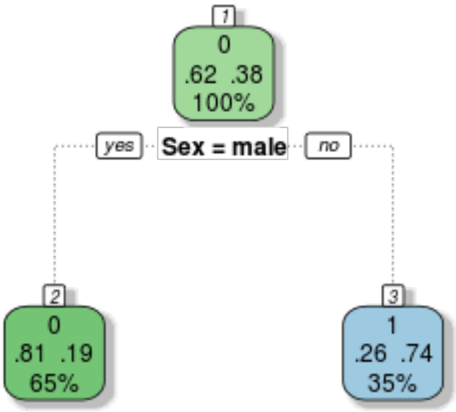
We are now in position 1192, moved up 505 positions and our score is 77.99%!



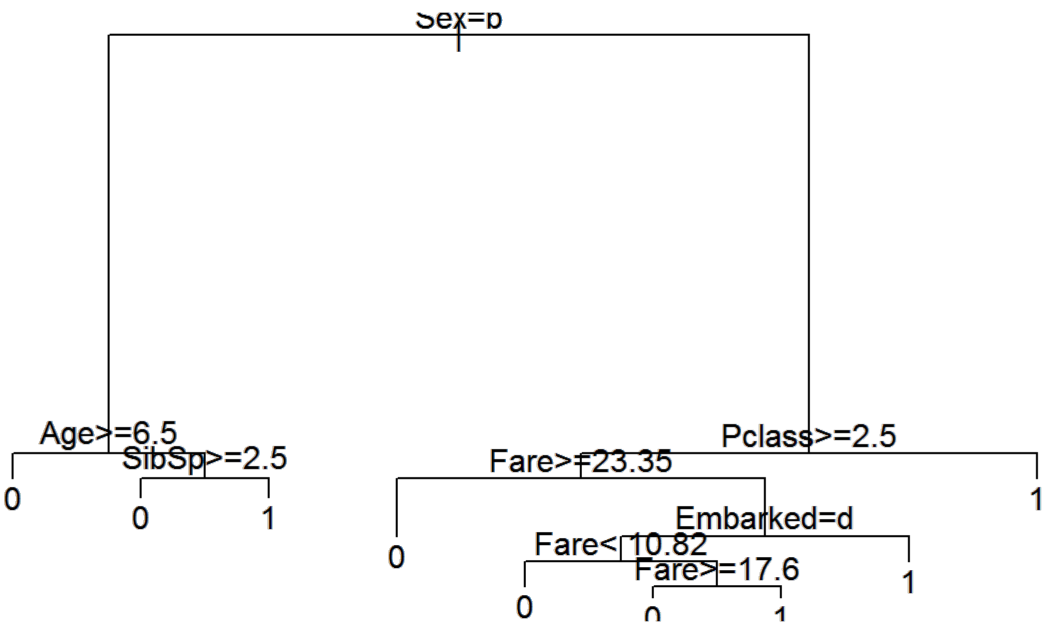
Prediction 3: Decision trees

Decision trees have a number of advantages. After running the model you can see exactly what decisions will be made for unseen data that you want to predict. They are intuitive and can be read by people with little experience.

The algorithm start with all of the data at the root node and scans all of the variables for the best one to split on.




```
library(rpart)
fit <- rpart(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked, data=train, method
="class")
plot(fit)
text(fit)
```

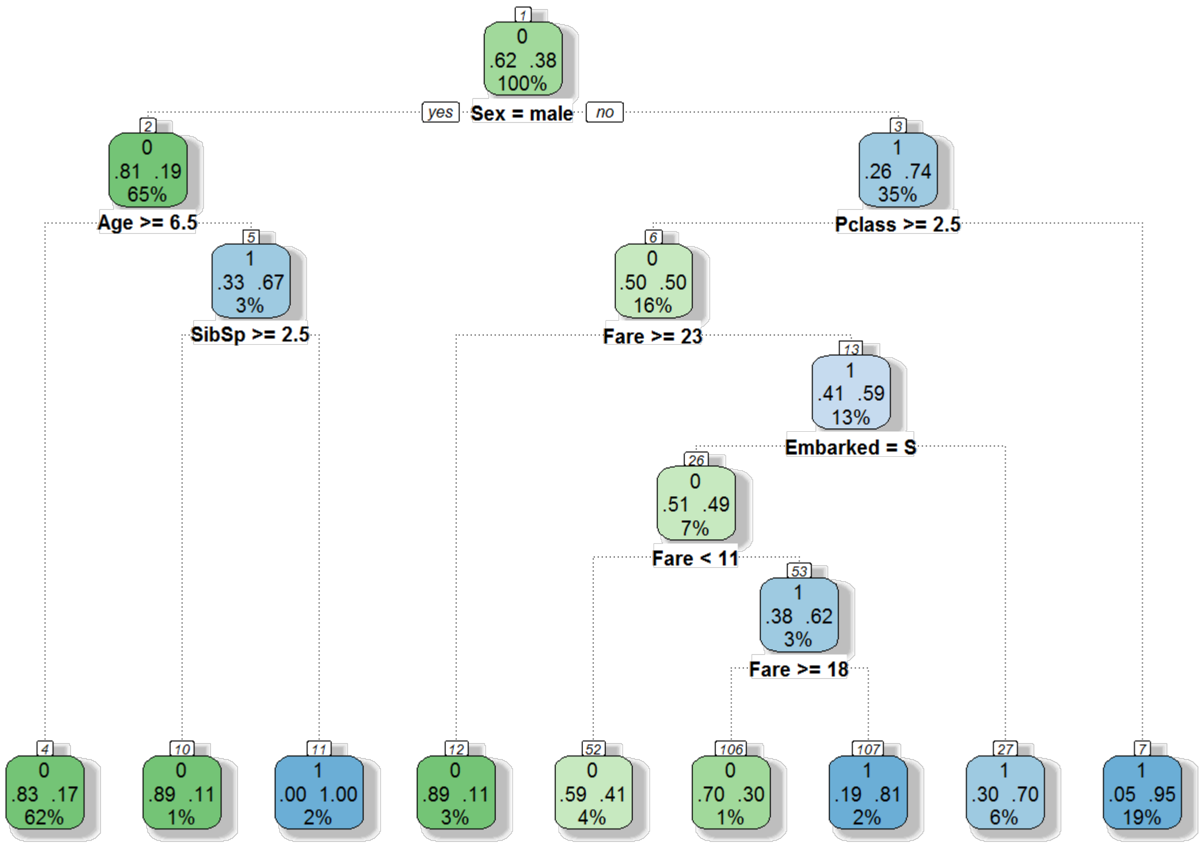


To create more informative graphic, using some external packages we can get some fancy trees.

```
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.3.0 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(rpart.plot)
library(RColorBrewer)
fancyRpartPlot(fit)
```



Rattle 2015-Mar-27 12:42:03 aruizga7

Let’s create the prediction based on this decision tree and submit it to Kaggle.

```
Prediction <- predict(fit, test, type = "class")
submit <- data.frame(PassengerId = test$PassengerId, Survived = Prediction)
write.csv(submit, file = "Prediction4.csv", row.names = FALSE)
```

Now we scored 78.469%, which is an improvement of 0.478% and we moved up 324 positions!

Prediction 4: Feature Engineering

Feature engineering is very important. Even a simple model with great features can outperform a complicated algorithm with poor ones.

In this case we are going to concentrate in the Name attribute. Obviously no one in the boat shared the same name but there is something they all share. The persons title, and this might give us a little insight.

```r
train <- read.csv("~/GitHub/TitanicShinyApplication/data/train.csv")
test <- read.csv("~/GitHub/TitanicShinyApplication/data/test.csv")
train$Name[1]
```

```
## [1] Braund, Mr. Owen Harris
## 891 Levels: Abbing, Mr. Anthony ... Zimmerman, Mr. Leo
```

Next step is going to be to extract these title to make new variables. This will be performed in the training and testing set.

We can use the function **strsplit** to break apart our original name. We use **regular expressions** to do that.

```r
test$Survived <- NA
combi <- rbind(train, test)
combi$Name <- as.character(combi$Name)
combi$Name[1]
```

```
## [1] "Braund, Mr. Owen Harris"
```

```r
strsplit(combi$Name[1], split='[,.]')
```

```
## [[1]]
## [1] "Braund"       " Mr"          " Owen Harris"
```

```r
strsplit(combi$Name[1], split='[,.]')[[1]]
```

```
## [1] "Braund"       " Mr"          " Owen Harris"
```

```r
strsplit(combi$Name[1], split='[,.]')[[1]][2]
```

```
## [1] " Mr"
```

```r
combi$Title <- sapply(combi$Name, FUN=function(x) {strsplit(x, split='[,.]')[[1]][2]})
combi$Title <- sub(' ', '', combi$Title)
table(combi$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
```

There are a few rare titles that is better to combine them into a single category. We do that using the **%in%**

operator, that checks to see if a value is part of the vector we're comparing to. We create different categories called **Sir**, **Lady**, **Mlle**, and we change the variable type to a factor.

```
combi$Title[combi$Title %in% c('Mme', 'Mlle')] <- 'Mlle'
combi$Title[combi$Title %in% c('Capt', 'Don', 'Major', 'Sir')] <- 'Sir'
combi$Title[combi$Title %in% c('Dona', 'Lady', 'the Countess', 'Jonkheer')] <- 'Lady'
combi$Title <- factor(combi$Title)
```

Maybe some families had more trouble than other to get lifeboats? Next step is to extract the Surname of the passengers and group them to find families. Using this we create a new category called family size.

```
combi$FamilySize <- combi$SibSp + combi$Parch + 1
combi$Surname <- sapply(combi$Name, FUN=function(x) {strsplit(x, split='[,.]')[[1]][1]})
combi$FamilyID <- paste(as.character(combi$FamilySize), combi$Surname, sep="")
combi$FamilyID[combi$FamilySize <= 2] <- 'Small'
table(combi$FamilyID)
```

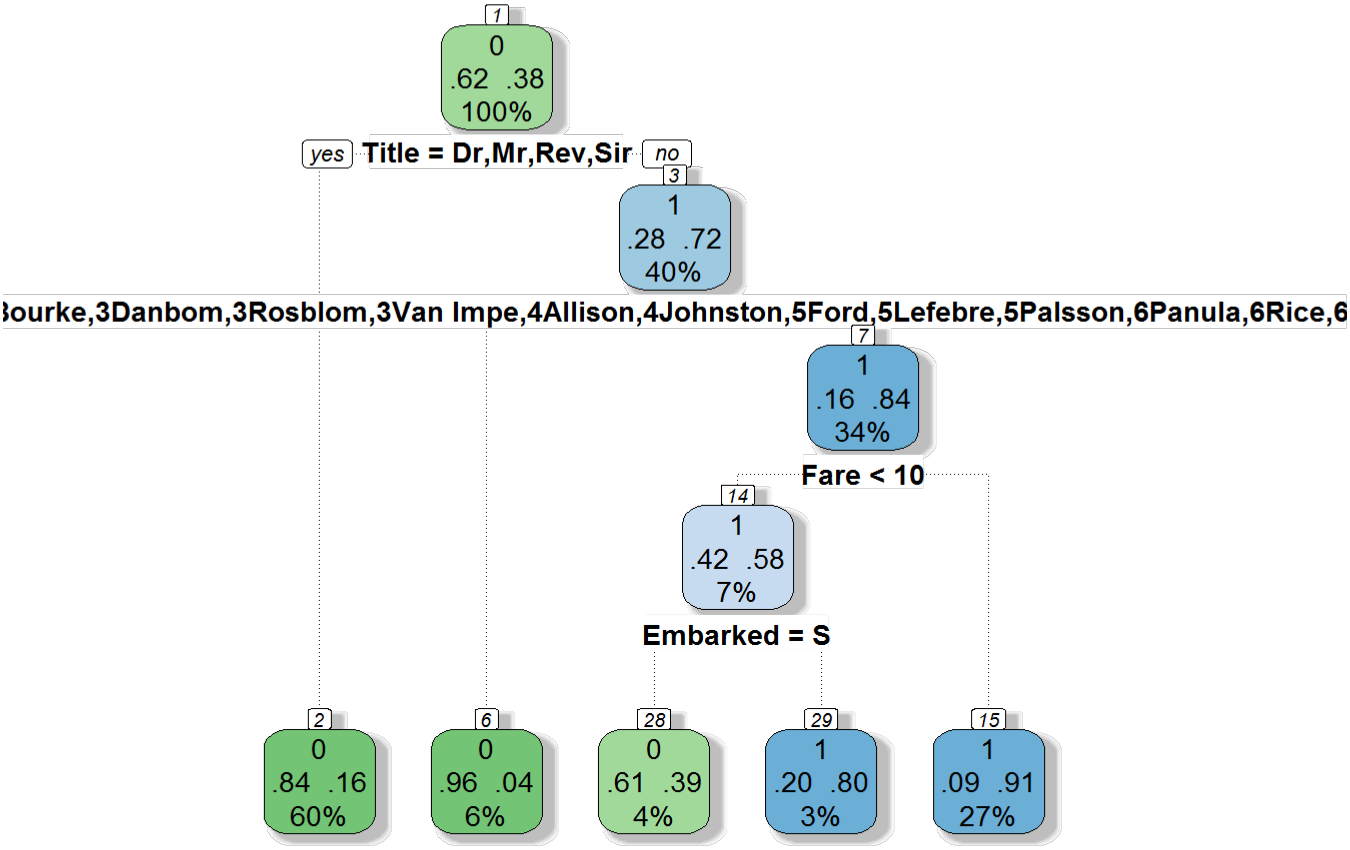
```
##
##           11Sage           3Abbott         3Appleton         3Beckwith
##               11                 3                 1                 2
##          3Boulos           3Bourke            3Brown         3Caldwell
##                3                 3                 4                 3
##         3Christy          3Collyer          3Compton          3Cornell
##                2                 3                 3                 1
##          3Coutts           3Crosby           3Danbom           3Davies
##                3                 3                 3                 5
##           3Dodge          3Douglas             3Drew            3Elias
##                3                 1                 3                 3
##      3Frauenthal        3Frolicher 3Frolicher-Stehli        3Goldsmith
##                1                 1                 2                 3
##      3Gustafsson       3Hamalainen           3Hansen             3Hart
##                2                 2                 1                 3
##            3Hays          3Hickman          3Hiltunen         3Hirvonen
##                2                 3                 1                 1
##        3Jefferys          3Johnson             3Kink    3Kink-Heilmann
##                2                 3                 2                 2
##          3Klasen         3Lahtinen           3Mallet            3McCoy
##                3                 2                 3                 3
##         3Minahan         3Moubarek            3Nakid         3Navratil
##                1                 3                 3                 3
##          3Newell           3Newsom         3Nicholls          3Peacock
##                1                 1                 1                 3
##           3Peter            3Quick         3Richards          3Rosblom
##                3                 3                 2                 3
##          3Samaan        3Sandstrom           3Silven          3Spedden
##                3                 3                 1                 3
##           3Strom          3Taussig           3Thayer           3Thomas
##                1                 3                 3                 1
##           3Touma     3van Billiard         3Van Impe    3Vander Planke
##                3                 3                 3                 2
##           3Wells             3Wick          3Widener          4Allison
##                3                 3                 3                 4
##       4Backstrom          4Baclini           4Becker           4Carter
##                1                 4                 4                 4
##        4Davidson             4Dean           4Herman          4Hocking
##                1                 4                 4                 2
##       4Jacobsohn         4Johnston          4Laroche           4Renouf
##                1                 4                 4                 1
##   4Vander Planke             4West             5Ford          5Hocking
##                1                 4                 5                 1
##   5Kink-Heilmann          5Lefebre          5Palsson          5Ryerson
##                1                 5                 5                 5
##         6Fortune           6Panula             6Rice         6Richards
##                6                 6                 6                 1
##           6Skoog        7Andersson          7Asplund          8Goodwin
##                6                 9                 7                 8
##            Small
##             1025
```

```
famIDs <- data.frame(table(combi$FamilyID))
famIDs <- famIDs[famIDs$Freq <= 2,]
combi$FamilyID[combi$FamilyID %in% famIDs$Var1] <- 'Small'
combi$FamilyID <- factor(combi$FamilyID)
```

We split again the test and train datasets, and do predictions using these new categories. There are many methods to do feature selection.

Let's do our predictions!

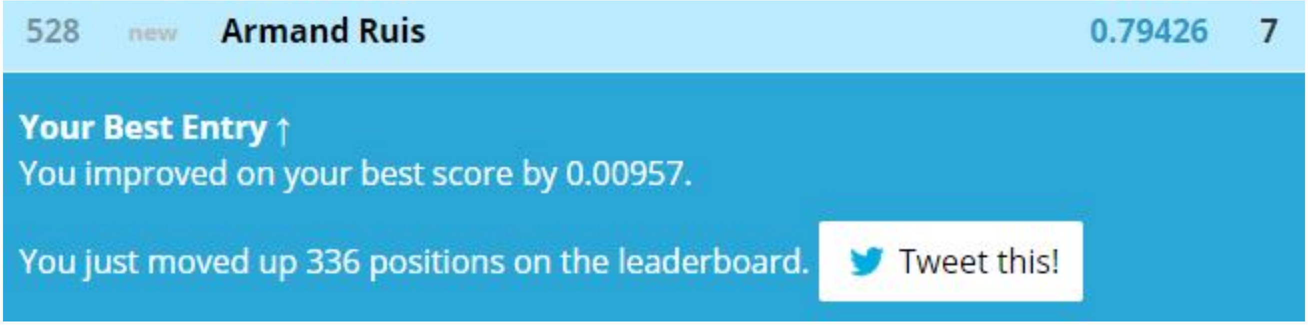
```
train <- combi[1:891,]
test <- combi[892:1309,]
fit <- rpart(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked + Title + FamilySiz
e + FamilyID,
             data=train, method="class")
fancyRpartPlot(fit)
```



Rattle 2015-Mar-27 12:42:04 aruizga7

```
#Submitting
Prediction <- predict(fit, test, type = "class")
submit <- data.frame(PassengerId = test$PassengerId, Survived = Prediction)
write.csv(submit, file = "Prediction5.csv", row.names = FALSE)
```

With feature selection, we improved the performance of Decision Trees by 0.957% and moved up 336 positions!



Prediction 5: Random Forest

Decision trees can have some limitations. We will try to use a different method using the powerful Random Forest algorithm.

```
train <- read.csv("~/GitHub/TitanicShinyApplication/data/train.csv")
test <- read.csv("~/GitHub/TitanicShinyApplication/data/test.csv")
library(rpart)
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(party)
```

```
## Warning: package 'party' was built under R version 3.1.3
```

```
## Loading required package: grid
## Loading required package: mvtnorm
```

```
## Warning: package 'mvtnorm' was built under R version 3.1.3
```

```
## Loading required package: modeltools
```

```
## Warning: package 'modeltools' was built under R version 3.1.3
```

```
## Loading required package: stats4
## Loading required package: strucchange
```

```
## Warning: package 'strucchange' was built under R version 3.1.3
```

```
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package: sandwich
```

```
## Warning: package 'sandwich' was built under R version 3.1.3
```

```r
# Join together the test and train sets for easier feature engineering
test$Survived <- NA
combi <- rbind(train, test)

# Convert to a string
combi$Name <- as.character(combi$Name)

# Engineered variable: Title
combi$Title <- sapply(combi$Name, FUN=function(x) {strsplit(x, split='[,.]')[[1]][2]})
combi$Title <- sub(' ', '', combi$Title)
# Combine small title groups
combi$Title[combi$Title %in% c('Mme', 'Mlle')] <- 'Mlle'
combi$Title[combi$Title %in% c('Capt', 'Don', 'Major', 'Sir')] <- 'Sir'
combi$Title[combi$Title %in% c('Dona', 'Lady', 'the Countess', 'Jonkheer')] <- 'Lady'
# Convert to a factor
combi$Title <- factor(combi$Title)

# Engineered variable: Family size
combi$FamilySize <- combi$SibSp + combi$Parch + 1

# Engineered variable: Family
combi$Surname <- sapply(combi$Name, FUN=function(x) {strsplit(x, split='[,.]')[[1]][1]})
combi$FamilyID <- paste(as.character(combi$FamilySize), combi$Surname, sep="")
combi$FamilyID[combi$FamilySize <= 2] <- 'Small'
# Delete erroneous family IDs
famIDs <- data.frame(table(combi$FamilyID))
famIDs <- famIDs[famIDs$Freq <= 2,]
combi$FamilyID[combi$FamilyID %in% famIDs$Var1] <- 'Small'
# Convert to a factor
combi$FamilyID <- factor(combi$FamilyID)

# Fill in Age NAs
summary(combi$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.17   21.00   28.00   29.88   39.00   80.00     263
```

```
Agefit <- rpart(Age ~ Pclass + Sex + SibSp + Parch + Fare + Embarked + Title + FamilySize,
                data=combi[!is.na(combi$Age),], method="anova")
combi$Age[is.na(combi$Age)] <- predict(Agefit, combi[is.na(combi$Age),])
# Check what else might be missing
summary(combi)
```

```
##   PassengerId      Survived          Pclass          Name
##  Min.   :   1   Min.   :0.0000   Min.   :1.000   Length:1309
##  1st Qu.: 328   1st Qu.:0.0000   1st Qu.:2.000   Class :character
##  Median : 655   Median :0.0000   Median :3.000   Mode  :character
##  Mean   : 655   Mean   :0.3838   Mean   :2.295
##  3rd Qu.: 982   3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :1309   Max.   :1.0000   Max.   :3.000
##                 NA's   :418
##      Sex           Age            SibSp            Parch
##  female:466   Min.   : 0.17   Min.   :0.0000   Min.   :0.000
##  male  :843   1st Qu.:22.00   1st Qu.:0.0000   1st Qu.:0.000
##               Median :28.86   Median :0.0000   Median :0.000
##               Mean   :29.70   Mean   :0.4989   Mean   :0.385
##               3rd Qu.:36.50   3rd Qu.:1.0000   3rd Qu.:0.000
##               Max.   :80.00   Max.   :8.0000   Max.   :9.000
##
##       Ticket          Fare                     Cabin      Embarked
##  CA. 2343:  11   Min.   :  0.000                  :1014    :  2
##  1601    :   8   1st Qu.:  7.896   C23 C25 C27    :   6   C:270
##  CA 2144 :   8   Median : 14.454   B57 B59 B63 B66:   5   Q:123
##  3101295 :   7   Mean   : 33.295   G6             :   5   S:914
##  347077  :   7   3rd Qu.: 31.275   B96 B98        :   4
##  347082  :   7   Max.   :512.329   C22 C26        :   4
##  (Other) :1261   NA's   :1         (Other)        : 271
##      Title       FamilySize       Surname              FamilyID
##  Mr     :757   Min.   : 1.000   Length:1309        Small     :1074
##  Miss   :260   1st Qu.: 1.000   Class :character   11Sage    :  11
##  Mrs    :197   Median : 1.000   Mode  :character   7Andersson:   9
##  Master : 61   Mean   : 1.884                      8Goodwin  :   8
##  Dr     :  8   3rd Qu.: 2.000                      7Asplund  :   7
##  Rev    :  8   Max.   :11.000                      6Fortune  :   6
##  (Other): 18                                       (Other)   : 194
```

```
# Fill in Embarked blanks
summary(combi$Embarked)
```

```
##       C   Q   S
##   2 270 123 914
```

```
which(combi$Embarked == '')
```

```
## [1]  62 830
```

```
combi$Embarked[c(62,830)] = "S"
combi$Embarked <- factor(combi$Embarked)
# Fill in Fare NAs
summary(combi$Fare)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##   0.000   7.896  14.450  33.300  31.280 512.300       1
```

```
which(is.na(combi$Fare))
```

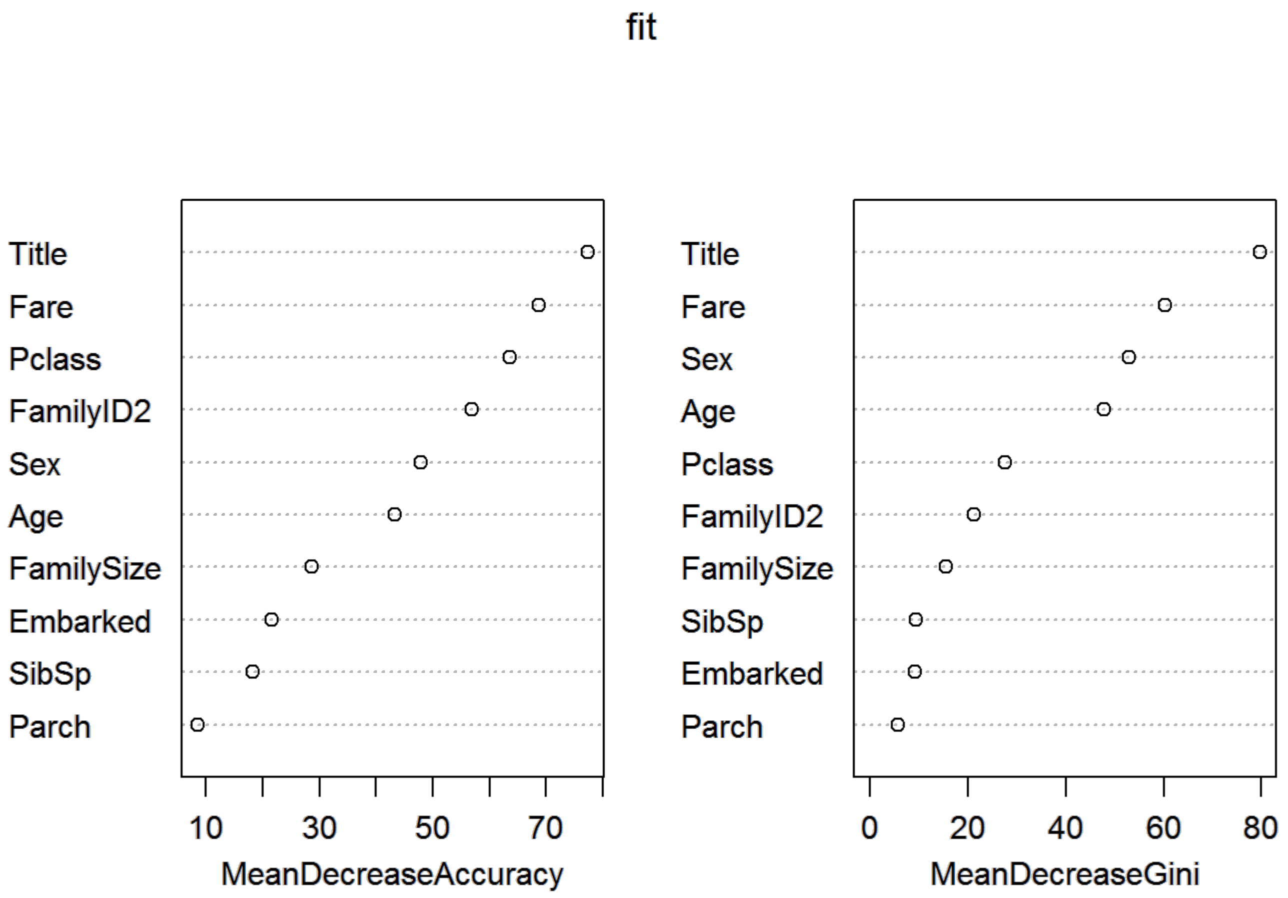
```
## [1] 1044
```

```
combi$Fare[1044] <- median(combi$Fare, na.rm=TRUE)

# New factor for Random Forests, only allowed <32 levels, so reduce number
combi$FamilyID2 <- combi$FamilyID
# Convert back to string
combi$FamilyID2 <- as.character(combi$FamilyID2)
combi$FamilyID2[combi$FamilySize <= 3] <- 'Small'
# And convert back to factor
combi$FamilyID2 <- factor(combi$FamilyID2)

# Split back into test and train sets
train <- combi[1:891,]
test <- combi[892:1309,]

# Build Random Forest Ensemble
set.seed(415)
fit <- randomForest(as.factor(Survived) ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked +
 Title + FamilySize + FamilyID2,
                    data=train, importance=TRUE, ntree=2000)
# Look at variable importance
varImpPlot(fit)
```




```
# Now let's make a prediction and write a submission file
Prediction <- predict(fit, test)
submit <- data.frame(PassengerId = test$PassengerId, Survived = Prediction)
write.csv(submit, file = "firstforest.csv", row.names = FALSE)

# Build condition inference tree Random Forest
set.seed(415)
fit <- cforest(as.factor(Survived) ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked + Titl
e + FamilySize + FamilyID,
               data = train, controls=cforest_unbiased(ntree=2000, mtry=3))
# Now let's make a prediction and write a submission file
Prediction <- predict(fit, test, OOB=TRUE, type = "response")
submit <- data.frame(PassengerId = test$PassengerId, Survived = Prediction)
write.csv(submit, file = "Prediction6.csv", row.names = FALSE)
```