# Titanic: Data Analysis

Victor Bernal Arzola

May 2, 2016

victor.bernal@mathmods.eu

## Introduction

Data analysis is a process for obtaining raw data and converting it into information useful for decision-making by users. Data is collected and analyzed to answer questions, test hypotheses or disprove theories.[1]. Its a process of inspecting, cleaning, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making. In statistical applications, some people divide data analysis into descriptive statistics, exploratory data analysis (EDA), and confirmatory data analysis (CDA). EDA focuses on discovering new features in the data and CDA on confirming or falsifying existing hypotheses. Predictive analytics focuses on application of statistical models for predictive forecasting or classification. All are varieties of data analysis.

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 https://es.wikipedia.org/wiki/RMS_Titanic#cite_note-nota-5. 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. After leaving Southampton on 10 April 1912, Titanic called at Cherbourg in France and Queenstown in Ireland before heading west to New York.

In this project, we want to complete the analysis of what sorts of people were likely to survive. In particular, we would like to apply the tools of machine learning to predict which passengers survived the tragedy. We will use the data provided at https://www.kaggle.com/c/titanic/data where the reader can find also a full description of the .csv data set.

## 1 Load and Check the Data

We begin by loading the data to the *R* environment. We have a training set consisting in 891 passengers each one with 11 features, and for the test set 418 passengers with 10 attributes (the survived row is missing). In this way we proceed to add the missing column to the test data and combine both data set.

```
# Load packages
library(ggplot2)
library(rpart)
library(rpart.plot)
library('randomForest')

# Load raw train and test data
train <- read.csv("train.csv", header = TRUE)
test <- read.csv("test.csv", header = TRUE)

# Add a "Survived" variable to the test set to allow for combining data sets
test.survived <- data.frame(survived = rep("None", nrow(test)), test[,])

# Combine data sets
data.combined <- rbind(train, test.survived)
```

Now we confirm if the are any missing values in the features. This is a very important step because *NA* values affect the analysis of the data, dealing with them could be delicate.

```
# No class missing
table(is.na(data.combined$pclass))
table(is.nan(data.combined$pclass))

# No sex missing
table(is.na(data.combined$sex))
table(is.nan(data.combined$sex))

## 263 ages missing (NA not available)
table(is.na(data.combined$age))
table(is.nan(data.combined$age))

#No name missing (2 repeated names but different persons)
table(is.na(data.combined$name))
length(unique(as.character(data.combined$name)))-
length((as.character(data.combined$name)))

# One 60.5 years old man no fare, embarked by S in 3rd class
data.combined[which(is.na(data.combined$fare)),]

# 2 People with no embarked place survived
train[(which((train$embarked)=="")),]
unique(as.character(train$embarked))
```

We have that

- No class, name or sex are missing.
- 263 ages are missing.
- There is 1 missing Fare who embarked at Southampton in 3rd class.
- There are 2 people with no embarked place.

Because there is a high number of missing values for the age, we will not involve this feature by now. We can observe in Fig.1.1 and Fig.1.2 that *Sex* and *Class* are an important predictors for survival. A more clear view is shown in Fig.1.3.

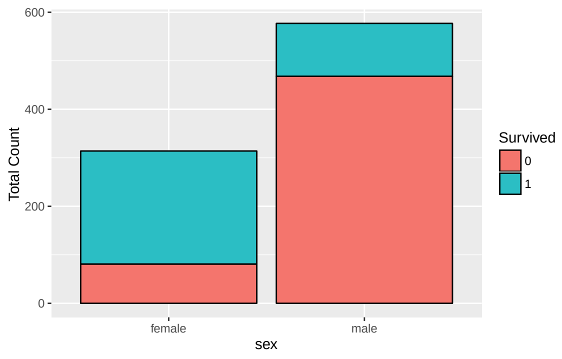

Figure 1.1: Survival Rate by Sex

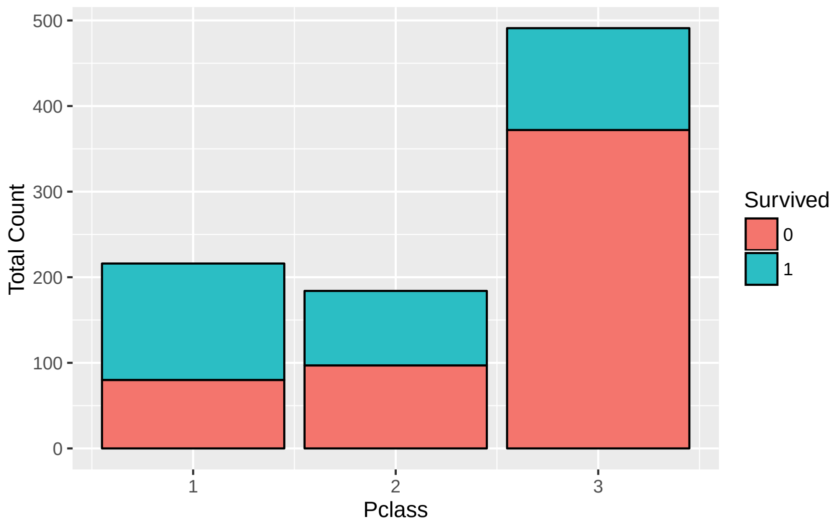

Figure 1.2: Survival Rate per Class

| | **Description** | **Correct** |
|---|---|---|
| **Prior Probability** | Predicting everybody died (larger label) | 61.6 % |
| **Maximum Likelihood** | Predicting that being dead you are man / alive women | 81.11 % / 74.20 % |
| **Bayesian a Posterior** | Predicting that being men you died / alive women | 77.17 % / 80.82 % |

Table 1: Simple Statistical Predictors of Survival Rates

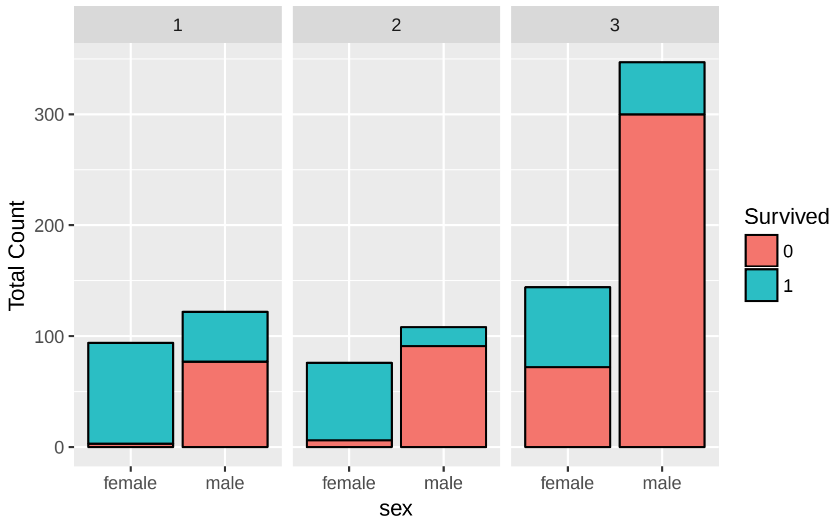

Figure 1.3: Survival Rate per Class per Sex

## 2 Features Engineering

Feature Engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Now that we see that the variable *Sex* and *Class* is important we can break down the variable *Passenger Name* into additional meaningful variables which can be used in the creation of additional new variables. For instance, *Passenger Title* can be separated and investigated. and we can reassign equivalent titles in other languages and rare titles to be gathered in one label.

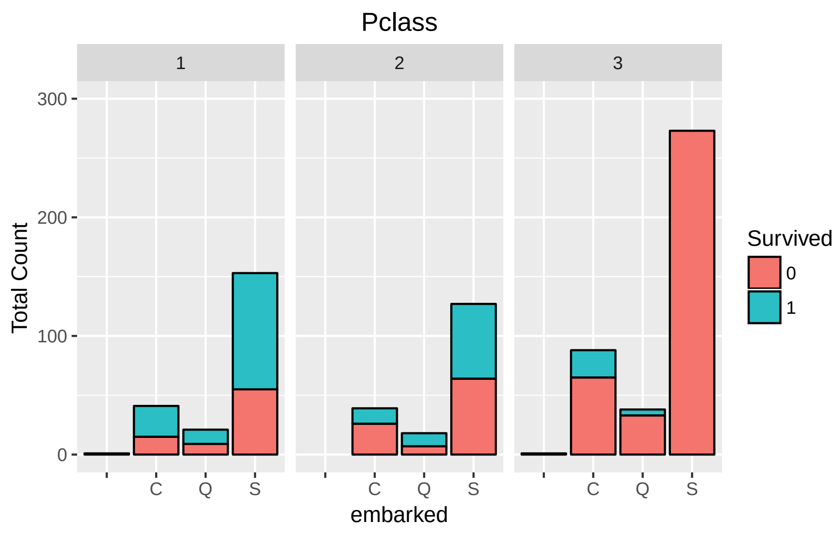

Figure 2.1: Survival Rate per Embarked place and Class
We can observe that all $3rd$ class passengers that boarded at Southampton died.

```
data.combined$title <- gsub('(.*, )|(\\..*)', '', data.combined$name)
data.combined(data.combined$sex, data.combined$title)
rare.titles<-c('Capt', 'Col', 'Don', 'Dr', 'Major', 'Rev',
'Sir', 'Jonkheer','Dona', 'the Countess')

# Also reassign Lady, Mlle, Ms, and Mme accordingly
data.combined$title[which(data.combined$title == 'Mlle')] <- 'Miss'
data.combined$title[which(data.combined$title == 'Ms')] <- 'Miss'
data.combined$title[which(data.combined$title == 'Mme')]  <- 'Mrs'
data.combined$title[which(data.combined$title == 'Lady')] <- 'Miss'
data.combined$title[data.combined$title %in% rare.titles]  <- 'Rare title'

#And check if they were assigned correctly according with the sex
table(data.combined$sex, data.combined$title)
```

In this way we have separated the titles and gathered them in a subclass consisting of *Mr. Mr.s Master.* and *Ms.* We accomplished this by replacing French title by their English equivalencies and we collected other unusual titles in the data as Dr., Rev., Don. Major, Capt., Sir, Jonkheer, Donna and Countess in the category of *Rare Titles.* Now we can investigate the survival rate according to *Titles* together with *Class.*

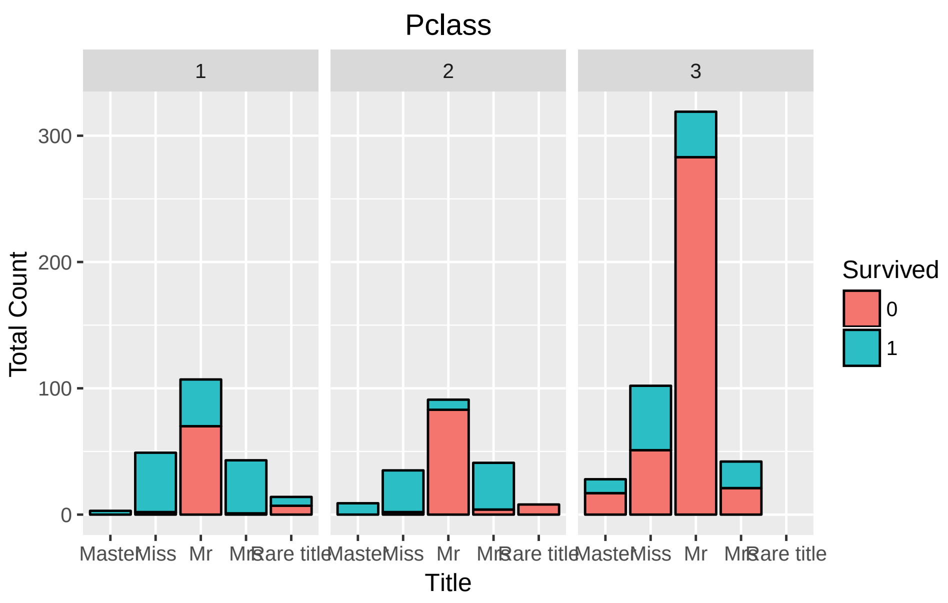

Figure 2.2: Titles Distribution per Class

From 2.2 it is apparent that *Titles* per *Class* are good predictor as they encode *Sex, Age* and *Class*. We can also create a *Family Size* feature by adding *Sibsp, Parch* plus a unit (the passenger himself). It becomes apparent in Fig. 2.3 that there is a **penalty over single passengers and families bigger than four**.

```
# Combine sinsp and parch
data.combined$family.size <- as.factor(data.combined$sibsp+ data.combined$parch + 1)

#Check that all family size members are numbers
table(is.na(data.combined$family.size))
```

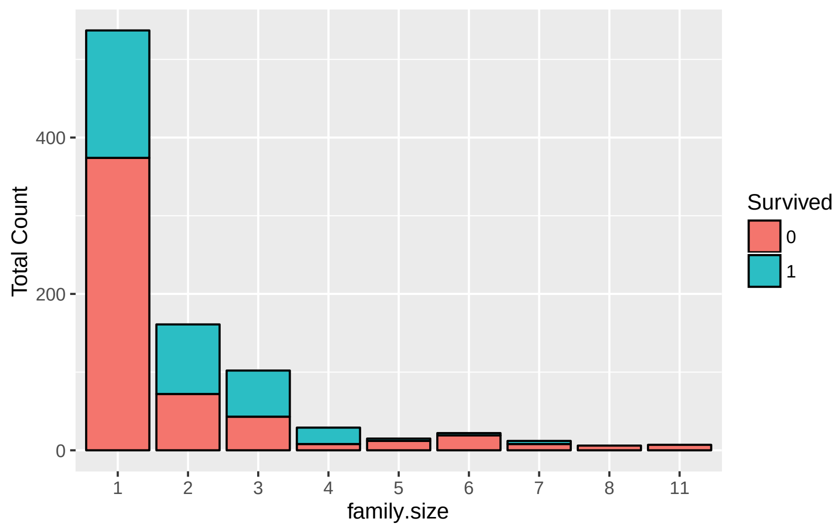

Figure 2.3: Survival Rate per Family Size

## 3 Missing Data

There are a number of different ways we could proceed about dealing with missing data. Given the small size of the data set, we probably should not go for deleting entire observations containing missing values. We are left with the option of either replacing missing values with a sensible values given the distribution of the data, e.g., the mean, median or mode. In the case of the *Fare* we can observe in Fig. 3.1 that people with No Embarked gate *Survived*, they all paid a *Fare* of 80 Dollars and were 1rst *Class* passengers. This price almost matches the median ticket fare for 1rst class embarked in *Cherbourg* as shown in Fig. 3.1, and its out of two standard deviation for other classes. In this way we proceed to assign it as embarked at C. In the same way a passenger which embarked in *Southampton* in 3rd *Class* whose *Fare* is missing. We can assign the median of the *Fare* value for a 3rd *Class* ticket and *Embarked* place *S* as seen in Fig. 3.2.

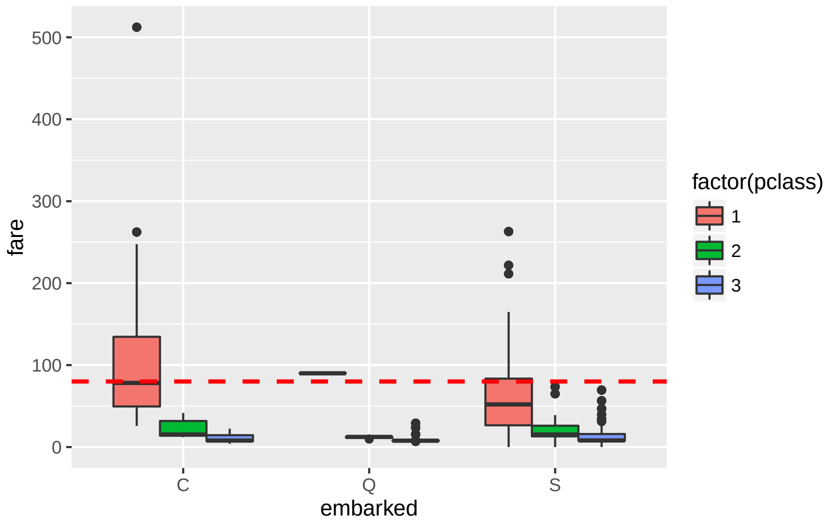

Figure 3.1: Fare by Embarked Gate

```
# Missing Gate
# Assign gate C, people with no embarked gate paid 80 dolars
train[(which((train$embarked)=="")),]
data.combined[which(data.combined$embarked==''),'embarked'] <-as.factor('C')

# One 60.5 years old man no fare, embarked by S in 3rd class
# Assign the median of 3rd class gate S
data.combined[which(is.na(data.combined$fare)),]
table(data.combined$embarked)
hist(data.combined[which(data.combined$pclass == '3' & data.combined$embarked == 'S')
,'fare'],+ breaks=10, col="blue",  main="Fare␣for␣3rd␣class␣Gate␣S",  xlab="Fare")
median.fare  <-
median(na.omit(data.combined[which(data.combined$pclass == '3' & data.combined$embarked =
```

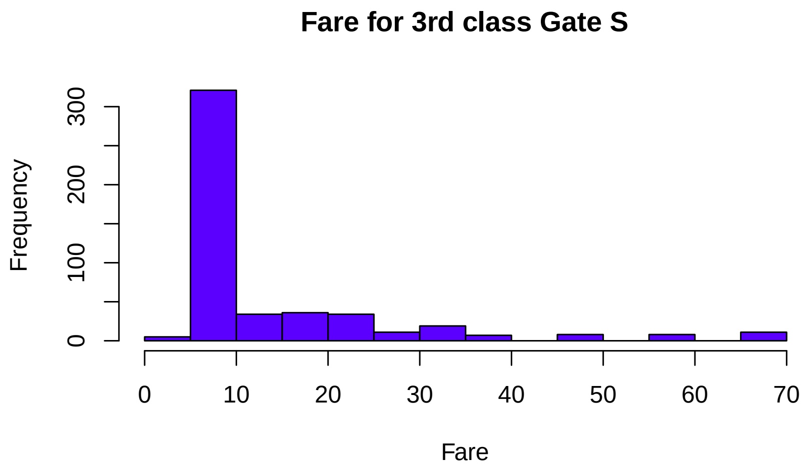

Figure 3.2: 3rd Class Fare Distribution for Southampton Passengers

```
# How to fill in missing Age values?
# This time you give method="anova" since you are predicting a continuous variable.
set.seed(1234)
predicted_age <- rpart(age ~ pclass+fare+age,
data=data.combined[!is.na(data.combined$age),], method="anova")

# Reassign ages
data.combined[which(is.na(data.combined$age)),"age"]
<- predict(predicted_age, data.combined[which(is.na(data.combined$age)),])
```

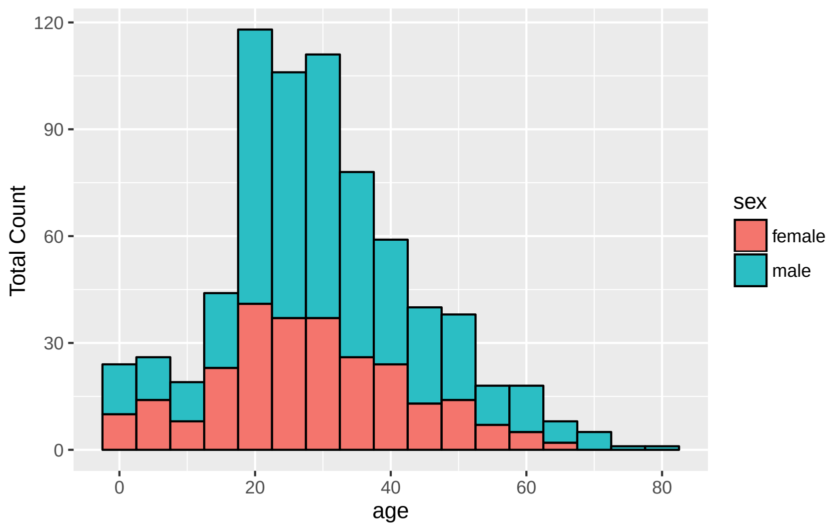

Figure 3.3: Survival Rate per Age Distribution

## 4 Decision Tree

Tree models where the target variable can take a finite set of values are called classification trees. The goal is to create a model that predicts the value of a target variable based on several input variables. Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf. The arcs coming from a node labeled with a feature are labeled with each of the possible values of the feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes. A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions. This process of top-down induction of decision trees is an example of a greedy algorithm, and it is by far the most common strategy for learning decision trees from data. A greedy algorithm is an algorithm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum. In many problems, a greedy strategy does not in general produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a global optimal solution in a reasonable time.

```
# Make explicit factor levels for specific variables: Sex + Pclass
train$sex <- as.factor(train$sex)
test$pclass <- as.factor(test$pclass)
library(rpart)
set.seed(1234)

# Train on entire training set for Title Class Family.size
fol <- formula(survived~title+pclass+family.size)
model<-rpart(fol,method="class",data=data.combined[1:891,],parms=list(split="gini"))

library(rattle)
fancyRpartPlot(model)
```

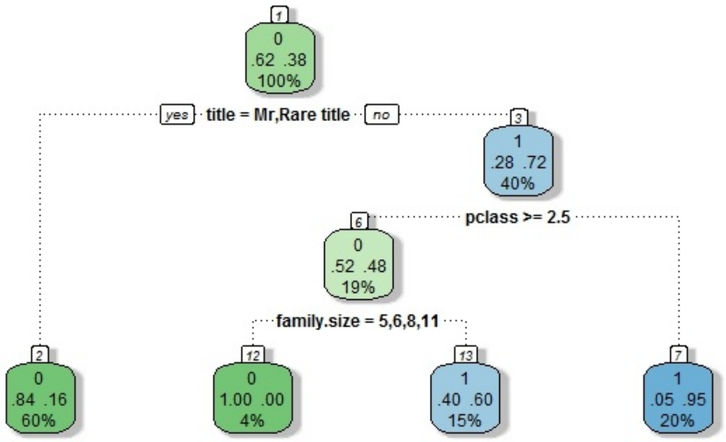

Figure 4.1: Decision Tree Model on Training Data

| | **Died** | **Lived** |
|---|---|---|
| **Train Data** | 549 | 342 |
| **Prediction on Train Data** | 573 | 318 |

Table 2: Decision Tree Training Error

## 5 Random Forest

Random forests is a notion of the general technique of random decision forests[2] [3]that 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. Random decision forests correct for decision trees' habit of over fitting to their training set.

In particular, trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, because they have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance of the final model.[4].Random forests can be used to rank the importance of variables in a regression or classification problem in a natural way.

```
 # Random forest
library('randomForest') # classification algorithm
# Make as factor
data.combined$title<-as.factor(data.combined$title)
data.combined$age<-as.factor(data.combined$age)
# Train a Random Forest with the parameters pclass, title, family.size
rf.train.5 <- data.combined[1:891, c("pclass", "title", "family.size")]
rf.label <- as.factor(train$survived)
set.seed(1234)
rf.5 <- randomForest(x = rf.train.5, y = rf.label, importance = TRUE, ntree = 1000)
rf.5 varImpPlot(rf.5)

#         OOB estimate of   error rate: 17.17%
# Confusion matrix:
#      0    1   class.error
#   0 490   59    0.1074681
#   1  94  248    0.2748538
```

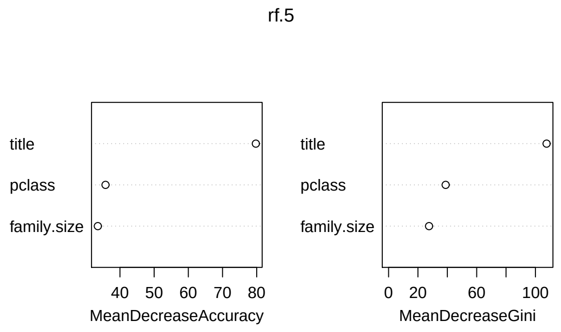

Figure 5.1: Variable Importance

In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k − 1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once. 10-fold cross-validation is commonly used, but in general k remains an unfixed parameter.

```
# 10 fold Cross Validation
set.seed(2348)
cv.10.folds <- createMultiFolds(rf.label, k = 10, times = 10)
# Set up caret's trainControl object per above.
ctrl.1 <- trainControl(method = "repeatedcv", number = 10,
repeats = 10, index = cv.10.folds)
set.seed(34324)
rf.5.cv.1 <- train(x = rf.train.5, y = rf.label, method = "rf",
tuneLength = 3, ntree = 1000, trControl = ctrl.1)
# Check out results rf.5.cv.1


#mtry   Accuracy    Kappa
#2      0.8134761   0.5955706
#3      0.8085446   0.5836802
```

## References

[1] Charles M Judd, Gary H McClelland, and Carey S Ryan. *Data analysis: A model comparison approach.* Routledge, 2011.

[2] Tin Kam Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.

[3] Tin Kam Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):832–844, 1998.

[4] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.