Titanic Survival Analysis

Jed Aureus Gonzales

Friday, 21 November, 2014

Titanic Survival Analysis

This writeup includes detailed and exploratory data analysis that serves as an accompaniment to the **Shiny** application, *Would you have the voyage of the RMS Titanic?*. This study has been conducted as the course project for Johns Hopkins University's Developing Data Products course in Coursera. The problem was inspired by the Kaggle competition, Titanic: Machine Learning from Disaster.

Background

According to its Wikipedia entry, the RMS Titanic set sail on its maiden voyage from Southampton, UK to New York City, USA on the 11th of April 1912 and met its untimely end on the 15th of April 1912 at 11:40 PM, colliding with an iceberg and killing 1502 out of 2224 passengers and crew.

Queenstown
11 April 1912
Southampton
10 April 1912
Cherbourg
10 April 1912
New York
41°43.5'N 49°56.8'W
15 April 1912

The ship could accomodate 2566 passengers (1034 First Class, 510 Second Class, and 1022 Third Class) and was designed to carry 32 lifeboats. However, cost-cutting measures have reduced the number of lifeboats to 20 (enough for 1180 people) for its maiden voyage. Another likely reason for the removal of the lifeboats was the notion that the RMS Titanic was "unsinkable."

Assumptions

The 1912 was a time of progress and great divide; gender, race and social status were cause for most of the segregation. Without looking at the data, it is safe to assume that these notions could factor in to the outcome of the analysis. Considering the women and children first protocol was enacted, it is safe to assume that majority of survivors would have been a woman and/or a child. People of caucasian descent would have a greater chance of survival and people who belong to the upper class would have easier access to the upper floors of the cabins, therefore having easier access to the lifeboats. Taking into account the time of the accident, it is also safe to assume that most of the passengers had likely retired to their cabins.

Getting and Cleaning the Data

First and foremost, load all the necessary libraries and set the seed to ensure reproducibility.

```
library(caret)
library(Hmisc)
library(gridExtra)
```

After loading all the necessary libraries, download the training and testing data from their respective links. Define the column and missing types for faster processing.

```
if (!file.exists("train.csv")) {
    download.file(url = "https://www.kaggle.com/c/titanic-gettingStarted/download/train.csv",
                  destfile = "train.csv",
                  method = "curl")
}
if (!file.exists("test.csv")) {
    download.file(url = "https://www.kaggle.com/c/titanic-gettingStarted/download/test.csv",
                  destfile = "test.csv",
                  method = "curl")
}

train.colClasses <- c('integer',   # PassengerId
                        'factor',    # Survived
                        'factor',    # Pclass
                        'character', # Name
                        'factor',    # Sex
                        'numeric',   # Age
                        'integer',   # SibSp
                        'integer',   # Parch
                        'character', # Ticket
                        'numeric',   # Fare
                        'character', # Cabin
                        'factor'     # Embarked
)
test.colClasses <- train.colClasses[-2]

train.df <- read.csv("train.csv", header = TRUE,
                     na.strings = c("NA", ""), colClasses = train.colClasses)
test.df <- read.csv("test.csv", header = TRUE,
                     na.strings = c("NA", ""), colClasses = test.colClasses)
```

A quick `summary` on `train.df` will verify that `Age` has 177 `NA` values. Further analysis of `Name`, with comparison to `Age`, would suggest that the passengers' "title" varies with age. For this study, the missing `Age` values will be substituted by the average of the "title" they belong to. To aid this process, the creation of a formal `Title` variable in `train.df` is required.

```
title.upper <- regexpr("\\,[A-Z ]{1,20}\\.", train.df$Name, TRUE)
title.lower <- title.upper + attr(title.upper, "match.length") - 1
train.df$Title <- substr(train.df$Name, title.upper + 2, title.lower - 1)
unique(train.df$Title)
```

```
##  [1] "Mr"           "Mrs"          "Miss"         "Master"
##  [5] "Don"          "Rev"          "Dr"           "Mme"
##  [9] "Ms"           "Major"        "Lady"         "Sir"
## [13] "Mlle"         "Col"          "Capt"         "the Countess"
## [17] "Jonkheer"
```

Use the `bystats` function from the `Hmisc` package to identify the observations with missing `Age` values and their respective mean and median.

```
bystats(train.df$Age, train.df$Title,
        fun = function(x) c(Mean = mean(x), Median = median(x)))
```

```
##
##  c(2, 15, 2, 63, 15, 63, 2, 2) of train.df$Age by train.df$Title
##
##                N Missing   Mean Median
## Capt           1       0 70.000   70.0
## Col            2       0 58.000   58.0
## Don            1       0 40.000   40.0
## Dr             6       1 42.000   46.5
## Jonkheer       1       0 38.000   38.0
## Lady           1       0 48.000   48.0
## Major          2       0 48.500   48.5
## Master        36       4  4.574    3.5
## Miss         146      36 21.774   21.0
## Mlle           2       0 24.000   24.0
## Mme            1       0 24.000   24.0
## Mr           398     119 32.368   30.0
## Mrs          108      17 35.898   35.0
## Ms             1       0 28.000   28.0
## Rev            6       0 43.167   46.5
## Sir            1       0 49.000   49.0
## the Countess   1       0 33.000   33.0
## ALL          714     177 29.699   28.0
```

From the result, it could be noted that the `Title` values `Dr`, `Master`, `Miss`, `Mr` and `Mrs` have missing values. Fill in the missing values by imputing the median for their respective title. Remove the estimates by rounding the `Age` values to `0`.

```
missing.titles <- c("Dr", "Master", "Miss", "Mr", "Mrs")
for(title in missing.titles) {
    train.df$Age[which(train.df$Title == title)] <- round(impute(train.df$Age[which(train.df$Title == t
}
```

Notice, also, that `Embarked` also has 2 `NA` values. Since most of the passengers had embarked from Southampton (`S`), it is safe to assume that it is where the two passengers embarked there as well.

```
train.df$Embarked[which(is.na(train.df$Embarked))] <- 'S'
```

As an aside, recall the motion picture *Titanic*, Leonardo diCaprio's character and his friend checked in at the very last minute. One might wonder if their tickets have not been verified as having embarked from Southampton. But, that is a fictional curiosity for some other time.

Perform the same exact procedures for the test data so that there wouldn't be any conflict when predicting later. The same replication of procedures must be done for subsequent changes as well.

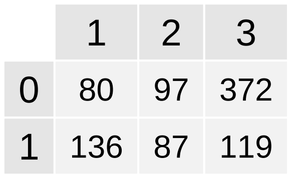
Now that the `NA` values have been removed and the data set, exploratory data analysis could now be conducted.

Exploratory Data Analysis

Before features could be selected for predictions, exploratory data analysis should first be conducted on the training set. If necessary, additional features would be created by the end of this section for better prediction.

First, test the earlier assumptions on the processed data. For the process of elimination, the `Survived` variable will be used as the outcome since it is the only variable missing in the test data.

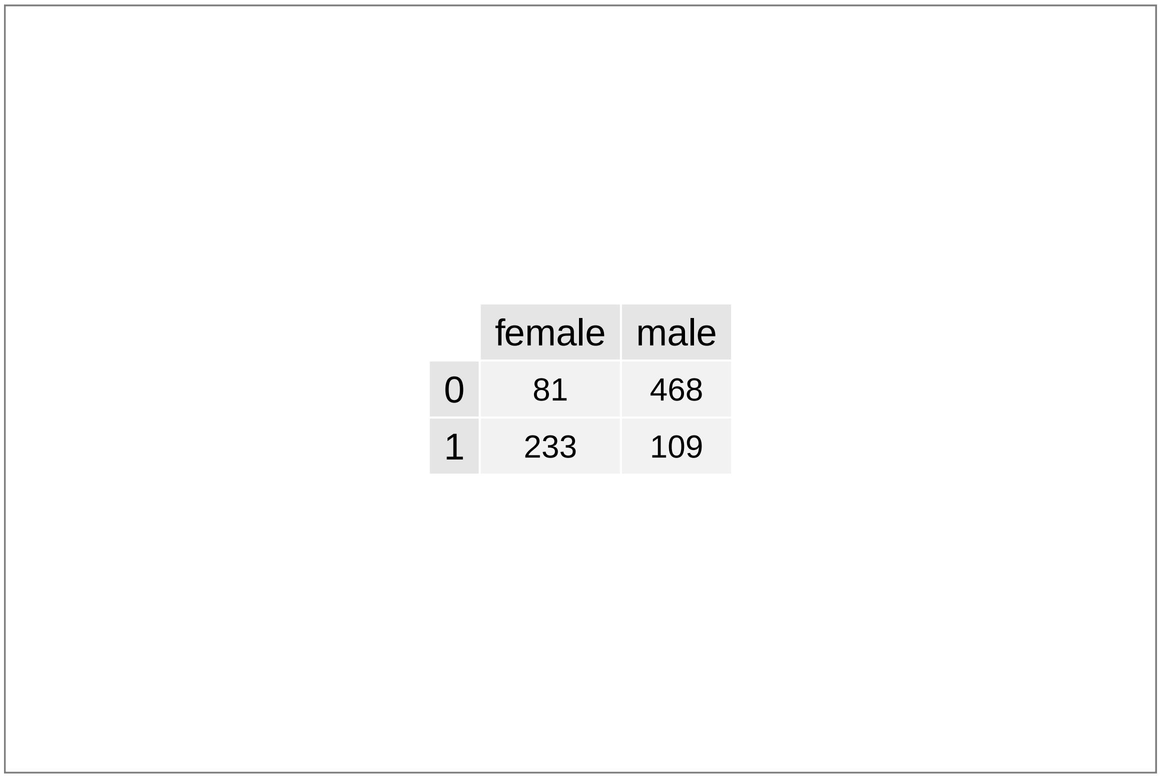
Plotting `Pclass` against `Survived` yields the following table (Passenger class corresponds to first (`1`), second (`2`) and third (`3`), `0` if did **not** survive):



	1	2	3
0	80	97	372
1	136	87	119

Even though the percentage of third class passenger is the lowest, the number of third class passengers who survived outnumbered that of the second class passengers. This is a very interesting feature, considering the class segregation at the time.

Plotting `Sex` against `Survived` yields the following table:

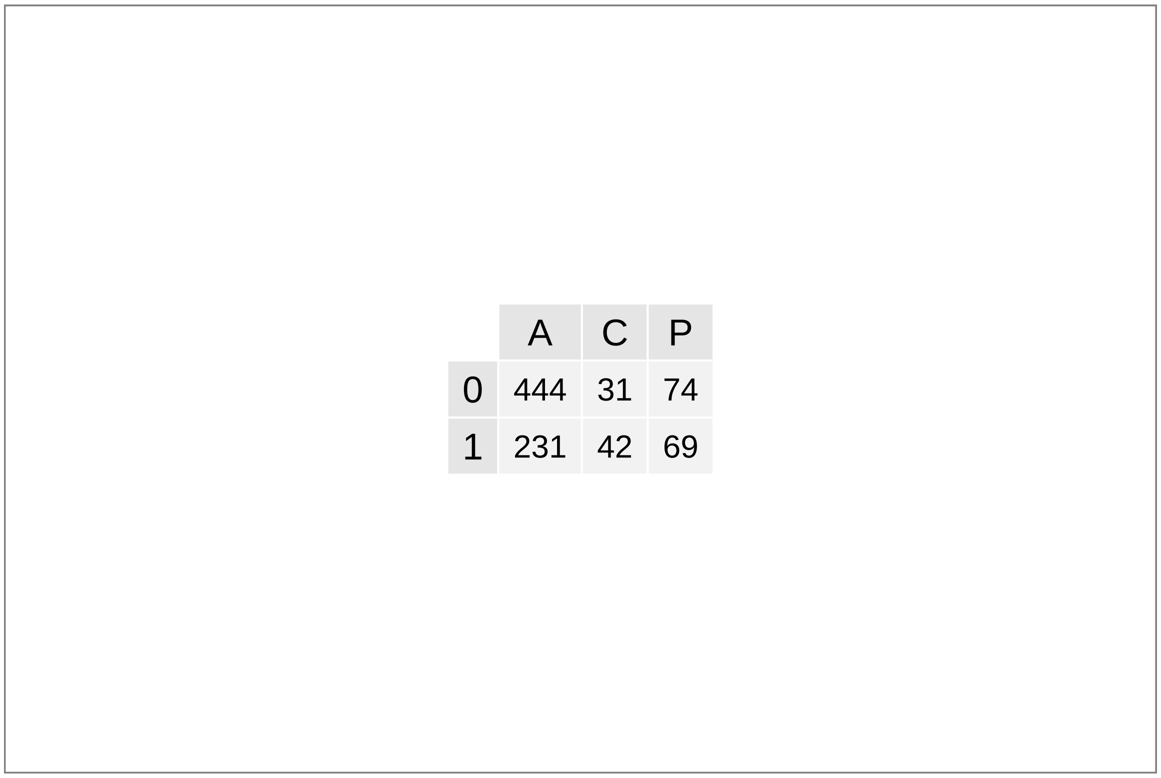


	female	male
0	81	468
1	233	109

Based from the table, the earlier assumption of *women first* could already be confirmed. Later on, `Age` and `Sex` will be factored together. Before that, look at the table produced by the `Age` against `Survived`.

This is where it gets tricky to visualize. An important aspect of the `Age` variable is that it could be subdivided further into factors. Taking `Title` into consideration, it could be observed that passengers up to the age of `12` is considered as a child. Taking `Parch` into consideration, passengers could further be classified as a parent. Blanket the considerations to a variable `PAC`, based on Transactional analysis.

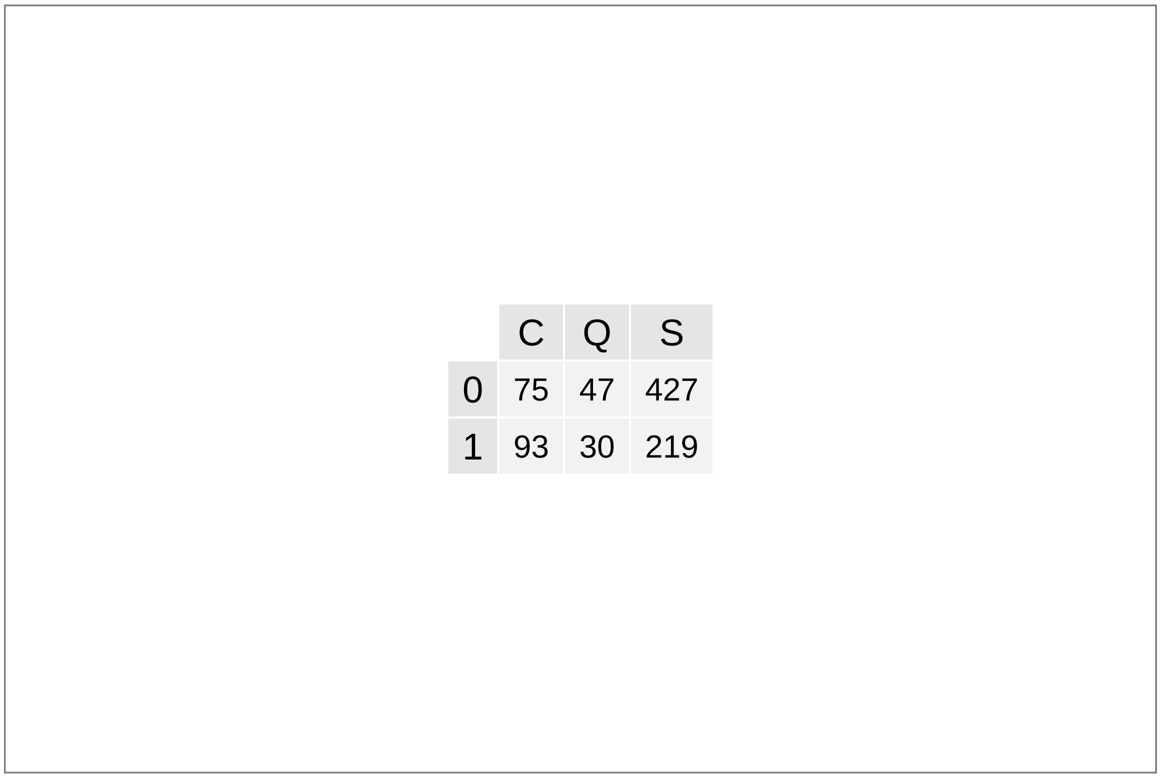
```
train.df$PAC <- ifelse(train.df$Age <= 12, "C", ifelse(train.df$Parch > 0, "P", "A"))
test.df$PAC <- ifelse(test.df$Age <= 12, "C", ifelse(train.df$Parch > 0, "P", "A"))
```



	A	C	P
0	444	31	74
1	231	42	69

Based on the table, being a child (`C`) did not ensure survival. The uneven number of parents (`P`) and children could also mean that some passengers were parents of adults (`A`).

Plotting `Embarked` against `Survived` yields the following table:



	C	Q	S
0	75	47	427
1	93	30	219

The `Embarked` variable doesn't really show much variance but will still be considered for feature selection.

The other variables, which mostly had the string data type, will most likely not be used for fitting the model.

Model Fitting

Split the training set further into training (60%) and validation sets (40%), using `Survived` as the outcome. Remove the variables `PassengerId`, `Name`, `Ticket`, `Fare`, `Cabin` and `Title` as they have been analyzed as superficial.

```r
set.seed(1337)
features <- c("Survived", "Pclass", "Sex", "Age", "SibSp", "Parch", "Embarked", "PAC")
data <- train.df[features]
inTrain <- createDataPartition(data$Survived, p=.60, list=FALSE)
training <- data[inTrain,]
validation <- data[-inTrain,]
rownames(training) <- NULL
rownames(validation) <- NULL
```

Fit the data into different models to see which one yields the most accuracy.

Using `glm` with all the remaining features:

```r
model.glm.1 <- train(Survived ~ ., family = binomial,
                     data = training, method = "glm")
summary(model.glm.1)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.238  -0.537  -0.407   0.610   2.351
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)    3.918      0.600    6.53  6.7e-11 ***
## Pclass2       -1.425      0.366   -3.89  0.00010 ***
## Pclass3       -2.328      0.333   -7.00  2.5e-12 ***
## Sexmale       -2.775      0.268  -10.37  < 2e-16 ***
## Age           -0.030      0.012   -2.49  0.01274 *
## SibSp         -0.507      0.161   -3.16  0.00160 **
## Parch         -0.538      0.289   -1.86  0.06270 .
## EmbarkedQ     -0.366      0.509   -0.72  0.47227
## EmbarkedS     -0.165      0.313   -0.53  0.59765
## PACC           2.306      0.682    3.38  0.00073 ***
## PACP           0.813      0.587    1.39  0.16598
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 714.11  on 535  degrees of freedom
## Residual deviance: 456.63  on 525  degrees of freedom
## AIC: 478.6
##
## Number of Fisher Scoring iterations: 5
```

Notice that `Pclass` and `Sex` had the most significant influence, which gives credence to the earlier assumption about social class and gender. `Age` and `SibSp` follows closely, the former supporting the child hypothesis and the latter emerging somewhat as a surprising feature. The newly created `PAC` follows close behind. `Embarked` and `Parch` scored the lowest and can either be dropped or modified.

Modify the predictors to fit earlier assumptions and test it on `glm`, continue until the fit is optimal:

```
model.glm.2 <- train(Survived ~ Pclass + SibSp + I(PAC == "C")
                   + I(Sex == "male") + I(Sex == "female" & Parch > 0)
                   + I(Embarked == "S"),
                   family = binomial, data = training, method = "glm")
summary(model.glm.2)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.412  -0.583  -0.439   0.622   2.364
##
## Coefficients:
```

```
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                           2.852      0.361    7.90  2.9e-15
## Pclass2                              -1.175      0.346   -3.40  0.00068
## Pclass3                              -2.085      0.287   -7.26  3.8e-13
## SibSp                                -0.440      0.150   -2.93  0.00338
## `I(PAC == "C")TRUE`                   2.430      0.490    4.96  7.0e-07
## `I(Sex == "male")TRUE`               -2.925      0.292  -10.02  < 2e-16
## `I(Sex == "female" & Parch > 0)TRUE` -0.528      0.383   -1.38  0.16828
## `I(Embarked == "S")TRUE`             -0.132      0.265   -0.50  0.61835
##
## (Intercept)                        ***
## Pclass2                            ***
## Pclass3                            ***
## SibSp                              **
## `I(PAC == "C")TRUE`                ***
## `I(Sex == "male")TRUE`             ***
## `I(Sex == "female" & Parch > 0)TRUE`
## `I(Embarked == "S")TRUE`
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 714.11  on 535  degrees of freedom
## Residual deviance: 467.77  on 528  degrees of freedom
## AIC: 483.8
##
## Number of Fisher Scoring iterations: 5
```

```
model.glm.3 <- train(Survived ~ Pclass * Sex + SibSp + I(PAC == "C")
                   + I(Sex == "male") * Parch,
                   family = binomial, data = training, method = "glm")
```

```
summary(model.glm.3)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.828  -0.541  -0.466   0.370   2.234
##
## Coefficients: (1 not defined because of singularities)
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     3.980      0.759    5.25  1.6e-07 ***
## Pclass2                        -1.334      0.905   -1.47   0.1406
## Pclass3                        -3.657      0.771   -4.74  2.1e-06 ***
## Sexmale                        -4.380      0.797   -5.50  3.9e-08 ***
## SibSp                          -0.498      0.165   -3.01   0.0026 **
## `I(PAC == "C")TRUE`             2.444      0.504    4.85  1.2e-06 ***
## `I(Sex == "male")TRUE`             NA         NA      NA       NA
## Parch                          -0.371      0.195   -1.90   0.0575 .
## `Pclass2:Sexmale`              -0.430      1.033   -0.42   0.6770
## `Pclass3:Sexmale`               2.209      0.834    2.65   0.0081 **
## `I(Sex == "male")TRUE:Parch`    0.481      0.307    1.57   0.1167
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 714.11  on 535  degrees of freedom
## Residual deviance: 444.36  on 526  degrees of freedom
## AIC: 464.4
##
## Number of Fisher Scoring iterations: 6
```

```
model.glm.4 <- train(Survived ~ Pclass * Sex + I(PAC == "C"),
                     family = binomial, data = training, method = "glm")
summary(model.glm.4)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.608  -0.512  -0.512   0.460   2.139
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)            3.367      0.719    4.68  2.8e-06 ***
## Pclass2               -1.175      0.891   -1.32   0.1873
## Pclass3               -3.610      0.752   -4.80  1.6e-06 ***
## Sexmale               -3.872      0.762   -5.08  3.8e-07 ***
## `I(PAC == "C")TRUE`    1.688      0.395    4.27  1.9e-05 ***
## `Pclass2:Sexmale`     -0.500      1.015   -0.49   0.6224
## `Pclass3:Sexmale`      2.148      0.817    2.63   0.0086 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 714.11  on 535  degrees of freedom
## Residual deviance: 464.47  on 529  degrees of freedom
## AIC: 478.5
##
## Number of Fisher Scoring iterations: 6
```

Test the performance of different algorithms. The next models will be fitted against Linear Discriminant Analysis, SVM, Ada boost and, finally, Random Forest.

```
model.lda <- train(Survived ~ ., data = training, method = "lda")
model.lda
```

```
## Linear Discriminant Analysis
```

```
##
## 536 samples
##   7 predictors
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 536, 536, 536, 536, 536, 536, ...
##
## Resampling results
##
##   Accuracy  Kappa  Accuracy SD  Kappa SD
##   0.8       0.6    0.03         0.06
##
##
```

```r
library(kernlab)
model.svm <- train(Survived ~ ., data = training, method = "svmRadial")
```

```r
model.svm
```

```
## Support Vector Machines with Radial Basis Function Kernel
```

```
## Loading required package: kernlab
```

```
## 536 samples
##   7 predictors
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 536, 536, 536, 536, 536, 536, ...
##
## Resampling results across tuning parameters:
##
##   C    Accuracy  Kappa  Accuracy SD  Kappa SD
##   0.2  0.8       0.6    0.02         0.05
##   0.5  0.8       0.6    0.03         0.05
##   1.0  0.8       0.6    0.03         0.06
##
## Tuning parameter 'sigma' was held constant at a value of 0.1164
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were sigma = 0.1164 and C = 0.25.
```

```r
model.ada <- train(Survived ~ ., data = training, method = "ada")
```

```r
model.ada
```

```
## Boosted Classification Trees
##
```

```
## 536 samples
##   7 predictors
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 536, 536, 536, 536, 536, 536, ...
##
## Resampling results across tuning parameters:
##
##   iter  maxdepth  Accuracy  Kappa  Accuracy SD  Kappa SD
##    50   1         0.8       0.6    0.02         0.05
##    50   2         0.8       0.6    0.03         0.05
##    50   3         0.8       0.6    0.02         0.04
##   100   1         0.8       0.6    0.02         0.04
##   100   2         0.8       0.6    0.02         0.05
##   100   3         0.8       0.6    0.02         0.04
##   150   1         0.8       0.6    0.02         0.04
##   150   2         0.8       0.6    0.02         0.05
##   150   3         0.8       0.6    0.02         0.04
##
## Tuning parameter 'nu' was held constant at a value of 0.1
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were iter = 100, maxdepth = 3 and nu
##  = 0.1.
```

```r
rf.grid <- data.frame(.mtry = c(2, 3))
model.rf <- train(Survived ~ ., data = training, method = "rf")
model.rf
```

```
## Random Forest
##
## 536 samples
##   7 predictors
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 536, 536, 536, 536, 536, 536, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.8       0.6    0.02         0.05
##    6    0.8       0.5    0.02         0.05
##   10    0.8       0.5    0.02         0.04
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
model.rf.2 <- train(Survived ~ Pclass + SibSp + I(PAC == "C")
                    + I(Sex == "male") + I(Sex == "female" & Parch > 0)
                    + I(Embarked == "S"), data = training, method = "rf")
model.rf.2
```

```
## Random Forest
##
## 536 samples
##   7 predictors
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 536, 536, 536, 536, 536, 536, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##   2     0.8       0.6    0.02         0.05
##   4     0.8       0.6    0.02         0.05
##   7     0.8       0.5    0.02         0.04
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Model Evaluation

Cross-validate the validation set with the fitted models.

Measure the accuracy of `model.glm.1`:

```
confusionMatrix(validation$Survived, predict(model.glm.1, validation))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 182  37
##          1  37  99
##
##                Accuracy : 0.792
##                  95% CI : (0.746, 0.833)
##     No Information Rate : 0.617
##     P-Value [Acc > NIR] : 1.23e-12
##
##                   Kappa : 0.559
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.831
##             Specificity : 0.728
##          Pos Pred Value : 0.831
##          Neg Pred Value : 0.728
##              Prevalence : 0.617
##          Detection Rate : 0.513
##    Detection Prevalence : 0.617
##       Balanced Accuracy : 0.779
##
##        'Positive' Class : 0
##
```

`model.glm.1` yielded an Accuracy of `0.7915` and a Kappa of `0.599`.

Measure the accuracy of `model.glm.2`:

```
confusionMatrix(validation$Survived, predict(model.glm.2, validation))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 197  22
##          1  38  98
##
##                Accuracy : 0.831
##                  95% CI : (0.788, 0.868)
##     No Information Rate : 0.662
##     P-Value [Acc > NIR] : 7.89e-13
##
##                   Kappa : 0.634
##  Mcnemar's Test P-Value : 0.0528
##
##             Sensitivity : 0.838
##             Specificity : 0.817
##          Pos Pred Value : 0.900
##          Neg Pred Value : 0.721
##              Prevalence : 0.662
##          Detection Rate : 0.555
##    Detection Prevalence : 0.617
##       Balanced Accuracy : 0.827
##
##        'Positive' Class : 0
##
```

`model.glm.2` yielded an Accuracy of `0.831` and a Kappa of `0.6343`, an improvement over the previous iteration.

Measure the accuracy of `model.glm.3`:

```
confusionMatrix(validation$Survived, predict(model.glm.3, validation))
```

```
## Warning: prediction from a rank-deficient fit may be misleading
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   0   1
##          0 201  18
##          1  48  88
##
##                Accuracy : 0.814
##                  95% CI : (0.77, 0.853)
##     No Information Rate : 0.701
##     P-Value [Acc > NIR] : 8.71e-07
##
##                   Kappa : 0.59
##  Mcnemar's Test P-Value : 0.000357
##
##             Sensitivity : 0.807
##             Specificity : 0.830
##          Pos Pred Value : 0.918
##          Neg Pred Value : 0.647
##              Prevalence : 0.701
##          Detection Rate : 0.566
##    Detection Prevalence : 0.617
##       Balanced Accuracy : 0.819
##
##        'Positive' Class : 0
##
```

`model.glm.3` yielded an Accuracy of `0.8141` and a Kappa of `0.5895`.

Measure the accuracy of `model.glm.4`:

```r
confusionMatrix(validation$Survived, predict(model.glm.4, validation))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 209  10
##          1  67  69
##
##                Accuracy : 0.783
##                  95% CI : (0.737, 0.825)
##     No Information Rate : 0.777
##     P-Value [Acc > NIR] : 0.429
##
##                   Kappa : 0.502
##  Mcnemar's Test P-Value : 1.75e-10
##
##             Sensitivity : 0.757
##             Specificity : 0.873
##          Pos Pred Value : 0.954
##          Neg Pred Value : 0.507
##              Prevalence : 0.777
##          Detection Rate : 0.589
##    Detection Prevalence : 0.617
##       Balanced Accuracy : 0.815
```

```
##
##        'Positive' Class : 0
##
```

`model.glm.4` yielded an Accuracy of `0.7831` and a Kappa of `0.5015`.

Measure the accuracy of `model.lda`:

```
confusionMatrix(validation$Survived, predict(model.lda, validation))
```

```
## Loading required package: MASS
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 190  29
##          1  36 100
##
##                Accuracy : 0.817
##                  95% CI : (0.773, 0.856)
##     No Information Rate : 0.637
##     P-Value [Acc > NIR] : 7.86e-14
##
##                   Kappa : 0.609
##  Mcnemar's Test P-Value : 0.457
##
##             Sensitivity : 0.841
##             Specificity : 0.775
##          Pos Pred Value : 0.868
##          Neg Pred Value : 0.735
##              Prevalence : 0.637
##          Detection Rate : 0.535
##    Detection Prevalence : 0.617
##       Balanced Accuracy : 0.808
##
##        'Positive' Class : 0
##
```

`model.lda` yielded an Accuracy of `0.8169` and a Kappa of `0.6088`.

Measure the accuracy of `model.ada`:

```
confusionMatrix(validation$Survived, predict(model.ada, validation))
```

```
## Loading required package: ada
## Loading required package: rpart
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 201  18
##          1  40  96
##
##                Accuracy : 0.837
##                  95% CI : (0.794, 0.874)
##     No Information Rate : 0.679
##     P-Value [Acc > NIR] : 1.14e-11
##
##                   Kappa : 0.643
##  Mcnemar's Test P-Value : 0.00583
##
##             Sensitivity : 0.834
##             Specificity : 0.842
##          Pos Pred Value : 0.918
##          Neg Pred Value : 0.706
##              Prevalence : 0.679
##          Detection Rate : 0.566
##    Detection Prevalence : 0.617
##       Balanced Accuracy : 0.838
##
##        'Positive' Class : 0
##
```

`model.ada` yielded an Accuracy of `0.8394` and a Kappa of `0.656`. Currently the most accurate algorithm.

Measure the accuracy of `model.svm`:

```
confusionMatrix(validation$Survived, predict(model.svm, validation))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 200  19
##          1  37  99
##
##                Accuracy : 0.842
##                  95% CI : (0.8, 0.879)
##     No Information Rate : 0.668
##     P-Value [Acc > NIR] : 8.23e-14
##
##                   Kappa : 0.658
##  Mcnemar's Test P-Value : 0.0231
##
##             Sensitivity : 0.844
##             Specificity : 0.839
##          Pos Pred Value : 0.913
##          Neg Pred Value : 0.728
##              Prevalence : 0.668
##          Detection Rate : 0.563
##    Detection Prevalence : 0.617
##       Balanced Accuracy : 0.841
##
##        'Positive' Class : 0
##
```

`model.svm` yielded an Accuracy of `0.8394` and a Kappa of `0.647`. Almost the same as `model.ada`.

Measure the accuracy of `model.rf`:

```
confusionMatrix(validation$Survived, predict(model.rf, validation))
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:Hmisc':
##
##     combine
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 210   9
##          1  44  92
##
##                Accuracy : 0.851
##                  95% CI : (0.809, 0.886)
##     No Information Rate : 0.715
##     P-Value [Acc > NIR] : 1.44e-09
##
##                   Kappa : 0.668
##  Mcnemar's Test P-Value : 3.01e-06
##
##             Sensitivity : 0.827
##             Specificity : 0.911
##          Pos Pred Value : 0.959
##          Neg Pred Value : 0.676
##              Prevalence : 0.715
##          Detection Rate : 0.592
##    Detection Prevalence : 0.617
##       Balanced Accuracy : 0.869
##
##        'Positive' Class : 0
##
```

`model.rf` yielded an Accuracy of `0.8451` and a Kappa of `0.6574`. More accurate than both `model.svm` and `model.ada`.

Measure the accuracy of `model.rf.2`:

```
confusionMatrix(validation$Survived, predict(model.rf.2, validation))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 210   9
##          1  45  91
##
##                Accuracy : 0.848
##                  95% CI : (0.806, 0.884)
##     No Information Rate : 0.718
##     P-Value [Acc > NIR] : 6.29e-09
##
##                   Kappa : 0.661
##  Mcnemar's Test P-Value : 1.91e-06
##
##             Sensitivity : 0.824
##             Specificity : 0.910
##          Pos Pred Value : 0.959
##          Neg Pred Value : 0.669
##              Prevalence : 0.718
##          Detection Rate : 0.592
##    Detection Prevalence : 0.617
##       Balanced Accuracy : 0.867
##
##        'Positive' Class : 0
##
```

`model.rf.2` yielded an Accuracy of `0.8535` and a Kappa of `0.6756`. This model is a modification of `model.rf` and `model.glm.2`.

Compare the algorithms with each other to arrive with a more definitive conclusion.

```r
GLM1 <- predict(model.glm.1, test.df)
GLM2 <- predict(model.glm.2, test.df)
GLM3 <- predict(model.glm.3, test.df)
```

```
## Warning: prediction from a rank-deficient fit may be misleading
```

```r
GLM4 <- predict(model.glm.4, test.df)
LDA <- predict(model.lda, test.df)
ADA <- predict(model.ada, test.df)
SVM <- predict(model.svm, test.df)
RF <- predict(model.rf, test.df)
RF2 <- predict(model.rf.2, test.df)
```

GLM1 vs GLM2:

```r
table(GLM1, GLM2)
```

```
##     GLM2
## GLM1   0   1
##    0 249  11
##    1  21 137
```

GLM1 vs GLM3:

```r
table(GLM1, GLM3)
```

```
##     GLM3
## GLM1   0   1
##    0 248  12
##    1  31 127
```

GLM1 vs GLM4:

```r
table(GLM1, GLM4)
```

```
##     GLM4
## GLM1   0   1
##    0 255   5
##    1  75  83
```

GLM2 vs GLM3:

```r
table(GLM2, GLM3)
```

```
##     GLM3
## GLM2   0   1
##    0 263   7
##    1  16 132
```

GLM3 vs GLM4:

```r
table(GLM3, GLM4)
```

```
##     GLM4
## GLM3   0   1
##    0 278   1
##    1  52  87
```

LDA vs ADA:

```r
table(LDA, ADA)
```

```
##    ADA
## LDA   0   1
##   0 251  12
##   1  19 136
```

SVM vs RF:

```r
table(SVM, RF)
```

```
##    RF
## SVM   0   1
##   0 260   2
##   1  28 128
```

RF vs RF2:

```
table(RF, RF2)
```

```
##    RF2
## RF    0   1
##   0 282   6
##   1   7 123
```

Submitting to Kaggle

Format the output according to the required output for the competition. For the purpose of further comparison, `GLM2`, `ADA`, `SVM` and `RF` will be submitted.

```
kaggle.submit <- function(model) {
    results <- vector()
    filename <- paste(deparse(substitute(model)), ".csv", sep = "")
    for(i in 1:length(model)) {
        results[i] <- model[i]
    }
    results <- gsub(1, 0, results)
    results <- gsub(2, 1, results)
    file.submit <- cbind(test.df$PassengerId, results)
    colnames(file.submit) <- c("PassengerId", "Survived")
    write.csv(file.submit, file = filename, row.names = FALSE)
}
```

As of writing, GLM2 scored **0.76555**, ADA scored **0.76077**, SVM scored **0.78947**, RF scored **0.79426** and the modified RF2 scored **0.77990**.

Conclusion

Overall, 3 out of 5 models fitted fared better than the *Gender, Price and Class Based Model* benchmark but ultimately wasn't that remarkable. Further modifications could still be done, with the *random forest* algorithm showing the most promise.