Titanic Survival Prediction Project

Nadim Tabbal

9/29/2019

1. Introduction

1.1 Topic Overview

On April 15, 1912, the largest passenger liner ever made collided with an iceberg during her maiden voyage. When the Titanic sank it killed 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships. One of the reasons that the shipwreck resulted in 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.

1.2 Description of Dataset

A dataset is provided in csv format for 891 passengers with information about each passenger. It is clean and imputed for missing values. We represent each passenger in a row with features in columns such as age, cabin, point of embarking the ship, the ticket price, the name of the passenger, the number of parents/children aboard the ship, the unique ID, the class, the gender, the number of siblings or spouses aboard the ship, survival, the ticket number the title of the passenger and the family size. We read below both the original training and test sets as provided on Kaggle **Titanic Dataset Source**. The original training set has 891 rows while the test set has 418 rows. The test set has no information for survival, only the training set has. For this reason, the original training set has been divided into two sets, one for training and a second for validation. The accuracy of prediction was assessed on the validation set whereas predictions carried out on the test set. Worth mentioning is that in almost all instances predictions where provided on the validation set yet what counts are those on the test set. The training set was used for model derivation and parameters optimizations.

1.3 Project Goals

The goal of the project is to develop models that best predict whether a passenger would survive or not given his profile described with some of the above features. We develop 10 prediction models and assess for each its prediction performance, with the Accuracy metric. Since this is a classification problem, Accuracy is measured on the validation set as the proportion of correct predictions from the total predictions. The models are the following

1. Naive Approach
2. Naive Best Cutoff
3. F1 Sensitivity and Specificity Balancing
4. QDA (Quadratic Discriminant Analysis)
5. LDA (Linear Discriminant Analysis)
6. Linear Regression
7. Logistic Regression
8. K Nearest Neighbors
9. Classification Tree
10. Random Forest

2. Analysis Description

2.1 Importing Data

We import both the original training and test sets from the github repository (https://github.com/NATabbal/Titanic-Project) through the setup procedure.

2.2 Data Cleaning

No data cleaning has been carried out. In fact, the dataset provided is clean and imputed for NA's.

2.3 Data Exploration

We explore our data with a series of plots and tables for a better insight of variables effects and their different levels.

2.4 Models

For the first three models, the Naive Approach, the Naive Best Cutoff and F1 Sensitivity and Specificity Balancing, we have included one predictor that is the Fare. As mentioned above we hypothesized that passenger paying higher fares where better served and seated and hence had higher chances for survival. For QDA, LDA and Linear Regression we added the Age predictor. For K Nearest Neighbors, we kept only the categorical variables as the algorithm will treat inter-category distances equally which doesn't pose a problem.
For Logistic Regression, Classification Tree and Random Forest, we expand to six predictors to include Age, Class, point of Embarking, Fare, Sex and Title.

2.5 Insights

Almost no correlation between the age and fare variables. Chances for survival is independent of age Chances for survival is dependent of fare: higher fare gives a higher chance of survival Passengers traveling in first class managed to survive more than other classes Females managed to survive more than males

3. Setup, Visualization, Modeling and Results

3.1 Setup

```
if(!require(tidyverse)) install.packages("tidyverse",
                                         repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse

## -- Attaching packages ----------------------------------------------------------------------------- ti

## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.1     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------ tidyvers
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
if(!require(data.table)) install.packages("data.table",
                                          repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
```

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

```r
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest",
                                            repos = "http://cran.us.r-project.org")
```

```
## Loading required package: randomForest

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
if(!require(purrr)) install.packages("purrr", repos = "http://cran.us.r-project.org")
if(!require(descr)) install.packages("descr", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: descr
```

```r
if(!require(kableExtra)) install.packages("kableExtra",
                                          repos = "http://cran.us.r-project.org")
```

```
## Loading required package: kableExtra

##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```r
if(!require(plyr)) install.packages("plyr", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: plyr

## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:purrr':
##
##     compact
```

```r
library(tidyverse)
library(dplyr)
library(caret)
library(data.table)
library(rpart)
library(ggplot2)
library(randomForest)
library(purrr)
library(descr)
library(kableExtra)
library(plyr)
```

3.2 Reading the data from the Github repository

Reading both the titanic and test sets from the github repository

```r
titanic_train <-
  read_csv("https://raw.githubusercontent.com/NATabbal/Titanic-Project/master/train_clean.csv")
```

```
## Parsed with column specification:
## cols(
##   Age = col_double(),
##   Cabin = col_character(),
##   Embarked = col_character(),
##   Fare = col_double(),
##   Name = col_character(),
##   Parch = col_double(),
##   PassengerId = col_double(),
##   Pclass = col_double(),
##   Sex = col_character(),
##   SibSp = col_double(),
##   Survived = col_double(),
##   Ticket = col_character(),
##   Title = col_character(),
##   Family_Size = col_double()
## )
```

```r
titanic_test <-
  read_csv("https://raw.githubusercontent.com/NATabbal/Titanic-Project/master/test_clean.csv")
```

```
## Parsed with column specification:
## cols(
##   Age = col_double(),
##   Cabin = col_character(),
##   Embarked = col_character(),
##   Fare = col_double(),
##   Name = col_character(),
##   Parch = col_double(),
##   PassengerId = col_double(),
##   Pclass = col_double(),
##   Sex = col_character(),
```

```
##   SibSp = col_double(),
##   Survived = col_logical(),
##   Ticket = col_character(),
##   Title = col_character(),
##   Family_Size = col_double()
## )
```

Display of the first six entries of the titanic_train set as provided on Kaggle

```r
options(dplyr.width = Inf)
head(titanic_train)
```

```
## # A tibble: 6 x 14
##     Age Cabin Embarked  Fare
##   <dbl> <chr> <chr>    <dbl>
## 1    22 <NA>  S         7.25
## 2    38 C85   C        71.3
## 3    26 <NA>  S         7.92
## 4    35 C123  S        53.1
## 5    35 <NA>  S         8.05
## 6    30 <NA>  Q         8.46
##   Name                                                Parch PassengerId
##   <chr>                                               <dbl>       <dbl>
## 1 Braund, Mr. Owen Harris                                 0           1
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer)     0           2
## 3 Heikkinen, Miss. Laina                                  0           3
## 4 Futrelle, Mrs. Jacques Heath (Lily May Peel)            0           4
## 5 Allen, Mr. William Henry                                0           5
## 6 Moran, Mr. James                                        0           6
##   Pclass Sex    SibSp Survived Ticket           Title Family_Size
##    <dbl> <chr>  <dbl>    <dbl> <chr>            <chr>       <dbl>
## 1      3 male       1        0 A/5 21171        Mr              1
## 2      1 female     1        1 PC 17599         Mrs             1
## 3      3 female     0        1 STON/O2. 3101282 Miss            0
## 4      1 female     1        1 113803           Mrs             1
## 5      3 male       0        0 373450           Mr              0
## 6      3 male       0        0 330877           Mr              0
```

Display of the first six entries of the titanic_test set as provided on Kaggle

```r
options(dplyr.width = Inf)
head(titanic_test)
```

```
## # A tibble: 6 x 14
##     Age Cabin Embarked  Fare Name
##   <dbl> <chr> <chr>    <dbl> <chr>
## 1  34.5 <NA>  Q         7.83 Kelly, Mr. James
## 2  47   <NA>  S         7    Wilkes, Mrs. James (Ellen Needs)
## 3  62   <NA>  Q         9.69 Myles, Mr. Thomas Francis
## 4  27   <NA>  S         8.66 Wirz, Mr. Albert
## 5  22   <NA>  S        12.3  Hirvonen, Mrs. Alexander (Helga E Lindqvist)
## 6  14   <NA>  S         9.22 Svensson, Mr. Johan Cervin
```

```
##   Parch PassengerId Pclass Sex    SibSp Survived Ticket  Title Family_Size
##   <dbl>       <dbl>  <dbl> <chr>  <dbl> <lgl>    <chr>   <chr>       <dbl>
## 1     0         892      3 male       0 NA       330911  Mr              0
## 2     0         893      3 female     1 NA       363272  Mrs             1
## 3     0         894      2 male       0 NA       240276  Mr              0
## 4     0         895      3 male       0 NA       315154  Mr              0
## 5     1         896      3 female     1 NA       3101298 Mrs             2
## 6     0         897      3 male       0 NA       7538    Mr              0
```

Dimension of the original titanic training set as provided on Kaggle

```r
dim(titanic_train)
```

```
## [1] 891  14
```

Variables' names of the titanic training set provided on Kaggle

```r
names(titanic_train)
```

```
##  [1] "Age"         "Cabin"       "Embarked"    "Fare"        "Name"
##  [6] "Parch"       "PassengerId" "Pclass"      "Sex"         "SibSp"
## [11] "Survived"    "Ticket"      "Title"       "Family_Size"
```

Variables' names of the titanic test set provided on Kaggle

```r
names(titanic_test)
```

```
##  [1] "Age"         "Cabin"       "Embarked"    "Fare"        "Name"
##  [6] "Parch"       "PassengerId" "Pclass"      "Sex"         "SibSp"
## [11] "Survived"    "Ticket"      "Title"       "Family_Size"
```

Dimension of the original titanic test set as provided on Kaggle

```r
dim(titanic_test)
```

```
## [1] 418  14
```

Below are the description of the training and test sets variables 1. Age: Age of passenger 2. Cabin: Cabin number 3. Embarked: Point of embarking the ship (C = Cherbourg; Q = Queenstown; S = Southampton) 4. Fare: Price of ticket 5. Name: Name of passenger 6. Parch: Number of parents/children aboard the ship 7. PassengerId: Unique ID of passenger 8. Pclass: Class of passenger (1 = 1st; 2 = 2nd; 3 = 3rd) 9. Sex: Gender of passenger 10. SibSp: Number of siblings/spouses aboard the ship 11. Survived: True if passenger survived the disaster (0 = No; 1 = Yes) 12. Ticket: Ticket number 13. Title: Title of passenger (Mr, Mrs etc.) 14. Family_Size: Parch + SibSp

Setting the seed to 1

```r
set.seed(1)
```

We generate indices to partition the training set (the one provided on Kaggle) into a new training set that we will designate "train", and a test set that we designate "validation"

```r
index <- createDataPartition(titanic_train$Survived, times = 1, p = 0.5, list= FALSE)
length(index)
```

```
## [1] 446
```

Derivation of both the new train and validation sets

```r
train <- titanic_train[-index,]
validation <- titanic_train[index,]
```

Displaying the first six entries of the train set

```r
options(dplyr.width = Inf)
head(train)
```

```
## # A tibble: 6 x 14
##     Age Cabin Embarked  Fare
##   <dbl> <chr> <chr>    <dbl>
## 1    22 <NA>  S         7.25
## 2    38 C85   C        71.3
## 3    26 <NA>  S         7.92
## 4    35 C123  S        53.1
## 5    35 <NA>  S         8.05
## 6    54 E46   S        51.9
##   Name                                                Parch PassengerId
##   <chr>                                               <dbl>       <dbl>
## 1 Braund, Mr. Owen Harris                                 0           1
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer)     0           2
## 3 Heikkinen, Miss. Laina                                  0           3
## 4 Futrelle, Mrs. Jacques Heath (Lily May Peel)            0           4
## 5 Allen, Mr. William Henry                                0           5
## 6 McCarthy, Mr. Timothy J                                 0           7
##   Pclass Sex    SibSp Survived Ticket           Title Family_Size
##    <dbl> <chr>  <dbl>    <dbl> <chr>            <chr>       <dbl>
## 1      3 male       1        0 A/5 21171        Mr              1
## 2      1 female     1        1 PC 17599         Mrs             1
## 3      3 female     0        1 STON/O2. 3101282 Miss            0
## 4      1 female     1        1 113803           Mrs             1
## 5      3 male       0        0 373450           Mr              0
## 6      1 male       0        0 17463            Mr              0
```

Transforming in "train" the variables Pclass, Embarked, Sex, Title and Survived to factor type

```r
train <- train %>% mutate (Class_f = as.factor(train$Pclass),
                           Embarked_f = as.factor(train$Embarked),
                           Sex_f = as.factor(train$Sex),
                           Title_f = as.factor(train$Title),
                           Survived_f = as.factor(train$Survived))
```

Transforming in "validation" the variables Pclass, Embarked, Sex, Title and Survived to factor type

```r
validation <- validation %>% mutate (Class_f = as.factor(validation$Pclass),
                                     Embarked_f = as.factor(validation$Embarked),
                                     Sex_f = as.factor(validation$Sex),
                                     Title_f = as.factor(validation$Title),
                                     Survived_f = as.factor(validation$Survived))
```

In titanic_test, we transform the variables Pclass, Embarked, Sex, Title and Survived to factor type. This is where the final predictions are going to be made

```r
titanic_test <- titanic_test %>% mutate (Class_f = as.factor(titanic_test$Pclass),
                                         Embarked_f = as.factor(titanic_test$Embarked),
                                         Sex_f = as.factor(titanic_test$Sex),
                                         Title_f = as.factor(titanic_test$Title),
                                         Survived_f = as.factor(titanic_test$Survived))
```

3.3 Data Visualization

Exploring and ploting data prevalence

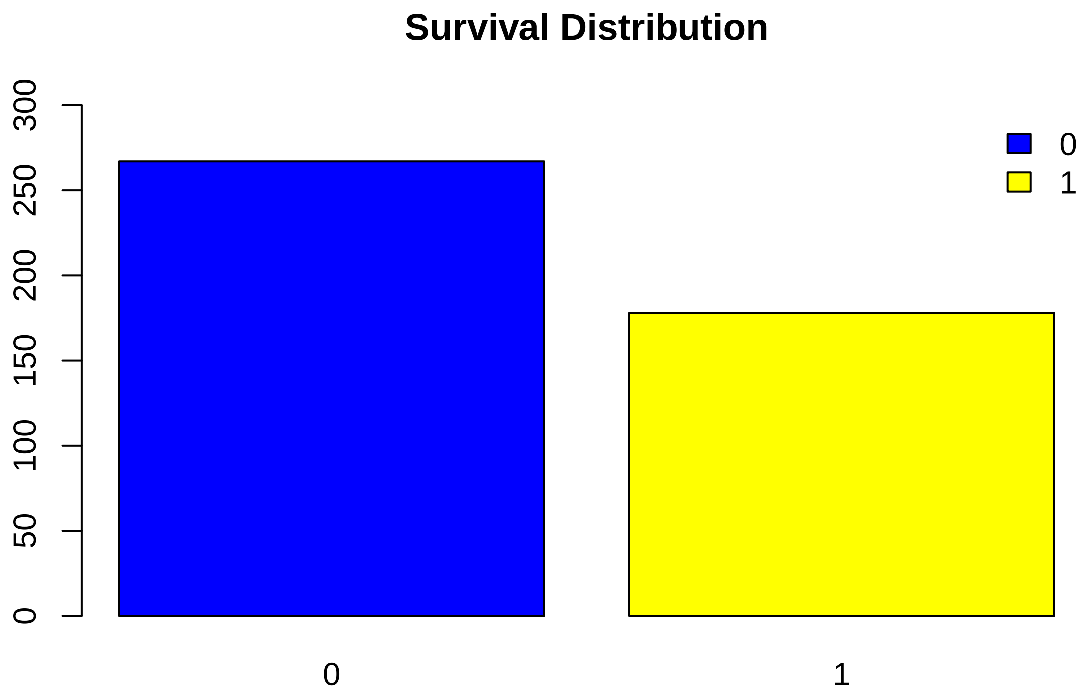
```r
prev <- table(train$Survived_f)
prev
```

```
##
##   0   1
## 267 178
```

```r
pr_prev <- prop.table(prev)
pr_prev
```

```
##
##   0   1
## 0.6 0.4
```

```r
barplot(prev, main = "Survival Distribution", col=c("blue", "yellow"),
        legend = rownames(prev),
        args.legend = list(bty = "n", x = "topright"), ylim=c(0,300))
```



The data is moderately imbalanced between survival and non-survival with 60% of non-survival (level "0") to be considered as the positive class, and 40% of survival (level "1")

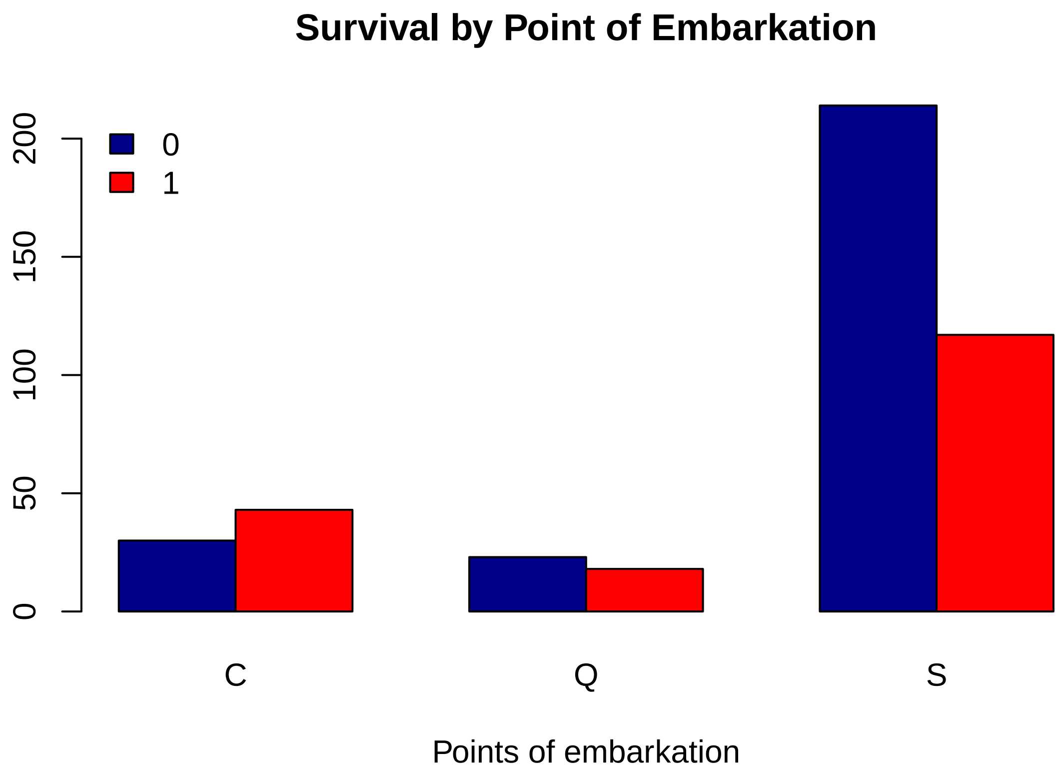
Visualization of survival per class of passenger in a bar plot

```
S_cla <- table(train$Survived_f, train$Class_f)
barplot(S_cla, main="Survival by Class of Passenger", xlab="Class of passenger",
        col=c("red", "yellow"), legend = rownames(S_cla),
        args.legend = list(bty = "n", x = "topleft"),
        ylim=c(0,200), beside=TRUE)
```

Class 1 had a higher survival rate while class 3 had a significant non-survival rate

Visualization of survival per point of embarkation in a bar plot

```
S_emb <- table(train$Survived_f, train$Embarked_f)
barplot(S_emb, main="Survival by Point of Embarkation", xlab="Points of embarkation",
        col=c("darkblue", "red"), legend = rownames(S_emb),
        args.legend = list(bty = "n", x = "topleft"),
        beside=TRUE)
```



Southampton had a significant non-survival rate compared to survival, as opposed to Queenstown and Cherbourg

Visualization of survival per gender in a bar plot

```
S_sex <- table(train$Survived_f, train$Sex_f)
S_sex
```

```
##
##     female male
##   0     36  231
##   1    118   60
```

```
prop.table(S_sex, 2)
```

```
##
##        female      male
##   0 0.2337662 0.7938144
##   1 0.7662338 0.2061856
```

```
barplot(S_sex, main="Survival by Gender", xlab="Gender of passenger",
        col=c("grey", "yellow"), legend = rownames(S_sex),
        args.legend = list(bty = "n", x = "topright"),
        ylim=c(0,250), beside=TRUE)
```

76% of females survived against 21% of male

Visualization of survival per title of passenger in a bar plot

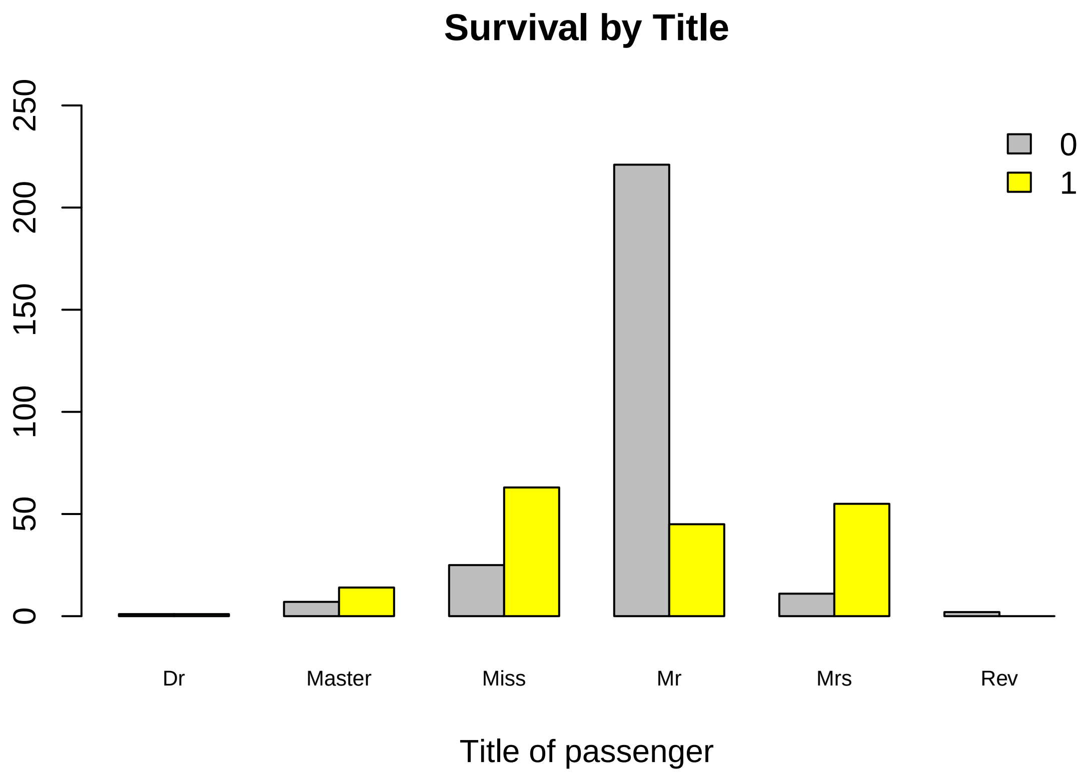
```
S_title <- table(train$Survived_f, train$Title_f)
S_title
```

```
##
##      Dr Master Miss  Mr Mrs Rev
##   0   1      7   25 221  11   2
##   1   1     14   63  45  55   0
```

```
prop.table(S_title, 2)
```

```
##
##            Dr    Master      Miss        Mr       Mrs       Rev
##   0 0.5000000 0.3333333 0.2840909 0.8308271 0.1666667 1.0000000
##   1 0.5000000 0.6666667 0.7159091 0.1691729 0.8333333 0.0000000
```

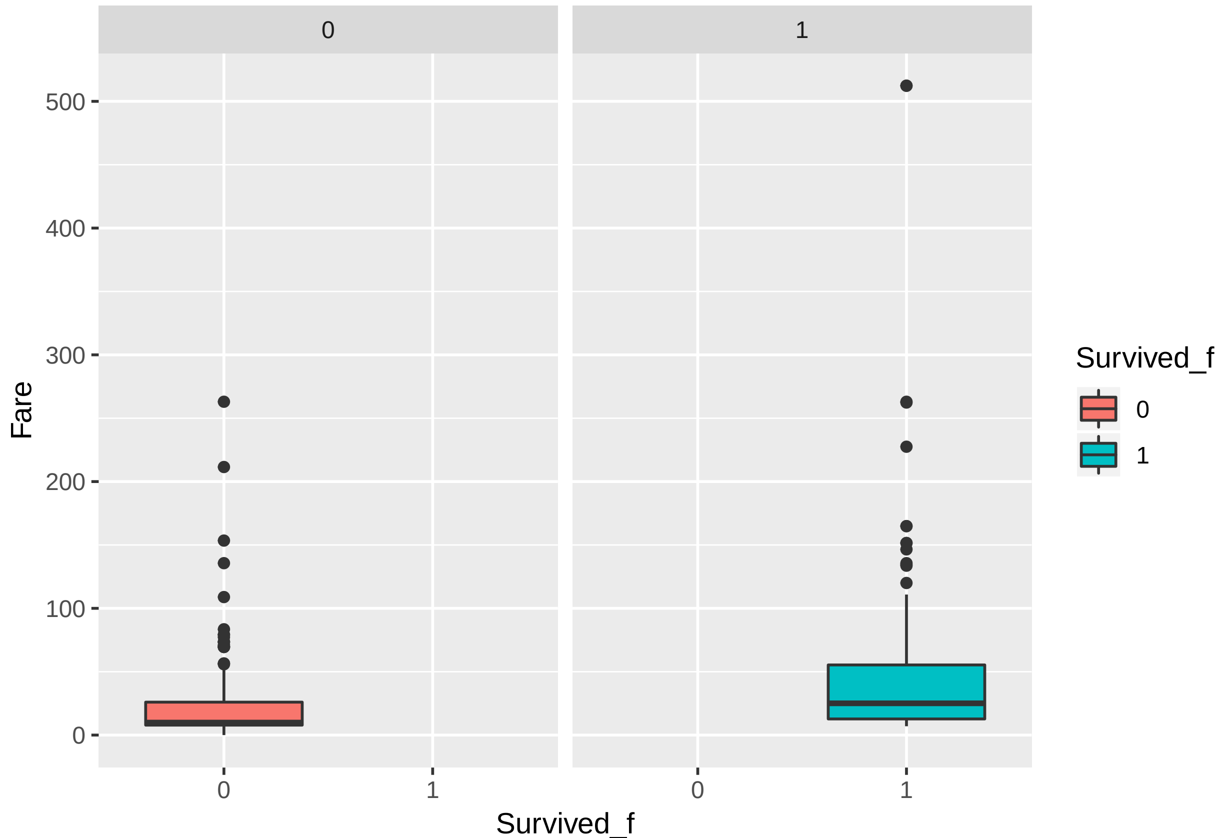
```
barplot(S_title, main="Survival by Title", xlab="Title of passenger",
        col=c("grey", "yellow"), legend = rownames(S_title),
        args.legend = list(bty = "n", x = "topright"),
        ylim=c(0,250), beside=TRUE, cex.names = 0.7)
```



There were significantly non-surviving passengers among those holding the title "Mr." as opposed to "Mrs." and "Ms."

Faceted boxplot of fare per levels of survival

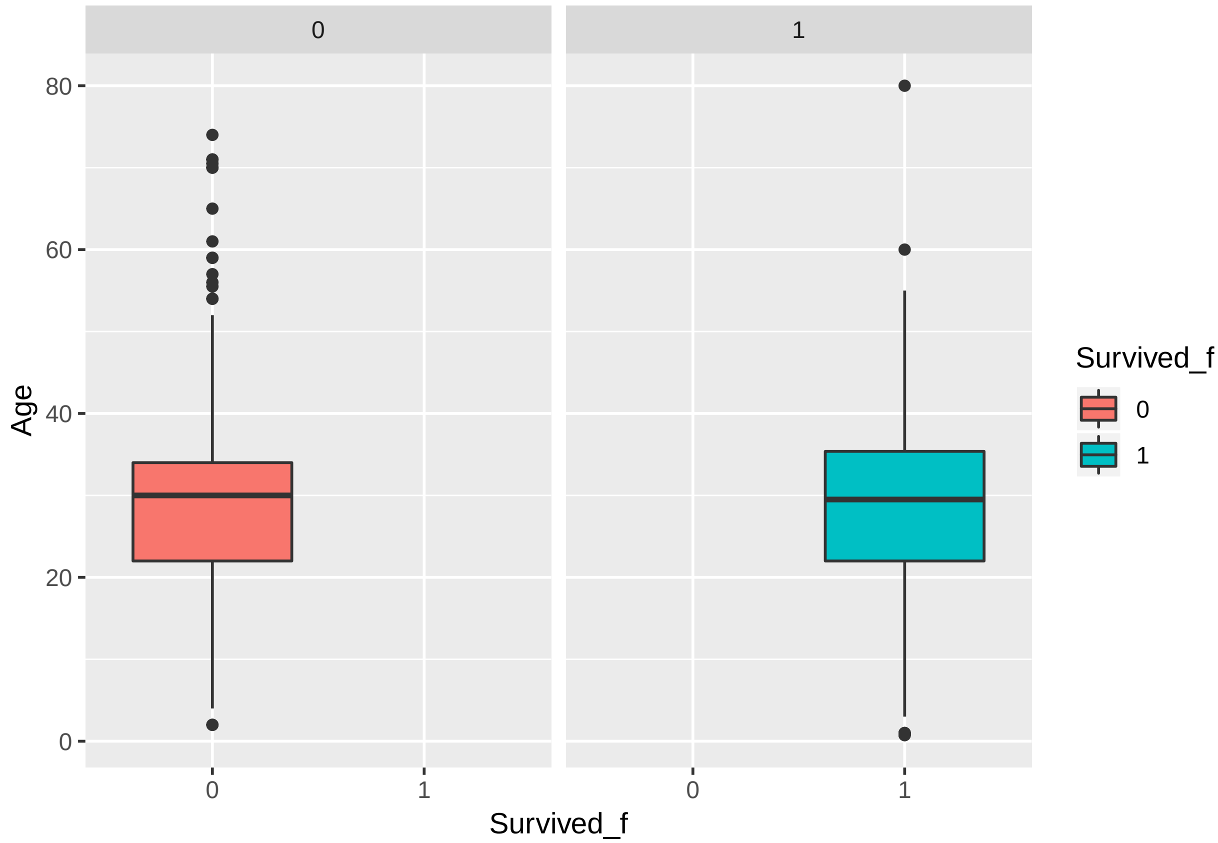
```
fare_box <- ggplot(train, aes(x=Survived_f, y=Fare, fill=Survived_f)) +
  geom_boxplot() + facet_grid(.~Survived_f)
fare_box
```



The higher the Fare the higher the chance of survival

Faceted boxplot of age per levels of survival

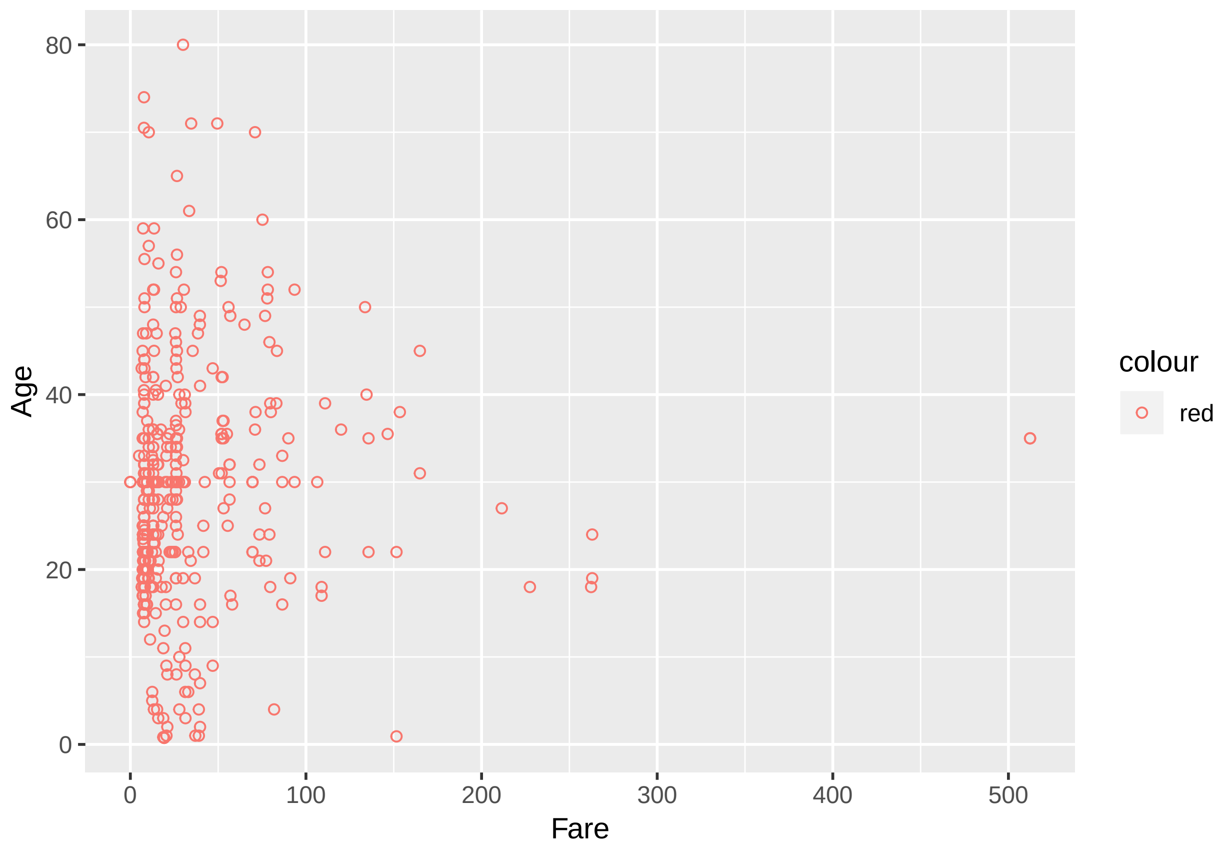
```
Age_box <- ggplot(train, aes(x=Survived_f, y=Age, fill=Survived_f)) +
  geom_boxplot() + facet_grid(.~Survived_f)
Age_box
```



The age distribution was almost identical for those who survived and those who did not

Scatter plot of fare and age in order to examine if any linear pattern

```
ggplot(train, aes(x=Fare, y=Age, color="red")) + geom_point(shape=1)
```



Correlation between fare and age

```
cor(train$Fare, train$Age)
```

```
## [1] 0.05919878
```

The correlation is as low as 0.07: Both will be included in prediction models

3.4 Modeling

3.4.1 Naive Approach

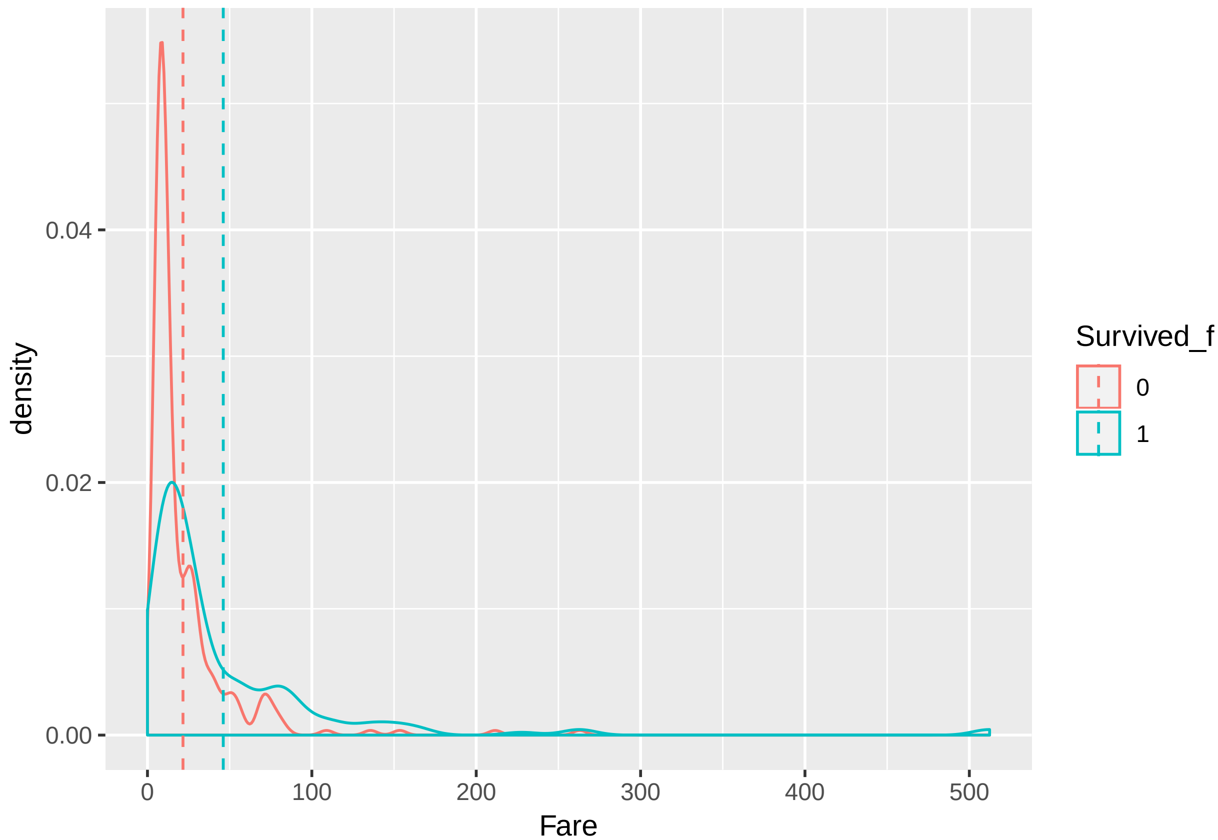
E(Y/ X > x) In our case, we predict Y, to be 1 i.e. survival whenever the Fare is larger than 22.5 Y = 1 / X >= 22.5 Y = 0 / X < 22.5

Density plot stratified by survival i.e. by Survived_f

```
mu <- ddply(train, "Survived_f", summarise, grp.mean=mean(Fare))
mu
```

```
##   Survived_f grp.mean
## 1          0 21.60314
## 2          1 46.10110
```

```r
den_plo_f_Sf <-ggplot(train, aes(x=Fare, color=Survived_f)) +
  geom_density() +
  geom_vline(data=mu, aes(xintercept=grp.mean, color=Survived_f),
             linetype="dashed")
den_plo_f_Sf
```



We notice that somewhere below 22.5 the chances for survival are lower than non-survival, however above 22.5 the chances of survival are higher

We develop a naive approach to guessing survival on the validation set: if Fare paid is higher than 22.5 we predict survival otherwise non-survival

```r
y_hat_naive <- ifelse(validation$Fare > 22.5, "1" , "0") %>%
  factor(levels = levels(validation$Survived_f))
y_hat_naive
```

```
##   [1] 0 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1
##  [36] 0 0 1 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 1 0 1 0 1 0 1
##  [71] 0 1 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
## [106] 0 1 0 1 0 0 0 1 0 0 0 0 1 1 0 1 0 1 0 0 0 1 0 0 1 0 1 1 0 0 1 0 0 1 1
## [141] 0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 1 1 0 1
## [176] 0 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0 0 0
## [211] 0 1 0 1 1 0 0 0 0 1 1 0 1 1 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1 0 0
## [246] 0 0 0 1 1 0 0 1 0 0 1 1 0 0 1 1 0 1 1 1 0 1 0 1 0 1 0 1 1 0 1 1 1 1 1
## [281] 1 0 0 1 1 1 0 0 0 1 0 1 0 1 1 1 0 1 0 1 0 0 0 0 1 1 1 0 0 1 0 0 0 0 1
## [316] 0 1 0 0 1 1 0 1 1 1 1 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 1 1 1 1 1
```

```
## [351] 0 0 0 1 1 0 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 0 1 1 0 1 0 1 0 0 0 0 0 0 1
## [386] 0 0 0 0 0 1 0 1 0 1 0 1 1 0 1 1 1 1 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 1 0
## [421] 0 0 0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0 0 1 0
## Levels: 0 1
```

Naive Bayes accuracy achieved on the validation set

```r
naive_acc <- confusionMatrix(y_hat_naive, validation$Survived_f)$overall["Accuracy"]
naive_acc
```

```
##  Accuracy
## 0.6636771
```

Naive Bayes prediction model on the test set

```r
y_hat_naive <- ifelse(titanic_test$Fare > 22.5, "1" , "0")
y_hat_naive
```

```
##   [1] "0" "0" "0" "0" "0" "0" "0" "1" "0" "1" "0" "1" "1" "1" "1" "1" "0"
##  [18] "0" "0" "0" "1" "0" "1" "1" "1" "0" "1" "0" "1" "0" "1" "1" "0" "1"
##  [35] "1" "0" "0" "0" "0" "1" "0" "1" "0" "0" "1" "0" "1" "0" "1" "0" "1"
##  [52] "0" "1" "1" "0" "1" "0" "0" "0" "1" "0" "0" "0" "0" "1" "0" "0" "1"
##  [69] "1" "1" "0" "0" "0" "1" "1" "1" "0" "1" "0" "0" "0" "1" "1" "0" "0"
##  [86] "0" "0" "0" "0" "1" "0" "0" "1" "0" "1" "0" "1" "0" "0" "0" "1" "1"
## [103] "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "1" "0" "1" "0" "0" "0" "1"
## [120] "1" "0" "0" "1" "0" "0" "0" "0" "1" "0" "0" "0" "1" "1" "0" "0" "0"
## [137] "0" "0" "0" "1" "1" "1" "1" "1" "1" "0" "1" "0" "1" "1" "1" "0" "0"
## [154] "0" "1" "0" "1" "0" "1" "0" "0" "0" "0" "0" "0" "0" "1" "1" "1" "0"
## [171] "0" "0" "0" "0" "1" "1" "1" "1" "1" "1" "0" "1" "1" "0" "1" "0" "0"
## [188] "0" "1" "0" "1" "1" "0" "0" "1" "0" "1" "0" "0" "0" "0" "0" "1" "1"
## [205] "0" "1" "0" "0" "1" "0" "1" "0" "1" "1" "0" "1" "0" "1" "1" "0" "0"
## [222] "0" "0" "0" "1" "0" "0" "0" "0" "0" "1" "1" "0" "0" "1" "0" "1" "0"
## [239] "0" "1" "1" "1" "1" "0" "1" "1" "0" "1" "1" "0" "1" "0" "1" "0" "0"
## [256] "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "1" "0"
## [273] "1" "0" "0" "1" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0"
## [290] "0" "1" "0" "0" "1" "0" "0" "1" "0" "1" "0" "0" "0" "0" "0" "0" "1"
## [307] "1" "0" "1" "0" "0" "0" "0" "0" "1" "0" "1" "0" "0" "1" "0" "0" "0"
## [324] "1" "1" "0" "1" "1" "1" "0" "1" "1" "0" "0" "0" "1" "0" "0" "1" "0"
## [341] "0" "0" "1" "1" "0" "0" "0" "0" "0" "0" "1" "0" "1" "1" "0" "1" "1"
## [358] "0" "0" "0" "1" "1" "0" "0" "1" "1" "0" "1" "1" "0" "0" "1" "0" "0"
## [375] "1" "1" "0" "0" "1" "1" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1"
## [392] "1" "0" "0" "0" "1" "0" "1" "0" "0" "1" "0" "1" "1" "1" "0" "0" "1"
## [409] "0" "0" "0" "1" "0" "0" "1" "0" "0" "0"
```

Initializing a data frame to display the Naive Bayes accuracy

```r
df1 <- data_frame(Model="Naive Bayes", Accuracy=naive_acc)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```r
df1 %>% knitr::kable() %>% kable_styling("striped" , full_width = T)
```

Model	Accuracy
Naive Bayes	0.6636771

3.4.2 Naive Best Cutoff

E(Y/ X > x1, x2, x3…xp)

In our case, we predict Y, to be 1 i.e. survival whenever the Fare is larger than 22.5

Looking for the optimal cutoff value of the fare that maximizes accuracy

```r
cutoff <- seq (0, 512, 10)
accuracy <- map_dbl (cutoff, function(x) {
  y_hat <- ifelse (train$Fare > x, "1", "0") %>%
    factor (levels = levels(validation$Survived_f))
  mean(y_hat == train$Survived_f)
})
max(accuracy)
```

```
## [1] 0.6629213
```

The best fare cutoff value leading to the highest accuracy

```r
best_cutoff  <-  cutoff[which.max(accuracy)]
best_cutoff
```

```
## [1] 50
```

The best cutoff predictions on the validation set

```r
y_hat_naive_bc <- ifelse(validation$Fare > 50, "1" , "0") %>%
  factor(levels = levels(validation$Survived_f))
y_hat_naive_bc
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0
##  [36] 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1
##  [71] 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
## [106] 0 1 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 1
## [141] 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0
## [176] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0
## [211] 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
## [246] 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1
## [281] 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0
## [316] 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1
## [351] 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 1
## [386] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0
## [421] 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0
## Levels: 0 1
```

Accuracy achieved on the validation set with the best cutoff of 50

```r
naive_acc_best <- confusionMatrix(y_hat_naive_bc,
                                  validation$Survived_f)$overall["Accuracy"]
naive_acc_best
```

```
##  Accuracy
## 0.6995516
```

Predictions of the best cutoff model on the test set

```r
y_hat_naive_bc_t <- ifelse(titanic_test$Fare > 50, "1" , "0")
y_hat_naive_bc_t
```

```
##   [1] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "1" "0" "0"
##  [18] "0" "0" "0" "1" "0" "0" "1" "1" "0" "1" "0" "0" "0" "0" "0" "0" "0"
##  [35] "1" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1" "0" "0" "0" "1" "0" "1"
##  [52] "0" "0" "1" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1" "0" "0" "0"
##  [69] "0" "1" "0" "0" "0" "0" "1" "1" "0" "0" "0" "0" "0" "1" "0" "0" "0"
##  [86] "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "1" "0" "0" "0" "1" "0"
## [103] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "1"
## [120] "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
## [137] "0" "0" "0" "0" "0" "1" "1" "0" "0" "0" "1" "0" "0" "0" "1" "0" "0"
## [154] "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0"
## [171] "0" "0" "0" "0" "0" "0" "0" "1" "0" "1" "0" "1" "1" "0" "1" "0" "0"
## [188] "0" "1" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "1" "0"
## [205] "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1" "1" "0" "0"
## [222] "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "1" "0" "1" "0"
## [239] "0" "1" "0" "0" "1" "0" "0" "1" "0" "0" "0" "0" "0" "0" "1" "0" "0"
## [256] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0"
## [273] "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0"
## [290] "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
## [307] "1" "0" "1" "0" "0" "0" "0" "0" "1" "0" "1" "0" "0" "0" "0" "0" "0"
## [324] "0" "1" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
## [341] "0" "0" "1" "1" "0" "0" "0" "0" "0" "0" "1" "0" "1" "1" "0" "0" "1"
## [358] "0" "0" "0" "1" "0" "0" "0" "1" "1" "0" "0" "1" "0" "0" "1" "0" "0"
## [375] "1" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1"
## [392] "0" "0" "0" "0" "1" "0" "1" "0" "0" "1" "0" "1" "0" "0" "0" "0" "1"
## [409] "0" "0" "0" "1" "0" "0" "1" "0" "0" "0"
```

Adding to the data frame the accuracy achieved with the best cutoff model

```r
df2 <- bind_rows(df1, data_frame(Model="Naive Best Cutoff", Accuracy=naive_acc_best))
df2 %>% knitr::kable() %>% kable_styling("striped" , full_width = T)
```

Model	Accuracy
Naive Bayes	0.6636771
Naive Best Cutoff	0.6995516

3.4.3 F1 Sensitivity and Specificity Balancing

Sensitivity = TP/(TP+FN) Specificity = TP/(TP + FP) F1 = 2. (Sensitivity . Specificity)/(Sensitivity + Specificity)

We generate a sequence of cutoff values

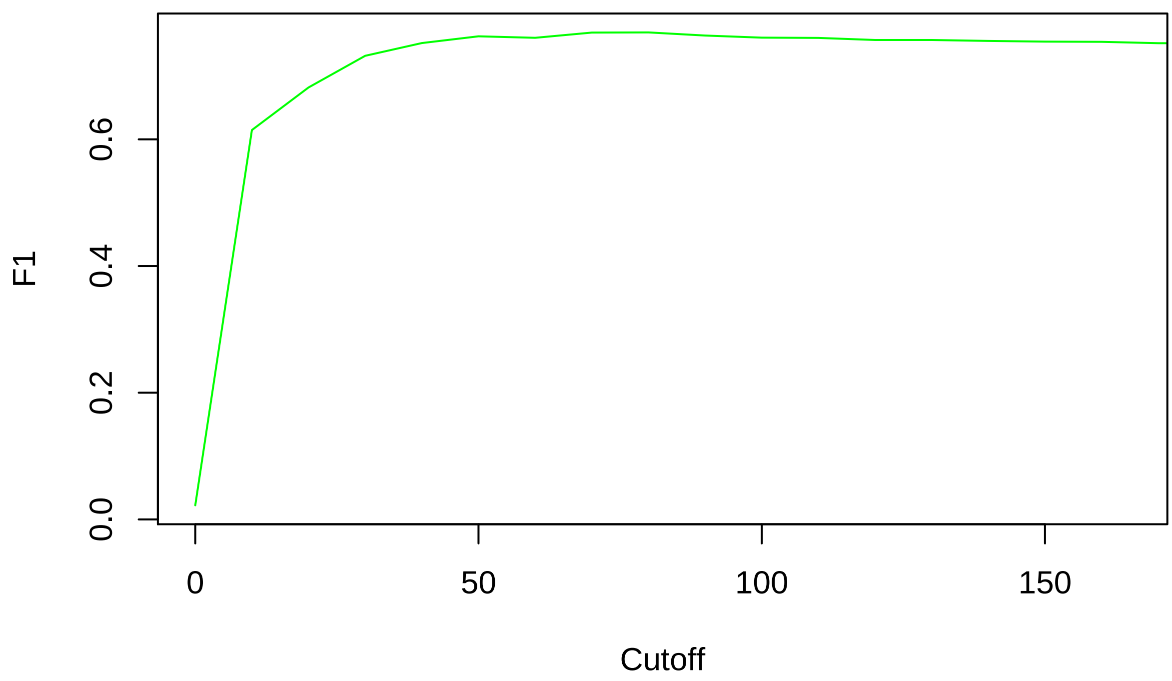
```
Cutoff <- seq(0, 512, 10)
```

We develop a function that calculates for each cutoff the F1 value that balances sensitivity and specificity

```
F1 <- map_dbl (cutoff, function(x){
  y_hat <- ifelse (train$Fare > x, "1", "0") %>%
    factor (levels = levels(validation$Survived_f))
  F_meas(data=y_hat, reference = factor(train$Survived_f))
})
```

Plot of the cutoff against F1

```
plot(Cutoff, F1, xlim = c(0, 165), type = "l", col="green")
```



The maximum achieved F1 value

```
max(F1)
```

```
## [1] 0.7687776
```

The maximum achieved balance between sensitivity and specificity is at 77.2%

The cutoff that balances best sensitivity and specificity

```r
best_cutoff  <-  cutoff[which.max(F1)]
best_cutoff
```

```
## [1] 80
```

The best cutoff is 50

Predictions on the validation set based on best F1 model

```r
y_hat_f1 <- ifelse(validation$Fare > best_cutoff, "1" , "0") %>%
  factor (levels= levels(validation$Survived_f))
y_hat_f1
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [36] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
##  [71] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
## [106] 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0
## [141] 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0
## [176] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0
## [211] 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [246] 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1
## [281] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
## [316] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1
## [351] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1
## [386] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [421] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
## Levels: 0 1
```

Confusion Matrix on the validation set of the F1 model

```r
confusionMatrix(data = y_hat_f1, reference = validation$Survived_f)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 271 134
##          1  11  30
##
##                Accuracy : 0.6749
##                  95% CI : (0.6292, 0.7182)
##     No Information Rate : 0.6323
##     P-Value [Acc > NIR] : 0.03379
##
##                   Kappa : 0.1707
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.9610
##             Specificity : 0.1829
##          Pos Pred Value : 0.6691
##          Neg Pred Value : 0.7317
```

```
##               Prevalence : 0.6323
##           Detection Rate : 0.6076
##     Detection Prevalence : 0.9081
##        Balanced Accuracy : 0.5720
##
##         'Positive' Class : 0
##
```

Achieved accuracy of the F1 model on the validation set

```r
acc_f1 <- confusionMatrix(data = y_hat_f1, reference =
                            validation$Survived_f)$overall["Accuracy"]
acc_f1
```

```
##  Accuracy
## 0.6748879
```

Predictions on the test set for the F1 model

```r
y_hat_f1_t <- ifelse(titanic_test$Fare > best_cutoff, "1" , "0")
y_hat_f1_t
```

```
##   [1] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0"
##  [18] "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0"
##  [35] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
##  [52] "0" "0" "1" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1" "0" "0" "0"
##  [69] "0" "1" "0" "0" "0" "0" "1" "1" "0" "0" "0" "0" "0" "1" "0" "0" "0"
##  [86] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
## [103] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0"
## [120] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
## [137] "0" "0" "0" "0" "0" "1" "1" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0"
## [154] "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
## [171] "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "1" "0" "0" "1" "0" "0"
## [188] "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "1" "0"
## [205] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "1" "0" "0"
## [222] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
## [239] "0" "1" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0"
## [256] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
## [273] "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0"
## [290] "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
## [307] "1" "0" "1" "0" "0" "0" "0" "0" "1" "0" "1" "0" "0" "0" "0" "0" "0"
## [324] "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
## [341] "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
## [358] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0"
## [375] "1" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1"
## [392] "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "1"
## [409] "0" "0" "0" "1" "0" "0" "1" "0" "0" "0"
```

Adding to the data frame the accuracy achieved with the F1 model

```r
df3 <- bind_rows(df2, data_frame(Model="F1 Balancing", Accuracy=acc_f1))
df3 %>% knitr::kable() %>% kable_styling("striped" , full_width = T)
```

Model	Accuracy
Naive Bayes	0.6636771
Naive Best Cutoff	0.6995516
F1 Balancing	0.6748879

3.4.4. QDA (Quadratic Discriminant Analysis)

The distributions of Pr X/Y=0 (X) and Pr X/Y=1 (X) are multivariate normal where X in our case represents Age and Fare Fare = a. Age^2 + b. Age + c

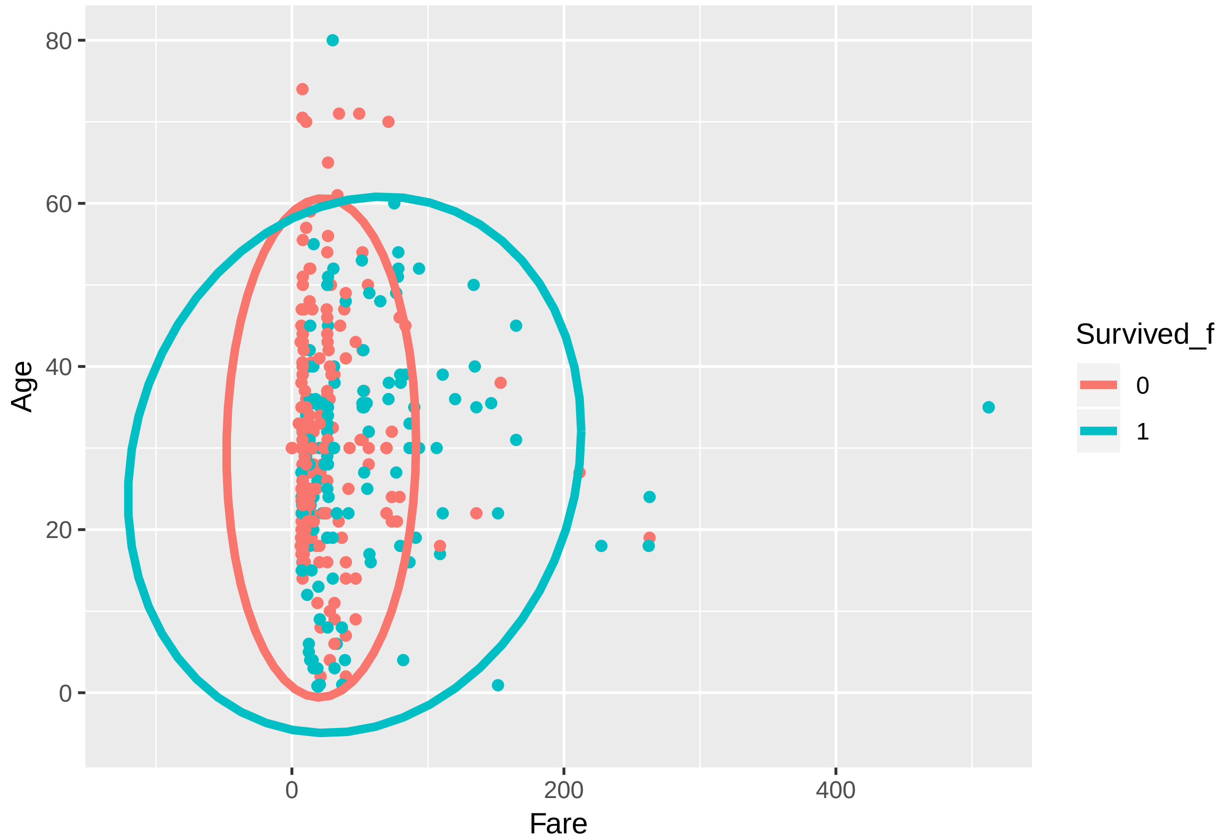
Deriving the average, standard deviation and correlation for Age and Fare per survival group

```r
params <- train %>%
  group_by(Survived_f) %>%
  summarize(avg_1 = mean(Fare), avg_2 = mean(Age),
            sd_1= sd(Fare), sd_2 = sd(Age),
            r = cor(Fare, Age))
params
```

```
##      avg_1    avg_2     sd_1     sd_2          r
## 1 31.40232 29.18933 49.41233 12.82086 0.05919878
```

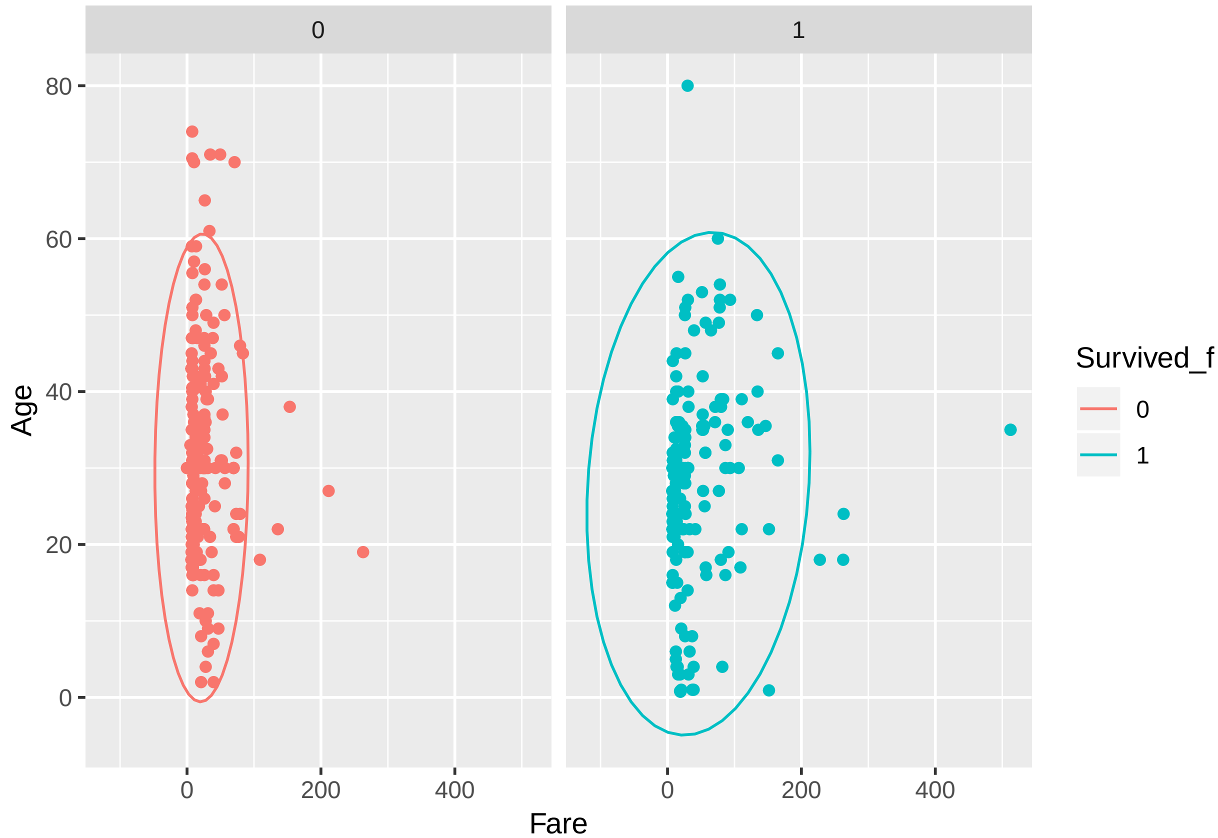
Contour plot of the Age and Fare

```r
train  %>%
  ggplot(aes(Fare, Age, fill = Survived_f, color=Survived_f)) +
  geom_point(show.legend = FALSE) +
  stat_ellipse(type="norm", lwd = 1.5)
```



Faceted contour plot of the Age and Fare

```
train %>%
  ggplot(aes(Fare, Age, fill = Survived_f, color=Survived_f)) +
  geom_point(show.legend = FALSE) +
  stat_ellipse(type="norm") +
  facet_wrap(~Survived_f)
```



We notice that both conditional distributions are not bivariate normal yet we fit QDA that assumes the conditional distributions are bivariate normal

```
train_qda <- train(Survived_f~Fare + Age ,
          method= "qda", data = train)
```

QDA model predictions on the validation set

```
fit_qda <- predict(train_qda, validation, type="raw")
fit_qda
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
##  [36] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
##  [71] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
## [106] 0 1 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 1
## [141] 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0
## [176] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0
## [211] 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [246] 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1
## [281] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
## [316] 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1
## [351] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1
## [386] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
## [421] 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0
## Levels: 0 1
```

Accuracy achieved with the QDA model

```
acc_qda <- confusionMatrix(predict(train_qda, validation),
                           validation$Survived_f)$overall["Accuracy"]
acc_qda
```

```
##  Accuracy
## 0.6816143
```

QDA predictions on the test set

```
fit_qda_t <- predict(train_qda, titanic_test)
fit_qda_t
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
##  [36] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1
##  [71] 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
## [106] 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [141] 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [176] 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
## [211] 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 1 0 0
## [246] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0
## [281] 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1
## [316] 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0
## [351] 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0
## [386] 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0
## Levels: 0 1
```

Adding to the data frame the accuracy achieved with the QDA model

```
df4 <-  bind_rows(df3, data_frame(Model="QDA", Accuracy=acc_qda))
df4 %>% knitr::kable() %>% kable_styling("striped" , full_width = T)
```

Model	Accuracy
Naive Bayes	0.6636771
Naive Best Cutoff	0.6995516
F1 Balancing	0.6748879
QDA	0.6816143

3.4.5. LDA (Linear Discriminant Analysis)

Fare = a. Age + b

We assume that the correlation structure is the same for both classes of survival

```
params <- params %>% mutate(Fare_sd = mean(sd_1), Age_sd = mean(sd_2), r=mean(r))
params
```

```
##      avg_1    avg_2     sd_1     sd_2          r  Fare_sd   Age_sd
## 1 31.40232 29.18933 49.41233 12.82086 0.05919878 49.41233 12.82086
```

LDA fit that assumes the conditional distributions are bivariate normal

```r
train_lda <- train(Survived_f~Fare + Age,
                   method= "lda", data = train)
train_lda
```

```
## Linear Discriminant Analysis
##
## 445 samples
##   2 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 445, 445, 445, 445, 445, 445, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.6347344  0.1470033
```

LDA model predictions on the validation set

```r
fit_lda <- predict(train_lda, validation, type="raw")
fit_lda
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
##  [36] 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
##  [71] 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
## [106] 0 1 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
## [141] 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0
## [176] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0 0 0
## [211] 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [246] 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1
## [281] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0
## [316] 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1
## [351] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1
## [386] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [421] 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
## Levels: 0 1
```

Accuracy achieved with the LDA model

```r
acc_lda <- confusionMatrix(predict(train_lda, validation),
                           validation$Survived_f)$overall["Accuracy"]
acc_lda
```

```
## Accuracy
##  0.67713
```

LDA predictions on the test set

```r
fit_lda_t <- predict(train_lda, titanic_test)
fit_lda_t
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0
##  [36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1
##  [71] 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [106] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [141] 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [176] 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
## [211] 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0
## [246] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
## [281] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
## [316] 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
## [351] 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
## [386] 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0
## Levels: 0 1
```

Adding to the data frame the accuracy achieved with the LDA model

```r
df5 <-  bind_rows(df4, data_frame(Model="LDA", Accuracy=acc_lda))
df5 %>% knitr::kable() %>% kable_styling("striped" , full_width = T)
```

Model	Accuracy
Naive Bayes	0.6636771
Naive Best Cutoff	0.6995516
F1 Balancing	0.6748879
QDA	0.6816143
LDA	0.6771300

3.4.6. Linear Regression

Survived = a + b.Fare + c.Age

Fitting a linear regression model

```r
lm_fit <- mutate(train, y=as.numeric(Survived_f == "1")) %>% lm(y~Fare + Age, data = .)
lm_fit
```

```
##
## Call:
## lm(formula = y ~ Fare + Age, data = .)
##
## Coefficients:
## (Intercept)         Fare          Age
##    0.427683     0.002469    -0.003604
```

Probability of survival prediction on the validation set

```r
p_hat <- predict(lm_fit, validation, type="response")
p_hat
```

```
##         1         2         3         4         5         6         7
## 0.3404323 0.4544953 0.2841740 0.4923798 0.3603850 0.3175630 0.4144035
##         8         9        10        11        12        13        14
## 0.3373874 0.3390435 0.3519444 0.2157132 0.5296274 0.3718642 0.4072427
##        15        16        17        18        19        20        21
## 0.4049751 0.3068980 0.3390435 0.5195231 0.3578173 0.3675191 0.3730729
##        22        23        24        25        26        27        28
## 0.3464129 0.4071946 0.3428044 0.4781718 0.5038240 0.3793404 0.3859733
##        29        30        31        32        33        34        35
## 0.3553541 0.3382641 0.4858018 0.5334514 0.3696531 0.3390435 0.4962884
##        36        37        38        39        40        41        42
## 0.3503490 0.3676116 0.4430423 0.3923306 0.3478681 0.4548792 0.9940927
##        43        44        45        46        47        48        49
## 0.3430287 0.4129117 0.3847645 0.5012041 0.3390435 0.3301026 0.3138845
##        50        51        52        53        54        55        56
## 0.3462524 0.3102080 0.3866554 0.4111041 0.3682597 0.4021031 0.3715555
##        57        58        59        60        61        62        63
## 0.3750005 0.9522722 0.4976879 0.3394242 0.4238558 0.3282302 0.2940730
##        64        65        66        67        68        69        70
## 0.3873448 0.3819270 0.4240886 0.3496099 0.4801103 0.2747792 0.5128118
##        71        72        73        74        75        76        77
## 0.3376034 0.3953937 0.3394242 0.3409364 0.5849749 0.3088365 0.3877941
##        78        79        80        81        82        83        84
## 0.5220617 0.3343646 0.4851710 0.4515652 0.3715555 0.3016183 0.3821941
##        85        86        87        88        89        90        91
## 0.4761391 0.3516451 0.2979232 0.3567063 0.4676420 0.4429932 0.3217729
##        92        93        94        95        96        97        98
## 0.3174169 0.3912939 0.4810602 0.3375267 0.5803658 0.2970454 0.3732717
##        99       100       101       102       103       104       105
## 0.3502129 0.3211694 0.2815186 0.3826775 0.4463046 0.3891457 0.3585820
##       106       107       108       109       110       111       112
## 0.3516451 0.5956062 0.3499289 0.5006950 0.3454729 0.3898863 0.3390435
##       113       114       115       116       117       118       119
## 0.5129123 0.3714692 0.3851218 0.3716913 0.3948984 0.4112592 0.5076163
##       120       121       122       123       124       125       126
## 0.3423497 0.4871523 0.3670996 0.3332782 0.3866528 0.3490774 0.3659760
##       127       128       129       130       131       132       133
## 0.4912857 0.3567675 0.3489848 0.2697562 0.3607941 0.4952596 0.4368974
##       134       135       136       137       138       139       140
## 0.2835055 0.3675191 0.5975040 0.3375720 0.3280439 0.3676441 0.3930720
##       141       142       143       144       145       146       147
## 0.2846170 0.3195498 0.4743576 0.3515222 0.2125282 0.3790730 0.3301231
##       148       149       150       151       152       153       154
## 0.3678790 0.5286377 0.3297099 0.3630259 0.3879888 0.7946308 0.8585568
##       155       156       157       158       159       160       161
## 0.3675191 0.3591987 0.3788026 0.4601004 0.5464832 0.4053670 0.2676075
##       162       163       164       165       166       167       168
## 0.3498568 0.4199825 0.6327837 0.2232118 0.3666189 0.4057865 0.3340437
##       169       170       171       172       173       174       175
## 0.4144516 0.6296894 0.3390435 0.4875808 0.6119635 0.2853577 0.4846646
##       176       177       178       179       180       181       182
## 0.3300185 0.3394742 0.3675562 0.4059602 0.3914643 0.3502129 0.4841730
##       183       184       185       186       187       188       189
## 0.3228096 0.3678380 0.3915932 0.3011688 0.3374491 0.3175734 0.3675191
```

```
##       190       191       192       193       194       195       196
## 0.5122689 0.3790730 0.4689010 0.5025941 0.3655005 0.3783941 0.8380254
##       197       198       199       200       201       202       203
## 0.4629427 0.3390435 0.5442648 0.5398684 0.3386322 0.3960339 0.3712366
##       204       205       206       207       208       209       210
## 0.6244417 0.3824065 0.3353364 0.3707040 0.3538420 0.3762464 0.2629904
##       211       212       213       214       215       216       217
## 0.3364821 0.5309345 0.3195998 0.3853702 0.4512617 0.3425966 0.3623104
##       218       219       220       221       222       223       224
## 0.3386836 0.3394742 0.3404871 0.6734848 0.3874677 0.8463108 0.3302910
##       225       226       227       228       229       230       231
## 0.3790485 0.3567675 0.3395785 0.4572070 0.3688450 0.3880609 0.4710519
##       232       233       234       235       236       237       238
## 0.3394242 0.3426480 0.2733842 0.3386836 0.3202184 0.3250065 0.3195498
##       239       240       241       242       243       244       245
## 0.3386218 0.3374491 0.3121009 0.3772476 0.4479309 0.3569783 0.3405494
##       246       247       248       249       250       251       252
## 0.4508103 0.3195498 0.2242737 0.5624344 0.2919509 0.3394242 0.4344980
##       253       254       255       256       257       258       259
## 0.3047393 0.3718642 0.3568297 0.7117287 0.4734482 0.3422880 0.3394242
##       260       261       262       263       264       265       266
## 0.3628235 0.3796940 0.3596826 0.3422672 0.3791730 0.3621137 0.3373874
##       267       268       269       270       271       272       273
## 0.4122306 0.2733842 0.8670930 0.3731729 0.4846646 0.3409364 0.4672596
##       274       275       276       277       278       279       280
## 0.3310317 0.3553484 0.4705942 0.4724568 0.5102102 0.2611895 0.6401636
##       281       282       283       284       285       286       287
## 0.3945537 0.3662229 0.3675808 0.2697562 0.8812786 0.4007996 0.3786923
##       288       289       290       291       292       293       294
## 0.3751856 0.2301309 0.3630704 0.3372274 0.4251195 0.3354210 0.4116380
##       295       296       297       298       299       300       301
## 0.2972339 0.3969865 0.3410598 0.4069509 0.3191183 0.3575463 0.2510655
##       302       303       304       305       306       307       308
## 0.2889621 0.3363141 0.3390435 0.4056636 0.6623838 0.3643237 0.3369554
##       309       310       311       312       313       314       315
## 0.3866528 0.5016528 0.3406838 0.3737163 0.3598907 0.3660486 0.2876080
##       316       317       318       319       320       321       322
## 0.2527205 0.5444584 0.3534613 0.2612725 0.3876414 0.4641244 0.3749852
##       323       324       325       326       327       328       329
## 0.5122689 0.4893555 0.4590304 0.4441041 0.3134792 0.3381898 0.3390435
##       330       331       332       333       334       335       336
## 0.4195870 0.3677146 0.3794679 0.3768762 0.4982864 0.3013430 0.3164293
##       337       338       339       340       341       342       343
## 0.3751600 0.3387453 0.3797912 0.3195498 0.3819985 0.3871009 1.5627954
##       344       345       346       347       348       349       350
## 0.3077025 0.3820499 0.8953804 0.4566708 0.4590304 0.2769651 0.5248216
##       351       352       353       354       355       356       357
## 0.2951834 0.3984886 0.3533586 0.3513004 0.3411969 0.4509051 0.4633957
##       358       359       360       361       362       363       364
## 0.3850982 0.3830510 0.3780855 0.8524431 0.3562863 0.3578173 0.3279320
##       365       366       367       368       369       370       371
## 0.2795564 0.4017626 0.8449183 0.3195498 0.3768762 0.3647052 0.3395372
##       372       373       374       375       376       377       378
## 0.3390435 0.4925933 0.9997585 0.3809252 0.4902955 0.3350791 0.4700491
```

```
##       379       380       381       382       383       384       385 
## 0.3321908 0.4610666 0.3911951 0.3250065 0.2974917 0.3734421 0.5941872 
##       386       387       388       389       390       391       392 
## 0.3892074 0.3368813 0.3329869 0.2740609 0.3373874 0.2898273 0.3386527 
##       393       394       395       396       397       398       399 
## 0.7944561 0.3986732 0.3972203 0.3813092 0.4707532 0.3953337 0.3192052 
##       400       401       402       403       404       405       406 
## 0.3150813 0.3791730 0.3807531 0.6842981 0.4471957 0.2871099 0.3192052 
##       407       408       409       410       411       412       413 
## 0.4398333 0.3467330 0.3376220 0.3643467 0.4073038 0.3517497 0.2907143 
##       414       415       416       417       418       419       420 
## 0.3611623 0.3367085 0.4017171 0.3641719 0.3733763 0.3928752 0.3959351 
##       421       422       423       424       425       426       427 
## 0.3192232 0.2949365 0.3210213 0.4082313 0.5197118 0.4904790 0.4328829 
##       428       429       430       431       432       433       434 
## 0.4672853 0.3858870 0.3887226 0.3147325 0.3186857 0.5200950 0.3083919 
##       435       436       437       438       439       440       441 
## 0.3645774 0.4407519 0.3534613 0.3880237 0.3786923 0.3390435 0.4311412 
##       442       443       444       445       446 
## 0.3743497 0.3549775 0.3624584 0.4080336 0.3314747
```

Prediction of survival on the validation set if the probability is greater than 0.5

```r
y_hat_lr <- ifelse(p_hat > 0.5, "1", "0") %>% factor()
```

Accuracy achieved on the validation set with the linear regression model

```r
acc_lr <- confusionMatrix(y_hat_lr, validation$Survived_f)$overall["Accuracy"]
acc_lr
```

```
##  Accuracy 
## 0.6793722
```

Probability of survival prediction on the test set

```r
p_hat_lr_t <- predict(lm_fit, titanic_test, type="response")
p_hat_lr_t
```

```
##         1         2         3         4         5         6         7 
## 0.3226591 0.2755565 0.2281250 0.3517497 0.3787215 0.3999961 0.3383853 
##         8         9        10        11        12        13        14 
## 0.4055647 0.3806510 0.4116129 0.3390435 0.3260694 0.5478864 0.2647939 
##        15        16        17        18        19        20        21 
## 0.4093073 0.4096154 0.3320182 0.3698273 0.3499289 0.2833208 0.3760896 
##        22        23        24        25        26        27        28 
## 0.4030713 0.3779473 0.5035268 0.9024387 0.2832597 0.5014037 0.3644207 
##        29        30        31        32        33        34        35 
## 0.3552014 0.3730729 0.3116516 0.4189458 0.3595335 0.3576203 0.4621269 
##        36        37        38        39        40        41        42 
## 0.3788488 0.3682597 0.3733763 0.3610263 0.4590304 0.3202339 0.3850982 
##        43        44        45        46        47        48        49 
## 0.2992816 0.3516451 0.3952326 0.3571378 0.3388086 0.3386836 0.3997707
```

```
##        50        51        52        53        54        55        56
## 0.3371782 0.4893084 0.3674784 0.4123781 0.9760705 0.3580128 0.4635443
##        57        58        59        60        61        62        63
## 0.3210213 0.3564589 0.3592986 0.9456920 0.3859012 0.3456707 0.3819368
##        64        65        66        67        68        69        70
## 0.3674573 1.0285940 0.3515716 0.3822558 0.3629543 0.3864007 0.8607285
##        71        72        73        74        75        76        77
## 0.3603102 0.3714835 0.3427200 0.3933954 0.8236928 0.8327039 0.3394242
##        78        79        80        81        82        83        84
## 0.2928888 0.3516451 0.3603102 0.4436962 0.7337288 0.3152560 0.3390435
##        85        86        87        88        89        90        91
## 0.3459872 0.3552353 0.3498158 0.3826775 0.3675191 0.4772580 0.3827026
##        92        93        94        95        96        97        98
## 0.3387453 0.4587442 0.3394242 0.4017626 0.3568188 0.3484158 0.3427200
##        99       100       101       102       103       104       105
## 0.3749852 0.3286109 0.4095703 0.3945537 0.3386836 0.3531630 0.3910386
##       106       107       108       109       110       111       112
## 0.3823699 0.3712983 0.3386836 0.3410598 0.3930961 0.3170471 0.3675912
##       113       114       115       116       117       118       119
## 0.3761349 0.3789824 0.7481466 0.3984886 0.3354432 0.4653086 0.4836849
##       120       121       122       123       124       125       126
## 0.3873448 0.4233144 0.3386836 0.4441047 0.3446580 0.3386836 0.4061563
##       127       128       129       130       131       132       133
## 0.3676321 0.4057865 0.3083919 0.3610508 0.3322153 0.3070105 0.3625993
##       134       135       136       137       138       139       140
## 0.3354432 0.2921858 0.3605674 0.3500029 0.3660629 0.3646553 0.3992953
##       141       142       143       144       145       146       147
## 0.5074284 0.6828932 0.8555810 0.3909492 0.3418450 0.3603850 0.4475914
##       148       149       150       151       152       153       154
## 0.3682597 0.3850982 0.3837404 0.5500876 0.3390435 0.2294889 0.3280022
##       155       156       157       158       159       160       161
## 0.4583168 0.3598164 0.8706974 0.3641719 0.3418450 0.3679762 0.3674778
##       162       163       164       165       166       167       168
## 0.4400918 0.3672973 0.3368319 0.3119963 0.3883444 0.3793373 0.4476704
##       169       170       171       172       173       174       175
## 0.3681644 0.3705126 0.3381898 0.3482007 0.3790981 0.3373978 0.3609970
##       176       177       178       179       180       181       182
## 0.4699022 0.4463251 0.3699215 0.3942090 0.4023057 0.3516451 0.4996255
##       183       184       185       186       187       188       189
## 0.4938999 0.3386836 0.9414589 0.3230073 0.4038360 0.3862819 0.5200950
##       190       191       192       193       194       195       196
## 0.3156007 0.3693226 0.3837404 0.4220305 0.2383028 0.4790856 0.3281275
##       197       198       199       200       201       202       203
## 0.7381188 0.3819985 0.3707040 0.3197541 0.3866528 0.4620452 0.8200032
##       204       205       206       207       208       209       210
## 0.4630380 0.3634951 0.3831027 0.3206614 0.3670996 0.3771755 0.3570657
##       211       212       213       214       215       216       217
## 0.3679522 0.3369554 0.5478692 0.2756072 0.3099098 0.3812234 0.3678380
##       218       219       220       221       222       223       224
## 0.6292638 0.7696263 0.3394242 0.3537641 0.3718642 0.3743084 0.3712366
##       225       226       227       228       229       230       231
## 0.3044078 0.3373653 0.3640277 0.3675191 0.3189831 0.3300185 0.5376971
##       232       233       234       235       236       237       238
## 0.4175382 0.3680270 0.3390025 0.4630988 0.3749852 0.3827811 0.3734318
```

```
##       239       240       241       242       243       244       245 
## 0.3948984 0.5174190 0.2978778 0.3395493 0.5975458 0.3390230 0.4711601 
##       246       247       248       249       250       251       252 
## 0.4079426 0.4002315 0.3565347 0.3873448 0.3354109 0.4928779 0.3751600 
##       253       254       255       256       257       258       259 
## 0.6680527 0.3641986 0.3339930 0.3381898 0.3386836 0.3466331 0.3912939 
##       260       261       262       263       264       265       266 
## 0.3711852 0.3390793 0.3713808 0.3799382 0.4541575 0.3509970 0.3390435 
##       267       268       269       270       271       272       273 
## 0.3195498 0.3381898 0.3682597 0.3877941 0.4476405 0.3386836 0.6716571 
##       274       275       276       277       278       279       280 
## 0.3866528 0.3373874 0.4197847 0.3526818 0.3476960 0.3713960 0.3743084 
##       281       282       283       284       285       286       287 
## 0.3661675 0.4589883 0.3675191 0.4328829 0.4703761 0.3158225 0.3374491 
##       288       289       290       291       292       293       294 
## 0.5442819 0.3373978 0.3394242 0.4173169 0.3367085 0.3373978 0.4387450 
##       295       296       297       298       299       300       301 
## 0.3213774 0.3534613 0.5267320 0.3730729 0.4318833 0.3425453 0.3315364 
##       302       303       304       305       306       307       308 
## 0.3566959 0.3245383 0.3625630 0.3675191 0.2625474 0.6937065 0.4477752 
##       309       310       311       312       313       314       315 
## 0.4602779 0.3003148 0.3841897 0.3662229 0.3382515 0.3134525 0.5642994 
##       316       317       318       319       320       321       322 
## 0.3855000 0.5839703 0.3851218 0.3497541 0.4261546 0.3531630 0.3554200 
##       323       324       325       326       327       328       329 
## 0.3660629 0.3742849 0.8088739 0.3621864 0.4807155 0.4574130 0.3873448 
##       330       331       332       333       334       335       336 
## 0.3840850 0.3454008 0.3604352 0.3373874 0.3980628 0.3498568 0.3837404 
##       337       338       339       340       341       342       343 
## 0.3444362 0.3049578 0.4153413 0.4311132 0.3887262 0.3310530 0.4912595 
##       344       345       346       347       348       349       350 
## 1.4834978 0.4490638 0.3888988 0.3660629 0.3085623 0.3745062 0.3677916 
##       351       352       353       354       355       356       357 
## 0.4219064 0.3634951 0.5442648 0.4115418 0.4778671 0.3116516 0.3421164 
##       358       359       360       361       362       363       364 
## 0.3390025 0.3386836 0.3579407 0.5471283 0.4325349 0.3677916 0.3517497 
##       365       366       367       368       369       370       371 
## 0.4744502 0.4714351 0.3552455 0.4463685 0.4121339 0.3573686 0.3803817 
##       372       373       374       375       376       377       378 
## 0.6480079 0.2510655 0.3011830 0.4351405 0.9132520 0.3697719 0.3803817 
##       379       380       381       382       383       384       385 
## 0.3528822 0.4871523 0.3386836 0.3534203 0.3355240 0.3989474 0.3513365 
##       386       387       388       389       390       391       392 
## 0.5016528 0.3603719 0.2543253 0.3711235 0.4580877 0.5756199 0.3411300 
##       393       394       395       396       397       398       399 
## 0.4308198 0.2841975 0.3775311 0.5109351 0.3590758 0.4502042 0.3675808 
##       400       401       402       403       404       405       406 
## 0.3350379 0.7265837 0.3425605 0.4950360 0.4826911 0.3411311 0.3898189 
##       407       408       409       410       411       412       413 
## 0.3707040 0.7696263 0.3674470 0.4508783 0.3675191 0.5165167 0.3459542 
##       414       415       416       417       418 
## 0.3394242 0.5559694 0.3068114 0.3394242 0.4684649
```

Prediction of survival on the test set if the probability is greater than 0.5

```r
y_hat_lr_t <- ifelse(p_hat_lr_t > 0.5, "1", "0")
y_hat_lr_t
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18 
## "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" 
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 
## "0" "0" "0" "0" "0" "1" "1" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" 
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 
## "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" 
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 
## "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1" "0" "0" 
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90 
## "0" "0" "1" "1" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" 
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 
## "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" 
## 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 
## "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" 
## 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 
## "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "1" "1" "0" 
## 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 
## "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" 
## 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 
## "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" 
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 
## "0" "0" "0" "0" "1" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "1" "0" 
## 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 
## "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" 
## 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 
## "0" "1" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" 
## 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 
## "0" "0" "0" "0" "0" "1" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" 
## 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 
## "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" 
## 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 
## "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" 
## 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 
## "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" 
## 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 
## "1" "0" "0" "0" "0" "0" "0" "0" "1" "0" "1" "0" "0" "0" "0" "0" "0" "0" 
## 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 
## "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" 
## 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 
## "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" 
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 
## "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "1" "0" "0" 
## 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 
## "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1" 
## 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 
## "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "1" "0" "0" 
## 415 416 417 418 
## "1" "0" "0" "0"
```

Adding to the data frame the accuracy achieved with the linear regression model

```r
df6 <-  bind_rows(df5, data_frame(Model="Linear Regression", Accuracy=acc_lr))
df6 %>% knitr::kable() %>% kable_styling("striped" , full_width = T)
```

Model	Accuracy
Naive Bayes	0.6636771
Naive Best Cutoff	0.6995516
F1 Balancing	0.6748879
QDA	0.6816143
LDA	0.6771300
Linear Regression	0.6793722

Note that this model has the risk of predicting negative probabilities the reason why we develop a logistic model where the log of odds is modeled by a linear regression

3.4.7. Logistic Regression

g(p(Age, Class_f, Embarked_f, Fare, Sex_f, Title_f)) = a + b.Age + c.Class_f + d.Embarked_f + e.Fare + f.Sex_f + g.Title_f where g(p) = log(p/1-p)

Fitting a logistic regression model

```r
glm_fit <- train %>% mutate (y = as.numeric(Survived_f == "1")) %>%
  glm(y ~ Age + Class_f + Embarked_f + Fare + Sex_f + Title_f, data = .,
      family = "binomial")
glm_fit
```

```
##
## Call:  glm(formula = y ~ Age + Class_f + Embarked_f + Fare + Sex_f +
##     Title_f, family = "binomial", data = .)
##
## Coefficients:
##   (Intercept)            Age       Class_f2       Class_f3    Embarked_fQ
##      3.892116      -0.015734      -1.028141      -2.411670       0.073233
##   Embarked_fS           Fare      Sex_fmale  Title_fMaster    Title_fMiss
##     -0.668653      -0.000851      -2.705267       2.178105      -0.413190
##     Title_fMr     Title_fMrs     Title_fRev
##     -0.338119             NA     -13.287277
##
## Degrees of Freedom: 444 Total (i.e. Null);  433 Residual
## Null Deviance:       599
## Residual Deviance: 379.8     AIC: 403.8
```

Probability of surviving given a paid fare after logistic smoothing on the validation set

```r
p_hat_logit <- predict(glm_fit, newdata = validation, type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```r
p_hat_logit
```

```
##            1             2             3             4             5 
## 1.225062e-01 5.796795e-01 8.670515e-01 7.251726e-01 5.765770e-01 
##            6             7             8             9            10 
## 7.141771e-01 4.278400e-01 1.149555e-01 6.236737e-02 5.487945e-01 
##           11            12            13            14            15 
## 1.306290e-01 5.837698e-01 7.117057e-02 5.249998e-01 6.979216e-01 
##           16            17            18            19            20 
## 5.434874e-01 1.148975e-01 9.143702e-01 1.218635e-01 6.873495e-01 
##           21            22            23            24            25 
## 1.137100e-01 4.435692e-01 4.201552e-01 1.173782e-01 8.428847e-01 
##           26            27            28            29            30 
## 5.179056e-01 7.327272e-02 5.310573e-01 6.611026e-02 2.041640e-01 
##           31            32            33            34            35 
## 5.267137e-01 2.242599e-01 1.208601e-01 6.236737e-02 8.362277e-01 
##           36            37            38            39            40 
## 4.790079e-01 6.873426e-01 4.254254e-01 8.184272e-01 5.693255e-01 
##           41            42            43            44            45 
## 7.498098e-02 9.024369e-01 6.328601e-02 3.553233e-01 6.548714e-02 
##           46            47            48            49            50 
## 6.065766e-01 6.236737e-02 5.962735e-02 5.622770e-02 6.423302e-02 
##           51            52            53            54            55 
## 5.539989e-02 3.535098e-01 6.956821e-01 7.013749e-02 6.872905e-01 
##           56            57            58            59            60 
## 7.117760e-02 2.104518e-01 5.647582e-01 5.843173e-01 6.235969e-02 
##           61            62            63            64            65 
## 3.240328e-01 1.101839e-01 5.149904e-01 8.477325e-01 3.649175e-01 
##           66            67            68            69            70 
## 9.233650e-01 6.519041e-02 5.566925e-01 4.516815e-07 9.437817e-01 
##           71            72            73            74            75 
## 6.239640e-02 4.985227e-01 6.235969e-02 6.232923e-02 5.405391e-01 
##           76            77            78            79            80 
## 5.032745e-02 7.540844e-02 5.585121e-01 5.200003e-01 7.188571e-01 
##           81            82            83            84            85 
## 5.922792e-01 7.117760e-02 4.853876e-01 7.436610e-02 5.498405e-01 
##           86            87            88            89            90 
## 2.089725e-01 5.743245e-02 3.397919e-01 5.785750e-01 4.171521e-01 
##           91            92            93            94            95 
## 1.060029e-01 5.706981e-02 2.390197e-01 8.318555e-01 9.599308e-01 
##           96            97            98            99           100 
## 9.199523e-01 5.217578e-02 8.011449e-01 6.415102e-02 5.885288e-02 
##          101           102           103           104           105 
## 9.237610e-02 7.435463e-02 5.886126e-01 7.072692e-01 6.816686e-02 
##          106           107           108           109           110 
## 2.089725e-01 9.475885e-01 4.917644e-01 9.483646e-01 2.093244e-01 
##          111           112           113           114           115 
## 7.654960e-02 6.236737e-02 3.788619e-01 7.006538e-02 2.394068e-01 
##          116           117           118           119           120 
## 7.173768e-02 2.418932e-01 5.076980e-01 9.309745e-01 6.329988e-02 
##          121           122           123           124           125 
## 5.727857e-01 2.253770e-01 1.732937e-01 6.859305e-01 2.119403e-01 
##          126           127           128           129           130 
## 7.018885e-02 6.203942e-01 4.996623e-01 5.857973e-01 3.061900e-01 
##          131           132           133           134           135 
## 7.332551e-01 9.632502e-01 3.305180e-01 3.895362e-01 6.873495e-01 
```

```
##           136          137          138          139          140
## 8.984691e-01 6.755113e-02 8.225405e-01 5.600383e-01 8.522959e-01
##           141          142          143          144          145
## 4.216510e-01 2.108070e-01 7.092200e-01 5.606717e-01 7.453838e-02
##           146          147          148          149          150
## 7.327898e-02 1.101200e-01 7.014605e-02 9.116624e-01 3.193475e-01
##           151          152          153          154          155
## 5.033669e-01 5.873779e-01 9.340013e-01 9.475893e-01 6.873495e-01
##           156          157          158          159          160
## 7.374554e-02 3.380854e-01 9.506626e-01 9.539357e-01 8.576117e-01
##           161          162          163          164          165
## 2.023830e-01 6.518523e-02 8.611061e-01 9.425190e-01 3.929223e-02
##           166          167          168          169          170
## 5.760524e-01 6.845080e-01 3.635362e-01 7.595321e-02 9.276536e-01
##           171          172          173          174          175
## 6.236737e-02 4.175430e-01 9.381616e-01 4.990457e-02 8.340447e-01
##           176          177          178          179          180
## 1.937959e-01 5.596036e-01 6.905399e-02 4.202589e-01 1.412322e-01
##           181          182          183          184          185
## 6.415102e-02 9.181395e-01 7.637223e-01 6.873259e-01 3.409034e-01
##           186          187          188          189          190
## 6.813838e-01 6.239951e-02 7.141763e-01 6.873495e-01 9.693959e-01
##           191          192          193          194          195
## 7.327898e-02 5.826030e-01 9.631606e-01 1.322713e-01 7.329488e-02
##           196          197          198          199          200
## 9.324220e-01 7.384884e-01 6.236737e-02 2.325780e-01 5.569981e-01
##           201          202          203          204          205
## 1.225729e-01 8.978081e-01 7.118487e-02 9.534989e-01 6.034780e-01
##           206          207          208          209          210
## 4.760635e-01 2.930225e-01 6.614244e-02 5.149552e-01 4.562523e-02
##           211          212          213          214          215
## 1.226526e-01 9.505635e-01 5.612911e-01 8.365429e-01 5.549564e-01
##           216          217          218          219          220
## 6.329484e-02 5.887987e-01 1.225710e-01 5.596036e-01 8.194152e-01
##           221          222          223          224          225
## 9.232884e-01 8.583634e-01 2.590966e-01 8.122927e-01 7.213505e-02
##           226          227          228          229          230
## 6.713558e-02 6.235658e-02 7.255578e-01 6.176939e-02 5.873719e-01
##           231          232          233          234          235
## 5.004912e-01 6.235969e-02 1.165072e-01 7.284004e-01 1.225710e-01
##           236          237          238          239          240
## 3.548648e-01 5.877967e-02 2.108070e-01 1.225733e-01 6.239951e-02
##           241          242          243          244          245
## 5.536576e-02 8.392290e-01 4.167384e-01 1.976778e-01 6.333668e-02
##           246          247          248          249          250
## 5.882364e-01 2.108070e-01 4.532430e-01 5.933637e-01 4.777432e-01
##           251          252          253          254          255
## 6.235969e-02 5.323298e-01 3.293295e-01 7.117057e-02 6.200983e-02
##           256          257          258          259          260
## 9.370952e-01 6.364114e-02 1.242731e-01 6.235969e-02 3.992244e-01
##           261          262          263          264          265
## 9.522096e-01 6.814276e-02 3.570142e-01 1.210779e-01 8.329654e-01
##           266          267          268          269          270
## 1.149555e-01 9.589286e-01 7.284004e-01 3.820980e-01 2.279859e-01
```

```
##          271          272          273          274          275
## 8.492256e-01 4.798176e-01 8.388451e-01 3.657431e-01 6.203953e-02
##          276          277          278          279          280
## 9.565061e-01 5.573435e-01 4.928735e-01 2.996445e-01 6.193768e-01
##          281          282          283          284          285
## 2.150090e-01 1.283951e-01 5.114604e-01 3.061900e-01 5.456502e-01
##          286          287          288          289          290
## 6.724834e-02 7.328789e-02 5.836044e-01 1.379437e-01 3.992040e-01
##          291          292          293          294          295
## 7.748898e-01 9.283947e-01 7.129190e-01 7.962900e-01 1.518973e-01
##          296          297          298          299          300
## 5.617284e-01 1.148268e-01 4.594205e-01 5.795820e-02 5.633003e-02
##          301          302          303          304          305
## 4.731155e-02 5.065587e-02 5.672031e-02 6.236737e-02 4.327399e-01
##          306          307          308          309          310
## 8.859822e-01 5.427883e-01 6.240947e-02 6.859305e-01 7.940014e-01
##          311          312          313          314          315
## 5.848143e-02 5.960472e-01 2.199310e-01 1.191983e-01 3.084940e-01
##          316          317          318          319          320
## 8.631692e-07 9.178532e-01 6.615055e-02 4.565102e-02 6.586812e-01
##          321          322          323          324          325
## 5.580519e-01 7.222885e-02 9.544511e-01 5.850147e-01 5.999225e-02
##          326          327          328          329          330
## 5.070568e-01 4.843665e-01 6.238458e-02 6.236737e-02 8.144754e-01
##          331          332          333          334          335
## 6.873350e-01 7.008905e-01 2.277613e-01 4.600342e-01 9.989179e-02
##          336          337          338          339          340
## 5.708813e-02 7.222481e-02 6.237338e-02 8.224149e-01 2.108070e-01
##          341          342          343          344          345
## 7.437074e-02 5.267307e-01 4.616634e-01 1.388039e-01 7.436952e-02
##          346          347          348          349          350
## 9.164016e-01 4.118881e-01 5.999225e-02 3.129153e-01 4.958580e-01
##          351          352          353          354          355
## 5.220753e-02 6.838999e-01 6.615274e-02 1.848550e-01 3.768127e-01
##          356          357          358          359          360
## 7.062783e-01 9.551779e-01 4.220118e-01 3.499224e-01 7.330210e-02
##          361          362          363          364          365
## 9.362813e-01 7.938667e-01 1.218635e-01 5.966931e-02 1.616774e-01
##          366          367          368          369          370
## 2.203676e-01 8.979051e-01 2.108070e-01 2.277613e-01 6.334629e-02
##          371          372          373          374          375
## 5.068329e-01 6.236737e-02 3.885464e-01 9.490807e-01 6.767930e-02
##          376          377          378          379          380
## 4.590806e-01 1.208889e-01 8.454859e-01 5.958700e-02 8.382522e-01
##          381          382          383          384          385
## 2.421273e-01 5.877967e-02 5.301398e-02 1.319575e-01 9.279048e-01
##          386          387          388          389          390
## 7.656614e-02 6.579163e-01 6.052981e-02 4.772113e-02 1.149555e-01
##          391          392          393          394          395
## 7.902016e-01 1.225721e-01 9.142825e-01 7.019432e-01 4.251365e-01
##          396          397          398          399          400
## 5.272283e-01 7.059647e-01 5.867642e-01 1.865278e-01 9.191205e-01
##          401          402          403          404          405
## 5.791384e-01 8.435980e-01 9.184235e-01 7.189843e-01 3.932840e-01
```

```
##          406          407          408          409          410 
## 1.865278e-01 9.345679e-01 5.387290e-02 6.190130e-02 5.074965e-01 
##          411          412          413          414          415 
## 3.321144e-01 6.514548e-02 3.970445e-01 5.929980e-01 1.226442e-01 
##          416          417          418          419          420 
## 9.452443e-01 6.912902e-02 7.113612e-02 5.869697e-01 2.481070e-01 
##          421          422          423          424          425 
## 1.080108e-01 5.221174e-02 1.071364e-01 6.374171e-07 9.629507e-01 
##          426          427          428          429          430 
## 5.486168e-01 7.135484e-01 9.258765e-01 6.272959e-01 7.477134e-01 
##          431          432          433          434          435 
## 5.271644e-02 9.202824e-01 4.983204e-01 8.210464e-01 8.822786e-01 
##          436          437          438          439          440 
## 5.528576e-01 6.615055e-02 9.197730e-01 7.328789e-02 6.236737e-02 
##          441          442          443          444          445 
## 9.498019e-01 5.108774e-01 6.717423e-02 6.586399e-07 6.020780e-01 
##          446 
## 1.192268e-01
```

Predictions of survival on the validation set with the logistic model

```r
y_hat_logit <- ifelse(p_hat_logit > 0.5, "1", "0") %>% factor ()
y_hat_logit
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18 
##   0   1   1   1   1   1   0   0   0   1   0   1   0   1   1   1   0   1 
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 
##   0   1   0   0   0   0   1   1   0   1   0   0   1   0   0   0   1   0 
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 
##   1   0   1   1   0   1   0   0   0   1   0   0   0   0   0   0   1   0 
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 
##   1   0   0   1   1   0   0   0   1   1   0   1   0   1   0   1   0   0 
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90 
##   0   0   1   0   0   1   1   1   1   0   0   0   1   0   0   0   1   0 
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 
##   0   0   0   1   1   1   0   1   0   0   0   0   1   1   0   0   1   0 
## 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 
##   1   0   0   0   0   0   0   0   0   1   1   0   1   0   0   1   0   0 
## 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 
##   1   0   1   0   1   1   0   0   1   1   0   1   1   1   0   0   1   1 
## 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 
##   0   0   0   0   1   0   1   1   1   1   1   0   0   1   1   1   0   0 
## 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 
##   1   1   0   1   1   0   0   1   0   0   1   0   1   0   1   0   0   0 
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 
##   0   1   1   1   0   1   0   1   1   1   0   1   1   0   0   1   1   0 
## 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 
##   0   1   0   1   0   1   1   0   0   0   1   0   0   1   1   1   1   0 
## 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 
##   1   0   1   1   1   1   0   1   0   0   0   1   0   1   1   0   0   1 
## 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 
##   0   0   0   0   0   0   0   1   0   0   0   1   0   0   1   0   0   1 
## 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
```

```
##   0   0   0   1   0   0   0   0   1   0   0   0   1   0   1   1   0   0
## 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
##   1   0   1   0   0   1   1   0   0   1   0   0   1   0   1   0   0   1
## 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
##   0   0   1   1   1   1   0   1   0   0   0   0   0   0   0   0   0   1
## 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
##   1   0   1   1   0   1   0   0   0   0   1   0   0   1   1   0   1   1
## 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
##   0   1   0   0   0   1   1   1   0   0   0   0   0   0   1   0   0   1
## 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
##   0   0   0   1   0   0   0   0   0   1   0   0   0   1   1   0   0   0
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378
##   1   1   0   0   0   0   1   0   0   0   1   0   0   1   0   0   0   1
## 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396
##   0   1   0   0   0   0   1   0   1   0   0   0   1   0   1   1   0   1
## 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414
##   1   1   0   1   1   1   1   1   0   0   1   0   0   1   0   0   0   1
## 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432
##   0   1   0   0   1   0   0   0   0   0   1   1   1   1   1   1   0   1
## 433 434 435 436 437 438 439 440 441 442 443 444 445 446
##   0   1   1   1   0   1   0   0   1   1   0   0   1   0
## Levels: 0 1
```

Accuracy achieved on the validation set with the logistic model

```
acc_logit <- confusionMatrix(y_hat_logit, validation$Survived_f)$overall["Accuracy"]
acc_logit
```

```
##  Accuracy
## 0.7869955
```

Probability of survival given a paid fare after logistic smoothing on the test set

```
p_hat_logit_t <- predict(glm_fit, newdata = titanic_test, type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
p_hat_logit_t
```

```
##            1            2            3            4            5
## 1.151509e-01 5.165844e-01 2.516376e-01 6.514548e-02 6.118749e-01
##            6            7            8            9           10
## 7.873129e-02 6.597076e-01 2.172420e-01 7.669362e-01 7.027020e-02
##           11           12           13           14           15
## 6.236737e-02 3.622092e-01 9.422195e-01 1.345402e-01 9.192300e-01
##           16           17           18           19           20
## 9.214912e-01 3.390886e-01 1.301661e-01 4.917644e-01 6.827178e-01
##           21           22           23           24           25
## 4.832165e-01 5.350255e-01 9.332652e-01 6.144593e-01 9.485164e-01
##           26           27           28           29           30
```

```
## 4.606219e-02 9.560621e-01 1.275173e-01 3.796698e-01 1.137100e-01
##           31           32           33           34           35
## 1.601836e-01 2.222725e-01 5.683394e-01 5.580616e-01 5.811711e-01
##           36           37           38           39           40
## 1.346843e-01 5.114019e-01 5.152023e-01 6.704370e-02 5.999225e-02
##           41           42           43           44           45
## 1.008366e-01 4.220118e-01 5.298302e-02 8.471285e-01 9.220645e-01
##           46           47           48           49           50
## 6.712759e-02 5.288304e-01 1.225710e-01 9.470089e-01 5.577059e-01
##           51           52           53           54           55
## 4.381721e-01 3.504624e-01 8.096735e-01 8.952880e-01 3.396908e-01
##           56           57           58           59           60
## 6.993907e-01 5.792239e-02 6.714224e-02 6.196035e-02 9.363892e-01
##           61           62           63           64           65
## 7.545394e-02 2.037495e-01 7.437220e-02 6.873541e-01 9.496525e-01
##           66           67           68           69           70
## 8.346452e-01 7.006890e-01 3.553791e-01 5.833906e-01 8.865173e-01
##           71           72           73           74           75
## 6.805477e-01 7.117924e-02 4.839024e-01 5.930858e-01 9.398158e-01
##           76           77           78           79           80
## 5.392723e-01 6.235969e-02 9.118351e-01 2.089725e-01 6.805477e-01
##           81           82           83           84           85
## 6.997084e-01 2.567762e-01 3.513777e-01 6.236737e-02 3.572560e-01
##           86           87           88           89           90
## 1.143311e-01 6.701756e-01 5.271108e-01 6.873495e-01 8.343978e-01
##           91           92           93           94           95
## 6.115489e-01 6.237338e-02 9.401618e-01 6.235969e-02 6.066538e-01
##           96           97           98           99          100
## 6.713447e-02 8.765999e-01 6.329231e-02 5.193027e-01 5.965618e-02
##          101          102          103          104          105
## 9.596275e-01 2.150090e-01 1.225710e-01 6.615690e-02 7.723208e-01
##          106          107          108          109          110
## 6.348880e-02 1.386241e-01 1.225710e-01 6.232674e-02 2.404535e-01
##          111          112          113          114          115
## 3.021065e-01 6.873442e-01 9.471281e-01 6.991432e-01 8.852919e-01
##          116          117          118          119          120
## 1.348847e-01 1.150237e-01 5.911347e-01 5.543585e-01 8.477325e-01
##          121          122          123          124          125
## 8.291943e-01 1.225710e-01 9.641637e-01 6.426606e-02 1.225710e-01
##          126          127          128          129          130
## 5.293245e-01 7.015160e-02 6.845080e-01 1.794715e-01 6.811282e-02
##          131          132          133          134          135
## 6.054493e-02 4.976507e-01 5.576383e-01 1.150237e-01 5.142426e-02
##          136          137          138          139          140
## 6.812340e-02 1.206780e-01 2.195662e-01 5.074700e-01 5.211195e-02
##          141          142          143          144          145
## 5.501700e-01 8.967944e-01 4.171986e-01 2.123654e-01 3.767603e-01
##          146          147          148          149          150
## 6.095956e-02 4.167668e-01 7.013749e-02 4.220118e-01 6.213621e-07
##          151          152          153          154          155
## 9.695126e-01 1.148975e-01 3.953166e-02 5.584859e-01 5.133434e-01
##          156          157          158          159          160
## 6.813984e-02 8.970876e-01 5.075116e-01 3.767603e-01 5.965234e-01
##          161          162          163          164          165
```

```
## 6.873525e-01 6.963922e-01 7.960153e-01 6.241196e-02 5.284288e-07
##          166          167          168          169          170
## 5.948326e-01 5.126355e-01 7.282745e-02 9.647698e-01 5.112079e-01
##          171          172          173          174          175
## 6.238458e-02 1.198457e-01 6.879868e-02 1.149552e-01 5.276787e-02
##          176          177          178          179          180
## 8.194968e-01 8.078639e-01 4.879880e-01 8.314206e-01 9.434494e-01
##          181          182          183          184          185
## 2.089725e-01 5.488008e-01 9.476024e-01 1.225710e-01 9.629124e-01
##          186          187          188          189          190
## 1.837690e-01 8.075018e-01 7.544478e-02 4.983204e-01 1.841522e-01
##          191          192          193          194          195
## 1.970038e-01 4.221260e-01 5.228235e-01 2.541823e-01 8.197473e-01
##          196          197          198          199          200
## 5.966553e-02 9.591484e-01 5.271691e-01 2.281357e-01 5.612780e-01
##          201          202          203          204          205
## 6.859305e-01 5.664018e-01 4.789228e-01 8.367359e-01 2.226421e-01
##          206          207          208          209          210
## 5.877860e-01 6.418083e-01 2.253770e-01 9.496039e-01 6.712914e-02
##          211          212          213          214          215
## 5.984809e-02 6.240947e-02 2.353980e-01 7.736763e-01 4.487094e-01
##          216          217          218          219          220
## 3.735786e-01 6.873259e-01 2.979609e-01 9.490909e-01 6.235969e-02
##          221          222          223          224          225
## 8.772871e-01 7.117057e-02 8.629783e-01 7.118487e-02 9.541255e-01
##          226          227          228          229          230
## 7.127818e-01 6.913221e-02 6.873495e-01 5.306824e-02 1.937959e-01
##          231          232          233          234          235
## 2.453360e-01 9.210919e-01 7.125805e-02 1.225591e-01 5.435059e-01
##          236          237          238          239          240
## 7.222885e-02 4.446655e-01 1.319579e-01 8.157579e-01 9.546244e-01
##          241          242          243          244          245
## 9.527178e-01 8.118057e-01 5.065668e-01 6.236778e-02 5.502652e-01
##          246          247          248          249          250
## 3.753978e-01 8.619183e-01 1.770409e-01 8.477325e-01 7.129197e-01
##          251          252          253          254          255
## 8.511999e-01 7.222481e-02 5.762828e-01 6.804399e-02 6.002936e-02
##          256          257          258          259          260
## 6.238458e-02 1.225710e-01 6.422514e-02 8.133814e-01 7.118604e-02
##          261          262          263          264          265
## 5.621729e-02 7.118158e-02 8.480617e-01 5.920634e-01 3.402335e-01
##          266          267          268          269          270
## 6.236737e-02 4.275321e-01 6.238458e-02 5.114019e-01 7.540844e-02
##          271          272          273          274          275
## 5.152357e-01 1.225710e-01 9.666459e-01 6.859305e-01 1.149555e-01
##          276          277          278          279          280
## 8.092798e-01 2.145800e-01 1.824960e-01 2.078494e-01 2.309180e-01
##          281          282          283          284          285
## 5.073397e-01 5.649090e-01 6.873495e-01 7.135484e-01 5.866020e-01
##          286          287          288          289          290
## 5.709939e-02 6.239951e-02 4.335131e-01 1.149552e-01 6.235969e-02
##          291          292          293          294          295
## 4.193055e-01 6.598374e-01 1.149552e-01 4.048402e-01 5.699639e-02
##          296          297          298          299          300
```

```
## 6.615055e-02 9.168021e-01 1.137100e-01 4.180835e-01 6.329588e-02
##          301          302          303          304          305
## 6.055824e-02 3.397927e-01 1.761760e-01 6.807975e-02 6.873495e-01
##          306          307          308          309          310
## 8.997027e-01 3.963037e-01 5.655250e-01 3.175986e-01 5.229289e-01
##          311          312          313          314          315
## 7.431877e-02 1.283951e-01 6.238333e-02 6.345425e-01 9.484049e-01
##          316          317          318          319          320
## 7.040041e-01 4.569165e-01 2.394068e-01 6.518738e-02 2.277596e-01
##          321          322          323          324          325
## 6.615690e-02 1.232045e-01 2.195662e-01 4.105431e-01 8.825524e-01
##          326          327          328          329          330
## 6.917307e-02 8.263738e-01 5.143944e-01 2.097457e-01 2.333432e-01
##          331          332          333          334          335
## 8.035832e-01 5.522710e-01 1.149555e-01 7.628039e-01 6.518523e-02
##          336          337          338          339          340
## 4.221260e-01 2.038185e-01 1.013151e-01 2.195646e-01 7.076915e-01
##          341          342          343          344          345
## 2.422835e-01 6.056772e-02 5.936885e-02 9.271424e-01 5.521493e-01
##          346          347          348          349          350
## 5.350318e-01 2.195662e-01 7.060800e-01 2.249316e-01 7.818574e-01
##          351          352          353          354          355
## 9.581211e-01 2.226421e-01 2.325780e-01 1.578495e-01 5.934919e-01
##          356          357          358          359          360
## 3.478003e-01 9.047695e-01 6.236819e-02 1.225710e-01 5.809211e-01
##          361          362          363          364          365
## 4.993278e-01 9.209177e-01 8.441865e-01 6.514548e-02 9.692788e-01
##          366          367          368          369          370
## 5.483657e-01 1.143308e-01 5.046732e-01 9.582561e-01 3.435589e-01
##          371          372          373          374          375
## 2.335717e-01 9.466842e-01 3.564367e-01 1.748843e-01 9.092239e-01
##          376          377          378          379          380
## 9.274153e-01 5.112717e-01 2.335717e-01 3.256749e-01 5.446940e-01
##          381          382          383          384          385
## 1.225710e-01 1.294893e-01 5.599391e-01 6.222610e-01 2.089901e-01
##          386          387          388          389          390
## 7.940014e-01 6.812768e-02 1.473002e-01 1.386313e-01 5.429679e-01
##          391          392          393          394          395
## 4.350300e-01 9.158712e-01 5.157108e-01 1.684752e-01 6.258468e-02
##          396          397          398          399          400
## 9.473101e-01 1.331396e-01 9.556175e-01 7.015276e-02 1.208904e-01
##          401          402          403          404          405
## 9.000673e-01 1.878859e-01 9.561542e-01 4.681751e-01 5.370820e-01
##          406          407          408          409          410
## 3.761607e-01 2.281357e-01 4.705534e-01 6.873548e-01 5.841129e-01
##          411          412          413          414          415
## 6.873495e-01 9.646553e-01 4.878645e-01 6.235969e-02 9.602979e-01
##          416          417          418
## 5.501822e-02 6.235969e-02 7.050212e-01
```

Predictions of survival on the test set with the logistic model

```r
y_hat_logit_t <- ifelse(p_hat_logit_t > 0.5, "1", "0")
y_hat_logit_t
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18 
## "0" "1" "0" "0" "1" "0" "1" "0" "1" "0" "0" "0" "1" "0" "1" "1" "0" "0" 
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 
## "0" "1" "0" "1" "1" "1" "1" "0" "1" "0" "0" "0" "0" "0" "1" "1" "1" "0" 
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 
## "1" "1" "0" "0" "0" "0" "0" "1" "1" "0" "1" "0" "1" "1" "0" "0" "1" "1" 
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 
## "0" "1" "0" "0" "0" "1" "0" "0" "0" "1" "1" "1" "1" "0" "1" "1" "1" "0" 
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90 
## "0" "1" "1" "1" "0" "1" "0" "1" "1" "0" "0" "0" "0" "0" "1" "1" "1" "1" 
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 
## "1" "0" "1" "0" "1" "0" "1" "0" "1" "0" "1" "0" "0" "0" "1" "0" "0" "0" 
## 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 
## "0" "0" "0" "1" "1" "1" "1" "0" "0" "1" "1" "1" "1" "0" "1" "0" "0" "1" 
## 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 
## "0" "1" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "1" "0" "1" "1" "0" "0" 
## 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 
## "0" "0" "0" "0" "0" "0" "1" "0" "0" "1" "1" "0" "1" "1" "0" "1" "1" "1" 
## 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 
## "1" "0" "0" "1" "1" "0" "1" "1" "0" "0" "0" "0" "0" "1" "1" "0" "1" "1" 
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 
## "0" "1" "1" "0" "1" "0" "1" "0" "0" "0" "0" "0" "1" "0" "1" "0" "1" "1" 
## 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 
## "0" "1" "1" "1" "0" "1" "0" "1" "1" "0" "1" "0" "0" "0" "0" "1" "0" "0" 
## 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 
## "1" "0" "1" "0" "1" "0" "1" "0" "1" "1" "0" "1" "0" "0" "0" "1" "0" "0" 
## 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 
## "1" "0" "0" "0" "1" "1" "1" "1" "1" "0" "1" "0" "1" "0" "1" "1" "1" "0" 
## 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 
## "1" "0" "0" "0" "0" "0" "1" "0" "0" "0" "1" "1" "0" "0" "0" "0" "1" "0" 
## 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 
## "1" "0" "1" "1" "0" "1" "0" "0" "0" "0" "1" "1" "1" "1" "1" "0" "0" "0" 
## 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 
## "0" "0" "0" "1" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "1" "1" 
## 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 
## "0" "1" "0" "1" "0" "0" "0" "1" "1" "1" "0" "0" "0" "0" "0" "0" "0" "0" 
## 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 
## "1" "0" "1" "1" "0" "0" "1" "1" "0" "1" "0" "0" "0" "0" "0" "1" "0" "0" 
## 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 
## "0" "1" "1" "1" "0" "1" "0" "1" "1" "0" "0" "0" "1" "0" "1" "0" "0" "1" 
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 
## "0" "1" "1" "0" "1" "1" "0" "1" "1" "0" "0" "1" "0" "0" "1" "1" "1" "0" 
## 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 
## "0" "1" "0" "0" "1" "1" "0" "1" "0" "0" "0" "1" "0" "1" "1" "0" "0" "1" 
## 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 
## "0" "1" "0" "0" "1" "0" "1" "0" "1" "0" "0" "0" "1" "1" "1" "1" "0" "0" 
## 415 416 417 418 
## "1" "0" "0" "1" 
```

Adding to the data frame the accuracy achieved with the logistic regression model

```r
df7 <-  bind_rows(df6, data_frame(Model="Logistic Regression", Accuracy=acc_logit))
df7 %>% knitr::kable() %>% kable_styling("striped" , full_width = T)
```

Model	Accuracy
Naive Bayes	0.6636771
Naive Best Cutoff	0.6995516
F1 Balancing	0.6748879
QDA	0.6816143
LDA	0.6771300
Linear Regression	0.6793722
Logistic Regression	0.7869955

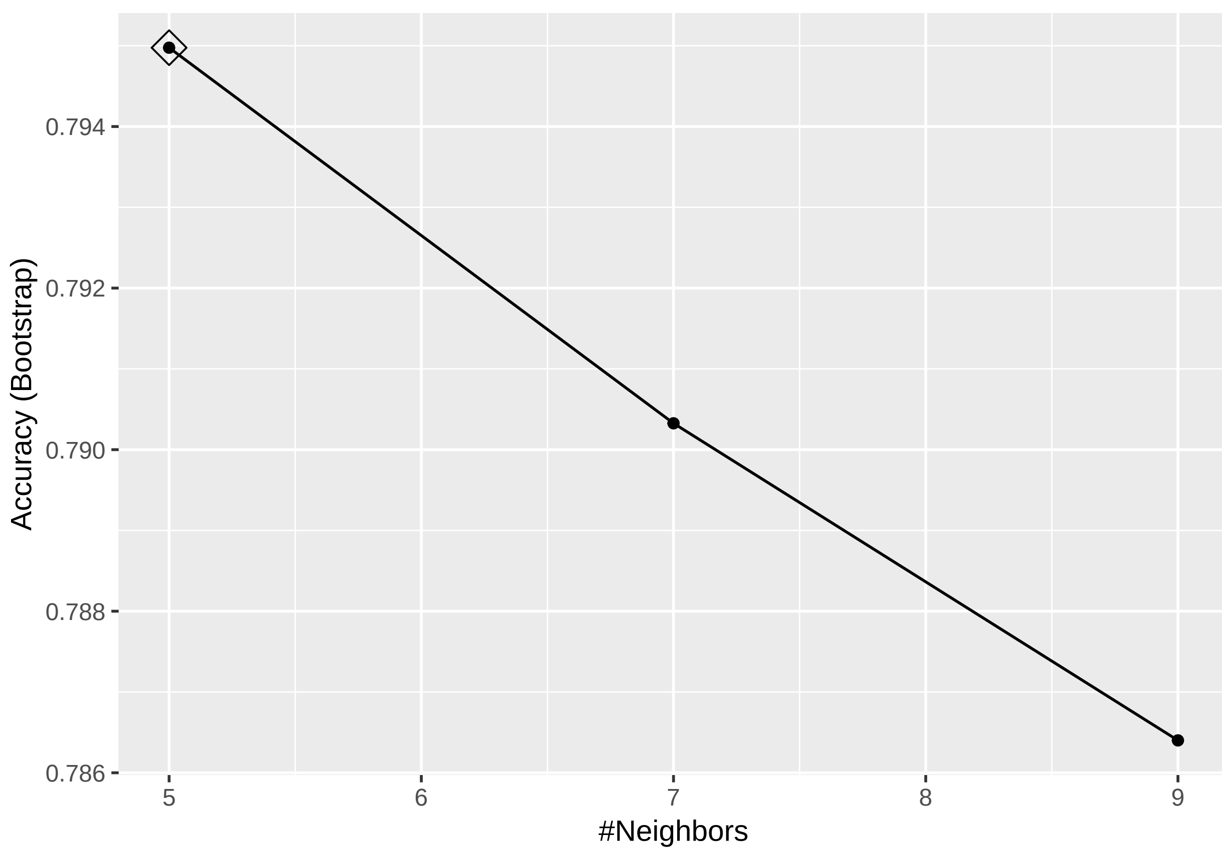
3.4.8 K Nearest Neighbors

Training the model with the default values of k from 1 to 9

```r
train_knn <- train(Survived_f~Class_f + Embarked_f + Sex_f + Title_f,
                   method = "knn",
                   data = train)
```

Plot of nearest neighbors versus accuracy before tuning of k

```r
ggplot(train_knn, highlight = TRUE)
```



KNN predictions of survival on the validation set

```r
y_hat_knn <- predict(train_knn, validation, type="raw")
```

Best k that leads to the maximum accuracy before tuning

```r
train_knn$bestTune
```

```
##   k
## 1 5
```

Best performing model i.e. the predictions yet on the training set before tuning

```r
train_knn$finalModel
```

```
## 5-nearest neighbor model
## Training set outcome distribution:
##
##   0   1
## 267 178
```

Accuracy achieved on the validation set without KNN tuning of parameters

```r
confusionMatrix(predict(train_knn, validation), validation$Survived_f)$overall["Accuracy"]
```

```
##  Accuracy
## 0.8251121
```

KNN predictions of survival on the test set before tuning

```r
y_hat_knn_t <- predict(train_knn, titanic_test, type="raw")
y_hat_knn_t
```

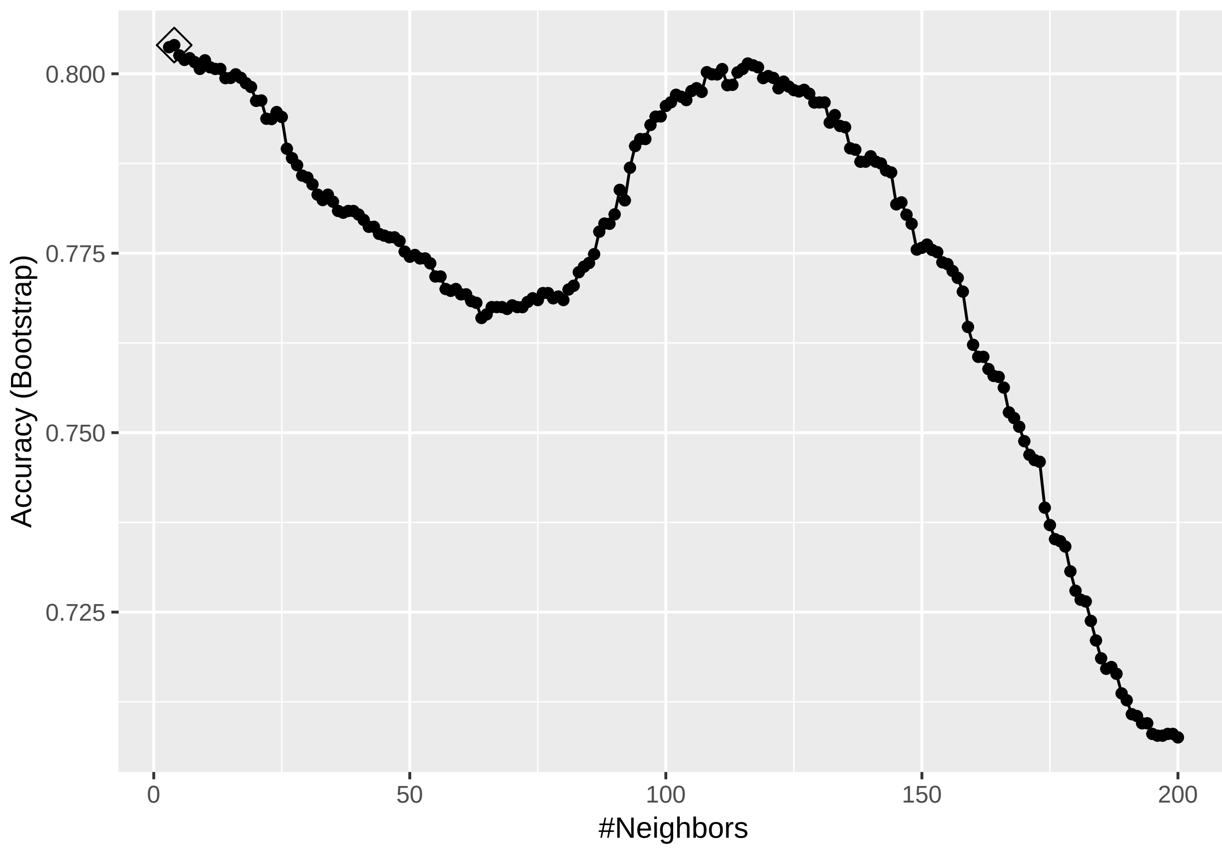
```
##   [1] 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0
##  [36] 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1
##  [71] 1 0 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 1 0 0 0 1 0 0 0 1
## [106] 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0
## [141] 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0
## [176] 1 1 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0 1 0 1 0
## [211] 0 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0
## [246] 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0
## [281] 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 1
## [316] 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1
## [351] 1 0 0 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 1 1 0
## [386] 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1
## Levels: 0 1
```

Tuning of parameter k by selecting 200 values

```r
train_knn_tune <- train(Survived_f~Class_f + Embarked_f + Sex_f + Title_f,
                   method = "knn", tuneGrid = data.frame(k = seq(3, 200)),
                   data = train)
```

Plot of nearest neighbors versus accuracy after tuning

```r
ggplot(train_knn_tune, highlight = TRUE)
```



KNN tuned predictions of survival on the validation set

```r
y_hat_knn_tune <- predict(train_knn_tune, validation, type="raw")
```

Best k that leads to the maximum accuracy after tuning

```r
train_knn_tune$bestTune
```

```
##   k
## 2 4
```

Best performing model i.e. the predictions yet on the training set after tuning

```r
train_knn_tune$finalModel
```

```
## 4-nearest neighbor model
## Training set outcome distribution:
##
##   0   1
## 267 178
```

Accuracy achieved on the validation set after tuning of the KNN model parameters

```
acc_knn <- confusionMatrix(predict(train_knn_tune, validation),
                           validation$Survived_f)$overall["Accuracy"]
acc_knn
```

```
##  Accuracy
## 0.8206278
```

KNN predictions of survival on the test set

```
y_hat_knn_tune_t <- predict(train_knn_tune, titanic_test, type="raw")
y_hat_knn_tune_t
```

```
##   [1] 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0
##  [36] 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1
##  [71] 1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 1 0 0 0 1 0 0 0 1
## [106] 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0
## [141] 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0
## [176] 1 1 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0 1 0 1 0
## [211] 0 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0
## [246] 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0
## [281] 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 1
## [316] 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1
## [351] 1 0 0 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 1 1 0
## [386] 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 0
## Levels: 0 1
```

We conclude that tuning k does not improve the accuracy much. This is because best accuracies are achieved for low values of k

Adding to the data frame the accuracy achieved with the KNN model

```
df8 <-  bind_rows(df7, data_frame(Model="KNN", Accuracy=acc_knn))
df8 %>% knitr::kable() %>% kable_styling("striped" , full_width = T)
```

Model	Accuracy
Naive Bayes	0.6636771
Naive Best Cutoff	0.6995516
F1 Balancing	0.6748879
QDA	0.6816143
LDA	0.6771300
Linear Regression	0.6793722
Logistic Regression	0.7869955
KNN	0.8206278

3.4.9. Classification Tree

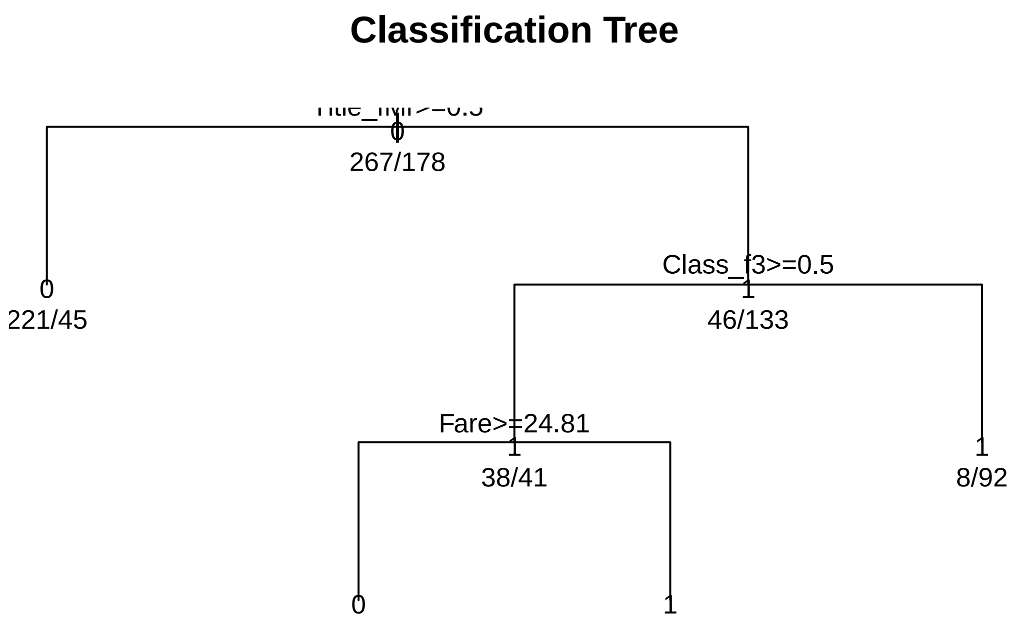
Training the classification tree model

```
train_rpart <- train(Survived_f~Age + Class_f + Embarked_f + Fare + Sex_f + Title_f,
                     method="rpart",
                     tuneGrid=data.frame(cp = seq(0.0, 0.1, len=25)),
                     data=train, na.action=na.omit)
train_rpart
```

```
## CART
##
## 445 samples
##   6 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 445, 445, 445, 445, 445, 445, ...
## Resampling results across tuning parameters:
##
##   cp           Accuracy   Kappa
##   0.000000000  0.7698443  0.5123979
##   0.004166667  0.7749989  0.5215811
##   0.008333333  0.7794445  0.5288804
##   0.012500000  0.7842082  0.5374335
##   0.016666667  0.7905434  0.5490237
##   0.020833333  0.7939880  0.5556299
##   0.025000000  0.7955393  0.5588882
##   0.029166667  0.7973453  0.5641492
##   0.033333333  0.7979077  0.5666931
##   0.037500000  0.7963492  0.5636909
##   0.041666667  0.7948184  0.5628028
##   0.045833333  0.7942706  0.5619044
##   0.050000000  0.7930296  0.5612249
##   0.054166667  0.7918042  0.5603250
##   0.058333333  0.7880866  0.5527185
##   0.062500000  0.7878318  0.5526908
##   0.066666667  0.7885962  0.5548009
##   0.070833333  0.7885962  0.5548009
##   0.075000000  0.7885962  0.5548009
##   0.079166667  0.7908322  0.5599660
##   0.083333333  0.7908322  0.5599660
##   0.087500000  0.7908322  0.5599660
##   0.091666667  0.7908322  0.5599660
##   0.095833333  0.7908322  0.5599660
##   0.100000000  0.7908322  0.5599660
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03333333.
```

Plot of the classification tree model

```r
plot(train_rpart$finalModel, uniform=TRUE, main="Classification Tree")
text(train_rpart$finalModel, use.n=TRUE, all=TRUE, cex=.8)
```



Accuracy achieved on the validation set with the classification tree model

```r
acc_class <- confusionMatrix(predict(train_rpart, validation),
                             validation$Survived_f)$overall["Accuracy"]
acc_class
```

```
##  Accuracy
## 0.8183857
```

Classification tree predictions on the validation set

```r
y_hat_ct <- predict(train_rpart, validation)
y_hat_ct
```

```
##   [1] 0 1 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1
##  [36] 1 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 1 1
##  [71] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 1 0 0 0 0 1 1 0
## [106] 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 1 1 0 1 1 0 0 1 1 0 1 0 1
## [141] 1 0 0 1 0 0 0 0 1 0 1 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1 0 0 1 0 0 1 0 1
## [176] 0 1 0 0 0 0 1 1 1 0 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 0 1 0 1 1 1 1 0 1 0
## [211] 0 1 1 1 1 0 1 0 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
```

```
## [246] 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 1 0 0 1 1 1 0 0 1 0 0 0 0
## [281] 0 0 1 0 0 0 0 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0
## [316] 1 1 0 0 1 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0
## [351] 0 1 0 0 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1
## [386] 0 1 0 0 0 1 0 1 1 0 1 0 0 0 1 1 1 1 1 0 0 1 0 0 1 0 0 0 1 0 1 0 0 0 0
## [421] 0 0 0 1 1 0 1 1 1 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 0
## Levels: 0 1
```

Classification tree predictions on the test set

```
y_hat_ct_t <- predict(train_rpart, titanic_test)
y_hat_ct_t
```

```
##   [1] 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 1 0 0 1 1 0 1 1 0 1 0 1 0 0 0 0 0 1 1 0
##  [36] 0 1 1 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1
##  [71] 1 0 1 0 1 0 0 1 0 1 1 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 1 0 1 0 1 0 0 0 1
## [106] 0 0 0 0 0 0 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0
## [141] 0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 0 0 1 1 0 0 0 0 0
## [176] 1 1 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 1 1 0 1 1 1 0 1 0 0 1 0 1 0
## [211] 0 0 0 1 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 1
## [246] 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0
## [281] 1 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 1 1
## [316] 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 1 1 0 1 0 1
## [351] 1 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 0 1 1 1 0 0 0 0 0 1 1 0
## [386] 1 0 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 1 1 1 1 0 1 0 0 1
## Levels: 0 1
```

Adding to the data frame the accuracy achieved with the classification tree model

```
df9 <-  bind_rows(df8, data_frame(Model="Classification Tree", Accuracy=acc_class))
df9 %>% knitr::kable() %>% kable_styling("striped" , full_width = T)
```

Model	Accuracy
Naive Bayes	0.6636771
Naive Best Cutoff	0.6995516
F1 Balancing	0.6748879
QDA	0.6816143
LDA	0.6771300
Linear Regression	0.6793722
Logistic Regression	0.7869955
KNN	0.8206278
Classification Tree	0.8183857

3.4.10. Random Forest

We build classification trees using the training set. We refer to the fitted models as T1,T2,. . . ,TB

For every observation in the test set, form a prediction ˆyj using tree Tj

predict ˆy with majority vote (most frequent class among $^{y1,\cdots}$yT)

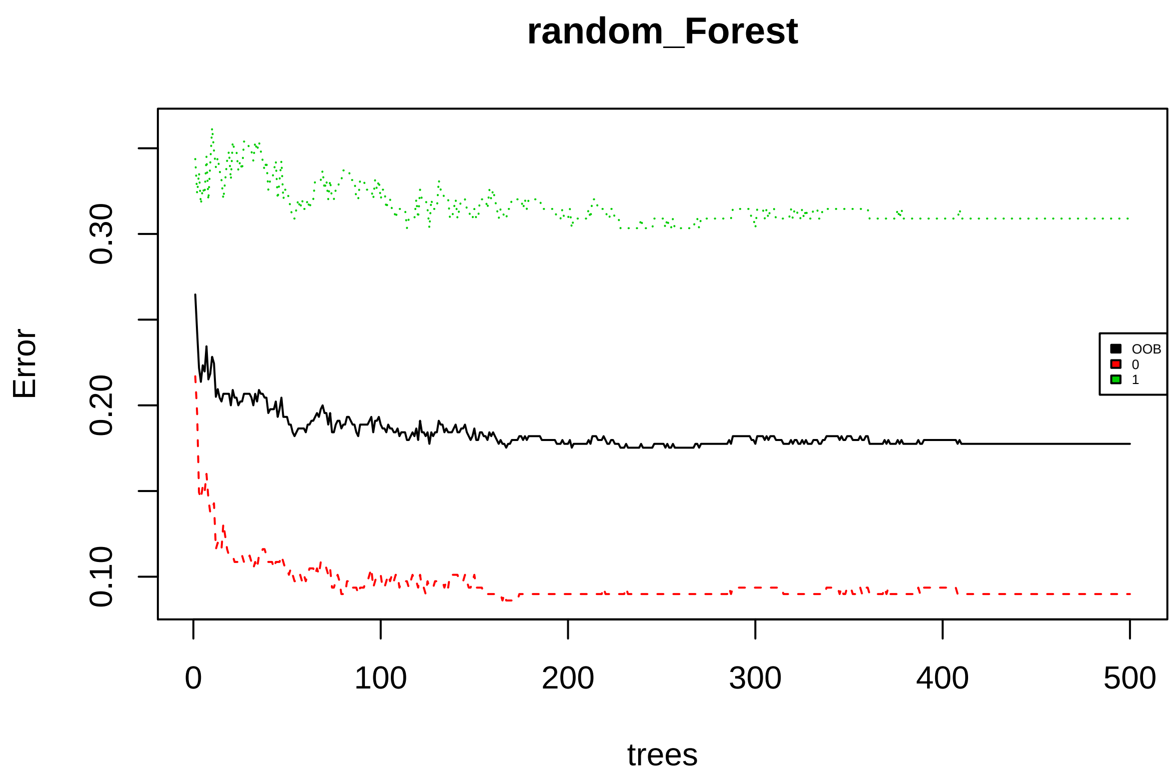
Random Forest model

```
random_Forest <- randomForest(Survived_f~Age + Class_f + Embarked_f + Fare + Sex_f +
                                Title_f,
                              data = train)
random_Forest
```

```
##
## Call:
##  randomForest(formula = Survived_f ~ Age + Class_f + Embarked_f +      Fare + Sex_f + Title_f, data =
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 17.75%
## Confusion matrix:
##     0   1 class.error
## 0 243  24  0.08988764
## 1  55 123  0.30898876
```

Ploting the object random_Forest

```
plot(random_Forest)
legend("right", colnames(random_Forest$err.rate), col=1:4, cex=0.4, fill=1:4)
```



The OOB error stabilizes after some 170 trees

Variables importance

```r
varImp(random_Forest)
```

```
##             Overall
## Age        26.264391
## Class_f    19.658044
## Embarked_f  6.589648
## Fare       32.585211
## Sex_f      28.584136
## Title_f    39.559818
```

Title_f and Fare are the two most important features in a sense that they reduce mostly impurity whenever used at tree nodes across all trees

Random forest accuracy achieved on the validation set

```r
acc_rf<- confusionMatrix(predict(random_Forest, validation),
                         validation$Survived_f)$overall["Accuracy"]
acc_rf
```

```
##  Accuracy
## 0.8430493
```

Random forest predictions on the validation set

```r
y_hat_rf <- predict(random_Forest, validation, type="class")
y_hat_rf
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##   0   1   1   0   1   1   0   0   0   1   0   0   0   0   1   0   0   1
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##   0   1   0   0   0   0   1   0   0   0   0   0   0   0   0   0   1   1
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##   1   0   1   1   0   1   0   0   0   0   0   0   0   0   0   0   1   0
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##   1   0   0   0   0   0   0   0   0   1   0   1   0   0   0   1   0   0
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
##   0   0   0   1   1   1   0   1   0   0   0   0   1   1   0   0   1   1
## 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
##   1   0   0   0   0   0   0   0   0   0   1   0   0   0   0   1   0   0
## 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
##   1   1   1   0   1   1   0   0   1   1   0   1   1   1   1   0   0   1
## 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
##   0   0   0   0   1   0   0   0   1   1   1   0   0   1   1   1   0   0
## 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   1   1   0   1   1   0   0   1   0   0   1   0   1   0   1   0   0   0
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
##   0   1   1   1   0   0   0   1   1   1   0   0   1   0   0   1   1   0
## 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
##   0   0   0   1   0   1   1   1   0   0   0   0   0   1   1   1   0   0
## 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
```

```
##   1   0   1   1   1   1   0   1   0   0   0   1   0   0   0   0   0   1
## 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
##   0   0   0   0   0   0   0   1   0   0   0   1   0   0   0   0   0   1
## 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
##   0   0   0   1   0   0   0   1   1   0   0   0   1   0   1   1   0   0
## 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
##   1   0   1   1   0   1   0   0   0   0   0   0   1   0   0   0   0   1
## 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
##   0   1   1   1   1   1   0   0   0   0   0   0   0   0   0   0   1   1
## 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
##   0   0   1   1   0   1   0   0   0   0   1   0   0   0   0   0   1   0
## 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
##   0   0   0   0   0   1   1   1   0   0   0   0   0   0   1   0   0   0
## 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
##   1   0   0   1   0   0   0   0   0   1   0   0   0   1   1   1   0   0
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378
##   1   1   0   0   0   0   1   0   0   0   0   0   0   1   0   0   0   1
## 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396
##   0   1   0   0   0   0   1   0   1   0   0   0   1   0   1   1   1   1
## 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414
##   0   0   0   1   1   1   1   1   0   0   1   0   0   0   0   0   0   1
## 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432
##   0   1   0   0   0   0   0   0   0   0   1   0   1   1   0   1   0   1
## 433 434 435 436 437 438 439 440 441 442 443 444 445 446
##   0   1   1   1   0   1   0   0   1   0   0   0   1   0
## Levels: 0 1
```

Random forest predictions on the test set

```
y_hat_rf_t <- predict(random_Forest, titanic_test, type="class")
y_hat_rf_t
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##   0   1   0   0   1   0   1   0   1   0   0   0   1   0   1   1   0   0
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##   1   1   0   1   1   0   1   0   1   0   1   0   0   0   1   1   0   0
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##   0   0   0   0   0   1   0   1   1   0   0   0   1   1   0   0   1   1
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##   0   0   0   0   0   1   0   0   0   1   1   1   1   0   0   1   1   0
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##   0   0   1   0   0   1   0   1   1   0   0   0   0   0   1   0   1   1
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
##   1   0   1   0   0   0   1   0   1   0   1   0   0   0   1   0   0   0
## 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
##   0   0   0   1   1   1   1   0   0   1   1   1   1   0   1   0   0   0
## 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
##   0   1   0   0   0   0   1   0   0   0   0   0   0   0   0   1   0   0
## 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
##   1   0   0   0   1   0   1   0   0   1   0   0   1   1   1   1   1   1
## 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   1   0   0   1   0   0   1   0   0   0   0   0   0   1   1   0   1   1
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
```

```
##   0   0   1   0   1   0   1   0   0   0   0   0   1   0   1   0   1   0
## 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
##   0   1   1   1   0   1   0   0   1   0   1   0   0   0   0   1   1   0
## 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
##   1   0   1   0   1   0   1   0   1   0   0   1   0   0   0   1   0   0
## 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
##   0   0   0   0   1   1   1   1   0   0   0   0   1   0   1   1   1   0
## 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
##   0   0   0   0   0   0   1   0   0   0   1   1   0   0   0   0   0   0
## 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
##   0   0   1   1   0   1   0   0   0   0   0   1   1   1   1   0   0   0
## 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
##   0   0   0   1   0   1   0   0   1   0   0   0   0   0   0   0   1   1
## 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
##   0   1   0   0   0   0   0   1   1   1   0   0   0   0   0   0   0   1
## 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
##   1   0   1   0   0   0   1   1   0   1   0   0   0   0   0   1   0   0
## 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
##   0   1   1   1   0   1   0   1   1   0   0   0   1   0   1   0   0   1
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378
##   0   1   1   0   1   1   0   0   1   0   0   1   0   0   1   1   0   0
## 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396
##   0   0   0   0   1   1   0   1   0   0   0   0   0   1   1   0   0   1
## 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414
##   0   1   0   0   1   0   1   0   1   0   0   0   1   1   1   1   1   0
## 415 416 417 418
##   1   0   0   1
## Levels: 0 1
```

Adding to the data frame the accuracy achieved with the random forest model

```
df10 <-  bind_rows(df9, data_frame(Model="Random Forest", Accuracy=acc_rf))
df10 %>% knitr::kable() %>% kable_styling("striped" , full_width = T)
```

Model	Accuracy
Naive Bayes	0.6636771
Naive Best Cutoff	0.6995516
F1 Balancing	0.6748879
QDA	0.6816143
LDA	0.6771300
Linear Regression	0.6793722
Logistic Regression	0.7869955
KNN	0.8206278
Classification Tree	0.8183857
Random Forest	0.8430493

4. Conclusion/Brief Summary

Since Random Forest is the best performing model, we replace the survival variable in the original titanic_test set with the predicted ones using the following code:

Adding the survival predictions to the original test set

```r
titanic_test_S <- titanic_test %>% mutate (Survived_f = y_hat_rf_t)
head(titanic_test_S)
```

```
## # A tibble: 6 x 19
##     Age Cabin Embarked  Fare Name
##   <dbl> <chr> <chr>    <dbl> <chr>
## 1  34.5 <NA>  Q         7.83 Kelly, Mr. James
## 2  47   <NA>  S         7    Wilkes, Mrs. James (Ellen Needs)
## 3  62   <NA>  Q         9.69 Myles, Mr. Thomas Francis
## 4  27   <NA>  S         8.66 Wirz, Mr. Albert
## 5  22   <NA>  S        12.3  Hirvonen, Mrs. Alexander (Helga E Lindqvist)
## 6  14   <NA>  S         9.22 Svensson, Mr. Johan Cervin
##   Parch PassengerId Pclass Sex    SibSp Survived Ticket  Title Family_Size
##   <dbl>       <dbl>  <dbl> <chr>  <dbl> <lgl>    <chr>   <chr>       <dbl>
## 1     0         892      3 male       0 NA       330911  Mr              0
## 2     0         893      3 female     1 NA       363272  Mrs             1
## 3     0         894      2 male       0 NA       240276  Mr              0
## 4     0         895      3 male       0 NA       315154  Mr              0
## 5     1         896      3 female     1 NA       3101298 Mrs             2
## 6     0         897      3 male       0 NA       7538    Mr              0
##   Class_f Embarked_f Sex_f  Title_f Survived_f
##   <fct>   <fct>      <fct>  <fct>   <fct>
## 1 3       Q          male   Mr      0
## 2 3       S          female Mrs     1
## 3 2       Q          male   Mr      0
## 4 3       S          male   Mr      0
## 5 3       S          female Mrs     1
## 6 3       S          male   Mr      0
```

Clearly the best performing models are those providing high accuracies. In this respect, we note the Logistic model, the KNN, the Classification Tree and Random Forest. If we were to predict the risk of survival, the Random Forest would answer best yet short of roughly 16% accuracy. The variables used in the Random Forest model are good features to estimating patterns of survival in the event of any future ship sinking.