Competition Description

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

In this challenge, we ask you to complete the analysis of what sorts of people were likely to survive. In particular, we ask you to apply the tools of machine learning to predict which passengers survived the tragedy.

1.1 Load the required library

Load all the packages required for the analysis

```
library(dplyr) # Data Manipulation
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(naniar)
```

```
## Warning: package 'naniar' was built under R version 3.5.1
```

```
library(Amelia) # Missing Data: Missings Map
```

```
## Warning: package 'Amelia' was built under R version 3.5.1
```

```
## Loading required package: Rcpp
```

```
## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.7.5, built: 2018-05-07)
## ## Copyright (C) 2005-2018 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

```
library(ggplot2) # Visualization
library(scales) # Visualization
library(caTools) # Prediction: Splitting Data
```

```
## Warning: package 'caTools' was built under R version 3.5.1
```

```
library(car) # Prediction: Checking Multicollinearity
```

```
## Warning: package 'car' was built under R version 3.5.1
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 3.5.1
```

```
##
## Attaching package: 'car'
```

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

```
library(ROCR) # Prediction: ROC Curve
```

```
## Warning: package 'ROCR' was built under R version 3.5.1
```

```
## Loading required package: gplots
```

```
## Warning: package 'gplots' was built under R version 3.5.1
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(e1071) # Prediction: SVM, Naive Bayes, Parameter Tuning
library(rpart) # Prediction: Decision Tree
library(rpart.plot) # Prediction: Decision Tree
library(randomForest) # Prediction: Random Forest
```

```
## Warning: package 'randomForest' was built under R version 3.5.1
```

```
## randomForest 4.6-14
```

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

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

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

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

```
library(caret) # Prediction: k-Fold Cross Validation
```

```
## Loading required package: lattice
```

```
library(pROC) #Plot the ROC for logistic regression
```

```
## Warning: package 'pROC' was built under R version 3.5.1
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

1.2 Read the train and test data from titanic data

```
titanic_trainData = read.csv("E:/MachineLearning_Model/kaggle/classification_kaggle/titanic/train.csv")
titanic_testData =read.csv("E:/MachineLearning_Model/kaggle/classification_kaggle/titanic/test.csv")
```

Combine the train and test of titanic data

```
titanic_mainData= bind_rows(titanic_trainData, titanic_testData)
```

```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

Validate the data type of all features of titanic data set.

verify the data structure

```
str(titanic_mainData)
```

```
## 'data.frame':    1309 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)" "Heikkinen, Miss. La
ina" "Futrelle, Mrs. Jacques Heath (Lily May Peel)" ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

Investigate the titanic features statistical characteristics

summarize the data

```
summary(titanic_mainData)
```

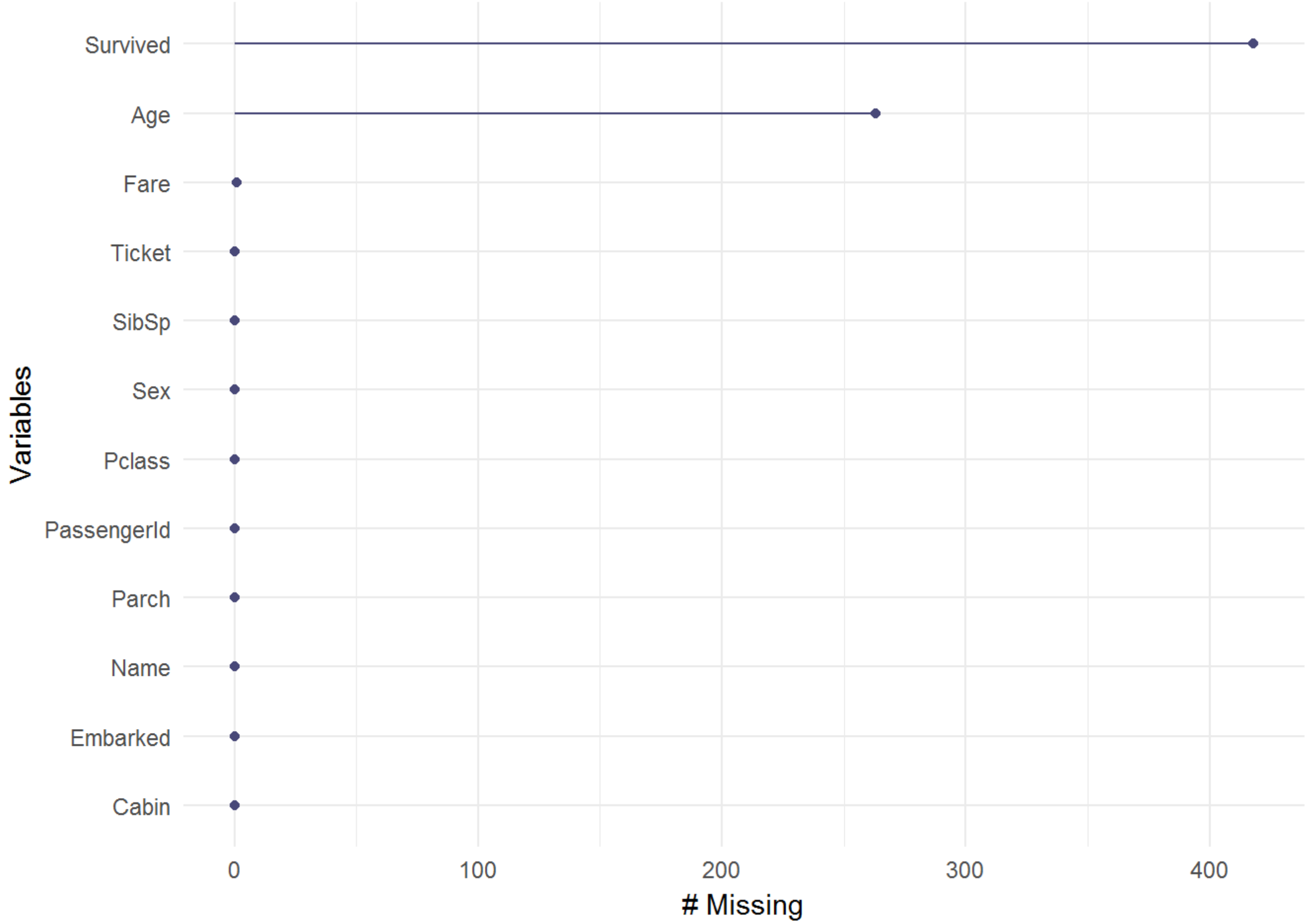
```
##   PassengerId      Survived          Pclass          Name
##  Min.   :   1   Min.   :0.0000   Min.   :1.000   Length:1309
##  1st Qu.: 328   1st Qu.:0.0000   1st Qu.:2.000   Class :character
##  Median : 655   Median :0.0000   Median :3.000   Mode  :character
##  Mean   : 655   Mean   :0.3838   Mean   :2.295
##  3rd Qu.: 982   3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :1309   Max.   :1.0000   Max.   :3.000
##                 NA's   :418
##      Sex             Age            SibSp            Parch
##  female:466   Min.   : 0.17   Min.   :0.0000   Min.   :0.000
##  male  :843   1st Qu.:21.00   1st Qu.:0.0000   1st Qu.:0.000
##               Median :28.00   Median :0.0000   Median :0.000
##               Mean   :29.88   Mean   :0.4989   Mean   :0.385
##               3rd Qu.:39.00   3rd Qu.:1.0000   3rd Qu.:0.000
##               Max.   :80.00   Max.   :8.0000   Max.   :9.000
##               NA's   :263
##     Ticket               Fare             Cabin
##  Length:1309        Min.   :  0.000   Length:1309
##  Class :character   1st Qu.:  7.896   Class :character
##  Mode  :character   Median : 14.454   Mode  :character
##                     Mean   : 33.295
##                     3rd Qu.: 31.275
##                     Max.   :512.329
##                     NA's   :1
##    Embarked
##  Length:1309
##  Class :character
##  Mode  :character
##
##
##
##
```

Above summarize data shows that out of all features of titanic data set features Survived, Age and Fare consist of missing values.

2. Handling missing data

Check the missing values in column (NA) not the blank values

```
gg_miss_var(titanic_mainData)
```



#Check missing values along with blanks

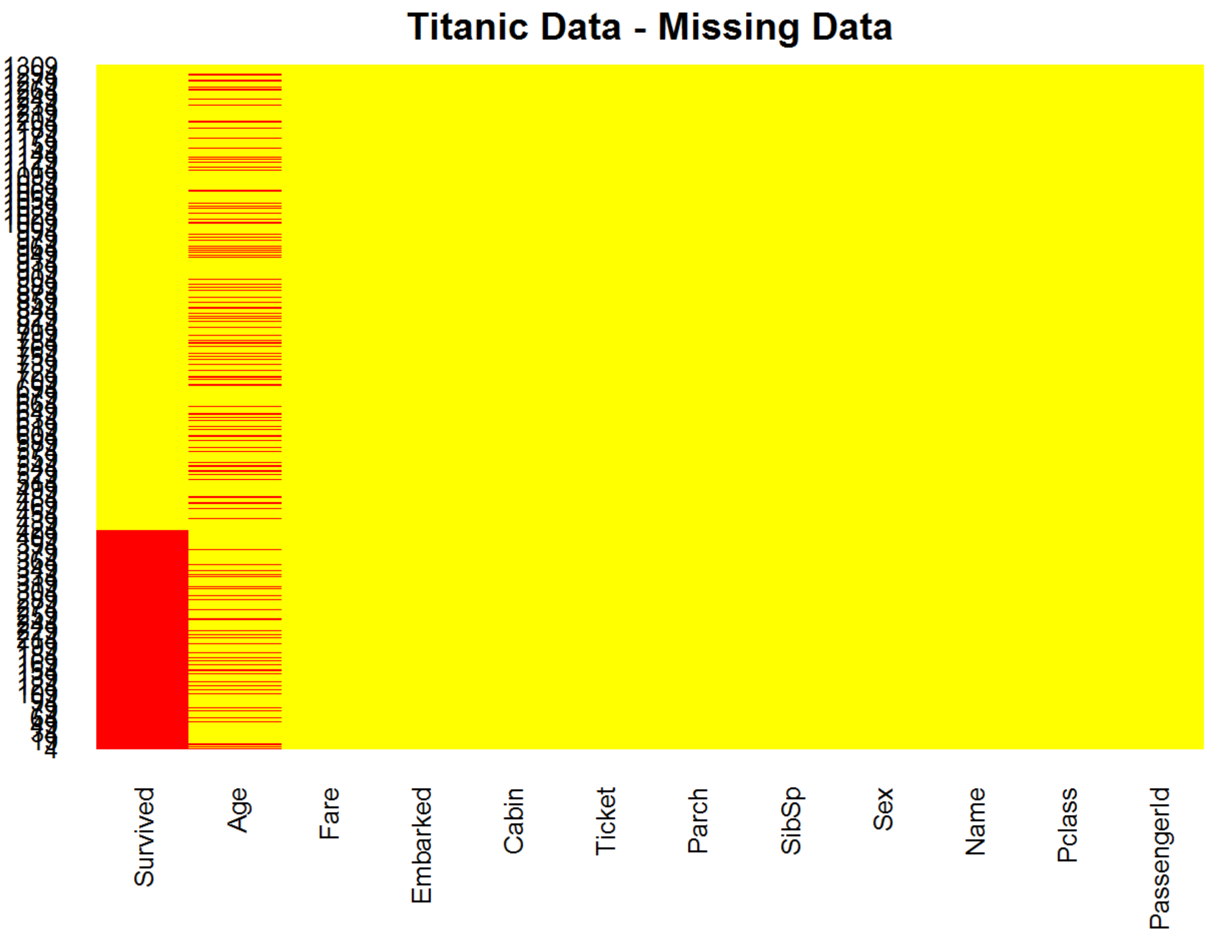
```
colSums(is.na(titanic_mainData) | titanic_mainData=='')
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0         418           0           0           0         263
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           1        1014           2
```

Note - Survived - 418, age-263 have missing of NAs, cabin- 1014, fare-1 and embarked-2 have blank or spaces

missmap allows us to explore how much missing data we have.

```
missmap(titanic_mainData, main = "Titanic Data - Missing Data", col = c("Red", "Yellow"), legend=FALSE)
```



2.1 Fix the missing data for Fare

Extract the row which contains the missing Fare

```
filter(titanic_mainData, is.na(Fare) | Fare == '')
```

```
##   PassengerId Survived Pclass               Name  Sex  Age SibSp Parch
## 1        1044       NA      3 Storey, Mr. Thomas male 60.5     0     0
##   Ticket Fare Cabin Embarked
## 1   3701   NA              S
```

Count the number of rows having missing or blank values in Fare column.

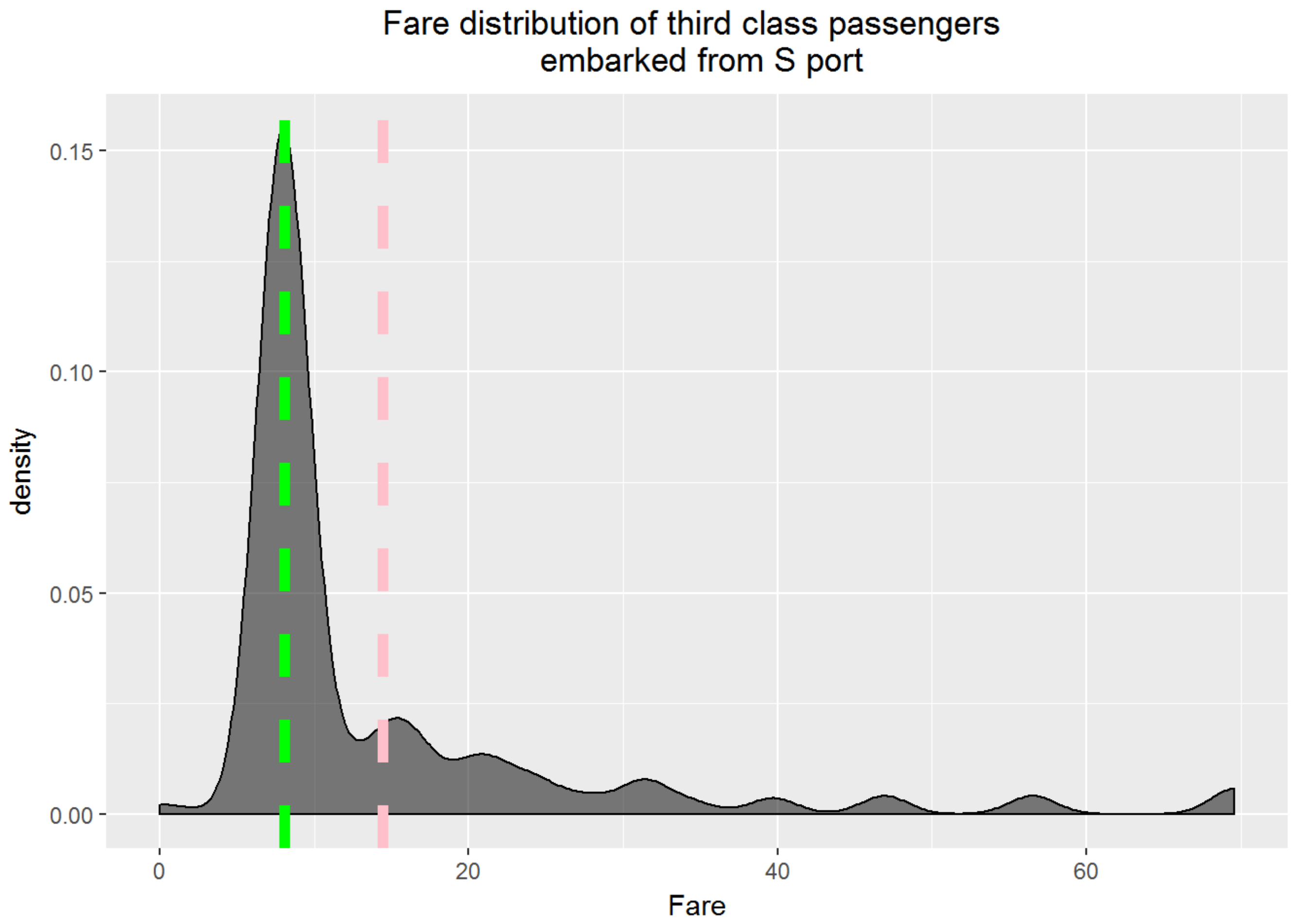
```
nrow(filter(titanic_mainData, is.na(Fare) | Fare == ''))
```

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

The passenger belong from class 3 , hence lets repalce the common value for fare from class 3 ticket fare. Also passenger started from S

```
ggplot(filter(titanic_mainData, Pclass == 3 & Embarked =="S"), aes(Fare))+
  geom_density(fill="black", alpha=0.5) +
  geom_vline(aes(xintercept=median(Fare, na.rm=T)), colour='green', linetype='dashed', size=2) +
  geom_vline(aes(xintercept=mean(Fare, na.rm=T)), colour='pink', linetype='dashed', size=2) +
  ggtitle("Fare distribution of third class passengers \n embarked from S port") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```



Since mean and median are very far and median is effective as comapred to mean, replace missing fare with [median] instead of mean.

```
titanic_mainData$Fare[is.na(titanic_mainData$Fare)== TRUE ] = median(filter(titanic_mainData, Pclass==3 & Embarked=="S")$Far
e, na.rm=TRUE)

#Verify the replaced values
colSums(is.na(titanic_mainData) | titanic_mainData=='')
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0         418           0           0           0         263
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           0        1014           2
```

Note Fare is replaced with value 8.05 and same columns doesnot have missing values

2.2 Fix the missing value for feature -Embarked.

List the columns having missing or blank values in titanic dataset.

```
filter(titanic_mainData, is.na(Embarked) | Embarked=='')
```

```
##   PassengerId Survived Pclass                                      Name
## 1          62        1      1                       Icard, Miss. Amelie
## 2         830        1      1 Stone, Mrs. George Nelson (Martha Evelyn)
##      Sex Age SibSp Parch Ticket Fare Cabin Embarked
## 1 female  38     0     0 113572   80   B28
## 2 female  62     0     0 113572   80   B28
```

The missing embarked female belong to cabin B28, Pclass 1 not sure why having same ticket, lets fix this.

First find the most frequent Embarked for Pclass passenger.

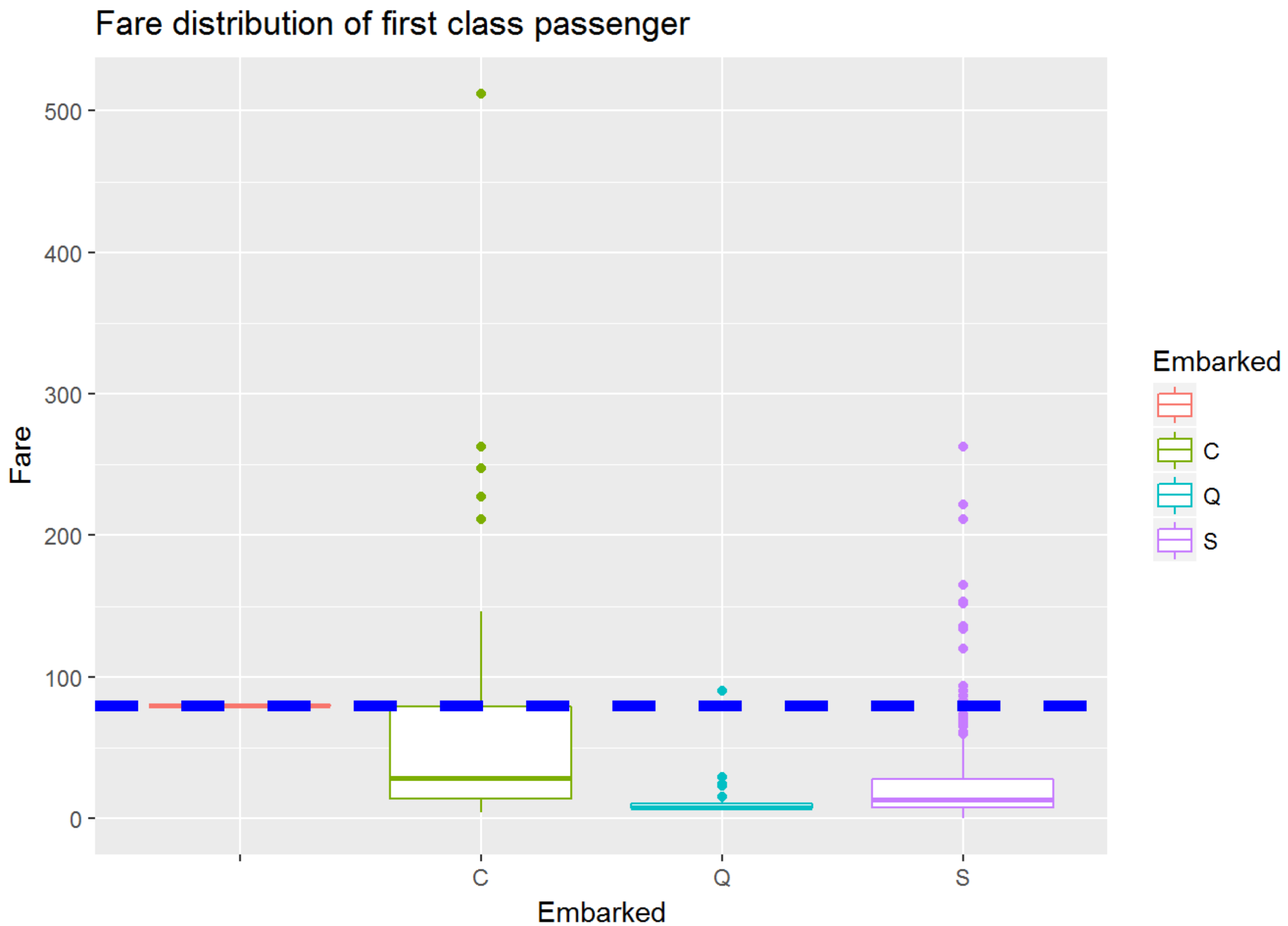
```
table(filter(titanic_mainData, Pclass==1)$Embarked)
```

```
##
##       C   Q   S
##   2 141   3 177
```

The most frequest Embark from first class passenger is “S” ie 177, we can replace value with the “S” for all missing Embarked.

Let us find the mean fair for each port to support above assumptions.

```
ggplot(filter(titanic_mainData, is.na(Embarked) == FALSE | Embarked != '' & Pclass == 1), aes(x=Embarked, y=Fare)) +
  geom_boxplot(aes(colour = Embarked)) +
  geom_hline(aes(yintercept=80), colour = "blue", linetype="dashed", size=2) +
  ggtitle("Fare distribution of first class passenger")
```



From above chart look like 80 fare belong to C port

Replace missing Embarked with port C and verify the replaced values.

```
titanic_mainData$Embarked[titanic_mainData$Embarked == ''] = "C"
colSums(is.na(titanic_mainData) | titanic_mainData=='')
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age 
##           0         418           0           0           0         263 
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked 
##           0           0           0           0        1014           0
```

Count the number of row having missing/Blank Embarked column post replacement.

```
filter(titanic_mainData, is.na(Embarked) | Embarked=='')
```

```
##  [1] PassengerId Survived    Pclass      Name        Sex        
##  [6] Age         SibSp       Parch       Ticket      Fare       
## [11] Cabin       Embarked   
## <0 rows> (or 0-length row.names)
```

There is no rows having blank Embarked columns, mean repalcment was done successfully.

2.3 Missing value fix for Age

Find the row having missing/blank Age

```
titanic_mainData %>% filter(is.na(Age) | Age=='') %>% group_by(Pclass)
```

```
## # A tibble: 263 x 12
## # Groups:   Pclass [3]
##    PassengerId Survived Pclass Name  Sex     Age SibSp Parch Ticket   Fare
##          <int>    <int>  <int> <chr> <fct> <dbl> <int> <int> <chr>   <dbl>
##  1           6        0      3 Mora~ male     NA     0     0 330877   8.46
##  2          18        1      2 Will~ male     NA     0     0 244373  13
##  3          20        1      3 Mass~ fema~    NA     0     0 2649     7.22
##  4          27        0      3 Emir~ male     NA     0     0 2631     7.22
##  5          29        1      3 "O'D~ fema~    NA     0     0 330959   7.88
##  6          30        0      3 Todo~ male     NA     0     0 349216   7.90
##  7          32        1      1 Spen~ fema~    NA     1     0 PC 17~ 147.
##  8          33        1      3 Glyn~ fema~    NA     0     0 335677   7.75
##  9          37        1      3 Mame~ male     NA     0     0 2677     7.23
## 10          43        0      3 Krae~ male     NA     0     0 349253   7.90
## # ... with 253 more rows, and 2 more variables: Cabin <chr>,
## #   Embarked <chr>
```

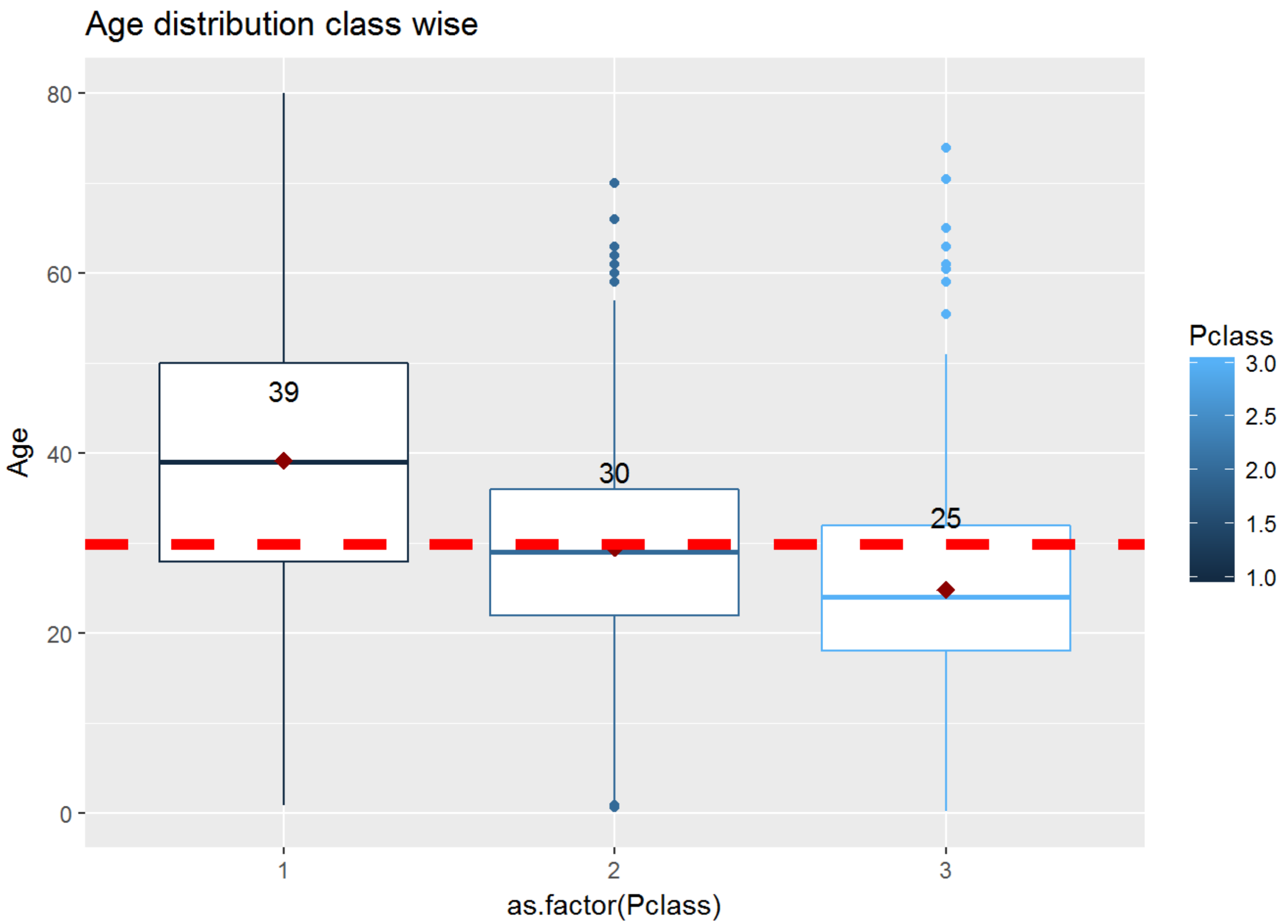
Class wise age Box plot with mean values of age in each class.

```
#This calcualtion is for mean calculation to print in box plot
means <- round(aggregate(Age ~  Pclass, titanic_mainData, mean))
ggplot(titanic_mainData, aes(x=as.factor(Pclass), y=Age)) +
  geom_boxplot(aes(colour=Pclass)) +
  stat_summary(fun.y=mean, colour="darkred", geom="point", shape=18, size=3,show_guide = FALSE) +
  geom_text(data = means,aes(label = Age, y = Age + 8)) + # This code print mean ,age + 8 is done for above printing
  geom_hline(aes(yintercept=mean(Age, na.rm=T)), colour = "red", linetype="dashed", size=2) +
  ggtitle("Age distribution class wise")
```

```
## Warning: `show_guide` has been deprecated. Please use `show.legend`
## instead.
```

```
## Warning: Removed 263 rows containing non-finite values (stat_boxplot).
```

```
## Warning: Removed 263 rows containing non-finite values (stat_summary).
```




```
means
```

```
##   Pclass Age
## 1      1  39
## 2      2  30
## 3      3  25
```

Repalce the missing age value the mean of age in each class.

```
titanic_mainData <- titanic_mainData

####################Replace the value with mean age of  class -1#############################
titanic_mainData$Age[is.na(titanic_mainData$Age) & titanic_mainData$Pclass == 1] = round(mean(filter(titanic_mainData,Pclass
==1)$Age, na.rm=TRUE),0)

#Verify if the replacement is correct
titanic_mainData$Age[titanic_mainData$Pclass == 1]
```

```
##   [1] 38.00 35.00 54.00 58.00 28.00 19.00 40.00 39.00 28.00 42.00 49.00
##  [12] 65.00 39.00 38.00 45.00 39.00 28.00 23.00 46.00 71.00 23.00 21.00
##  [23] 47.00 24.00 54.00 19.00 37.00 24.00 22.00 51.00 39.00 39.00 61.00
##  [34] 56.00 50.00 39.00 45.00 44.00 58.00 40.00 31.00 32.00 38.00 35.00
##  [45] 44.00 37.00 62.00 39.00 30.00 35.00 52.00 40.00 58.00 35.00 39.00
##  [56] 37.00 63.00 39.00 26.00 19.00 39.00  2.00 39.00 50.00  0.92 39.00
##  [67] 17.00 30.00 24.00 18.00 31.00 40.00 36.00 16.00 45.50 38.00 39.00
##  [78] 29.00 41.00 45.00 24.00 39.00 22.00 60.00 24.00 25.00 22.00 39.00
##  [89] 27.00 42.00 35.00 36.00 23.00 33.00 28.00 50.00 14.00 64.00  4.00
## [100] 34.00 52.00 30.00 49.00 65.00 39.00 48.00 47.00 56.00 39.00 25.00
## [111] 35.00 58.00 55.00 71.00 54.00 25.00 16.00 18.00 39.00 36.00 54.00
## [122] 47.00 30.00 44.00 39.00 45.00 30.00 22.00 36.00 50.00 64.00 17.00
## [133] 62.00 48.00 39.00 39.00 53.00 36.00 39.00 39.00 36.00 18.00 60.00
## [144] 52.00 49.00 39.00 35.00 27.00 40.00 42.00 61.00 21.00 80.00 32.00
## [155] 39.00 24.00 48.00 56.00 58.00 50.00 47.00 39.00 31.00 36.00 27.00
## [166] 15.00 31.00 60.00 49.00 18.00 35.00 42.00 22.00 24.00 39.00 48.00
## [177] 38.00 27.00 29.00 35.00 39.00 36.00 21.00 70.00 19.00 33.00 36.00
## [188] 51.00 39.00 43.00 17.00 29.00 46.00 39.00 49.00 11.00 39.00 33.00
## [199] 39.00 52.00 38.00 62.00 39.00 39.00 30.00 39.00 16.00 45.00 51.00
## [210] 48.00 31.00 47.00 33.00 56.00 19.00 26.00 46.00 23.00 47.00 55.00
## [221] 39.00 21.00 48.00 22.00 41.00 30.00 39.00 45.00 45.00 60.00 24.00
## [232] 28.00 36.00 13.00 47.00 31.00 60.00 28.50 35.00 32.50 55.00 67.00
## [243] 49.00 27.00 25.00 76.00 43.00 36.00 63.00 36.00 35.00 53.00 33.00
## [254] 61.00 42.00 39.00 39.00 23.00 29.00 42.00 48.00 39.00 54.00 64.00
## [265] 37.00 18.00 27.00 39.00  6.00 47.00 39.00 33.00 42.00 57.00 50.00
## [276] 53.00 21.00 39.00 64.00 48.00 55.00 45.00 41.00 27.00 39.00 46.00
## [287] 26.00 24.00 39.00 53.00 30.00 64.00 30.00 55.00 55.00 57.00 33.00
## [298] 39.00 46.00 39.00 30.00 58.00 45.00 50.00 59.00 25.00 45.00 31.00
## [309] 49.00 54.00 45.00 55.00 23.00 51.00 18.00 48.00 30.00 22.00 17.00
## [320] 43.00 50.00 37.00 39.00
```

```
# As per from box plot the average is almost 39
round(mean(filter(titanic_mainData,Pclass==1)$Age, na.rm=TRUE),0)
```

```
## [1] 39
```

```
##############Replace age for class =2 #####################################
titanic_mainData$Age[is.na(titanic_mainData$Age) & titanic_mainData$Pclass == 2] = round(mean(filter(titanic_mainData,Pclass
==2)$Age, na.rm=TRUE),0)

#Verify if the replacement is correct
titanic_mainData$Age[titanic_mainData$Pclass == 2]
```

```
##   [1] 14.00 55.00 30.00 35.00 34.00 66.00 27.00  3.00 29.00 21.00  5.00
##  [12] 29.00 32.00 21.00  0.83 17.00 34.00 34.00 29.00 21.00 32.50 32.50
##  [23] 29.00 25.00 23.00 18.00 19.00 36.50 42.00 51.00 40.00 30.00 30.00
##  [34]  1.00 32.00 19.00  3.00 24.00 35.00 30.00 42.00 30.00 27.00 19.00
##  [45] 18.00 59.00 24.00 44.00  8.00 19.00 33.00 29.00 24.00 54.00 50.00
##  [56] 36.00 41.00 30.00 42.00 36.00 30.00 30.00 26.00 43.00 24.00 54.00
##  [67] 30.00 22.00 36.00  2.00 28.00 25.00 36.00 24.00 40.00 38.00 29.00
##  [78] 18.00 36.00 17.00 46.00 23.00 28.00 34.00  3.00 30.00 34.00 18.00
##  [89] 30.00 28.00 19.00 42.00 24.00 31.00 45.00 28.00 13.00 36.00 50.00
## [100] 48.00 30.00 33.00 23.00 34.00 30.00 33.00 34.00 36.00 50.00 23.00
## [111]  2.00  7.00 32.00 19.00 30.00  8.00 27.00 28.00 62.00 34.00 25.00
## [122] 54.00 47.00 37.00 30.00 24.00 22.00 24.00  4.00 26.00 57.00 28.00
## [133] 31.00 18.00 24.00 23.00 32.00 25.00 40.00 70.00 31.00 30.00 60.00
## [144] 25.00 52.00 39.00 45.00 52.00 27.00  6.00 34.00 50.00 30.00 25.00
## [155] 30.00 23.00 23.00 30.00  4.00 48.00  0.67 18.00 57.00 54.00 16.00
## [166] 39.00 34.00 31.00 39.00 35.00 31.00  1.00  0.83 16.00 28.00 44.00
## [177] 21.00 24.00 42.00 27.00 28.00 25.00 28.00 27.00 62.00 26.00 63.00
## [188] 24.00 35.00 50.00 24.00 30.00 27.00 20.00 30.00 32.00 30.00 30.00
## [199] 30.00  2.00 27.00 18.50 41.00 29.00 12.00 42.00 26.00 28.00 30.00
## [210] 26.00 41.00 15.00 20.00 36.00 30.00 40.00 21.00 40.00 34.00 61.00
## [221]  8.00 23.00  8.00 25.00 24.00 17.00 60.00 30.00 22.00 36.00 14.00
## [232] 18.00 45.00 22.00 42.00 29.00  0.92 19.00 29.00 30.00 20.00 28.00
## [243] 40.00 30.00 22.00  1.00 30.00 43.00 19.00 22.00 26.00 12.00 29.00
## [254] 21.00 48.00 32.00 25.00 18.00 26.00 24.00 31.00 25.00 18.00 49.00
## [265] 24.00 31.00 29.00 21.00 44.00 21.00 30.00 24.00 57.00 47.00 38.00
## [276] 20.00 23.00
```

```
# As per from box plot the average is almost 30
round(mean(filter(titanic_mainData,Pclass==2)$Age, na.rm=TRUE),0)
```

```
## [1] 30
```

```
##############Replace age for class =3 ##########################################
titanic_mainData$Age[is.na(titanic_mainData$Age) & titanic_mainData$Pclass == 3] = round(mean(filter(titanic_mainData,Pclass
==3)$Age, na.rm=TRUE),0)

#Verify if the replacement is correct
titanic_mainData$Age[titanic_mainData$Pclass == 3]
```

```
##   [1] 22.00 26.00 35.00 25.00  2.00 27.00  4.00 20.00 39.00 14.00  2.00
##  [12] 31.00 25.00 15.00  8.00 38.00 25.00 25.00 25.00 25.00 25.00 21.00
##  [23] 18.00 14.00 40.00 25.00 19.00 25.00 25.00 25.00 25.00 18.00  7.00
##  [34] 21.00 28.50 11.00 22.00  4.00 25.00 19.00 17.00 26.00 16.00 26.00
##  [45] 32.00 25.00 25.00 25.00 30.00 22.00 29.00 25.00 33.00 16.00 25.00
##  [56] 24.00 29.00 20.00 26.00 59.00 25.00 28.00 25.00 33.00 37.00 28.00
##  [67] 21.00 25.00 38.00 25.00 14.50 22.00 20.00 17.00 21.00 70.50  2.00
##  [78] 25.00 12.00 25.00 24.00 25.00 45.00 33.00 20.00 47.00 16.00 25.00
##  [89] 22.00 24.00 19.00 27.00  9.00 55.50 40.50 25.00 16.00 30.00 25.00
## [100] 25.00 44.00 26.00 17.00  1.00  9.00 45.00 28.00  4.00  1.00 21.00
## [111] 18.00 25.00 36.00 25.00  9.00  4.00 25.00 40.00 36.00 19.00 25.00
## [122] 42.00 25.00 28.00 25.00 34.00 45.50 18.00  2.00 32.00 26.00 16.00
## [133] 24.00 22.00 25.00 27.00 16.00 51.00 25.00 22.00 20.50 25.00 29.00
## [144]  5.00 25.00 25.00 25.00 22.00 30.00 25.00 25.00 29.00 30.00 41.00
## [155] 29.00 25.00  3.00 25.00 16.00 25.00 25.00 25.00 45.00  7.00 35.00
## [166] 65.00 28.00 16.00 19.00 33.00 30.00 22.00 22.00 24.00 24.00 23.50
## [177] 25.00 25.00 19.00 25.00 28.00 26.00 22.00 27.00 25.00 61.00 31.00
## [188] 25.00 16.00 25.00 45.00 25.00  3.00 42.00 23.00 15.00 25.00 25.00
## [199] 28.00 25.00 25.00 40.00 45.00 35.00 25.00 30.00 25.00 25.00 18.00
## [210] 19.00  3.00 22.00 20.00 19.00  1.00 32.00 25.00  1.00 25.00 21.00
## [221] 28.00 24.00 22.00 31.00 39.00 26.00 21.00 28.00 20.00 51.00 21.00
## [232] 25.00 25.00 25.00 44.00 25.00 10.00 25.00 21.00 29.00 28.00 18.00
## [243] 25.00 25.00 32.00 25.00 17.00 21.00 20.00 25.00 25.00  5.00 25.00
## [254] 25.00 29.00 25.00 34.00 25.00 38.00 25.00  0.75 25.00 38.00 22.00
## [265] 29.00 22.00  2.00  9.00 50.00 63.00 25.00 30.00  9.00 25.00 21.00
## [276] 21.00 25.00 25.00 24.00 17.00 21.00 25.00 37.00 28.00 26.00 29.00
## [287] 25.00 24.00 25.00 32.00 22.00 25.00 25.00 40.50 39.00 25.00 17.00
## [298] 25.00 30.00 25.00  9.00 11.00 33.00 25.00 22.00 22.00 36.00 25.00
## [309] 40.00 25.00 25.00 24.00 19.00 29.00 25.00 32.00 25.00 16.00 19.00
## [320] 25.00 32.00 25.00 22.00 25.00 35.00 47.00 25.00 36.00 49.00 25.00
## [331] 25.00 44.00 36.00 30.00 39.00 25.00 25.00 25.00 35.00 34.00 26.00
## [342] 27.00 20.00 21.00 21.00 26.00 25.00 51.00  9.00 32.00 41.00 25.00
## [353] 20.00  2.00 25.00  0.75 19.00 25.00 23.00 25.00 21.00 25.00 18.00
## [364] 25.00 32.00 40.00 36.00 20.00 25.00 43.00 18.00 24.50 18.00 43.00
## [375] 25.00 20.00 14.00 14.00 19.00 18.00  4.00 25.00 25.00 44.00 25.00
## [386] 42.00 18.00 25.00 26.00 25.00 29.00 19.00 25.00 33.00 17.00 20.00
## [397] 25.00 25.00 11.00 28.50 48.00 25.00 25.00 24.00 31.00 16.00 31.00
## [408]  6.00 33.00 23.00 28.00 34.00 25.00 41.00 20.00 16.00 30.50 25.00
## [419] 32.00 24.00 48.00 25.00 18.00 25.00  5.00 25.00 13.00 25.00 25.00
```

```
## [430] 25.00 18.00  8.00  1.00 25.00 25.00 25.00 31.00 30.00 30.00  0.42
## [441] 27.00 31.00 18.00 26.00 39.00  6.00 30.50 23.00 43.00 10.00 27.00
## [452] 27.00  2.00 25.00 25.00 25.00 15.00 25.00 23.00 18.00 21.00 25.00
## [463] 32.00 20.00 34.50 17.00 42.00 25.00 35.00  4.00 74.00  9.00 18.00
## [474] 24.00 25.00 41.00 25.00 25.00  4.00 26.00 47.00 15.00 20.00 19.00
## [485] 25.00 33.00 22.00 25.00 39.00 25.00 32.00 34.50 47.00 27.00 22.00
## [496] 14.00 30.00 18.00 21.00 25.00 21.00 27.00 45.00  9.00 50.00 22.50
## [507] 25.00 33.00 25.00 18.50 25.00 21.00 25.00 25.00 39.00 41.00 25.00
## [518] 25.00 36.00 10.00 35.00 25.00 25.00 17.00 18.00 22.00 18.00 24.00
## [529] 21.00 29.00 25.00 24.00  6.00 25.00 25.00 27.00 18.00 25.00 22.00
## [540] 25.00 25.00 25.00 29.00 20.00 33.00 25.00 26.00 16.00 28.00 21.00
## [551] 25.00 25.00 25.00 18.50 18.00 25.00  1.00 25.00 28.00 25.00 17.00
## [562] 22.00 25.00 24.00 32.00 25.00 25.00 43.00 24.00 26.50 23.00 40.00
## [573] 10.00 31.00 22.00 25.00 60.50 36.00 13.00 24.00 23.00 26.00 25.00
## [584]  7.00 25.00 26.00 18.00 22.00 25.00 27.00 23.00 25.00 40.00 25.00
## [595] 17.00 25.00 11.50 33.00 18.00 25.00 25.00  0.33 35.00 25.00 32.00
## [606] 25.00 38.00 25.00 25.00 21.00 21.00 25.00 23.00 25.00 40.50 21.00
## [617] 25.00 20.00 20.00 25.00 25.00 25.00 20.00 24.00 32.50 25.00 25.00
## [628] 28.00 21.00 36.50 21.00  1.00 25.00 25.00 25.00 17.00 25.00 25.00
## [639] 25.00 23.00  0.75 25.00  9.00  2.00 36.00 25.00 25.00 25.00 30.00
## [650] 25.00 36.00 26.00 25.00 29.00 32.00 24.00 25.00  0.83 45.00 18.00
## [661] 22.00 25.00 37.00 17.00 27.00 26.00 25.00 23.00 25.00 19.00 27.00
## [672] 39.00 25.00 32.00 25.00 25.00 16.00 38.00  0.17 25.00 25.00 30.00
## [683] 14.50 27.00 25.00 25.00 22.00 22.00  5.00 25.00 26.00 25.00 19.00
## [694] 24.00 21.00  6.00 13.00 29.00 24.00 22.00 31.00 25.00  3.00 25.00
## [705] 28.00 25.00 38.50 25.00 25.00
```

```
# As per from box plot the average is almost 25
round(mean(filter(titanic_mainData,Pclass==3)$Age, na.rm=TRUE),0)
```

```
## [1] 25
```

Checking missing values

```
colSums(is.na(titanic_mainData)|titanic_mainData=='')
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0         418           0           0           0           0
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           0        1014           0
```

3. Feature engineering

3.1 Passenger Tile - Convert titles for male and female into standard format ie Mr or Mrs

```
head(titanic_mainData$Name)
```

```
## [1] "Braund, Mr. Owen Harris"
## [2] "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
## [3] "Heikkinen, Miss. Laina"
## [4] "Futrelle, Mrs. Jacques Heath (Lily May Peel)"
## [5] "Allen, Mr. William Henry"
## [6] "Moran, Mr. James"
```

Extract passenger title from name

```
titanic_mainData$Title <- gsub("^.*, (.*?)\\..*$", "\\1", titanic_mainData$Name)
head(titanic_mainData$Title)
```

```
## [1] "Mr"   "Mrs"  "Miss" "Mrs"  "Mr"   "Mr"
```

Occurance of title based on sex

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

```
##
##          Capt Col Don Dona  Dr Jonkheer Lady Major Master Miss Mlle Mme
##   female    0   0   0    1   1        0    1     0      0  260    2   1
##   male      1   4   1    0   7        1    0     2     61    0    0   0
##
##           Mr Mrs  Ms Rev Sir the Countess
##   female   0 197   2   0   0            1
##   male   757   0   0   8   1            0
```

Regularize the title for female

```
titanic_mainData$Title[titanic_mainData$Title %in%  c("Mlle", "Ms")] <- "Miss"
titanic_mainData$Title[titanic_mainData$Title == "Mme"] <- "Mrs"
```

```
titanic_mainData$Title[titanic_mainData$Title %in%  c(c('Dona', 'Dr', 'Lady', 'the Countess','Capt', 'Col', 'Don', 'Jonkhee
r', 'Major', 'Rev', 'Sir'))] <- "other"
```

Verify replacement

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

```
##
##          Master Miss  Mr Mrs other
##   female      0  264   0 198     4
##   male       61    0 757   0    25
```

Check family size, add feature

```
FamilyMember <- titanic_mainData$SibSp + titanic_mainData$Parch + 1
```

```
table(FamilyMember)
```

```
## FamilyMember
##   1   2   3   4   5   6   7   8  11 
## 790 235 159  43  22  25  16   8  11
```

```
titanic_mainData$FamilyMember <- FamilyMember
head(titanic_mainData$FamilyMember)
```

```
## [1] 2 2 1 2 1 1
```

```
#titanic_mainData$FamilyMember[titanic_mainData$FamilyMember==1] <- "Single"
#titanic_mainData$FamilyMember[titanic_mainData$FamilyMember=="Single"]
```

```
#titanic_mainData$FamilyMember
#titanic_mainData$FamilyMember[titanic_mainData$FamilyMember > 1 & titanic_mainData$FamilyMember <= 4 & titanic_mainData$Fam
ilyMember != "Single"] <- "Small"
```

```
#titanic_mainData$FamilyMember
#titanic_mainData$FamilyMember[titanic_mainData$FamilyMember >= 4 & titanic_mainData$FamilyMember != "Small" & titanic_mainD
ata$FamilyMember != "Single"] <- "Large"
```

```
#titanic_mainData$FamilyMember=="Single"
```

```
#table(titanic_mainData$FamilyMember)
```

```
titanic_mainData$FamilyMember <- sapply(1:nrow(titanic_mainData), function(x)
                          ifelse(FamilyMember[x]==1, "Single",
                          ifelse(FamilyMember[x]>4, "Large", "Small")))
```

```
table(titanic_mainData$FamilyMember)
```

```
##
##  Large Single  Small
##     82    790    437
```

4.Data visualization

```
str(titanic_mainData)
```

```
## 'data.frame':    1309 obs. of  14 variables:
##  $ PassengerId : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived    : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass      : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name        : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)" "Heikkinen, Miss. L
aina" "Futrelle, Mrs. Jacques Heath (Lily May Peel)" ...
##  $ Sex         : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age         : num  22 38 26 35 35 25 54 2 27 14 ...
##  $ SibSp       : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch       : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket      : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare        : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin       : chr  "" "C85" "" "C123" ...
##  $ Embarked    : chr  "S" "C" "S" "S" ...
##  $ Title       : chr  "Mr" "Mrs" "Miss" "Mrs" ...
##  $ FamilyMember: chr  "Small" "Small" "Single" "Small" ...
```

```
class(titanic_mainData$Survived)
```

```
## [1] "integer"
```

```
titanic_mainData$Survived <- factor(titanic_mainData$Survived)
titanic_mainData$Pclass = factor(titanic_mainData$Pclass)
titanic_mainData$Sex = factor(titanic_mainData$Sex)
titanic_mainData$Embarked = factor(titanic_mainData$Embarked)
titanic_mainData$Title = factor(titanic_mainData$Title)
titanic_mainData$FamilyMember = factor(titanic_mainData$FamilyMember)
```

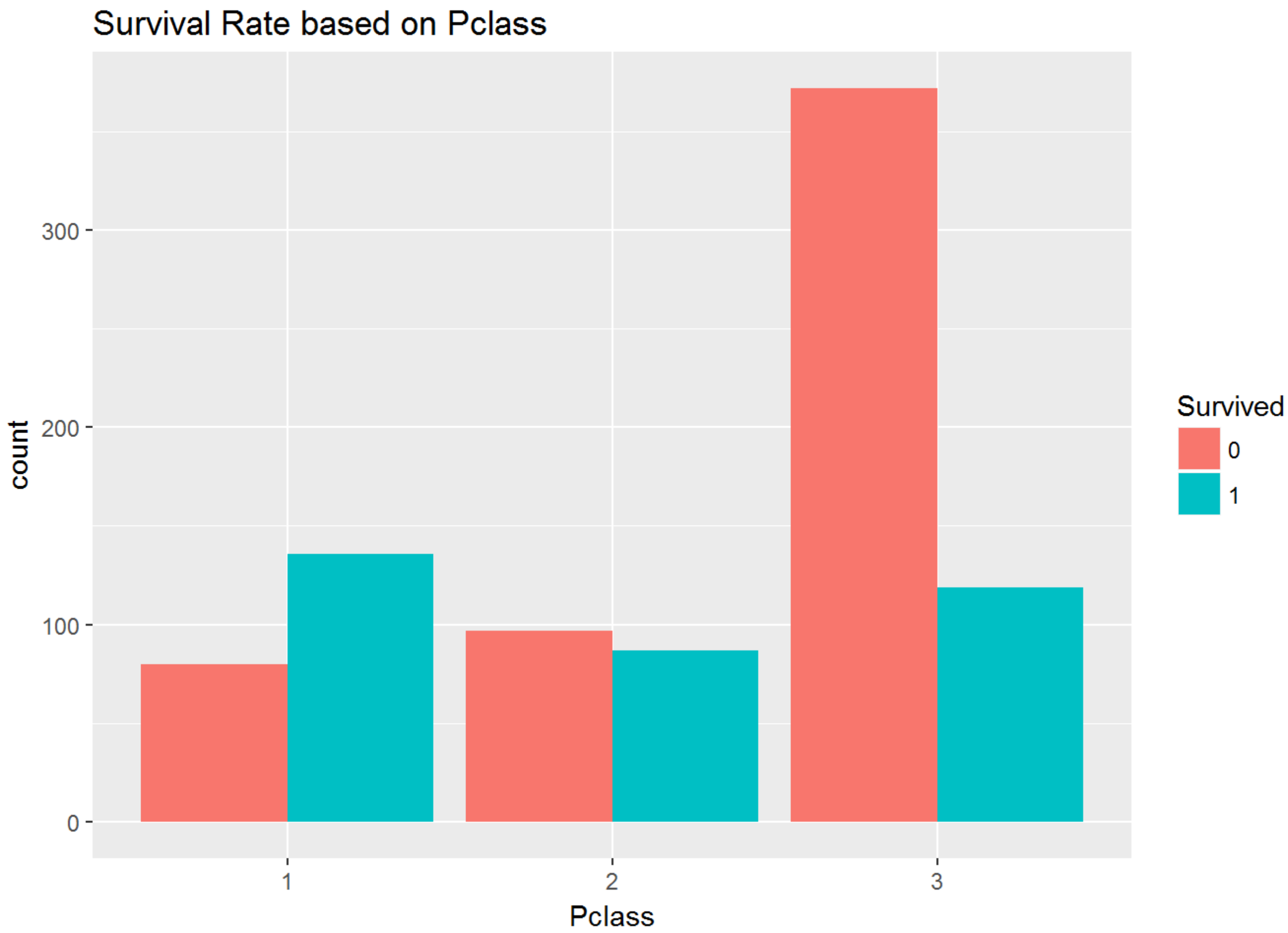
Below are the new structure with factor columns.

```
str(titanic_mainData)
```

```
## 'data.frame':    1309 obs. of  14 variables:
##  $ PassengerId : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived    : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ Pclass      : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
##  $ Name        : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)" "Heikkinen, Miss. L
aina" "Futrelle, Mrs. Jacques Heath (Lily May Peel)" ...
##  $ Sex         : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age         : num  22 38 26 35 35 25 54 2 27 14 ...
##  $ SibSp       : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch       : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket      : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare        : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin       : chr  "" "C85" "" "C123" ...
##  $ Embarked    : Factor w/ 3 levels "C","Q","S": 3 1 3 3 3 2 3 3 3 1 ...
##  $ Title       : Factor w/ 5 levels "Master","Miss",..: 3 4 2 4 3 3 3 1 4 4 ...
##  $ FamilyMember: Factor w/ 3 levels "Large","Single",..: 3 3 2 3 2 2 2 1 3 3 ...
```

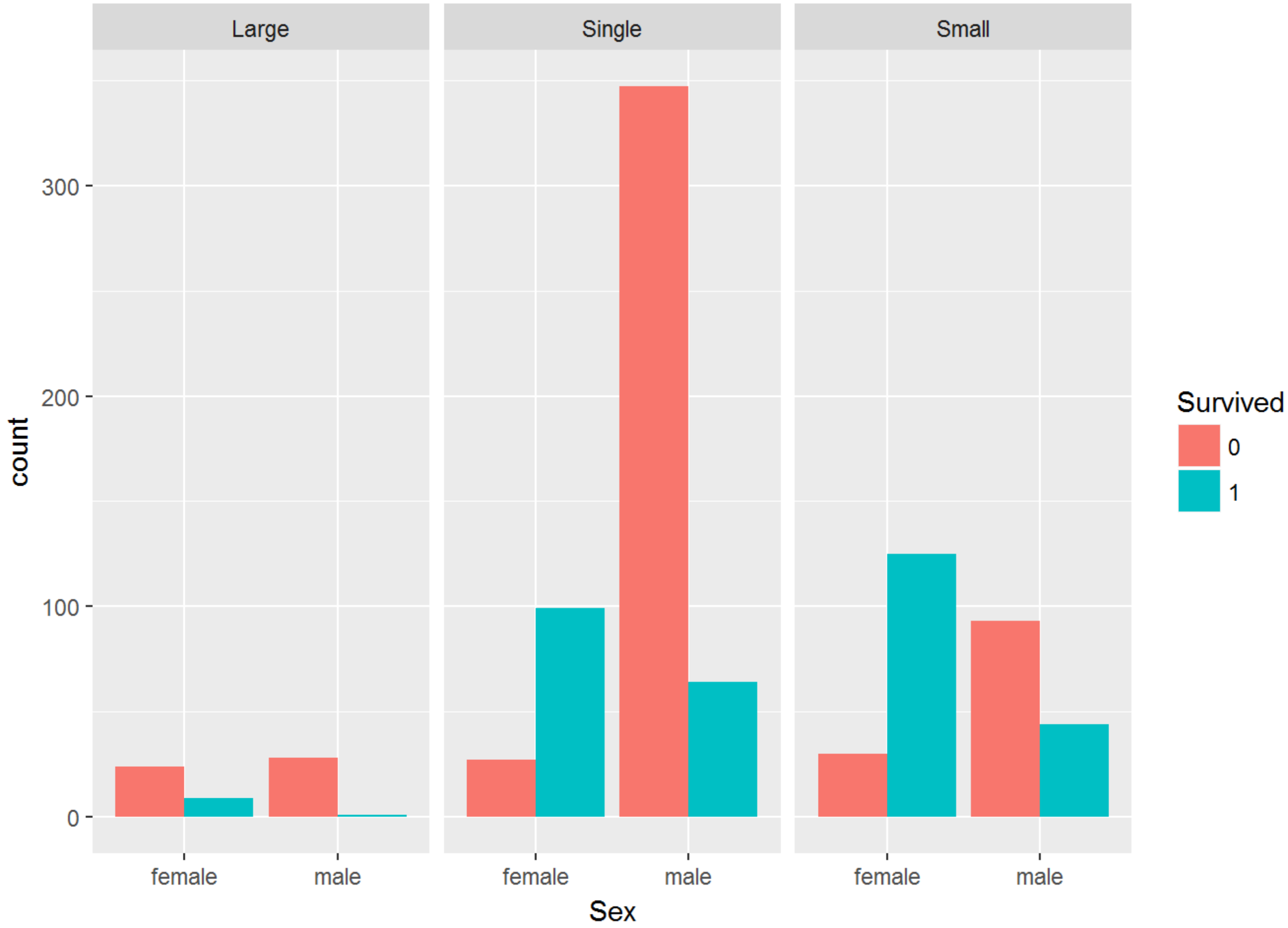
Data analysis for survival based on Pclass.

```
ggplot(data=filter(titanic_mainData, is.na(Survived)==FALSE), aes(Pclass, fill=Survived)) +
  geom_bar(position="dodge") +
  ggtitle("Survival Rate based on Pclass")
```



Survival based on Sex.

```
ggplot(data=filter(titanic_mainData, is.na(Survived)==FALSE), aes(Sex, fill=Survived)) +
  geom_bar(position="dodge" ) +
  facet_wrap(~FamilyMember)
```

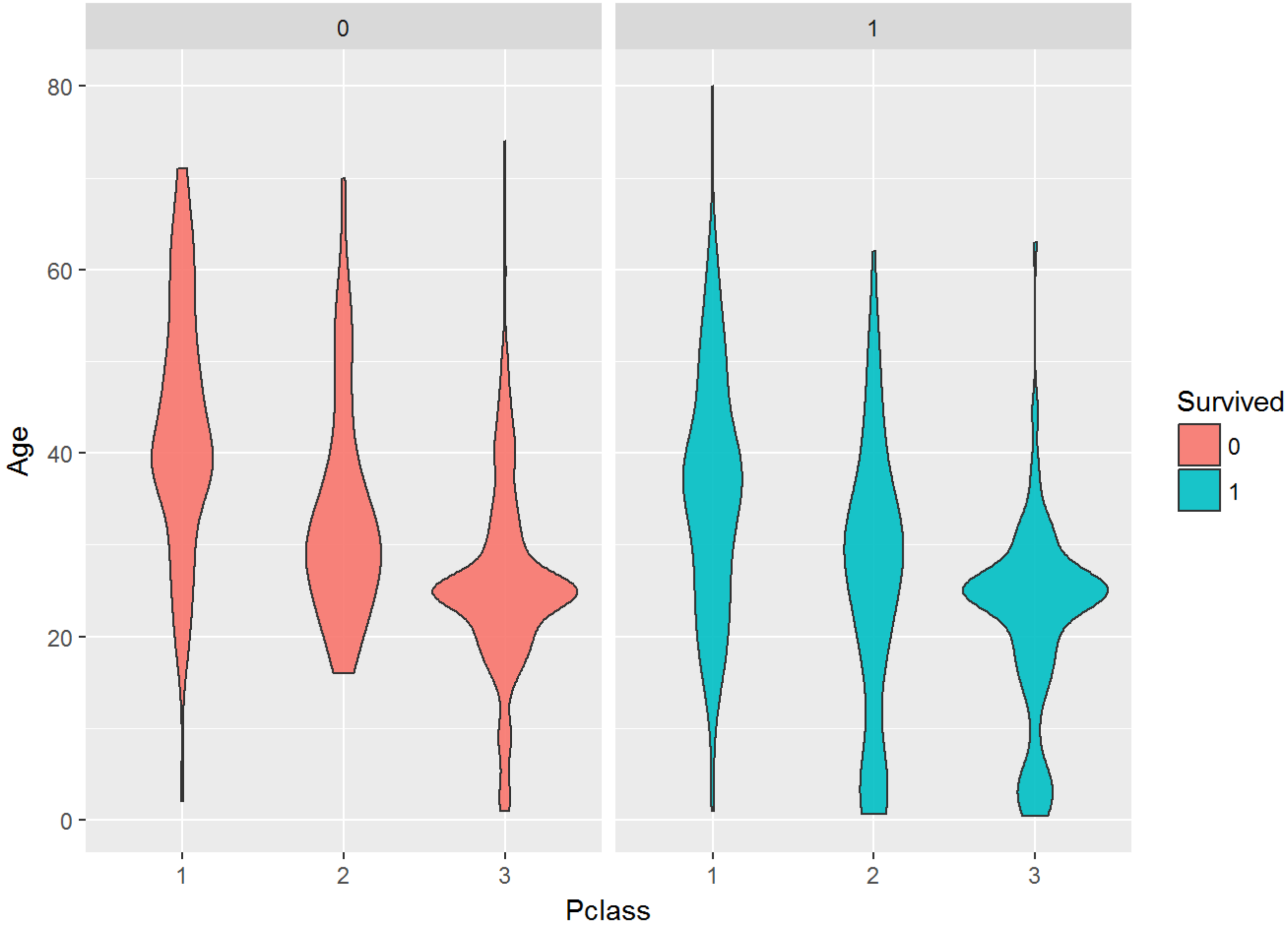



```
ggtitle("Survival Rate based on Sex")
```

```
## $title
## [1] "Survival Rate based on Sex"
##
## $subtitle
## NULL
##
## attr(,"class")
## [1] "labels"
```

Survival based on age.

```
ggplot(data=filter(titanic_mainData, is.na(Survived)==FALSE), aes(y=Age,x=Pclass)) + geom_violin(aes(fill=Survived), alpha=
0.9) +
  facet_wrap(~Survived)
```

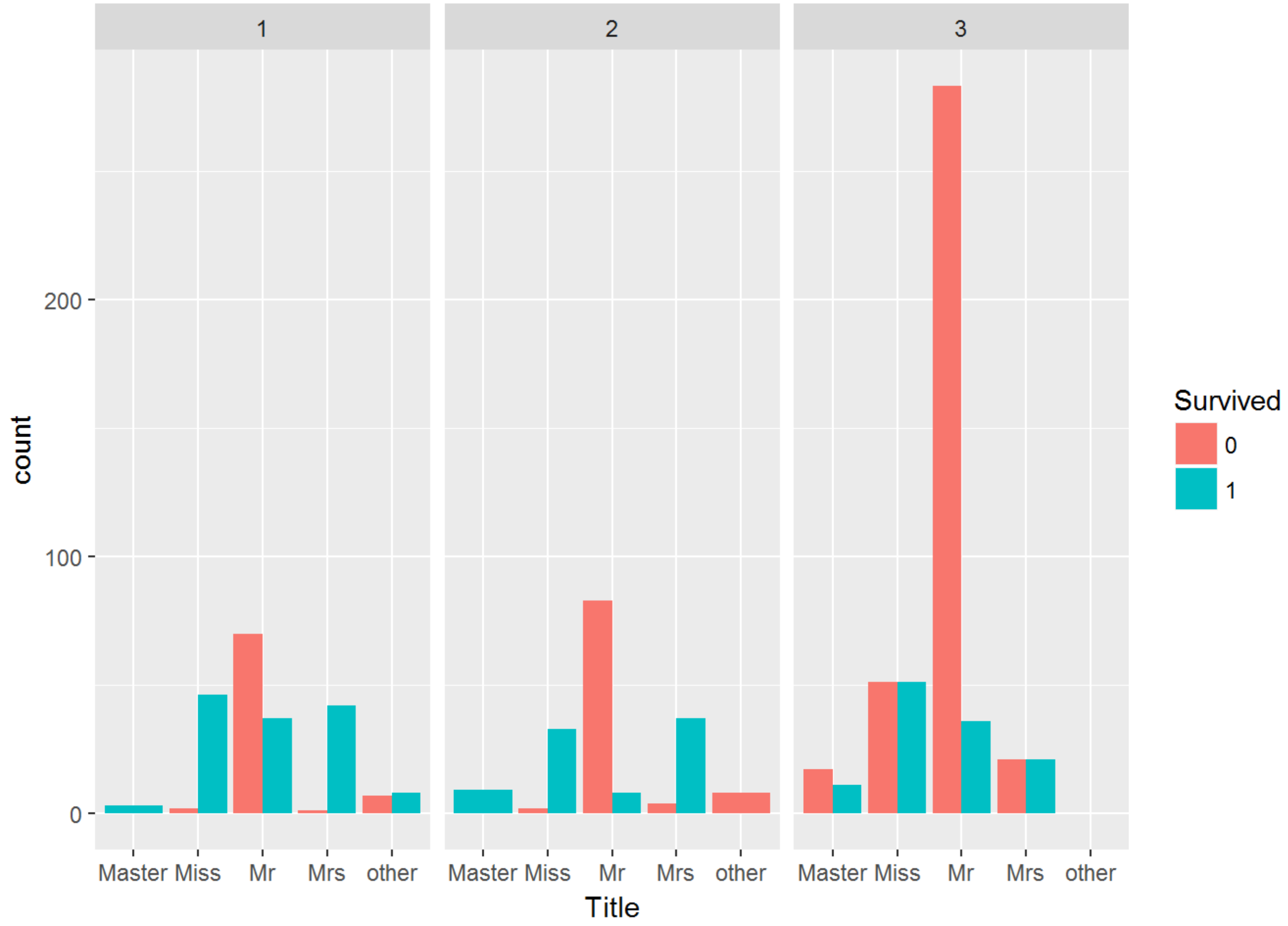



```
  ggtitle("Survival Rate based on Pclass")
```

```
## $title
## [1] "Survival Rate based on Pclass"
##
## $subtitle
## NULL
##
## attr(,"class")
## [1] "labels"
```

Survival based in title.

```
ggplot(data=filter(titanic_mainData, is.na(Survived)==FALSE), aes(Title, fill=Survived)) +
  geom_bar(position="dodge" ) +
  facet_wrap(~Pclass)
```

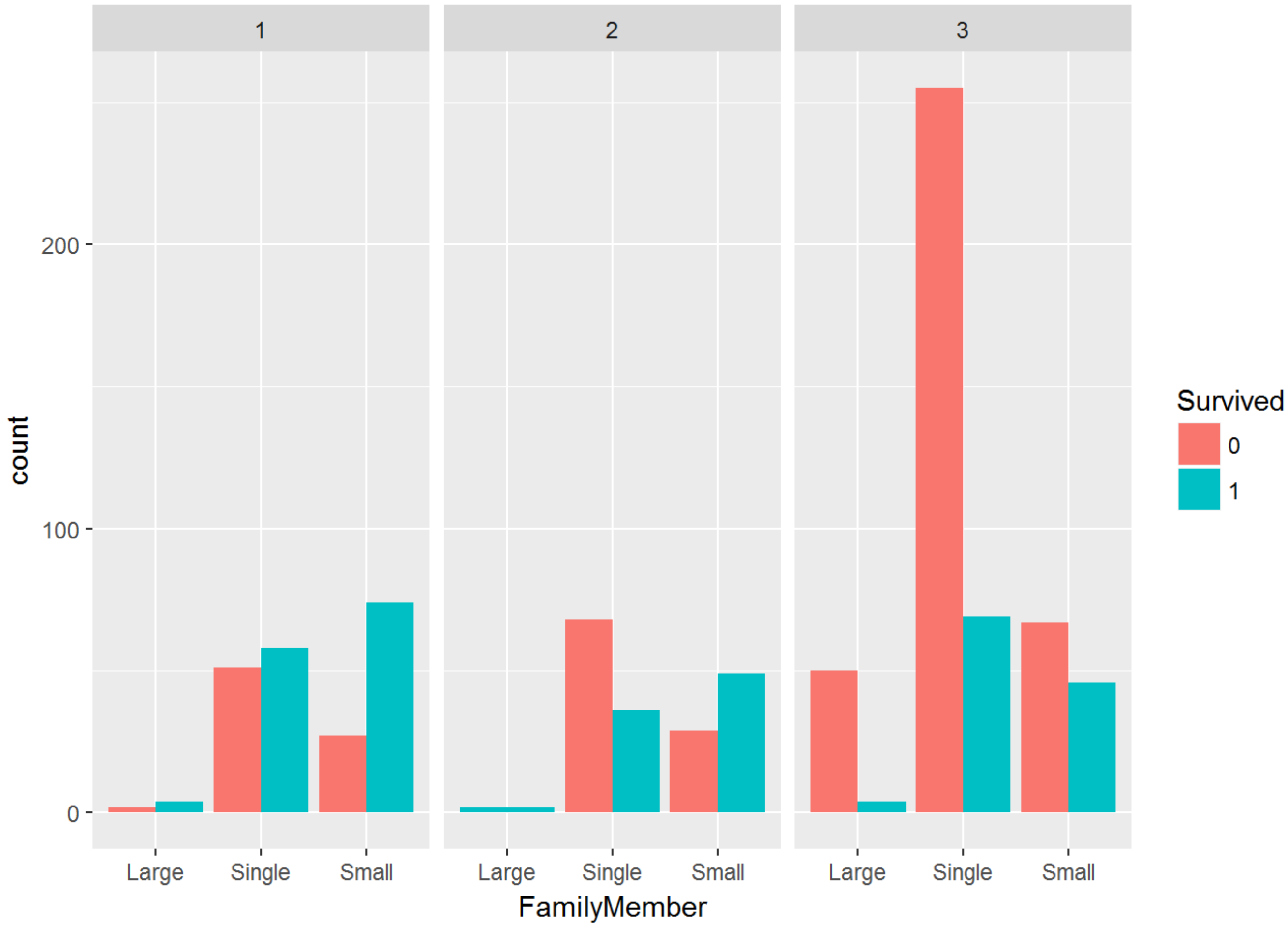



```
ggtitle("Survival Rate based on Title")
```

```
## $title
## [1] "Survival Rate based on Title"
##
## $subtitle
## NULL
##
## attr(,"class")
## [1] "labels"
```

Survival based on familySize.

```
ggplot(data=filter(titanic_mainData, is.na(Survived)==FALSE), aes(FamilyMember, fill=Survived)) +
  geom_bar(position="dodge" ) +
  facet_wrap(~Pclass)
```

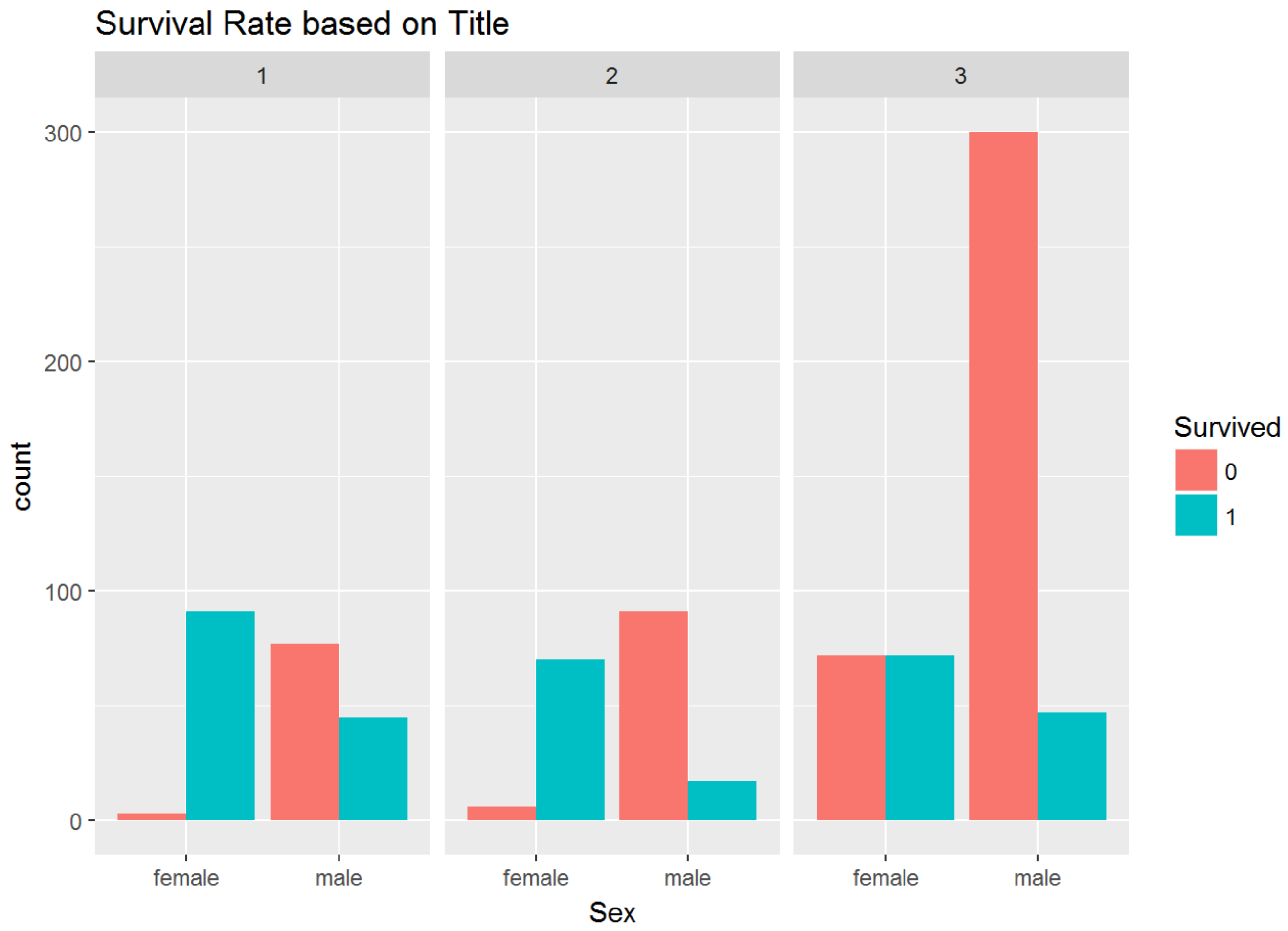



```
  ggtitle("Survival Rate based on Title")
```

```
## $title
## [1] "Survival Rate based on Title"
##
## $subtitle
## NULL
##
## attr(,"class")
## [1] "labels"
```

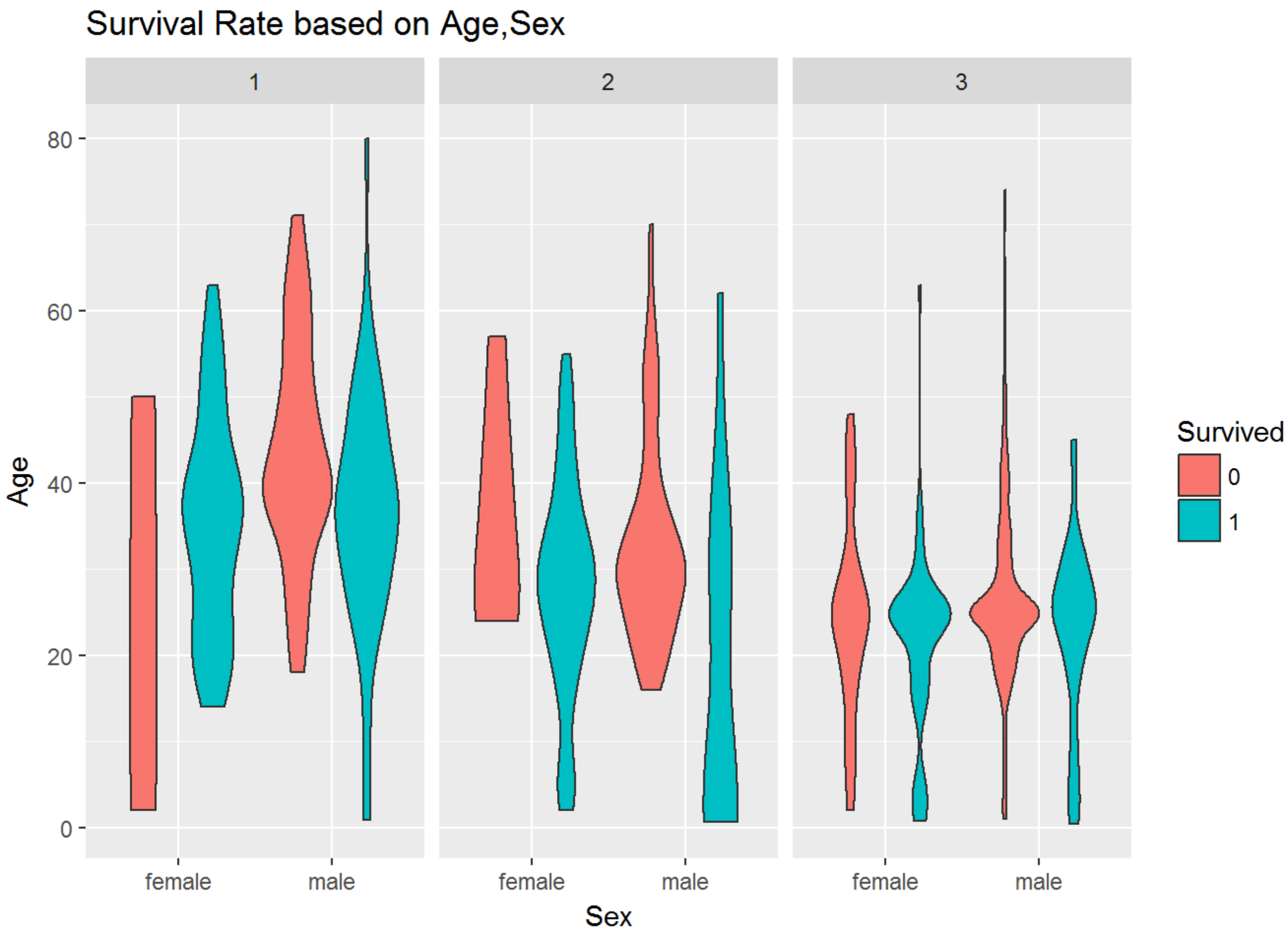
```
ggplot(data=filter(titanic_mainData, is.na(Survived)==FALSE), aes(Sex, fill=Survived)) +
  #geom_bar(stat="count")+
  geom_bar(position="dodge" ) +
  facet_wrap(~Pclass) +

  ggtitle("Survival Rate based on Title")
```

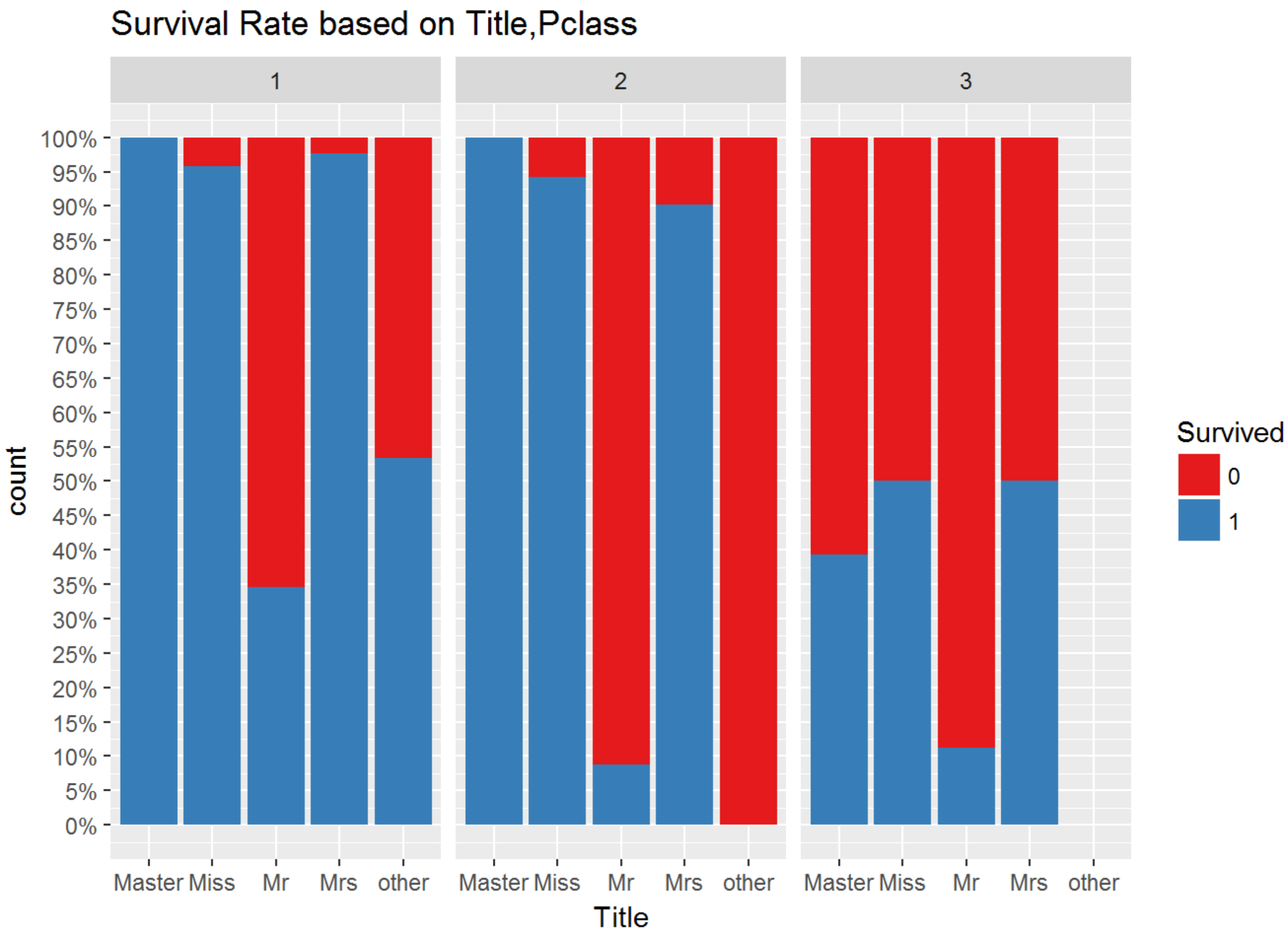



```
ggplot(data=filter(titanic_mainData, is.na(Survived)==FALSE), aes(y=Age,x=Sex)) +
  #geom_point() +
  geom_violin(aes(fill=Survived)) +
  facet_wrap(~Pclass) +

  #scale_y_continuous(labels = percent)+
  ggtitle("Survival Rate based on Age,Sex")
```



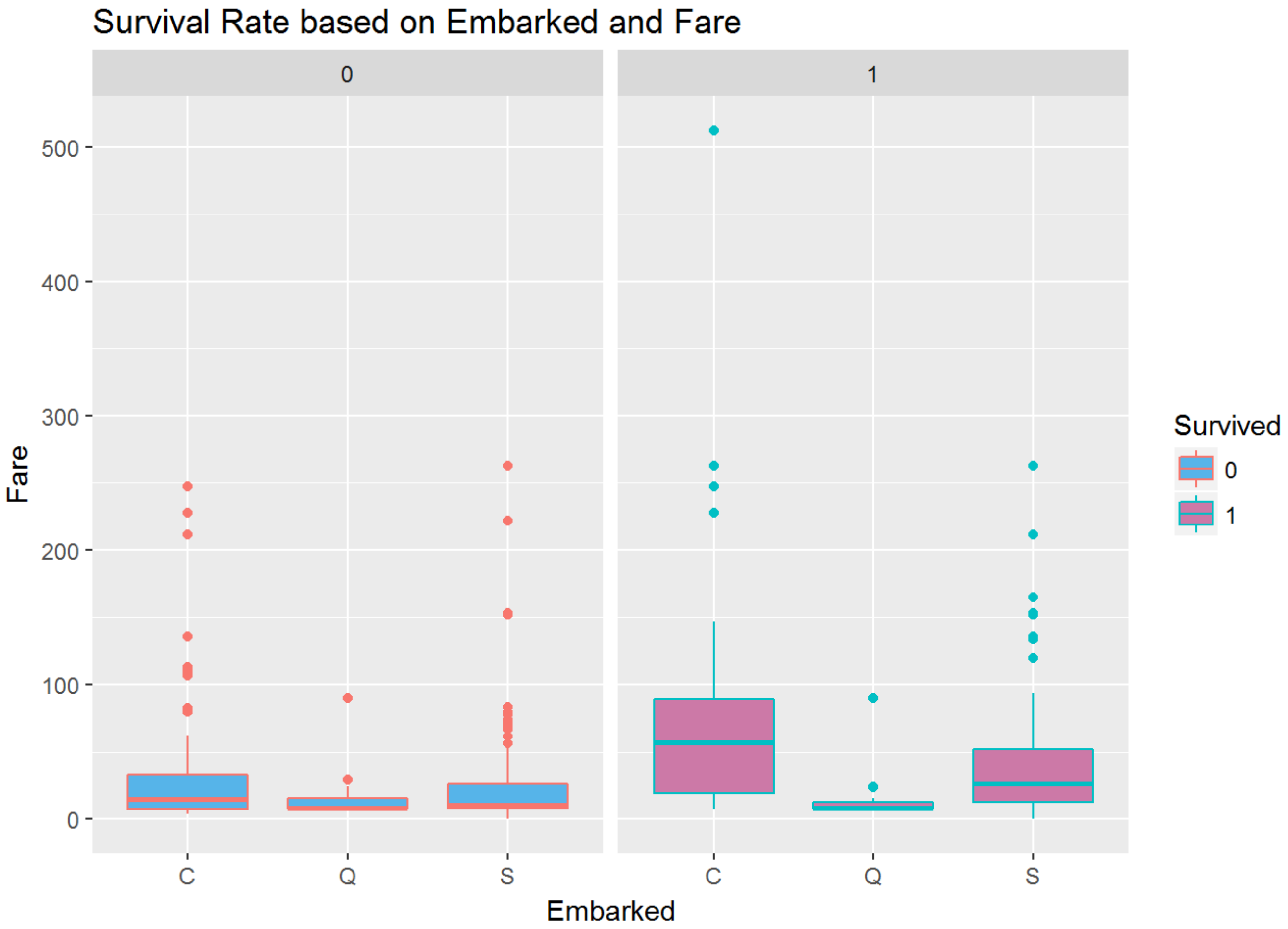

```
ggplot(data=filter(titanic_mainData, is.na(Survived)==FALSE), aes(Title, fill= Survived)) +
 geom_bar(position="fill")+
  facet_wrap(~Pclass) +
  scale_fill_brewer(palette="Set1") +
  scale_y_continuous(labels=percent, breaks=seq(0,1,0.05)) +
   ggtitle("Survival Rate based on Title,Pclass")
```



Data Analysis based on fare and embarked.

Survival based on Embarked and Fare.

```
ggplot(filter(titanic_mainData, is.na(Survived)==FALSE), aes(Embarked, Fare, colour = Survived)) +
  #geom_boxplot(aes(fill=Survived), alpha=0.9) +
  geom_boxplot(aes(fill=Survived)) +
  facet_wrap(~Survived) +
  scale_fill_manual(values=c("#56B4E9", "#CC79A7")) +
  ggtitle("Survival Rate based on Embarked and Fare")
```



Interestingly, there is a substantial variation of fares in the survived category, especially from Cherbourg and Southampton ports.

Visual analysis of data concludes:

-The wealthier passengers in the first class had a higher survival rate;

-Females had a higher survival rate than males in each class;

-Male “Mr” passengers had the lowest survival rate amongst all the classes; and

-Large families had the worst survival rate than singletons and small families.

Looking into visualization features :Pclass, Sex, Age, SibSp, Parch, Fare, Embarked, Title and FamilyMember are useful, ignore the Name, Ticket and Cabin

5. Algorithm

5.1 Splitting the dataset into the Training set and Test set

We have done all the data manipulation and data transformation, we can divide test and train

Divide the dataset into the Training set and Test set

```
train_original <- titanic_mainData[1: 891, c("Survived", "Pclass", "Sex","Age","SibSp","Parch","Fare","Embarked","Title","FamilyMember")]
```

```
train_test <- titanic_mainData[892: 1309, c( "Pclass", "Sex","Age","SibSp","Parch","Fare","Embarked","Title","FamilyMember")]
```

5.2 Splitting the training set into the Training set and Validation set

cor(train_original[sapply(train_original, is.numeric)])

Split the training set into the training set (80% of training data) and validation set (20% of training data) for the evaluation purposes of the fitted models.

Splitting the Training set into the Training set and Validation set

```
set.seed(789)

split = sample.split(train_original$Survived, SplitRatio = 0.8)
train = subset(train_original, split == TRUE)
test = subset(train_original, split == FALSE)
```

5.3 Logistic Regression

Before we go ahead with Logistic regression, Let's check the Logistic Regression assumptions: features should be independent from each other and residuals are not autocorrelated.

Show the correlation of numeric features

```
cor(train_original[sapply(train, is.numeric)])
```

```
##              Age      SibSp      Parch      Fare
## Age    1.0000000 -0.2438627 -0.1760528 0.1222818
## SibSp -0.2438627  1.0000000  0.4148377 0.1596510
## Parch -0.1760528  0.4148377  1.0000000 0.2162249
## Fare   0.1222818  0.1596510  0.2162249 1.0000000
```

In statistics, two variables are strongly correlated if the correlation coefficient is more than 0.75 (Threshold values can be 0.70 or 0.8) or less than -0.75. After looking into the correlation matrix, none of the numeric features are strongly correlated. Hence, the Multicollinearity (a given feature in the model can be approximated by a linear combination of the other features in the model) does not exist among numeric features.

Show the p-value of Chi Square tests

```
ps = chisq.test(train$Pclass, train$Sex)$p.value
pe = chisq.test(train$Pclass, train$Embarked)$p.value
pt = chisq.test(train$Pclass, train$Title)$p.value
```

```
## Warning in chisq.test(train$Pclass, train$Title): Chi-squared approximation
## may be incorrect
```

```
pf = chisq.test(train$Pclass, train$FamilyMember)$p.value
se = chisq.test(train$Sex, train$Embarked)$p.value
st = chisq.test(train$Sex, train$Title)$p.value
sf = chisq.test(train$Sex, train$FamilyMember)$p.value
et = chisq.test(train$Embarked, train$Title)$p.value
```

```
## Warning in chisq.test(train$Embarked, train$Title): Chi-squared
## approximation may be incorrect
```

```
ef = chisq.test(train$Embarked, train$FamilyMember)$p.value
tf = chisq.test(train$Title, train$FamilyMember)$p.value
```

```
## Warning in chisq.test(train$Title, train$FamilyMember): Chi-squared
## approximation may be incorrect
```

```
cormatrix = matrix(c(0, ps, pe, pt, pf,
                     ps, 0, se, st, sf,
                     pe, se, 0, et, ef,
                     pt, st, et, 0, tf,
                     pf, sf, ef, tf, 0),
                   5, 5, byrow = TRUE)


cormatrix
```

```
##              [,1]          [,2]         [,3]          [,4]         [,5]
## [1,] 0.000000e+00  2.532566e-03 1.053100e-23  5.962301e-10 1.108964e-10
## [2,] 2.532566e-03  0.000000e+00 1.321593e-02 1.116723e-150 4.591649e-15
## [3,] 1.053100e-23  1.321593e-02 0.000000e+00  1.383169e-04 2.490631e-06
## [4,] 5.962301e-10 1.116723e-150 1.383169e-04  0.000000e+00 2.204782e-51
## [5,] 1.108964e-10  4.591649e-15 2.490631e-06  2.204782e-51 0.000000e+00
```

```
colnames(cormatrix) = c("Pclass", "Sex", "Embarked", "Title", "FamilySize")
row.names(cormatrix) = c("Pclass", "Sex", "Embarked", "Title", "FamilySize")
cormatrix
```

```
##                  Pclass           Sex     Embarked         Title
## Pclass     0.000000e+00  2.532566e-03 1.053100e-23  5.962301e-10
## Sex        2.532566e-03  0.000000e+00 1.321593e-02 1.116723e-150
## Embarked   1.053100e-23  1.321593e-02 0.000000e+00  1.383169e-04
## Title      5.962301e-10 1.116723e-150 1.383169e-04  0.000000e+00
## FamilySize 1.108964e-10  4.591649e-15 2.490631e-06  2.204782e-51
##              FamilySize
## Pclass     1.108964e-10
## Sex        4.591649e-15
## Embarked   2.490631e-06
## Title      2.204782e-51
## FamilySize 0.000000e+00
```

Chi Square is used to find the corelation between between the categorical/qualitative features. We can see that all the features has $p < 0.05$, hence they are corelated. The features are not independent and multicollinearity exists among them.

Fitting the Logistic regression model on traning set

```
glm.fit <- glm(Survived ~ .,  family=binomial(link='logit') , data=train)
#glm.fit <- glm(Survival ~ .,  family="binomial" , data=train)
```

Using the best model by selecting AIC

Step() will run the model for each variable iteratively and give the best model on top as per ranking on AIC

```
glm.fit <- step(glm.fit)
```

```
## Start:  AIC=612.29
## Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked +
##     Title + FamilyMember
##
##                Df Deviance    AIC
## - SibSp         1   580.29 610.29
## - Embarked      2   582.69 610.69
## - Fare          1   580.81 610.81
## - Parch         1   581.59 611.59
## <none>              580.29 612.29
## - Sex           1   584.37 614.37
## - Age           1   585.69 615.69
## - FamilyMember  2   590.68 618.68
## - Title         4   616.14 640.14
## - Pclass        2   624.73 652.73
##
## Step:  AIC=610.29
## Survived ~ Pclass + Sex + Age + Parch + Fare + Embarked + Title +
##     FamilyMember
##
##                Df Deviance    AIC
## - Embarked      2   582.71 608.71
## - Fare          1   580.82 608.82
## - Parch         1   582.05 610.05
## <none>              580.29 610.29
## - Sex           1   584.37 612.37
## - Age           1   585.70 613.70
## - FamilyMember  2   609.33 635.33
## - Title         4   616.76 638.76
## - Pclass        2   624.87 650.87
##
## Step:  AIC=608.71
## Survived ~ Pclass + Sex + Age + Parch + Fare + Title + FamilyMember
##
##                Df Deviance    AIC
## - Fare          1   583.56 607.56
## - Parch         1   584.41 608.41
## <none>              582.71 608.71
## - Sex           1   586.56 610.56
```

```
## - Age           1   588.11 612.11
## - FamilyMember  2   615.00 637.00
## - Title         4   619.32 637.32
## - Pclass        2   628.03 650.03
##
## Step:  AIC=607.56
## Survived ~ Pclass + Sex + Age + Parch + Title + FamilyMember
##
##                Df Deviance    AIC
## - Parch         1   585.45 607.45
## <none>              583.56 607.56
## - Sex           1   587.31 609.31
## - Age           1   589.24 611.24
## - FamilyMember  2   615.02 635.02
## - Title         4   619.43 635.43
## - Pclass        2   662.23 682.23
##
## Step:  AIC=607.45
## Survived ~ Pclass + Sex + Age + Title + FamilyMember
##
##                Df Deviance    AIC
## <none>              585.45 607.45
## - Sex           1   589.15 609.15
## - Age           1   591.29 611.29
## - Title         4   623.47 637.47
## - FamilyMember  2   622.80 640.80
## - Pclass        2   664.64 682.64
```

```
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age + Title + FamilyMember,
##     family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6190  -0.5712  -0.3802   0.5462   2.4571
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          16.06908  506.79961   0.032 0.974706
## Pclass2              -1.42636    0.32191  -4.431 9.38e-06 ***
## Pclass3              -2.55761    0.31174  -8.204 2.32e-16 ***
## Sexmale             -14.63628  506.79929  -0.029 0.976960
## Age                  -0.02518    0.01062  -2.370 0.017789 *
## TitleMiss           -15.04068  506.79960  -0.030 0.976324
## TitleMr              -3.36409    0.60123  -5.595 2.20e-08 ***
## TitleMrs            -14.59001  506.79969  -0.029 0.977033
## Titleother           -2.96720    0.85828  -3.457 0.000546 ***
## FamilyMemberSingle    2.65324    0.50371   5.267 1.38e-07 ***
## FamilyMemberSmall     2.41867    0.48424   4.995 5.89e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 949.90  on 712  degrees of freedom
## Residual deviance: 585.45  on 702  degrees of freedom
## AIC: 607.45
##
## Number of Fisher Scoring iterations: 13
```

We can see that pvalue for Sexmale, TitleMiss, TitleMrs are greater than 0.05 ie above threshold. Also the Std Errors are high for above features due to High corelation as we have seen during ChiSqare test.

Test variable inflation factor

```
vif(glm.fit)
```

```
##                      GVIF Df GVIF^(1/(2*Df))
## Pclass       1.667655e+00  2        1.136388
## Sex          5.751701e+06  1     2398.270329
## Age          1.894599e+00  1        1.376444
## Title        1.224494e+07  4        7.691208
## FamilyMember 1.812345e+00  2        1.160273
```

We can see that GVIF (generalized variable inflation factor) is high for Sex and title, which clearly proove the multicollinearity between sex and title.

We will drop sex from our model it has high degree of multicollinearity.

Fit the logistic regression model on training data after removing the sex feature.

```
glm.fit <- glm(Survived ~ . -Sex, family = binomial(link='logit'), data=train)
```

Using the best model by selecting AIC

Step() will run the model for each variable iteratively and give the best model on top as per ranking on AIC.

```
glm.fit <-step(glm.fit)
```

```
## Start:  AIC=614.37
## Survived ~ (Pclass + Sex + Age + SibSp + Parch + Fare + Embarked +
##     Title + FamilyMember) - Sex
##
##                Df Deviance    AIC
## - SibSp         1   584.37 612.37
## - Embarked      2   586.52 612.52
## - Fare          1   584.83 612.83
## - Parch         1   585.58 613.58
## <none>              584.37 614.37
## - Age           1   589.95 617.95
## - FamilyMember  2   594.60 620.60
## - Pclass        2   629.59 655.59
## - Title         4   772.00 794.00
##
## Step:  AIC=612.37
## Survived ~ Pclass + Age + Parch + Fare + Embarked + Title + FamilyMember
##
##                Df Deviance    AIC
## - Embarked      2   586.56 610.56
## - Fare          1   584.84 610.84
## - Parch         1   586.10 612.10
## <none>              584.37 612.37
## - Age           1   589.95 615.95
## - FamilyMember  2   613.53 637.53
## - Pclass        2   629.78 653.78
## - Title         4   772.02 792.02
##
## Step:  AIC=610.56
## Survived ~ Pclass + Age + Parch + Fare + Title + FamilyMember
##
##                Df Deviance    AIC
## - Fare          1   587.31 609.31
## - Parch         1   588.22 610.22
## <none>              586.56 610.56
## - Age           1   592.09 614.09
## - FamilyMember  2   618.78 638.78
## - Pclass        2   632.93 652.93
## - Title         4   783.98 799.98
```

```
##
## Step:  AIC=609.31
## Survived ~ Pclass + Age + Parch + Title + FamilyMember
##
##                Df Deviance    AIC
## - Parch         1   589.15 609.15
## <none>              587.31 609.31
## - Age           1   593.14 613.14
## - FamilyMember  2   618.78 636.78
## - Pclass        2   667.53 685.53
## - Title         4   785.52 799.52
##
## Step:  AIC=609.15
## Survived ~ Pclass + Age + Title + FamilyMember
##
##                Df Deviance    AIC
## <none>              589.15 609.15
## - Age           1   595.17 613.17
## - FamilyMember  2   626.64 642.64
## - Pclass        2   669.92 685.92
## - Title         4   793.80 805.80
```

Verify the Coefficients of best model and test the p-value

```
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Age + Title + FamilyMember,
##     family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6304  -0.5750  -0.3792   0.5637   2.4607
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)         1.45654    0.59736   2.438  0.01476 *
## Pclass2            -1.46793    0.32032  -4.583 4.59e-06 ***
## Pclass3            -2.58021    0.31125  -8.290  < 2e-16 ***
## Age                -0.02543    0.01059  -2.403  0.01627 *
## TitleMiss          -0.40382    0.55940  -0.722  0.47037
## TitleMr            -3.36730    0.60132  -5.600 2.15e-08 ***
## TitleMrs            0.05208    0.63190   0.082  0.93431
## Titleother         -2.56210    0.81122  -3.158  0.00159 **
## FamilyMemberSingle  2.65769    0.50374   5.276 1.32e-07 ***
## FamilyMemberSmall   2.42568    0.48461   5.005 5.57e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 949.90  on 712  degrees of freedom
## Residual deviance: 589.15  on 703  degrees of freedom
## AIC: 609.15
##
## Number of Fisher Scoring iterations: 5
```

Test the variable inflation factor for model, again to verify the collineary among feattures

```
vif(glm.fit)
```

```
##                  GVIF Df GVIF^(1/(2*Df))
## Pclass       1.688094  2        1.139854
## Age          1.909003  1        1.381667
## Title        2.618396  4        1.127858
## FamilyMember 1.805038  2        1.159102
```

durbinWatsonTest is done to check if residuals ie error are not correlated which is assumption of regression.

```
durbinWatsonTest(glm.fit)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      0.02148659      1.956557   0.576
##  Alternative hypothesis: rho != 0
```

Below are the observation from above 3 test(summary, vif and durbinwatsonTest) 1.The std Error are in reasonable range 2.The GVIF values all are less than 5 3.The D-W Statistic p-value values are 1.95 and .575 respectively. pvalue is greater than 0.05, hence we do not reject H0. the residuals are not autocorrelated.

Now according to best model Pclass,Age, Title, FamilyMember significantly contributes in model for predicting the survuval.

Predict the survival for validation test

```
survived_prob <- predict(glm.fit, type="response", newdata=test)
```

Calculate prediction

```
survived_pred <- ifelse(survived_prob > 0.5 , 1, 0)
```

Checking the prediction accuracy

```
table(test$Survived,survived_pred )
```

```
##    survived_pred
##       0   1
##   0 101   9
##   1  17  51
```

Accuracy is below

```
mean(test$Survived == survived_pred)
```

```
## [1] 0.8539326
```

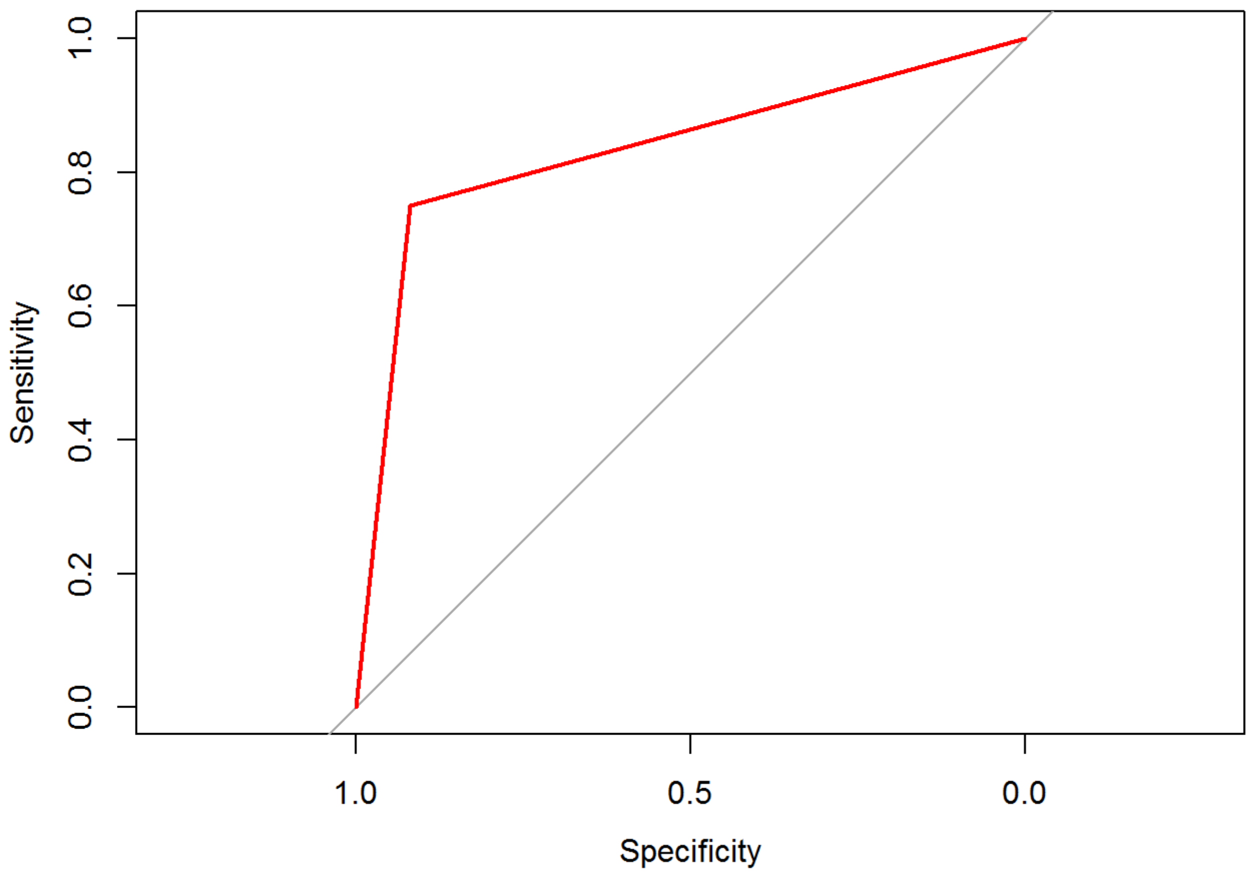
We will see the performance of model using Plot and Graph

Find the ROC

```
ROC <- roc(test$Survived, survived_pred)
```

Plot the area

```
plot(ROC, col= "red")
```



Find the total areas under curve

```
auc(ROC)
```

```
## Area under the curve: 0.8341
```

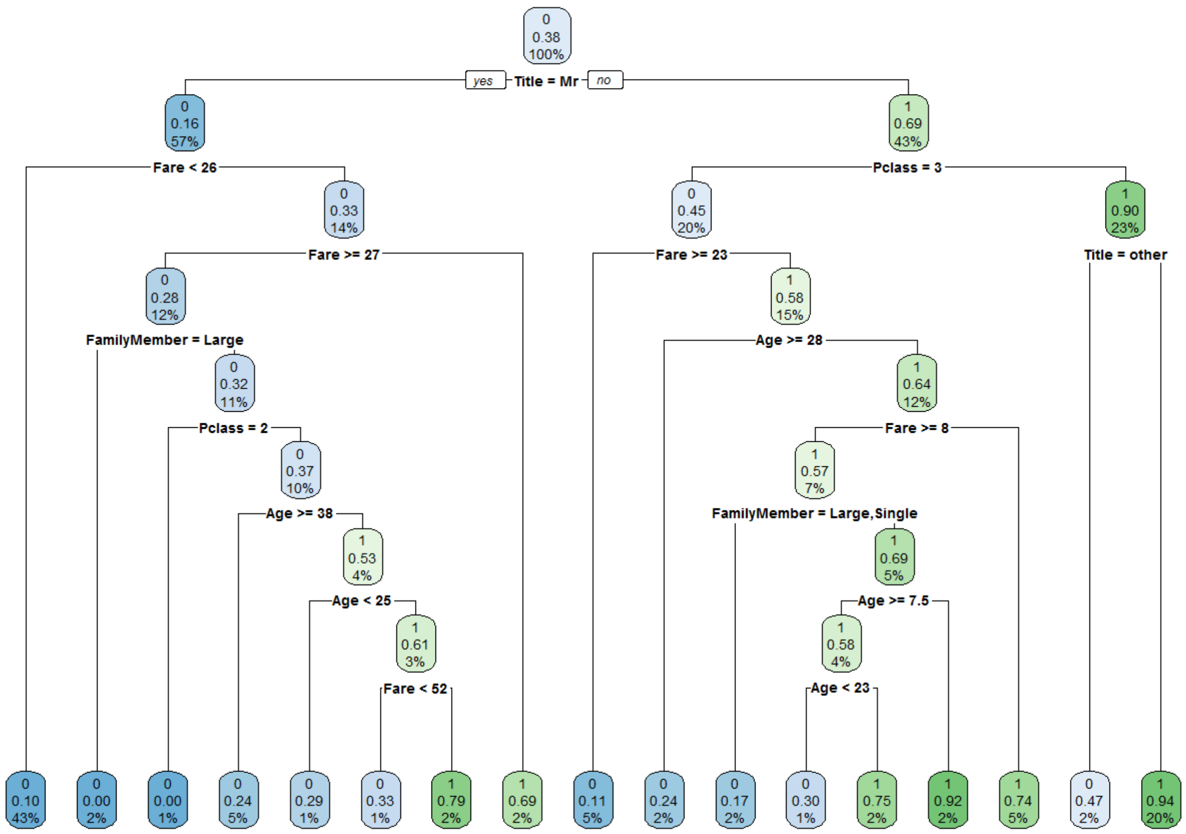
The ROC (Receiver Operating Characteristics) curve is a graphical representation of the performnace of the classifier and it shows the performance of our model rises well above the diagonal line. This indicates that our logistic regression model performs better than just a random guess. The logistic regression model delivers a 0.8342 accuracy interms of predicting the survival.

5.4 Decision Tree #Fit decision tree on train data

```
decision_tree.fit <- rpart(Survived ~ . , data=train, method = "class", control = rpart.control(cp=0))
```

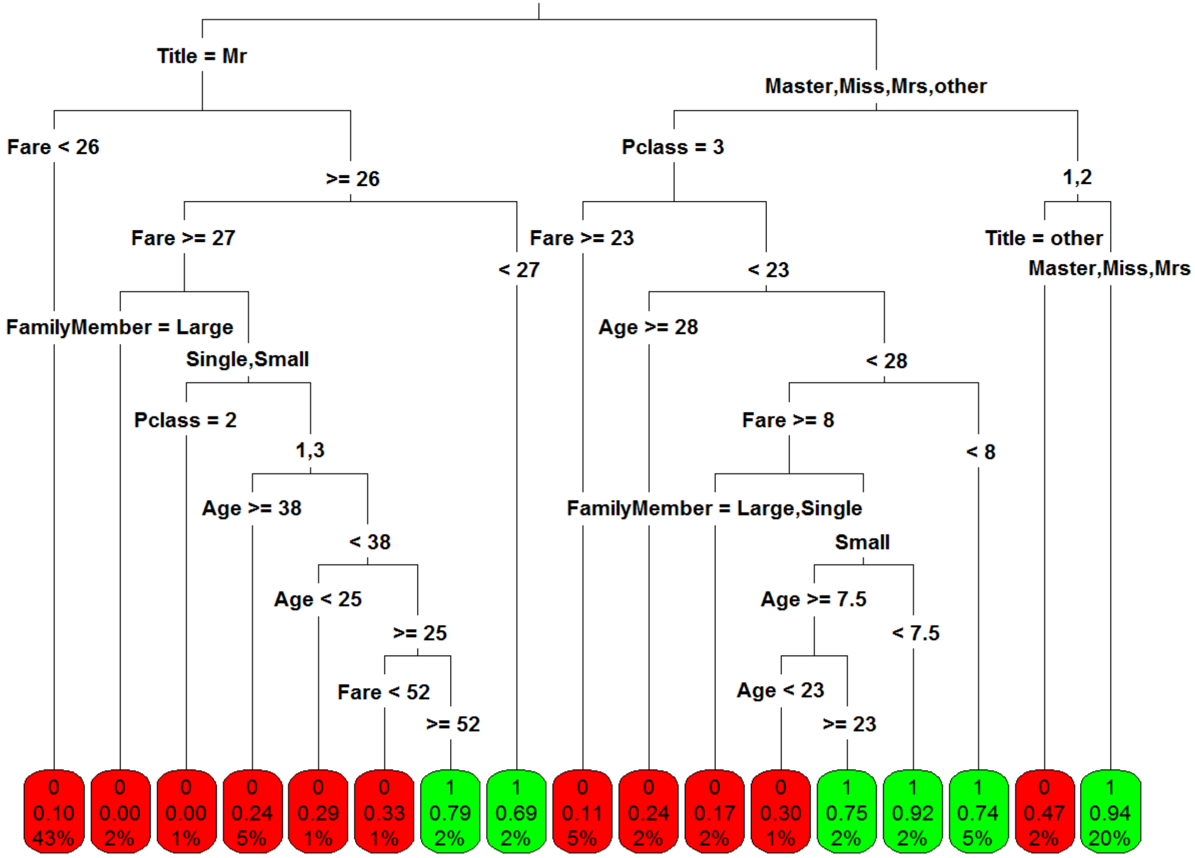
Plot tree

```
rpart.plot(decision_tree.fit)
```



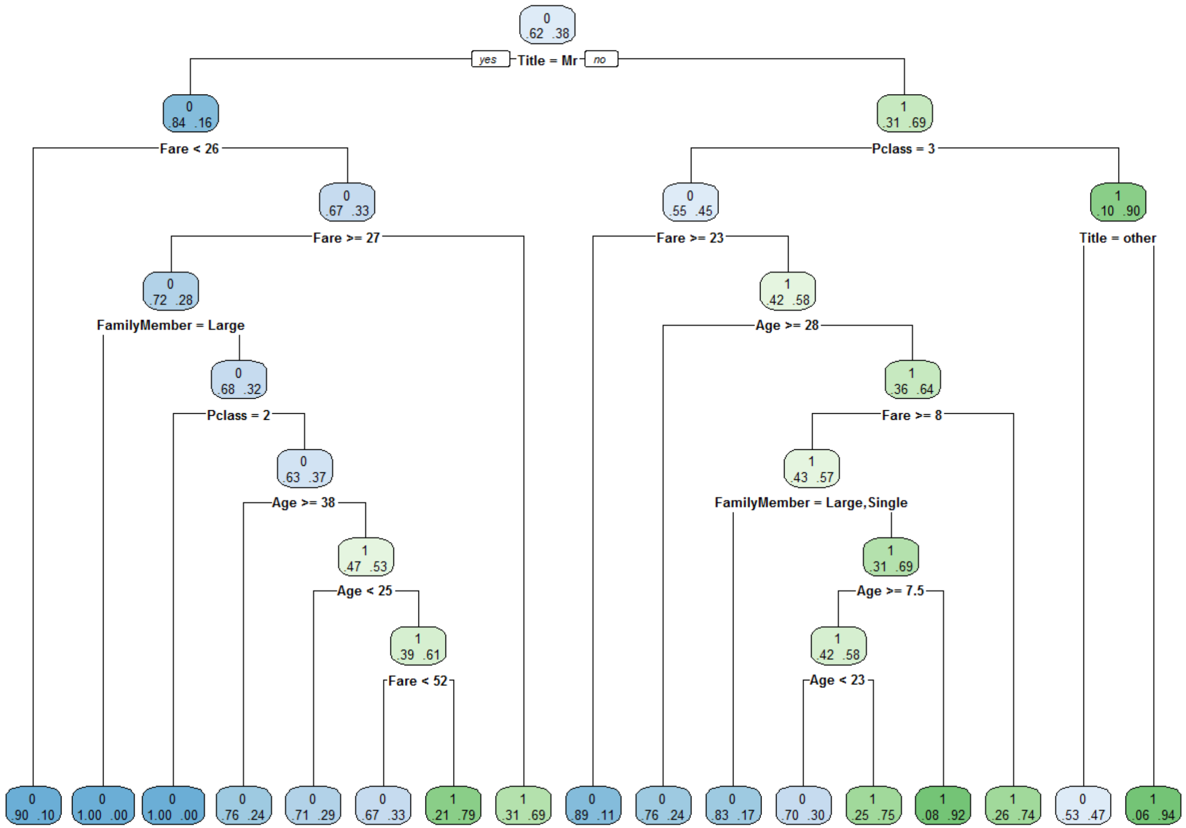
plot tree with customized setting

```
rpart.plot(decision_tree.fit,type=3, box.palette = c("red", "green"), fallen.leaves = TRUE)
```



Tree uses features like title, fare, familyMember, Pclass, age.

```
rpart.plot(decision_tree.fit, extra=4)
```



Predict the validation set on above model

use predict function for validation set

```
y_prob = predict(decision_tree.fit, newdata=test[,-which(names(test)=="Survived")], type = "class")
```

Examine the confusion matrix

```
table(test$Survived, y_prob)
```

```
##    y_prob
##       0   1
##   0 103   7
##   1  19  49
```

Compute accuracy on test data

```
mean(test$Survived == y_prob)
```

```
## [1] 0.8539326
```

The accuracy is : 0.8539326

Overfitting can easily occur in Decision Tree classification. We can idenfity that evaluating the model using k-Fold Cross Validation

Apply above K-Fold

```
set.seed(789)
folds <- createMultiFolds(train$Survived, k = 10, times =5)
train.control <- trainControl(method = "repeatedcv", index = folds)
dt_cv <- train(Survived ~ . , data = train, method = "rpart", trControl = train.control)
```

Plot

```
print(dt_cv$finalModel)
```

```
## n= 713
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 713 274 0 (0.6157083 0.3842917)
##    2) TitleMr>=0.5 407  64 0 (0.8427518 0.1572482) *
##    3) TitleMr< 0.5 306  96 1 (0.3137255 0.6862745)
##      6) Pclass3>=0.5 144  65 0 (0.5486111 0.4513889)
##       12) Fare>=23.35 38   4 0 (0.8947368 0.1052632) *
##       13) Fare< 23.35 106  45 1 (0.4245283 0.5754717) *
##      7) Pclass3< 0.5 162  17 1 (0.1049383 0.8950617) *
```

```
names(dt_cv)
```

```
##  [1] "method"       "modelInfo"    "modelType"    "results"
##  [5] "pred"         "bestTune"     "call"         "dots"
##  [9] "metric"       "control"      "finalModel"   "preProcess"
## [13] "trainingData" "resample"     "resampledCM"  "perfNames"
## [17] "maximize"     "yLimits"      "times"        "levels"
## [21] "terms"        "coefnames"    "contrasts"    "xlevels"
```

```
#rpart.plot(dt_cv$finalModel)
```

Predict on test data

```
#y_pred_cv = predict(dt_cv, newdata=test[,-which(names(test)=="Survived")])
y_pred_cv = predict(dt_cv, newdata=test)
```

```
head(test)
```

```
##    Survived Pclass    Sex Age SibSp Parch    Fare Embarked Title
## 12        1      1 female  58     0     0 26.5500        S  Miss
## 16        1      2 female  55     0     0 16.0000        S   Mrs
## 20        1      3 female  25     0     0  7.2250        C   Mrs
## 22        1      2   male  34     0     0 13.0000        S    Mr
## 33        1      3 female  25     0     0  7.7500        Q  Miss
## 40        1      3 female  14     1     0 11.2417        C  Miss
##    FamilyMember
## 12       Single
## 16       Single
## 20       Single
## 22       Single
## 33       Single
## 40        Small
```

find confusion matrix

```
table(test$Survived, y_pred_cv)
```

```
##    y_pred_cv
##      0  1
##   0 99 11
##   1 17 51
```

Find accuracy

```
mean(test$Survived == y_pred_cv)
```

```
## [1] 0.8426966
```

We were not much able to improve the model after 10-fold cross validation. The accuracy is almost same to 0.8427 but note the improved model uses only three features Title, Pclass and Fare for classification.

5.5 Random Forest

This is similar to decision tree, in random forest many trees are created called as forest instead of one tree like decision tree algorithm.In decision tree, tree root is created based on variable having gini index and information gain. In random forest trees are created on each variables and letter based on voting or ensemble method, model predict the outcomes.

building a simple random forest

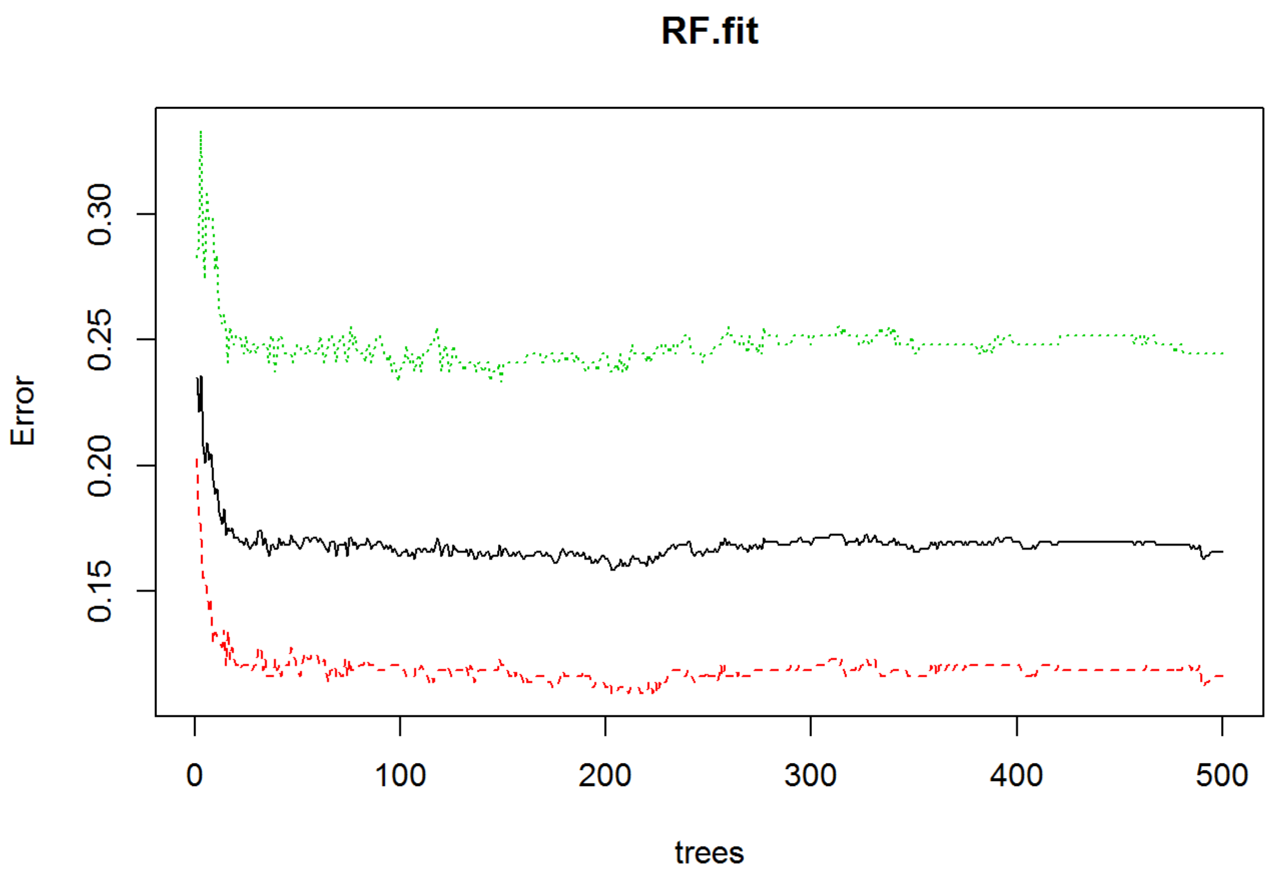
set.seed(432)

```
RF.fit = randomForest(Survived ~ ., data = train)
nrow(RF.fit)
```

```
## NULL
```

Plot the random forest

```
plot(RF.fit)
```



The green, black and red lines represent error rate for death, overall and survival, respectively. The overall error rate converges to around 17%. Interestingly, our model predicts death better than survival. Since the overall error rate converges to a constant and does not seem to further decrease, our choice of default 500 trees in the randomForest function is a good choice

Predicting on test set results

```
nrow(test)
```

```
## [1] 178
```

```
y_pred <-predict(RF.fit, newdata = test[,-which(names(test)=="Survived")])
```

Create confusion matrix

```
table(test$Survived,y_pred)
```

```
##    y_pred
##      0  1
##   0 99 11
##   1 18 50
```

Find accuracy of model

```
mean(test$Survived == y_pred)
```

```
## [1] 0.8370787
```

The accuracy of model is : 83%

The accuracy of model is less than decision tree model, we will run now K-fold cross validation to check if that improves the accuracy.

Apply cross validation

```
set.seed(651)
folds <- createMultiFolds(train$Survived, k=10)

control <- trainControl(method = "repeatedcv", index=folds)

RF.fit_cv <- train(Survived ~ . , data=train, method = "rf",trControl=control)
```

Predict the test data

```
y_pred_cv <- predict(RF.fit_cv, newdata=test)
```

Find accuracy and confusion matrix

```
table(test$Survived, y_pred_cv)
```

```
##    y_pred_cv
##      0  1
##   0 95 15
##   1 19 49
```

```
mean(test$Survived == y_pred_cv)
```

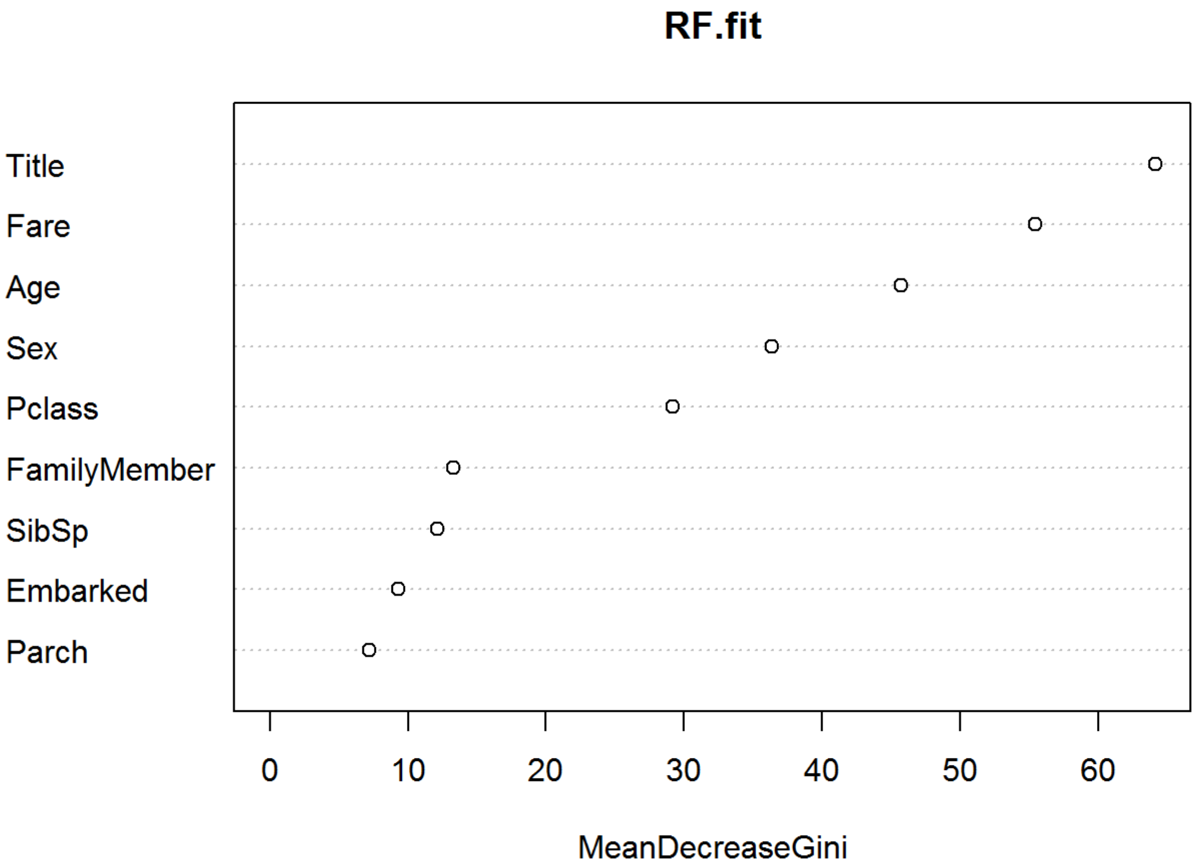
```
## [1] 0.8089888
```

We can see that k-Fold validation has not improved the prediction.

The interpretation of random forest is not easy, let us find the importance variable of entire forest modeled above.

Plot the important features used in random forest

```
varImpPlot(RF.fit)
```



We can check the important features as per gini index

```
varImp(RF.fit)
```

```
##                Overall
## Pclass       29.186090
## Sex          36.363553
## Age          45.701275
## SibSp        12.173765
## Parch         7.189153
## Fare         55.475892
## Embarked      9.345116
## Title        64.122275
## FamilyMember 13.313642
```

The feature Title has the highest mean gini index, hence the highest importance. Fare is also realtively high important and it is followed by Age of the passenger

5.6 Naive bayes

Naive Bayes is based on the assumption that conditional probability of each feature given the class is independent of all the other features. The assumption of independent conditional probabilities means the features are completely independent of each other. This assumption was already checked in the Logistic Regression section and we have found that numeric features are independent to each other, however, the categorical features are not. By assuming the idependence assumption of all the features, let’s fit a naive bayes model to our training data.

Create model

```
NB.fit <- naiveBayes(Survived ~ ., data = train)
```

Predict using model

```
NB.y_pred= predict(NB.fit, newdata=test[,-which(names(test)=="Survived")])
```

Find accuracy and confusion matrix

```
table(test$Survived, NB.y_pred)
```

```
##    NB.y_pred
##      0  1
##   0 99 11
##   1 17 51
```

Find accuracy

```
mean(test$Survived ==NB.y_pred)
```

```
## [1] 0.8426966
```

The accuracy of Naive bayes model is : 84 percent Naive bayes perform well, its accuracy is quite good and comparable to other algorithms.

5.7 Discussion

Comparision of Models

Logistic model

-The accuracy of model is 0.8341 -Pclass, Age, Title, FamilyMember are the features contributed in model. -Confusion matix has 9 false positives and 17 false negatives.

Decision Tree

-The accuracy of model is 0.8427 -Pclass, Fare, Title are the features contributed in model. -Confusion matix has 11 false positives and 17 false negatives.

Random Forest

-The accuracy of model is 0.8371 and K-fold cross validation did not improved the accuracy. -Title, Fare, Age are the features contributed in model in same order respectively. -Confusion matix has 18 false positives and 15 false negatives.

Naive Bayes

-The accuracy of model is 0.8341 -The features where independent -Confusion matix has 11 false positives and 17 false negatives.

6.Final Prediction on Model

```
glm.fit.predict <- predict(glm.fit, newdata = train_test, type="response")
```

```
decision_tree.fit.predict <- predict(decision_tree.fit, newdata=train_test)
```

```
RF.fit.predict <- predict(RF.fit, newdata=train_test, type="class")
```

```
NB.fit.predict <- predict(NB.fit, newdata=train_test, type="class")
```

Store the results

```
glm.fit.predict_results <- data.frame(PassengerID = titanic_mainData[892:1309,"PassengerId"], Survived = glm.fit.predict)

decision_tree.fit.predict_results <- data.frame(PassengerID = titanic_mainData[892:1309,"PassengerId"], Survived = decision_
tree.fit.predict)

RF.fit.predict_results <- data.frame(PassengerID = titanic_mainData[892:1309,"PassengerId"], Survived = RF.fit.predict)

NB.fit.predict_results <- data.frame(PassengerID = titanic_mainData[892:1309,"PassengerId"], Survived = NB.fit.predict)
```

Save the results in Excel sheet

```
write.csv(glm.fit.predict_results, file = 'PredictingTitanicSurvival_glm.csv', row.names = FALSE, quote=FALSE)
write.csv(decision_tree.fit.predict_results, file = 'PredictingTitanicSurvival_dt.csv', row.names = FALSE, quote=FALSE)
write.csv(RF.fit.predict_results, file = 'PredictingTitanicSurvival_rf.csv', row.names = FALSE, quote=FALSE)
write.csv(NB.fit.predict_results, file = 'PredictingTitanicSurvival_nb.csv', row.names = FALSE, quote=FALSE)
```