Titanic-Prediction

Suprasanna Pradhan

10 December 2018

Introduction

This analysis attempts to predicate the probability for survival of the Titanic passengers. In order to do this, I will use the different features available about the passengers, use a subset of the data to train an algorithm and then run the algorithm on the rest of the data set to get a prediction.

We will try out the following algorithms:

Logistic Regression Decision tree - CART Random Forest Naive Bayes and mlr Bagging XGBoost K-Fold Cross Validation

Load packages

Here we will load all the necessary packages that we will use during our analyses.

```
#Loading required packages
#install.packages("tidyverse")
#library(tidyverse)
library(ggplot2)
library(caret)
```

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

```
library(caretEnsemble)
```

```
##
## Attaching package: 'caretEnsemble'
```

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

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(Amelia)
```

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

```
## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.7.5, built: 2018-05-07)
## ## Copyright (C) 2005-2020 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

```
library(mice)
```

```
##
## Attaching package: 'mice'
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(gutenbergr)
library(tidytext)
library(dplyr)
```

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

```
## The following object is masked from 'package:GGally':
##
##     nasa
```

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

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

```
library(janeaustenr)
library(stringi)
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:mice':
##
##     complete
```

```
library(rpart)
library(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:psych':
##
##     outlier
```

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

Importing the data

```
setwd ("C:/Users/SuprasannaPradhan/Documents/My Files/R/R project files/Titanic")
getwd()
```

```
## [1] "C:/Users/SuprasannaPradhan/Documents/My Files/R/R project files/Titanic"
```

```
t.test<- read.table("test.csv", sep = ",", header = T)
t.train <- read.table("train.csv", sep = ",", header = T)
str(t.train)
```

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 358 277 16 5
59 520 629 417 581 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 681 levels "110152","110413",..: 524 597 670 50 473 276 8
6 396 345 133 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 1 57 1 1 131 1 1 1 ...
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
```

The train data set is consist of 891 observation with 12 variables

```
str(t.test)
```

```
## 'data.frame':    418 obs. of  11 variables:
##  $ PassengerId: int  892 893 894 895 896 897 898 899 900 901 ...
##  $ Pclass     : int  3 3 2 3 3 3 3 2 3 3 ...
##  $ Name       : Factor w/ 418 levels "Abbott, Master. Eugene Joseph",..: 210 409 27
3 414 182 370 85 58 5 104 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 2 2 1 2 1 2 1 2 ...
##  $ Age        : num  34.5 47 62 27 22 14 30 26 18 21 ...
##  $ SibSp      : int  0 1 0 0 1 0 0 1 0 2 ...
##  $ Parch      : int  0 0 0 0 1 0 0 1 0 0 ...
##  $ Ticket     : Factor w/ 363 levels "110469","110489",..: 153 222 74 148 139 262 1
59 85 101 270 ...
##  $ Fare       : num  7.83 7 9.69 8.66 12.29 ...
##  $ Cabin      : Factor w/ 77 levels "","A11","A18",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Embarked   : Factor w/ 3 levels "C","Q","S": 2 3 2 3 3 3 2 3 1 3 ...
```

Test data set having 418 observation and 11 variables

Since the datasets are given seperately as trained and tested data, they will be kept as it is. The thing that needed to be done is to merge the actual survival outcome of passengers from tested data with other information in that dataset. The column of survival outcome (dependent variable) is merged with the rest of the independent variables/features of the passegers from the tested dataset by passengerId. The trained dataset contains 891 observations (passenger information) and 12 features (information of passengers), and the tested dataset contains 418 observations

```
summary(t.train)
```

```
##   PassengerId       Survived          Pclass     
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000  
##  1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000  
##  Median :446.0   Median :0.0000   Median :3.000  
##  Mean   :446.0   Mean   :0.3838   Mean   :2.309  
##  3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000  
##  Max.   :891.0   Max.   :1.0000   Max.   :3.000  
##                                                  
##                                     Name         Sex           Age       
##  Abbing, Mr. Anthony                  :  1   female:314   Min.   : 0.42  
##  Abbott, Mr. Rossmore Edward          :  1   male  :577   1st Qu.:20.12  
##  Abbott, Mrs. Stanton (Rosa Hunt)     :  1                Median :28.00  
##  Abelson, Mr. Samuel                  :  1                Mean   :29.70  
##  Abelson, Mrs. Samuel (Hannah Wizosky):  1                3rd Qu.:38.00  
##  Adahl, Mr. Mauritz Nils Martin       :  1                Max.   :80.00  
##  (Other)                              :885                NA's   :177    
##      SibSp           Parch             Ticket         Fare       
##  Min.   :0.000   Min.   :0.0000   1601    :  7   Min.   :  0.00  
##  1st Qu.:0.000   1st Qu.:0.0000   347082  :  7   1st Qu.:  7.91  
##  Median :0.000   Median :0.0000   CA. 2343:  7   Median : 14.45  
##  Mean   :0.523   Mean   :0.3816   3101295 :  6   Mean   : 32.20  
##  3rd Qu.:1.000   3rd Qu.:0.0000   347088  :  6   3rd Qu.: 31.00  
##  Max.   :8.000   Max.   :6.0000   CA 2144 :  6   Max.   :512.33  
##                                   (Other) :852                   
##          Cabin     Embarked
##             :687    :  2   
##  B96 B98    :  4   C:168   
##  C23 C25 C27:  4   Q: 77   
##  G6         :  4   S:644   
##  C22 C26    :  3           
##  D          :  3           
##  (Other)    :186           
```

```
describe(t.train)
```

```
##             vars   n   mean     sd median trimmed    mad  min    max
## PassengerId    1 891 446.00 257.35 446.00  446.00 330.62 1.00 891.00
## Survived       2 891   0.38   0.49   0.00    0.35   0.00 0.00   1.00
## Pclass         3 891   2.31   0.84   3.00    2.39   0.00 1.00   3.00
## Name*          4 891 446.00 257.35 446.00  446.00 330.62 1.00 891.00
## Sex*           5 891   1.65   0.48   2.00    1.68   0.00 1.00   2.00
## Age            6 714  29.70  14.53  28.00   29.27  13.34 0.42  80.00
## SibSp          7 891   0.52   1.10   0.00    0.27   0.00 0.00   8.00
## Parch          8 891   0.38   0.81   0.00    0.18   0.00 0.00   6.00
## Ticket*        9 891 339.52 200.83 338.00  339.65 268.35 1.00 681.00
## Fare          10 891  32.20  49.69  14.45   21.38  10.24 0.00 512.33
## Cabin*        11 891  18.63  38.14   1.00    8.29   0.00 1.00 148.00
## Embarked*     12 891   3.53   0.80   4.00    3.66   0.00 1.00   4.00
##              range  skew kurtosis   se
## PassengerId 890.00  0.00    -1.20 8.62
## Survived      1.00  0.48    -1.77 0.02
## Pclass        2.00 -0.63    -1.28 0.03
## Name*       890.00  0.00    -1.20 8.62
## Sex*          1.00 -0.62    -1.62 0.02
## Age          79.58  0.39     0.16 0.54
## SibSp         8.00  3.68    17.73 0.04
## Parch         6.00  2.74     9.69 0.03
## Ticket*     680.00  0.00    -1.28 6.73
## Fare        512.33  4.77    33.12 1.66
## Cabin*      147.00  2.09     3.07 1.28
## Embarked*     3.00 -1.27    -0.16 0.03
```

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

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

```
prop.table(table(t.train$Survived))*100
```

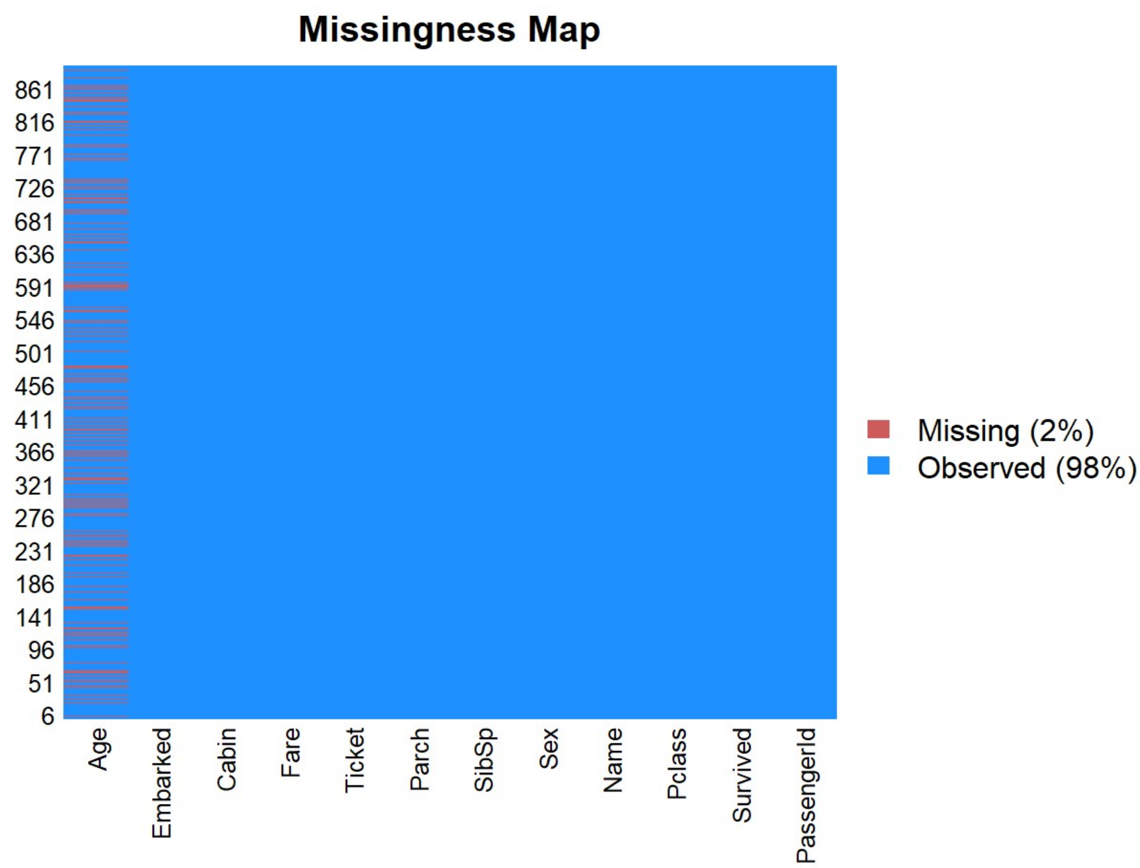
```
## 
##        0        1 
## 61.61616 38.38384
```

We got 61% is zero and 38% for 1

```
##Checking Data ##
names(t.train)
```

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

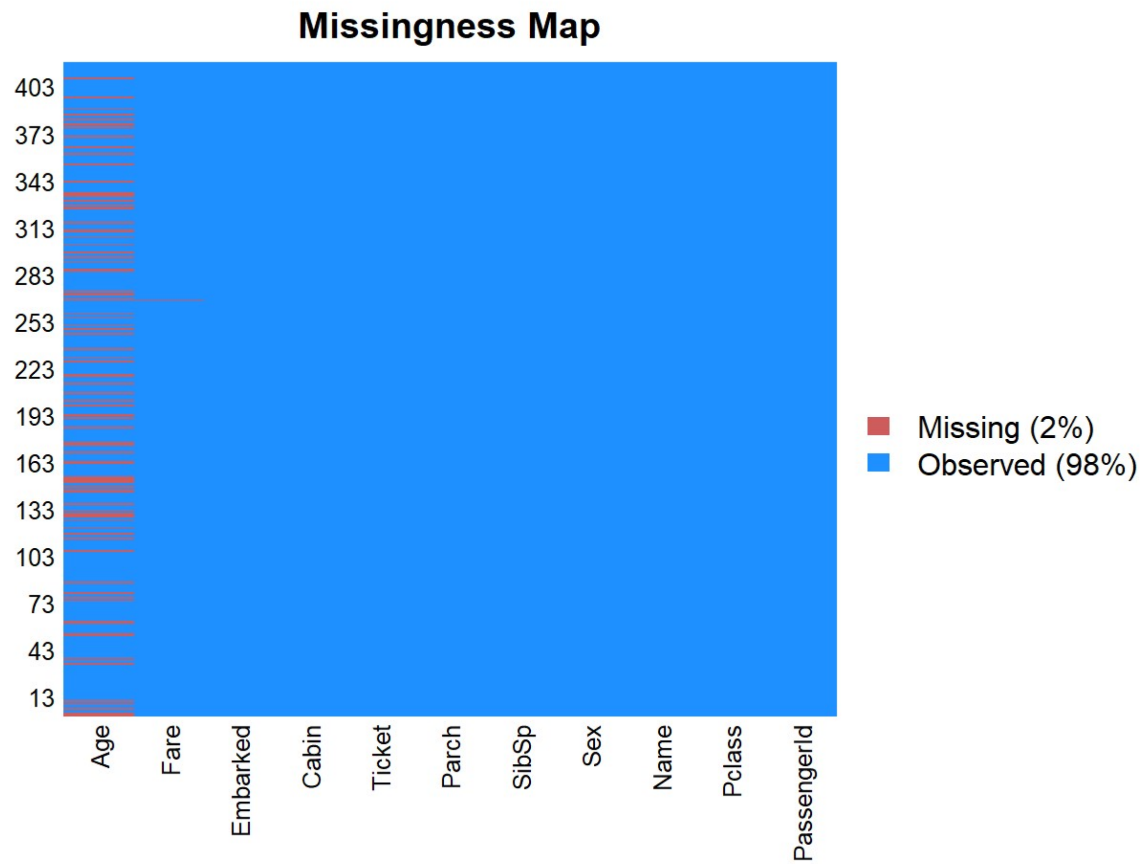
```
#visualize the missing data
missmap(t.train)
```




```
sum(is.na(t.train))
```

```
## [1] 177
```

```
missmap(t.test)
```




```
sum(is.na(t.test))
```

```
## [1] 87
```

```
colSums(is.na(t.train))
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0           0           0           0           0         177
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           0           0           0
```

We got 177 missing values in age only

```
t.train[is.na(t.train)]<- 0
sum(is.na(t.train))
```

```
## [1] 0
```

```
t.test[is.na(t.test)]<- 0
sum(is.na(t.test))
```

```
## [1] 0
```

There are some missing value in Age,these missing values may have any impact on passenger class and gender, we have to check it.

```
library(dplyr)
t.train$Survived = as.factor(t.train$Survived)
```

We have created some more variables for EDA like Agegroup and family size

```
titanic.train1<-t.train%>% mutate(AgeGroup=as.factor(findInterval(Age,c(0,18,35,10
0))))
titanic.train1$Gender<-ifelse(titanic.train1$Sex=="male",1,0)
titanic.train1$NF <- titanic.train1$SibSp+titanic.train1$Parch+1
t.test$Gender<-ifelse(t.test$Sex=="male",1,0)
t.test$NF <-t.test$SibSp+t.test$Parch+1
```

Adding title is addtional variable

```
titanic.train1$Title <- gsub('(.*, )|(\\..*)', '', titanic.train1$Name)
titanic.train1$Title[titanic.train1$Title == 'Mlle']<- 'Miss'
titanic.train1$Title[titanic.train1$Title == 'Ms']<- 'Miss'
titanic.train1$Title[titanic.train1$Title == 'Mme']<- 'Mrs'
titanic.train1$Title[titanic.train1$Title == 'Lady']<- 'Miss'
titanic.train1$Title[titanic.train1$Title == 'Dona']<- 'Miss'
officer<- c('Capt','Col','Don','Dr','Jonkheer','Major','Rev','Sir','the Countess')
titanic.train1$Title[titanic.train1$Title %in% officer]<-'Officer'
titanic.train1$Title<- as.factor(titanic.train1$Title)
```

Summary

```
summary(titanic.train1[-4])
```

```
##   PassengerId    Survived     Pclass          Sex           Age
##  Min.   :  1.0   0:549    Min.   :1.000   female:314   Min.   : 0.0
##  1st Qu.:223.5   1:342    1st Qu.:2.000   male  :577   1st Qu.: 6.0
##  Median :446.0            Median :3.000                Median :24.0
##  Mean   :446.0            Mean   :2.309                Mean   :23.8
##  3rd Qu.:668.5            3rd Qu.:3.000                3rd Qu.:35.0
##  Max.   :891.0            Max.   :3.000                Max.   :80.0
##
##      SibSp           Parch            Ticket         Fare
##  Min.   :0.000   Min.   :0.0000   1601    :  7   Min.   :  0.00
##  1st Qu.:0.000   1st Qu.:0.0000   347082  :  7   1st Qu.:  7.91
##  Median :0.000   Median :0.0000   CA. 2343:  7   Median : 14.45
##  Mean   :0.523   Mean   :0.3816   3101295 :  6   Mean   : 32.20
##  3rd Qu.:1.000   3rd Qu.:0.0000   347088  :  6   3rd Qu.: 31.00
##  Max.   :8.000   Max.   :6.0000   CA 2144 :  6   Max.   :512.33
##                                   (Other) :852
##          Cabin     Embarked AgeGroup     Gender             NF
##             :687    :  2    1:290    Min.   :0.0000   Min.   : 1.000
##  B96 B98    :  4   C:168    2:366    1st Qu.:0.0000   1st Qu.: 1.000
##  C23 C25 C27:  4   Q: 77    3:235    Median :1.0000   Median : 1.000
##  G6         :  4   S:644             Mean   :0.6476   Mean   : 1.905
##  C22 C26    :  3                     3rd Qu.:1.0000   3rd Qu.: 2.000
##  D          :  3                     Max.   :1.0000   Max.   :11.000
##  (Other)    :186
##      Title
##  Master : 40
##  Miss   :186
##  Mr     :517
##  Mrs    :126
##  Officer: 22
##
##
```

```
str(titanic.train1)
```

```
## 'data.frame':    891 obs. of  16 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     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 358 277 16 5
59 520 629 417 581 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 0 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 681 levels "110152","110413",..: 524 597 670 50 473 276 8
6 396 345 133 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 1 57 1 1 131 1 1 1 ...
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
##  $ AgeGroup   : Factor w/ 3 levels "1","2","3": 2 3 2 3 3 1 3 1 2 1 ...
##  $ Gender     : num  1 0 0 0 1 1 1 1 0 0 ...
##  $ NF         : num  2 2 1 2 1 1 1 5 3 2 ...
##  $ Title      : Factor w/ 5 levels "Master","Miss",..: 3 4 2 4 3 3 3 1 4 4 ...
```

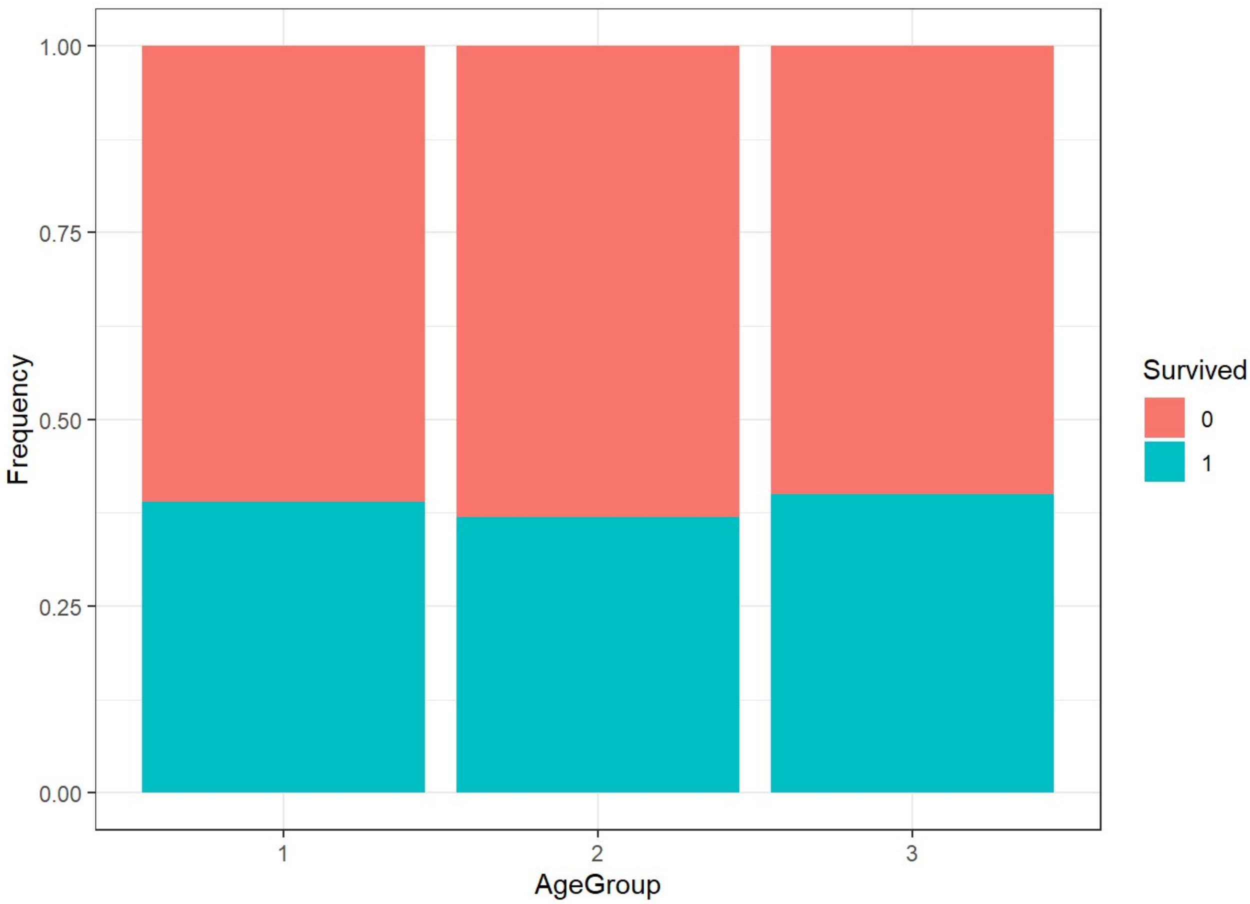
Exploratory Data Analysis

Age group wise passengers

```
table(titanic.train1$AgeGroup)
```

```
## 
##   1   2   3 
## 290 366 235
```

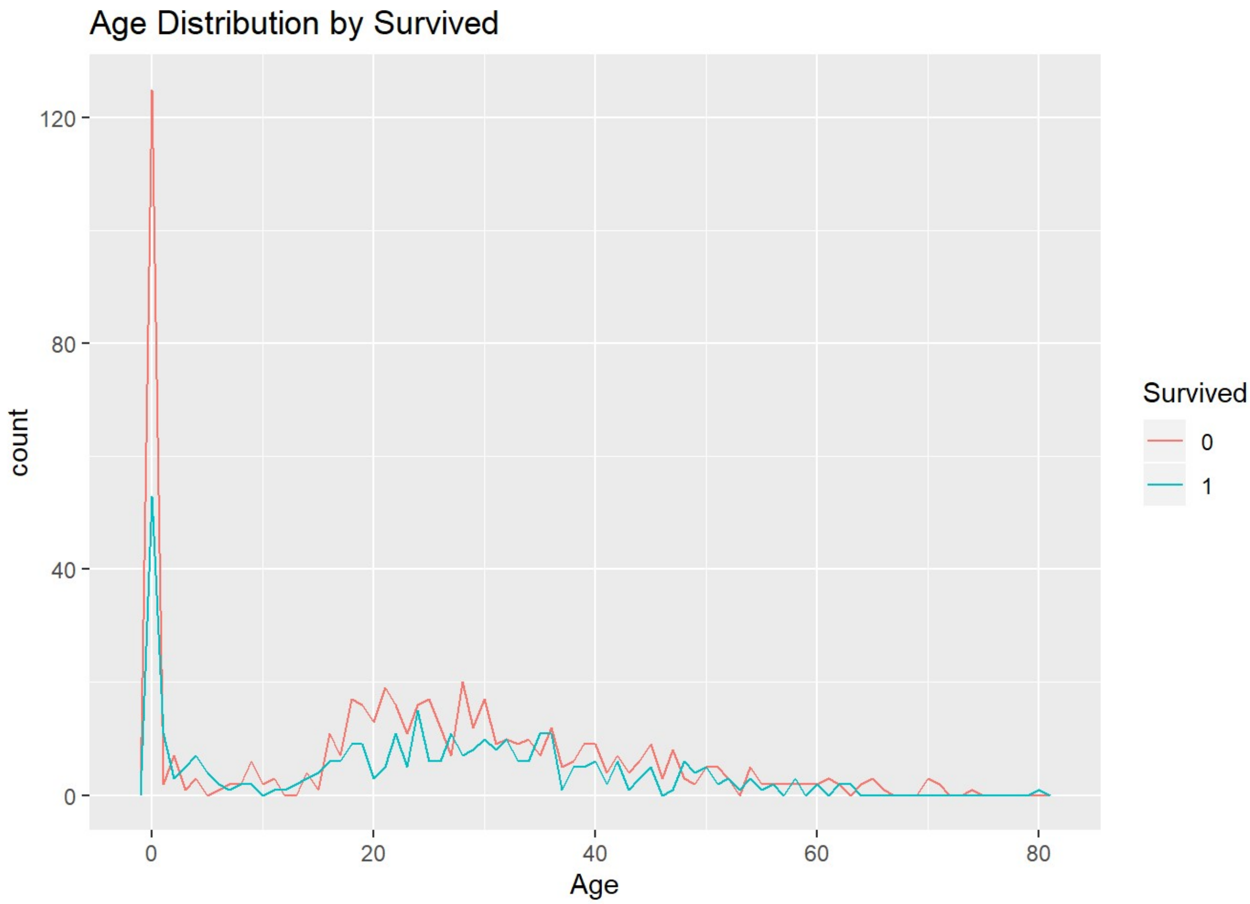
```
ag <-ggplot(data = titanic.train1,aes(x=AgeGroup,fill=Survived))+geom_bar(position="fi
ll")+ylab("Frequency")
ag + theme_bw()
```



We found maximum are Passenger belongs to second category of age

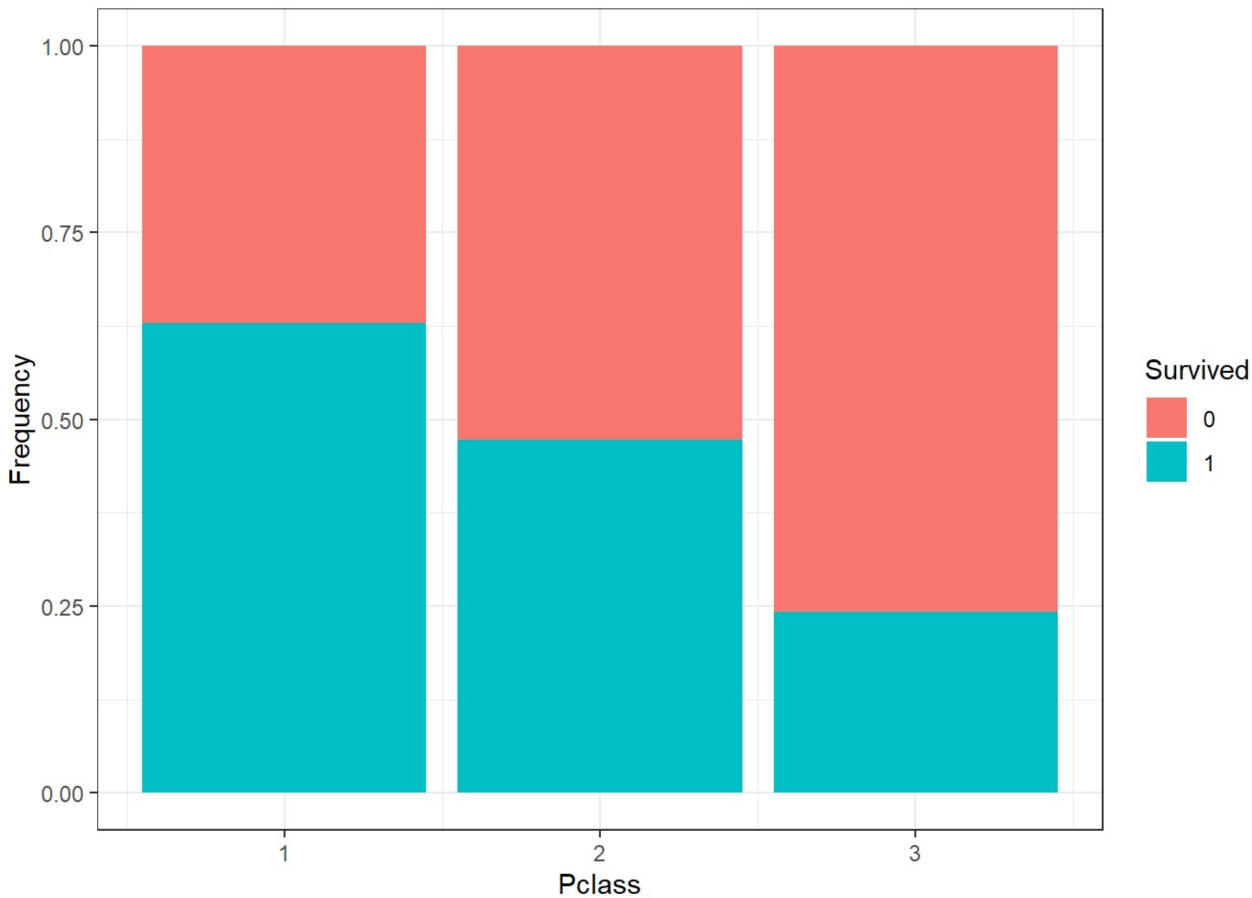
Survived by Age

```
library(ggplot2)
library(dplyr)
ggplot(titanic.train1,aes(x=Age, colour=Survived)) +
  geom_freqpoly(binwidth =1)+ labs(title="Age Distribution by Survived")
```



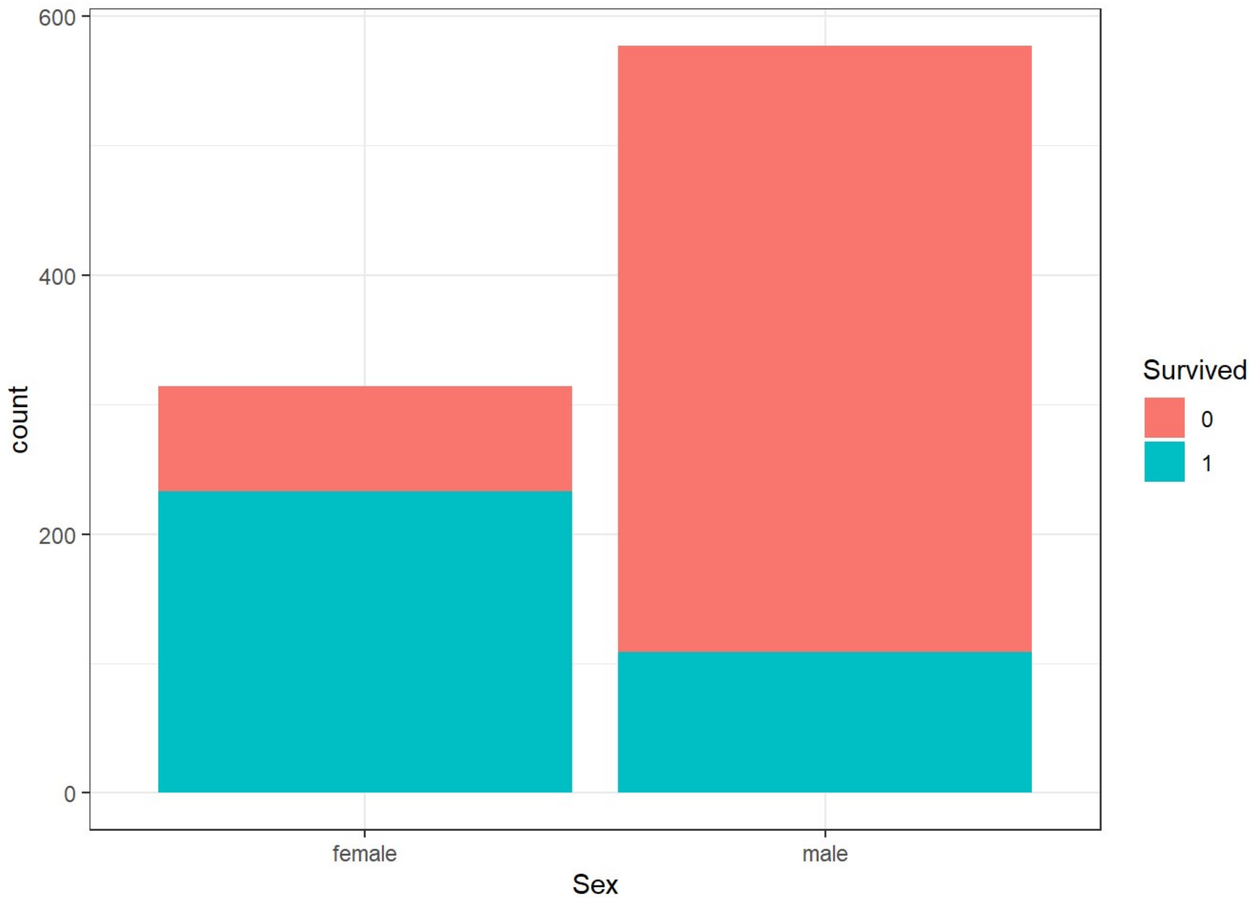
Passengers class wise survived

```
c <-ggplot(data = titanic.train1,aes(x=Pclass,fill=Survived))+geom_bar(position="fil
l")+ylab("Frequency")
c + theme_bw()
```



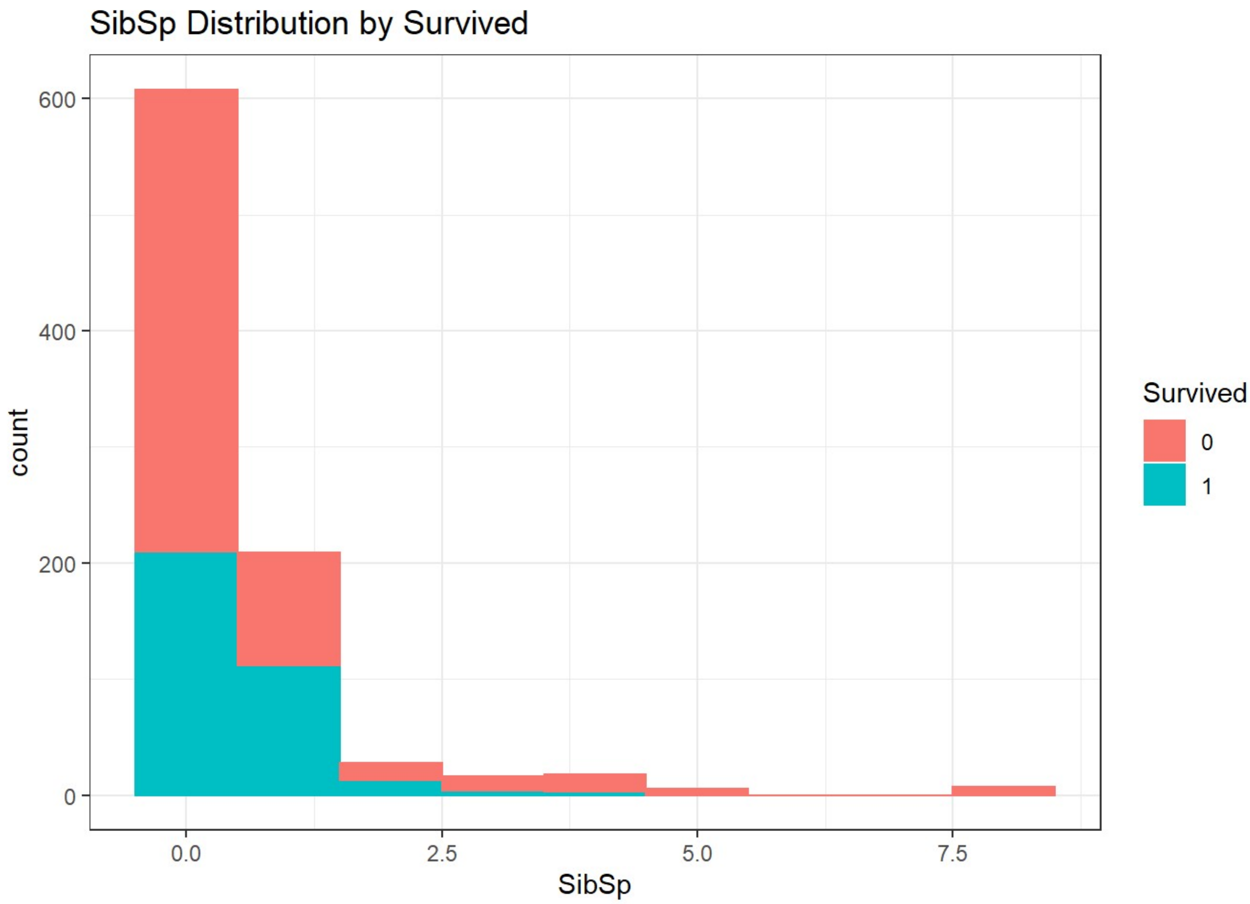
Survived by Gender

```
s <-ggplot(data=titanic.train1,aes(x=Sex,fill=Survived))+geom_bar()
s + theme_bw()
```



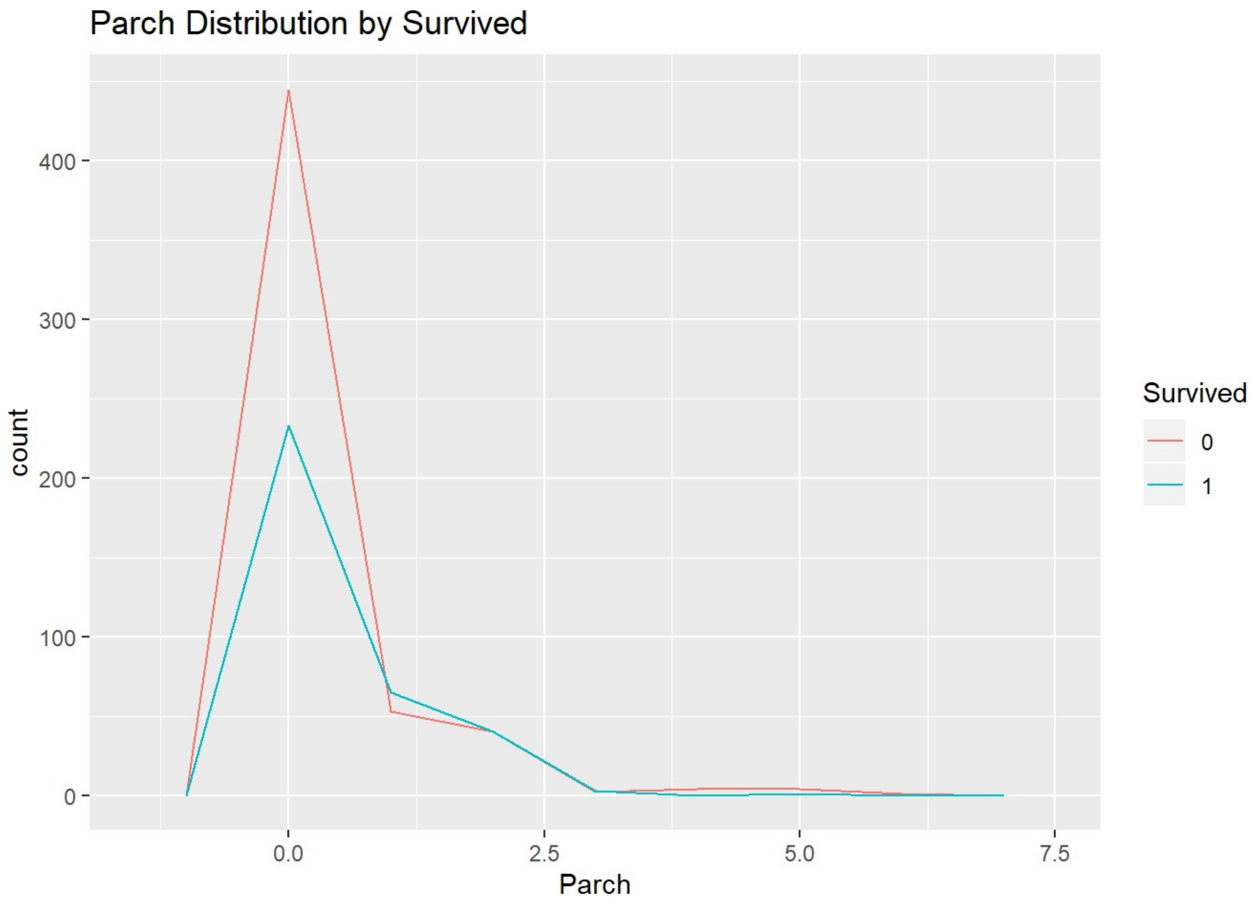
##Survived by SibSP

```
d <- ggplot(titanic.train1, aes(x=SibSp, fill=Survived, color=Survived)) +
  geom_histogram(binwidth = 1) + labs(title="SibSp Distribution by Survived")
d + theme_bw()
```



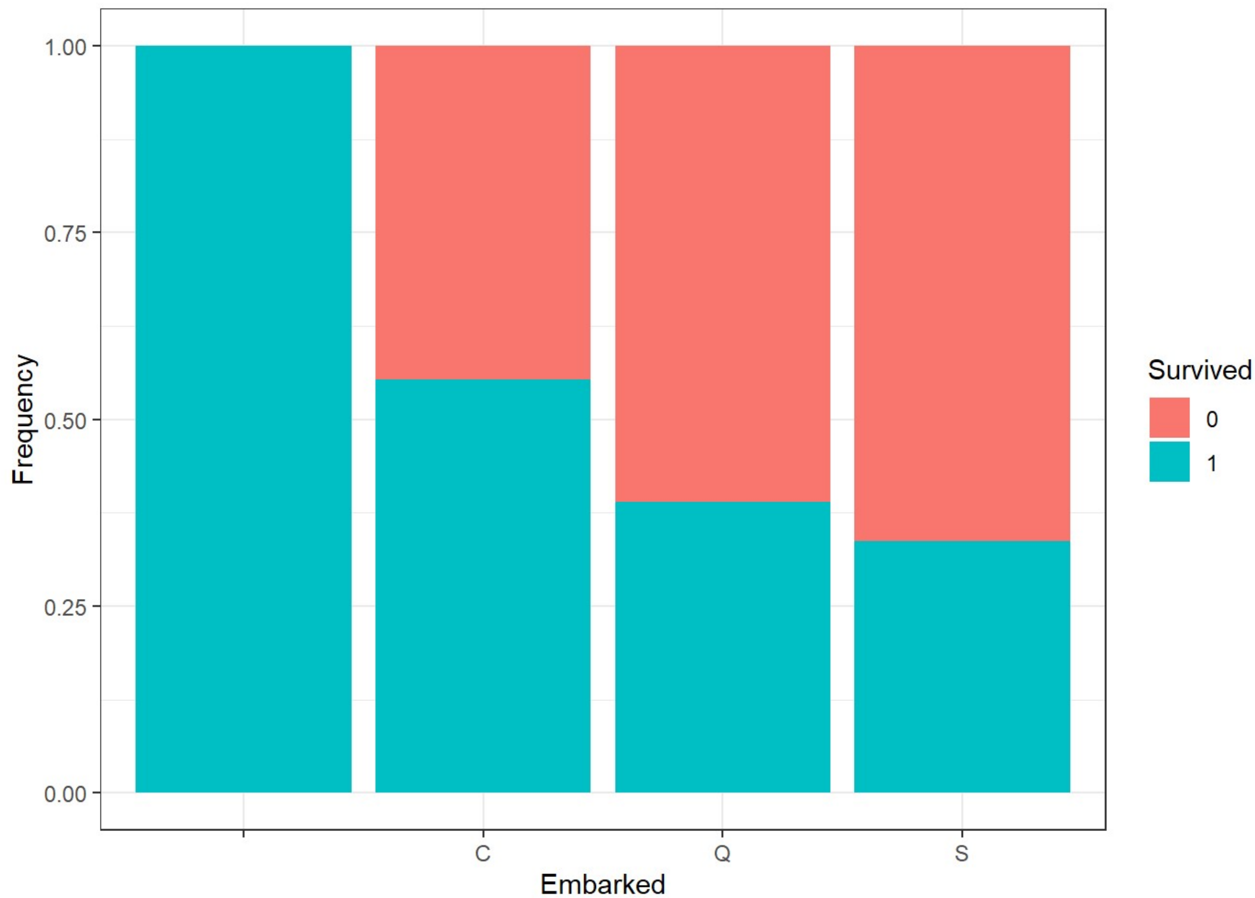
Survived by Parch

```
ggplot(titanic.train1, aes(Parch, colour = Survived)) +
  geom_freqpoly(binwidth = 1) + labs(title="Parch Distribution by Survived")
```



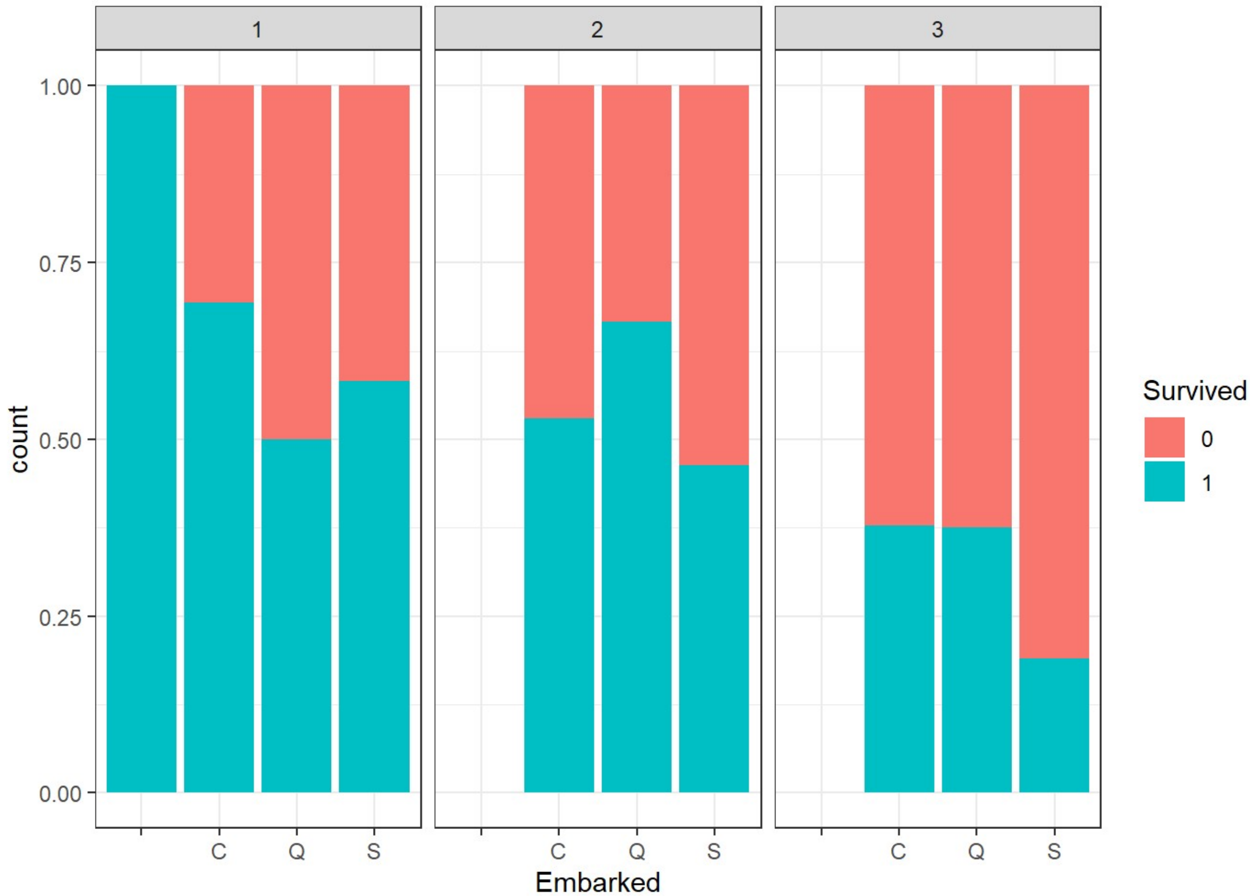
##Survivals by Embarked:

```
em <-ggplot(data = titanic.train1,aes(x=Embarked,fill=Survived))+geom_bar(position="fi
ll")+ylab("Frequency")
em + theme_bw()
```



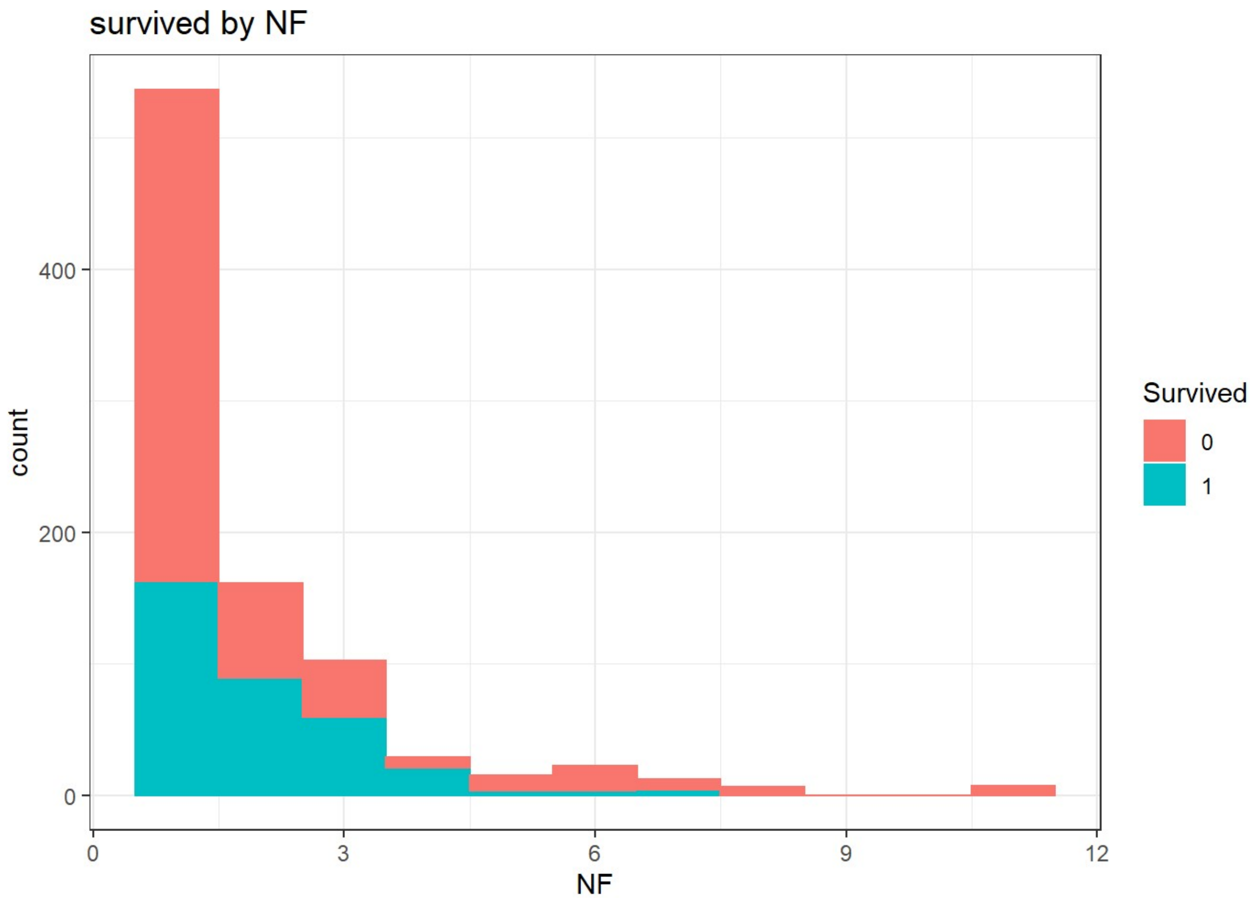
Survived by Embarked

```
ep<-ggplot(data = titanic.train1,aes(x=Embarked,fill=Survived))+geom_bar(position="fil
l")+facet_wrap(~Pclass)
ep + theme_bw()
```



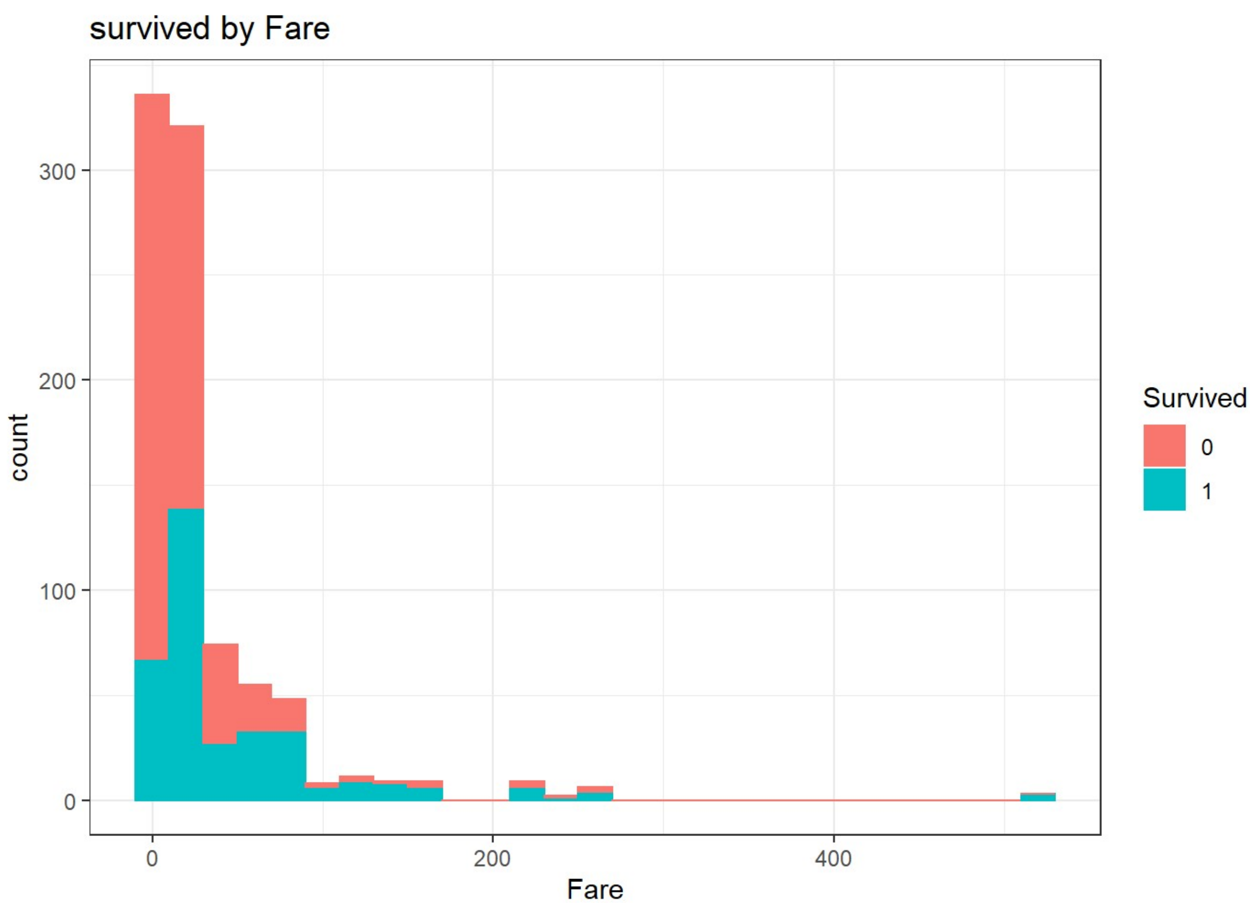
##Survived by Family size#

```
nf<- ggplot(titanic.train1,aes(x=NF,fill= Survived, color= Survived)) +
  geom_histogram(binwidth = 1) + labs(title="survived by NF")
nf + theme_bw()
```



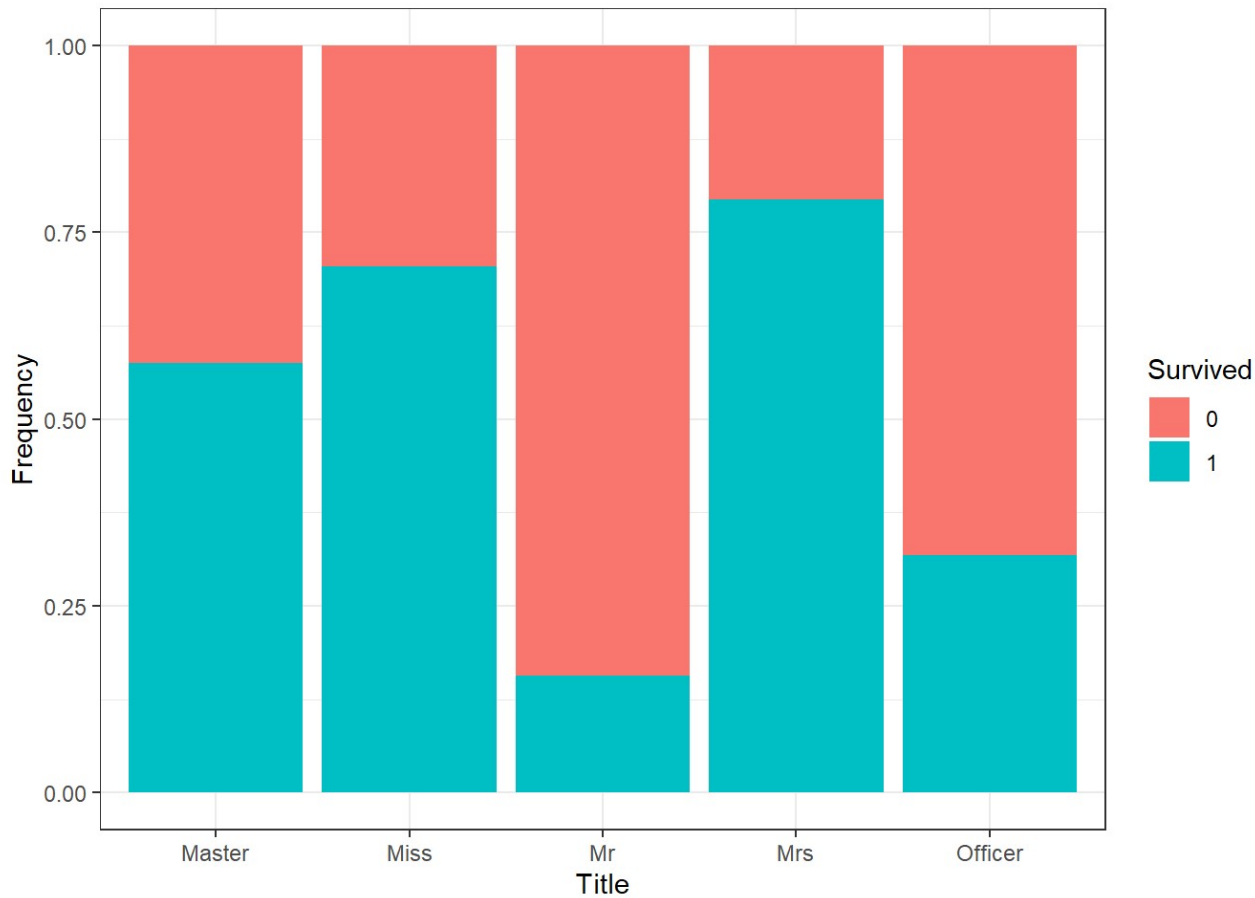
##Survived by Fare

```
ff<- ggplot(titanic.train1,aes(x=Fare,fill= Survived, color= Survived)) +
  geom_histogram(binwidth = 20) + labs(title="survived by Fare")
ff + theme_bw()
```



##Survived by Titles

```
nt<- ggplot(data = titanic.train1,aes(x=Title,fill=Survived))+geom_bar(position="fil
l")+ylab("Frequency")
nt + theme_bw()
```



visual of ggpair plot

```
library(dplyr)
titanic.train2 <- titanic.train1 %>% select (-Name,-Ticket,-Cabin,-PassengerId,-Sex)
#ggpairs(titanic.train2)
```

Chi suquar Testing

We check here with all variable with “Survived” about the null and alternative hypothesis (If p value is < 5 then null is true in our case)

```
tab1 <- table(titanic.train1$AgeGroup,titanic.train1$Survived)
tab1
```

```
##
##       0   1
##   1 177 113
##   2 231 135
##   3 141  94
```

```r
chisq.test(tab1)
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab1
## X-squared = 0.64856, df = 2, p-value = 0.723
```

```r
tab2 <- table(titanic.train1$Sex,titanic.train1$Survived)
tab2
```

```
##
##              0   1
##    female  81 233
##    male   468 109
```

```r
chisq.test(tab2)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tab2
## X-squared = 260.72, df = 1, p-value < 2.2e-16
```

```r
tab3 <- table(titanic.train1$Pclass,titanic.train1$Survived)
tab3
```

```
##
##         0   1
##    1  80 136
##    2  97  87
##    3 372 119
```

```r
chisq.test(tab3)
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab3
## X-squared = 102.89, df = 2, p-value < 2.2e-16
```

```
tab4<- table(titanic.train1$SibSp,titanic.train1$Survived)
tab4
```

```
##
##       0   1
##   0 398 210
##   1  97 112
##   2  15  13
##   3  12   4
##   4  15   3
##   5   5   0
##   8   7   0
```

```
chisq.test(tab4)
```

```
## Warning in chisq.test(tab4): Chi-squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab4
## X-squared = 37.272, df = 6, p-value = 1.559e-06
```

```
tab5 <- table(titanic.train1$Parch,titanic.train1$Survived)
tab5
```

```
##
##       0   1
##   0 445 233
##   1  53  65
##   2  40  40
##   3   2   3
##   4   4   0
##   5   4   1
##   6   1   0
```

```
chisq.test(tab5)
```

```
## Warning in chisq.test(tab5): Chi-squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab5
## X-squared = 27.926, df = 6, p-value = 9.704e-05
```

Checking cor plot

```
str(titanic.train1)
```

```
## 'data.frame':    891 obs. of  16 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     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 358 277 16 5
59 520 629 417 581 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 0 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 681 levels "110152","110413",..: 524 597 670 50 473 276 8
6 396 345 133 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 1 57 1 1 131 1 1 1 ...
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
##  $ AgeGroup   : Factor w/ 3 levels "1","2","3": 2 3 2 3 3 1 3 1 2 1 ...
##  $ Gender     : num  1 0 0 0 1 1 1 1 0 0 ...
##  $ NF         : num  2 2 1 2 1 1 1 5 3 2 ...
##  $ Title      : Factor w/ 5 levels "Master","Miss",..: 3 4 2 4 3 3 3 1 4 4 ...
```

```
titanic.train_dt= subset(titanic.train1, select = c(3,6:8,10,14:15))
titanic.train_dt = as.data.frame(titanic.train_dt)
titanic.train1_cor= cor(titanic.train_dt)
str(titanic.train1_cor)
```

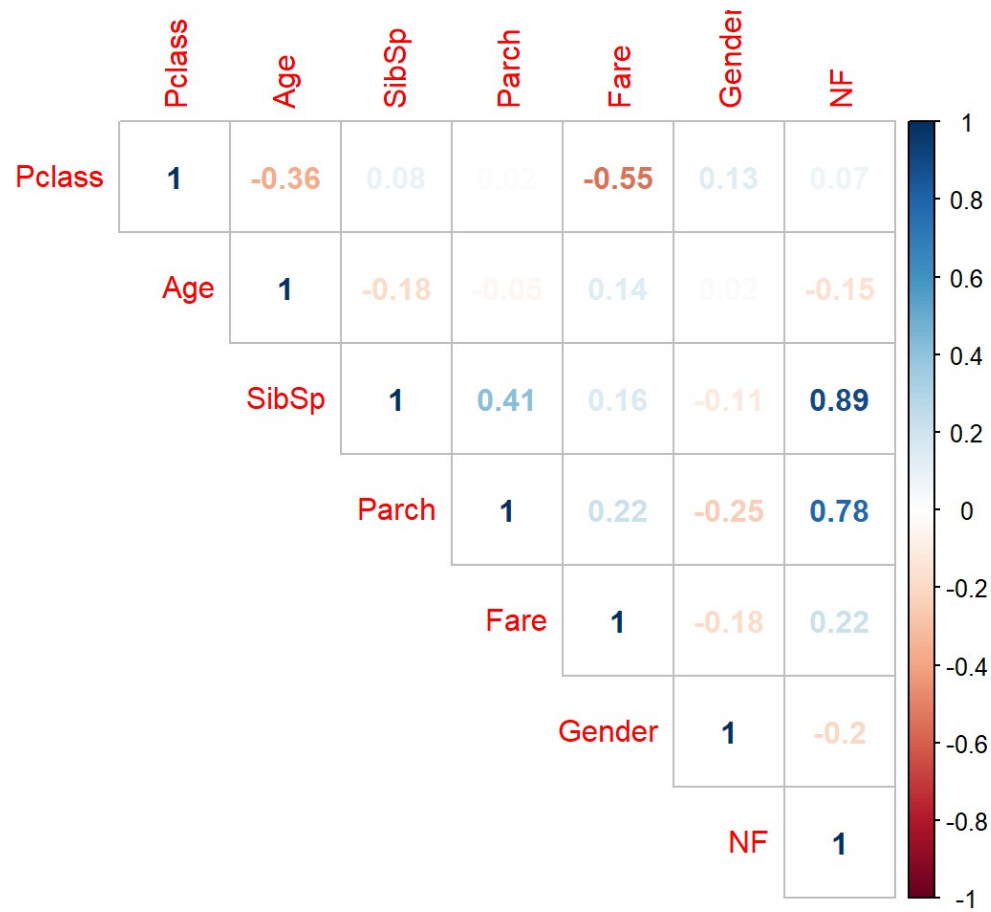
```
##  num [1:7, 1:7] 1 -0.3614 0.0831 0.0184 -0.5495 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:7] "Pclass" "Age" "SibSp" "Parch" ...
##   ..$ : chr [1:7] "Pclass" "Age" "SibSp" "Parch" ...
```

Plotting Corrplot

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
corrplot(titanic.train1_cor, type="upper", method="number")
```



Check the final data for modelling

```
str(titanic.train1)
```

```
## 'data.frame':    891 obs. of  16 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     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 358 277 16 5
59 520 629 417 581 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 0 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 681 levels "110152","110413",..: 524 597 670 50 473 276 8
6 396 345 133 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 1 57 1 1 131 1 1 1 ...
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
##  $ AgeGroup   : Factor w/ 3 levels "1","2","3": 2 3 2 3 3 1 3 1 2 1 ...
##  $ Gender     : num  1 0 0 0 1 1 1 1 0 0 ...
##  $ NF         : num  2 2 1 2 1 1 1 5 3 2 ...
##  $ Title      : Factor w/ 5 levels "Master","Miss",..: 3 4 2 4 3 3 3 1 4 4 ...
```

We have considered only Passenger calss ,Age,Sibsp,Parch and Gender for further anlysis

Model Planning and Building

Logit Model

Since the dependent variable is binary and we need check the multicollinearity of the variable before preparing the final model of logistic regression. We found five variables are carried out good coefficient and allof them are negative coefficient.

Further we have redefined the model to perform better

Here we have Performed the initial regression and later on we have redefined the data set after doing these steps 1. Find out all significant variables. 2. Remove non-performing variables from the module 3. Check multicollinearity with VIF function and rebuild the module

```
titanic_lg<-glm(Survived ~ .,data = titanic.train1[-c(4,9,11)], family = "binomial"(li
nk="logit"))
summary(titanic_lg)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial(link = "logit"),
##     data = titanic.train1[-c(4, 9, 11)])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3816  -0.5507  -0.4024   0.5564   2.5509
##
## Coefficients: (2 not defined because of singularities)
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.140e+01  8.750e+02   0.036   0.9714
## PassengerId   7.817e-05  3.659e-04   0.214   0.8309
## Pclass       -1.017e+00  1.518e-01  -6.703 2.04e-11 ***
## Sexmale      -1.511e+01  6.173e+02  -0.024   0.9805
## Age          -2.631e-03  1.460e-02  -0.180   0.8569
## SibSp        -5.157e-01  1.239e-01  -4.162 3.16e-05 ***
## Parch        -3.082e-01  1.329e-01  -2.319   0.0204 *
## Fare          3.532e-03  2.567e-03   1.376   0.1688
## EmbarkedC    -1.135e+01  6.201e+02  -0.018   0.9854
## EmbarkedQ    -1.158e+01  6.201e+02  -0.019   0.9851
## EmbarkedS    -1.180e+01  6.201e+02  -0.019   0.9848
## AgeGroup2     1.230e-01  4.112e-01   0.299   0.7649
## AgeGroup3    -3.178e-01  6.940e-01  -0.458   0.6470
## Gender               NA         NA      NA       NA
## NF                   NA         NA      NA       NA
## TitleMiss    -1.592e+01  6.173e+02  -0.026   0.9794
## TitleMr      -3.948e+00  5.268e-01  -7.494 6.70e-14 ***
## TitleMrs     -1.541e+01  6.173e+02  -0.025   0.9801
## TitleOfficer -4.033e+00  7.669e-01  -5.259 1.44e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.7  on 890  degrees of freedom
## Residual deviance:  729.9  on 874  degrees of freedom
## AIC: 763.9
##
## Number of Fisher Scoring iterations: 13
```

The initial regression we have chekced with all varaibles where we found Pclass,SibSp,TitleMrand TitleOfficer

Checking only with significian varibales

```
train_fd <- subset(titanic.train1,select= -c(4,5,9:13,15:16))
test_fd <- subset(t.test,select= -c(3,4,8:11,13))
str(train_fd)
```

```
## 'data.frame':    891 obs. of  7 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     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Age        : num  22 38 26 35 35 0 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 ...
##  $ Gender     : num  1 0 0 0 1 1 1 1 0 0 ...
```

Revised Logit model

```
titanic_lg1<-glm(Survived ~ .,data = train_fd[-1], family = "binomial"(link="logit"))
summary(titanic_lg1)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial(link = "logit"),
##     data = train_fd[-1])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3804  -0.6174  -0.4205   0.6373   2.4671
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.215868   0.394962  10.674  < 2e-16 ***
## Pclass      -1.085324   0.117702  -9.221  < 2e-16 ***
## Age         -0.017884   0.005397  -3.314 0.000921 ***
## SibSp       -0.275861   0.099877  -2.762 0.005745 **
## Parch       -0.023242   0.111305  -0.209 0.834593
## Gender      -2.750855   0.195832 -14.047  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1186.66  on 890  degrees of freedom
## Residual deviance:  807.83  on 885  degrees of freedom
## AIC: 819.83
##
## Number of Fisher Scoring iterations: 5
```

Check Multi-Collinearity Effect:

```
library(car)
```

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

```
## Registered S3 methods overwritten by 'car':
##   method                          from
##   influence.merMod                lme4
##   cooks.distance.influence.merMod lme4
##   dfbeta.influence.merMod         lme4
##   dfbetas.influence.merMod        lme4
```

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

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

```
## The following object is masked from 'package:psych':
## 
##     logit
```

```
#( VIF = 1 - Not Correlated # VIF > 1 < 5 - Moderately Correlated # VIF > 5- Highly Co
rrelated)
vif(titanic_lg1)
```

```
##   Pclass      Age    SibSp    Parch   Gender 
## 1.293235 1.227889 1.204500 1.217237 1.194664
```

Predction of train data

```
pred_train = predict.glm(titanic_lg1,newdata=train_fd,type="response")
train.lg <- table(train_fd$Survived,pred_train>0.5)
train.lg
```

```
##    
##     FALSE TRUE
##   0   469   80
##   1   106  236
```

```
accuracy.lg= sum(diag(train.lg))/sum(train.lg)
accuracy.lg
```

```
## [1] 0.7912458
```

Model Performance Measure - Confusion Matrix

Checking train data AUC

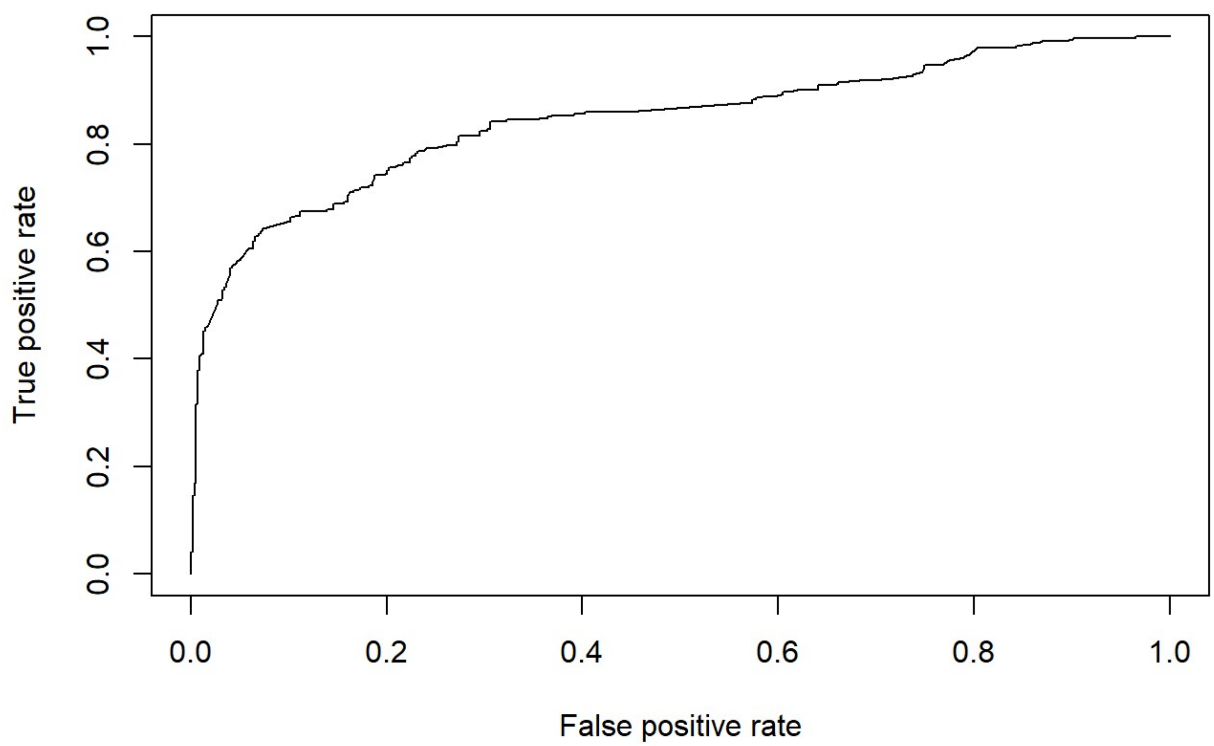
```
library(ROCR)
```

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

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

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

```
DTpredROC1 = ROCR::prediction(pred_train, train_fd$Survived)
perf1 = performance(DTpredROC1, "tpr", "fpr")
plot(perf1)
```



##AUC

```
Auc <- as.numeric(performance(DTpredROC1, "auc")@y.values)
Auc
```

```
## [1] 0.844976
```

KS

```
KS1 <- max(attr(perf1, 'y.values')[[1]]-attr(perf1, 'x.values')[[1]])
KS1
```

```
## [1] 0.5704151
```

Gini

```
## Gini Coefficient
library(ineq)
gini1 = ineq(train_fd$Survived, type="Gini")
gini1
```

```
## [1] 0.1709061
```

Predcting of test data

```
pred_test = predict.glm(titanic_lg1,newdata=t.test,type="response")
head(pred_test)
```

```
##          1          2          3          4          5          6
## 0.08256858 0.46095440 0.14010399 0.09331445 0.56644215 0.11493112
```

Merging with test data

```
test_lg <-ifelse(pred_test>0.5,1,0)
#View(test_fd)
output_lg<- cbind(test_fd[1],test_lg )
View(output_lg)
colnames(output_lg)[2] <- "Survived"
View(test_fd)
```

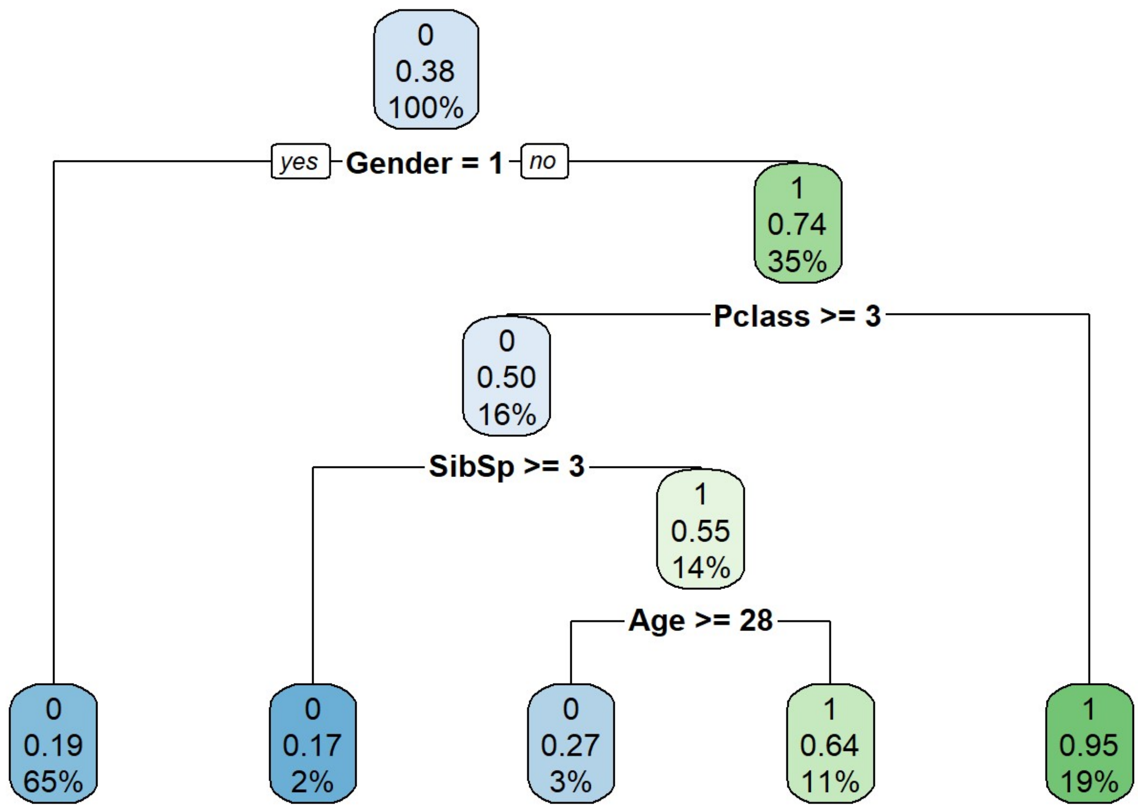
Classification and Regression Tree

CART method has enabled us to determine the complex interactions among variables in the final tree, in contrast to identifying and defining the interactions in a multivariable logistic regression model.

```r
#CART Model
library(rpart)
library(rpart.plot)
r.ctrl = rpart.control(minsplit = 100, minbucket = 10, cp = 0, xval = 10)
CT_model = rpart(Survived ~ ., data = train_fd, method = "class", control = r.ctrl)
CT_model
```

```
## n= 891
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 891 342 0 (0.61616162 0.38383838)
##    2) Gender>=0.5 577 109 0 (0.81109185 0.18890815) *
##    3) Gender< 0.5 314  81 1 (0.25796178 0.74203822)
##      6) Pclass>=2.5 144  72 0 (0.50000000 0.50000000)
##       12) SibSp>=2.5 18   3 0 (0.83333333 0.16666667) *
##       13) SibSp< 2.5 126  57 1 (0.45238095 0.54761905)
##         26) Age>=27.5 30   8 0 (0.73333333 0.26666667) *
##         27) Age< 27.5 96  35 1 (0.36458333 0.63541667) *
##      7) Pclass< 2.5 170   9 1 (0.05294118 0.94705882) *
```

```r
rpart.plot(CT_model)
```




```
attributes(CT_model)
```

```
## $names
##  [1] "frame"               "where"               "call"
##  [4] "terms"               "cptable"             "method"
##  [7] "parms"               "control"             "functions"
## [10] "numresp"             "splits"              "variable.importance"
## [13] "y"                   "ordered"
##
## $xlevels
## named list()
##
## $ylevels
## [1] "0" "1"
##
## $class
## [1] "rpart"
```

```
CT_model$cptable
```

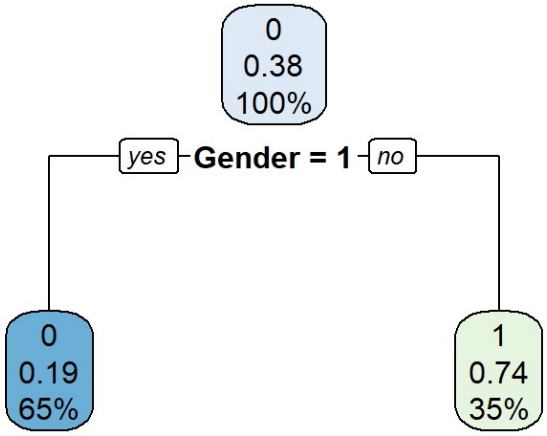
```
##           CP nsplit rel error    xerror       xstd
## 1 0.44444444      0 1.0000000 1.0000000 0.04244576
## 2 0.02534113      1 0.5555556 0.5555556 0.03574957
## 3 0.00000000      4 0.4795322 0.5555556 0.03574957
```

Pruning the tree

```
ptree = prune(CT_model, .035, "CP")
ptree
```

```
## n= 891
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 891 342 0 (0.6161616 0.3838384)
##   2) Gender>=0.5 577 109 0 (0.8110919 0.1889081) *
##   3) Gender< 0.5 314  81 1 (0.2579618 0.7420382) *
```

```
rpart.plot(ptree)
```




```
CT_model$variable.importance
```

```
##      Gender      Pclass         Age       Parch       SibSp PassengerId 
##  124.426330   31.163132   16.820193   13.880781    8.033999    5.337148
```

CART validation data

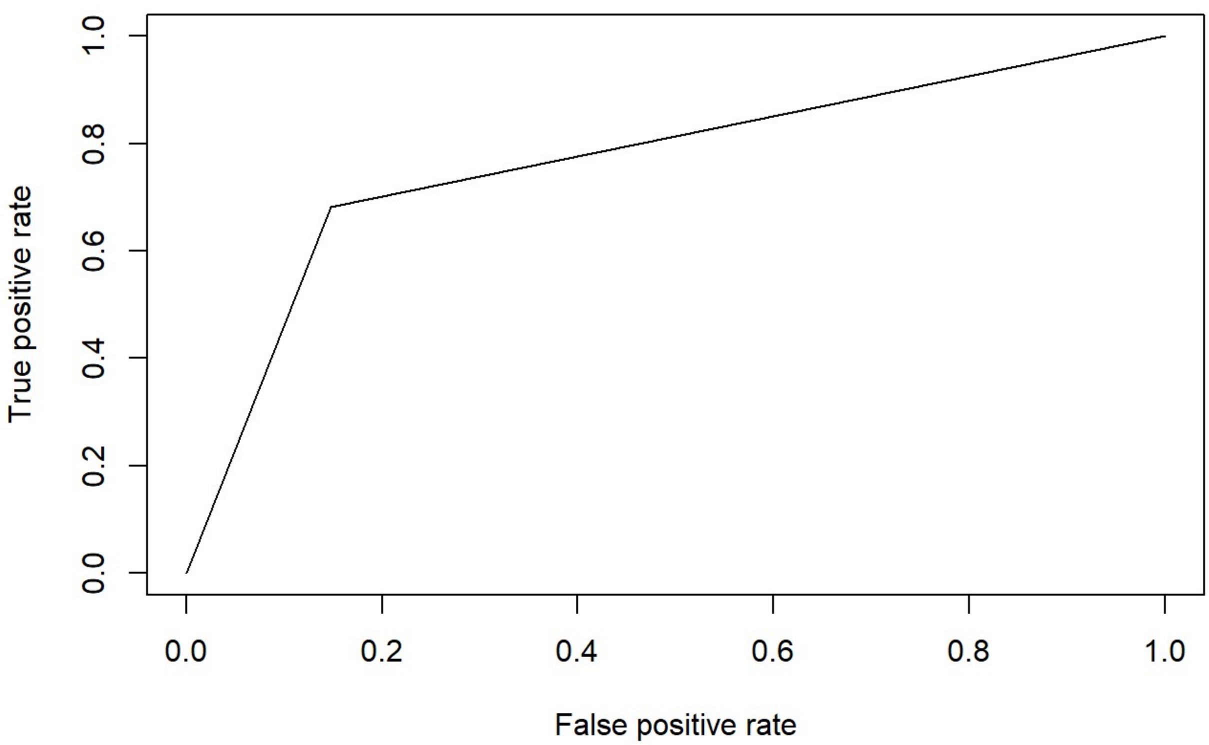
```
str(test_fd)
```

```
## 'data.frame':    418 obs. of  6 variables:
##  $ PassengerId: int  892 893 894 895 896 897 898 899 900 901 ...
##  $ Pclass     : int  3 3 2 3 3 3 3 2 3 3 ...
##  $ Age        : num  34.5 47 62 27 22 14 30 26 18 21 ...
##  $ SibSp      : int  0 1 0 0 1 0 0 1 0 2 ...
##  $ Parch      : int  0 0 0 0 1 0 0 1 0 0 ...
##  $ Gender     : num  1 0 1 1 0 1 0 1 0 1 ...
```

```
predTrain = predict(ptree, newdata = train_fd)
pred_class = predict(ptree, newdata = train_fd[,-2], type = "class")
predDT = predict(ptree, newdata = test_fd[,-7], type = "prob")
predDT_cl = predict(ptree, newdata = test_fd[,-7], type = "class")
#predDT = predict(ptree, newdata = t.test)
```

Validation on train data

```
library(ROCR)
DTpredROC_CT = prediction(predTrain [,2], titanic.train1$Survived)
perf2 =performance(DTpredROC_CT, "tpr", "fpr")
plot(perf2)
```




```
Auc <- as.numeric(performance(DTpredROC_CT, "auc")@y.values)
Auc
```

```
## [1] 0.7668728
```

```
table(titanic.train1$Survived, pred_class)
```

```
##    pred_class
##       0   1
##   0 468  81
##   1 109 233
```

```
accuracy.ct= (468+233)/(468+233+81+109)
accuracy.ct
```

```
## [1] 0.7867565
```

Predcting with test data

```
output_cart<- cbind(test_fd[1],predDT_cl )
View(output_cart)
colnames(output_cart)[2] <- "Survived"
```

```
str(titanic.train1)
```

```
## 'data.frame':    891 obs. of  16 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     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 358 277 16 5
59 520 629 417 581 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 0 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 681 levels "110152","110413",..: 524 597 670 50 473 276 8
6 396 345 133 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 1 57 1 1 131 1 1 1 ...
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
##  $ AgeGroup   : Factor w/ 3 levels "1","2","3": 2 3 2 3 3 1 3 1 2 1 ...
##  $ Gender     : num  1 0 0 0 1 1 1 1 0 0 ...
##  $ NF         : num  2 2 1 2 1 1 1 5 3 2 ...
##  $ Title      : Factor w/ 5 levels "Master","Miss",..: 3 4 2 4 3 3 3 1 4 4 ...
```

```
train__rf <- subset (titanic.train1,select = c(4))
```

Random Forest

RF we have used because its tree-based strategies naturally it ranks by how well the model improve the purity of the node. This mean decrease in impurity over all trees (called Gini impurity)and It reduces the complexity of a model and makes it easier to interpret.

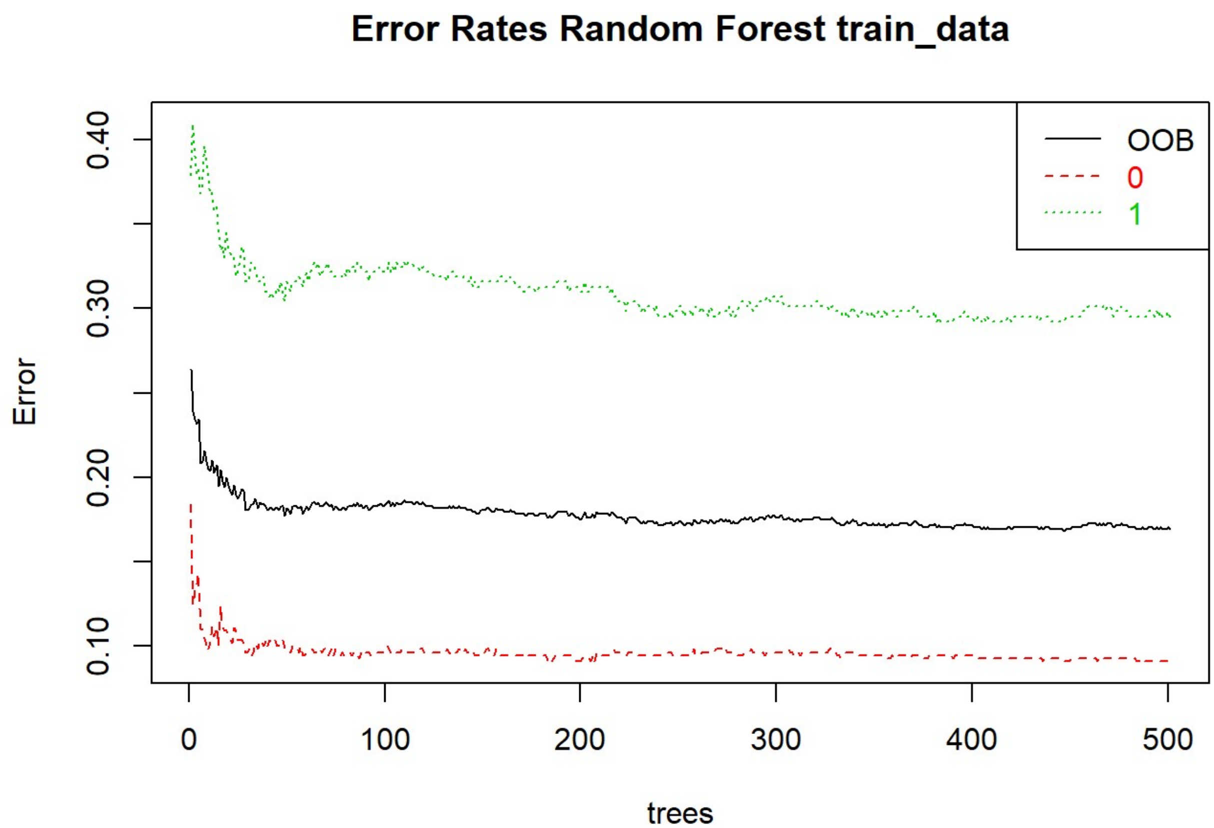
```
library(randomForest)
seed=101
set.seed(seed)
RF_model = randomForest(Survived ~ ., data = train_fd, mtry = 3, nodesize =10, ntree =
501, importance = TRUE)
print(RF_model)
```

```
##
## Call:
##  randomForest(formula = Survived ~ ., data = train_fd, mtry = 3,      nodesize = 1
0, ntree = 501, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 501
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 16.95%
## Confusion matrix:
##     0   1 class.error
## 0 499  50  0.09107468
## 1 101 241  0.29532164
```

In above we got 16.95% is out of bag

ploting RF moden

```
plot(RF_model, main="")
legend("topright", c("OOB", "0", "1"), text.col=1:6, lty=1:3, col=1:3)
title(main="Error Rates Random Forest train_data")
```



Non survied are having greater accuracy .

Checking OOB

```
rf_err_rate <-  as.data.frame(RF_model$err.rate)
rf_err_rate$ID <- seq.int(nrow(rf_err_rate))
rf_err_rate[which(rf_err_rate$OOB==min(rf_err_rate$OOB)),]
```

```
##           OOB          0         1  ID
## 447 0.1683502 0.09107468 0.2923977 447
```

```
min_tree<-min(rf_err_rate[which(rf_err_rate$OOB==min(rf_err_rate$OOB)),]$ID)
```

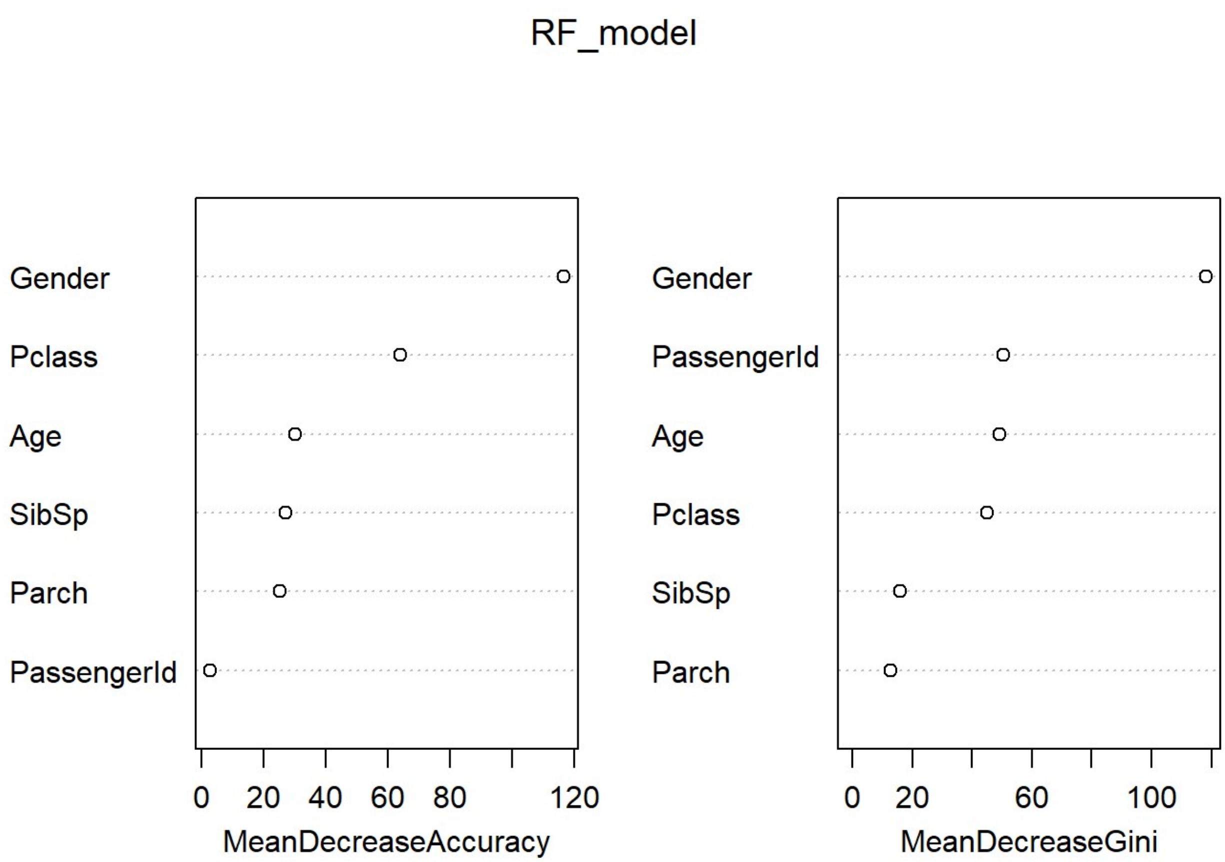
List the importance of the variables

```
## List the importance of the variables.
impVar <- round(randomForest::importance(RF_model), 2)
impVar[order(impVar[,3],decreasing = TRUE),]
```

```
##                 0     1 MeanDecreaseAccuracy MeanDecreaseGini
## Gender      91.43 93.03               116.60           118.25
## Pclass      34.53 60.96                63.84            45.02
## Age         15.95 26.61                30.13            49.06
## SibSp       30.67 -2.94                27.12            15.94
## Parch       22.65  9.96                25.27            12.83
## PassengerId  0.36  4.08                 2.84            50.63
```

Variable Importance: Graphical representation

```
varImpPlot(RF_model)
```



Checking Accuracy of train data

```
#Pass the Test data through RF model
predRF = predict(RF_model, newdata = train_fd, type="class")
predRF1 = predict(RF_model, newdata = train_fd, type="prob")
#Check model performance using confusion matrix
table(train_fd$Survived, predRF)
```

```
##    predRF
##       0   1
##   0 519  30
##   1  72 270
```

```
accuracy.rf=(519+270)/(519+270+30+72)
accuracy.rf
```

```
## [1] 0.8855219
```

Validating with test data

```
predRF_test = predict(RF_model, newdata = test_fd[-7], type="class")
predRF1_test = predict(RF_model, newdata = test_fd[-7], type="prob")
```

Predicting wit test data

```
output_RF<- cbind(test_fd[1],predRF_test )
View(output_RF)
colnames(output_RF)[2] <- "Survived"
write.csv(output_RF,"titanic_kaggle_submission.csv",row.names = FALSE)
```

Machine learning approach

Naive Bayes

We have used Naive Bayes classifiers because Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods.

The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

```
#naive bayes
library(e1071)
library(caret)
train_fd$Survived = as.factor(train_fd$Survived)
NB = naiveBayes(x =train_fd[-2], y =train_fd$Survived)
pred.train.NB = predict(NB, newdata =train_fd[-2])
tab.NB =table(train_fd[,2], pred.train.NB)
accuracy.nb = sum(diag(tab.NB))/sum(tab.NB)
accuracy.nb
```

```
## [1] 0.7699214
```

Clasifiying all variable with Survived - using NaiveBays classifer

```
names(titanic.train1)
```

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

```
library(dplyr)
titanic.train3 <- titanic.train1 %>% select (-Name,-Ticket,-Cabin,-PassengerId,-Sex)
str(titanic.train3)
```

```
## 'data.frame':    891 obs. of  11 variables:
##  $ Survived: Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ Pclass  : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Age     : num  22 38 26 35 35 0 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 ...
##  $ Fare    : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Embarked: Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
##  $ AgeGroup: Factor w/ 3 levels "1","2","3": 2 3 2 3 3 1 3 1 2 1 ...
##  $ Gender  : num  1 0 0 0 1 1 1 1 0 0 ...
##  $ NF      : num  2 2 1 2 1 1 1 5 3 2 ...
##  $ Title   : Factor w/ 5 levels "Master","Miss",..: 3 4 2 4 3 3 3 1 4 4 ...
```

```r
#install.packages(mlr)
library(mlr)
```

```
## Loading required package: ParamHelpers
```

```
##
## Attaching package: 'mlr'
```

```
## The following object is masked from 'package:e1071':
##
##     impute
```

```
## The following object is masked from 'package:ROCR':
##
##     performance
```

```
## The following object is masked from 'package:caret':
##
##     train
```

```r
#Create a classification task for learning on training data Data set and specify Survi
ved feature
task = makeClassifTask(data =titanic.train3,target = "Survived")
#Initialize the Naive Bayes classifier
selected_model = makeLearner("classif.naiveBayes")
#Train the model
NB_mlr = train(selected_model, task)
#Read the model learned
NB_mlr$learner.model
```

```
## 
## Naive Bayes Classifier for Discrete Predictors
## 
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
## 
## A-priori probabilities:
## Y
##         0         1 
## 0.6161616 0.3838384 
## 
## Conditional probabilities:
##    Pclass
## Y       [,1]      [,2]
##   0 2.531876 0.7358050
##   1 1.950292 0.8633206
## 
##    Age
## Y       [,1]     [,2]
##   0 23.65301 17.89615
##   1 24.03412 17.12672
## 
##    SibSp
## Y        [,1]      [,2]
##   0 0.5537341 1.2883991
##   1 0.4736842 0.7086875
## 
##    Parch
## Y        [,1]     [,2]
##   0 0.3296903 0.823166
##   1 0.4649123 0.771712
## 
##    Fare
## Y       [,1]     [,2]
##   0 22.11789 31.38821
##   1 48.39541 66.59700
## 
##    Embarked
## Y                         C           Q           S
##   0 0.000000000 0.136612022 0.085610200 0.777777778
##   1 0.005847953 0.271929825 0.087719298 0.634502924
## 
##    AgeGroup
## Y           1         2         3
##   0 0.3224044 0.4207650 0.2568306
##   1 0.3304094 0.3947368 0.2748538
## 
##    Gender
```

```
## Y         [,1]      [,2]
##   0 0.8524590 0.3549678
##   1 0.3187135 0.4666604
##
##    NF
## Y        [,1]     [,2]
##   0 1.883424 1.830669
##   1 1.938596 1.186076
##
##    Title
## Y        Master       Miss         Mr        Mrs    Officer
##   0 0.03096539 0.10018215 0.79417122 0.04735883 0.02732240
##   1 0.06725146 0.38304094 0.23684211 0.29239766 0.02046784
```

Confusion matrix to check accuracy

```
predictions_mlr = as.data.frame(predict(NB_mlr, newdata = titanic.train3[,2:10]))

table(predictions_mlr[,1],titanic.train3$Survived)
```

```
##
##       0   1
##   0 452  95
##   1  97 247
```

```
accuracy.nb_mlr= (452+247)/(452+247+95+97)
accuracy.nb_mlr
```

```
## [1] 0.7845118
```

Bagging

Bootstrapping is a sampling technique in which we create subsets of observations from the original dataset, with replacement. The size of the subsets is the same as the size of the original set. Bagging (or Bootstrap Aggregating) technique uses these subsets (bags) to get a fair idea of the distribution (complete set). The size of subsets created for bagging may be less than the original set. . Multiple subsets are created from the original dataset, selecting observations with replacement. . A base model (weak model) is created on each of these subsets. . The models run in parallel and are independent of each other. . The final predictions are determined by combining the predictions from all the models.

```
#Bagging#
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
#install.packages('xgboost')
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
```

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

```
#install.packages('caret')
library(caret)
library(ipred)
library(rpart)

titanic_bagging <- bagging(Survived ~.,data=train_fd,
                         control=rpart.control(maxdepth=5, minsplit=4))

pred_class <- predict(titanic_bagging, train_fd)

tab.bg <- table(train_fd$Survived,pred_class)
accuracy.bg = sum(diag(tab.bg))/sum(tab.bg)
accuracy.bg
```

```
## [1] 0.8540965
```

XGBoost

XGBoost (extreme Gradient Boosting) is an advanced implementation of the gradient boosting algorithm. XGBoost has proved to be a highly effective ML algorithm, extensively used in machine learning competitions and hackathons. XGBoost has high predictive power and is almost 10 times faster than the other gradient boosting techniques. It also includes a variety of regularization which reduces overfitting and improves overall performance. Hence it is also known as 'regularized boosting' technique.

```
# XGBoost
#install.packages('xgboost')
library(xgboost)
set.seed(123)
classifier = xgboost(data = as.matrix(train_fd[,-2]), label = train_fd$Survived, nroun
ds = 10)
```

```
## [1]  train-rmse:0.748994
## [2]  train-rmse:0.574208
## [3]  train-rmse:0.460650
## [4]  train-rmse:0.389583
## [5]  train-rmse:0.340342
## [6]  train-rmse:0.310141
## [7]  train-rmse:0.293900
## [8]  train-rmse:0.278271
## [9]  train-rmse:0.269142
## [10] train-rmse:0.261325
```

```
#Predicting the Test set results
y_pred <- predict(classifier, newdata = as.matrix(train_fd[-2]))
y_pred = (y_pred >= 0.5)

# Making the Confusion Matrix
cm = table(train_fd$Survived, y_pred)
cm
```

```
##    y_pred
##     TRUE
##   0  549
##   1  342
```

```
accuracy.bs = sum(diag(cm))/sum(cm)
accuracy.bs
```

```
## [1] 0.6161616
```

K-Fold Cross Validation

We know ,in the K-fold cross-validation method, the dataset is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the remaining k 1 subsets are put together to form a training set. The average error across all k trials is then calculated. The advantage of this method is that every data point has one chance to be in a test set exactly once and has the chance to be in a training set k 1 times. The variance of the resulting estimate is reduced as k is increased. The disadvantage of this method is that it suffers from heavy computational complexity, because the training algorithm has to be rerun from scratch k times, so it takes k times as much computation to make an evaluation. For our data set we have created 10 CV folds

```
library(caret)
folds_titanic = createFolds(train_fd$Survived, k =10)
cv = lapply(folds_titanic, function(x) {
tr_fold = train_fd[-x, ]
tt_fold = train_fd[x, ]
classifier = xgboost(data = as.matrix(train_fd[-2]), label = train_fd$Survived, nround
s = 10)
y_pred = predict(classifier, newdata = as.matrix(tt_fold[-2]))
y_pred = (y_pred >= 0.5)
cmx= table(tt_fold[,2], y_pred)
accuracy = (cmx[1,1] + cmx[2,1])  / (cmx[1,1] + cmx[2,1] + cmx[1,1] + cmx[2,1])
return(accuracy)
})
```

```
## [1]  train-rmse:0.748994
## [2]  train-rmse:0.574208
## [3]  train-rmse:0.460650
## [4]  train-rmse:0.389583
## [5]  train-rmse:0.340342
## [6]  train-rmse:0.310141
## [7]  train-rmse:0.293900
## [8]  train-rmse:0.278271
## [9]  train-rmse:0.269142
## [10] train-rmse:0.261325
## [1]  train-rmse:0.748994
## [2]  train-rmse:0.574208
## [3]  train-rmse:0.460650
## [4]  train-rmse:0.389583
## [5]  train-rmse:0.340342
## [6]  train-rmse:0.310141
## [7]  train-rmse:0.293900
## [8]  train-rmse:0.278271
## [9]  train-rmse:0.269142
## [10] train-rmse:0.261325
## [1]  train-rmse:0.748994
## [2]  train-rmse:0.574208
## [3]  train-rmse:0.460650
## [4]  train-rmse:0.389583
## [5]  train-rmse:0.340342
## [6]  train-rmse:0.310141
## [7]  train-rmse:0.293900
## [8]  train-rmse:0.278271
## [9]  train-rmse:0.269142
## [10] train-rmse:0.261325
## [1]  train-rmse:0.748994
## [2]  train-rmse:0.574208
## [3]  train-rmse:0.460650
## [4]  train-rmse:0.389583
## [5]  train-rmse:0.340342
## [6]  train-rmse:0.310141
## [7]  train-rmse:0.293900
## [8]  train-rmse:0.278271
## [9]  train-rmse:0.269142
## [10] train-rmse:0.261325
## [1]  train-rmse:0.748994
## [2]  train-rmse:0.574208
## [3]  train-rmse:0.460650
## [4]  train-rmse:0.389583
## [5]  train-rmse:0.340342
## [6]  train-rmse:0.310141
## [7]  train-rmse:0.293900
## [8]  train-rmse:0.278271
```

```
## [9]  train-rmse:0.269142
## [10] train-rmse:0.261325
## [1]  train-rmse:0.748994
## [2]  train-rmse:0.574208
## [3]  train-rmse:0.460650
## [4]  train-rmse:0.389583
## [5]  train-rmse:0.340342
## [6]  train-rmse:0.310141
## [7]  train-rmse:0.293900
## [8]  train-rmse:0.278271
## [9]  train-rmse:0.269142
## [10] train-rmse:0.261325
## [1]  train-rmse:0.748994
## [2]  train-rmse:0.574208
## [3]  train-rmse:0.460650
## [4]  train-rmse:0.389583
## [5]  train-rmse:0.340342
## [6]  train-rmse:0.310141
## [7]  train-rmse:0.293900
## [8]  train-rmse:0.278271
## [9]  train-rmse:0.269142
## [10] train-rmse:0.261325
## [1]  train-rmse:0.748994
## [2]  train-rmse:0.574208
## [3]  train-rmse:0.460650
## [4]  train-rmse:0.389583
## [5]  train-rmse:0.340342
## [6]  train-rmse:0.310141
## [7]  train-rmse:0.293900
## [8]  train-rmse:0.278271
## [9]  train-rmse:0.269142
## [10] train-rmse:0.261325
## [1]  train-rmse:0.748994
## [2]  train-rmse:0.574208
## [3]  train-rmse:0.460650
## [4]  train-rmse:0.389583
## [5]  train-rmse:0.340342
## [6]  train-rmse:0.310141
## [7]  train-rmse:0.293900
## [8]  train-rmse:0.278271
## [9]  train-rmse:0.269142
## [10] train-rmse:0.261325
## [1]  train-rmse:0.748994
## [2]  train-rmse:0.574208
## [3]  train-rmse:0.460650
## [4]  train-rmse:0.389583
## [5]  train-rmse:0.340342
## [6]  train-rmse:0.310141
## [7]  train-rmse:0.293900
```

```
## [8]  train-rmse:0.278271
## [9]  train-rmse:0.269142
## [10] train-rmse:0.261325
```

```
accuracy.kf = mean(as.numeric(cv))
accuracy.kf
```

```
## [1] 0.5
```

Summary of Accuracy

```
#Accuracy of Logistic Regression
accuracy.lg
```

```
## [1] 0.7912458
```

```
#Accuracy of CART
accuracy.ct
```

```
## [1] 0.7867565
```

```
#Accuracy of Random Forest
accuracy.rf
```

```
## [1] 0.8855219
```

```
#Accuracy of Naive Bayes
accuracy.nb
```

```
## [1] 0.7699214
```

```
#Accuracy of Bagging
accuracy.bg
```

```
## [1] 0.8540965
```

```
#Accuracy of XBoost
accuracy.bs
```

```
## [1] 0.6161616
```

```
#Accuracy of KNN cross folding
accuracy.kf
```

```
## [1] 0.5
```

Submission csv

```
write.csv(output_RF,"titanic_kaggle_submission.csv",row.names = FALSE)
```

Conclusion

The best models are Random forest , by which we predicted with 88 % accuracy, whether passenger is survived or not . Whereas Bagging method can 85% accuracy . Age group third (more than 35) , belongs to passenger class 1 and married female with earmarked “c” might had more probability to be survived

Thank you for your precious time , If you like it please upvote