TITANIC DATA ANALYSIS

PROBLEM DISCRIPTION

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 will 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.

GOAL

to predict if a passenger survived the sinking of the Titanic or not. For each in the test set, you must predict a 0 or 1 value for the variable.

DATA OVERVIEW

The data has been split into two groups:

training set (train.csv)

test set (test.csv)

The training set should be used to build machine learning models. For the training set, outcome (also known as the "ground truth") for each passenger will be provided. model will be based on "features" like passengers' gender and class. We will use feature engineering to create new features.

The test set should be used to see how well your model performs on unseen data. For the test set, we do not provide the ground truth for each passenger. It is your job to predict these outcomes. For each passenger in the test set, use the model you trained to predict whether or not they survived the sinking of the Titanic.

LETS SEE THE DATA DICTIONARY WHICH IS THE DESCRIPTION OF THE ATTRIBUTES THAT TO BE USED FOR ANALYSIS-:

DATA DICTIONARY

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Variable Notes

pclass: A proxy for socio-economic status (SES)
1st = Upper
2nd = Middle
3rd = Lower

age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

sibsp: The dataset defines family relations in this way...
Sibling = brother, sister, stepbrother, stepsister
Spouse = husband, wife (mistresses and fiancés were ignored)

parch: The dataset defines family relations in this way...
Parent = mother, father
Child = daughter, son, stepdaughter, stepson
Some children travelled only with a nanny, therefore parch=0 for them.

ALGORITHM TO BE USED-:

RANDOM FOREST ALGORITHM

TOOL-:

RSTUDIO

RANDOM FOREST ALGORITHM

Random Forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because it's simplicity and the fact that it can be used for both classification and regression tasks.

Random forest algorithm is a supervised classification algorithm. As the name suggest, this algorithm creates the forest with a number of trees.

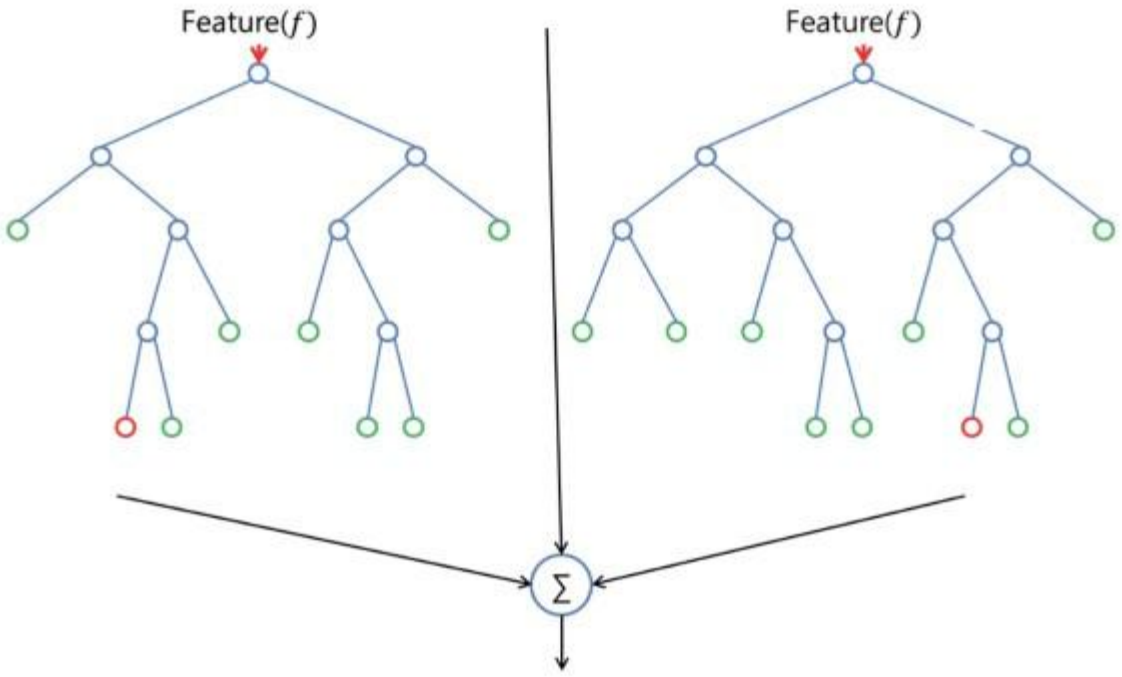
In general, the more trees in the forest the more robust the forest looks like. In the same way in the random forest classifier, the higher the number of trees in the forest gives the high accuracy results.

Lets see in brief how it works-:

Random Forest is a supervised learning algorithm. Like you can already see from it's name, it creates a forest and makes it somehow random. The „forest" it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result.

One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. I will talk about random forest in classification, since classification is sometimes considered the

building block of machine learning. Below you can see how a random forest would look like with two trees:



Random Forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, you don't have to combine a decision tree with a bagging classifier and can just easily use the classifier-class of Random Forest. Like I already said, with Random Forest, you can also deal with Regression tasks by using the Random Forest regressor.

Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random, by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

WHY RANDOM FOREST-:

Another great quality of the random forest algorithm is that it is very easy to measure the relative importance of each feature on the prediction. Sklearn provides a great tool for this, that measures a features importance by looking at how much the tree nodes, which use that feature, reduce impurity across all trees in the forest. It computes this score automatically for each feature after training and scales the results, so that the sum of all importance is equal to 1.

We want to do feature selection from data that is by seeing which variable we can tell the survival rate. That's why this is the best algorithm to be selected.

LETS SEE THE IMPLEMENTATION PART THAT IS DONE USING R LANGUAGE-

IMPLEMENTATION

THE IMPLEMENTATION IS DIVIDED INTO TWO PART-:

1.ANALYSIS

2.ALGORITHM IMPLEMENTATION

In Analysis part we will be analyzing how much one variable support the survival rate.

In algorithm part we will come on conclusion

1.ANALYSIS

Here there are two data sets namely test and train. Test have additional survive column that train doesn't have. So first we will copy the whole column and then put it in to train. Than we combine those datasets respectively and then make another object named data.combined . all those which having null values will separated and then we will take one by one variable to check how much survival rate it can give.

We will make one object named title which will be having categorization as whether one person is MR.,or MRS. Etc.

Since R cant handle characters so we will categorize it into factors so we can use it. After seeing all the graph we will see than title is having more relation to survival rate Than any one.

IMPLEMENTATION

train<-read.csv("train.csv",header = TRUE)

test<-read.csv("test.csv",header=TRUE)

view(train)

test.survived <- data.frame(Survived = rep("None", nrow(test)), test[,])

data.combined<-rbind(train,test.survived)

str(data.combined)

```
> str(data.combined)
'data.frame':   1309 obs. of  12 variables:
 $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
 $ Survived   : chr  "0" "1" "1" "1" ...
 $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
 $ Name       : Factor w/ 1307 levels "Abbing, Mr. Anthony",..: 109 191 358
277 16 559 520 629 417 581 ...
 $ 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/ 929 levels "110152","110413",..: 524 597 670 50 4
3 276 86 396 345 133 ...
 $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
 $ Cabin      : Factor w/ 187 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 ...
```

data.combined$Survived<-as.factor(data.combined$Survived)

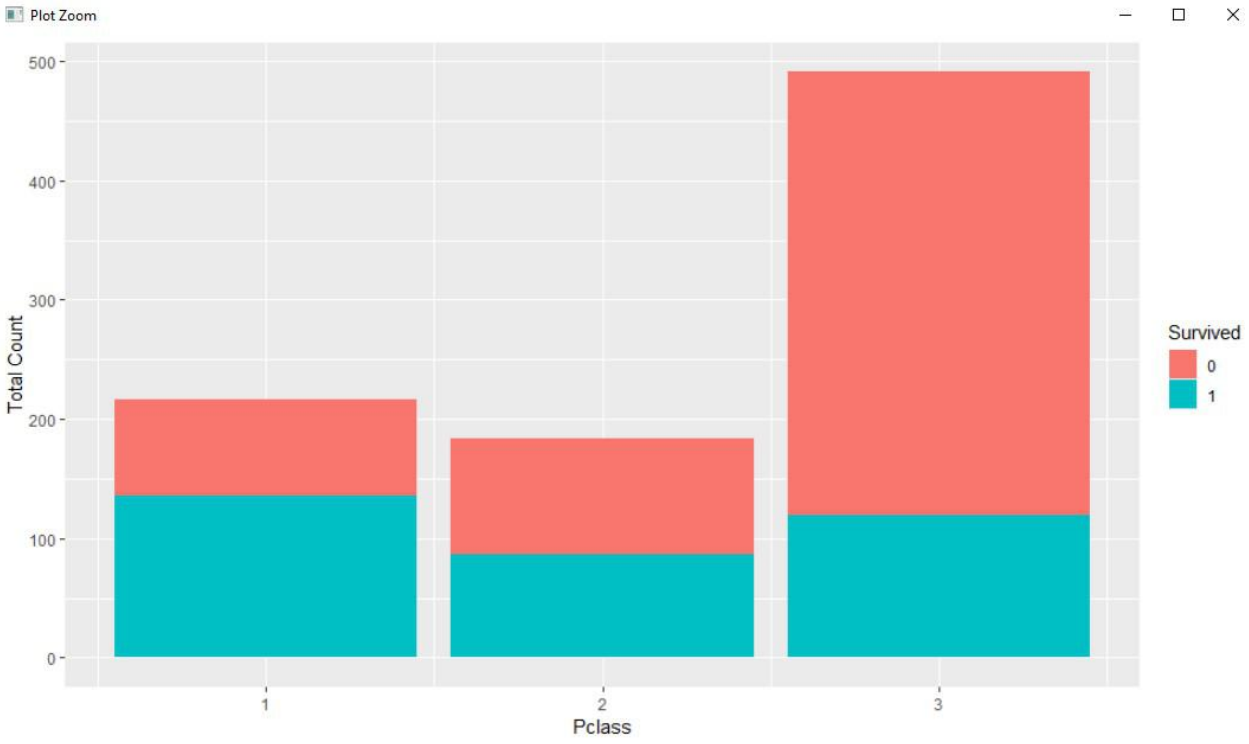
data.combined$Pclass<-as.factor(data.combined$Pclass)

table(data.combined$Survived)

table(data.combined$Pclass)

library(ggplot2)

```
ggplot(train, aes(x = Pclass, fill =
 factor(Survived))) + geom_bar() +
 xlab("Pclass") +
 ylab("Total Count") +
 labs(fill = "Survived")
```

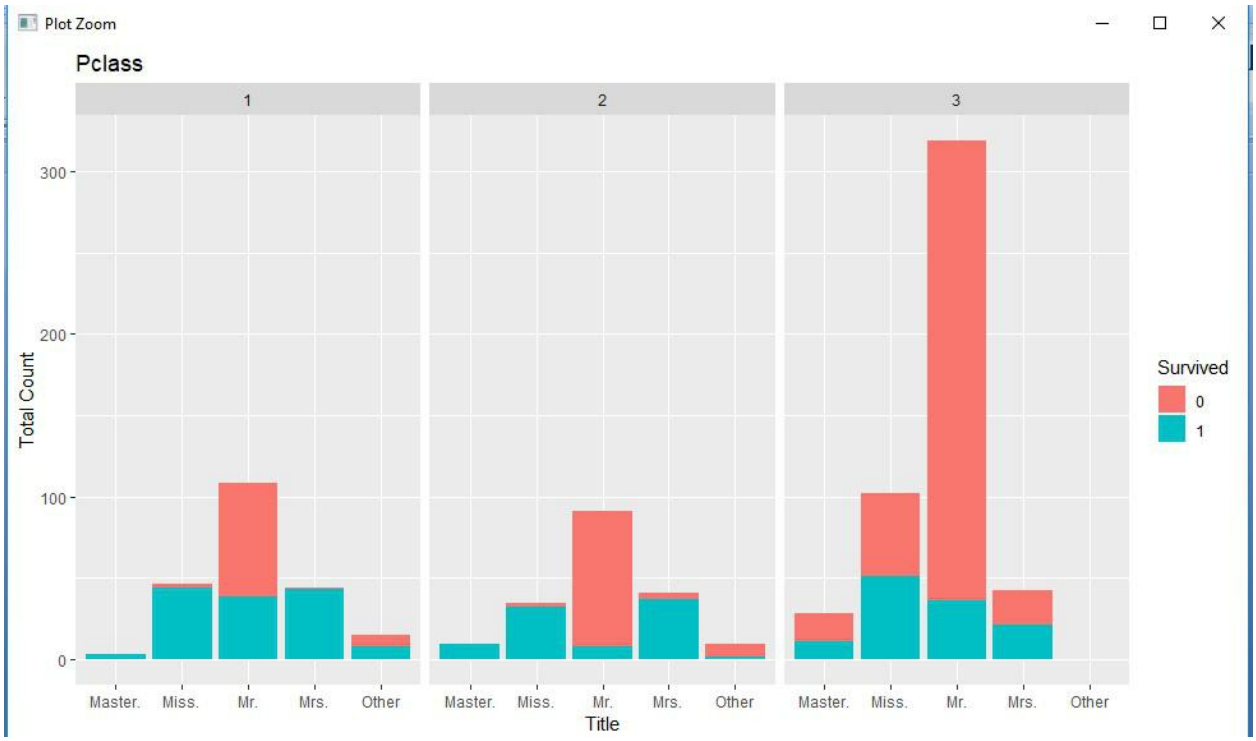



```
 head(as.character(train$Name))
 length(as.character(data.combined$Name))
 length(unique(as.character(data.combined$Name)))
 dup.name <-
as.character(data.combined[which(duplicated(as.character(data.co
mbined$Name))), "Name"])
 data.combined[which(data.combined$Name %in% dup.name),]
```

```
 library(stringr)
 misses<-
data.combined[which(str_detect(data.combined$Name,"Miss.")),]
 misses[1:5,]
 mrs<-
data.combined[which(str_detect(data.combined$Name,"Mrs.")),]
 mrs[1:5,]
 males<-data.combined[which(train$Sex=="male"),]
 males[1:5,]
 master<-
data.combined[which(str_detect(data.combined$Name,"Master.")),]
 master[1:5,]
 master[,]
 name<-data.combined$Name
 extractTitle <- function(name) {
  name <- as.character(name)

  if (length(grep("Miss.", name)) >
   0) { return ("Miss.")
  } else if (length(grep("Master.", name)) >
   0) { return ("Master.")
```

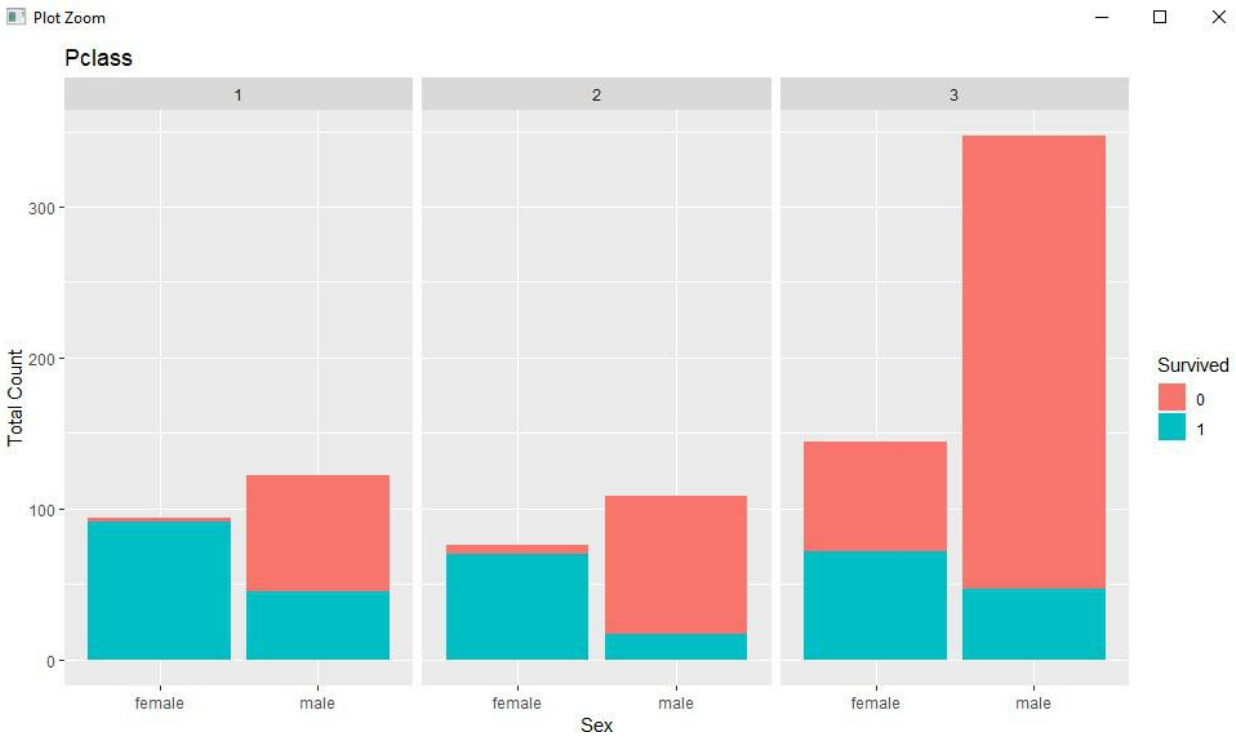
```
  } else if (length(grep("Mrs.", name)) >
    0) { return ("Mrs.")
  } else if (length(grep("Mr.", name)) >
    0) { return ("Mr.")
  } else {
    return ("Other")
  }
}



titles <- NULL
for (i in 1:nrow(data.combined)) {
  titles <- c(titles, extractTitle(data.combined[i,"Name"]))
}
data.combined$title <- as.factor(titles)

ggplot(data.combined[1:891,], aes(x = title, fill =
  Survived)) + geom_bar() +
  facet_wrap(~Pclass) +
  ggtitle("Pclass") +
```

```
xlab("Title") +
ylab("Total Count") +
labs(fill = "Survived")
```

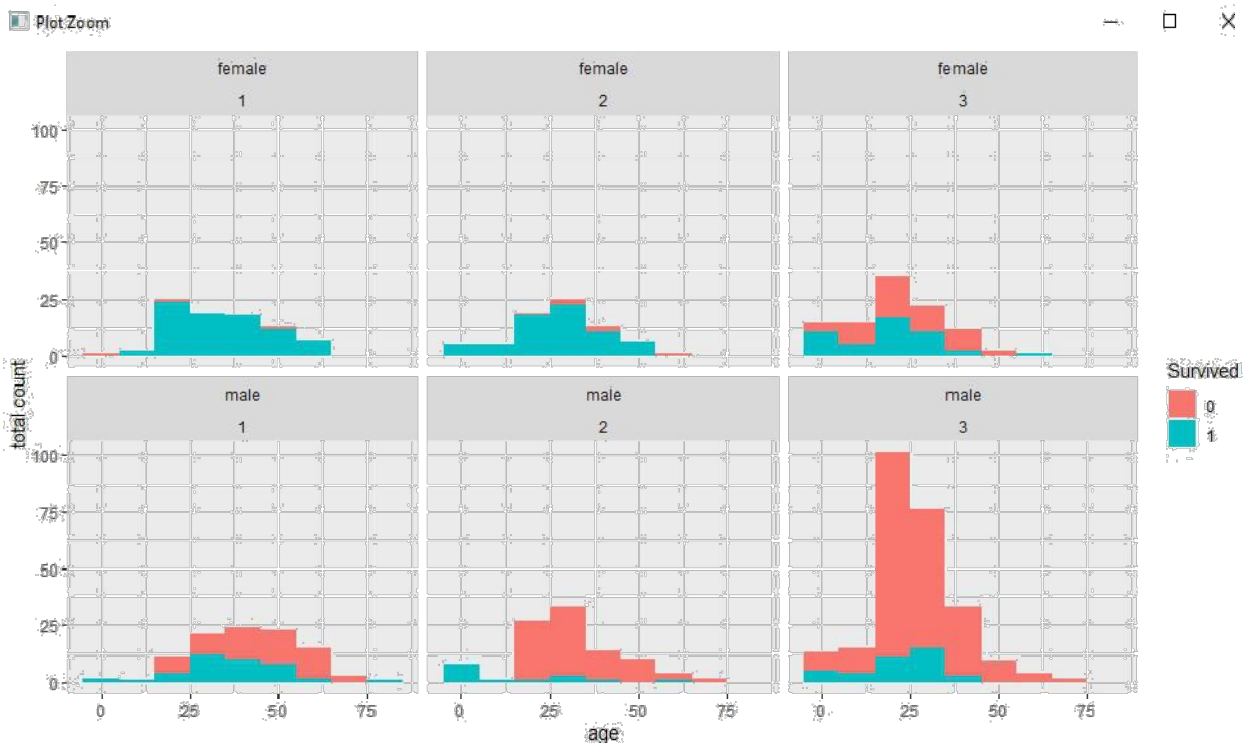



```
table(data.combined$Sex)
ggplot(data.combined[1:891,], aes(x = Sex, fill =
 Survived)) + geom_bar() +
 facet_wrap(~Pclass) +
 ggtitle("Pclass") +
 xlab("Sex") +
 ylab("Total Count") +
```

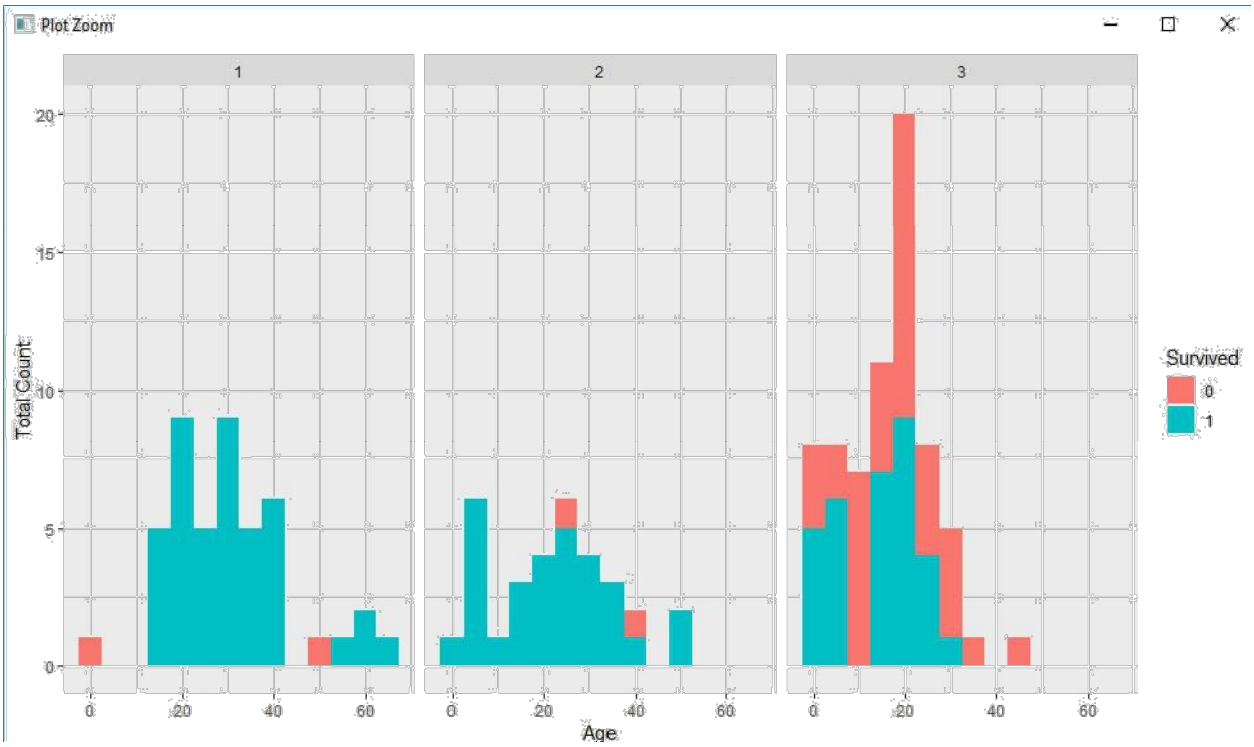
```
  labs(fill = "Survived")
```




```
  summary(data.combined$Age)
summary(data.combined[1:891,"Age"])
  ggplot(data.combined[1:891,],aes(x=Age,fill=Survived))+
   facet_wrap(~Sex+Pclass)+
   geom_histogram(binwidth = 10)+
   ylab("total count")+
   xlab("age")
```

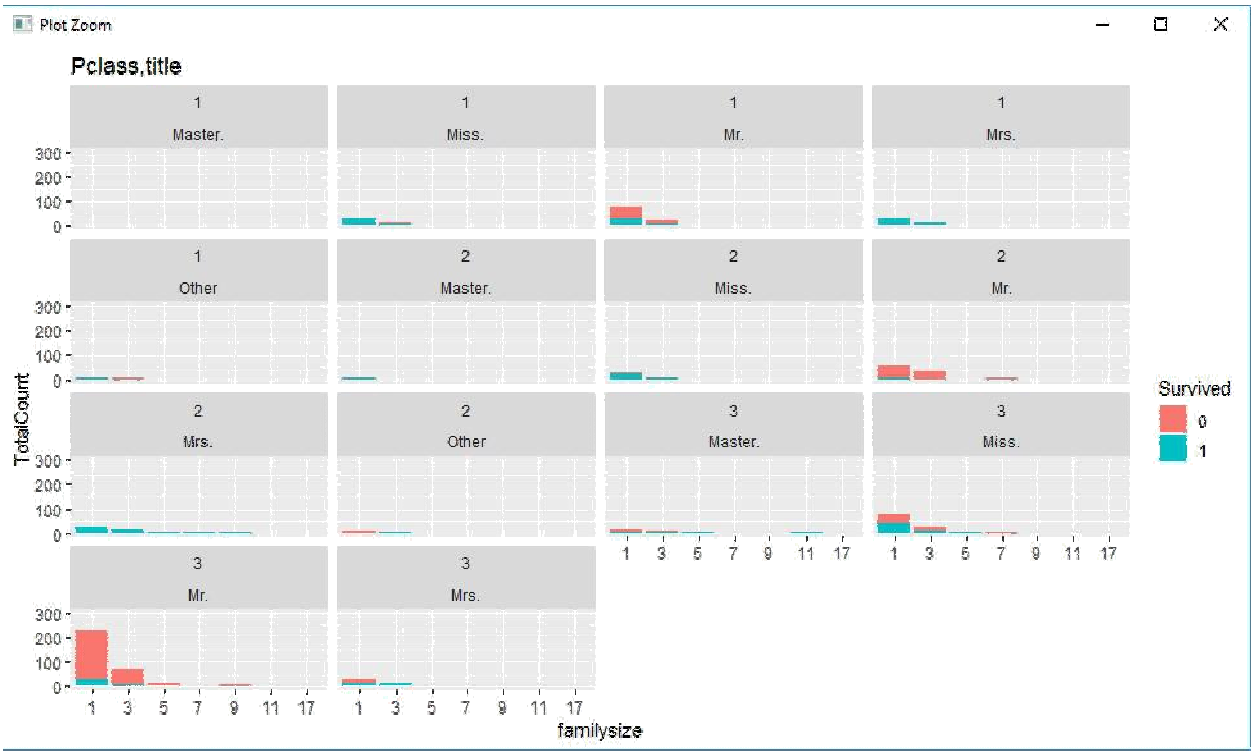



```
boys <- data.combined[which(data.combined$title ==
"Master."),] summary(boys$Age)
misses <- data.combined[which(data.combined$title ==
"Miss."),] summary(misses$Age)
ggplot(misses[misses$Survived !=
"None",],aes(x=Age,fill=Survived))+
 facet_wrap(~Pclass) +
 geom_histogram(binwidth = 5) +
 xlab("Age")+
 ylab("Total Count")
```




```
misses.alone <- misses[which(misses$SibSp==0
& misses$Parch==0),]

summary(misses.alone$Age)

length(which(misses.alone$Age<=14.5))

summary(data.combined$SibSp)

length(unique(data.combined$SibSp))

ggplot(data.combined[1:891,],aes(x=SibSp,fill=Survived))+
 facet_wrap(~Pclass+title)+
 geom_histogram(binwidth = 1)+
 ggtitle("Pclass,title")+
 xlab("SibSp")+
 ylab("TotalCount")+
```

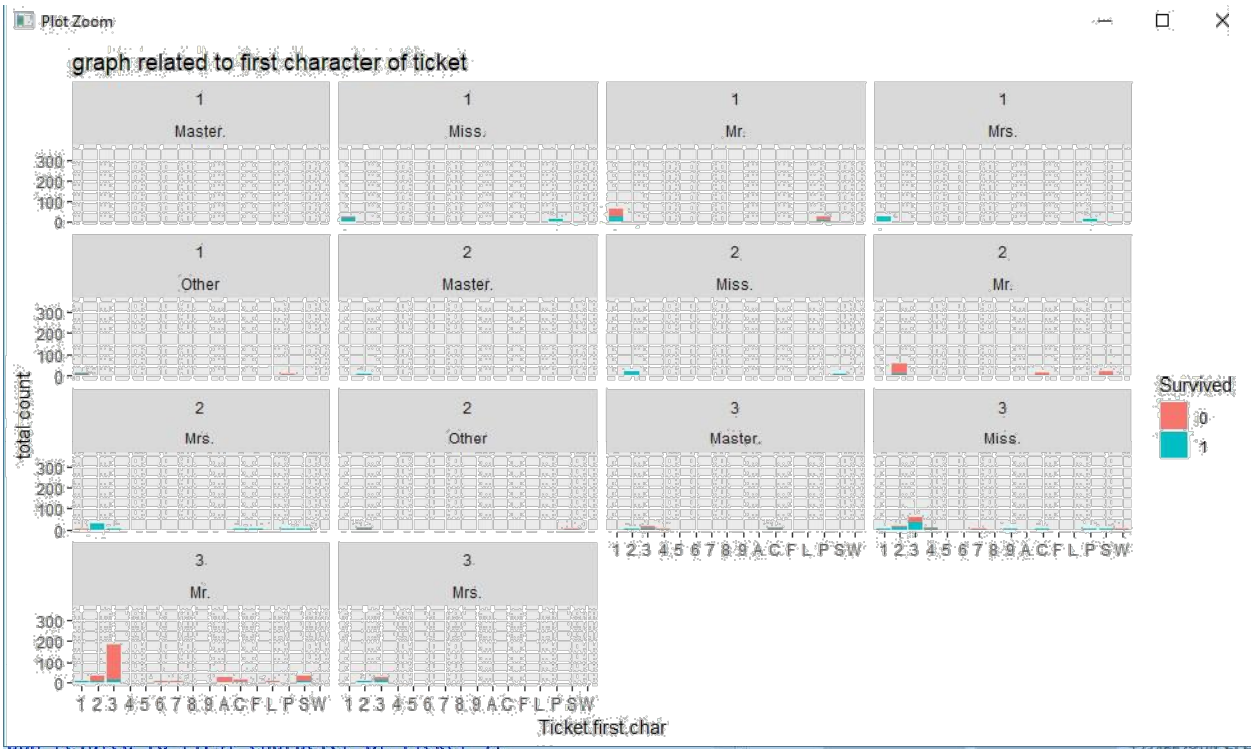
```
ylim(0,300)+
labs(fill="Survived")


data.combined$Parch <- as.factor(data.combined$Parch)
data.combined$parch <- as.factor(data.combined$parch)
ggplot(data.combined[1:891,], aes(x = Parch, fill =
Survived)) + geom_bar() +
 facet_wrap(~Pclass + title) +
 ggtitle("Pclass, Title") +
 xlab("ParCh") +
 ylab("Total Count") +
 ylim(0,300) +
 labs(fill = "Survived")
```



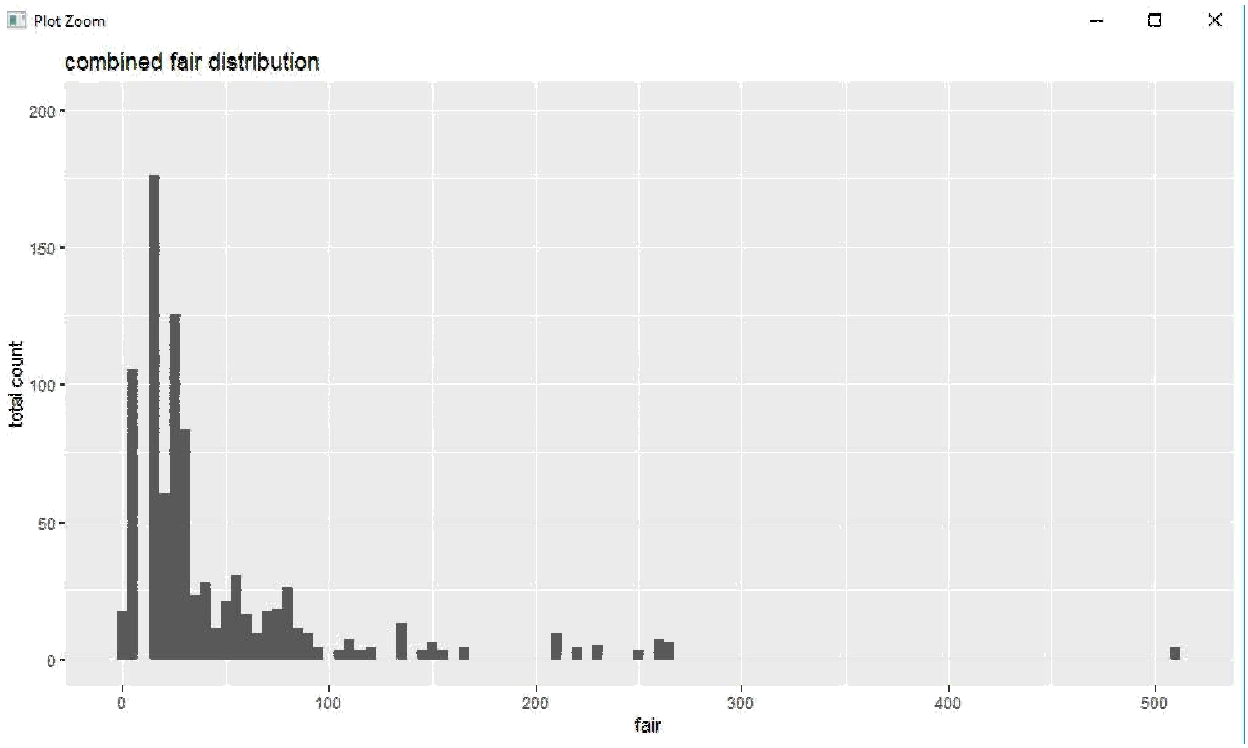

```
temp.sibsp<-c(test$SibSp,train$SibSp)

temp.parch<-c(test$SibSp,train$SibSp)

data.combined$familysize<-as.factor(temp.parch+temp.sibsp+1)

ggplot(data.combined[1:891,],aes(x=familysize,fill=Survived))+

 facet_wrap(~Pclass+title)+

 geom_bar()+

 ggtitle("Pclass,title")+

 xlab("familysize")+

 ylab("TotalCount")+

 ylim(0,300)+

 labs(fill="Survived")
```

```
str(data.combined$Ticket)


data.combined$Ticket<-as.character(data.combined$Ticket)

data.combined$Ticket[1:20]

Ticket.first.char<-ifelse(data.combined$Ticket=='',
',substr(data.combined$Ticket,1,1))

unique(Ticket.first.char)

data.combined$Ticket.first.char<- as.factor(Ticket.first.char)

ggplot(data.combined[1:891,],aes(x=Ticket.first.char,fill=Survived))
+
 geom_bar()+
 ggtitle("graph related to first character of ticket")+
 xlab("Ticket.first.char")+
 ylab("total count")+
 ylim(0,350)+
 labs(fill="Survived")
ggplot(data.combined[1:891,],aes(x=Ticket.first.char,fill=Survived))
+
 geom_bar()+
 facet_wrap(~Pclass)+
```

```
 ggtitle("Pclass")+
 xlab("Ticket.first.char")+
 ylab("total count")+
 ylim(0,350)+
 labs(fill="Survived")
ggplot(data.combined[1:891,],aes(x=Ticket.first.char,fill=Survived))
+
 geom_bar()+
 facet_wrap(~Pclass+title)+
 ggtitle("graph related to first character of ticket")+
 xlab("Ticket.first.char")+
 ylab("total count")+
 ylim(0,350)+
 labs(fill="Survived")
```



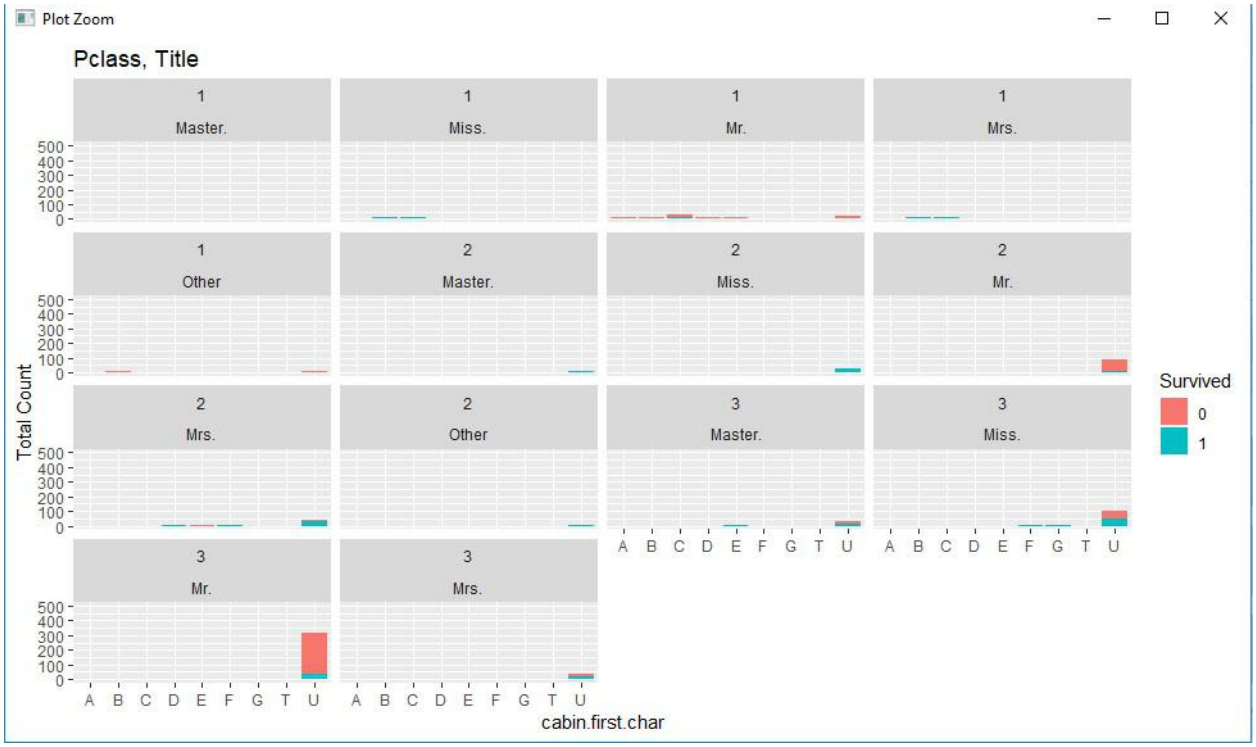

```
summary(data.combined$Fare)

length(unique(data.combined$Fare))

ggplot(data.combined,aes(x=Fare))+
  geom_histogram(binwidth = 5)+
  ggtitle("combined fair distribution")+
  xlab("fair")+
  ylab("total count")+
  ylim(0,200)
```




```
ggplot(data.combined[1:891,],aes(x=Fare,fill=Survived))+
  geom_histogram(binwidth = 5)+
  facet_wrap(~Pclass+title)+
  ggtitle("pclass,title")+
  xlab("fare")+
  ylab("total count")+
  ylim(0,350)+
  labs(fill="Survived")
str(data.combined$Cabin)
data.combined$Cabin<-as.character(data.combined$Cabin)
data.combined$Cabin[1:100]
```

```
data.combined[which(data.combined$Cabin == ""),

"Cabin"] <- "U" data.combined$Cabin[1:100]

cabin.first.char <- as.factor(substr(data.combined$Cabin, 1, 1))

str(cabin.first.char)

levels(cabin.first.char)

data.combined$cabin.first.char <- cabin.first.char

ggplot(data.combined[1:891,], aes(x =
cabin.first.char, fill = Survived)) +

 geom_bar() +

 ggtitle("Survivability by cabin.first.char")

 + xlab("cabin.first.char") + ylab("Total

 Count") +

 ylim(0,750) +

 labs(fill = "Survived")

ggplot(data.combined[1:891,], aes(x =
cabin.first.char, fill = Survived)) +

 geom_bar() +

 facet_wrap(~Pclass) +

 ggtitle("Survivability by cabin.first.char") +

 xlab("Pclass") +

 ylab("Total Count") +
```

```
  ylim(0,500) +
  labs(fill = "Survived")
ggplot(data.combined[1:891,], aes(x =
cabin.first.char, fill = Survived)) +
  geom_bar() +
  facet_wrap(~Pclass + title) +
  ggtitle("Pclass, Title") +
  xlab("cabin.first.char") +
  ylab("Total Count") +
  ylim(0,500) +
  labs(fill = "Survived")
```

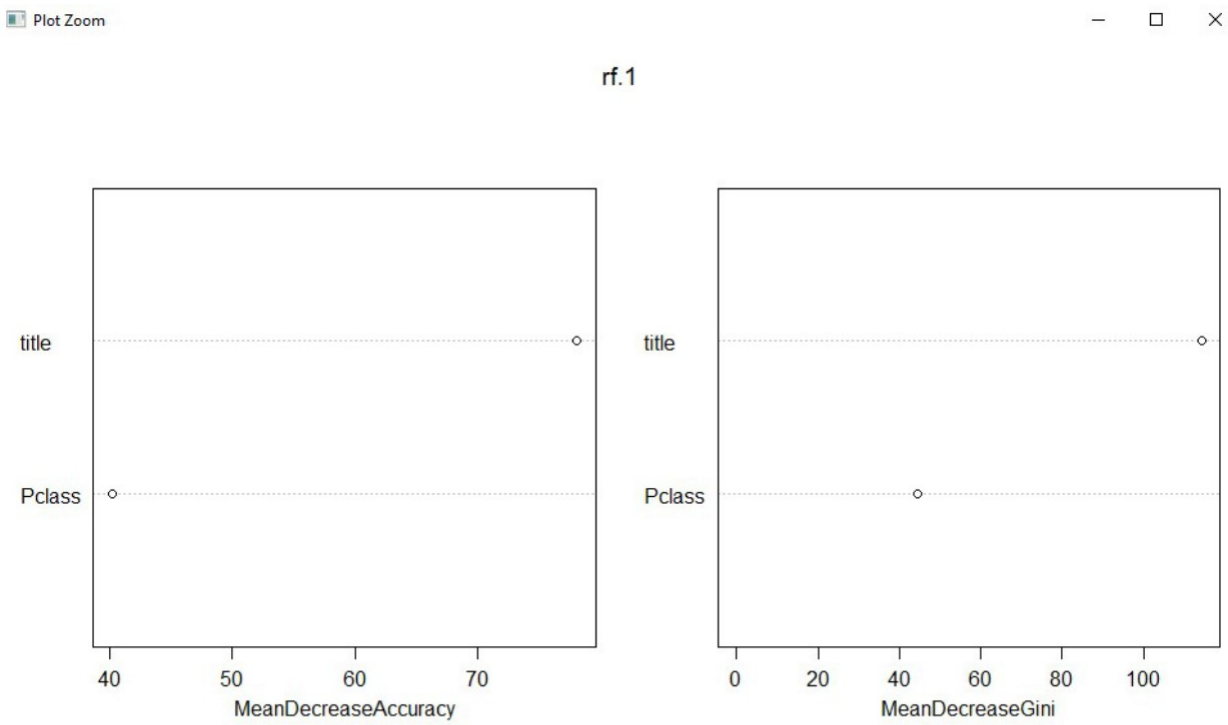


ALGORITHM

After seeing all the graph we can say that title,Pclass,family size etc have much correspondence to the survival rate than any other.to find out the most effective combination we will use random forest algorithm and will be seeing OOB value (OUT OF BOX) which says the error percentage. From that we will be make a conclusion.

CODE

```
library(randomForest)

rf.train<-data.combined[1:891,c("Pclass","title")]

rf.label=as.factor(train$Survived)

set.seed(1234)

rf.1<-randomForest(x=rf.train,y=rf.label,importance =
TRUE,ntree = 1000)

rf.1
```




```
        OOB estimate of  error rate: 20.99%
Confusion matrix:
    0   1 class.error
0 536  13  0.02367942
1 174 168  0.50877193
> varImpPlot(rf.1)
```

varImpPlot(rf.1)

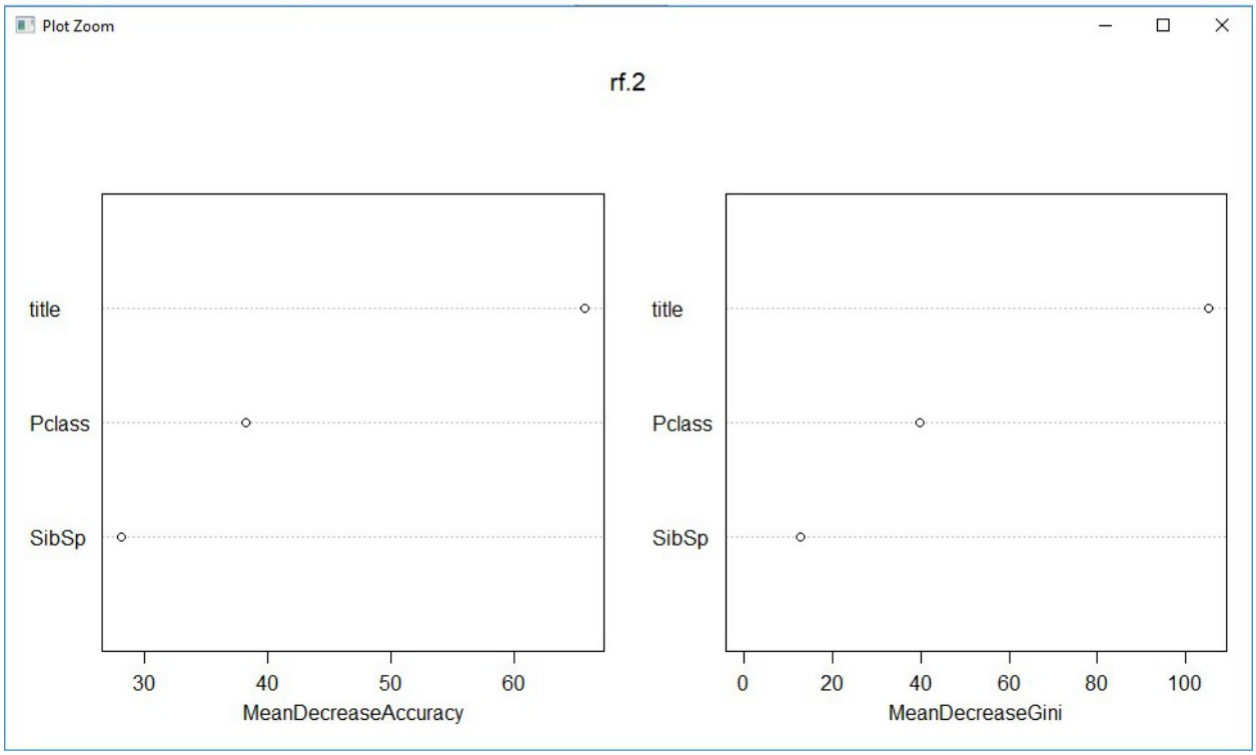
rf.train.2<-data.combined[1:891,c("Pclass","title","SibSp")]

rf.2<-rf.1<-randomForest(x=rf.train.2,y=rf.label,importance = TRUE,ntree = 1000)

rf.2

varImpPlot(rf.2)

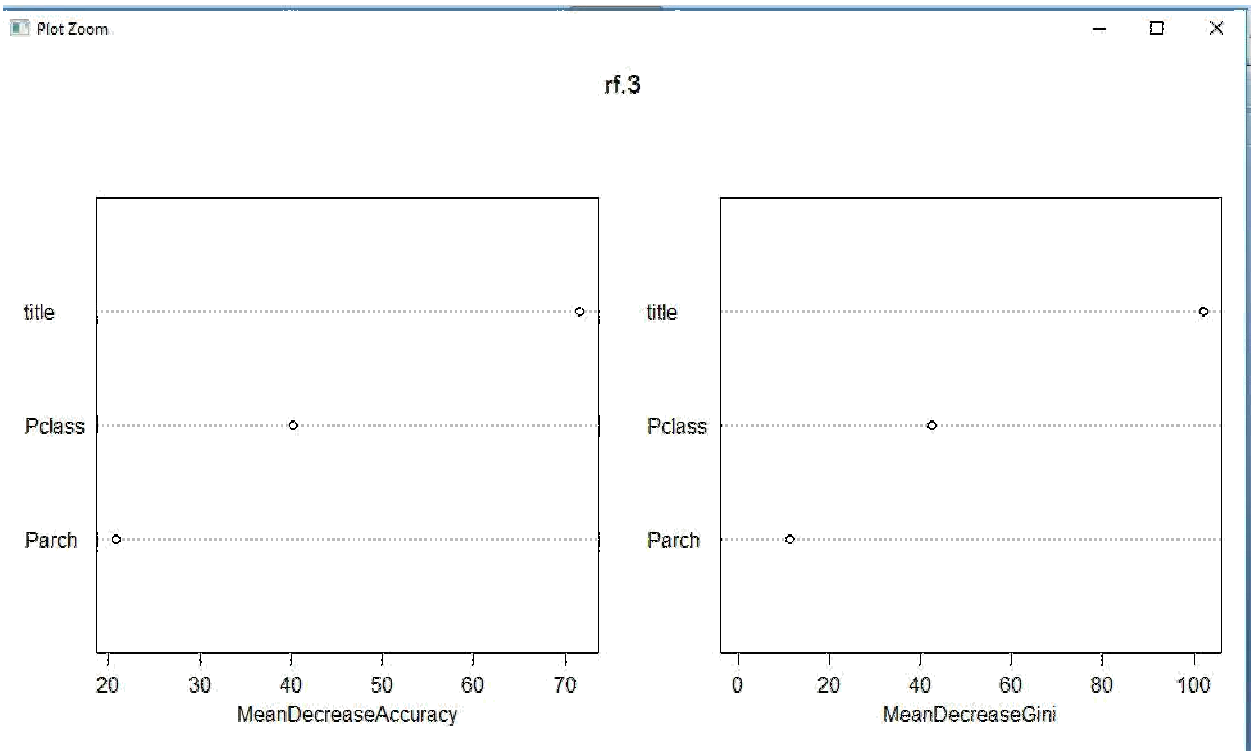
```
        OOB estimate of  error rate: 19.64%
Confusion matrix:
    0   1 class.error
0 495  54  0.09836066
1 121 221  0.35380117
```



rf.train.3<-data.combined[1:891,c("Pclass","title","Parch")] rf.3<-randomForest(x=rf.train.3,y=rf.label,importance = TRUE,ntree = 1000)

rf.3

```
        OOB estimate of  error rate: 19.75%
Confusion matrix:
    0   1 class.error
0 497  52  0.09471767
1 124 218  0.36257310
```



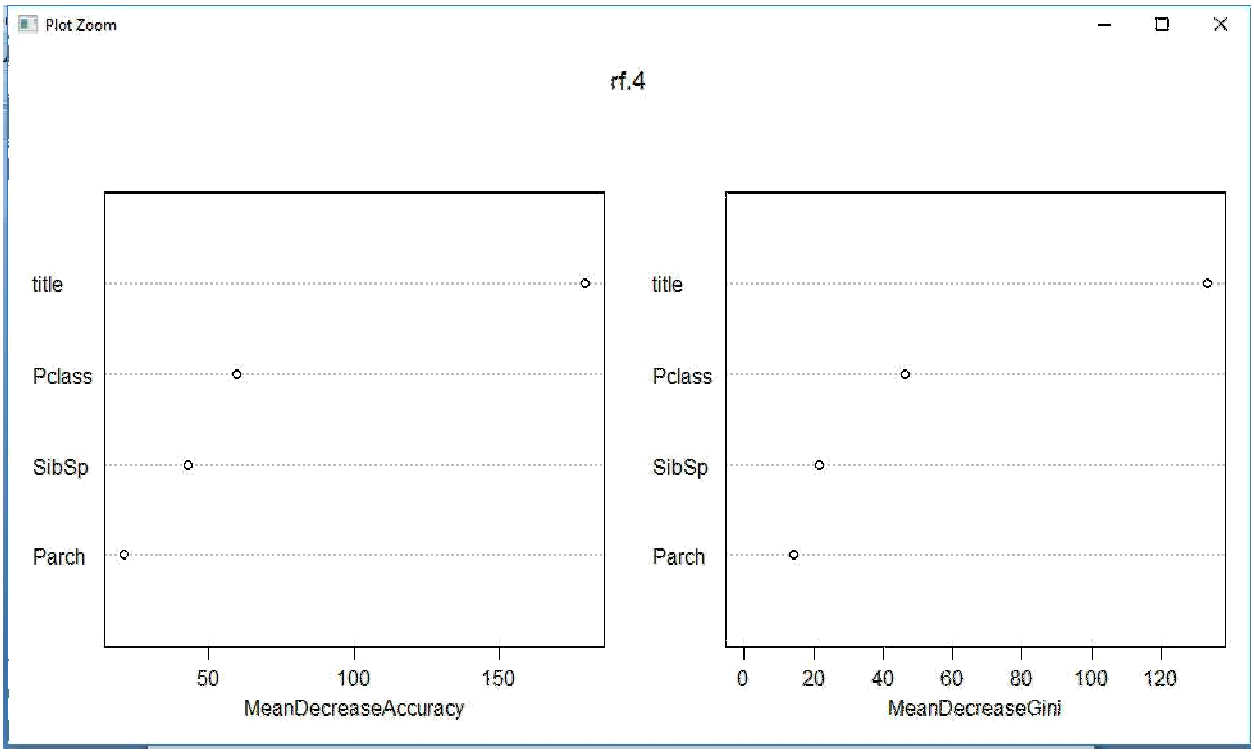
varImpPlot(rf.3)

rf.train.4<-data.combined[1:891,c("Pclass","title","SibSp","Parch")]

rf.4<-randomForest(x=rf.train.4,y=rf.label,importance = TRUE,ntree = 1000)

rf.4

```
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 2

        OOB estimate of  error rate: 18.74%
Confusion matrix:
    0   1 class.error
0 489  60   0.1092896
1 107 235   0.3128655
> varImpPlot(rf.4)
>
```



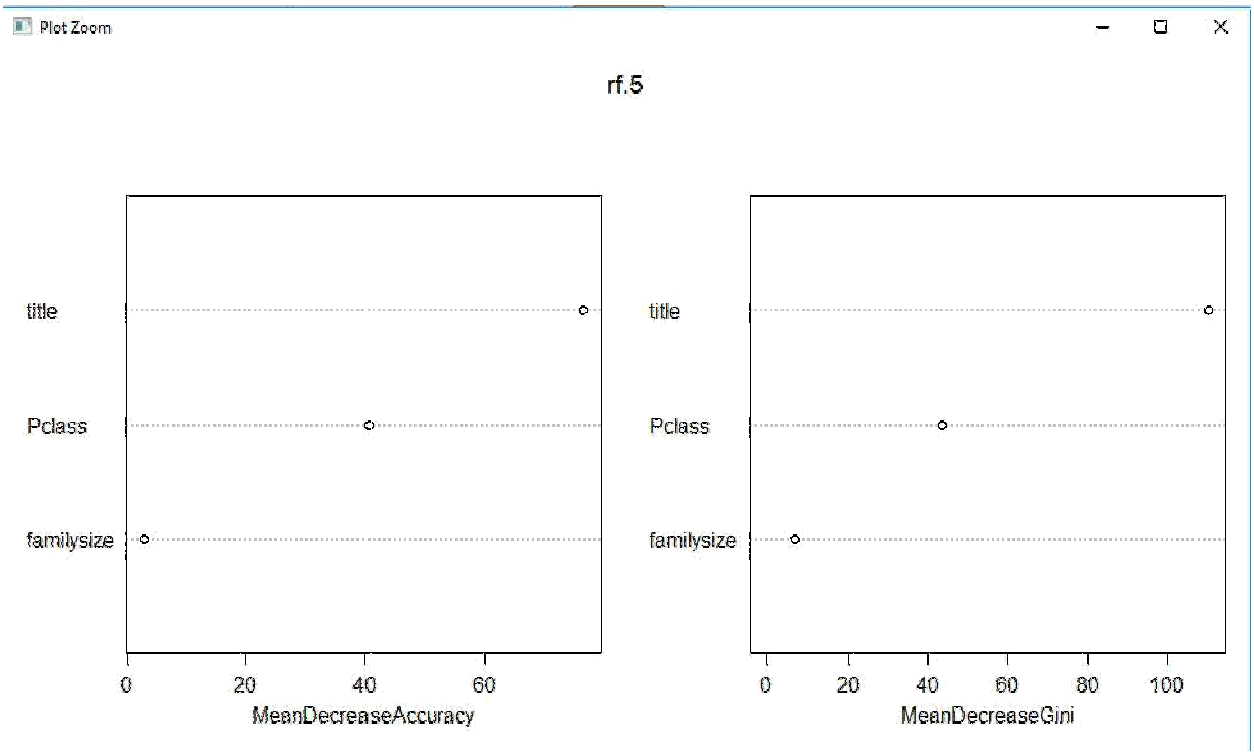
varImpPlot(rf.4)

rf.train.5<-data.combined[1:891,c("Pclass","title","familysize")]

rf.5<-randomForest(x=rf.train.5,y=rf.label,importance = TRUE,ntree = 1000)

rf.5

```
        OOB estimate of  error rate: 22.56%
Confusion matrix:
    0   1 class.error
0 504  45  0.08196721
1 156 186  0.45614035
```



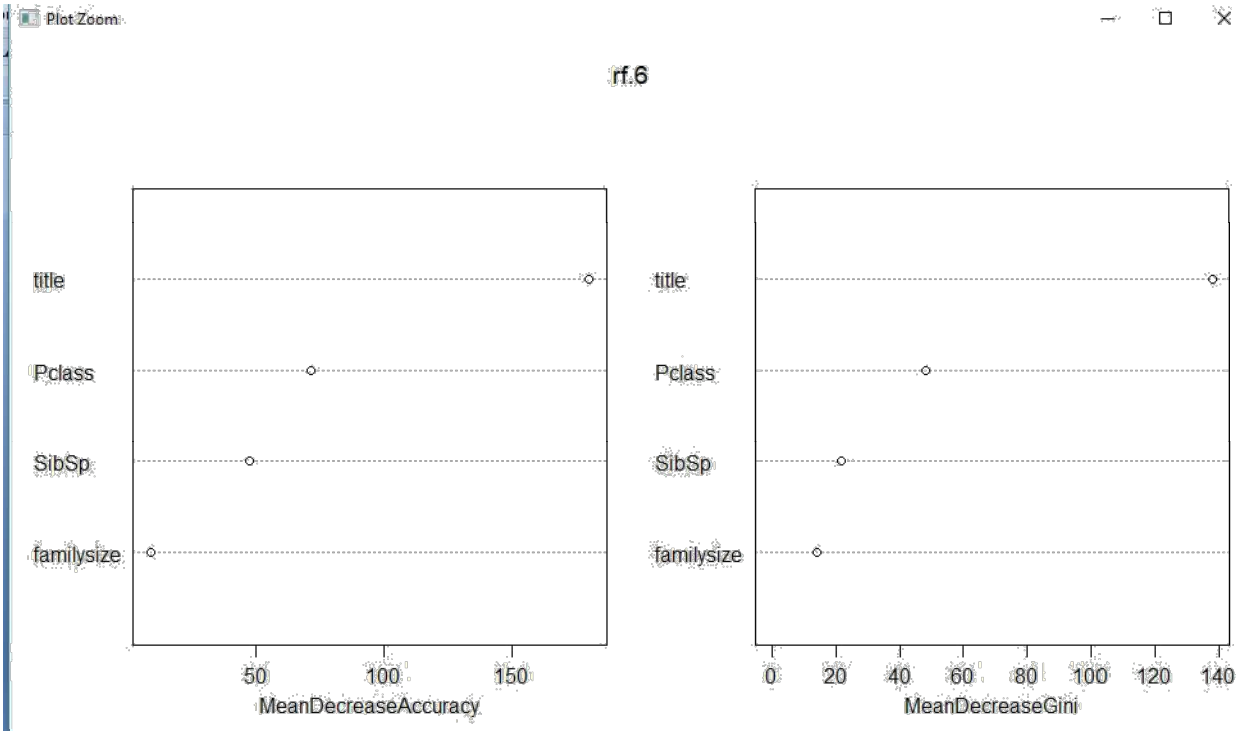
varImpPlot(rf.5)

rf.train.6<-data.combined[1:891,c("Pclass","title","SibSp","familysize")]

rf.6<-randomForest(x=rf.train.6,y=rf.label,importance = TRUE,ntree = 1000)

rf.6

```
        OOB estimate of  error rate: 18.52%
Confusion matrix:
    0   1 class.error
0 492  57   0.1038251
1 108 234   0.3157895
> varImpPlot(rf.6)
```



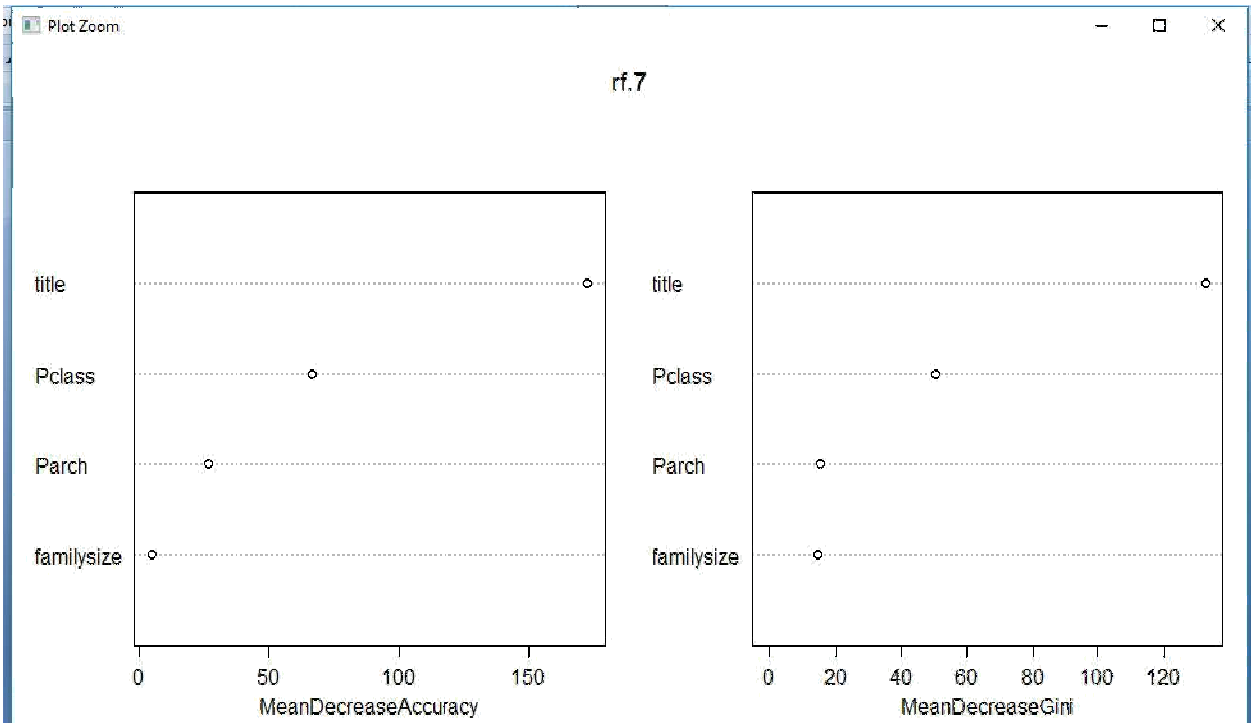
varImpPlot(rf.6)

rf.train.7<-
data.combined[1:891,c("Pclass","title","Parch","familysize")]

rf.7<-randomForest(x=rf.train.7,y=rf.label,importance = TRUE,ntree = 1000)

rf.7

```
        OOB estimate of  error rate: 19.75%
Confusion matrix:
    0   1 class.error
0 491  58   0.1056466
1 118 224   0.3450292
> varImpPlot(rf.7)
>
```



varImpPlot(rf.7)

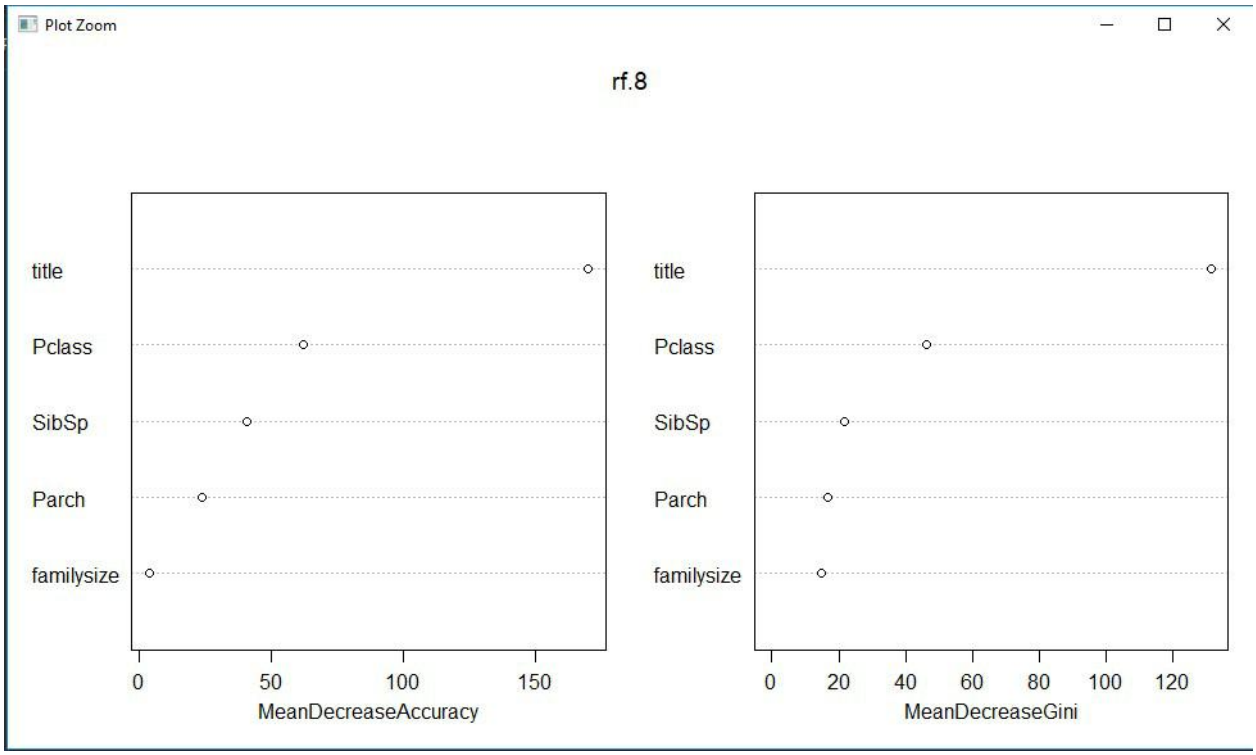
rf.train.8<-
data.combined[1:891,c("Pclass","title","SibSp","Parch","familysize")]

rf.8<-randomForest(x=rf.train.8,y=rf.label,importance = TRUE,ntree = 1000)

rf.8

```
        OOB estimate of  error rate: 19.75%
Confusion matrix:
    0   1 class.error
0 486  63   0.1147541
1 113 229   0.3304094
> varImpPlot(rf.8)
>
```

varImpPlot(rf.8)



CONCLUSION

By comparing the OOB of all and seeing the graph we can say that the best combination for survival rate will be ("Pclass","title","SibSp","familysize")

Since its OOB is 18.52%
