Group 9 - Titanic: Machine Learning from Disaster: Logistic Regress, Random Forest Classification Tree and XGBoost Model

Tian Tian, Xiang Fan, Sean Fan

6/26/2020

#set the directory of data folder

```
setwd("C:/Users/yuxia/OneDrive/Desktop/Kaggle")
```

#input datasets and take a look at the metadata & schema

```
train = read.csv("train.csv",stringsAsFactors=FALSE)
summary(train)
```

```
##   PassengerId       Survived          Pclass          Name
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000   Length:891
##  1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000   Class :character
##  Median :446.0   Median :0.0000   Median :3.000   Mode  :character
##  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
##
##      Sex                 Age            SibSp           Parch
##  Length:891         Min.   : 0.42   Min.   :0.000   Min.   :0.0000
##  Class :character   1st Qu.:20.12   1st Qu.:0.000   1st Qu.:0.0000
##  Mode  :character   Median :28.00   Median :0.000   Median :0.0000
##                     Mean   :29.70   Mean   :0.523   Mean   :0.3816
##                     3rd Qu.:38.00   3rd Qu.:1.000   3rd Qu.:0.0000
##                     Max.   :80.00   Max.   :8.000   Max.   :6.0000
##                     NA's   :177
##     Ticket               Fare           Cabin              Embarked
##  Length:891         Min.   :  0.00   Length:891         Length:891
##  Class :character   1st Qu.:  7.91   Class :character   Class :character
##  Mode  :character   Median : 14.45   Mode  :character   Mode  :character
##                     Mean   : 32.20
##                     3rd Qu.: 31.00
##                     Max.   :512.33
##
```

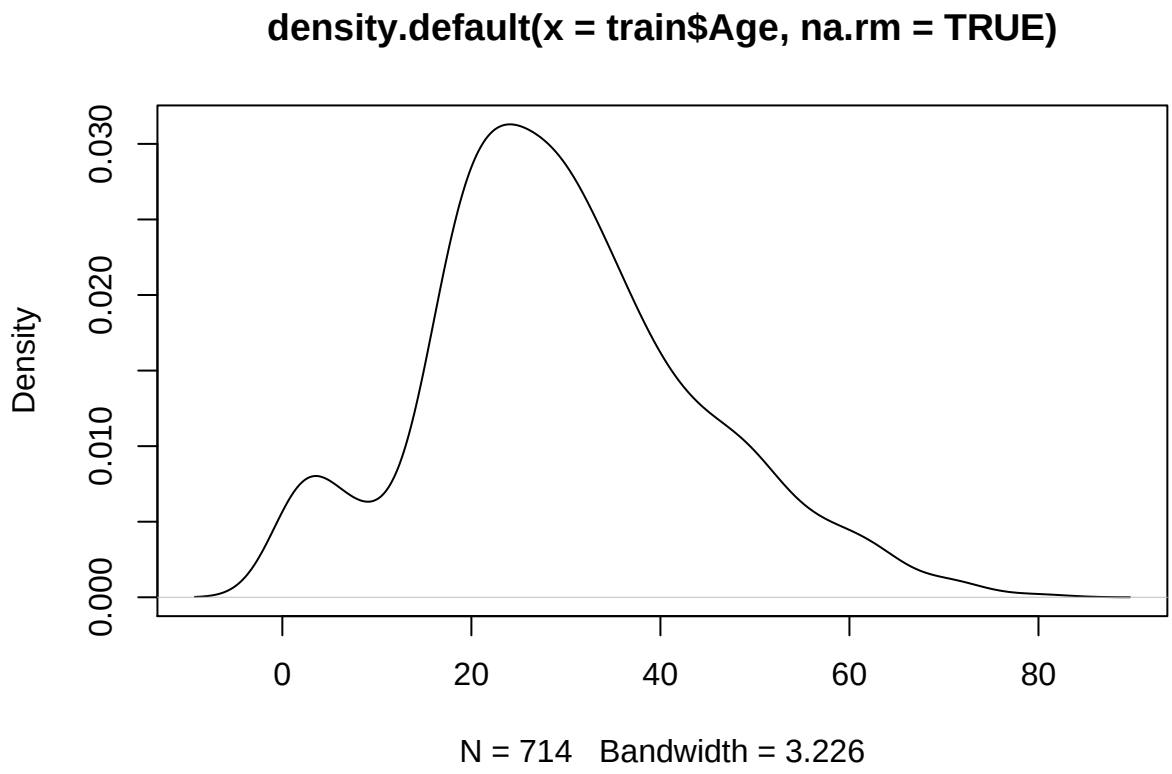
```
head(train,10)
```

```
##    PassengerId Survived Pclass
## 1            1        0      3
## 2            2        1      1
## 3            3        1      3
## 4            4        1      1
## 5            5        0      3
## 6            6        0      3
## 7            7        0      1
## 8            8        0      3
```

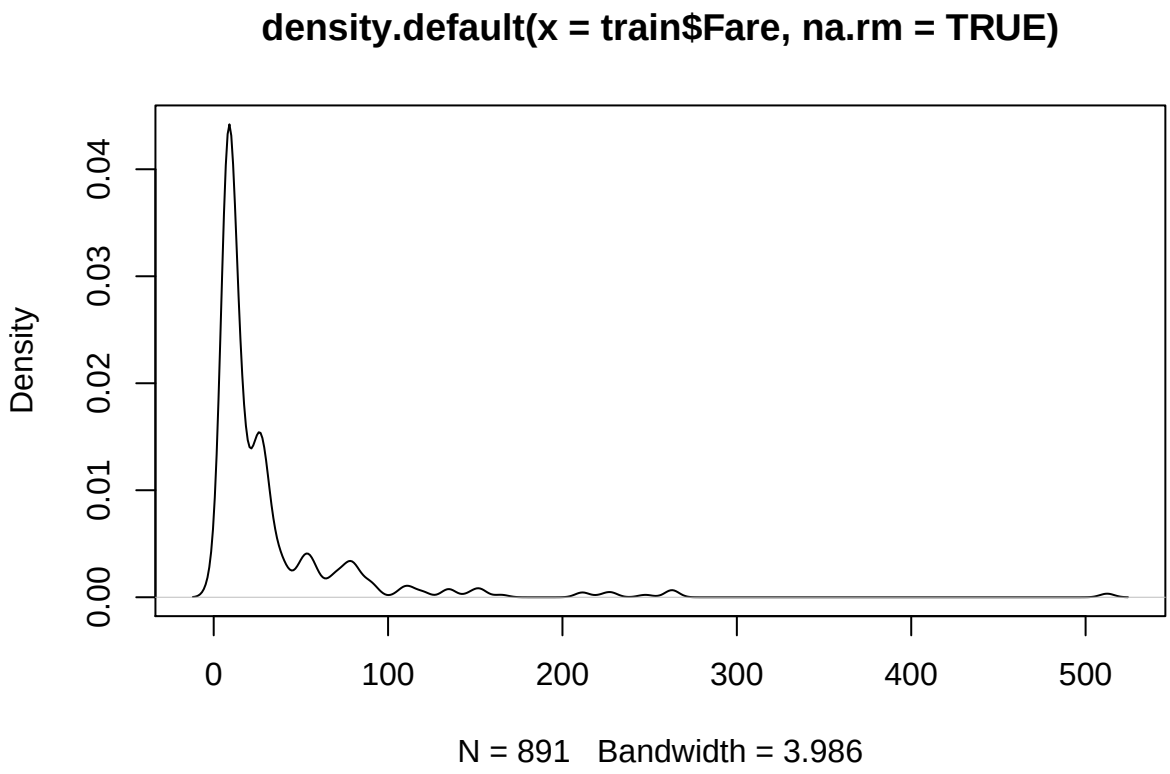
```
## 9            9        1      3
## 10          10        1      2
##                                                  Name    Sex Age SibSp
## 1                             Braund, Mr. Owen Harris   male  22     1
## 2  Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1
## 3                              Heikkinen, Miss. Laina female  26     0
## 4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1
## 5                            Allen, Mr. William Henry   male  35     0
## 6                                   Moran, Mr. James   male  NA     0
## 7                             McCarthy, Mr. Timothy J   male  54     0
## 8                      Palsson, Master. Gosta Leonard   male   2     3
## 9    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg) female  27     0
## 10                Nasser, Mrs. Nicholas (Adele Achem) female  14     1
##    Parch           Ticket    Fare Cabin Embarked
## 1      0        A/5 21171  7.2500              S
## 2      0         PC 17599 71.2833   C85        C
## 3      0 STON/O2. 3101282  7.9250              S
## 4      0           113803 53.1000  C123        S
## 5      0           373450  8.0500              S
## 6      0           330877  8.4583              Q
## 7      0            17463 51.8625   E46        S
## 8      1           349909 21.0750              S
## 9      2           347742 11.1333              S
## 10     0           237736 30.0708              C
```

#Making basic visualizations for data questions investigation.

```
par(mfrow=c(1,1))
plot(density(train$Age,na.rm=TRUE))
```

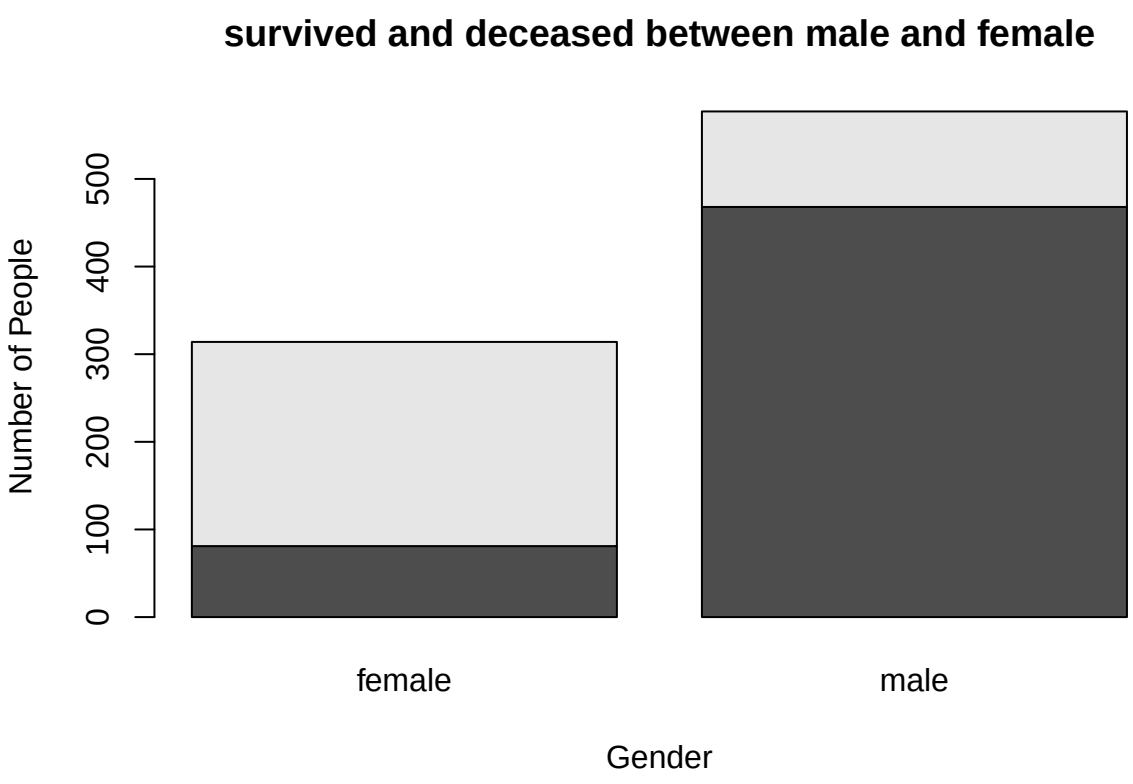



```
plot(density(train$Fare,na.rm=TRUE))
```



#Survival Rate by Sex Barplot

```
counts <- table(train$Survived, train$Sex)
barplot(counts, xlab = "Gender", ylab = "Number of People", main = "survived and deceased between male a
```




```
counts[2] / (counts[1] + counts[2])
```

```
## [1] 0.7420382
```

```
counts[4] / (counts[3] + counts[4])
```

```
## [1] 0.1889081
```

```
#74.2% of women survived versus 18.9% of men.
```

#Survival Rate by Passenger Class Barplot

```
Pclass_survival <- table(train$Survived, train$Pclass)
barplot(Pclass_survival, xlab="Cabin Class", ylab = "Number of People", main = "survived and deceased b
```


```
Pclass_survival[2] / (Pclass_survival[1]+Pclass_survival[2])
```

```
## [1] 0.6296296
```

```
Pclass_survival[4] / (Pclass_survival[3]+Pclass_survival[4])
```

```
## [1] 0.4728261
```

```
Pclass_survival[6] / (Pclass_survival[5]+Pclass_survival[6])
```

```
## [1] 0.2423625
```

```
#It seems like the Pclass column might also be informative in survival prediction as the survival rate
```

#Some variables have many 'missing' values, like Age, so we will remove these variable that we do not use for the Model:PassengerID, Ticket, Fare, Cabin, and Embarked.

```
train = train[-c(1, 9, 11, 12)]
```

#Replacing Gender variable (Male/Female) with a Dummy Variable (0/1). Additionally, we need to replace qualitative variables (such as gender) into quantitative variables (0 for male, 1 for female etc) in order to fit our model.

```
train$Sex = gsub('female', 1, train$Sex)
train$Sex = gsub('male', 0, train$Sex)
```

#Making Inferences on Misiing Age Values: We assuming that Mrs. will be comparetively older than Ms.; so we will group people with the same titles in closer age. #We also replace the name by her/his prefix ti allows for standardization

```
i_mr = grep("Mr.", train$Name, fixed = TRUE)
i_mrs = grep("Mrs.", train$Name, fixed = TRUE)
i_miss = grep("Miss.", train$Name, fixed = TRUE)
i_master = grep("Master.", train$Name, fixed = TRUE)

for(i in i_mr){
  train$Title[i] = "Mr"
}
for(i in i_mrs){
  train$Title[i] = "Mrs"
}
for(i in i_miss){
  train$Title[i] = "Miss"
}
for(i in i_master){
  train$Title[i] = "Master"
}
```

#Making Inference on Missing Age Values: Inputting Title-group averages; We replace the missing ages with their respective title-group average. This means that if we have a missing age entry for a man named Mr. Bond, we substitute his age for the average age for all passenger with the title Mr. Similarly for Master, Miss, Mrs, and Dr. We then write a for loop that goes through the entire Train data set and checks if the age value is missing. If it is, we assign it according to the surname of the observation. This code snippet is a bit complicated.

```
mr_age = round(mean(train$Age[train$Title=="Mr"], na.rm=TRUE), digits=2)
mrs_age = round(mean(train$Age[train$Title=="Mrs"], na.rm=TRUE), digits=2)
miss_age = round(mean(train$Age[train$Title=="Miss"], na.rm=TRUE), digits=2)
master_age = round(mean(train$Age[train$Title=="Master"], na.rm=TRUE), digits=2)


for (i in 1:nrow(train)){
  if (is.na(train[i, 5])){
    if (train$Title[i]=="Mr"){
      train$Age[i] = mr_age
    }
    else if (train$Title[i]=="Mrs"){
      train$Age[i] = mrs_age
    }
    else if (train$Title[i]=="Miss"){
      train$Age[i] = miss_age
    }
    else if (train$Title[i]=="Master"){
      train$Age[i] = master_age
    }
    else {
      print("Uncaught Title")
    }
  }
}
```

#Same process for erase missing values in `Fare` as we filled the missing value in column `Age`.

```
first_class_fare = round(mean(train$Fare[train$Pclass == 1],na.rm = TRUE), digits = 2)
second_class_fare = round(mean(train$Fare[train$Pclass == 2],na.rm = TRUE), digits = 2)
```

```
thrid_class_fare = round(mean(train$Fare[train$Pclass == 3],na.rm = TRUE),digits = 2)

for (i in 1:nrow(train)){
  if(is.na(train[i,8])){
    if(train$Pclass[i] == 1){
      train$Fare[i] = first_class_fare
    }else if(train$Pclass[i] == 2){
      train$Fare[i] = second_class_fare
    }else if(train$Pclass[i] == 3){
      train$Fare[i] = thrid_class_fare
    }else{
      print("Unknown Fare")
    }
  }
}
```

#Creating New Variables to Strengthen Our Model #By creating new variables we may be able to predict the survival of the passengers even more closely. This part of the walkthrough specifically includes three variables which we found to help our model.

#vairable 1: child #We create a column title "Child", and value "1" as passenger under the age of 12 and "2" otherwise.

```
train$Child <- NA
for (i in 1:nrow(train)){
  if (train$Age[i] <= 12) {
    train$Child[i] = 1
  }
  else {
    train$Child[i] = 0
  }
}
```

#variable 2: Family #We create a column title "Family" which count the total number of family size for each apssenger by summing up Sibiling/Spouses and Parents/Children, +1 means plus the passenger herself/himself.

```
train$Family = NA
for (i in 1:nrow(train)) {
  x = train$SibSp[i]
  y = train$Parch[i]
  train$Family[i] = x+y+1
}
```

#variable 3: Mother #We create a column to title the passenger if she is a mother or not, we will use "if" to decided whether the passenger is married and have "Parch" greater than 1. Mother = 1 means passenger is a mother, vice versa.

```
train$Mother = NA
for (i in 1:nrow(train)){
  if (train$Title[i] == "Mrs" & train$Parch[i] > 0){
    train$Mother[i] = 1
  }
  else {
    train$Mother[i] = 0
  }
```

```
}
```

Clean the Test dataset #We need to repeat all steps we have done on Train dataset to Test dataset so we can have the same state, the only difference between two datasets are the column amount. We need to becareful when we use the index of column.

```
test = read.csv("test.csv", stringsAsFactors=FALSE)
test_cp = test
PassengerID = test_cp$PassengerId
head(test,10)
```

```
##    PassengerId Pclass                                          Name    Sex
## 1          892      3                              Kelly, Mr. James   male
## 2          893      3              Wilkes, Mrs. James (Ellen Needs) female
## 3          894      2                     Myles, Mr. Thomas Francis   male
## 4          895      3                              Wirz, Mr. Albert   male
## 5          896      3 Hirvonen, Mrs. Alexander (Helga E Lindqvist) female
## 6          897      3                    Svensson, Mr. Johan Cervin   male
## 7          898      3                          Connolly, Miss. Kate female
## 8          899      2                  Caldwell, Mr. Albert Francis   male
## 9          900      3    Abrahim, Mrs. Joseph (Sophie Halaut Easu) female
## 10         901      3                        Davies, Mr. John Samuel   male
##     Age SibSp Parch    Ticket    Fare Cabin Embarked
## 1  34.5     0     0    330911  7.8292              Q
## 2  47.0     1     0    363272  7.0000              S
## 3  62.0     0     0    240276  9.6875              Q
## 4  27.0     0     0    315154  8.6625              S
## 5  22.0     1     1   3101298 12.2875              S
## 6  14.0     0     0      7538  9.2250              S
## 7  30.0     0     0    330972  7.6292              Q
## 8  26.0     1     1    248738 29.0000              S
## 9  18.0     0     0      2657  7.2292              C
## 10 21.0     2     0 A/4 48871 24.1500              S
```

#Remove useless input variables.

```
test = test[-c(1, 8,10,11)]

test$Sex = gsub("female", 1, test$Sex)
test$Sex = gsub("male", 0, test$Sex)
```

#Replace passenger's name with prefix.

```
itest_master = grep("Master.",test$Name, fixed = TRUE)
itest_miss = grep("Miss.", test$Name, fixed = TRUE)
itest_mrs = grep("Mrs.", test$Name, fixed = TRUE)
itest_mr = grep("Mr.", test$Name, fixed = TRUE)
itest_dr = grep("Dr.", test$Name, fixed = TRUE)

for(i in itest_master) {
  test$Title[i] = "Master"
}
for(i in itest_miss) {
  test$Title[i] = "Miss"
}
for(i in itest_mrs) {
```

```
  test$Title[i] = "Mrs"
}
for(i in itest_mr) {
  test$Title[i] = "Mr"
}
for(i in itest_dr) {
  test$Title[i] = "Dr"
}
```

#Fill Age's missing value with group average ages.

```
test_master_age = round(mean(test$Age[test$Title == "Master"], na.rm = TRUE), digits = 2)
test_miss_age = round(mean(test$Age[test$Title == "Miss"], na.rm = TRUE), digits =2)
test_mrs_age = round(mean(test$Age[test$Title == "Mrs"], na.rm = TRUE), digits = 2)
test_mr_age = round(mean(test$Age[test$Title == "Mr"], na.rm = TRUE), digits = 2)
test_dr_age = round(mean(test$Age[test$Title == "Dr"], na.rm = TRUE), digits = 2)

for (i in 1:nrow(test)) {
  if (is.na(test$Age[i]) & !is.na(test$Title[i])) {
    if (test$Title[i] == "Master") {
      test$Age[i] = test_master_age
    } else if (test$Title[i] == "Miss") {
      test$Age[i] = test_miss_age
    } else if (test$Title[i] == "Mrs") {
      test$Age[i] = test_mrs_age
    } else if (test$Title[i] == "Mr") {
      test$Age[i] = test_mr_age
    } else {
      next()
    }
  }
}
```

#Manually check if there is still NA

```
test[is.na(test$Age),]
```

```
##    Pclass                     Name Sex Age SibSp Parch Fare Title
## 89      3 O'Donoghue, Ms. Bridget   1  NA     0     0 7.75  <NA>
```

#Manually fill the missing information

```
test$Title[89] = "Ms"
test$Age[89] = test_miss_age
```

#New variable 1:Child

```
test["Child"] = NA

for (i in 1:nrow(test)) {
  if (test$Age[i] <= 12) {
    test$Child[i] = 1
  } else {
    test$Child[i] = 0
  }
}
```

#New variable 2:Family

```
test["Family"] = NA

for(i in 1:nrow(test)) {
  test$Family[i] = test$SibSp[i] + test$Parch[i] + 1
}
```

#New variable 3:Mother

```
test["Mother"] = NA

for (i in 1:nrow(test)) {
  if (!is.na(test$Title[i]) & test$Title[i] == "Mrs" & test$Parch[i] > 0) {
    test$Mother[i] = 1
  } else {
    test$Mother[i] = 0
  }
}
```

#Take a look at the manipulated Test dataset

```
head(test)
```

```
##   Pclass                                          Name Sex  Age SibSp Parch
## 1      3                              Kelly, Mr. James   0 34.5     0     0
## 2      3              Wilkes, Mrs. James (Ellen Needs)   1 47.0     1     0
## 3      2                     Myles, Mr. Thomas Francis   0 62.0     0     0
## 4      3                              Wirz, Mr. Albert   0 27.0     0     0
## 5      3 Hirvonen, Mrs. Alexander (Helga E Lindqvist)   1 22.0     1     1
## 6      3                    Svensson, Mr. Johan Cervin   0 14.0     0     0
##      Fare Title Child Family Mother
## 1  7.8292    Mr     0      1      0
## 2  7.0000   Mrs     0      2      0
## 3  9.6875    Mr     0      1      0
## 4  8.6625    Mr     0      1      0
## 5 12.2875   Mrs     0      3      1
## 6  9.2250    Mr     0      1      0
```

#Output cleaned Train and Test datasets for future use.

```
# write.csv(train, file = "train_clean.csv", row.names = FALSE)
# write.csv(test, file = "test_clean.csv", row.names = FALSE)
```

GLM #train a simple Logistic Regression in GLM model

```
glm.fit <- glm(Survived ~ Pclass + Sex + Age + Child + Sex*Pclass + Family + Mother, family = binomial,
```

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

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age + Child + Sex * Pclass +
##     Family + Mother, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4273  -0.5934  -0.4668   0.4349   2.4625
##
```

```
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.903838   0.535096   3.558 0.000374 ***
## Pclass      -0.875047   0.140033  -6.249 4.13e-10 ***
## Sex1         6.058403   0.882682   6.864 6.71e-12 ***
## Age         -0.028360   0.009757  -2.907 0.003654 **
## Child        1.936235   0.478069   4.050 5.12e-05 ***
## Family      -0.441954   0.095438  -4.631 3.64e-06 ***
## Mother       0.997302   0.461990   2.159 0.030873 *
## Pclass:Sex1 -1.328563   0.323093  -4.112 3.92e-05 ***
## ---
## 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:  743.71  on 883  degrees of freedom
## AIC: 759.71
##
## Number of Fisher Scoring iterations: 6
```

```
nrow(test)
```

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

#Make prediction based on the Test dataset without known result; Purpose: Kaggle Compete

```
glm.probs=predict(glm.fit,test,type="response")
glm.pred=rep(0,418)
glm.pred[glm.probs>0.5]=1
```

#Output submission csv file

```
kaggle.sub1 <- cbind(PassengerID, glm.pred)
colnames(kaggle.sub1) <- c("PassengerId", "Survived")
write.csv(kaggle.sub1, file = "submission1.csv", row.names=FALSE)
```

Random Forest #Training a classification tree

```
#install.packages("randomForest")
#install.packages("gbm")
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.3
```

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

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

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.6.3
```

```
## Loaded gbm 2.1.5
```

```
library(MASS)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

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

```
## Loading required package: ggplot2
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
library(rpart)
class.tree = rpart(formula=train$Survived~Pclass + Sex + Age + Child +SibSp +Parch + Family + Mother, da
```

```
printcp(class.tree)#Find out how the tree performs
```

```
##
## Classification tree:
## rpart(formula = train$Survived ~ Pclass + Sex + Age + Child +
##     SibSp + Parch + Family + Mother, data = train, method = "class")
##
## Variables actually used in tree construction:
## [1] Age    Child  Family Pclass Sex    SibSp
##
## Root node error: 342/891 = 0.38384
##
## n= 891
##
##         CP nsplit rel error  xerror     xstd
## 1 0.444444      0   1.00000 1.00000 0.042446
## 2 0.030702      1   0.55556 0.55556 0.035750
## 3 0.014620      5   0.43275 0.46199 0.033336
## 4 0.010000      6   0.41813 0.47368 0.033663
```

##Classification Tree Prunning

```
class.ptree<- prune(class.tree,
        cp=class.tree$cptable[which.min(class.tree$cptable[,"xerror"]),"CP"])# select the one having th
```

#Making prediction based on the result-unknown Test dataset

```
#Accuracy on Training Set
class.ptree_probs = predict(class.ptree,test,type="vector")
class.ptree_pred=rep(0,418)
class.ptree_pred[class.ptree_probs>1]=1
```

#Output submission CSV file

```
kaggle.sub_classficationtree <- cbind(PassengerID, class.ptree_pred)
colnames(kaggle.sub_classficationtree) <- c("PassengerId", "Survived")
write.csv(kaggle.sub_classficationtree, file = "submission2.csv", row.names=FALSE)
```

XGBoost #DATA MATRIX: set Survived column with all initialized with 0 for XGBoost successful prediction.

```
test["Survived"]=0
```

```
x.train <- model.matrix(Survived ~ Pclass + Sex + Age + Child +SibSp +Parch + Family + Mother,train)
y.train <- train$Survived
x.test <- model.matrix(Survived ~Pclass + Sex + Age + Child +SibSp +Parch + Family + Mother,test)
y.test <- test$Survived
```

```
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 3.6.3
```

```
# Transform the two data sets into xgb.Matrix
xgb.train <- xgb.DMatrix(data=x.train,label=y.train)
xgb.test <- xgb.DMatrix(data=x.test,label=y.test)
```

```
params <- list(booster = "gbtree", objective = "binary:logistic", eta=0.3, gamma=0, max_depth=6,min_chi
```

#Using Cross-Validation to calculate the best nround for this model.

```
set.seed(1)
xgbcv <- xgb.cv(params = params, data = xgb.train,nrounds = 200, nfold = 5, showsd = T, stratified = T,
```

```
## [1]  train-error:0.144496+0.005410   test-error:0.180675+0.012671
## Multiple eval metrics are present. Will use test_error for early stopping.
## Will train until test_error hasn't improved in 20 rounds.
##
## [11] train-error:0.124578+0.003432   test-error:0.180675+0.017663
## [21] train-error:0.115040+0.003576   test-error:0.181805+0.019831
## Stopping. Best iteration:
## [9]  train-error:0.123738+0.001407   test-error:0.176174+0.018663
```

#Training XGBoost model with nround = 9 and our Train dataset

```
#We get best iteration =9. The model returns lowest validation error at the 9th (nround) iteration.
xgb1 <- xgb.train (params = params, data = xgb.train, nrounds =9, print_every_n = 10,  maximize = F , ev
```

#Making prediction based on our Test data for accuracy

```
xgb_probs = predict(xgb1,xgb.test)
xgb_pred=rep(0,418)
xgb_pred[xgb_probs>0.5]=1
```

#Output the prediction as submission CSV file for Kaggle Compete.

```
kaggle.sub_xgb <- cbind(PassengerID, xgb_pred)
colnames(kaggle.sub_xgb) <- c("PassengerId", "Survived")
write.csv(kaggle.sub_xgb, file = "submission3.csv", row.names=FALSE)
```