Binary Classification – An Model Optimization Study for Titanic Disaster Example

This is a project for partial fulfillment
of
MH8111 Analytics and Software
at
Nanyang Technological University
by

G1701150H Guo Yang
G1700632B Wang Longling
G1700584F Yu Hefei
G1700687K Zheng Siya

NANYANG TECHNOLOGICAL UNIVERSITY

Submission Date:

31/08/2017

Abstract

The project took the example of Titanic Disaster in Kaggle. The Titanic Disaster competition required the contestants to use the train data set to build a predictive model and by employing this model, to predict the people who can survive in the test data set.

This project aims to get higher prediction score via a thorough optimization study. The optimization study includes feature engineering, choice of features, choice of algorithms, Box-cox transformation, tuning of an algorithm.

The final code for this study is 0.80383, which is within the 15% percentile of whole competitors' pool. The key concept delivered by this project is not how to get the highest score in the competition. Since the death was determined by a lot of factors, a model build from many features was not reliable to predict. Getting an absolute high score does not reflect the accuracy of the model.

The essence of this project to present the whole process of machine learning optimization. The optimization process was a valuable studying process for us four team members. This lead us to explore several algorithms and cross validation strategies.

The distribution of work is as follows:

Yu Hefei:
Lead and streamline of the whole project, Model Optimization, Report Compiling

Guo Yang:
Feature Engineering, Report writing, In charging of PPT

Wang Longling:
In charging of code management and debugging, Report writing

Zheng Siya:
Variable extraction, Report writing, In charging of brief instruction, Report Compiling

Contents

1. Problem Statement

Titanic Survival Prediction is a competition with the largest number of entries in Kaggle. As we all know, killing 1502 out of 2224 people's life, the sinking of Titanic is one of the most infamous shipwrecks in history. This competition requires contestants to analysis which type of passengers has more tendency to survive that disaster. Moreover, it requires to predict whether the people in the test data set can survive the tragedy or not by applying machine learning method.

2. Data Description

In this competition, we are given two data set: tran.csv (training set) and test.csv (test set). The training set is used to build a model, in which the outcome (also known as the "ground truth") is provided for each passenger. The test set is used to see how well the model performs on unseen data, in which, the outcome for each passenger is unknown.

Each record in the dataset describes a passenger. The attributes are defined as follows:

Table 1 Original Data Description

Attribute	Description
Passenger ID	Unique passenger identification number
Survived	Did a passenger survive or not (0 = died, 1 = survived)
Pclass	Ticket Class (1 = 1st; 2 = 2nd; 3 = 3rd)
Name	Name of Passenger
Sex	Sex of Passenger
Age	Passenger Age
SibSp	Number of Siblings/Spouses Aboard
Parch	Number of Parents/Children Aboard
Ticket	Ticket Number
Fare	Passenger Fare
Cabin	Cabin
Embarked	Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

3. Methodology and Implementation

3.1 Data Preparation

In the first place, we installed all the packages that would be relevant for the plots and algorithms in the later process and loaded the training dataset and testing dataset into our model. There are 891 observations in our training dataset and 418 observations in the testing dataset. Both datasets have missing values and exact 13 variables including Passenger ID, Survival, Passenger Class, Name, Sex, Age, Number of Siblings or Spouses, Number of Parents or Children, Ticket Number, Fare, Cabin Number, Port of Embarkation and an Indicator of the training set that we have introduced. The value of Survival variable is NA in the testing set and the indicator variable would print TRUE if the observation is in training set and FALSE for that of the testing set.

Next, we combined the training set and the testing set into a new dataset - full. With this new dataset, we could better investigate the characteristic and interrelation of variables and predict the missing values more precisely since more observations were used. Another reason using full dataset was consistency. Whenever we made changes or created new features in full dataset, the training set and the testing set would be automatically updated as well.

3.2 Variables Analysis and Missing Values

We were about to build our model based on the training set in order to predict the unknown data of survival in the testing set. Therefore, we first made value of survival feature as a factor and explored how other features might impact it.

Impact of Pclass and Sex

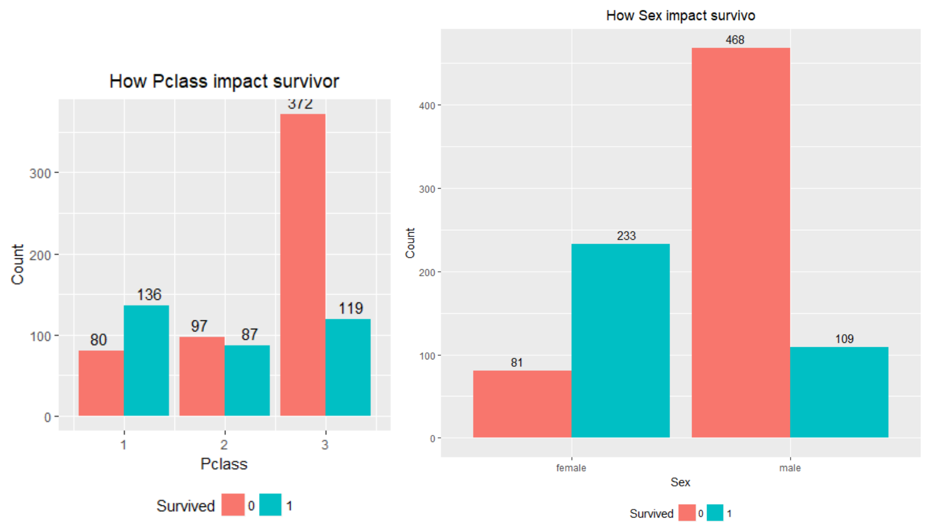


Figure 1 How Pclass and Sex Impact Survivor

The above features have significant effects on survival rate. The first plot shows that nearly three-fifth people in third class die. However, as the class gets pricey, the possibility of survival increases and reaches the peak in first class. For the sex feature, around 80% males die in contrast of the survival of 74% females.

Impact of Parch and SibSp

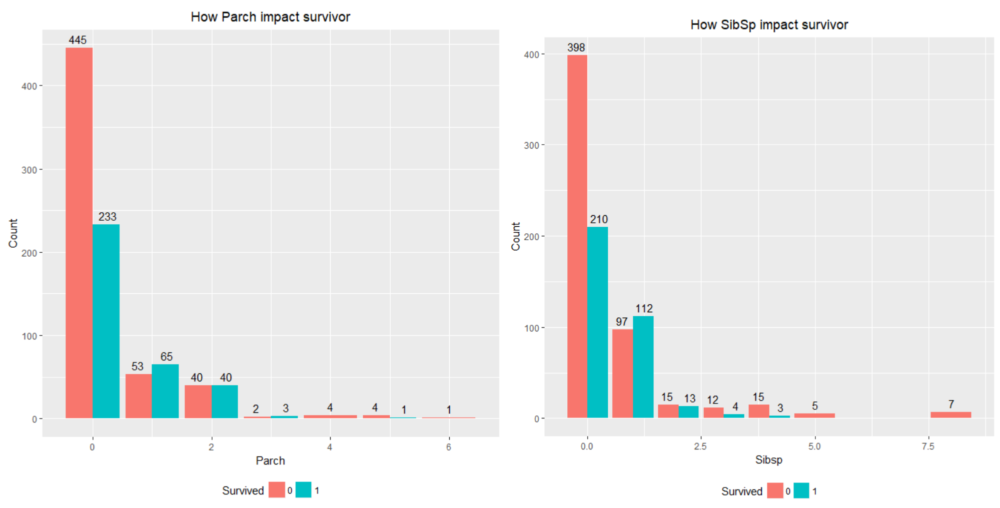


Figure 2 How Parch and SibSp Impact Survivor

The plots above indicate both the number of parents or children and the number of siblings or spouses are highly related to the survival. People who travel alone are most likely to die while people who travel with one or two members have larger chance to survive. However, as the family size increases, the possibility of survival decreases again.

Impact of Cabin

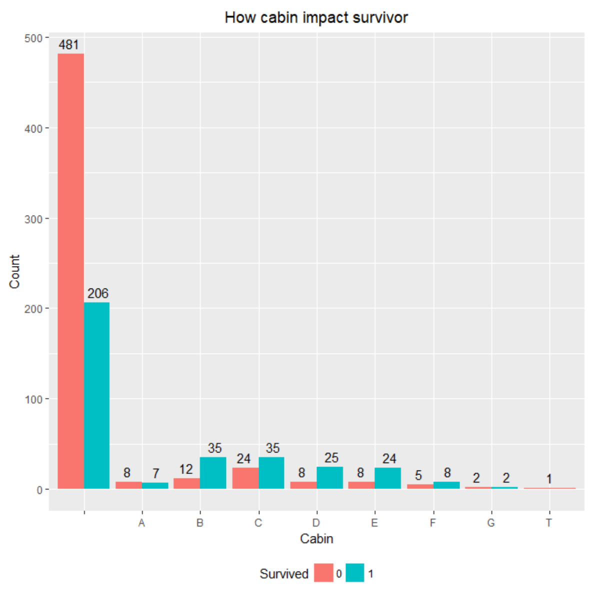
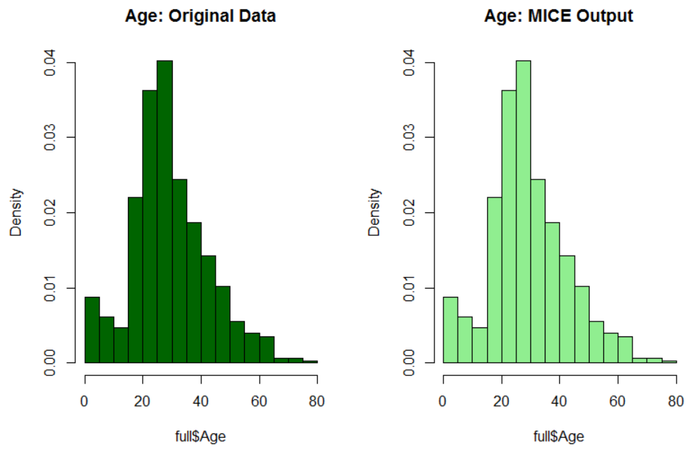


Figure 3 How Cabin Impact Survivor

This graph suggests most passengers with cabin number B, C, D, E, F survive at the end.

Impact of Age



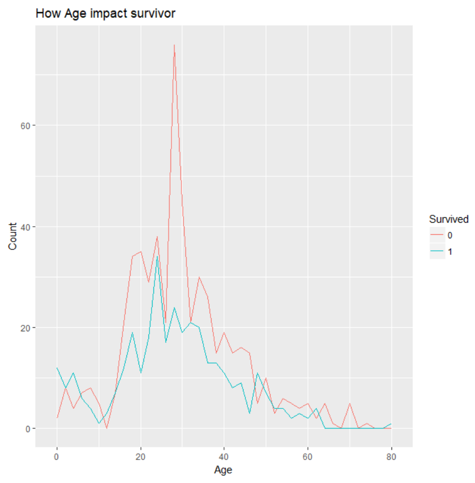


Figure 4 Age's Original Distribution and Prediction Distribution

Figure 5 How Age Impact Survivor

The overall dataset lacks 263 information about age. Based on other known features, we applied the random forest to predict the missing values. From Figure 4, we conclude that our predicted distribution of age is the same as the original one. The Figure 5 above is the impact of age on survival with no missing values. The survival rate is higher only when the age is smaller than 10.

Impact of Embarked and Fare

Two passengers who are in the first class have missing values in the port of embarkation. We replaced them with Southampton from which most people staying in the first class depart. In addition, the fare has one missing value and we used the regression to predict it. With the complete information, it is easy to conclude that people who from Cherbourg, France and people who have fare above 70 have higher survival rate.

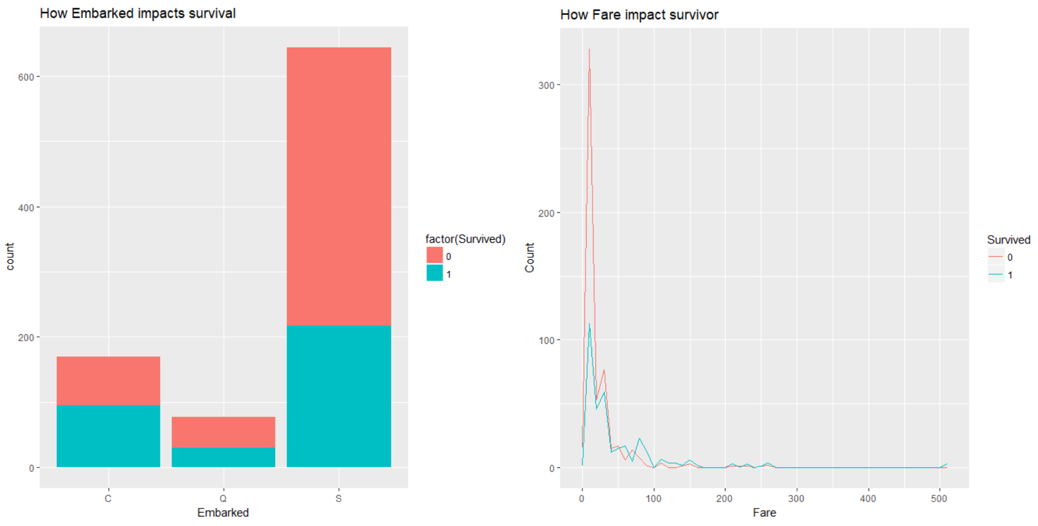


Figure 6 How Embarked and Fare Impact Survivor

3.3 Feature Engineering

Title

First, we noticed that there is a significant feature in the passenger name (“Name”): each name contains a specific title such as “Mr”, “Sir”, “Lady” … etc. This part of the information can be extracted as a useful new variable to reflect social positions of passengers to help us predict “Survived” factor.

```
# Extract title
full$Title <- sapply(full$Name, FUN=function(x) {strsplit(x, split='[,.]')[[1]][2]})
```

After exploring the “Name” factor, we find three kinds of commonly used titles for French ladies, English gentlemen and English ladies and we grouped them into three respective group.

```
full$Title <- as.character(full$Title)
full$Title[full$Title %in% c(' Mme', ' Mlle')] <- 'Mlle'
full$Title[full$Title %in% c(' Capt', ' Don', ' Major', ' Sir')] <- 'Sir'
full$Title[full$Title %in% c(' Dona', ' Lady', ' the Countess', ' Jonkheer')] <- 'Lady'
full$Title <- factor(full$Title)
```

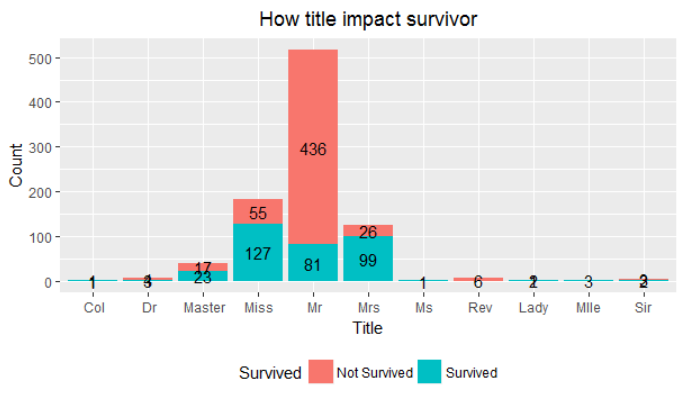


Figure 7 How Tile Impact Survivor

Then we can get the plot as shown in the previous page: the surviving rates of title "Mrs" and "Miss" are pretty much higher than other titles. That implies that ladies with high social positions are more likely to survive.

Family Size and Family ID

Here comes the question: does family size relate to surviving rate? In common sense, we may assume that the whole family members would survive or be dead together. So, we combine the variables of "SibSp" and "Parch" to show the whole family sizes of passengers.

```
#Imapct of family size
full$FamilySize <- full$SibSp + full$Parch + 1
ggplot(data = full[1:nrow(train),], mapping = aes(x = FamilySize, y = ..count.., fill=Survived)) +
  geom_bar(stat = 'count', position='dodge') +
  xlab('FamilySize') +
  ylab('Count') +
  ggtitle('How FamilySize impact survivor') +
  geom_text(stat = "count", aes(label = ..count..), position=position_dodge(width=1), , vjust=-0.5) +
  theme (plot. title = element_text (hjust = 0.5), legend. position="bottom")
```

We can clearly find the following characteristics through the graph below:

1. For those whose family sizes >4, the number of death is greater than the number of survived
2. For those whose family sizes between 2 and 4, the number of death is less than the number of survived

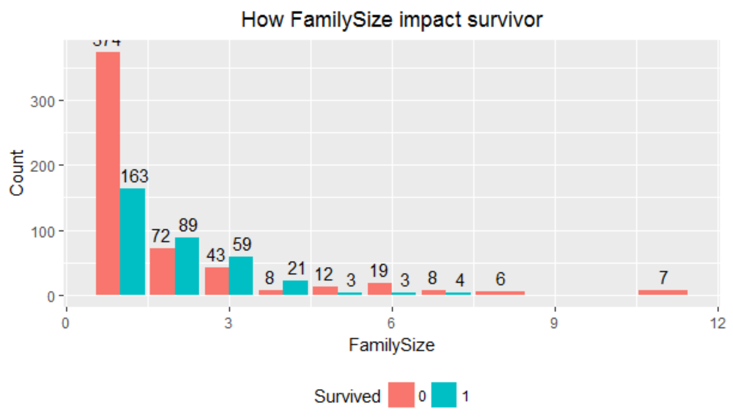


Figure 8 How Family Size Impact Survivor

How can we distinguish these big families? We may try to extract passenger's surname and combine it to family size to create a new variable "Family ID". Usually, different family ID can imply different social status that may link to the surviving opportunities.

```
# Extract Surname
full$Surname <- sapply(full$Name, FUN=function(x) {strsplit(x, split='[,.]')[[1]][1]})
data.frame(table(full$Surname))
full$Surname <- factor(full$Surname)
```

Then we let those whose family size<2 and surname frequency <2 be the small family ID, otherwise be the big family ID. The plot shown below: the surviving rate of small family ID is higher than big family ID, which has very significant impact on survivor.

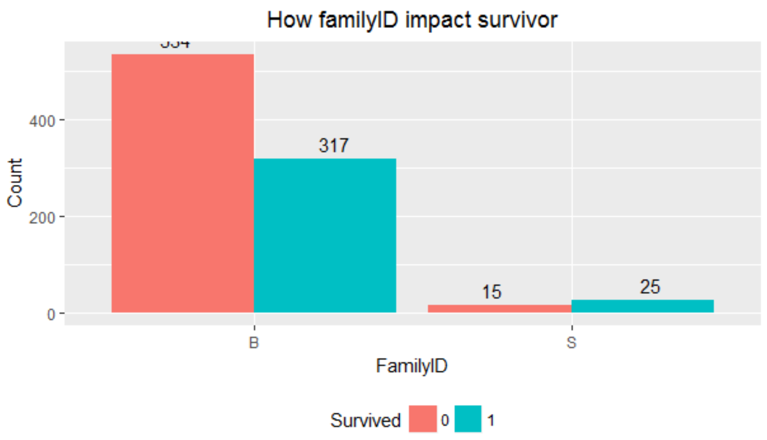


Figure 9 How FamilyID impact Survivor

Gender and Pclass Combination

According to what we have analyzed in the second part, gender and class of the passenger are most relevant to survivor. We can easily assume that women with high social status have the highest surviving rate among all. So, we decide to combine social class with gender to create a new variable "PclassAndSex".

```
#Impact of gender and class combination
full$PclassAndSex = 1
full[full$Sex=="female",]$PclassAndSex <-sapply(full[full$Sex=="female",]$Pclass,function(x) if(x==1)
as.character("female_high_class") else if(x==2) as.character("female_middle_class") else
as.character("female_low_class"))
full[full$Sex=="male",]$PclassAndSex <-sapply(full[full$Sex=="male",]$Pclass,function(x) if(x==1)
as.character("male_high_class") else if(x==2) as.character("male_middle_class")else
as.character("male_low_class"))
```

Ticket Number

It is speculated that people with the same ticket number are supposed to be in the same family and are more likely to survive or dead at the same time. So, we decide to divide ticket number into two groups: one is unique ticket number, the other is sharing ticket number. And we find that there is a total of 929 different ticket numbers, up to 11 people sharing a same number.

```
ticket.count <- aggregate(full$Ticket, by = list(full$Ticket), function(x) sum(!is.na(x)))
full$TicketCount <- apply(full, 1, function(x) ticket.count[which(ticket.count[, 1] == x['Ticket']), 2])
full$TicketCount <- factor(sapply(full$TicketCount, function(x) ifelse(x > 1, 'Share', 'Unique')))
```

We can observe from figure 10 that people who share ticket with others have significant higher survivor rate than people who have unique ticket. So, we add the new variable "TicketCount" to the classification model.

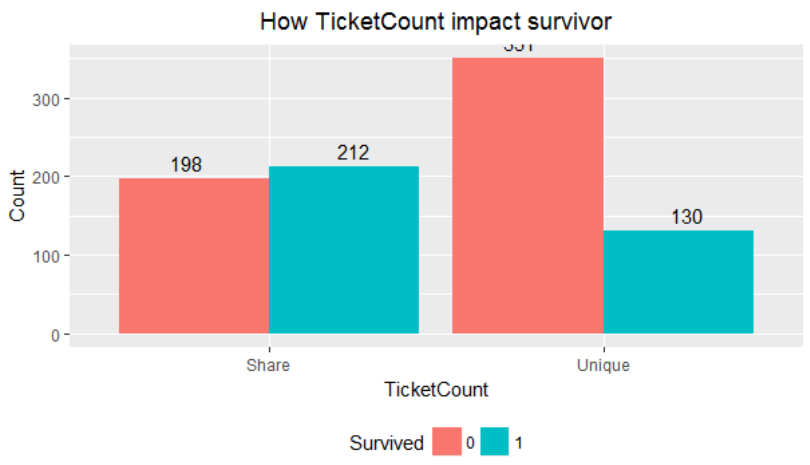


Figure 10 How TicketCount Impact Survivor

Mother

As watched in the Titanic movie, the captain always shouted "Women and children first!". In this case, we may suspect that women with children, in other words, mothers went in the very first when escaping from the ship. So, we generate a new variable "Mother" refers to those passengers whose gender =female, age >18, parch>0 and title! =Miss as shown below:

```
full$Mother <- 'NotMother'
full$Mother[full$Sex =='female' & full$Parch > 0 & full$Age > 18 & full$Title != 'Miss'] <- 'Mother'
table(full$Mother, full$Survived)
full$Mother <- factor(full$Mother)
```

3.4 Variables Correlations

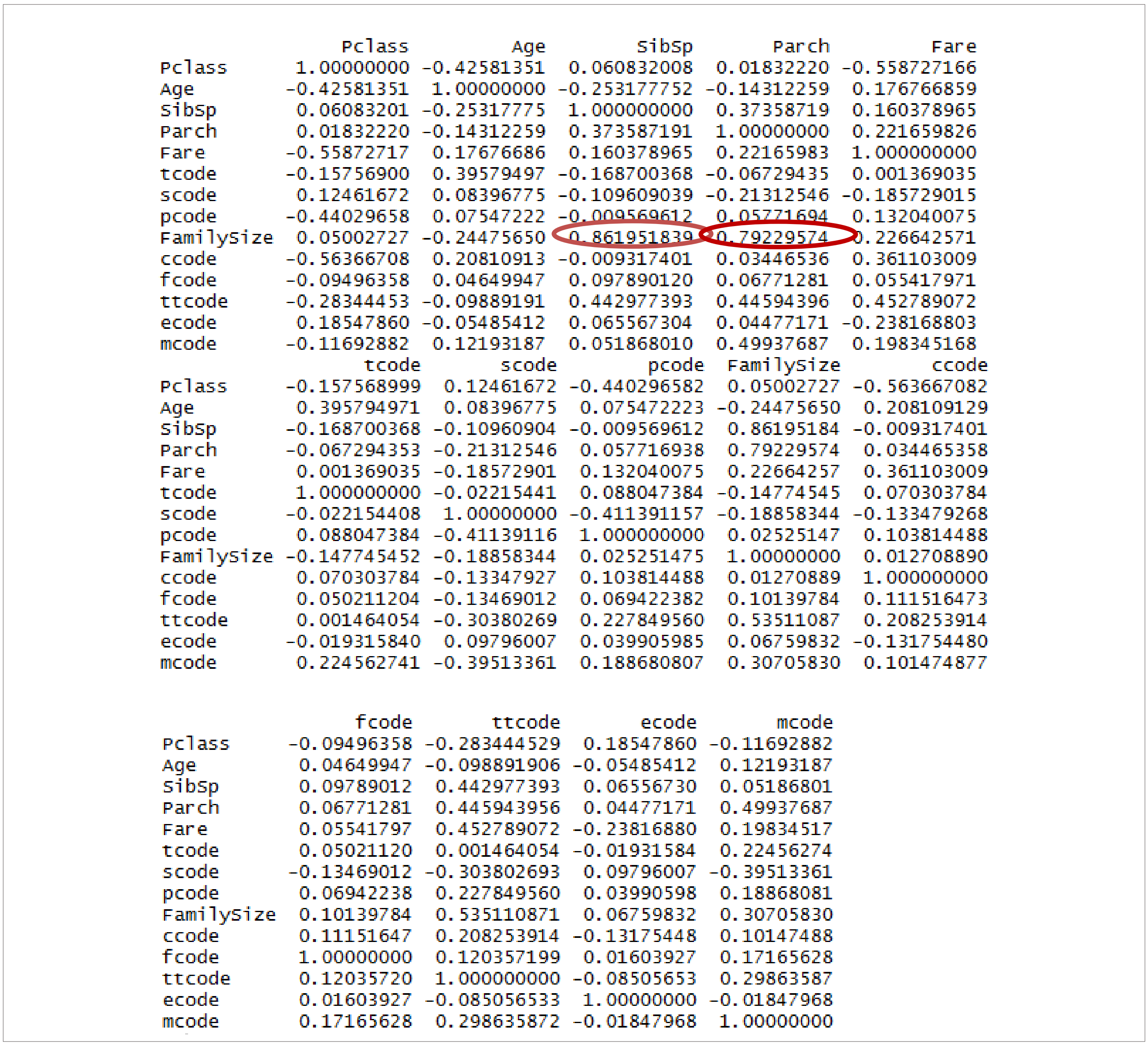
In most of the time variables with high correlation to the other can be removed to prevent duplication and noises. As the correlation function can only be used to numeric variables, we need to transform all 8 categorical variables to numeric forms first:

"Mother" – "mcode"; "Embark" – "ecode"; "TicketCount" – "ttcode"; "FamilyID" – "fcode"; "cabin" – "ccode"; "PclassAndSex" – "pcode"; "Sex" – "scode"; "Title – "tcode";

```
# take "Sex"— "ecode" as an example, using similar codes below to transform all categorical variables
 full$scode[full$Sex=="male"]<-"1"
 full$scode[full$Sex=="female"]<-"0" , full$scode<-as.numeric(full$scode)
```

Now we can use the correlation function to see the relationship among all attributes；

Check the correlation between the attributes, usually values above 0.75 or below -0.75 are indicative of high positive or high negative correlation. As we can see in the table in the next page, “familysize” has significant high positive correlation with “SibSp” and “Parch” (0.86 and 0.79 respectively). So, we can remove the factor “familysize” when training the classification model.



```
                Pclass         Age         SibSp        Parch         Fare
Pclass      1.00000000 -0.42581351  0.060832008  0.01832220 -0.558727166
Age        -0.42581351  1.00000000 -0.253177752 -0.14312259  0.176766859
SibSp       0.06083201 -0.25317775  1.000000000  0.37358719  0.160378965
Parch       0.01832220 -0.14312259  0.373587191  1.00000000  0.221659826
Fare       -0.55872717  0.17676686  0.160378965  0.22165983  1.000000000
tcode      -0.15756900  0.39579497 -0.168700368 -0.06729435  0.001369035
scode       0.12461672  0.08396775 -0.109609039 -0.21312546 -0.185729015
pcode      -0.44029658  0.07547222 -0.009569612  0.05771694  0.132040075
FamilySize  0.05002727 -0.24475650  0.861951839  0.79229574  0.226642571
ccode      -0.56366708  0.20810913 -0.009317401  0.03446536  0.361103009
fcode      -0.09496358  0.04649947  0.097890120  0.06771281  0.055417971
ttcode     -0.28344453 -0.09889191  0.442977393  0.44594396  0.452789072
ecode       0.18547860 -0.05485412  0.065567304  0.04477171 -0.238168803
mcode      -0.11692882  0.12193187  0.051868010  0.49937687  0.198345168
                  tcode       scode        pcode  FamilySize        ccode
Pclass     -0.157568999  0.12461672 -0.440296582  0.05002727 -0.563667082
Age         0.395794971  0.08396775  0.075472223 -0.24475650  0.208109129
SibSp      -0.168700368 -0.10960904 -0.009569612  0.86195184 -0.009317401
Parch      -0.067294353 -0.21312546  0.057716938  0.79229574  0.034465358
Fare        0.001369035 -0.18572901  0.132040075  0.22664257  0.361103009
tcode       1.000000000 -0.02215441  0.088047384 -0.14774545  0.070303784
scode      -0.022154408  1.00000000 -0.411391157 -0.18858344 -0.133479268
pcode       0.088047384 -0.41139116  1.000000000  0.02525147  0.103814488
FamilySize -0.147745452 -0.18858344  0.025251475  1.00000000  0.012708890
ccode       0.070303784 -0.13347927  0.103814488  0.01270889  1.000000000
fcode       0.050211204 -0.13469012  0.069422382  0.10139784  0.111516473
ttcode      0.001464054 -0.30380269  0.227849560  0.53511087  0.208253914
ecode      -0.019315840  0.09796007  0.039905985  0.06759832 -0.131754480
mcode       0.224562741 -0.39513361  0.188680807  0.30705830  0.101474877


                 fcode       ttcode       ecode       mcode
Pclass     -0.09496358 -0.283444529  0.18547860 -0.11692882
Age         0.04649947 -0.098891906 -0.05485412  0.12193187
SibSp       0.09789012  0.442977393  0.06556730  0.05186801
Parch       0.06771281  0.445943956  0.04477171  0.49937687
Fare        0.05541797  0.452789072 -0.23816880  0.19834517
tcode       0.05021120  0.001464054 -0.01931584  0.22456274
scode      -0.13469012 -0.303802693  0.09796007 -0.39513361
pcode       0.06942238  0.227849560  0.03990598  0.18868081
FamilySize  0.10139784  0.535110871  0.06759832  0.30705830
ccode       0.11151647  0.208253914 -0.13175448  0.10147488
fcode       1.00000000  0.120357199  0.01603927  0.17165628
ttcode      0.12035720  1.000000000 -0.08505653  0.29863587
ecode       0.01603927 -0.085056533  1.00000000 -0.01847968
mcode       0.17165628  0.298635872 -0.01847968  1.00000000
```

Figure 11 Cor Among Variables

4. Experimental Results

4.1 Evaluate Algorithms

For this project, we will evaluate 3 categories of algorithms: linear, non-linear and ensembles models.

Algorithm	Brief Description
Linear Regression	
Logistic Regression(LG)	This algorithm transforms the dependent variable in the logit function and employs these results to predict the event.
Linear Discriminate Analysis(LDA)	This algorithm employs Fisher's linear discriminant to find a linear combination of features characterizing different object.
Regularized Logistic Regression(GLMNET)	This algorithm fits a generalized linear model via penalized maximum likelihood.
Non-Linear Regression	
k-Nearest Neighbors(KNN)	This algorithm is a type of instance-based learning, its output is a class membership. By this method, an object is classified by a majority vote of its neighbors.
Classification and Regression Trees(CART)	This algorithm refers both to classification tree and regression tree method. It employs decision tree learning which is based from class-labeled training tuples.
Na ïve Bayes(NB)	This algorithm is a simple probabilistic classifiers based on applying Bayes" theorem with na ïve independence assumption between features
Support Vector Machines with Radial Basis Function(SVM)	The support vector machines are supervised learning models with associated learning algorithms that analyze data. In this example, we add a kernel trick: Radial Basis Function to enable it to perform a non-linear classification.
Ensemble Methods: Bagging	
Bagged CART (BAG)	This algorithm (Bootstrap Aggregation) reduces the variance of the prediction from decision trees by employing multiple sub -tress.
Random Forest (RF)	RF is an improved version of BAG. It changes the algorithm for the way that the sub-trees are learned so that the resulting predictions from all the subtrees have less correlation. It limits the learning algorithm to a random sample of features of which to search.
Ensemble Methods: Boosting	
Stochastic Gradient Boosting (GBM)	This algorithm builds the model in a stage-wise fashion like other booting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. The stochastic feature enables the algorithm's base learner be fit on a subsample of the training set drawn at random without replacement.
C5.0 (C50)	C5.0 is a popular algorithm used for classification and regression

Table 2 Various Algorithm Used in This Project

4.2. Cross Validation

We define a test harness first. We will use 10-fold cross validation with 3 repeats. In this sense, the data will be re-sample 30 times for each algorithm. Since this is a binary classification problem, we only care about accuracy and kappa metrics.

In addition, before doing above, we need to split the full data set into original data set.

```
#Split dataset back out into train and test
train<-full[full$IsTrainSet==TRUE,]
test<-full[full$IsTrainSet==FALSE,]
#Define a test harness
trainControl <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "Accuracy"
```

Here we train all models by using seed (7).

```
# LG
set.seed(7)
fit.glm <- train(Survived~., data=train[c(2,3,6,7,8,10,15,16,18,20,24,25,27,30)], method="glm",
metric=metric, trControl=trainControl)
# LDA
set.seed(7)
fit.lda <- train(Survived~., data=train[c(2,3,6,7,8,10,15,16,18,20,24,25,27,30)], method="lda", metric=metric,
trControl=trainControl)
#GLMNET
set.seed(7)
fit.glmnet <- train(Survived~., data=train[c(2,3,6,7,8,10,15,16,18,20,24,25,27,30)], method="glmnet",
metric=metric, trControl=trainControl)
#KNN
set.seed(7)
fit.knn <- train(Survived~., data=train[c(2,3,6,7,8,10,15,16,18,19,20,24,26,27,30)], method="knn",
metric=metric, preProc=c("BoxCox"), trControl=trainControl)
#CART
set.seed(7)
fit.glmnet <- train(Survived~., data=train[c(2,3,6,7,8,10,15,16,18,20,24,25,27,30)], method="rpart",
metric=metric, trControl=trainControl)
# Naive Bayes
set.seed(7)
fit.glmnet <- train(Survived~., data=train[c(2,3,6,7,8,10,15,16,18,20,24,25,27,30)], method="nb",
metric=metric, trControl=trainControl)
# SVM
set.seed(7)
fit.glmnet <- train(Survived~., data=train[c(2,3,6,7,8,10,15,16,18,20,24,25,27,30)], method="svmRadial",
metric=metric, trControl=trainControl)
# Bagged CART
set.seed(7)
fit.treebag <- train(Survived~., data=train[c(2,3,6,7,8,10,15,16,18,19,20,24,26,27,30)], method="treebag",
metric=metric, trControl=trainControl)
# Random Forest
set.seed(7)
fit.rf <- train(Survived~., data=train[c(2,3,6,7,8,10,15,16,18,19,20,24,26,27,30)], method="rf", metric=metric,
trControl=trainControl)
# C5.0
set.seed(7)
fit.c50 <- train(Survived~., data=train[c(2,3,6,7,8,10,15,16,18,19,20,24,26,27,30)], method="C5.0",
metric=metric, trControl=trainControl)
# Stochastic Gradient Boosting
set.seed(7)
```

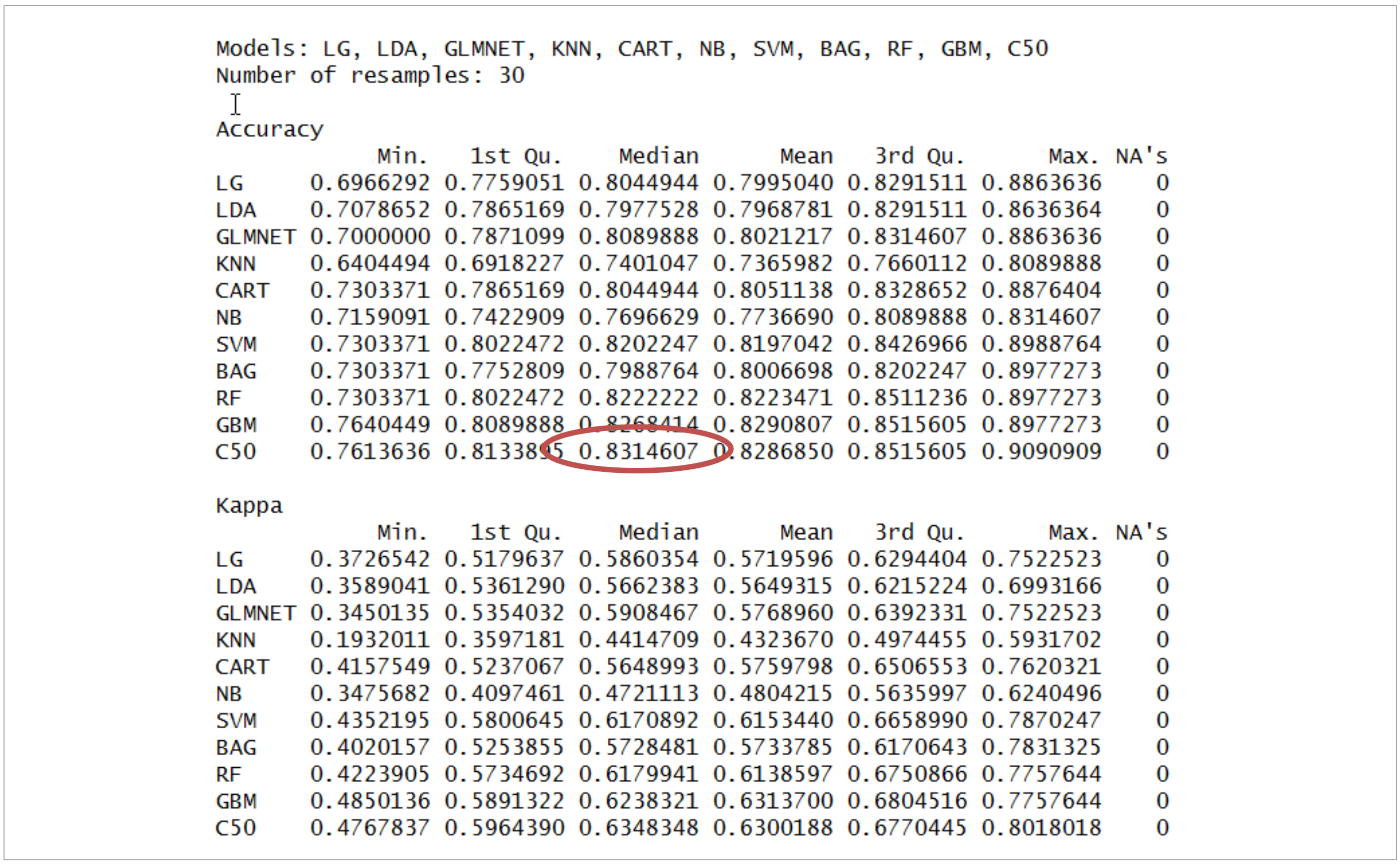
```
fit.gbm <- train(Survived~., data=train[c(2,3,6,7,8,10,15,16,18,19,20,24,26,27,30)], method="gbm",
metric=metric, trControl=trainControl, verbose=FALSE)
```

5. Model Optimization

We compare all these algorithms' mean and median accuracy.

```
# Compare algorithms
results <- resamples(list(LG=fit.glm, LDA=fit.lda, GLMNET=fit.glmnet, KNN=fit.knn, CART=fit.cart, NB=fit.nb,
SVM=fit.svm, BAG=fit.treebag, RF=fit.rf, GBM=fit.gbm, C50=fit.c50))
summary(results)
```

Here is the results:



```
Models: LG, LDA, GLMNET, KNN, CART, NB, SVM, BAG, RF, GBM, C50
Number of resamples: 30

Accuracy
            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
LG     0.6966292 0.7759051 0.8044944 0.7995040 0.8291511 0.8863636    0
LDA    0.7078652 0.7865169 0.7977528 0.7968781 0.8291511 0.8636364    0
GLMNET 0.7000000 0.7871099 0.8089888 0.8021217 0.8314607 0.8863636    0
KNN    0.6404494 0.6918227 0.7401047 0.7365982 0.7660112 0.8089888    0
CART   0.7303371 0.7865169 0.8044944 0.8051138 0.8328652 0.8876404    0
NB     0.7159091 0.7422909 0.7696629 0.7736690 0.8089888 0.8314607    0
SVM    0.7303371 0.8022472 0.8202247 0.8197042 0.8426966 0.8988764    0
BAG    0.7303371 0.7752809 0.7988764 0.8006698 0.8202247 0.8977273    0
RF     0.7303371 0.8022472 0.8222222 0.8223471 0.8511236 0.8977273    0
GBM    0.7640449 0.8089888 0.8268414 0.8290807 0.8515605 0.8977273    0
C50    0.7613636 0.8133895 0.8314607 0.8286850 0.8515605 0.9090909    0

Kappa
            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
LG     0.3726542 0.5179637 0.5860354 0.5719596 0.6294404 0.7522523    0
LDA    0.3589041 0.5361290 0.5662383 0.5649315 0.6215224 0.6993166    0
GLMNET 0.3450135 0.5354032 0.5908467 0.5768960 0.6392331 0.7522523    0
KNN    0.1932011 0.3597181 0.4414709 0.4323670 0.4974455 0.5931702    0
CART   0.4157549 0.5237067 0.5648993 0.5759798 0.6506553 0.7620321    0
NB     0.3475682 0.4097461 0.4721113 0.4804215 0.5635997 0.6240496    0
SVM    0.4352195 0.5800645 0.6170892 0.6153440 0.6658990 0.7870247    0
BAG    0.4020157 0.5253855 0.5728481 0.5733785 0.6170643 0.7831325    0
RF     0.4223905 0.5734692 0.6179941 0.6138597 0.6750866 0.7757644    0
GBM    0.4850136 0.5891322 0.6238321 0.6313700 0.6804516 0.7757644    0
C50    0.4767837 0.5964390 0.6348348 0.6300188 0.6770445 0.8018018    0
```

Figure 12 Model Performance Comparison

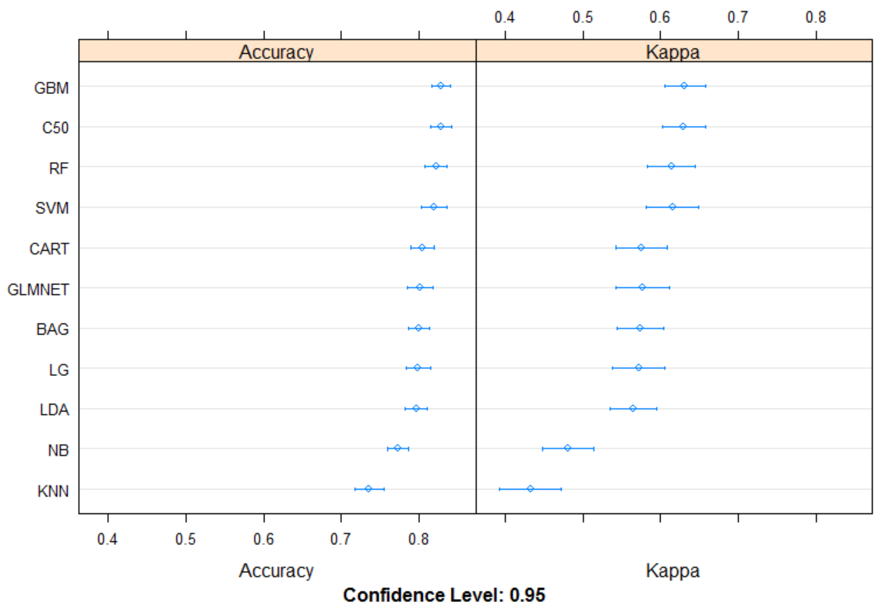


Figure 13 Model Performance Comparison

From the results, we could find that the best model is C5.0 with a median accuracy of 0.831.

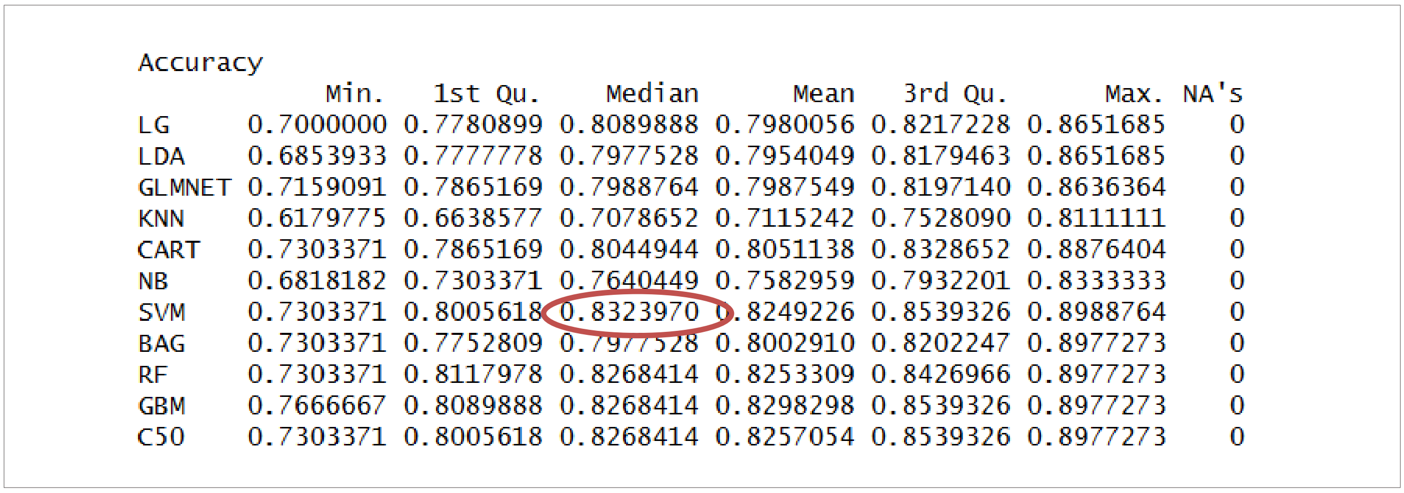
5.1 Box-Cox Transformation

To further optimize, we apply Box-Cox transform to flatten out the distribution.

Box-Cox transformation's function in this project was to transform the full data set to let it follow approximately a normal distribution. This transformation can relatively reduce the prediction error and correlation between predictive variable.

```
# We take SVM Model for an example
set.seed(7)
fit.glmnet <- train(Survived~., data=train[c(2,3,6,7,8,10,15,16,18,20,24,25,27,30)], method="svmRadial", metric=metric, preProc=c("BoxCox"), trControl=trainControl)
```

The results is as follows



```
Accuracy
            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
LG     0.7000000 0.7780899 0.8089888 0.7980056 0.8217228 0.8651685    0
LDA    0.6853933 0.7777778 0.7977528 0.7954049 0.8179463 0.8651685    0
GLMNET 0.7159091 0.7865169 0.7988764 0.7987549 0.8197140 0.8636364    0
KNN    0.6179775 0.6638577 0.7078652 0.7115242 0.7528090 0.8111111    0
CART   0.7303371 0.7865169 0.8044944 0.8051138 0.8328652 0.8876404    0
NB     0.6818182 0.7303371 0.7640449 0.7582959 0.7932201 0.8333333    0
SVM    0.7303371 0.8005618 0.8323970 0.8249226 0.8539326 0.8988764    0
BAG    0.7303371 0.7752809 0.7977528 0.8002910 0.8202247 0.8977273    0
RF     0.7303371 0.8117978 0.8268414 0.8253309 0.8426966 0.8977273    0
GBM    0.7666667 0.8089888 0.8268414 0.8298298 0.8539326 0.8977273    0
C50    0.7303371 0.8005618 0.8268414 0.8257054 0.8539326 0.8977273    0
```

Figure 14 Model Performance Comparison After Box-Cox Transformation

From the results, we conclude that SVM is the best model with a median accuracy of 0.832.

5.2 Tuning SVM

After SVM method was selected, we tuned its parameters to further improve accuracy. The SVM algorithm has two parameters that we can tune with. The two parameters are sigma and C. Sigma is the smoothing term and c is a cost constraint.

In this tuning, we set the sigma value to c(0.025, 0.05, 0.1, 0.15) and set the C value to seq(1,10,1). Hence, we have 40 combination of sigma value and C value. Here is the code:

```r
set.seed(7)
grid <- expand.grid(.sigma=c(0.025, 0.05, 0.1, 0.15), .C=seq(1, 10, by=1))
fit.svm <- train(Survived~., data=train[c(2,3,6,7,8,10,15,16,18,19,20,24,26,27,30)], method="svmRadial",
metric=metric, tuneGrid=grid, preProc=c("BoxCox"), trControl=trainControl)
print(fit.svm)
```

Here are the results:

```
# sigma   C    Accuracy    Kappa
#0.025    1   0.8335329   0.6433075
#0.025    2   0.8324093   0.6404208
#0.025    3   0.8331668   0.6419237
#0.025    4   0.8312856   0.6372897
#0.025    5   0.8309069   0.6359743
#0.025    6   0.8282935   0.6304750
#0.025    7   0.8286766   0.6314225
#0.025    8   0.8279190   0.6295103
#0.025    9   0.8267912   0.6269989
#0.025   10   0.8249311   0.6228956
#0.050    1   0.8305450   0.6362687
#0.050    2   0.8275320   0.6293711
#0.050    3   0.8230542   0.6195774
#0.050    4   0.8193088   0.6103811
#0.050    5   0.8200580   0.6116951
#0.050    6   0.8204493   0.6115382
#0.050    7   0.8234498   0.6175531
#0.050    8   0.8238244   0.6180519
#0.050    9   0.8264419   0.6238688
#0.050   10   0.8253225   0.6214785
#0.100    1   0.8215434   0.6166546
#0.100    3   0.8223177   0.6152577

# sigma   C    Accuracy    Kappa
#0.100    2   0.8212025   0.6130388
#0.100    4   0.8163336   0.6031798
#0.100    5   0.8129669   0.5954187
#0.100    6   0.8114813   0.5916998
#0.100    7   0.8084765   0.5850388
#0.100    8   0.8096002   0.5877959
#0.100    9   0.8084891   0.5860016
#0.100   10   0.8114855   0.5929343
#0.150    1   0.8192962   0.6106993
#0.150    2   0.8140904   0.5991889
#0.150    3   0.8107237   0.5918708
#0.150    4   0.8077358   0.5851296
#0.150    5   0.8084890   0.5875903
#0.150    6   0.8118557   0.5956381
#0.150    7   0.8118558   0.5962959
#0.150    8   0.8099873   0.5922232
#0.150    9   0.8092424   0.5904366
#0.150   10   0.8099874   0.5919317
#Accuracy was used to select the optimal model using
the largest value.
#The final values used for the model were sigma = 0.025
and C = 1.
```

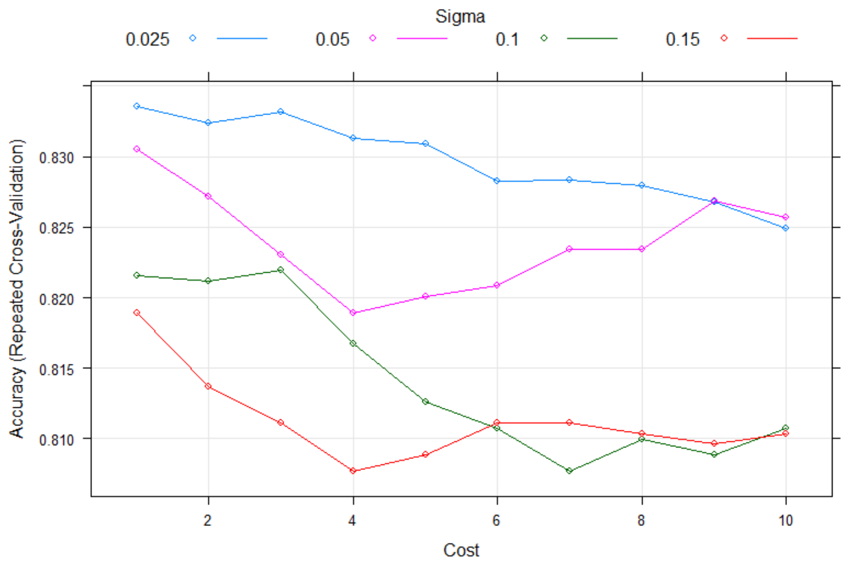


Figure 15 SVM by Various Parameters

From the results, we found that the best value for sigma and C is 0.05 and C=1

5.3 Variable Optimization

We have concluded 13 variables that would be used in our prediction in the previous part. However, it may not be the best if we use them all since there may be noise generated in the prediction process. Hence, we need to optimize the variables we used.

The variable combinations were generated by choosing more than 4 variables among 13 variables. Then each variable combination would be trained as one model. The final results would be compared by their mean and median accuracy value.

```
x=c(3,6,7,8,10,15,16,18,19,20,24,26,27,30)
# y1=combn(x, 5, simplify = FALSE); y2=combn(x, 6, simplify = FALSE); y3=combn(x, 7, simplify = FALSE)
# y4=combn(x, 8, simplify = FALSE); y5=combn(x, 9, simplify = FALSE); y6=combn(x, 10, simplify = FALSE)
# y7=combn(x, 11, simplify = FALSE); y8=combn(x, 12, simplify = FALSE); y9=combn(x, 13, simplify = FALSE)
# y=c(y1,y2,y3,y4,y5,y6,y7,y8,y9)
for (i in 1:length(y))
# { set.seed(7)
# assign(paste("fit.svm", i, sep = ""), train(Survived~., data=train[c(y[[i]])], method="svmRadial", metric=metric,
# preProc=c("BoxCox"), trControl=trainControl))}
# for (i in 1:length(y))
# {z[i]=paste("fit.svm", i, sep = "")}
# Compare algorithms
# transformResults1 <- resamples(list(z))
# summary(transformResults1)
```

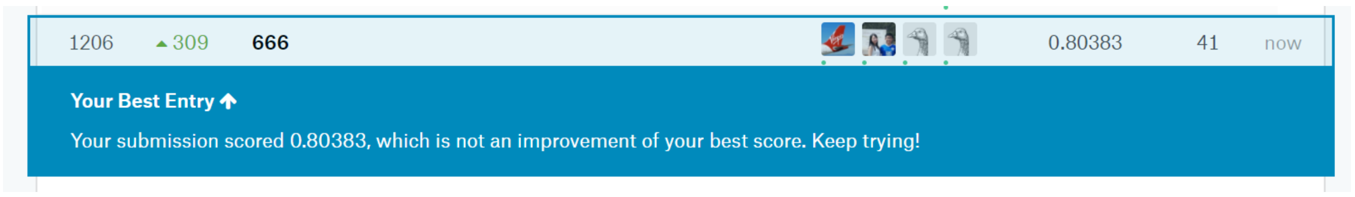
This process may take very long and the content is too long to present here. The final optimization result is that we need to maintain all the 13 variables.

5.4 Final Results

Let's predict the test dataset using all the above model.

```
#Predict the test dataset(exclude family size)
model<-svm(Survived~.,data=train[c(2,3,6,7,8,10,15,16,18,20,24,26,27,30)])
testData<-test[c(3,6,7,8,10,15,16,18,20,24,26,27,30)]
preprocessParams <- preProcess(testData, method=c("BoxCox"))
testData <- predict(preprocessParams, testData)
predictions <- predict(model, testData, type="class")
submit <- data.frame(PassengerId = test$PassengerId, Survived = predictions)
write.csv(submit, file = "Submission.csv", row.names = FALSE)
```

The final result is 0.80383, which is among top 15% percentile.



1206 ▲309 666 0.80383 41 now

Your Best Entry ↑

Your submission scored 0.80383, which is not an improvement of your best score. Keep trying!

Figure 16 Kaggle Results

6. Conclusions

6.1 Summary of project achievement

Model Result has an accuracy of 80.4% (evaluated by Kaggle) by using the approach of SVM (Support vector machine), which means we can successfully predict whether or not the passenger can survive with a probability of 80.4%. This was ranked in the top 15% percentile.

6.2 Future Directions for improvement

Expanding Variables Numbers

One way to fully evaluate the variable's weightage in the prediction is to expand all the variables' subcategories. For example, now we only have one variable for sex, but we have two sub-categories. We can expand the sex variables and get two new variables: Male and Female. For continuous number variable, like age and fair, we may do so by split them into different intervals. If we expand all the variable's sub-categories, we may have 100+ new variables. We could use these 100+ new variables to further optimize which one should be dropped for prediction.

PCA Techniques

The other improvement can be done is that the dimensionality reduction can be performed to decrease complexity. The PCA (Principal component analysis) is the technique to extract low dimensional set of features from a high dimensional data set (>=3) with a motive to capture as much information as possible. In this example, our variable dimension is only 13. However, in some real-world problems, we may have millions of variables. In such sense, PCA Technique is very valuable – It can be used for further dimensionality reduction to reduce the complexity and improve the computing performance.