Bohua Huang, Deepanshu Sethi, Jingyin Chen, Runhan Xu, Tingyin Ye, Yuanzhao Hou

MSBA 435

Marketing Models and Digital Analytics

Titanic Project

Background :

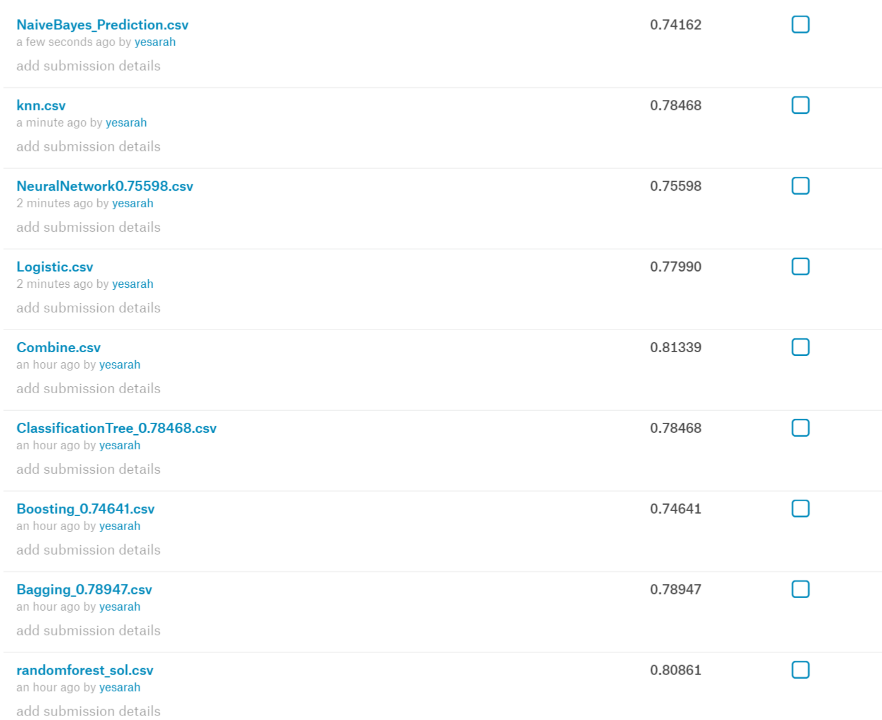
Titanic: Machine Learning from Disaster is the famous machine learning competition in Kaggle. As the competition description said, the sinking of the Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the widely considered "unsinkable" RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren't enough lifeboats for everyone on board, resulting in the death of 1502 out of 2224 passengers and crew. While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

What we did in this project is to find out those elements which could influence a person to survive or not using the passengers and combine these elements to build the models or using machine learning algorithm to make the predictions to test data in the real case. And Kaggle will give a correct rate as the score of each prediction. To make a better prediction, our group tried several different models to get the highest score models and used model ensemble to improve the score.

Our group has tried these models: (with its score). The table sorted by accuracy from high to low. We choose the top 4 models to describe below. They are Ensemble, Random Forest, Bagging and KNN models.

Index	Model	Accuracy
1	Ensemble	0.8134

2	**Random Forest**	**0.8086**
3	**Bagging**	**0.7895**
4	**KNN**	**0.7847**
5	Classification Tree	0.7847
6	Logistic Regression	0.7799
7	Neural Network	0.7559
8	Boosting	0.7464
9	Naïve Bayes	0.7416



NaiveBayes_Prediction.csv a few seconds ago by yesarah add submission details	0.74162
knn.csv a minute ago by yesarah add submission details	0.78468
NeuralNetwork0.75598.csv 2 minutes ago by yesarah add submission details	0.75598
Logistic.csv 2 minutes ago by yesarah add submission details	0.77990
Combine.csv an hour ago by yesarah add submission details	0.81339
ClassificationTree_0.78468.csv an hour ago by yesarah add submission details	0.78468
Boosting_0.74641.csv an hour ago by yesarah add submission details	0.74641
Bagging_0.78947.csv an hour ago by yesarah add submission details	0.78947
randomforest_sol.csv an hour ago by yesarah add submission details	0.80861

Data Preprocessing and Feature Engineering :

1. Data summary with missing value

There are 2 datasets in the project. One dataset is titled `train.csv` for training the predict models and the other is titled `test.csv` for test models. There are 891 observations in the training dataset and 418 observations in the test dataset. The variable structure is the same in two datasets as below:

Variable	Definition	Notw
survival	Survival	0 = No, 1 = Yes(Not contain in test dataset)
pclass	Ticket class	A proxy for socio-economic status (SES) 1st = Upper 2nd = Middle 3rd = Lower
sex	Sex	
Age	Age in years	Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5
sibsp	# of siblings / spouses aboard the Titanic	The dataset defines family relations in this way... Sibling = brother, sister, stepbrother, stepsister Spouse = husband, wife (mistresses and fiancés were ignored)
parch	# of parents / children aboard the Titanic	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.
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

To check every variable in the 2 datasets:

Train Dataset:

```
> summary(dat1)
  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           
```

We can see that 687 observations have " "(Null) in Cabin variable and 2 observations has " " (Null) in Embarked variable.

```
> nrow(dat1[which(is.na(dat1)),])
[1] 177
```

177 observations have NA values. After manually checking every variable, we can find all of those NA values are in Age variable.

Test Dataset:

```
> summary(dat2)
  PassengerId        Pclass                                                  Name    
 Min.   : 892.0   Min.   :1.000   Abbott, Master. Eugene Joseph                :  1  
 1st Qu.: 996.2   1st Qu.:1.000   Abelseth, Miss. Karen Marie                  :  1  
 Median :1100.5   Median :3.000   Abelseth, Mr. Olaus Jorgensen                :  1  
 Mean   :1100.5   Mean   :2.266   Abrahamsson, Mr. Abraham August Johannes :  1  
 3rd Qu.:1204.8   3rd Qu.:3.000   Abrahim, Mrs. Joseph (Sophie Halaut Easu):  1  
 Max.   :1309.0   Max.   :3.000   Aks, Master. Philip Frank                    :  1  
                                  (Other)                                      :412  

     Sex           Age            SibSp            Parch            Ticket   
 female:152   Min.   : 0.17   Min.   :0.0000   Min.   :0.0000   PC 17608:  5  
 male  :266   1st Qu.:21.00   1st Qu.:0.0000   1st Qu.:0.0000   113503  :  4  
              Median :27.00   Median :0.0000   Median :0.0000   CA. 2343:  4  
              Mean   :30.27   Mean   :0.4474   Mean   :0.3923   16966   :  3  
              3rd Qu.:39.00   3rd Qu.:1.0000   3rd Qu.:0.0000   220845  :  3  
              Max.   :76.00   Max.   :8.0000   Max.   :9.0000   347077  :  3  
              NA's   :86                                        (Other) :396  

      Fare                     Cabin     Embarked
 Min.   :  0.000                  :327   C:102   
 1st Qu.:  7.896   B57 B59 B63 B66:  3   Q: 46   
 Median : 14.454   A34            :  2   S:270   
 Mean   : 35.627   B45            :  2           
 3rd Qu.: 31.500   C101           :  2           
 Max.   :512.329   C116           :  2           
 NA's   :1         (Other)        : 80           
```

Only Cabin variable has 327 Null values.

```
> nrow(dat1[which(is.na(dat2)),])
[1] 87
```

87 observations have NA values and after checking, all the NA values are in Age variable.

So that the missing value table is:

Dataset	Variable Name	Missing Values	Importance(Missing number/All number)
Train	Cabin	687	0.771043771
	Embarked	2	0.002244669
	Age	177	0.198653199
Test	Cabin	327	0.782296651
	Age	87	0.208133971
Total	Cabin	1014	0.774637128
	Embarked	2	0.001527884
	Age	264	0.201680672

We can see that Cabin and Age are 2 important missing values which means we could not ignore them and should pay attention and find a better way to fill these missing values. Embarked is a less important missing value that we could use some simple ways to solve it.

2. Missing value Solution:

① Cabin:

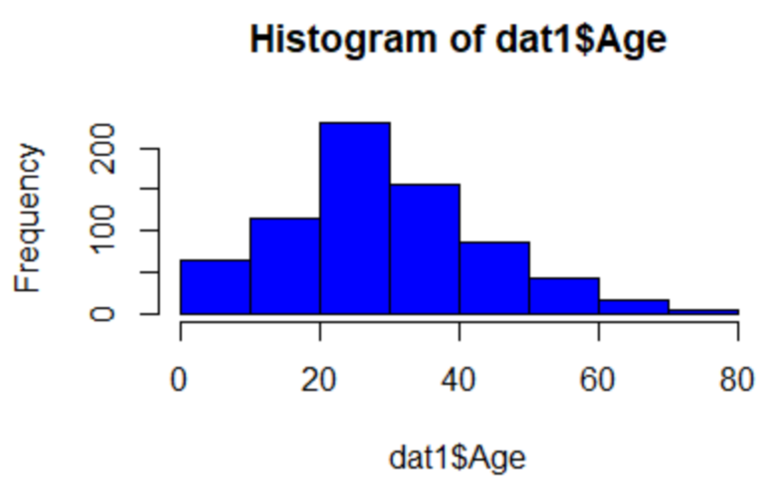
As Cabin missing values is up to 77% in the total dataset and the meaning of the Cabin value is unclear. We can not get any direct information from the alphabet or the number in Cabin and it's very hard to find out the useful information in it. But in our feature engineering we found that Cabin value exists indeed influence the survival rate.

So our group defined a new variable for Cabin and made it as a binary variable. (1 for those having cabin values, and 0 for Null values)

Cabin	Embarked	Cabin2
A23	S	1
	S	0
A5	C	1

② Age:

Age variable has about 20% missing values, so deleting those observations is not a good choice.



Checking the Histogram of Age, we could find that Age variable is skewed. So using the average to fill the missing value is not a good choice. Below is the solution we used as learning from the pattern of Age variable.

1) Median

```
> median(entire$Age[!is.na(entire$Age)])
[1] 29
```

From the pattern we could notice the Age is skewed and the highest volume is in 20-30. So using the median as the missing value could be a better choice than average value.

2) Median in each title

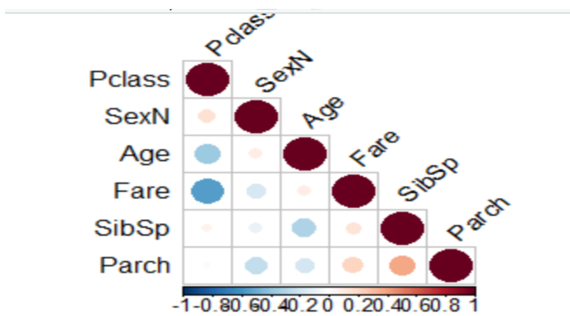
From our feature engineering, we create a new variable called title which used the name variable and find out the title of each person like "Master","Mr" and"Mrs". And we find out a close relationship between title and Age:

	Group.1	x
1	Master	4
2	Miss	22
3	Mr	29
4	Mrs	35
5	Officer	49
6	Royalty	39

Different titles have different median in Age attribute. Using the median for each title to fill in the Age missing value is a more reasonable choice. However, Mr has 67% missing values in Age miss values and its age median is the same as the whole median. So method 2 does not have a large difference with method 1. But it slightly improves the prediction accuracy. Our group used this method in some models.

3) Linear Regression Prediction

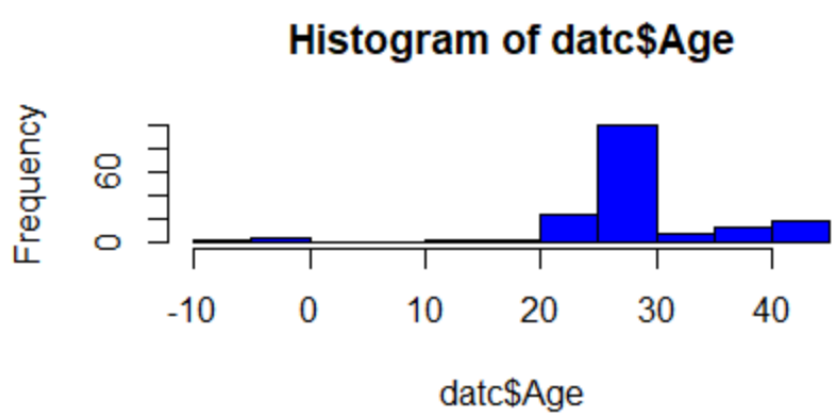
Checking the correlation from Age to other variables:




```
$p
         Pclass    SexN     Age    Fare   SibSp Parch
Pclass        0
SexN      3e-05       0
Age     1.8e-24   0.013       0
Fare      1e-58 6.4e-07    0.01       0
SibSp     0.073  0.0054 3.5e-17 0.00021       0
Parch      0.49 2.2e-11 3.6e-07 3.2e-08 1.8e-26     0
```

From the correlation table, we could see that these variables have some relationship with Age. So we could use these variables to build a linear regression to predict the missing Age.

```
aom<-lm(Age~Pclass+SexN+SibSp+Parch+Fare,data =dat1p)
datc<-dat1[is.na(dat1$Age),]
datc$Age<-predict(aom,datc)
```



However, we could also find out that the prediction of missing Age is very similar to that of method 2, that a large portion of values is very close to the total median. Although this method could bring a slight benefit in prediction, considering the complexity, we only use this method in logistic regression model because it could lead to a great improvement in logistic regression model. (But it does not improve a lot in other models)

In conclusion, these 3 methods are the methods we did in solving missing values in Age. Although method 2 and 3 is better than method 1, considering the complicatedness, we choose and use all these 3 methods in our models.

After solving the missing values and stage division in Age, we create dummy variables to make each stage a binary variable (1 for within the age range, and 0 for not).

③ Embarked:

As Embarked has only 2 mssing values, it has little influence in our models. We use mode as the filling value. So we use "s" as filling values.

```
Embarked
C:102
Q: 46
S:270
```

After fixing all the missing values and setting the dummy variables, we divided the dataset into 70% training data and 30% validation data.

Model:

After trying different models, our group selected the top 4 model with the highest performance. The three models are Ensemble, Random Forest, Bagging, and KNN. For Ensemble mode, we combine all of our model results to get. Considering the different characteristics of each model, we added different processing to each different method based on feature engineering.

1.Random Forest:

Score: 0.80861

The random forest model can be considered as a bagging model of classification tree model in some way. However, it doesn't use all the features of the data when constructing a single tree. In order to help the random forest model to perform better, we created some more features based on the Name, Sibling, and Parch. Title is a new feature based on the title of each person's name which is also mentioned in our Background. Specifically, people with title "Lady", "Dona", "Countess", "Sir", and "Jonkheer" will be regarded as Royalty. People with title "Capt", "Col", "Don", "Dr", "Major",and "Rev" will be regarded as Officer. Furthermore, we combined the number of sibling and parch into family size which is also used as a new feature in the following models.

```r
#3. Title
entire$Title <- gsub(".*,"," ", entire$Name)
entire$Title <- gsub("\\..*"," ", entire$Title)
entire$Title<-gsub(" ", "",entire$Title)
unique(entire$Title)

# Reassign mlle, ms, and mme, and Special Identity
entire$Title[entire$Title == "Mlle"]<- "Miss"
entire$Title[entire$Title == "Ms"] <- "Miss"
entire$Title[entire$Title == "Mme"]<- "Mrs"
entire$Title<-ifelse(entire$Title=="Dona"|entire$Title=="Lady"|entire$Title=="theCountess"|entire$Titl
                     |entire$Title=="Jonkheer","Royalty",entire$Title)
entire$Title<-ifelse(entire$Title=="Capt"|entire$Title=="Col"|entire$Title=="Don"|entire$Title=="Dr"
                     |entire$Title=="Major"|entire$Title=="Rev","Officer",entire$Title)

unique(entire$Title)
ggplot(entire[1:891,], aes(Title,fill = factor(Survived))) +
  geom_bar(stat = "count")+
  xlab('Title') +
  ylab("Count") +
  scale_fill_discrete(name = " Survived") +
  ggtitle("Title vs Survived")
```

After creating the new features, we began to test the model using 60% of the training data. 40% of the training data will be used as a validation set. The result shows an accuracy of 0.8235.

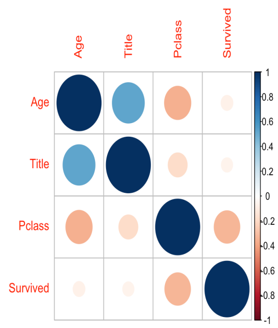
```r
set.seed(1)
rf_model <- randomForest(factor(Survived) ~ Pclass + Sex + Fare + Embarked +  Title +
                         FsizeRank, data = train.df)
prediction <- predict(rf_model, valid.df)
confusionMatrix(prediction,factor(valid.df$Survived))
```

```r

# random forest
library("randomForest")

set.seed(111)
rf_model <- randomForest(factor(Survived) ~ Pclass + Sex + Fare + Embarked + Title +
                         FsizeRank, data = train)

rf_model
prediction <- predict(rf_model, test)
solution <- data.frame(PassengerID = test$PassengerId,Survived = prediction)

```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 202  49
         1  14  92

               Accuracy : 0.8235
                 95% CI : (0.7799, 0.8616)
    No Information Rate : 0.605
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6141

 Mcnemar's Test P-Value : 1.839e-05
```

```
 Mcnemar's Test P-Value : 1.839e-05

            Sensitivity : 0.9352
            Specificity : 0.6525
         Pos Pred Value : 0.8048
         Neg Pred Value : 0.8679
             Prevalence : 0.6050
         Detection Rate : 0.5658
   Detection Prevalence : 0.7031
      Balanced Accuracy : 0.7938

       'Positive' Class : 0
```

Considering the correlation of the features, we decided to exclude some features which have highly-correlated one. The accuracy of using just Title and no Age is the same as just using Age and no Title. Although the accuracy of using both is a little bit high, the model will lose the predicting power caused by collinearity. In this situation, we decided to use the feature of Title because it contains the information of Name. The correlation plot is shown below.




```
##Correlation graph

train <- entire[1:891,]
test <- entire[892:1309,]

corr_data <- entire[1:891,]
corr_data$Title <- revalue(corr_data$Title,
                           c("Mr" = 1, "Master" = 2,"Officer" = 3,
                             "Mrs" = 4,"Royalty" = 5,"Miss" = 6))
corr_data$Title <- as.numeric(corr_data$Title)
corr_data$Pclass <- as.numeric(corr_data$Pclass)
corr_data$Survived <- as.numeric(corr_data$Survived)

corr_data <-corr_data[,c("Age","Title","Pclass","Survived")]
mcorr_data <- cor(corr_data)
corrplot(mcorr_data,method="circle")
```

We used the default parameters in the model. The number of trees is set to 500. The cutoff value is set to 1/k where k is the number of classes, 2. After using all the given training data in the random forest model, we got a prediction on the test dataset. The final model used on the test data got a score of 0.80861.

2. Bagging:

Score: 0.78947.

Firstly, we developed a classification tree model using training dataset. Using this model, we predict if people survived both in the training dataset and the validation dataset. The accuracy of both predictions are 0.9294 and 0.75 respectively as shown below.

```

#Run classification tree
Titanic.ct <- rpart(Survived ~ ., data = Train.df, method = "class",
               cp = 0.00001, minsplit = 5, xval = 15)
prp(Titanic.ct, type = 1, extra = 1, under = TRUE, split.font = 1,
    box.col=ifelse(Titanic.ct$frame$var == "<leaf>", 'gray', 'white'))
printcp(Titanic.ct)
length(Titanic.ct$frame$var[Titanic.ct$frame$var == "<leaf>"])

#Predict Validation Dataset and Show Accuracy
Titanic.ct.point.pred.train <- predict(Titanic.ct,Train.df,type = "class")
confusionMatrix(Titanic.ct.point.pred.train, Train.df$Survived)
Titanic.ct.point.pred.val <- predict(Titanic.ct,Valid.df,type = "class")
confusionMatrix(Titanic.ct.point.pred.val, Valid.df$Survived)

#Prune by cp of Smallest tree within 1 std. error of min. error
pruned.ct <- prune(Titanic.ct, cp = 0.0113475)
prp(pruned.ct, type = 1, extra = 1, under = TRUE, split.font = 1,
    box.col=ifelse(pruned.ct$frame$var == "<leaf>", 'gray', 'white'))

#Predict Validation Dataset and Show Accuracy
pruned.ct.point.pred.train <- predict(pruned.ct,Train.df,type = "class")
```

```
> #Predict Validation Dataset and Show Accuracy
> Titanic.ct.point.pred.train <- predict(Titanic.ct,Train.df,type = "class")
> confusionMatrix(Titanic.ct.point.pred.train, Train.df$Survived)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 368  27
         1  17 211

               Accuracy : 0.9294
                 95% CI : (0.9063, 0.9482)
    No Information Rate : 0.618
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8492

 Mcnemar's Test P-Value : 0.1748

            Sensitivity : 0.9558
            Specificity : 0.8866
         Pos Pred Value : 0.9316
         Neg Pred Value : 0.9254
             Prevalence : 0.6180
         Detection Rate : 0.5907
   Detection Prevalence : 0.6340
      Balanced Accuracy : 0.9212

       'Positive' Class : 0
```

```
> confusionMatrix(Titanic.ct.point.pred.val, Valid.df$Survived)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 136  39
         1  28  65

               Accuracy : 0.75
                 95% CI : (0.6937, 0.8007)
    No Information Rate : 0.6119
    P-Value [Acc > NIR] : 1.251e-06

                  Kappa : 0.4632

 Mcnemar's Test P-Value : 0.2218

            Sensitivity : 0.8293
            Specificity : 0.6250
         Pos Pred Value : 0.7771
         Neg Pred Value : 0.6989
             Prevalence : 0.6119
         Detection Rate : 0.5075
   Detection Prevalence : 0.6530
      Balanced Accuracy : 0.7271

       'Positive' Class : 0
```

Then, we pruned the model by cp at which the Xerror is within 1 std. error plus the minimum Xerror. We get the updated model and use the model to predict for both training dataset and validation dataset. The accuracy is 0.8587 and 0.806 respectively, which has been improved.

```
#Predict Validation Dataset and Show Accuracy
pruned.ct.point.pred.train <- predict(pruned.ct,Train.df,type = "class")
confusionMatrix(pruned.ct.point.pred.train, Train.df$Survived)
pruned.ct.point.pred.val <- predict(pruned.ct,Valid.df,type = "class")
confusionMatrix(pruned.ct.point.pred.val, Valid.df$Survived)
```

Next, we developed a bagging model using the training data to see if it performs better than the classification tree. I used the bagging model to predict validation dataset, and it shows that the accuracy is 0.8321, which is again higher than that of the classification tree model with pruning.

```
#bagging
bag<-bagging(Survived ~ ., data = Train.df)
bag_pred<-predict(bag, Valid.df, type = "class")
confusionMatrix(as.factor(bag_pred$class), Valid.df$Survived)
#Improve compared with Classification Tree after pruning
```

Finally, we developed a boosting model using the whole clean Titanic dataset to see if it performs better than the bagging model. I used the boosting model to predict the validation dataset, but the confusion matrix shows that the accuracy is 1, which shows that the problem of overfitting.

```
#boosting
boost<-boosting(Survived ~ ., data = Titanic.df)
boost_pred<-predict(boost, Valid.df, type = "class")
confusionMatrix(as.factor(boost_pred$class),Valid.df$Survived)
```

Therefore, we picked the bagging model as the most accurate model and used this model to predict the test dataset. The final accuracy of the prediction is 0.78947.

3. KNN

Score: 0.78468

Based on the characteristics of KNN, in this model we only filled the missing value of Age, and did not set dummy variable for it during the data preprocess. Also, in order to offset the impact of variables' value magnitude on classification, we normalized the data.

```
#initialize normalizated training, validatation data
train.norm<-train.data
valid.norm<-valid.data
data.norm<-data2

test.norm<-test

library(caret)
norm.values<-preProcess(train.data[,-13],method=c('center','scale'))

train.norm[, -13] <- predict(norm.values, train.data[, -13])
valid.norm[, -13] <- predict(norm.values, valid.data[, -13])
data.norm[, -13] <- predict(norm.values, data2[, -13])

test.norm[,1:13]<-predict(norm.values,test[,1:13])
```

class_1	Pclass_2	Pclass_3	Sex	Age	SibSp	Parch	Fare	Embarked_C	Embarked_Q	Embarked_S	Cabin01	Surv
1.659938	-0.4825138	-1.0932924	1.4128888	0.63368751	0.3663616	-0.4937325	0.712032137	1.939279	-0.2956558	-1.5622363	1.7612134	1
-0.601304	-0.4825138	0.9129555	-0.7064444	0.40945263	-0.4487929	-0.4937325	-0.511253128	-0.514690	-0.2956558	0.6389093	-0.5667271	0
-0.601304	-0.4825138	0.9129555	-0.7064444	-0.11376211	-0.4487929	-0.4937325	-0.503354326	-0.514690	3.3759771	-1.5622363	-0.5667271	0
1.659938	-0.4825138	-1.0932924	1.4128888	2.12858677	-0.4487929	-0.4937325	-0.153359781	-0.514690	-0.2956558	0.6389093	1.7612134	1
-0.601304	-0.4825138	0.9129555	-0.7064444	-0.71172182	-0.4487929	-0.4937325	-0.511253128	-0.514690	-0.2956558	0.6389093	-0.5667271	0
-0.601304	-0.4825138	0.9129555	-0.7064444	0.70843248	0.3663616	5.6377359	-0.061951885	-0.514690	-0.2956558	0.6389093	-0.5667271	0
-0.601304	-0.4825138	0.9129555	1.4128888	-1.16019159	-0.4487929	-0.4937325	-0.515040994	-0.514690	-0.2956558	0.6389093	-0.5667271	0
-0.601304	2.0685987	-1.0932924	1.4128888	1.90435188	-0.4487929	-0.4937325	-0.357455717	-0.514690	-0.2956558	0.6389093	-0.5667271	1
-0.601304	-0.4825138	0.9129555	-0.7064444	-2.05713114	2.8118250	0.7325612	-0.103544896	-0.514690	3.3759771	-1.5622363	-0.5667271	0
-0.601304	2.0685987	-1.0932924	-0.7064444	0.40945263	-0.4487929	-0.4937325	-0.163999853	-0.514690	-0.2956558	0.6389093	-0.5667271	0
-0.601304	-0.4825138	0.9129555	1.4128888	-1.08544663	-0.4487929	-0.4937325	-0.511655517	-0.514690	3.3759771	-1.5622363	-0.5667271	1
-0.601304	-0.4825138	0.9129555	1.4128888	-1.60866137	1.9966705	0.7325612	-0.259276866	-0.514690	-0.2956558	0.6389093	-0.5667271	0
-0.601304	-0.4825138	0.9129555	-0.7064444	-0.11376211	-0.4487929	-0.4937325	-0.527213237	1.939279	-0.2956558	-1.5622363	-0.5667271	0
-0.601304	-0.4825138	0.9129555	1.4128888	-0.11376211	-0.4487929	-0.4937325	-0.514557355	-0.514690	3.3759771	-1.5622363	-0.5667271	1

Then We ran the KNN model using training dataset and validation dataset for different K, and we find the best k is 5 with the accuracy 81.79%. Using all "train" data to predict the test data. The result is 0.70813.

```
#KNN-First run
library(FNN)
accuracy.data <- data.frame(k = seq(1, 20, 1), accuracy = rep(0, 20))
library(caret)
# compute knn for different k on validation.
for(i in 1:20) {
  knn.pred <- FNN::knn(train.norm[, c(-13,-14)], valid.norm[, c(-13,-14)],
                       cl = train.norm[, 13], k = i)
  accuracy.data[i, 2] <- confusionMatrix(knn.pred, valid.norm[, 13],positive="1")$overall[1]
}
accuracy.data
plot(accuracy~k,accuracy.data,type="l")

knn.pred.test<-knn(data.norm[,c(-13,-14)],test.norm[,1:12],cl=data.norm[,13],k=5)

a<-as.data.frame(knn.pred.test)
PassengerId<-data_test[,1]
knn_csv<-cbind(PassengerId,a)
colnames(knn_csv)[2]<-'Survived'
write.csv(knn_csv,"knn5.csv",row.names=FALSE)

knn.pred <- FNN::knn(train.norm[, c(-13,-14)], valid.norm[, c(-13,-14)],
                     cl = train.norm[, 13], k = 5)
confusionMatrix(knn.pred, valid.norm[, 13],positive="1")
```

```
          Reference
Prediction   0   1
         0 192  39
         1  26 100

               Accuracy : 0.8179
                 95% CI : (0.7739, 0.8566)
    No Information Rate : 0.6106
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6105

 Mcnemar's Test P-Value : 0.1366

            Sensitivity : 0.7194
            Specificity : 0.8807
         Pos Pred Value : 0.7937
         Neg Pred Value : 0.8312
```

In order to improve the accuracy of this model, we thought of creating a new variable. For Parch and Sibsp, they all belong to the family members, which can give us information about whether this passenger is alone or not alone. Therefore, we created a new variable named “Alone” to show whether this person embarked by himself/herself or with her/his family. Now the variables are shown below:

```
#Combine sis and parh to new variable
data2$alone<-ifelse(data2$SibSp==0 & data2$Parch==0,1,0)
test$alone<-ifelse(test$SibSp==0 & test$Parch==0,1,0)
```

Pclass_2	Pclass_3	Sex	Age	SibSp	Parch	Fare	Embarked_C	Embarked_Q	Embarked_S	Cabin01	Survived	alone
1	0	1	14.00	1	0	30.0708	1	0	0	0	1	0
0	1	1	4.00	1	1	16.7000	0	0	1	1	1	0
0	0	1	58.00	0	0	26.5500	0	0	1	1	1	1
0	1	0	20.00	0	0	8.0500	0	0	1	0	0	1
0	1	0	39.00	1	5	31.2750	0	0	1	0	0	0
0	1	1	14.00	0	0	7.8542	0	0	1	0	0	1
1	0	1	55.00	0	0	16.0000	0	0	1	0	1	1
0	1	0	2.00	4	1	29.1250	0	1	0	0	0	0
1	0	0	28.00	0	0	13.0000	0	0	1	0	1	1

Then we rerun this model and got the best k =11, with the accuracy 81.51%. Using all training data to predict the test data. The result is 0.78468.

```
accuracy.data <- data.frame(k = seq(1, 20, 1), accuracy = rep(0, 20))
library(caret)
# compute knn for different k on validation.
for(i in 1:20) {
  knn.pred <- FNN::knn(train.norm[, -13], valid.norm[, -13],
                       cl = train.norm[, 13], k = i)
  accuracy.data[i, 2] <- confusionMatrix(knn.pred, valid.norm[, 13],positive="1")$overall[1]
}
accuracy.data
plot(accuracy~k,accuracy.data,type="l")

knn.pred <- FNN::knn(train.norm[, -13], valid.norm[, -13],
                     cl = train.norm[, 13], k = 11)
confusionMatrix(knn.pred, valid.norm[, 13],positive="1")


#k=11


knn.pred.test<-knn(data.norm[,-13],test.norm[,1:13],cl=data.norm[,13],k=11)

a<-as.data.frame(knn.pred.test)
PassengerId<-data_test[,1]
knn_csv<-cbind(PassengerId,a)
```

4. Model Ensemble

After trying different models to get the prediction, we considered implementing a model ensemble to improve our prediction. We generated our prediction for the test dataset in each method, and we tend to combine all the predictions together, generating the final prediction. The two common methods for the model ensemble are the majority vote and setting cut-off value. We used the majority vote first, combining all the models' prediction: random forest, knn, bagging, boosting, logistic regression, neural network, classification tree, and naive bayes. However, the final prediction performs no better than our best model random forest. So we tried setting cut-off value instead. Since few people are survivors in titanic, we set the not survived as the important class. And after several trials, we set the cut-off value as 0.4, and got the result slightly improved to 0.813.

Summary:

When solving a problem, the process of data preprocessing is even more important than building a model. There does not have the best model, there only has a proper model for a certain dataset. Different models have different characteristics, so the preprocessing of the same dataset might be different. Using the same model for the same dataset, the prediction accuracy may have a big difference due to the different feature engineering. For the ensemble model, it is a good way to help us to mitigate the overfitting and get better results than individual models. However, the ensemble model is also a black-box way, so sometimes it will not guarantee a better performance for the prediction.