Data Mining Homework3(Titanic)

In this homework, I consider the requirement of "classification" is equal to supervised learning and the requirement of "clustering" is equal to unsupervised learning. I choose the titanic data set which contains part of a list of passengers on Titanic. The list contains the personal information of the passengers including age, Name, sex, Cabin, family, fare and whether they survived the that famous disaster. The target of machine learning is to predict whether a specific person would survived if his personal information is given. However, it seems that the Kaggle does not offer a test set whit the correct test label. So, I only use the training set and use resubstitution error to measure the quality of a model. I also use 10-fold cross-validation to optimize the parameters of model in order to avoid over-fitting. Besides, the cross-validation error can also be considered as an indication of the quality of the model.

Data Import and Pre-processing

Data Import and Summary

First, import the package necessary for data pre-processing.

```
library(dplyr)
```

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

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

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

```
library(plyr)
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

The data titanic comes from Kaggle. It's part of the list of passengers on titanic. From the summary below we know it contains

```
titanic_train<-read.csv("train.csv")
summary(titanic_train)
```

```
##   PassengerId       Survived          Pclass
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000
```

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

Data Pre-processing: Fare_Per_Person

The fare attribute is indeed the price per ticket but what we need is the fare per person, so we have to check the ticket number to find out whether there exist people share a ticket with others.

```
ticket_counts<-plyr::count(titanic_train[["Ticket"]])
ticket_multi<-ticket_counts[which(ticket_counts$freq>1),]
ticket_multi[1:10,]
```

```
##         x freq
## 1  110152    3
## 2  110413    3
## 3  110465    2
## 8  111361    2
## 29 113505    2
## 33 113572    2
## 34 113760    4
## 37 113776    2
## 38 113781    4
## 44 113789    2
```

After having a look at the ticket_multi, we recognize that many people share a ticket number with others, which means that the "fare" of these people is actually the fare of more than one person. This must be fixed.

```
titanic_train[["Fare_Per_Person"]]<-titanic_train[["fare"]]
for (i in 1:42) {
  titanic_train[which(as.character(titanic_train[["Ticket"]])==as.character(ticket_multi[i,1])),"Fare_P
}
titanic_train[which(is.na(titanic_train[["Fare_Per_Person"]])),"Fare_Per_Person"]<-titanic_train[which(
```

```
aggregate(x=titanic_train[c('Fare_Per_Person')], by = list(titanic_train$Embarked, titanic_train$Pclass
```

```
##    Group.1 Group.2 Fare_Per_Person
## 1                1        40.00000
## 2        C       1        95.26005
## 3        Q       1        45.00000
## 4        S       1        44.05269
## 5        C       2        23.58946
## 6        Q       2        12.35000
## 7        S       2        18.30915
## 8        C       3        11.21408
## 9        Q       3        11.18339
## 10       S       3        13.68381
```

Data Pre-processing: Extraction of Title From Name

At first I thought the names of passengers have nothing to do with their possibility of survival. But then I noticed that there are NA values in "age" and the names contain the title of passengers, which can help to fill the missing values in age.

```
names <- titanic_train$Name
title <-  gsub("^.*, (.*?)\\..*$", "\\1", names)

titanic_train$title <- title

table(title)
```

```
## title
##         Capt          Col          Don           Dr     Jonkheer
##            1            2            1            7            1
##         Lady        Major       Master         Miss         Mlle
##            1            2           40          182            2
##          Mme           Mr          Mrs           Ms          Rev
##            1          517          125            1            6
##          Sir the Countess
##            1            1
```

The most titles are those directly relative to age and sex, which is exactly what we want. And rare titles which also tell us about sex and age of a person are converted to the common titles of the same meaning. But rare titles like "Don" which tell us nothing are simply refereed as "Rare Title".

```
titanic_train$title[titanic_train$title == 'Mlle']<- 'Miss'
titanic_train$title[titanic_train$title == 'Ms']<- 'Miss'
titanic_train$title[titanic_train$title == 'Mme']<- 'Mrs'
titanic_train$title[titanic_train$title == 'Lady']<- 'Miss'
titanic_train$title[titanic_train$title == 'Dona']<- 'Miss'
```

```
titanic_train$title[!(titanic_train$title %in% c('Master', 'Miss', 'Mr', 'Mrs'))] <- "Rare Title"
titanic_train$title <- as.factor(titanic_train$title)
```

Data Pre-processing: Extraction of Family From Sibsp and Parch

The attribute "Sibsp" and "Parch" both contain the family information of a person. We combine the two attribute into "Family", which is "Sibsp" + "Parch" + 1(the person himself).

```
titanic_train[["Family"]]<-titanic_train$SibSp+titanic_train$Parch+1
```

Data Pre-processing: Dealing With Missing Age Values

Use linear model to fit the age. We assume that age is somewhat relative to Pclass, Sex, SibSp, Parch and title.

```
AgeLM <- lm(Age ~ Pclass + Sex + SibSp + Parch + title, data=titanic_train[!is.na(titanic_train$Age),])
summary(AgeLM)
```

```
##
## Call:
## lm(formula = Age ~ Pclass + Sex + SibSp + Parch + title, data = titanic_train[!is.na(titanic_train$A
##     ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -24.910  -8.052  -1.154   6.419  44.593
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      15.9630     8.6955   1.836 0.066812 .
## Pclass           -5.5975     0.5220 -10.724  < 2e-16 ***
## Sexmale           8.0178     8.3658   0.958 0.338189
## SibSp            -1.9364     0.5294  -3.657 0.000274 ***
## Parch            -0.5020     0.5630  -0.892 0.372946
## titleMiss        19.4852     8.6434   2.254 0.024480 *
## titleMr          22.2187     2.2246   9.988  < 2e-16 ***
## titleMrs         32.7421     8.6553   3.783 0.000168 ***
## titleRare Title  30.6345     3.3962   9.020  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.25 on 705 degrees of freedom
## Multiple R-squared:  0.4072, Adjusted R-squared:  0.4005
## F-statistic: 60.55 on 8 and 705 DF,  p-value: < 2.2e-16
```

```
titanic_train$AgeLM <- predict(AgeLM, titanic_train)
```

Have a look at the result of linear model.

```
titanic_train[(is.na(titanic_train$Age) & titanic_train$AgeLM <18), c('Sex', 'SibSp', 'Parch', 'title',
```

```
##        Sex SibSp Parch  title Pclass Survived      AgeLM
## 66    male     1     1 Master      3        1  4.7500184
## 110 female     1     0   Miss      3        1 16.7194176
## 129 female     1     1   Miss      3        1 16.2174446
## 160   male     8     2 Master      3        0 -9.3067295
## 177   male     3     1 Master      3        0  0.8772256
## 181 female     8     2   Miss      3        0  2.1606967
## 202   male     8     2     Mr      3        0 12.9119212
```

```
## 230 female     3     1   Miss      3        0 12.3446518
## 241 female     1     0   Miss      3        0 16.7194176
## 242 female     1     0   Miss      3        1 16.7194176
## 325   male     8     2     Mr      3        0 12.9119212
## 331 female     2     0   Miss      3        1 14.7830212
## 410 female     3     1   Miss      3        0 12.3446518
## 486 female     3     1   Miss      3        0 12.3446518
## 594 female     0     2   Miss      3        0 17.6518679
## 613 female     1     0   Miss      3        1 16.7194176
## 710   male     1     1 Master      3        1  4.7500184
## 793 female     8     2   Miss      3        0  2.1606967
## 847   male     8     2     Mr      3        0 12.9119212
## 864 female     8     2   Miss      3        0  2.1606967
## 889 female     1     2   Miss      3        0 15.7154715
```

```
titanic_train[["Age_Na_Free"]]<-titanic_train$Age
titanic_train[which(is.na(titanic_train[["Age_Na_Free"]])),"Age_Na_Free"]<-titanic_train[which(is.na(ti
```

From the result above we notice that some missing ages filled by the linear model can be ridiculous. But using the age bands to replace actual number of ages can fix that. Because assumed that linear model is somewhat correct, the actual age of the person whose AgeLM is smaller than 0 can not too big. So we can use the age band "child" to describe that.

```
titanic_train[["Age_Band"]]<-titanic_train$Age
for (i in 1:891) {
  if(titanic_train[i,"Age_Na_Free"]<=13){
    titanic_train[i,"Age_Band"]<-"Child"
  }

  if(titanic_train[i,"Age_Na_Free"]>13 & titanic_train[i,"Age_Na_Free"]<=17){
    titanic_train[i,"Age_Band"]<-"Teens"
  }

  if(titanic_train[i,"Age_Na_Free"]>17 & titanic_train[i,"Age_Na_Free"]<=26){
    titanic_train[i,"Age_Band"]<-"Young"
  }

  if(titanic_train[i,"Age_Na_Free"]>26 & titanic_train[i,"Age_Na_Free"]<=42){
    titanic_train[i,"Age_Band"]<-"Mid_Age"
  }

  if(titanic_train[i,"Age_Na_Free"]>42 & titanic_train[i,"Age_Na_Free"]<=57){
    titanic_train[i,"Age_Band"]<-"Older_Than_Mid_Age"
  }

  if(titanic_train[i,"Age_Na_Free"]>57 ){
    titanic_train[i,"Age_Band"]<-"Old"
  }

}
```

Data Pre-processing: Build The Final Data Set verson 1

The final data set contains title, Fare_Per_Person, Family, Pclass, Sex, Age_Band and Survived. We will use the decision tree method and this data is perfect for a decision tree model or a random forest. However if we want to use some model which can only deal with "numeric" attributes, this data set still need further processing. The further processing is probably about changing the nominal attributes to numeric ones and the normalization of them.

```
titanic_ready_1<-titanic_train[,c("title","Fare_Per_Person","Family","Pclass","Sex","Age_Band","Survive
```

Data Nomailzation

In this part, there are two targets: change the nominal attribute to numeric, normalize the data. First, change the nominal attribute "Age_Band" and Sex to numeric attributes. In "Age_Band", the values are changed to numbers 1 to 6 to describe the 6 bands. In "Sex", use 1 to stand for "male" and 0 to stand for "female".

```
titanic_ready_3<-titanic_ready_1
```

```
titanic_train[["Age_Band_Numeric"]]<-titanic_train$Age
for (i in 1:891) {
  if(titanic_train[i,"Age_Na_Free"]<=13){
    titanic_train[i,"Age_Band_Numeric"]<-1
  }

  if(titanic_train[i,"Age_Na_Free"]>13 & titanic_train[i,"Age_Na_Free"]<=17){
    titanic_train[i,"Age_Band_Numeric"]<-2
  }

  if(titanic_train[i,"Age_Na_Free"]>17 & titanic_train[i,"Age_Na_Free"]<=26){
    titanic_train[i,"Age_Band_Numeric"]<-3
  }

  if(titanic_train[i,"Age_Na_Free"]>26 & titanic_train[i,"Age_Na_Free"]<=42){
    titanic_train[i,"Age_Band_Numeric"]<-4
  }

  if(titanic_train[i,"Age_Na_Free"]>42 & titanic_train[i,"Age_Na_Free"]<=57){
    titanic_train[i,"Age_Band_Numeric"]<-5
  }

  if(titanic_train[i,"Age_Na_Free"]>57 ){
    titanic_train[i,"Age_Band_Numeric"]<-6
  }

}

titanic_ready_3[["Age_Band_Numeric"]]<-titanic_train$Age_Band_Numeric
titanic_ready_3$Age_Band_Numeric<-as.numeric(titanic_ready_3$Age_Band_Numeric)
```

```
titanic_train[["Sex_Numeric"]]<-titanic_train$Sex
titanic_train$Sex_Numeric<-as.numeric(titanic_train$Sex_Numeric)
for (i in 1:891) {
  if(as.character(titanic_train[i,"Sex"])=="male"){
    titanic_train[i,"Sex_Numeric"]<- 1
```

```
  }

  if(as.character(titanic_train[i,"Sex"])=="female"){
    titanic_train[i,"Sex_Numeric"]<- 0
  }



}

titanic_ready_3[["Sex_Numeric"]]<-titanic_train$Sex_Numeric
```

```
titanic_ready_3<-titanic_ready_3[,c("Fare_Per_Person","Family","Pclass","Sex_Numeric","Age_Band_Numeric
```

Begin normalization.

```
for (i in 1:5) {
  titanic_ready_3[,i]<-scale(titanic_ready_3[,i],center = TRUE, scale = TRUE)
}
```

Classification(Survived Or Not)

In this section, two method are presented. I choose the SVM model in package"e1071" and decision tree model in package "rpart". The decision tree is a model that can be quite easy to visualized and the SVM model is famous for its simple algorithm and strong power. That's why I choose them. Besides, both packages have an build_in function for 10-fold-cross-validation, which is very convenient.

Decision Tree Classification

First we should convert the "Age_Band" and "Survived" to factor, which is the requirement of training a decision tree model. Then a decision tree is trained. But this tree can not be the final model used to make prediction. Some optimization is needed.

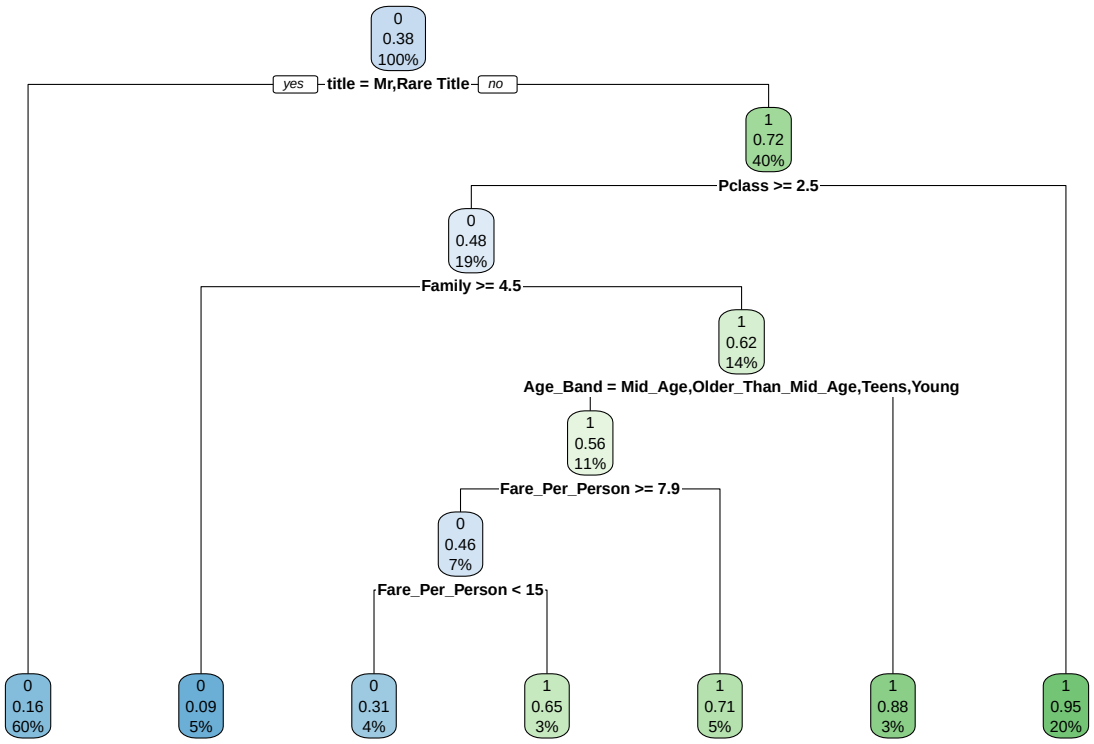
```
titanic_ready_1[,"Age_Band"]<-as.factor(titanic_ready_1$Age_Band)
titanic_ready_1[,"Survived"]<-as.factor(titanic_ready_1$Survived)
library(rpart)
decision_tree_1=rpart(Survived~.,data=titanic_ready_1,method="class",control = rpart.control(minsplit =
# a original tree is trained. The "xval" means the number of folds of cross-validation.

print(decision_tree_1)
```

```
## n= 891
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 891 342 0 (0.61616162 0.38383838)
##     2) title=Mr,Rare Title 539  88 0 (0.83673469 0.16326531) *
##     3) title=Master,Miss,Mrs 352  98 1 (0.27840909 0.72159091)
##       6) Pclass>=2.5 172  83 0 (0.51744186 0.48255814)
##        12) Family>=4.5 45   4 0 (0.91111111 0.08888889) *
##        13) Family< 4.5 127  48 1 (0.37795276 0.62204724)
##          26) Age_Band=Mid_Age,Older_Than_Mid_Age,Teens,Young 102  45 1 (0.44117647 0.55882353)
```

```
##            52) Fare_Per_Person>=7.8875 61  28 0 (0.54098361 0.45901639)
##              104) Fare_Per_Person< 14.8729 35  11 0 (0.68571429 0.31428571) *
##              105) Fare_Per_Person>=14.8729 26   9 1 (0.34615385 0.65384615) *
##            53) Fare_Per_Person< 7.8875 41  12 1 (0.29268293 0.70731707) *
##          27) Age_Band=Child,Old 25   3 1 (0.12000000 0.88000000) *
##       7) Pclass< 2.5 180   9 1 (0.05000000 0.95000000) *
```

Visualize the original tree.

```
library(rpart.plot)
rpart.plot(decision_tree_1)
```



Import the package caret which is necessary for following procedures. Form the output of the script below, we know that the accuracy of cross-validation can be calculated as (root node error)*xerror, which is about 17%. It seems that the effect is not bad.

```
library(caret)
```

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

```
## Loading required package: ggplot2
```

```
perdict_DT=predict(decision_tree_1,data=titanic_ready_1,type="class")

printcp(decision_tree_1)
```

```
##
## Classification tree:
## rpart(formula = Survived ~ ., data = titanic_ready_1, method = "class",
```

```
##     control = rpart.control(minsplit = 5, xval = 10))
##
## Variables actually used in tree construction:
## [1] Age_Band        Family          Fare_Per_Person Pclass
## [5] title
##
## Root node error: 342/891 = 0.38384
##
## n= 891
##
##         CP nsplit rel error  xerror     xstd
## 1 0.456140      0   1.00000 1.00000 0.042446
## 2 0.054094      1   0.54386 0.54386 0.035472
## 3 0.012671      3   0.43567 0.46199 0.033336
## 4 0.010000      6   0.39766 0.47368 0.033663
```

We choose the cp value with the lowest xerror and use this value to prune the original tree. This can avoid the problem of over-fitting.

```
decision_tree_2<-prune(decision_tree_1, cp= decision_tree_1$cptable[which.min(decision_tree_1$cptable[,
```

Use this tree to make the prediction. In the confusion matrix below, the quality of the model is shown clearly. The total accuracy is about 83% with a kappa of about 0.64. The sensitivity and specificity is also OK. The ROC plot below confirms that the quality of this model is acceptable.

```
perdict_DT_2=predict(decision_tree_2,data=titanic_ready_1,type="class")
confusionMatrix(perdict_DT_2,titanic_ready_1$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 492  92
##          1  57 250
##
##                Accuracy : 0.8328
##                  95% CI : (0.8066, 0.8567)
##     No Information Rate : 0.6162
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6395
##  Mcnemar's Test P-Value : 0.005346
##
##             Sensitivity : 0.8962
##             Specificity : 0.7310
##          Pos Pred Value : 0.8425
##          Neg Pred Value : 0.8143
##              Prevalence : 0.6162
##          Detection Rate : 0.5522
##    Detection Prevalence : 0.6554
##       Balanced Accuracy : 0.8136
##
##        'Positive' Class : 0
##
```

```r
printcp(decision_tree_2)
```

```
##
## Classification tree:
## rpart(formula = Survived ~ ., data = titanic_ready_1, method = "class",
##     control = rpart.control(minsplit = 5, xval = 10))
##
## Variables actually used in tree construction:
## [1] Family Pclass title
##
## Root node error: 342/891 = 0.38384
##
## n= 891
##
##         CP nsplit rel error  xerror     xstd
## 1 0.456140      0   1.00000 1.00000 0.042446
## 2 0.054094      1   0.54386 0.54386 0.035472
## 3 0.012671      3   0.43567 0.46199 0.033336
```

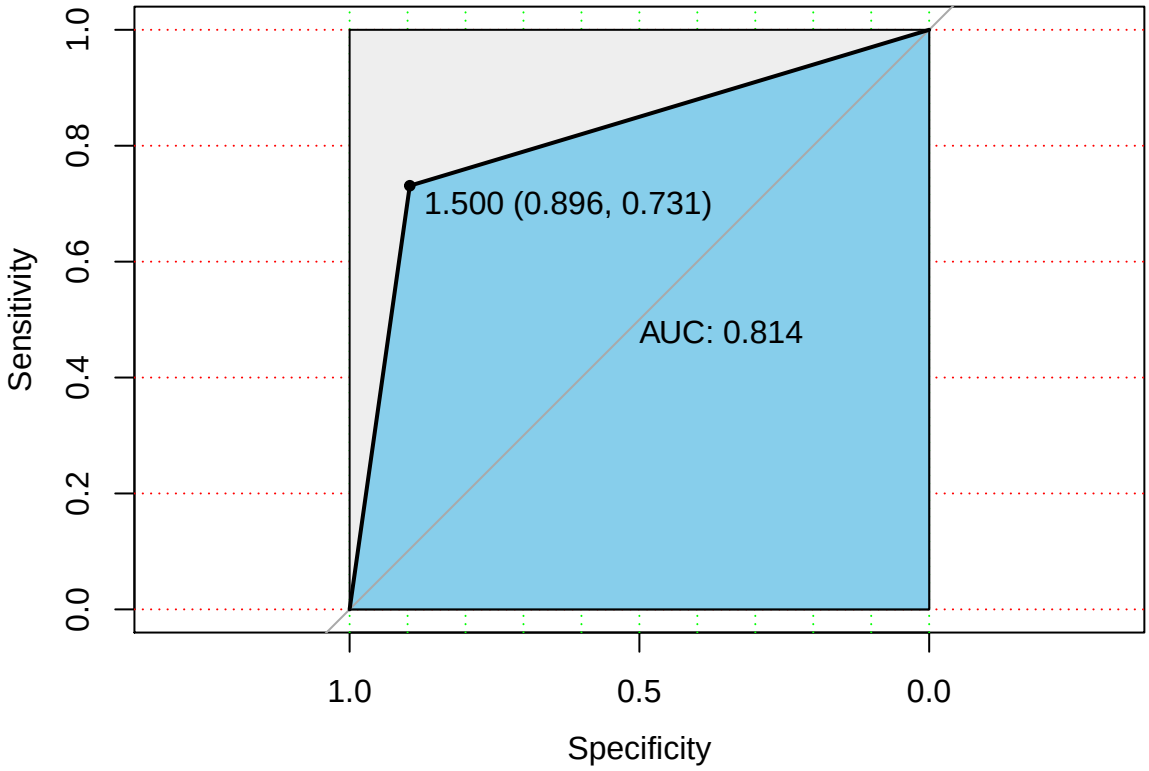
```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

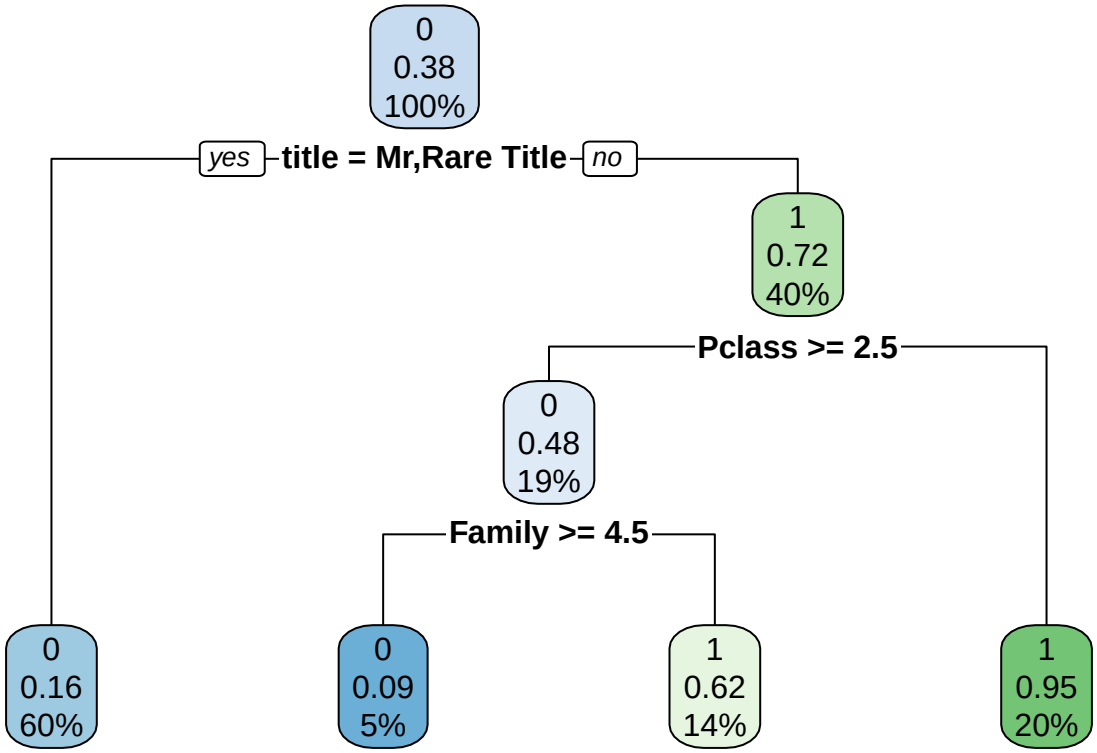
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
modelroc <- roc(titanic_ready_1$Survived,as.numeric((perdict_DT_2)))
plot(modelroc, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE)
```



From the decision tree below we can get some useful information about the survival rate of a Titanic passenger. If his title was Mr or something rare, which means he was probably a man, his chance to survived would be small. Also, People of "smaller Pclass", which means they were probably rich or noble people, have better chance to survived. The family size of a passenger also affects his possibility of survival.

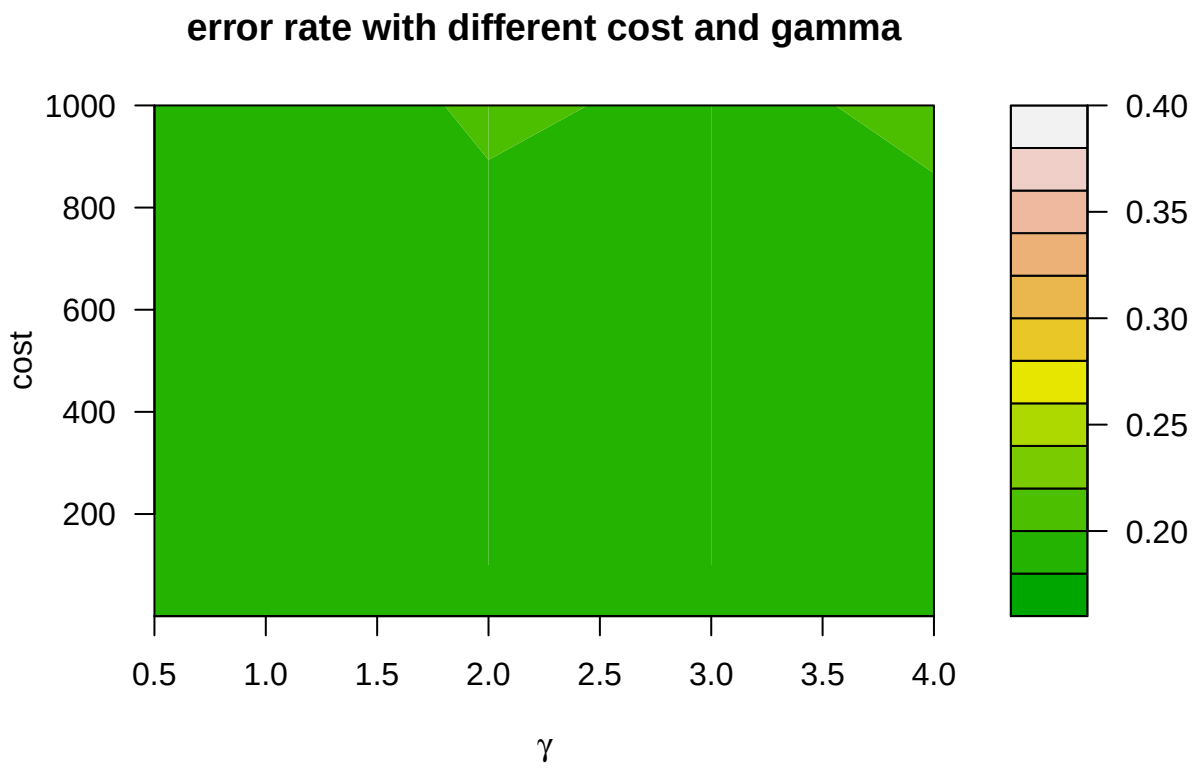
```
rpart.plot(decision_tree_2)
```



SVM

The package e1071 is imported and a series of SVM models of different cost and gamma are trained. A plot is created to visualize the accuracy of all models with different parameters. The model with the best accuracy is chosen as the best model for prediction.

```
library(e1071)
titanic_ready_3$Survived<-as.factor(titanic_ready_3$Survived)
tObj<-tune.svm(Survived~.,data=titanic_ready_3,type="C-classification",kernel="radial",     cost=c(0.001

## Visualize the accuracy of all models with different parameters
plot(tObj,xlab=expression(gamma),ylab="cost",main="error rate with different cost and gamma",nlevels=10
```




```
BestSvm<-tObj$best.model
```

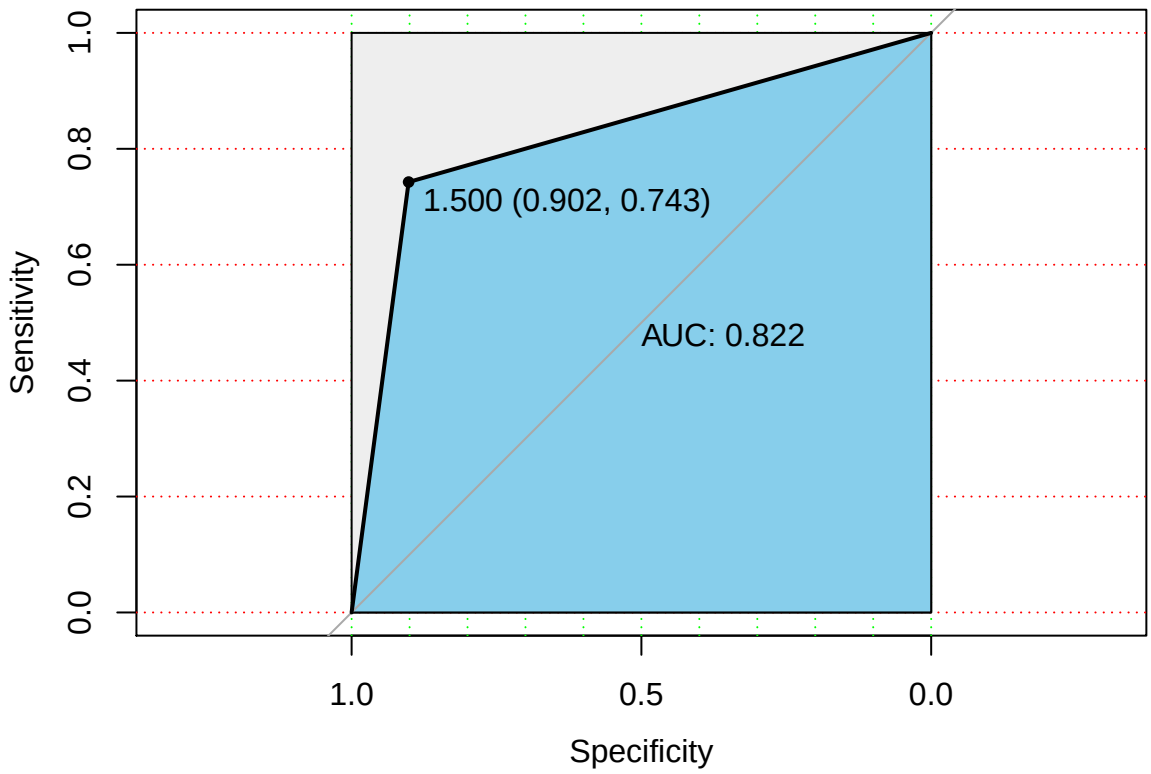
Use the best model to make the prediction and analysis the quality of the model. The total accuracy is about 83%, and the kappa is about 0.65. This model is also not bad. The sensitivity, specificity and the ROC plot are also presented.

```
tit_svm_pre<-predict(BestSvm,titanic_ready_3)
confusionMatrix(tit_svm_pre,titanic_ready_3$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 495  88
##          1  54 254
##
##                Accuracy : 0.8406
##                  95% CI : (0.8149, 0.8641)
##     No Information Rate : 0.6162
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6566
##  Mcnemar's Test P-Value : 0.005618
##
##             Sensitivity : 0.9016
##             Specificity : 0.7427
##          Pos Pred Value : 0.8491
```

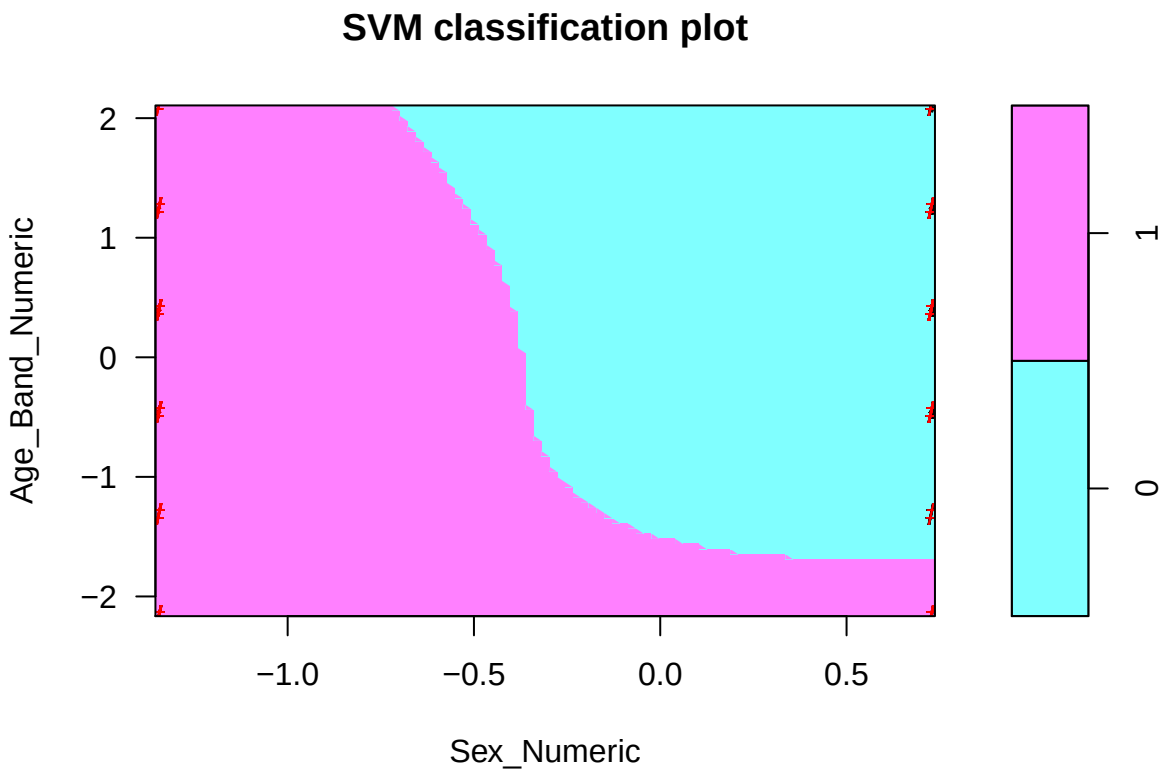
```
##          Neg Pred Value : 0.8247
##              Prevalence : 0.6162
##          Detection Rate : 0.5556
##    Detection Prevalence : 0.6543
##       Balanced Accuracy : 0.8222
##
##        'Positive' Class : 0
##
```

```
library(pROC)
modelroc <- roc(titanic_ready_3$Survived,as.numeric(tit_svm_pre))
plot(modelroc, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE)
```

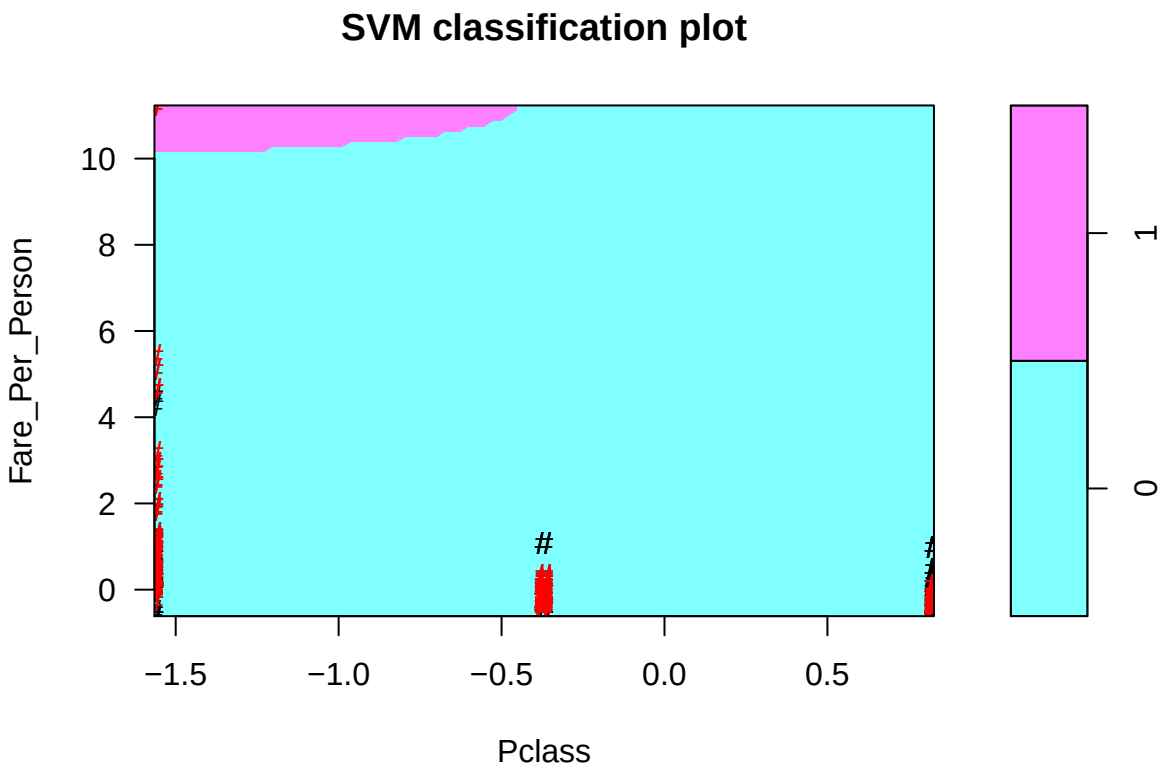


We use the following 2 plot to show the information we get from the model. The age vs sex plot shows that female are young children were likely to survived. The Fare_Per_Person vs Pclass plot shows that only rich or noble people were likely to survived. So we can make the conclusion that female and children in the "rich or noble" group had better chance to survived. But in the "not so rich or not so noble" group, the possibility of survival could be terrible. The famous law of "female or child leave first" probably only existed in "rich or noble" group.

```
plot(x=BestSvm,data=titanic_ready_3,formula=Age_Band_Numeric~Sex_Numeric,svSymbol="#",dataSymbol="*",gr
```




```
plot(x=BestSvm,data=titanic_ready_3,formula=Fare_Per_Person~Pclass,svSymbol="#",dataSymbol="*",grid=100)
```



Clustering

In this section, 2 method are presented to build unsupervised learning model. The target is still about "Survived" but is a little different from the supervised learning models above. Here we want to use clustering method to divide the data set into 2 groups. We will see whether the 2 group obtained can reflect the survived status of passengers.

Density-Based Spatial Clustering of Applications with Noise(DBSCAN)

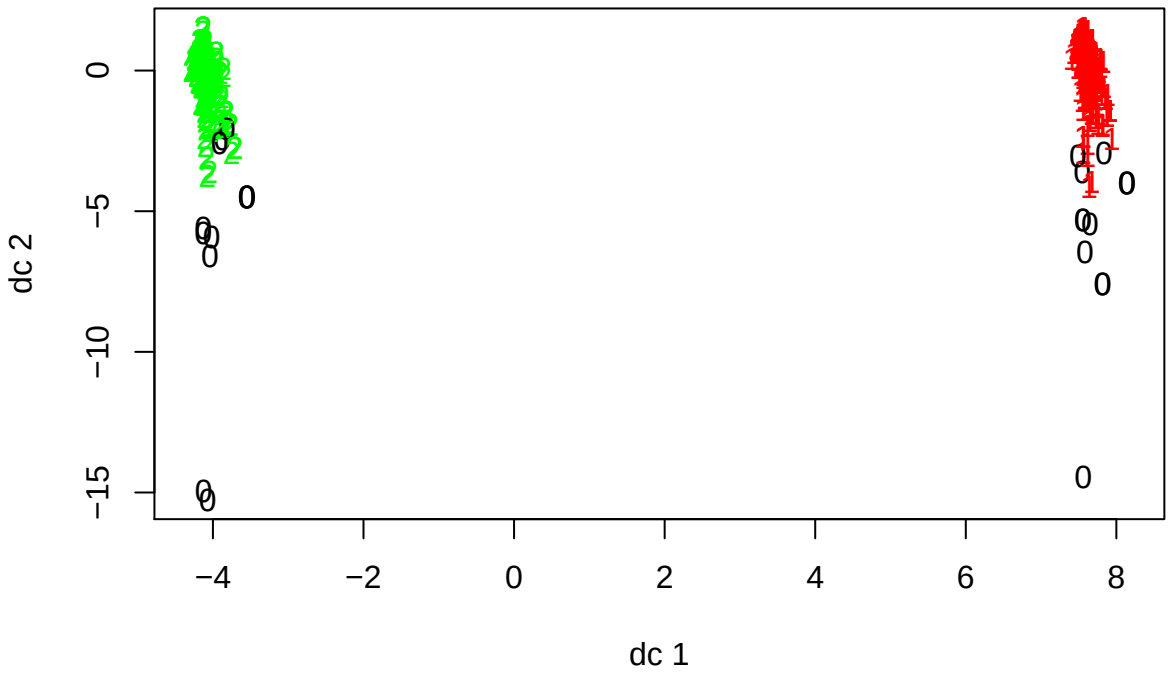
The first clustering model is famous DBSCAN. After setting the eps and MinPts and given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). The table below shows that the clustering result of DBSCAN is similar to the groups divided by survived or not. The data point with class "0" are outliers according to the DBSCAN. The "accuracy" is about 78%, not as good as SVM and decision tree model above. But since this is unsupervised learning, I think the accuracy is not too bad. The visualization of the clustering is presented. The points are projected on a 2-D plot.

```
library(fpc)
items<-sample(1:891,891)
 data=titanic_ready_3[items,1:5]
 tit_ds<-dbscan(data,eps=1.7,MinPts = 15)
 table(tit_ds$cluster,titanic_ready_3$Survived[items])
```

```
##
##       0   1
```

```
##    0  15  11
##    1  77 224
##    2 457 107
```

```
plotcluster(data,tit_ds$cluster)
```



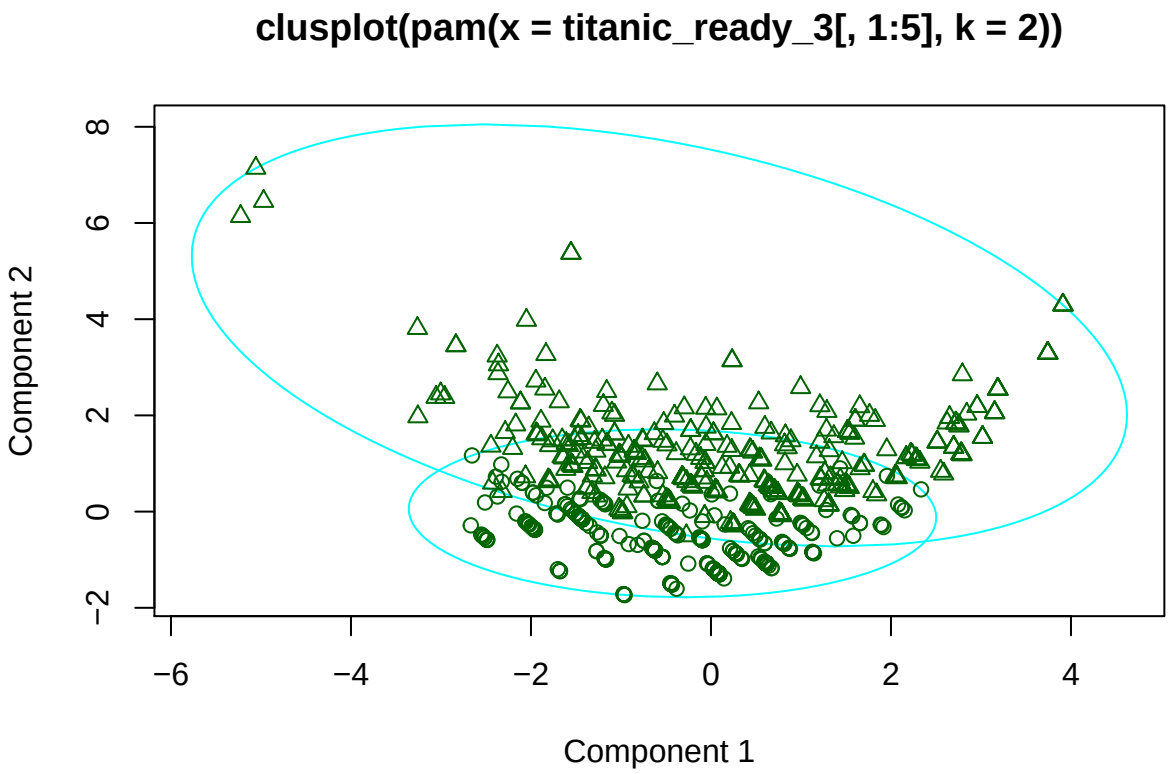
K-Medoids

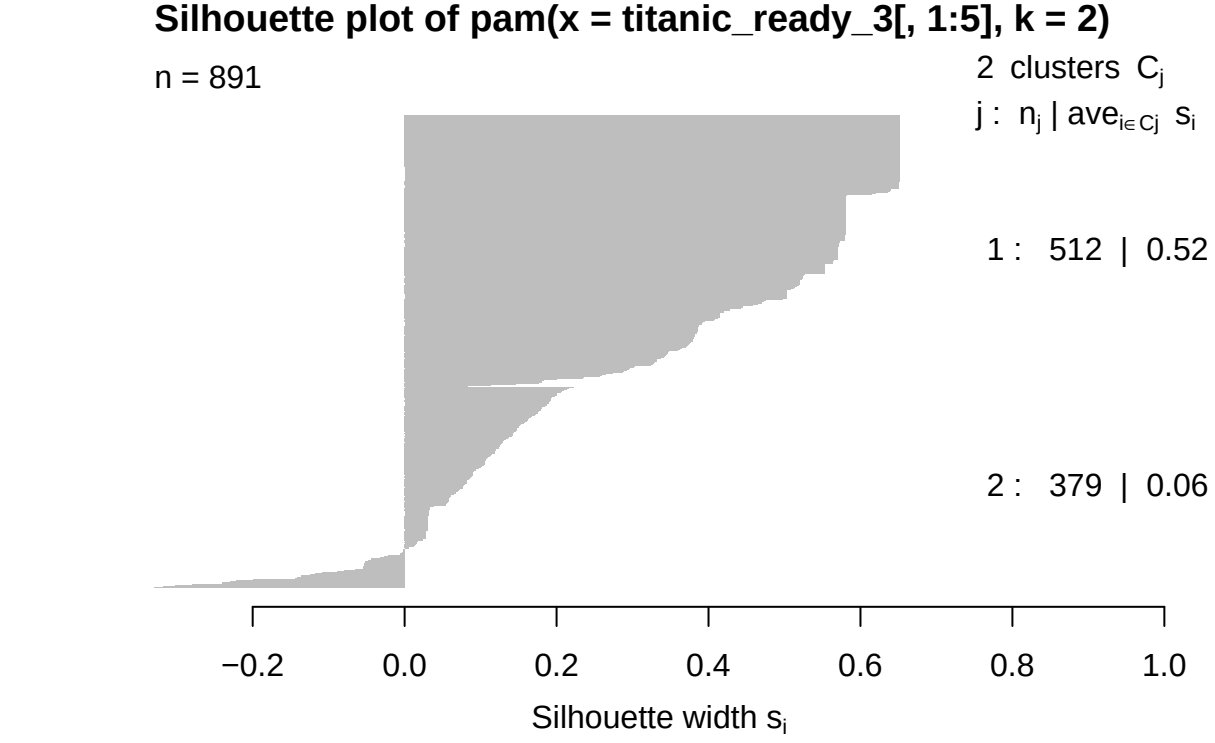
This method is very common. Unlike K-means method use geometrical center of some closely located data points calculated by the model as the center of cluster, K-medoids use certain data points as the center of cluster, which reduce the impact of some outliers on the whole model. This method is clearly more robust than the traditional k-means. After setting the number of cluster as 2, we can see that the 2 cluster obtained from k-medoids are similar to the groups divided by whether the passenger survived. The "accuracy" is about 77%. The visualization of the k-medoids is also presented.

```
library(cluster)
tit_pam<-pam(titanic_ready_3[,1:5],2)
table(tit_pam$clustering,titanic_ready_3$Survived)
```

```
##
##       0   1
##    1 427  85
##    2 122 257
```

```
plot(tit_pam)
```


clusplot(pam(x = titanic_ready_3[, 1:5], k = 2))
Component 2
8
6
4
2
0
−2
−6
−4
−2
0
2
4
Component 1
These two components explain 64.02 % of the point variability.


Silhouette plot of pam(x = titanic_ready_3[, 1:5], k = 2)
n = 891
2 clusters Cj
j : nj | avei∈Cj si
1 : 512 | 0.52
2 : 379 | 0.06
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
Silhouette width si
Average silhouette width : 0.32

Conclusions and Futher Works

In this small project, 4 machine learning model are trained to obtain the prediction of "Survived" of the part of passengers on Titanic according to their personal information. Some pre-processing work has been done including new features extraction, NA filling and the conversion of form suitable for the training of models. The quality of the 4 models are assessed and they are not bad. We also visualize the 4 models to make it more clearly. Some useful conclusions about the relationship of a passenger's information and the possibility of his survival are drawn. Further work includes using some more powerful models and inspecting the data set more carefully to find some more feature engineering method and most importantly, obtain the test set and use test set to assess the performance of models.