Titanic-multi-models

Bei Yu

# prepare data

First, load in train and test data

trainset <- read.csv("/Users/byu/Desktop/Data/titanic-train.csv")  
testset <- read.csv("/Users/byu/Desktop/Data/titanic-test.csv")

Second, convert Survived to nominal variable, Pclass to ordinal

trainset$Survived=factor(trainset$Survived)  
trainset$Pclass=ordered(trainset$Pclass)  
testset$Survived=factor(testset$Survived)  
testset$Pclass=ordered(testset$Pclass)

Third, replace missing value with mean and mode

library(RWeka)  
MS <- make\_Weka\_filter("weka/filters/unsupervised/attribute/ReplaceMissingValues")   
trainset\_nomissing <-MS(data=trainset, na.action = NULL)  
testset\_nomissing <-MS(data=testset, na.action = NULL)

Fourth, use RWeka InfoGain to rank feature relevance to prediction

library("RWeka")  
InfoGainAttributeEval(Survived ~ . , data = trainset\_nomissing)

## Pclass Sex Age SibSp Parch Ticket   
## 0.08383105 0.21766011 0.01670457 0.02650997 0.01538075 0.82593692   
## Fare Cabin Embarked PassengerId   
## 0.09616054 0.23838554 0.02404709 0.00000000

Fifth, select potentially relevant variables for analysis

myVars=c("Pclass", "Sex", "Age", "SibSp", "Fare", "Survived")  
newtrain=trainset\_nomissing[myVars]  
newtest=testset\_nomissing[myVars]  
str(newtrain)

## 'data.frame': 891 obs. of 6 variables:  
## $ Pclass : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...  
## $ Sex : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...  
## $ Age : num 22 38 26 35 35 ...  
## $ SibSp : num 1 1 0 1 0 0 0 3 0 1 ...  
## $ Fare : num 7.25 71.28 7.92 53.1 8.05 ...  
## $ Survived: Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...

str(newtest)

## 'data.frame': 418 obs. of 6 variables:  
## $ Pclass : Factor w/ 3 levels "1","2","3": 3 3 2 3 3 3 3 2 3 3 ...  
## $ Sex : Factor w/ 2 levels "female","male": 2 1 2 2 1 2 1 2 1 2 ...  
## $ Age : num 34.5 47 62 27 22 14 30 26 18 21 ...  
## $ SibSp : num 0 1 0 0 1 0 0 1 0 2 ...  
## $ Fare : num 7.83 7 9.69 8.66 12.29 ...  
## $ Survived: Factor w/ 1 level "?": 1 1 1 1 1 1 1 1 1 1 ...

Sixth, use the "infotheo" package to discretize numeric variable; combine train and test data for unified discretization

# Kaggle returned lower accuracy .727  
#install.packages("infotheo")  
library(infotheo)  
data <- rbind(newtrain, newtest)   
dData <- discretize(data[, 2:4], disc = "equalwidth", nbins=10)  
dData <- lapply(dData, as.factor)  
dData <- cbind(data[, c(1,6)], dData)  
dlabel <- data$Survived  
dData <- cbind(dData, dlabel)  
# separate train (1-891) and test  
train\_index <- 1:891  
train1<- dData[train\_index,]  
test1<- dData[-train\_index,]

# train and test naive Bayes model

library(e1071)  
nb=naiveBayes(Survived~., data = train1, laplace = 1, na.action = na.pass)  
nb

##   
## Naive Bayes Classifier for Discrete Predictors  
##   
## Call:  
## naiveBayes.default(x = X, y = Y, laplace = laplace)  
##   
## A-priori probabilities:  
## Y  
## 0 1 ?   
## 0.6161616 0.3838384 0.0000000   
##   
## Conditional probabilities:  
## Pclass  
## Y 1 2 3  
## 0 0.1467391 0.1775362 0.6757246  
## 1 0.3971014 0.2550725 0.3478261  
## ? 0.3333333 0.3333333 0.3333333  
##   
## Sex  
## Y 1 10  
## 0 0.1488203 0.8511797  
## 1 0.6802326 0.3197674  
## ? 0.5000000 0.5000000  
##   
## Age  
## Y 1 2 3 4 5  
## 0 0.033989267 0.050089445 0.205724508 0.411449016 0.119856887  
## 1 0.105113636 0.056818182 0.181818182 0.335227273 0.150568182  
## ? 0.100000000 0.100000000 0.100000000 0.100000000 0.100000000  
## Age  
## Y 6 7 8 9 10  
## 0 0.084078712 0.044722719 0.028622540 0.017889088 0.003577818  
## 1 0.071022727 0.062500000 0.028409091 0.002840909 0.005681818  
## ? 0.100000000 0.100000000 0.100000000 0.100000000 0.100000000  
##   
## SibSp  
## Y 1 2 3 4 6 7  
## 0 0.71762590 0.17625899 0.02877698 0.02338129 0.02877698 0.01079137  
## 1 0.60458453 0.32378223 0.04011461 0.01432665 0.01146132 0.00286533  
## ? 0.14285714 0.14285714 0.14285714 0.14285714 0.14285714 0.14285714  
## SibSp  
## Y 10  
## 0 0.01438849  
## 1 0.00286533  
## ? 0.14285714  
##   
## dlabel  
## Y 0 1 ?  
## 0 0.996376812 0.001811594 0.001811594  
## 1 0.002898551 0.994202899 0.002898551  
## ? 0.333333333 0.333333333 0.333333333

pred=predict(nb, newdata=test1, type=c("class"))  
myids=c("PassengerId")  
id\_col=testset[myids]  
newpred=cbind(id\_col, pred)  
colnames(newpred)=c("Passengerid", "Survived")  
write.csv(newpred, file="/Users/byu/Desktop/Data/titanic-binned-NB-pred.csv", row.names=FALSE)

# kNN in the "class" package

Now we will use the "class" package to run kNN. No missing values are allowed. No nominal values are allowed. Labels should be separated from train and test data. Kaggle returned accuracy .617

# install.packages("class")  
library(class)  
train\_labels = newtrain$Survived  
sex=as.numeric(newtrain$Sex)  
pclass=as.numeric(newtrain$Pclass)  
dtrain=cbind(sex, newtrain[, c(3,4)] )  
dtrain=cbind(dtrain, pclass)  
  
sex=as.numeric(newtest$Sex)  
pclass=as.numeric(newtest$Pclass)  
dtest=cbind(sex, newtest[, c(3,4)] )  
dtest=cbind(dtest, pclass)  
  
predKNN <- knn(train=dtrain, test=dtest, cl=train\_labels, k=3)  
myids=c("PassengerId")  
id\_col=testset[myids]  
newpred=cbind(id\_col, predKNN)  
colnames(newpred)=c("Passengerid", "Survived")  
write.csv(newpred, file="/Users/byu/Desktop/Data/titanic-kNN-pred.csv", row.names=FALSE)

# SVM

Kaggle returned prediction accuray .77990

library(e1071)  
svm<- svm(Survived~., data = newtrain)  
pred=predict(svm, newdata=newtest, type=c("class"))  
myids=c("PassengerId")  
id\_col=testset[myids]  
newpred=cbind(id\_col, pred)  
colnames(newpred)=c("Passengerid", "Survived")  
write.csv(newpred, file="/Users/byu/Desktop/Data/titanic-SVM-pred.csv", row.names=FALSE)

# random forest on non-discretized data

Kaggle returned accuracy .727

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

## randomForest 4.6-12

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

rfm <- randomForest(Survived~., data=newtrain, ntree=10)  
print(rfm)

##   
## Call:  
## randomForest(formula = Survived ~ ., data = newtrain, ntree = 10)   
## Type of random forest: classification  
## Number of trees: 10  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 18.93%  
## Confusion matrix:  
## 0 1 class.error  
## 0 471 70 0.1293900  
## 1 96 240 0.2857143

predRF <- predict(rfm, newtest, type=c("class"))  
myids=c("PassengerId")  
id\_col=testset[myids]  
newpred=cbind(id\_col, pred)  
colnames(newpred)=c("Passengerid", "Survived")  
write.csv(newpred, file="/Users/byu/Desktop/Data/titanic-RF-pred.csv", row.names=FALSE)