BUAN 4310 Assignment #2

Meng She

10/22/2018

library(class)

# Exploring and preparing the data

# 1. Import and read the csv data file.  
lego <- read.csv("lego\_sets\_1.csv")  
lego<-na.omit(lego)

# 2. Examine the structure of the wbcd data frame.  
str(lego)

## 'data.frame': 10486 obs. of 14 variables:  
## $ ages : Factor w/ 31 levels "10+","12-Aug",..: 4 4 4 7 7 7 7 7 7 7 ...  
## $ list\_price : num 30 20 13 100 80 ...  
## $ num\_reviews : int 2 2 11 23 14 7 37 24 23 11 ...  
## $ piece\_count : int 277 168 74 1032 744 597 598 780 468 444 ...  
## $ play\_star\_rating : num 4 4 4.3 3.6 3.2 3.7 3.7 4.4 3.6 3.6 ...  
## $ prod\_desc : Factor w/ 732 levels "Battle the Jedi with posable Jango Fett!鈩\xa2,75107",..: 121 405 140 250 509 131 130 508 129 128 ...  
## $ prod\_id : Factor w/ 745 levels "Hover into action on Rey's Speeder鈩\xa2! This cool landspeeder has everything Rey needs for scouting the surfa"| \_\_truncated\_\_,..: 672 671 670 152 156 158 150 151 155 154 ...  
## $ prod\_long\_desc : Factor w/ 987 levels "Join Batgirl鈥檚 pursuit of The Joker鈩\xa2 and Harley Quinn鈩\xa2 through the streets of Gotham City in this e"| \_\_truncated\_\_,..: 803 794 785 800 816 848 863 862 869 897 ...  
## $ review\_difficulty: Factor w/ 10 levels "Captain Phasma鈩\xa2,4.7",..: 4 6 6 4 5 4 4 4 4 4 ...  
## $ set\_name : Factor w/ 753 levels "TIE Fighter鈩\xa2,4.7",..: 189 508 507 685 581 565 487 203 407 219 ...  
## $ star\_rating : Factor w/ 57 levels "Star Wars鈩\xa2,3.4",..: 45 50 43 46 46 49 42 47 47 48 ...  
## $ theme\_name : Factor w/ 116 levels "Disney鈩\xa2,5",..: 33 33 32 55 55 55 55 55 55 55 ...  
## $ val\_star\_rating : Factor w/ 34 levels "","1","1.8","2",..: 34 34 34 25 23 26 23 25 23 27 ...  
## $ country : Factor w/ 3 levels "","NON US","US": 1 1 1 3 3 3 3 3 3 3 ...  
## - attr(\*, "na.action")=Class 'omit' Named int [1:1775] 23 33 49 70 85 93 95 109 154 163 ...  
## .. ..- attr(\*, "names")= chr [1:1775] "23" "33" "49" "70" ...

##### 10486 observation and 14 features

##### ages: age group

##### list\_price: price for sell

##### num\_reviews: numbers of reviews for this set

##### piece\_count: how many pieces in this Lego set

##### play\_star\_rating: star rating for this lego set, highest is 5 star

##### prod\_desc: product descriptions

##### prod\_id: product id

##### prod\_long\_desc:product description in details

##### review\_difficulty: difficulty level, 5 level with 'Very easy';'Easy'; 'Average';'Challenging'; 'Very challenging'

##### set\_name: name of this set

##### star\_rating: star rating of this set

##### theme\_name: theme name of this set

##### val\_star\_rating:

##### country: countries that sell this set

#### Each observation is a different lego set and there are features like how many pieces are in the set, how much the set sells for, etc.

#### numerical features:number of pieces,ratings,price

#### categorical: set\_name, ages\_bucket, difficulty

#### charcter :description

# 3. Drop features  
lego$prod\_long\_desc<-NULL  
lego$prod\_desc <- NULL  
lego$prod\_id <- NULL  
lego$set\_name <- NULL  
lego$theme\_name <- NULL  
lego=na.omit(lego)

* These features does not provide useful information, and we will need to exclude them from the model.

# 4. Find the target variable. How many levels this variable have?  
table(lego$country)

##   
## NON US US   
## 3442 6579 465

* 2 levels

# 5. Transform this nominal variable to factor and give its levels a better names. What are their correspondent鈥檚 percentage with 1 decimal place?  
lego$country <- factor(lego$country, levels = c("NON US","US"), labels = c("None USA","USA"))

round(prop.table(table(lego$country)) \* 100, digits = 1)

##   
## None USA USA   
## 93.4 6.6

# Transformation - normalizing numeric data

# 1. Create the normalize() function and check if it work correctly using a vector of numbers before applying this function to the whole data.  
# create normalization function  
normalize <- function(x) {  
return ((x - min(x)) / (max(x) - min(x)));  
}  
# test normalization function - result should be identical  
normalize(c(1, 2, 3, 4, 5))

## [1] 0.00 0.25 0.50 0.75 1.00

normalize(c(10, 20, 30, 40, 50))

## [1] 0.00 0.25 0.50 0.75 1.00

# 2. To apply the normalize() function to the whole data use the lapply() function.   
lego\_n <- as.data.frame(lapply(lego[2],normalize))

summary(lego\_n$list\_price)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00000 0.01725 0.03112 0.06175 0.06985 1.00000

lego=na.omit(lego)

lego$star\_rating=as.numeric(as.character(lego$star\_rating))  
lego$val\_star\_rating=as.numeric(as.character(lego$val\_star\_rating))  
  
print(apply(lego,2,function(x){  
 sum(is.na(x))  
}))

## ages list\_price num\_reviews piece\_count   
## 0 0 0 0   
## play\_star\_rating review\_difficulty star\_rating val\_star\_rating   
## 0 0 0 20   
## country   
## 0

M=mean(lego$val\_star\_rating,na.rm = TRUE)  
index=1:length(lego$val\_star\_rating)  
na.index=index[is.na(lego$val\_star\_rating)]  
lego$val\_star\_rating[na.index]=M  
  
  
lego$list\_price=normalize(lego$list\_price)  
lego$num\_reviews=normalize(lego$num\_reviews)  
lego$piece\_count=normalize(lego$piece\_count)  
lego$play\_star\_rating=normalize(lego$play\_star\_rating)  
  
  
lego$star\_rating=normalize(lego$star\_rating)  
lego$val\_star\_rating=normalize(lego$val\_star\_rating)

# Data preparation - creating training and test datasets

# 1. Create those training and test sets  
set.seed(12345)  
lego\_train\_sample <- order(runif(dim(lego)[1]))  
lego\_train <- lego[lego\_train\_sample[1:6000],-dim(lego)[2]]  
lego\_test <- lego[lego\_train\_sample[6001:length(lego\_train\_sample)],-dim(lego)[2]]

# 2. Can you store its correspondents target variable in 2 separate vectors?  
lego\_train\_lables <- lego[lego\_train\_sample[1:6000],"country"]  
lego\_test\_lables <- lego[lego\_train\_sample[6001:length(lego\_train\_sample)],"country"]

# Training a model on the data

# Install the class package in R  
install.packages("class")

## Warning: package 'class' is in use and will not be installed

library(class)  
# p <- knn(train, test, class, k)

# 2. Find k.  
K=ceiling(sqrt(dim(lego\_train)[1]))  
print(K)

## [1] 78

# k = 99

# 3. Use the knn() function from the class package to classify the test data.  
lego\_test\_pred <- knn(train = lego\_train[,-c(1,6)], test=lego\_test[,-c(1,6)], cl = lego\_train\_lables, k = K)

remove the variabel \*\* ages \*\* and \*\* review\_difficulty \*\* ,the model works.

# Evaluating model performan

# load the "gmodels" library(gmodel)  
library(gmodels)

#Create the cross tabulation   
CrossTable(x = lego\_test\_lables,y=lego\_test\_pred,prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 1044   
##   
##   
## | lego\_test\_pred   
## lego\_test\_lables | None USA | Row Total |   
## -----------------|-----------|-----------|  
## None USA | 967 | 967 |   
## | 0.926 | |   
## -----------------|-----------|-----------|  
## USA | 77 | 77 |   
## | 0.074 | |   
## -----------------|-----------|-----------|  
## Column Total | 1044 | 1044 |   
## -----------------|-----------|-----------|  
##   
##

## 2. Accuracy of the model on test data

confusion.mat=as.matrix(table(lego\_test\_lables,lego\_test\_pred))  
sprintf("the test accuracy is : %f",sum(diag(confusion.mat))/sum(confusion.mat))

## [1] "the test accuracy is : 0.926245"

## 3 Discuss and explain the results of using this model by looking into the false negative and the false positive numbers.

Based on the above table got from \*\* CrossTable \*\* table,the false negatives rate is **0.074**,the false positives rate is \*\* 0.0 **,the true positives rate is**  0.926 **,the true negatives rate is**  0 \*\*.

## Is it possible to improve the performance of this model

prop.table(table(lego\_train\_lables))

## lego\_train\_lables  
## None USA USA   
## 0.93533333 0.06466667

The \*\* prob.table \*\* shows us the label \*\* None USA \*\* ratio of possession is as high as 93.5%,which tells us that this binary classifier is trained on imbalanced dataset.So it is possible to improve the performance of this model

# 5. Improving model performance

## 1.rescaling our numeric features using z-score

lego\_train=lego\_train[,-c(1,6)]  
lego\_test=lego\_test[,-c(1,6)]  
  
lego\_train\_scale=as.data.frame(scale(lego\_train))  
lego\_test\_scale=as.data.frame(scale(lego\_test))  
  
summary(lego\_train\_scale)

## list\_price num\_reviews piece\_count play\_star\_rating   
## Min. :-0.72684 Min. :-0.4083 Min. :-0.64221 Min. :-3.9345   
## 1st Qu.:-0.50579 1st Qu.:-0.3823 1st Qu.:-0.51115 1st Qu.:-0.6276   
## Median :-0.34035 Median :-0.2781 Median :-0.35139 Median : 0.1992   
## Mean : 0.00000 Mean : 0.0000 Mean : 0.00000 Mean : 0.0000   
## 3rd Qu.: 0.07146 3rd Qu.:-0.1219 3rd Qu.: 0.08173 3rd Qu.: 1.0259   
## Max. : 9.64310 Max. : 9.1212 Max. : 6.74948 Max. : 1.0259   
## star\_rating val\_star\_rating   
## Min. :-4.4022 Min. :-4.8116   
## 1st Qu.:-0.4252 1st Qu.:-0.3645   
## Median : 0.3323 Median : 0.2284   
## Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.9004 3rd Qu.: 0.8214   
## Max. : 0.9004 Max. : 1.1179

### re-train model

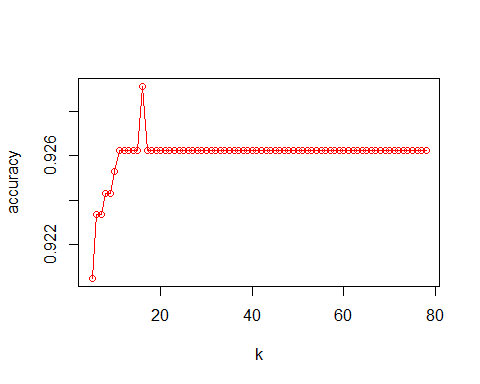
lego\_test\_pred\_new <- knn(train = lego\_train\_scale, test=lego\_test\_scale, cl = lego\_train\_lables, k = K)  
CrossTable(x = lego\_test\_lables,y=lego\_test\_pred\_new,prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 1044   
##   
##   
## | lego\_test\_pred\_new   
## lego\_test\_lables | None USA | Row Total |   
## -----------------|-----------|-----------|  
## None USA | 967 | 967 |   
## | 0.926 | |   
## -----------------|-----------|-----------|  
## USA | 77 | 77 |   
## | 0.074 | |   
## -----------------|-----------|-----------|  
## Column Total | 1044 | 1044 |   
## -----------------|-----------|-----------|  
##   
##

Compare to the previous model,unfortunately,it does not improve performance.So the strategy of rescaling our numeric features using z-score fail.

## 2.Try other k value

k=5:ceiling(sqrt(dim(lego\_train)[1]))  
accuracy=c()  
for(i in k){  
 pred=knn(train = lego\_train, test=lego\_test, cl = lego\_train\_lables, k = i)  
 confusion.m=as.matrix(table(lego\_test\_lables,pred))  
 acc=sum(diag(confusion.m))/sum(confusion.m)  
 accuracy<-c(accuracy,acc)  
}  
  
plot(k,accuracy,type = "o",col="red")

 the different k in model vs accuracy plot tells that ,changing k is not a good stategy to imporve this model

# Try anothet model(额外尝试而已,纯粹尝试，无关)

According to the knowlage,the \*\* random forest model\*\* can handle imbanlance dataset well.So we try random forest model

library(randomForest)

## randomForest 4.6-12

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

rf=randomForest(lego\_train,lego\_train\_lables,importance = TRUE,proximity = TRUE,oob.prox = TRUE)  
rf.pred=predict(rf,lego\_test)

confusion.mat.rf=as.matrix(table(lego\_test\_lables,rf.pred))  
sprintf("the test accuracy is : %f",sum(diag(confusion.mat.rf))/sum(confusion.mat.rf))

## [1] "the test accuracy is : 0.954023"

CrossTable(x = lego\_test\_lables,y=rf.pred,prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 1044   
##   
##   
## | rf.pred   
## lego\_test\_lables | None USA | USA | Row Total |   
## -----------------|-----------|-----------|-----------|  
## None USA | 961 | 6 | 967 |   
## | 0.994 | 0.006 | 0.926 |   
## | 0.958 | 0.146 | |   
## | 0.920 | 0.006 | |   
## -----------------|-----------|-----------|-----------|  
## USA | 42 | 35 | 77 |   
## | 0.545 | 0.455 | 0.074 |   
## | 0.042 | 0.854 | |   
## | 0.040 | 0.034 | |   
## -----------------|-----------|-----------|-----------|  
## Column Total | 1003 | 41 | 1044 |   
## | 0.961 | 0.039 | |   
## -----------------|-----------|-----------|-----------|  
##   
##

Fortunately,the accuracy of randomforest model is \*\* 95.3% \*\*,it increase 3%.So the randomforest model perform better than KNN model