Code with Markdown files compiled

Akhil Gupta, Harshit Gautam, Arpit Kapoor, Shreyansh Singhvi

May 13, 2017

**Regression and Ensemble**

library(dummies)

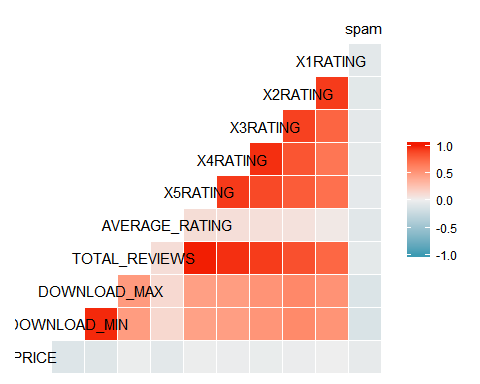
## dummies-1.5.6 provided by Decision Patterns

library(GGally)

## Warning: package 'GGally' was built under R version 3.3.3

DM <- read.csv(file = "app\_metadata\_cleaned\_removed\_min\_downloads\_above\_5m.csv", header = TRUE)  
attach(DM)  
  
set.seed(12345)  
df <- data.frame(CATEGORY,PRICE, CONTENT\_RATING,DOWNLOAD\_MIN,MIN\_REQ\_ANDROID\_FIRST,TOTAL\_REVIEWS,AVERAGE\_RATING) ## Converting to data frame, selected variables based on exploratory analysis.  
  
data <- df  
num.vars <- sapply(data, is.numeric)  
data[num.vars] <- lapply(data[num.vars], scale)  
df <- data   
  
df\_new <- dummy.data.frame(df, names = c("CATEGORY","CONTENT\_RATING","MIN\_REQ\_ANDROID\_FIRST"), sep = ".")  
df\_new <- cbind(df\_new, spam)  
ggcorr(DM)

## Warning in ggcorr(DM): data in column(s) 'APP\_ID', 'APP\_NAME', 'CATEGORY',  
## 'CONTENT\_RATING', 'DOWNLOADS', 'SIZE\_MEGABYTES', 'CURRENT\_VERSION',  
## 'MIN\_REQUIRED\_ANDROID', 'MIN\_REQ\_ANDROID\_FIRST', 'LASTUPDATED',  
## 'DEVELOPER\_SITE', 'DEVELOPER\_CONTACT', 'DEVELOPER\_NAME' are not numeric and  
## were ignored



train\_ind<-sample(nrow(df\_new),0.7\*nrow(df\_new)) ####Sampling Data into training and testing  
train <- df\_new[train\_ind,]  
test <- df\_new[-train\_ind,]  
nrow(train)

## [1] 5348

nrow(test)

## [1] 2292

####### Logistic Regresssion in sampled data #######  
bfit <- glm(as.numeric(spam)~., data = train, family = "binomial")

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(bfit)

##   
## Call:  
## glm(formula = as.numeric(spam) ~ ., family = "binomial", data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.1361 -0.4557 -0.3352 -0.1277 3.6833   
##   
## Coefficients: (2 not defined because of singularities)  
## Estimate Std. Error z value  
## (Intercept) -1.319e+13 1.871e+13 -0.705  
## CATEGORY.ArcadeandAction 1.319e+13 1.871e+13 0.705  
## CATEGORY.BooksandReference 1.319e+13 1.871e+13 0.705  
## CATEGORY.BrainandPuzzle 1.319e+13 1.871e+13 0.705  
## CATEGORY.Business 1.319e+13 1.871e+13 0.705  
## CATEGORY.CardsandCasino 1.319e+13 1.871e+13 0.705  
## CATEGORY.Casual 1.319e+13 1.871e+13 0.705  
## CATEGORY.Comics 1.319e+13 1.871e+13 0.705  
## CATEGORY.Communication 1.319e+13 1.871e+13 0.705  
## CATEGORY.Education 1.319e+13 1.871e+13 0.705  
## CATEGORY.Entertainment 1.319e+13 1.871e+13 0.705  
## CATEGORY.Finance 1.319e+13 1.871e+13 0.705  
## CATEGORY.HealthandFitness 1.319e+13 1.871e+13 0.705  
## CATEGORY.LibrariesandDemo 1.319e+13 1.871e+13 0.705  
## CATEGORY.Lifestyle 1.319e+13 1.871e+13 0.705  
## CATEGORY.MediaandVideo 1.319e+13 1.871e+13 0.705  
## CATEGORY.Medical 1.319e+13 1.871e+13 0.705  
## CATEGORY.MusicandAudio 1.319e+13 1.871e+13 0.705  
## CATEGORY.NewsandMagazines 1.319e+13 1.871e+13 0.705  
## CATEGORY.Personalisation 1.319e+13 1.871e+13 0.705  
## CATEGORY.Photography 1.319e+13 1.871e+13 0.705  
## CATEGORY.Productivity 1.319e+13 1.871e+13 0.705  
## CATEGORY.Racing 1.319e+13 1.871e+13 0.705  
## CATEGORY.Shopping 1.319e+13 1.871e+13 0.705  
## CATEGORY.Social 1.319e+13 1.871e+13 0.705  
## CATEGORY.Sports 1.319e+13 1.871e+13 0.705  
## CATEGORY.SportsGames 1.319e+13 1.871e+13 0.705  
## CATEGORY.Tools 1.319e+13 1.871e+13 0.705  
## CATEGORY.Transportation 1.319e+13 1.871e+13 0.705  
## CATEGORY.TravelandLocal 1.319e+13 1.871e+13 0.705  
## CATEGORY.Weather 1.319e+13 1.871e+13 0.705  
## PRICE -8.069e-01 1.989e-01 -4.057  
## CONTENT\_RATING.Everyone 1.694e+00 1.018e+00 1.663  
## CONTENT\_RATING.HighMaturity 2.087e+00 1.059e+00 1.970  
## CONTENT\_RATING.LowMaturity 1.984e+00 1.019e+00 1.947  
## CONTENT\_RATING.MediumMaturity 1.990e+00 1.036e+00 1.920  
## CONTENT\_RATING.NotRated NA NA NA  
## DOWNLOAD\_MIN -1.062e+00 2.770e-01 -3.833  
## MIN\_REQ\_ANDROID\_FIRST.1 8.589e-01 1.031e+00 0.833  
## MIN\_REQ\_ANDROID\_FIRST.2 1.028e+00 1.023e+00 1.004  
## MIN\_REQ\_ANDROID\_FIRST.3 -6.579e-02 1.197e+00 -0.055  
## MIN\_REQ\_ANDROID\_FIRST.4 -1.256e+00 1.437e+00 -0.874  
## `MIN\_REQ\_ANDROID\_FIRST.Varies with device` NA NA NA  
## TOTAL\_REVIEWS -1.677e+00 1.134e+00 -1.480  
## AVERAGE\_RATING -1.246e-01 4.411e-02 -2.824  
## Pr(>|z|)   
## (Intercept) 0.480870   
## CATEGORY.ArcadeandAction 0.480870   
## CATEGORY.BooksandReference 0.480870   
## CATEGORY.BrainandPuzzle 0.480870   
## CATEGORY.Business 0.480870   
## CATEGORY.CardsandCasino 0.480870   
## CATEGORY.Casual 0.480870   
## CATEGORY.Comics 0.480870   
## CATEGORY.Communication 0.480870   
## CATEGORY.Education 0.480870   
## CATEGORY.Entertainment 0.480870   
## CATEGORY.Finance 0.480870   
## CATEGORY.HealthandFitness 0.480870   
## CATEGORY.LibrariesandDemo 0.480870   
## CATEGORY.Lifestyle 0.480870   
## CATEGORY.MediaandVideo 0.480870   
## CATEGORY.Medical 0.480870   
## CATEGORY.MusicandAudio 0.480870   
## CATEGORY.NewsandMagazines 0.480870   
## CATEGORY.Personalisation 0.480870   
## CATEGORY.Photography 0.480870   
## CATEGORY.Productivity 0.480870   
## CATEGORY.Racing 0.480870   
## CATEGORY.Shopping 0.480870   
## CATEGORY.Social 0.480870   
## CATEGORY.Sports 0.480870   
## CATEGORY.SportsGames 0.480870   
## CATEGORY.Tools 0.480870   
## CATEGORY.Transportation 0.480870   
## CATEGORY.TravelandLocal 0.480870   
## CATEGORY.Weather 0.480870   
## PRICE 4.97e-05 \*\*\*  
## CONTENT\_RATING.Everyone 0.096214 .   
## CONTENT\_RATING.HighMaturity 0.048852 \*   
## CONTENT\_RATING.LowMaturity 0.051519 .   
## CONTENT\_RATING.MediumMaturity 0.054874 .   
## CONTENT\_RATING.NotRated NA   
## DOWNLOAD\_MIN 0.000127 \*\*\*  
## MIN\_REQ\_ANDROID\_FIRST.1 0.404714   
## MIN\_REQ\_ANDROID\_FIRST.2 0.315156   
## MIN\_REQ\_ANDROID\_FIRST.3 0.956178   
## MIN\_REQ\_ANDROID\_FIRST.4 0.382210   
## `MIN\_REQ\_ANDROID\_FIRST.Varies with device` NA   
## TOTAL\_REVIEWS 0.138973   
## AVERAGE\_RATING 0.004744 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2818.5 on 5347 degrees of freedom  
## Residual deviance: 2469.2 on 5305 degrees of freedom  
## AIC: 2555.2  
##   
## Number of Fisher Scoring iterations: 25

predict\_train <- predict(bfit, newdata = train[,1:45])

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

t3 <- ifelse(predict\_train > 0.5,1,0)  
#t3  
Confusion\_train <- table(as.numeric(train$spam),t3)  
Confusion\_train

## t3  
## 0  
## 0 4953  
## 1 395

actual <- as.numeric(train$spam)  
Metrics <- c("AE","RMSE","MAE")  
x1 <- mean(actual-predict\_train)  
x2 <- sqrt(mean((actual-predict\_train)^2))  
x3 <- mean(abs(actual-predict\_train))  
Values <- c(x1,x2,x3)  
X <- data.frame(Metrics,Values)  
X

## Metrics Values  
## 1 AE 3.845684  
## 2 RMSE 5.373691  
## 3 MAE 3.845684

predict\_test <- predict(bfit, newdata = test[,1:45])

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

t4 <- ifelse(predict\_test > 0.5,1,0)  
#t4  
confusion\_test <- table(as.numeric(test$spam),t4)  
confusion\_test

## t4  
## 0  
## 0 2136  
## 1 156

actual\_test <- as.numeric(test$spam)  
Metrics <- c("AE","RMSE","MAE")  
x1\_test <- mean(actual\_test-predict\_test)  
x2\_test <- sqrt(mean((actual\_test-predict\_test)^2))  
x3\_test <- mean(abs(actual\_test-predict\_test))  
Values <- c(x1\_test,x2\_test,x3\_test)  
X\_test <- data.frame(Metrics,Values)  
X\_test

## Metrics Values  
## 1 AE 3.819411  
## 2 RMSE 5.257797  
## 3 MAE 3.819411

#Under Sampling Data  
#Taking all the observations with spam = 1  
train\_under <- train[train$spam==1,]  
  
#Randomly select observations with spam = 0  
zero\_spam <- train[train$spam==0,]  
set.seed(123457)  
rearrangedZero\_spams <- zero\_spam[sample(nrow(zero\_spam), length(train\_under$spam)),]  
  
train\_under <- rbind(train\_under, rearrangedZero\_spams)  
  
############## Logistic regression on undersampled data ########################  
bfit\_under <- glm(as.numeric(spam)~., data = train\_under, family = "binomial")  
summary(bfit\_under)

##   
## Call:  
## glm(formula = as.numeric(spam) ~ ., family = "binomial", data = train\_under)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.36086 -0.98495 0.00017 0.92792 2.85842   
##   
## Coefficients: (3 not defined because of singularities)  
## Estimate Std. Error z value  
## (Intercept) -19.08962 969.09773 -0.020  
## CATEGORY.ArcadeandAction 15.24436 969.09576 0.016  
## CATEGORY.BooksandReference 16.31504 969.09573 0.017  
## CATEGORY.BrainandPuzzle 16.31423 969.09575 0.017  
## CATEGORY.Business 17.90131 969.09607 0.018  
## CATEGORY.CardsandCasino 15.51924 969.09603 0.016  
## CATEGORY.Casual 16.31664 969.09578 0.017  
## CATEGORY.Comics 16.32192 969.09597 0.017  
## CATEGORY.Communication 15.16462 969.09585 0.016  
## CATEGORY.Education 16.55317 969.09577 0.017  
## CATEGORY.Entertainment 16.90001 969.09571 0.017  
## CATEGORY.Finance 15.48241 969.09673 0.016  
## CATEGORY.HealthandFitness 18.32907 969.09604 0.019  
## CATEGORY.LibrariesandDemo -0.32257 1367.30351 0.000  
## CATEGORY.Lifestyle 15.75551 969.09580 0.016  
## CATEGORY.MediaandVideo 17.48990 969.09589 0.018  
## CATEGORY.Medical 16.53404 969.09701 0.017  
## CATEGORY.MusicandAudio 16.51040 969.09576 0.017  
## CATEGORY.NewsandMagazines 16.83934 969.09596 0.017  
## CATEGORY.Personalisation 15.98137 969.09570 0.016  
## CATEGORY.Photography 15.44734 969.09597 0.016  
## CATEGORY.Productivity 15.34324 969.09579 0.016  
## CATEGORY.Racing 15.22505 969.09604 0.016  
## CATEGORY.Shopping 17.25417 969.09626 0.018  
## CATEGORY.Social 14.33835 969.09634 0.015  
## CATEGORY.Sports 17.00033 969.09591 0.018  
## CATEGORY.SportsGames 16.32899 969.09604 0.017  
## CATEGORY.Tools 15.95576 969.09573 0.016  
## CATEGORY.Transportation 31.95433 1690.66314 0.019  
## CATEGORY.TravelandLocal 15.03147 969.09599 0.016  
## CATEGORY.Weather NA NA NA  
## PRICE -0.69999 0.25604 -2.734  
## CONTENT\_RATING.Everyone 2.00891 1.28598 1.562  
## CONTENT\_RATING.HighMaturity 2.60422 1.39742 1.864  
## CONTENT\_RATING.LowMaturity 2.68762 1.28520 2.091  
## CONTENT\_RATING.MediumMaturity 2.59589 1.32869 1.954  
## CONTENT\_RATING.NotRated NA NA NA  
## DOWNLOAD\_MIN -0.97753 0.28223 -3.464  
## MIN\_REQ\_ANDROID\_FIRST.1 -0.18793 1.51871 -0.124  
## MIN\_REQ\_ANDROID\_FIRST.2 -0.04376 1.50268 -0.029  
## MIN\_REQ\_ANDROID\_FIRST.3 -0.83054 1.80969 -0.459  
## MIN\_REQ\_ANDROID\_FIRST.4 -2.06666 1.86735 -1.107  
## `MIN\_REQ\_ANDROID\_FIRST.Varies with device` NA NA NA  
## TOTAL\_REVIEWS -1.87274 1.20848 -1.550  
## AVERAGE\_RATING -0.13544 0.07607 -1.780  
## Pr(>|z|)   
## (Intercept) 0.984284   
## CATEGORY.ArcadeandAction 0.987449   
## CATEGORY.BooksandReference 0.986568   
## CATEGORY.BrainandPuzzle 0.986569   
## CATEGORY.Business 0.985262   
## CATEGORY.CardsandCasino 0.987223   
## CATEGORY.Casual 0.986567   
## CATEGORY.Comics 0.986562   
## CATEGORY.Communication 0.987515   
## CATEGORY.Education 0.986372   
## CATEGORY.Entertainment 0.986086   
## CATEGORY.Finance 0.987253   
## CATEGORY.HealthandFitness 0.984910   
## CATEGORY.LibrariesandDemo 0.999812   
## CATEGORY.Lifestyle 0.987029   
## CATEGORY.MediaandVideo 0.985601   
## CATEGORY.Medical 0.986388   
## CATEGORY.MusicandAudio 0.986407   
## CATEGORY.NewsandMagazines 0.986136   
## CATEGORY.Personalisation 0.986843   
## CATEGORY.Photography 0.987282   
## CATEGORY.Productivity 0.987368   
## CATEGORY.Racing 0.987465   
## CATEGORY.Shopping 0.985795   
## CATEGORY.Social 0.988195   
## CATEGORY.Sports 0.986004   
## CATEGORY.SportsGames 0.986557   
## CATEGORY.Tools 0.986864   
## CATEGORY.Transportation 0.984921   
## CATEGORY.TravelandLocal 0.987625   
## CATEGORY.Weather NA   
## PRICE 0.006259 \*\*   
## CONTENT\_RATING.Everyone 0.118250   
## CONTENT\_RATING.HighMaturity 0.062379 .   
## CONTENT\_RATING.LowMaturity 0.036510 \*   
## CONTENT\_RATING.MediumMaturity 0.050734 .   
## CONTENT\_RATING.NotRated NA   
## DOWNLOAD\_MIN 0.000533 \*\*\*  
## MIN\_REQ\_ANDROID\_FIRST.1 0.901517   
## MIN\_REQ\_ANDROID\_FIRST.2 0.976770   
## MIN\_REQ\_ANDROID\_FIRST.3 0.646277   
## MIN\_REQ\_ANDROID\_FIRST.4 0.268409   
## `MIN\_REQ\_ANDROID\_FIRST.Varies with device` NA   
## TOTAL\_REVIEWS 0.121224   
## AVERAGE\_RATING 0.075007 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1095.2 on 789 degrees of freedom  
## Residual deviance: 847.4 on 748 degrees of freedom  
## AIC: 931.4  
##   
## Number of Fisher Scoring iterations: 15

################## Predicting on the testing data set #########################  
predict\_train\_under <- predict(bfit\_under, newdata = train\_under[,1:44])

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

t3\_under <- ifelse(predict\_train\_under > 0.5,1,0)  
#t3\_under  
Confusion\_train\_under <- table(as.numeric(train\_under$spam),t3\_under)  
Confusion\_train\_under

## t3\_under  
## 0 1  
## 0 323 72  
## 1 163 232

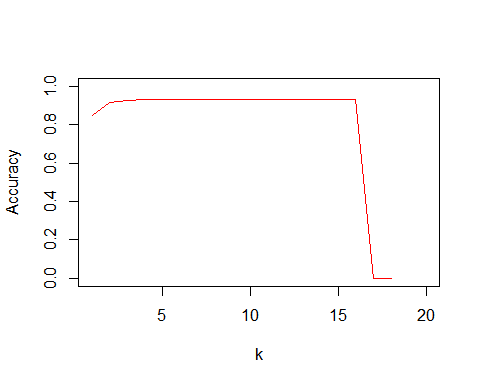
pt = -2  
Accuracy\_test\_under <- rep(0,18)  
predict\_test\_under <- predict(bfit\_under, newdata = test[,1:44])

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

for (i in 1:16){  
 pt = pt +.1  
 print(pt)   
 print(i)  
t4\_under <- ifelse(predict\_test\_under > i,1,0)  
confusion\_test\_under <- table(as.numeric(test$spam),t4\_under)  
confusion\_test\_under  
Accuracy\_test\_under[i] <- (confusion\_test\_under[1,1] + confusion\_test\_under[2,2])/(confusion\_test\_under[1,1] + confusion\_test\_under[1,2]  
 + confusion\_test\_under[2,1] + confusion\_test\_under[2,2])  
Accuracy\_test\_under[i]  
print (Accuracy\_test\_under[i])  
}

## [1] -1.9  
## [1] 1  
## [1] 0.8507853  
## [1] -1.8  
## [1] 2  
## [1] 0.9157941  
## [1] -1.7  
## [1] 3  
## [1] 0.9280105  
## [1] -1.6  
## [1] 4  
## [1] 0.9297557  
## [1] -1.5  
## [1] 5  
## [1] 0.9297557  
## [1] -1.4  
## [1] 6  
## [1] 0.9297557  
## [1] -1.3  
## [1] 7  
## [1] 0.9297557  
## [1] -1.2  
## [1] 8  
## [1] 0.9297557  
## [1] -1.1  
## [1] 9  
## [1] 0.9297557  
## [1] -1  
## [1] 10  
## [1] 0.9297557  
## [1] -0.9  
## [1] 11  
## [1] 0.9297557  
## [1] -0.8  
## [1] 12  
## [1] 0.9297557  
## [1] -0.7  
## [1] 13  
## [1] 0.9297557  
## [1] -0.6  
## [1] 14  
## [1] 0.9297557  
## [1] -0.5  
## [1] 15  
## [1] 0.930192  
## [1] -0.4  
## [1] 16  
## [1] 0.9306283

plot(c(1,20),c(0,1),type="n", xlab="k",ylab="Accuracy")  
lines(Accuracy\_test\_under,col="red")



library(ROCR)

## Loading required package: gplots

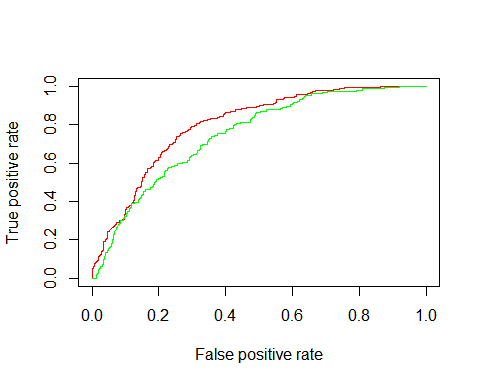
##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

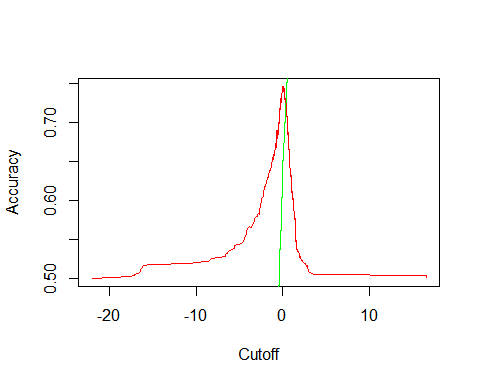
#  
pred <- prediction( predict\_train\_under, train\_under$spam)  
perf <- performance( pred, "tpr", "fpr" )  
  
t4\_under <- ifelse(predict\_test\_under > .2459,1,0)  
confusion\_test\_under <- table(as.numeric(test$spam),t4\_under)  
confusion\_test\_under

## t4\_under  
## 0 1  
## 0 1508 628  
## 1 58 98

#  
pred\_val <- prediction( predict\_test\_under, test$spam)  
perf\_val <- performance( pred\_val, "tpr", "fpr" )  
  
plot( perf , col="red")  
plot (perf\_val, add = TRUE, col="green")



####################### Calculating best cutoff ########################  
perf <- performance( pred, "acc")  
perf\_val <- performance( pred\_val, "acc")  
plot( perf , show.spread.at=seq(0, 1, by=0.1), col="red")  
plot( perf\_val , add= TRUE, show.spread.at=seq(0, 1, by=0.1), col="green")



ind = which.max(slot(perf,"y.values")[[1]])  
acc=slot(perf,"y.values")[[1]][ind]  
cutoff = slot(perf,"x.values")[[1]][ind]  
print(c(accuracy= acc, cutoff= cutoff))

## accuracy cutoff.2459   
## 0.7468354 0.0151191

############oversampling data######################  
train\_over <- train[train$spam==1,]  
train\_1 <- train\_over  
for (i in seq(from=1, to=6, by=1)){  
 train\_over <- rbind(train\_over, train\_1)  
}  
#train\_over  
train\_oversampling <- rbind(train, train\_over)  
  
##########################Logistic regression on oversampled data ###############  
bfit\_over <- glm(as.numeric(spam)~., data = train\_oversampling, family = "binomial")

## Warning: glm.fit: algorithm did not converge  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(bfit\_over)

##   
## Call:  
## glm(formula = as.numeric(spam) ~ ., family = "binomial", data = train\_oversampling)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.0052 -0.9467 -0.3400 1.0015 2.9271   
##   
## Coefficients: (2 not defined because of singularities)  
## Estimate Std. Error z value  
## (Intercept) 1.319e+12 2.015e+12 0.655  
## CATEGORY.ArcadeandAction -1.319e+12 2.015e+12 -0.655  
## CATEGORY.BooksandReference -1.319e+12 2.015e+12 -0.655  
## CATEGORY.BrainandPuzzle -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Business -1.319e+12 2.015e+12 -0.655  
## CATEGORY.CardsandCasino -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Casual -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Comics -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Communication -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Education -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Entertainment -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Finance -1.319e+12 2.015e+12 -0.655  
## CATEGORY.HealthandFitness -1.319e+12 2.015e+12 -0.655  
## CATEGORY.LibrariesandDemo -4.505e+15 2.015e+12 -2235.295  
## CATEGORY.Lifestyle -1.319e+12 2.015e+12 -0.655  
## CATEGORY.MediaandVideo -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Medical -1.319e+12 2.015e+12 -0.655  
## CATEGORY.MusicandAudio -1.319e+12 2.015e+12 -0.655  
## CATEGORY.NewsandMagazines -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Personalisation -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Photography -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Productivity -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Racing -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Shopping -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Social -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Sports -1.319e+12 2.015e+12 -0.655  
## CATEGORY.SportsGames -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Tools -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Transportation -1.319e+12 2.015e+12 -0.655  
## CATEGORY.TravelandLocal -1.319e+12 2.015e+12 -0.655  
## CATEGORY.Weather -1.319e+12 2.015e+12 -0.655  
## PRICE -7.498e-01 7.759e-02 -9.663  
## CONTENT\_RATING.Everyone 1.643e+00 3.871e-01 4.245  
## CONTENT\_RATING.HighMaturity 2.095e+00 4.143e-01 5.057  
## CONTENT\_RATING.LowMaturity 2.029e+00 3.869e-01 5.246  
## CONTENT\_RATING.MediumMaturity 1.973e+00 3.987e-01 4.950  
## CONTENT\_RATING.NotRated NA NA NA  
## DOWNLOAD\_MIN -9.002e-01 9.743e-02 -9.240  
## MIN\_REQ\_ANDROID\_FIRST.1 9.182e-01 4.150e-01 2.213  
## MIN\_REQ\_ANDROID\_FIRST.2 1.142e+00 4.107e-01 2.780  
## MIN\_REQ\_ANDROID\_FIRST.3 -5.347e-02 4.841e-01 -0.110  
## MIN\_REQ\_ANDROID\_FIRST.4 -1.150e+00 5.600e-01 -2.054  
## `MIN\_REQ\_ANDROID\_FIRST.Varies with device` NA NA NA  
## TOTAL\_REVIEWS -1.688e+00 4.335e-01 -3.893  
## AVERAGE\_RATING -1.576e-01 2.244e-02 -7.020  
## Pr(>|z|)   
## (Intercept) 0.51268   
## CATEGORY.ArcadeandAction 0.51268   
## CATEGORY.BooksandReference 0.51268   
## CATEGORY.BrainandPuzzle 0.51268   
## CATEGORY.Business 0.51268   
## CATEGORY.CardsandCasino 0.51268   
## CATEGORY.Casual 0.51268   
## CATEGORY.Comics 0.51268   
## CATEGORY.Communication 0.51268   
## CATEGORY.Education 0.51268   
## CATEGORY.Entertainment 0.51268   
## CATEGORY.Finance 0.51268   
## CATEGORY.HealthandFitness 0.51268   
## CATEGORY.LibrariesandDemo < 2e-16 \*\*\*  
## CATEGORY.Lifestyle 0.51268   
## CATEGORY.MediaandVideo 0.51268   
## CATEGORY.Medical 0.51268   
## CATEGORY.MusicandAudio 0.51268   
## CATEGORY.NewsandMagazines 0.51268   
## CATEGORY.Personalisation 0.51268   
## CATEGORY.Photography 0.51268   
## CATEGORY.Productivity 0.51268   
## CATEGORY.Racing 0.51268   
## CATEGORY.Shopping 0.51268   
## CATEGORY.Social 0.51268   
## CATEGORY.Sports 0.51268   
## CATEGORY.SportsGames 0.51268   
## CATEGORY.Tools 0.51268   
## CATEGORY.Transportation 0.51268   
## CATEGORY.TravelandLocal 0.51268   
## CATEGORY.Weather 0.51268   
## PRICE < 2e-16 \*\*\*  
## CONTENT\_RATING.Everyone 2.19e-05 \*\*\*  
## CONTENT\_RATING.HighMaturity 4.27e-07 \*\*\*  
## CONTENT\_RATING.LowMaturity 1.56e-07 \*\*\*  
## CONTENT\_RATING.MediumMaturity 7.42e-07 \*\*\*  
## CONTENT\_RATING.NotRated NA   
## DOWNLOAD\_MIN < 2e-16 \*\*\*  
## MIN\_REQ\_ANDROID\_FIRST.1 0.02692 \*   
## MIN\_REQ\_ANDROID\_FIRST.2 0.00544 \*\*   
## MIN\_REQ\_ANDROID\_FIRST.3 0.91205   
## MIN\_REQ\_ANDROID\_FIRST.4 0.03993 \*   
## `MIN\_REQ\_ANDROID\_FIRST.Varies with device` NA   
## TOTAL\_REVIEWS 9.89e-05 \*\*\*  
## AVERAGE\_RATING 2.21e-12 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 10847.5 on 8112 degrees of freedom  
## Residual deviance: 8908.4 on 8070 degrees of freedom  
## AIC: 8994.4  
##   
## Number of Fisher Scoring iterations: 25

#################### Predicting on testing data set#####################  
predict\_train\_over <- predict(bfit\_over, newdata = train\_oversampling[,1:44])

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

t3\_over <- ifelse(predict\_train\_over > 0.5,1,0)  
#t3\_over  
Confusion\_train\_over <- table(as.numeric(train\_oversampling$spam),t3\_over)  
Confusion\_train\_over

## t3\_over  
## 0 1  
## 0 4396 557  
## 1 2168 992

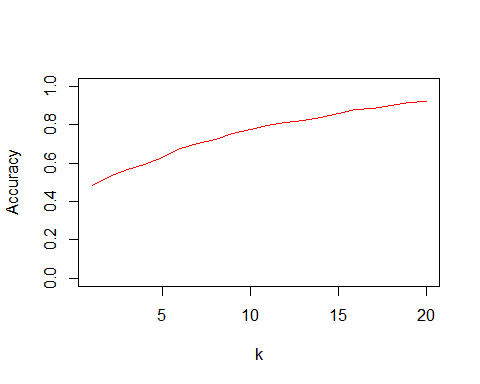
pt = -1  
Accuracy\_test\_over <- rep(0,20)  
predict\_test\_over <- predict(bfit\_over, newdata = test[,1:44])

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =  
## ifelse(type == : prediction from a rank-deficient fit may be misleading

for (i in 1:20){  
pt = pt +.1  
print(pt)   
print(i)  
t4\_over <- ifelse(predict\_test\_over > pt,1,0)  
confusion\_test\_over <- table(as.numeric(test$spam),t4\_over)  
confusion\_test\_over  
Accuracy\_test\_over[i] <- (confusion\_test\_over[1,1] + confusion\_test\_over[2,2])/(confusion\_test\_over[1,1] + confusion\_test\_over[1,2]  
 + confusion\_test\_over[2,1] + confusion\_test\_over[2,2])  
Accuracy\_test\_over[i]  
print (Accuracy\_test\_over[i])  
}

## [1] -0.9  
## [1] 1  
## [1] 0.4851658  
## [1] -0.8  
## [1] 2  
## [1] 0.5309773  
## [1] -0.7  
## [1] 3  
## [1] 0.5671902  
## [1] -0.6  
## [1] 4  
## [1] 0.5942408  
## [1] -0.5  
## [1] 5  
## [1] 0.632199  
## [1] -0.4  
## [1] 6  
## [1] 0.6771379  
## [1] -0.3  
## [1] 7  
## [1] 0.7033159  
## [1] -0.2  
## [1] 8  
## [1] 0.7225131  
## [1] -0.1  
## [1] 9  
## [1] 0.7565445  
## [1] -1.387779e-16  
## [1] 10  
## [1] 0.7779232  
## [1] 0.1  
## [1] 11  
## [1] 0.7953752  
## [1] 0.2  
## [1] 12  
## [1] 0.8115183  
## [1] 0.3  
## [1] 13  
## [1] 0.824171  
## [1] 0.4  
## [1] 14  
## [1] 0.8394415  
## [1] 0.5  
## [1] 15  
## [1] 0.8582024  
## [1] 0.6  
## [1] 16  
## [1] 0.8773997  
## [1] 0.7  
## [1] 17  
## [1] 0.8869983  
## [1] 0.8  
## [1] 18  
## [1] 0.9013962  
## [1] 0.9  
## [1] 19  
## [1] 0.9149215  
## [1] 1  
## [1] 20  
## [1] 0.921466

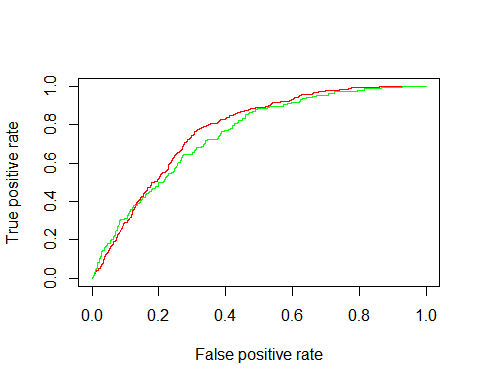
plot(c(1,20),c(0,1),type="n", xlab="k",ylab="Accuracy")  
lines(Accuracy\_test\_over,col="red")



library(ROCR)  
#  
pred\_over <- prediction( predict\_train\_over, train\_oversampling$spam)  
perf\_over <- performance( pred\_over, "tpr", "fpr" )  
  
t4\_over <- ifelse(predict\_test\_over > .39896,1,0)  
confusion\_test\_over <- table(as.numeric(test$spam),t4\_over)  
confusion\_test\_over

## t4\_over  
## 0 1  
## 0 1864 272  
## 1 97 59

################### Calculating the best cutoff ############################  
pred\_val\_over <- prediction( predict\_test\_over, test$spam)  
perf\_val\_over <- performance( pred\_val\_over, "tpr", "fpr" )  
  
plot( perf\_over , col="red")  
plot (perf\_val\_over, add = TRUE, col="green")



ind\_over = which.max(slot(perf\_over,"y.values")[[1]])  
acc\_over = slot(perf\_over,"y.values")[[1]][ind\_over]  
cutoff\_over = slot(perf\_over,"x.values")[[1]][ind\_over]  
print(c(accuracy= acc\_over, cutoff= cutoff\_over))

## accuracy cutoff   
## 1.000000 0.899253

###################Bagging + Random Forest + Boosting#####################  
  
Data <- dummy.data.frame(df, sep = ".", names = c("CATEGORY","CONTENT\_RATING","MIN\_REQ\_ANDROID\_FIRST"))  
  
colnames(Data)[which(names(Data) == "MIN\_REQ\_ANDROID\_FIRST.Varies with device")] <- "MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice"  
colnames(train\_under)[which(names(train\_under) == "MIN\_REQ\_ANDROID\_FIRST.Varies with device")] <- "MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice"  
colnames(train\_oversampling)[which(names(train\_oversampling) == "MIN\_REQ\_ANDROID\_FIRST.Varies with device")] <- "MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice"  
colnames(test)[which(names(test) == "MIN\_REQ\_ANDROID\_FIRST.Varies with device")] <- "MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice"  
  
  
Data <- cbind(Data, spam)  
  
################# Sampling of Data ###############  
train\_rf\_ind <- sample(nrow(Data), .7 \* nrow(Data))  
train\_rf <- Data[train\_rf\_ind,]  
validation\_rf <- Data[-train\_rf\_ind, ]  
  
library(MASS)  
library(tree)

## Warning: package 'tree' was built under R version 3.3.3

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.3.3

## randomForest 4.6-12

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

set.seed(1)  
bag.boston=randomForest(factor(spam)~.,data = train\_rf[,-1],mtry=44,importance=TRUE)

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within  
## valid range

bag.boston

##   
## Call:  
## randomForest(formula = factor(spam) ~ ., data = train\_rf[, -1], mtry = 44, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 43  
##   
## OOB estimate of error rate: 8.1%  
## Confusion matrix:  
## 0 1 class.error  
## 0 4904 67 0.01347817  
## 1 366 11 0.97082228

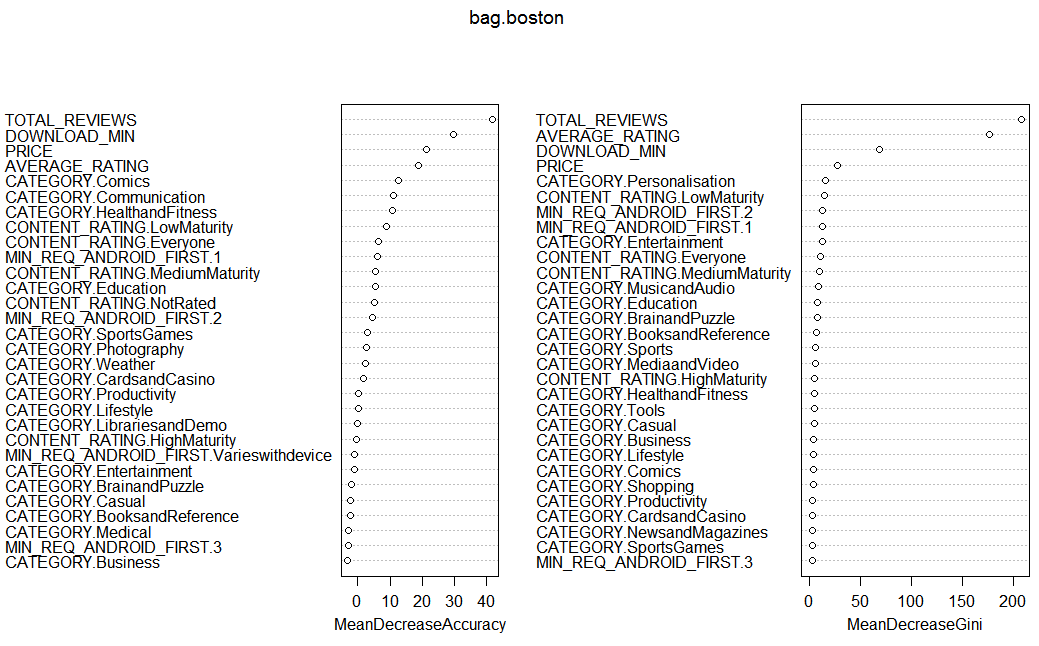
yhat.bag = predict(bag.boston,newdata=validation\_rf)  
boston.test=validation\_rf$spam  
t<- table(boston.test,yhat.bag)  
#t  
Accuracy\_test\_bag <- (t[1,1] + t[2,2])/(t[1,1] + t[1,2] + t[2,1] + t[2,2])  
Accuracy\_test\_bag

## [1] 0.9144852

importance(bag.boston)

## 0 1  
## CATEGORY.BooksandReference -3.6065235 4.0861743  
## CATEGORY.BrainandPuzzle -3.6179309 3.7777916  
## CATEGORY.Business -4.8298430 5.2313298  
## CATEGORY.CardsandCasino 1.7128991 -0.2195931  
## CATEGORY.Casual -2.4659701 0.7863017  
## CATEGORY.Comics 11.6899519 5.9375986  
## CATEGORY.Communication 11.5187930 0.2603683  
## CATEGORY.Education 3.5326130 9.3427664  
## CATEGORY.Entertainment -8.0504640 16.2275763  
## CATEGORY.Finance -5.0615385 -2.1937084  
## CATEGORY.HealthandFitness 8.9356224 10.2842340  
## CATEGORY.LibrariesandDemo 0.0000000 0.0000000  
## CATEGORY.Lifestyle 0.5509264 -1.9668422  
## CATEGORY.MediaandVideo -10.2817407 4.6886259  
## CATEGORY.Medical -1.7379611 -3.8834897  
## CATEGORY.MusicandAudio -11.9168167 12.3814529  
## CATEGORY.NewsandMagazines -10.2213510 6.9008021  
## CATEGORY.Personalisation -8.1421822 2.9064906  
## CATEGORY.Photography 4.0551663 -6.2381947  
## CATEGORY.Productivity 1.7362814 -4.6651745  
## CATEGORY.Racing -5.9295885 -0.7494649  
## CATEGORY.Shopping -7.8634949 6.3730205  
## CATEGORY.Social -7.5400164 2.3522099  
## CATEGORY.Sports -8.4156911 -1.9108026  
## CATEGORY.SportsGames 5.8683271 -8.4032850  
## CATEGORY.Tools -5.4245988 -8.6786501  
## CATEGORY.Transportation -8.5402353 5.9912437  
## CATEGORY.TravelandLocal -2.7162137 -3.8828650  
## CATEGORY.Weather 2.4641165 0.0000000  
## PRICE 19.1729958 8.2488447  
## CONTENT\_RATING.Everyone 6.6437223 -0.7318299  
## CONTENT\_RATING.HighMaturity -1.8821679 3.7471486  
## CONTENT\_RATING.LowMaturity 8.3903611 1.0898944  
## CONTENT\_RATING.MediumMaturity 2.6243488 8.5434071  
## CONTENT\_RATING.NotRated 5.4251728 -1.3009418  
## DOWNLOAD\_MIN 29.7268964 -7.2480294  
## MIN\_REQ\_ANDROID\_FIRST.1 8.2080383 -6.5871304  
## MIN\_REQ\_ANDROID\_FIRST.2 4.8782819 -0.2700717  
## MIN\_REQ\_ANDROID\_FIRST.3 -2.2420847 -4.4398291  
## MIN\_REQ\_ANDROID\_FIRST.4 -6.5879799 -3.7866010  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice -1.0010015 0.0000000  
## TOTAL\_REVIEWS 41.7088323 -24.1136627  
## AVERAGE\_RATING 20.6527280 -10.0068757  
## MeanDecreaseAccuracy  
## CATEGORY.BooksandReference -2.40372032  
## CATEGORY.BrainandPuzzle -2.04455560  
## CATEGORY.Business -3.15454073  
## CATEGORY.CardsandCasino 1.65721187  
## CATEGORY.Casual -2.25606952  
## CATEGORY.Comics 12.60043640  
## CATEGORY.Communication 11.03255622  
## CATEGORY.Education 5.52912484  
## CATEGORY.Entertainment -1.07884262  
## CATEGORY.Finance -5.34512635  
## CATEGORY.HealthandFitness 10.66807229  
## CATEGORY.LibrariesandDemo 0.00000000  
## CATEGORY.Lifestyle 0.07815747  
## CATEGORY.MediaandVideo -8.62583711  
## CATEGORY.Medical -2.72972888  
## CATEGORY.MusicandAudio -7.54668835  
## CATEGORY.NewsandMagazines -8.12187101  
## CATEGORY.Personalisation -7.20867476  
## CATEGORY.Photography 2.63254431  
## CATEGORY.Productivity 0.34733176  
## CATEGORY.Racing -6.07401515  
## CATEGORY.Shopping -5.07665162  
## CATEGORY.Social -6.39909606  
## CATEGORY.Sports -8.48954200  
## CATEGORY.SportsGames 3.15525501  
## CATEGORY.Tools -7.26110952  
## CATEGORY.Transportation -5.60708981  
## CATEGORY.TravelandLocal -3.40788551  
## CATEGORY.Weather 2.46418516  
## PRICE 21.35816991  
## CONTENT\_RATING.Everyone 6.41047774  
## CONTENT\_RATING.HighMaturity -0.46948624  
## CONTENT\_RATING.LowMaturity 8.91726876  
## CONTENT\_RATING.MediumMaturity 5.56534444  
## CONTENT\_RATING.NotRated 5.30475177  
## DOWNLOAD\_MIN 29.57599611  
## MIN\_REQ\_ANDROID\_FIRST.1 5.99008965  
## MIN\_REQ\_ANDROID\_FIRST.2 4.67943786  
## MIN\_REQ\_ANDROID\_FIRST.3 -2.95740754  
## MIN\_REQ\_ANDROID\_FIRST.4 -7.32262133  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice -1.00100150  
## TOTAL\_REVIEWS 41.75531128  
## AVERAGE\_RATING 18.96645062  
## MeanDecreaseGini  
## CATEGORY.BooksandReference 7.07279345  
## CATEGORY.BrainandPuzzle 7.12386922  
## CATEGORY.Business 4.14955490  
## CATEGORY.CardsandCasino 2.72524160  
## CATEGORY.Casual 4.69948856  
## CATEGORY.Comics 3.58552907  
## CATEGORY.Communication 2.20928102  
## CATEGORY.Education 7.76756559  
## CATEGORY.Entertainment 12.02021995  
## CATEGORY.Finance 2.04665685  
## CATEGORY.HealthandFitness 4.92024115  
## CATEGORY.LibrariesandDemo 0.04911905  
## CATEGORY.Lifestyle 3.77468273  
## CATEGORY.MediaandVideo 5.13994293  
## CATEGORY.Medical 1.15450705  
## CATEGORY.MusicandAudio 8.79369612  
## CATEGORY.NewsandMagazines 2.69789582  
## CATEGORY.Personalisation 14.94624270  
## CATEGORY.Photography 2.09212063  
## CATEGORY.Productivity 3.16223634  
## CATEGORY.Racing 0.47593317  
## CATEGORY.Shopping 3.45316380  
## CATEGORY.Social 1.40320803  
## CATEGORY.Sports 5.19067496  
## CATEGORY.SportsGames 2.58133519  
## CATEGORY.Tools 4.87188074  
## CATEGORY.Transportation 1.52447088  
## CATEGORY.TravelandLocal 1.51621054  
## CATEGORY.Weather 0.08691905  
## PRICE 27.21427795  
## CONTENT\_RATING.Everyone 10.20741878  
## CONTENT\_RATING.HighMaturity 4.94083546  
## CONTENT\_RATING.LowMaturity 13.94565154  
## CONTENT\_RATING.MediumMaturity 9.18224847  
## CONTENT\_RATING.NotRated 0.78799065  
## DOWNLOAD\_MIN 67.88360823  
## MIN\_REQ\_ANDROID\_FIRST.1 12.36237190  
## MIN\_REQ\_ANDROID\_FIRST.2 12.60664491  
## MIN\_REQ\_ANDROID\_FIRST.3 2.32779928  
## MIN\_REQ\_ANDROID\_FIRST.4 2.12854489  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 0.08146190  
## TOTAL\_REVIEWS 207.21879467  
## AVERAGE\_RATING 176.32178239

varImpPlot(bag.boston)



##################### Bagging Undersampled data##################  
  
set.seed(1)  
  
################ Training the model #########################  
bag.boston=randomForest(factor(spam)~.,data = train\_under[,-1],mtry=44,importance=TRUE)

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within  
## valid range

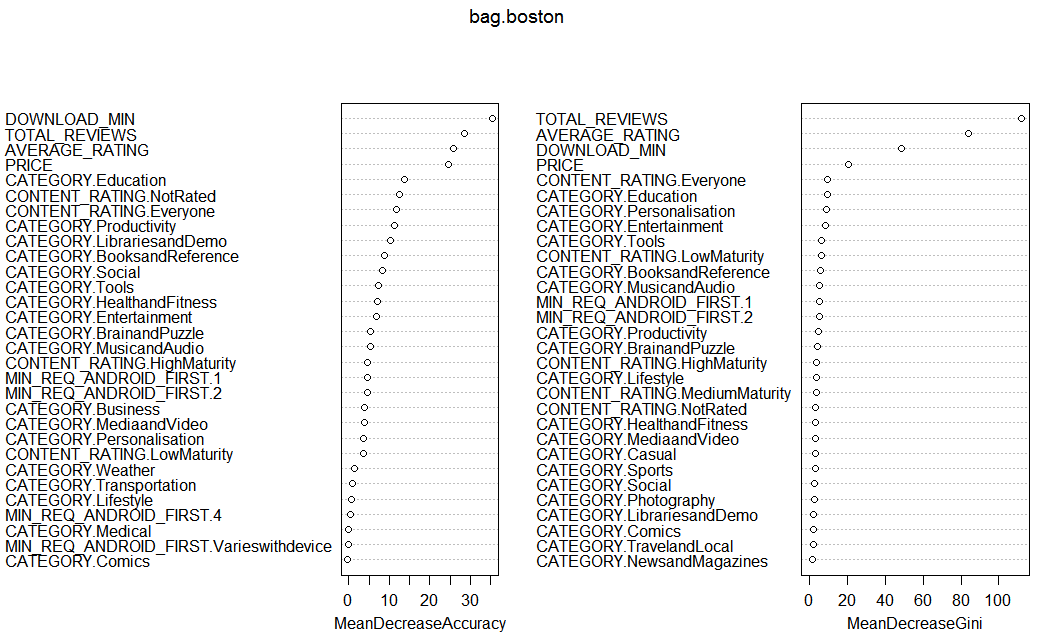
#bag.boston  
  
################# Prediction on the test data ###############  
yhat.bag = predict(bag.boston,newdata=test)  
boston.test=test$spam  
t<- table(boston.test,yhat.bag)  
#t  
Accuracy\_test\_bag <- (t[1,1] + t[2,2])/(t[1,1] + t[1,2] + t[2,1] + t[2,2])  
Accuracy\_test\_bag

## [1] 0.6400524

importance(bag.boston)

## 0 1  
## CATEGORY.BooksandReference 12.7895076 -0.66411391  
## CATEGORY.BrainandPuzzle 0.1970308 6.76662096  
## CATEGORY.Business 0.7445052 4.16780786  
## CATEGORY.CardsandCasino -4.2795583 -1.32554592  
## CATEGORY.Casual -8.9958435 0.08819921  
## CATEGORY.Comics -5.1885957 5.19217253  
## CATEGORY.Communication -2.0859196 1.83031163  
## CATEGORY.Education 15.3591118 3.42242199  
## CATEGORY.Entertainment 8.4419606 1.65688042  
## CATEGORY.Finance -0.6072235 -1.00100150  
## CATEGORY.HealthandFitness 8.1762951 2.70848713  
## CATEGORY.LibrariesandDemo 9.0535902 7.10893351  
## CATEGORY.Lifestyle 3.4822142 -1.91316870  
## CATEGORY.MediaandVideo 1.7105163 3.64959528  
## CATEGORY.Medical 0.0000000 0.00000000  
## CATEGORY.MusicandAudio 10.9511733 -2.53499909  
## CATEGORY.NewsandMagazines -0.5866449 -4.97305487  
## CATEGORY.Personalisation 6.6197506 -1.94349132  
## CATEGORY.Photography 0.9467275 -4.08905327  
## CATEGORY.Productivity 9.3045095 6.56918934  
## CATEGORY.Racing -1.6833498 -0.44369816  
## CATEGORY.Shopping -0.9366010 -5.93567122  
## CATEGORY.Social 7.9361820 4.34598691  
## CATEGORY.Sports -2.9854075 -3.28306463  
## CATEGORY.SportsGames -2.2116256 -1.55675158  
## CATEGORY.Tools 0.2961696 9.59471083  
## CATEGORY.Transportation 0.0000000 1.00100150  
## CATEGORY.TravelandLocal -2.0923228 -4.65165053  
## CATEGORY.Weather 0.6940629 1.73006971  
## PRICE 17.4108382 17.69007707  
## CONTENT\_RATING.Everyone 12.5827427 4.79970303  
## CONTENT\_RATING.HighMaturity 10.7763163 -3.63785684  
## CONTENT\_RATING.LowMaturity 5.4707386 -0.11622419  
## CONTENT\_RATING.MediumMaturity -4.1706645 3.89472310  
## CONTENT\_RATING.NotRated 13.0609306 4.07726666  
## DOWNLOAD\_MIN 18.9956561 22.17187398  
## MIN\_REQ\_ANDROID\_FIRST.1 -2.7719378 6.97998708  
## MIN\_REQ\_ANDROID\_FIRST.2 -1.9173008 6.72653271  
## MIN\_REQ\_ANDROID\_FIRST.3 -1.4160386 -2.92481065  
## MIN\_REQ\_ANDROID\_FIRST.4 -0.5880173 0.97710624  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 0.0000000 0.00000000  
## TOTAL\_REVIEWS 12.6416908 19.70514717  
## AVERAGE\_RATING 12.7454549 22.07946169  
## MeanDecreaseAccuracy  
## CATEGORY.BooksandReference 8.7068050  
## CATEGORY.BrainandPuzzle 5.4161146  
## CATEGORY.Business 3.7858829  
## CATEGORY.CardsandCasino -4.4869153  
## CATEGORY.Casual -6.6225131  
## CATEGORY.Comics -0.2965363  
## CATEGORY.Communication -0.7581797  
## CATEGORY.Education 13.8403158  
## CATEGORY.Entertainment 6.7452729  
## CATEGORY.Finance -1.0370782  
## CATEGORY.HealthandFitness 7.0893851  
## CATEGORY.LibrariesandDemo 10.2334224  
## CATEGORY.Lifestyle 0.7586982  
## CATEGORY.MediaandVideo 3.7814626  
## CATEGORY.Medical 0.0000000  
## CATEGORY.MusicandAudio 5.3368933  
## CATEGORY.NewsandMagazines -4.8257424  
## CATEGORY.Personalisation 3.6534288  
## CATEGORY.Photography -2.4139010  
## CATEGORY.Productivity 11.3157385  
## CATEGORY.Racing -1.4900161  
## CATEGORY.Shopping -5.7561952  
## CATEGORY.Social 8.2243733  
## CATEGORY.Sports -4.2751594  
## CATEGORY.SportsGames -2.7687324  
## CATEGORY.Tools 7.3930956  
## CATEGORY.Transportation 1.0010015  
## CATEGORY.TravelandLocal -5.0006755  
## CATEGORY.Weather 1.3922423  
## PRICE 24.4849576  
## CONTENT\_RATING.Everyone 11.6607865  
## CONTENT\_RATING.HighMaturity 4.6921671  
## CONTENT\_RATING.LowMaturity 3.5819781  
## CONTENT\_RATING.MediumMaturity -0.6040042  
## CONTENT\_RATING.NotRated 12.5875975  
## DOWNLOAD\_MIN 35.4537386  
## MIN\_REQ\_ANDROID\_FIRST.1 4.5570566  
## MIN\_REQ\_ANDROID\_FIRST.2 4.4987316  
## MIN\_REQ\_ANDROID\_FIRST.3 -3.1168030  
## MIN\_REQ\_ANDROID\_FIRST.4 0.3906260  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 0.0000000  
## TOTAL\_REVIEWS 28.5225777  
## AVERAGE\_RATING 25.9387113  
## MeanDecreaseGini  
## CATEGORY.BooksandReference 5.88959188  
## CATEGORY.BrainandPuzzle 3.92232387  
## CATEGORY.Business 1.11270925  
## CATEGORY.CardsandCasino 0.99975357  
## CATEGORY.Casual 3.05119150  
## CATEGORY.Comics 1.94926087  
## CATEGORY.Communication 1.32144086  
## CATEGORY.Education 9.20761878  
## CATEGORY.Entertainment 8.56334001  
## CATEGORY.Finance 0.34567370  
## CATEGORY.HealthandFitness 3.09564829  
## CATEGORY.LibrariesandDemo 2.04310952  
## CATEGORY.Lifestyle 3.44446293  
## CATEGORY.MediaandVideo 3.08890912  
## CATEGORY.Medical 0.32145596  
## CATEGORY.MusicandAudio 5.33180077  
## CATEGORY.NewsandMagazines 1.33867978  
## CATEGORY.Personalisation 8.95506888  
## CATEGORY.Photography 2.25398826  
## CATEGORY.Productivity 4.58641604  
## CATEGORY.Racing 0.95805196  
## CATEGORY.Shopping 1.19432056  
## CATEGORY.Social 2.41565992  
## CATEGORY.Sports 3.01902952  
## CATEGORY.SportsGames 0.76660683  
## CATEGORY.Tools 6.34932729  
## CATEGORY.Transportation 0.06057367  
## CATEGORY.TravelandLocal 1.80480921  
## CATEGORY.Weather 0.19247456  
## PRICE 20.24822596  
## CONTENT\_RATING.Everyone 9.30617900  
## CONTENT\_RATING.HighMaturity 3.80386727  
## CONTENT\_RATING.LowMaturity 6.22164167  
## CONTENT\_RATING.MediumMaturity 3.36846990  
## CONTENT\_RATING.NotRated 3.15468262  
## DOWNLOAD\_MIN 48.54253161  
## MIN\_REQ\_ANDROID\_FIRST.1 5.27818274  
## MIN\_REQ\_ANDROID\_FIRST.2 5.06805252  
## MIN\_REQ\_ANDROID\_FIRST.3 1.09875997  
## MIN\_REQ\_ANDROID\_FIRST.4 1.05129672  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 0.05693099  
## TOTAL\_REVIEWS 111.89812342  
## AVERAGE\_RATING 84.05531533

varImpPlot(bag.boston)



############################### Bagging Oversampled Data########################  
  
set.seed(1)  
bag.boston=randomForest(factor(spam)~.,data = train\_oversampling[,-1],mtry=44,importance=TRUE)

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within  
## valid range

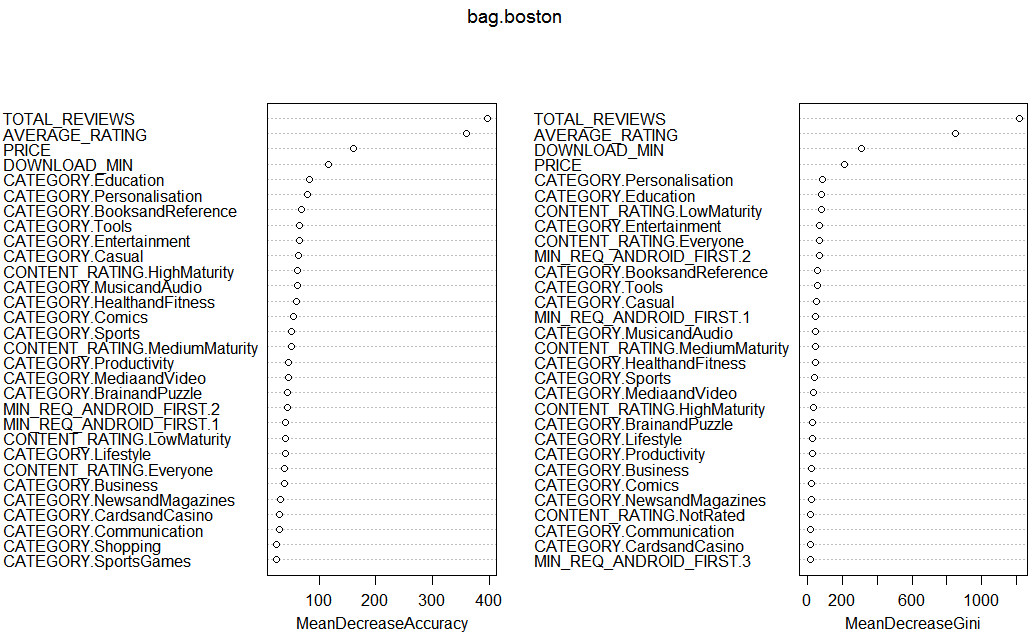
#bag.boston  
yhat.bag = predict(bag.boston,newdata=test)  
boston.test=test$spam  
t<- table(boston.test,yhat.bag)  
#t  
Accuracy\_test\_bag <- (t[1,1] + t[2,2])/(t[1,1] + t[1,2] + t[2,1] + t[2,2])  
Accuracy\_test\_bag

## [1] 0.8926702

importance(bag.boston)

## 0 1  
## CATEGORY.BooksandReference 14.1584068 68.413892  
## CATEGORY.BrainandPuzzle 8.5458615 45.351558  
## CATEGORY.Business 6.3376040 38.775455  
## CATEGORY.CardsandCasino 0.3628054 32.021285  
## CATEGORY.Casual 6.0466570 63.006483  
## CATEGORY.Comics 1.0028666 55.574706  
## CATEGORY.Communication -1.9468016 30.331602  
## CATEGORY.Education 27.3555047 88.754259  
## CATEGORY.Entertainment 23.2326634 64.460572  
## CATEGORY.Finance 6.7766532 14.008968  
## CATEGORY.HealthandFitness 26.9630299 54.776255  
## CATEGORY.LibrariesandDemo 8.2390597 10.206844  
## CATEGORY.Lifestyle 5.6616426 40.212520  
## CATEGORY.MediaandVideo -0.2780605 47.887868  
## CATEGORY.Medical 5.3059892 17.447136  
## CATEGORY.MusicandAudio 13.9517309 57.545019  
## CATEGORY.NewsandMagazines 16.8356105 28.325182  
## CATEGORY.Personalisation 10.4362064 80.012145  
## CATEGORY.Photography 4.6957093 16.600090  
## CATEGORY.Productivity 7.1780440 46.628293  
## CATEGORY.Racing 4.9468396 16.887221  
## CATEGORY.Shopping 1.0477862 25.549077  
## CATEGORY.Social 2.6788298 11.732220  
## CATEGORY.Sports 1.4992354 54.879782  
## CATEGORY.SportsGames -5.4903401 26.252851  
## CATEGORY.Tools 1.1064560 66.832424  
## CATEGORY.Transportation -5.0298728 13.772490  
## CATEGORY.TravelandLocal 6.3130525 19.688153  
## CATEGORY.Weather 0.1760396 8.024482  
## PRICE 35.8954204 181.288602  
## CONTENT\_RATING.Everyone 1.5069357 39.956433  
## CONTENT\_RATING.HighMaturity 9.6050119 61.489096  
## CONTENT\_RATING.LowMaturity -2.7759734 40.057679  
## CONTENT\_RATING.MediumMaturity -0.9471562 54.100874  
## CONTENT\_RATING.NotRated 13.2307717 21.690125  
## DOWNLOAD\_MIN 13.2231485 119.223004  
## MIN\_REQ\_ANDROID\_FIRST.1 -3.2592173 42.504413  
## MIN\_REQ\_ANDROID\_FIRST.2 2.1854255 44.623737  
## MIN\_REQ\_ANDROID\_FIRST.3 12.7340711 22.037518  
## MIN\_REQ\_ANDROID\_FIRST.4 3.3624221 22.793268  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 0.1628676 9.709960  
## TOTAL\_REVIEWS 7.0897416 371.598210  
## AVERAGE\_RATING 10.6725790 406.889763  
## MeanDecreaseAccuracy  
## CATEGORY.BooksandReference 68.444942  
## CATEGORY.BrainandPuzzle 44.805280  
## CATEGORY.Business 39.844149  
## CATEGORY.CardsandCasino 31.268768  
## CATEGORY.Casual 63.227758  
## CATEGORY.Comics 54.893336  
## CATEGORY.Communication 31.088262  
## CATEGORY.Education 83.287019  
## CATEGORY.Entertainment 65.224690  
## CATEGORY.Finance 14.956387  
## CATEGORY.HealthandFitness 60.972583  
## CATEGORY.LibrariesandDemo 12.049341  
## CATEGORY.Lifestyle 40.743423  
## CATEGORY.MediaandVideo 46.729864  
## CATEGORY.Medical 17.991867  
## CATEGORY.MusicandAudio 61.331483  
## CATEGORY.NewsandMagazines 32.501321  
## CATEGORY.Personalisation 80.084222  
## CATEGORY.Photography 17.252962  
## CATEGORY.Productivity 47.120430  
## CATEGORY.Racing 17.402788  
## CATEGORY.Shopping 26.060081  
## CATEGORY.Social 12.008953  
## CATEGORY.Sports 52.131549  
## CATEGORY.SportsGames 24.888097  
## CATEGORY.Tools 66.581016  
## CATEGORY.Transportation 12.919898  
## CATEGORY.TravelandLocal 19.768496  
## CATEGORY.Weather 7.450666  
## PRICE 160.947664  
## CONTENT\_RATING.Everyone 40.093372  
## CONTENT\_RATING.HighMaturity 62.520724  
## CONTENT\_RATING.LowMaturity 40.907657  
## CONTENT\_RATING.MediumMaturity 52.010975  
## CONTENT\_RATING.NotRated 22.347640  
## DOWNLOAD\_MIN 117.047085  
## MIN\_REQ\_ANDROID\_FIRST.1 41.621367  
## MIN\_REQ\_ANDROID\_FIRST.2 44.680128  
## MIN\_REQ\_ANDROID\_FIRST.3 22.661260  
## MIN\_REQ\_ANDROID\_FIRST.4 22.337107  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 9.356375  
## TOTAL\_REVIEWS 396.621219  
## AVERAGE\_RATING 359.767666  
## MeanDecreaseGini  
## CATEGORY.BooksandReference 54.392929  
## CATEGORY.BrainandPuzzle 29.101171  
## CATEGORY.Business 22.636495  
## CATEGORY.CardsandCasino 13.244593  
## CATEGORY.Casual 48.305488  
## CATEGORY.Comics 20.917829  
## CATEGORY.Communication 14.247447  
## CATEGORY.Education 80.432799  
## CATEGORY.Entertainment 68.453380  
## CATEGORY.Finance 6.349513  
## CATEGORY.HealthandFitness 42.071882  
## CATEGORY.LibrariesandDemo 5.567469  
## CATEGORY.Lifestyle 28.450905  
## CATEGORY.MediaandVideo 33.041665  
## CATEGORY.Medical 5.488907  
## CATEGORY.MusicandAudio 45.764098  
## CATEGORY.NewsandMagazines 19.459619  
## CATEGORY.Personalisation 82.163298  
## CATEGORY.Photography 7.869028  
## CATEGORY.Productivity 27.832199  
## CATEGORY.Racing 6.853693  
## CATEGORY.Shopping 9.699088  
## CATEGORY.Social 3.843504  
## CATEGORY.Sports 36.324637  
## CATEGORY.SportsGames 10.354145  
## CATEGORY.Tools 53.615055  
## CATEGORY.Transportation 3.483589  
## CATEGORY.TravelandLocal 9.057308  
## CATEGORY.Weather 2.225260  
## PRICE 210.274458  
## CONTENT\_RATING.Everyone 67.076970  
## CONTENT\_RATING.HighMaturity 32.641940  
## CONTENT\_RATING.LowMaturity 80.104808  
## CONTENT\_RATING.MediumMaturity 44.403613  
## CONTENT\_RATING.NotRated 15.683551  
## DOWNLOAD\_MIN 308.591499  
## MIN\_REQ\_ANDROID\_FIRST.1 47.379146  
## MIN\_REQ\_ANDROID\_FIRST.2 66.074454  
## MIN\_REQ\_ANDROID\_FIRST.3 13.235930  
## MIN\_REQ\_ANDROID\_FIRST.4 11.861882  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 2.381775  
## TOTAL\_REVIEWS 1216.962596  
## AVERAGE\_RATING 851.991094

varImpPlot(bag.boston)



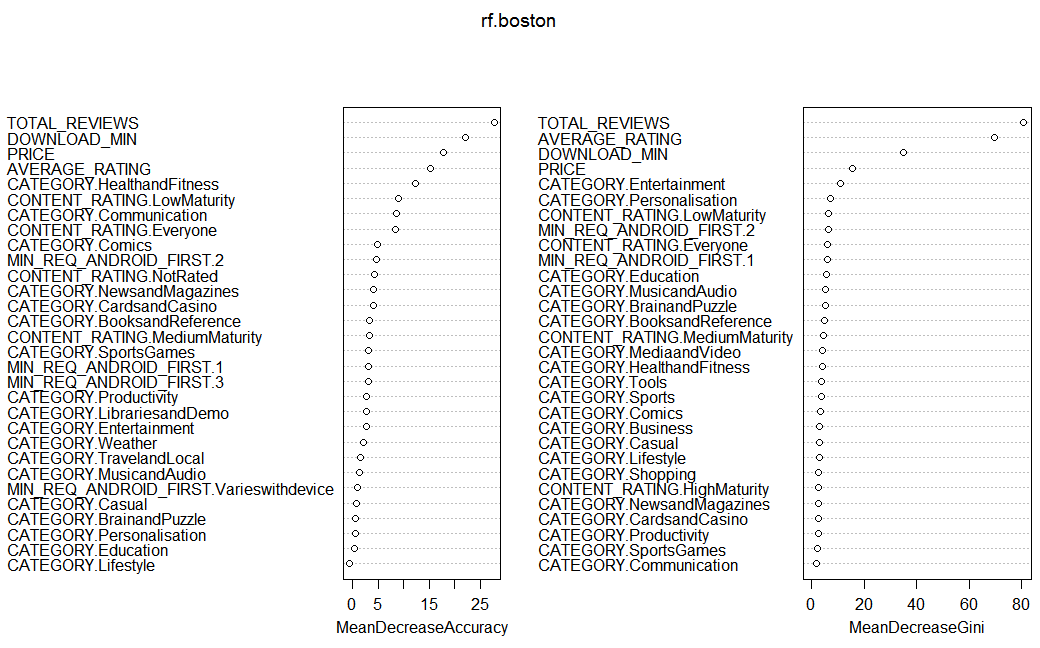
################### Random Forest #########################  
train\_rf$spam <- as.factor(train\_rf$spam)  
set.seed(1)  
  
#################### Training the model ###################  
rf.boston=randomForest(spam~.,data = train\_rf[,-1],mtry=7,importance=TRUE)  
#rf.boston  
  
################# Predicting the test data #######################  
yhat.rf = predict(rf.boston,newdata=validation\_rf)  
boston.test=Data[-train\_rf\_ind,"spam"]  
  
t<- table(boston.test,yhat.rf)  
#t  
Accuracy\_test\_bag <- (t[1,1] + t[2,2])/(t[1,1] + t[1,2] + t[2,1] + t[2,2])  
Accuracy\_test\_bag

## [1] 0.9240838

importance(rf.boston)

## 0 1  
## CATEGORY.BooksandReference 3.87496079 -1.80580583  
## CATEGORY.BrainandPuzzle -0.09556595 1.95827900  
## CATEGORY.Business -5.25867274 3.74828011  
## CATEGORY.CardsandCasino 4.87784070 -2.96741999  
## CATEGORY.Casual 1.15016519 -1.35180540  
## CATEGORY.Comics 5.87049978 -2.19091205  
## CATEGORY.Communication 8.50086954 0.62270230  
## CATEGORY.Education -0.24972331 2.58753226  
## CATEGORY.Entertainment 2.90408136 -0.76283411  
## CATEGORY.Finance -2.96820814 -2.54064357  
## CATEGORY.HealthandFitness 12.17373690 1.01390323  
## CATEGORY.LibrariesandDemo 2.68706708 1.00100150  
## CATEGORY.Lifestyle 0.15728034 -2.62565978  
## CATEGORY.MediaandVideo -9.70613983 3.24373952  
## CATEGORY.Medical -1.52947165 -2.08632540  
## CATEGORY.MusicandAudio 1.18611758 0.72916527  
## CATEGORY.NewsandMagazines 4.02034120 0.02334103  
## CATEGORY.Personalisation 2.19004362 -4.90881821  
## CATEGORY.Photography -0.90845230 -4.07996204  
## CATEGORY.Productivity 3.32836394 -2.07165558  
## CATEGORY.Racing -0.54430182 -1.41516968  
## CATEGORY.Shopping -6.83797702 4.54253512  
## CATEGORY.Social -3.55105302 1.04781757  
## CATEGORY.Sports -5.38976251 -2.75969365  
## CATEGORY.SportsGames 4.49325183 -4.45013353  
## CATEGORY.Tools -1.69276140 -2.39492079  
## CATEGORY.Transportation -7.23917436 4.97019359  
## CATEGORY.TravelandLocal 1.91536837 -1.08273186  
## CATEGORY.Weather 1.83184359 1.41402759  
## PRICE 17.33762698 2.67983681  
## CONTENT\_RATING.Everyone 8.54218015 -0.96989445  
## CONTENT\_RATING.HighMaturity -2.13007709 0.65115225  
## CONTENT\_RATING.LowMaturity 9.76737154 -7.06237376  
## CONTENT\_RATING.MediumMaturity 1.14065674 5.91976250  
## CONTENT\_RATING.NotRated 4.24838748 1.59944870  
## DOWNLOAD\_MIN 22.05137912 -6.66057284  
## MIN\_REQ\_ANDROID\_FIRST.1 3.97559575 -3.39339186  
## MIN\_REQ\_ANDROID\_FIRST.2 5.27378456 -2.81208535  
## MIN\_REQ\_ANDROID\_FIRST.3 3.65064515 -3.27772670  
## MIN\_REQ\_ANDROID\_FIRST.4 -2.29416339 -1.57204252  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 0.68078156 1.41705050  
## TOTAL\_REVIEWS 28.11061495 -19.54999993  
## AVERAGE\_RATING 16.23496053 -6.19787935  
## MeanDecreaseAccuracy  
## CATEGORY.BooksandReference 3.3577629  
## CATEGORY.BrainandPuzzle 0.6637765  
## CATEGORY.Business -3.8435406  
## CATEGORY.CardsandCasino 4.0778099  
## CATEGORY.Casual 0.8211650  
## CATEGORY.Comics 4.9634728  
## CATEGORY.Communication 8.6791790  
## CATEGORY.Education 0.4736649  
## CATEGORY.Entertainment 2.6942022  
## CATEGORY.Finance -3.5061218  
## CATEGORY.HealthandFitness 12.2538915  
## CATEGORY.LibrariesandDemo 2.7504037  
## CATEGORY.Lifestyle -0.5717544  
## CATEGORY.MediaandVideo -8.5760556  
## CATEGORY.Medical -1.9876336  
## CATEGORY.MusicandAudio 1.3998666  
## CATEGORY.NewsandMagazines 4.1130067  
## CATEGORY.Personalisation 0.5183088  
## CATEGORY.Photography -1.9390657  
## CATEGORY.Productivity 2.7561909  
## CATEGORY.Racing -0.6794521  
## CATEGORY.Shopping -4.7661098  
## CATEGORY.Social -3.2128631  
## CATEGORY.Sports -6.0585764  
## CATEGORY.SportsGames 3.2067368  
## CATEGORY.Tools -2.2837028  
## CATEGORY.Transportation -5.0572563  
## CATEGORY.TravelandLocal 1.5782253  
## CATEGORY.Weather 2.1515961  
## PRICE 17.6709482  
## CONTENT\_RATING.Everyone 8.4352270  
## CONTENT\_RATING.HighMaturity -1.8034437  
## CONTENT\_RATING.LowMaturity 8.9634405  
## CONTENT\_RATING.MediumMaturity 3.3566875  
## CONTENT\_RATING.NotRated 4.3510194  
## DOWNLOAD\_MIN 22.0941872  
## MIN\_REQ\_ANDROID\_FIRST.1 3.1683484  
## MIN\_REQ\_ANDROID\_FIRST.2 4.7222743  
## MIN\_REQ\_ANDROID\_FIRST.3 3.1527317  
## MIN\_REQ\_ANDROID\_FIRST.4 -2.5904073  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 0.9853496  
## TOTAL\_REVIEWS 27.7239319  
## AVERAGE\_RATING 15.2661350  
## MeanDecreaseGini  
## CATEGORY.BooksandReference 4.88743532  
## CATEGORY.BrainandPuzzle 5.06294492  
## CATEGORY.Business 3.03021620  
## CATEGORY.CardsandCasino 2.41692682  
## CATEGORY.Casual 3.00209193  
## CATEGORY.Comics 3.26869080  
## CATEGORY.Communication 1.95041837  
## CATEGORY.Education 5.64179293  
## CATEGORY.Entertainment 10.91818090  
## CATEGORY.Finance 1.53644746  
## CATEGORY.HealthandFitness 3.93516984  
## CATEGORY.LibrariesandDemo 0.18051699  
## CATEGORY.Lifestyle 2.76096874  
## CATEGORY.MediaandVideo 4.27673163  
## CATEGORY.Medical 0.97483401  
## CATEGORY.MusicandAudio 5.37668488  
## CATEGORY.NewsandMagazines 2.49039208  
## CATEGORY.Personalisation 7.05526715  
## CATEGORY.Photography 1.74193691  
## CATEGORY.Productivity 2.38756491  
## CATEGORY.Racing 0.58465469  
## CATEGORY.Shopping 2.74552055  
## CATEGORY.Social 1.19629589  
## CATEGORY.Sports 3.68226441  
## CATEGORY.SportsGames 2.30805846  
## CATEGORY.Tools 3.74047304  
## CATEGORY.Transportation 1.23523795  
## CATEGORY.TravelandLocal 1.51790539  
## CATEGORY.Weather 0.07790948  
## PRICE 15.67499159  
## CONTENT\_RATING.Everyone 5.92863475  
## CONTENT\_RATING.HighMaturity 2.70185590  
## CONTENT\_RATING.LowMaturity 6.39568186  
## CONTENT\_RATING.MediumMaturity 4.51753802  
## CONTENT\_RATING.NotRated 0.82614721  
## DOWNLOAD\_MIN 34.79425633  
## MIN\_REQ\_ANDROID\_FIRST.1 5.89674220  
## MIN\_REQ\_ANDROID\_FIRST.2 6.37814400  
## MIN\_REQ\_ANDROID\_FIRST.3 1.46881210  
## MIN\_REQ\_ANDROID\_FIRST.4 1.25213869  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 0.17296816  
## TOTAL\_REVIEWS 80.55104244  
## AVERAGE\_RATING 69.48256686

varImpPlot(rf.boston)



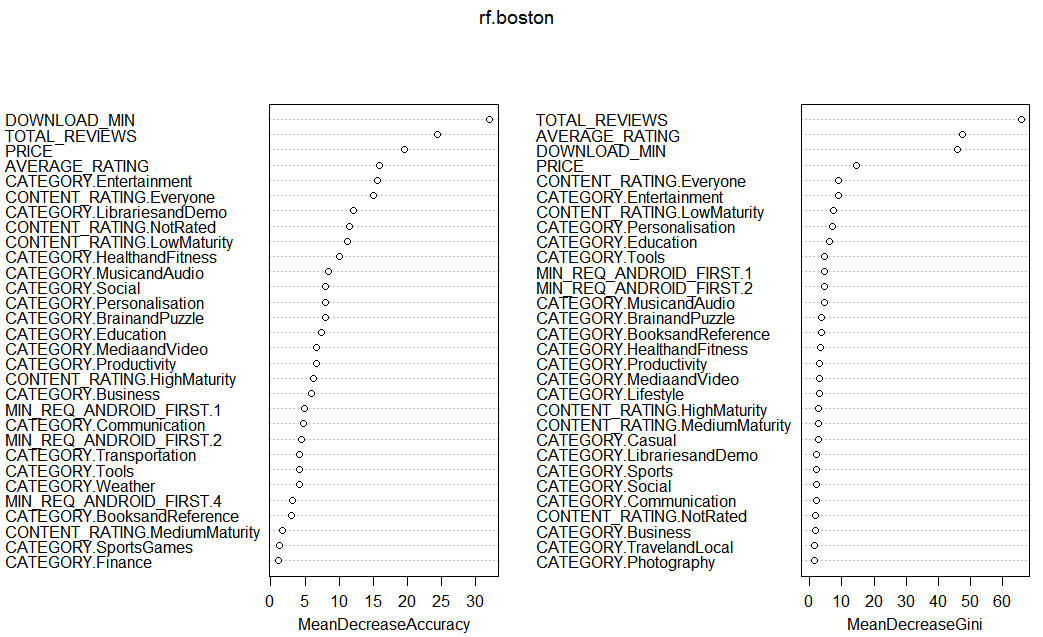
################### Random Forest for Undersampled dataset #########################  
train\_under$spam <- as.factor(train\_under$spam)  
set.seed(1)  
  
###################### Training the model ####################  
rf.boston=randomForest(spam~.,data = train\_under[,-1],mtry=7,importance=TRUE)  
#rf.boston  
  
##################### Predicting on the test data ################  
yhat.rf = predict(rf.boston,newdata=test)  
boston.test= test$spam  
  
t<- table(boston.test,yhat.rf)  
#t  
Accuracy\_test\_bag <- (t[1,1] + t[2,2])/(t[1,1] + t[1,2] + t[2,1] + t[2,2])  
Accuracy\_test\_bag

## [1] 0.675829

importance(rf.boston)

## 0 1  
## CATEGORY.BooksandReference 4.81809978 -0.69552933  
## CATEGORY.BrainandPuzzle 1.25425158 9.59626481  
## CATEGORY.Business 3.52253440 5.43575818  
## CATEGORY.CardsandCasino -4.11475900 -2.35499403  
## CATEGORY.Casual -5.06332784 -1.87787314  
## CATEGORY.Comics -3.72239731 -0.52718677  
## CATEGORY.Communication 2.49851719 4.48045814  
## CATEGORY.Education 7.95643784 2.48308716  
## CATEGORY.Entertainment 14.41347585 8.45808408  
## CATEGORY.Finance -1.41672858 2.24565102  
## CATEGORY.HealthandFitness 11.35704074 4.29166311  
## CATEGORY.LibrariesandDemo 10.49979470 7.86318516  
## CATEGORY.Lifestyle 2.14208926 -2.91112785  
## CATEGORY.MediaandVideo 5.45201677 4.61069913  
## CATEGORY.Medical -1.73367661 0.09342022  
## CATEGORY.MusicandAudio 11.74891309 -0.20123436  
## CATEGORY.NewsandMagazines -0.53181859 -0.21207421  
## CATEGORY.Personalisation 4.88996495 5.84568227  
## CATEGORY.Photography -2.80422909 -3.89286295  
## CATEGORY.Productivity 8.55728764 -0.31476470  
## CATEGORY.Racing -4.26381612 2.48912394  
## CATEGORY.Shopping -0.95346334 -5.59787706  
## CATEGORY.Social 8.59588441 3.28877955  
## CATEGORY.Sports -4.84550813 -2.00014512  
## CATEGORY.SportsGames -0.06213816 1.97111666  
## CATEGORY.Tools -0.64052955 6.72100779  
## CATEGORY.Transportation 1.73727046 3.87892006  
## CATEGORY.TravelandLocal 3.37208538 -2.81944427  
## CATEGORY.Weather 2.58605441 3.96969650  
## PRICE 14.21702090 14.06660131  
## CONTENT\_RATING.Everyone 12.95247482 8.17559003  
## CONTENT\_RATING.HighMaturity 10.80485890 -2.00305780  
## CONTENT\_RATING.LowMaturity 7.56917512 7.23691366  
## CONTENT\_RATING.MediumMaturity -0.83690224 2.85694077  
## CONTENT\_RATING.NotRated 12.30713483 2.48526728  
## DOWNLOAD\_MIN 16.31301415 23.19569109  
## MIN\_REQ\_ANDROID\_FIRST.1 -1.98576460 7.08569954  
## MIN\_REQ\_ANDROID\_FIRST.2 -2.96678962 7.87407040  
## MIN\_REQ\_ANDROID\_FIRST.3 -2.62466521 -2.07505439  
## MIN\_REQ\_ANDROID\_FIRST.4 1.86989694 2.87573346  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice -0.73527641 -1.04451675  
## TOTAL\_REVIEWS 11.34016091 19.00173836  
## AVERAGE\_RATING 11.44161515 10.16142508  
## MeanDecreaseAccuracy  
## CATEGORY.BooksandReference 3.08417568  
## CATEGORY.BrainandPuzzle 7.93150402  
## CATEGORY.Business 5.97398984  
## CATEGORY.CardsandCasino -4.85460031  
## CATEGORY.Casual -4.97178317  
## CATEGORY.Comics -3.04072928  
## CATEGORY.Communication 4.74692832  
## CATEGORY.Education 7.35182397  
## CATEGORY.Entertainment 15.56183649  
## CATEGORY.Finance 1.07571585  
## CATEGORY.HealthandFitness 10.07056407  
## CATEGORY.LibrariesandDemo 12.16730479  
## CATEGORY.Lifestyle -0.59021643  
## CATEGORY.MediaandVideo 6.68588077  
## CATEGORY.Medical -1.32393896  
## CATEGORY.MusicandAudio 8.50361245  
## CATEGORY.NewsandMagazines -0.47530616  
## CATEGORY.Personalisation 8.02921724  
## CATEGORY.Photography -4.78331308  
## CATEGORY.Productivity 6.63780497  
## CATEGORY.Racing -1.56753295  
## CATEGORY.Shopping -4.83061474  
## CATEGORY.Social 8.06764008  
## CATEGORY.Sports -4.75375811  
## CATEGORY.SportsGames 1.23521515  
## CATEGORY.Tools 4.22524804  
## CATEGORY.Transportation 4.24091103  
## CATEGORY.TravelandLocal 0.02372071  
## CATEGORY.Weather 4.16867580  
## PRICE 19.63364123  
## CONTENT\_RATING.Everyone 15.05526509  
## CONTENT\_RATING.HighMaturity 6.26358262  
## CONTENT\_RATING.LowMaturity 11.21295204  
## CONTENT\_RATING.MediumMaturity 1.72831915  
## CONTENT\_RATING.NotRated 11.44878188  
## DOWNLOAD\_MIN 32.08614220  
## MIN\_REQ\_ANDROID\_FIRST.1 4.94760456  
## MIN\_REQ\_ANDROID\_FIRST.2 4.51723552  
## MIN\_REQ\_ANDROID\_FIRST.3 -2.96910567  
## MIN\_REQ\_ANDROID\_FIRST.4 3.14331037  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice -1.08570392  
## TOTAL\_REVIEWS 24.45498032  
## AVERAGE\_RATING 15.85190139  
## MeanDecreaseGini  
## CATEGORY.BooksandReference 3.5858669  
## CATEGORY.BrainandPuzzle 3.6123316  
## CATEGORY.Business 1.6333359  
## CATEGORY.CardsandCasino 0.9960334  
## CATEGORY.Casual 2.5853333  
## CATEGORY.Comics 1.2557455  
## CATEGORY.Communication 2.0120701  
## CATEGORY.Education 5.9957408  
## CATEGORY.Entertainment 8.9627097  
## CATEGORY.Finance 0.5607926  
## CATEGORY.HealthandFitness 3.3863030  
## CATEGORY.LibrariesandDemo 2.2498816  
## CATEGORY.Lifestyle 2.9404299  
## CATEGORY.MediaandVideo 2.9629609  
## CATEGORY.Medical 0.3331034  
## CATEGORY.MusicandAudio 4.5162279  
## CATEGORY.NewsandMagazines 1.1796945  
## CATEGORY.Personalisation 7.1566661  
## CATEGORY.Photography 1.5636942  
## CATEGORY.Productivity 3.0687414  
## CATEGORY.Racing 1.3854916  
## CATEGORY.Shopping 1.0512406  
## CATEGORY.Social 2.0408595  
## CATEGORY.Sports 2.1039718  
## CATEGORY.SportsGames 1.2271295  
## CATEGORY.Tools 4.6620198  
## CATEGORY.Transportation 0.2725000  
## CATEGORY.TravelandLocal 1.5875448  
## CATEGORY.Weather 0.4601845  
## PRICE 14.5425605  
## CONTENT\_RATING.Everyone 9.0461249  
## CONTENT\_RATING.HighMaturity 2.7555444  
## CONTENT\_RATING.LowMaturity 7.2407017  
## CONTENT\_RATING.MediumMaturity 2.6505882  
## CONTENT\_RATING.NotRated 1.9133021  
## DOWNLOAD\_MIN 45.8732161  
## MIN\_REQ\_ANDROID\_FIRST.1 4.5598750  
## MIN\_REQ\_ANDROID\_FIRST.2 4.5434404  
## MIN\_REQ\_ANDROID\_FIRST.3 0.8796514  
## MIN\_REQ\_ANDROID\_FIRST.4 1.2831792  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 0.1710545  
## TOTAL\_REVIEWS 65.8708305  
## AVERAGE\_RATING 47.5315333

varImpPlot(rf.boston)



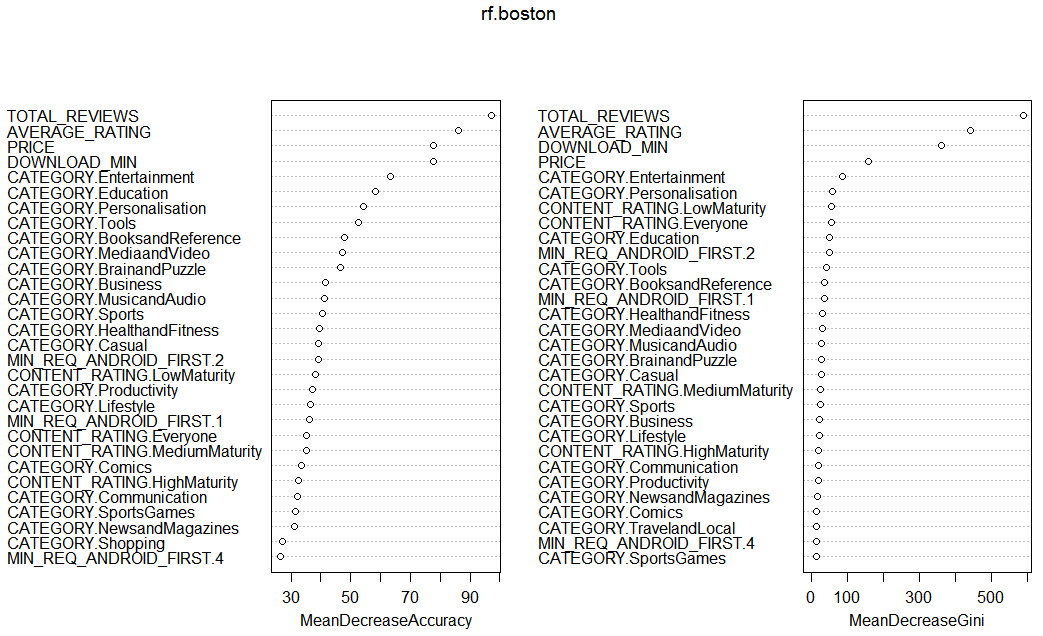
################### Random Forest for Oversampled Dataset #########################  
train\_oversampling$spam <- as.factor(train\_oversampling$spam)  
set.seed(1)  
rf.boston=randomForest(spam~.,data = train\_oversampling[,-1],mtry=7,importance=TRUE)  
#rf.boston  
yhat.rf = predict(rf.boston,newdata=test)  
boston.test= test$spam  
  
t<- table(boston.test,yhat.rf)  
#t  
Accuracy\_test\_bag <- (t[1,1] + t[2,2])/(t[1,1] + t[1,2] + t[2,1] + t[2,2])  
Accuracy\_test\_bag

## [1] 0.8346422

importance(rf.boston)

## 0 1  
## CATEGORY.BooksandReference 2.5360338 46.46339  
## CATEGORY.BrainandPuzzle 1.2480208 47.87803  
## CATEGORY.Business -2.5778519 43.44955  
## CATEGORY.CardsandCasino 6.2670492 26.21737  
## CATEGORY.Casual 0.5925599 41.34035  
## CATEGORY.Comics -3.8158273 38.12937  
## CATEGORY.Communication 1.0842386 31.29016  
## CATEGORY.Education 20.6646882 60.52342  
## CATEGORY.Entertainment 15.5846810 71.16601  
## CATEGORY.Finance 10.8798088 17.07729  
## CATEGORY.HealthandFitness 16.6569861 39.54772  
## CATEGORY.LibrariesandDemo 12.7625886 15.02513  
## CATEGORY.Lifestyle 2.2562492 38.14615  
## CATEGORY.MediaandVideo 5.8261619 50.36063  
## CATEGORY.Medical 2.5989409 16.15809  
## CATEGORY.MusicandAudio -2.4750445 41.46829  
## CATEGORY.NewsandMagazines 6.5630849 32.34323  
## CATEGORY.Personalisation -0.2117701 52.23288  
## CATEGORY.Photography 0.5197451 23.37783  
## CATEGORY.Productivity 10.0331592 35.65060  
## CATEGORY.Racing -3.6183051 24.95862  
## CATEGORY.Shopping -2.2104549 29.02362  
## CATEGORY.Social 1.4083315 16.26476  
## CATEGORY.Sports 0.9719599 44.11148  
## CATEGORY.SportsGames 1.3336732 33.33483  
## CATEGORY.Tools 0.3131113 51.55368  
## CATEGORY.Transportation -1.3565816 17.22507  
## CATEGORY.TravelandLocal 12.4860180 23.37699  
## CATEGORY.Weather -1.2118801 10.14784  
## PRICE 30.4819352 76.45289  
## CONTENT\_RATING.Everyone 9.8929764 33.15398  
## CONTENT\_RATING.HighMaturity 6.1020960 33.85372  
## CONTENT\_RATING.LowMaturity 1.0416789 39.51339  
## CONTENT\_RATING.MediumMaturity 1.0247809 34.59936  
## CONTENT\_RATING.NotRated 11.5846093 17.57621  
## DOWNLOAD\_MIN 10.7365796 84.80403  
## MIN\_REQ\_ANDROID\_FIRST.1 -8.7740162 37.20415  
## MIN\_REQ\_ANDROID\_FIRST.2 -2.3277503 40.41710  
## MIN\_REQ\_ANDROID\_FIRST.3 9.0777446 23.49807  
## MIN\_REQ\_ANDROID\_FIRST.4 4.5516431 27.47521  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 1.5294367 13.34290  
## TOTAL\_REVIEWS 6.2900439 99.41535  
## AVERAGE\_RATING 21.8263803 82.90430  
## MeanDecreaseAccuracy  
## CATEGORY.BooksandReference 47.89575  
## CATEGORY.BrainandPuzzle 46.46648  
## CATEGORY.Business 41.46470  
## CATEGORY.CardsandCasino 25.82782  
## CATEGORY.Casual 39.30841  
## CATEGORY.Comics 33.53681  
## CATEGORY.Communication 32.15556  
## CATEGORY.Education 58.17343  
## CATEGORY.Entertainment 63.17723  
## CATEGORY.Finance 18.76006  
## CATEGORY.HealthandFitness 39.34901  
## CATEGORY.LibrariesandDemo 18.47184  
## CATEGORY.Lifestyle 36.48264  
## CATEGORY.MediaandVideo 47.25905  
## CATEGORY.Medical 16.02222  
## CATEGORY.MusicandAudio 41.06407  
## CATEGORY.NewsandMagazines 31.25433  
## CATEGORY.Personalisation 54.35513  
## CATEGORY.Photography 22.20560  
## CATEGORY.Productivity 37.29107  
## CATEGORY.Racing 21.64837  
## CATEGORY.Shopping 27.00240  
## CATEGORY.Social 15.32016  
## CATEGORY.Sports 40.63793  
## CATEGORY.SportsGames 31.44456  
## CATEGORY.Tools 52.63411  
## CATEGORY.Transportation 16.34068  
## CATEGORY.TravelandLocal 25.61915  
## CATEGORY.Weather 9.21254  
## PRICE 77.57436  
## CONTENT\_RATING.Everyone 35.31723  
## CONTENT\_RATING.HighMaturity 32.32066  
## CONTENT\_RATING.LowMaturity 38.26270  
## CONTENT\_RATING.MediumMaturity 35.22474  
## CONTENT\_RATING.NotRated 17.83358  
## DOWNLOAD\_MIN 77.57243  
## MIN\_REQ\_ANDROID\_FIRST.1 36.15366  
## MIN\_REQ\_ANDROID\_FIRST.2 39.06716  
## MIN\_REQ\_ANDROID\_FIRST.3 22.46374  
## MIN\_REQ\_ANDROID\_FIRST.4 26.35773  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 12.87541  
## TOTAL\_REVIEWS 97.27524  
## AVERAGE\_RATING 86.22108  
## MeanDecreaseGini  
## CATEGORY.BooksandReference 36.614275  
## CATEGORY.BrainandPuzzle 28.105427  
## CATEGORY.Business 22.613829  
## CATEGORY.CardsandCasino 12.093445  
## CATEGORY.Casual 26.841173  
## CATEGORY.Comics 14.229462  
## CATEGORY.Communication 18.609231  
## CATEGORY.Education 50.305926  
## CATEGORY.Entertainment 84.640457  
## CATEGORY.Finance 6.319216  
## CATEGORY.HealthandFitness 31.032488  
## CATEGORY.LibrariesandDemo 6.494583  
## CATEGORY.Lifestyle 20.577960  
## CATEGORY.MediaandVideo 28.946869  
## CATEGORY.Medical 3.967536  
## CATEGORY.MusicandAudio 28.106312  
## CATEGORY.NewsandMagazines 16.888504  
## CATEGORY.Personalisation 57.134882  
## CATEGORY.Photography 8.867150  
## CATEGORY.Productivity 18.274021  
## CATEGORY.Racing 7.352752  
## CATEGORY.Shopping 8.801996  
## CATEGORY.Social 4.744043  
## CATEGORY.Sports 22.903585  
## CATEGORY.SportsGames 12.990600  
## CATEGORY.Tools 39.688238  
## CATEGORY.Transportation 5.466102  
## CATEGORY.TravelandLocal 13.670803  
## CATEGORY.Weather 2.518716  
## PRICE 156.641328  
## CONTENT\_RATING.Everyone 55.021472  
## CONTENT\_RATING.HighMaturity 18.914411  
## CONTENT\_RATING.LowMaturity 55.697465  
## CONTENT\_RATING.MediumMaturity 23.385914  
## CONTENT\_RATING.NotRated 12.499871  
## DOWNLOAD\_MIN 361.945667  
## MIN\_REQ\_ANDROID\_FIRST.1 35.666589  
## MIN\_REQ\_ANDROID\_FIRST.2 48.445543  
## MIN\_REQ\_ANDROID\_FIRST.3 10.577448  
## MIN\_REQ\_ANDROID\_FIRST.4 13.208053  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 3.388105  
## TOTAL\_REVIEWS 588.256494  
## AVERAGE\_RATING 440.382167

varImpPlot(rf.boston)



################ Boosting - AdaBoost #####################  
library(gbm)

## Warning: package 'gbm' was built under R version 3.3.3

## Loading required package: survival

## Loading required package: lattice

## Loading required package: splines

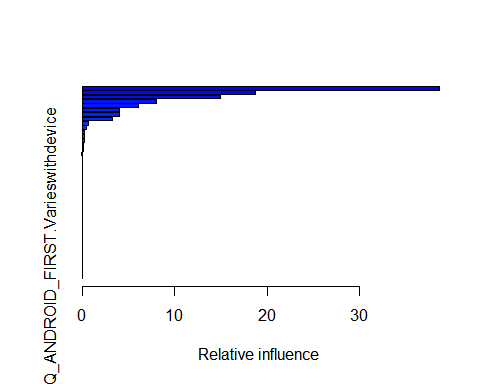
## Loading required package: parallel

## Loaded gbm 2.1.3

set.seed(1)  
  
############## Training the model ##############################  
boost.boston=gbm(factor(spam)~.,data=train\_rf,n.trees=5000,interaction.depth=4, dist="adaboost")  
boost.boston

## gbm(formula = factor(spam) ~ ., distribution = "adaboost", data = train\_rf,   
## n.trees = 5000, interaction.depth = 4)  
## A gradient boosted model with adaboost loss function.  
## 5000 iterations were performed.  
## There were 44 predictors of which 26 had non-zero influence.

summary(boost.boston)



## var  
## TOTAL\_REVIEWS TOTAL\_REVIEWS  
## DOWNLOAD\_MIN DOWNLOAD\_MIN  
## CATEGORY.Entertainment CATEGORY.Entertainment  
## PRICE PRICE  
## AVERAGE\_RATING AVERAGE\_RATING  
## CONTENT\_RATING.LowMaturity CONTENT\_RATING.LowMaturity  
## CONTENT\_RATING.Everyone CONTENT\_RATING.Everyone  
## CONTENT\_RATING.MediumMaturity CONTENT\_RATING.MediumMaturity  
## CATEGORY.Education CATEGORY.Education  
## CATEGORY.MediaandVideo CATEGORY.MediaandVideo  
## MIN\_REQ\_ANDROID\_FIRST.2 MIN\_REQ\_ANDROID\_FIRST.2  
## CATEGORY.MusicandAudio CATEGORY.MusicandAudio  
## CATEGORY.BrainandPuzzle CATEGORY.BrainandPuzzle  
## CATEGORY.ArcadeandAction CATEGORY.ArcadeandAction  
## CATEGORY.Personalisation CATEGORY.Personalisation  
## CATEGORY.BooksandReference CATEGORY.BooksandReference  
## CONTENT\_RATING.HighMaturity CONTENT\_RATING.HighMaturity  
## CATEGORY.Business CATEGORY.Business  
## CONTENT\_RATING.NotRated CONTENT\_RATING.NotRated  
## MIN\_REQ\_ANDROID\_FIRST.1 MIN\_REQ\_ANDROID\_FIRST.1  
## CATEGORY.Shopping CATEGORY.Shopping  
## CATEGORY.HealthandFitness CATEGORY.HealthandFitness  
## CATEGORY.Tools CATEGORY.Tools  
## CATEGORY.CardsandCasino CATEGORY.CardsandCasino  
## CATEGORY.Comics CATEGORY.Comics  
## CATEGORY.Sports CATEGORY.Sports  
## CATEGORY.Casual CATEGORY.Casual  
## CATEGORY.Communication CATEGORY.Communication  
## CATEGORY.Finance CATEGORY.Finance  
## CATEGORY.LibrariesandDemo CATEGORY.LibrariesandDemo  
## CATEGORY.Lifestyle CATEGORY.Lifestyle  
## CATEGORY.Medical CATEGORY.Medical  
## CATEGORY.NewsandMagazines CATEGORY.NewsandMagazines  
## CATEGORY.Photography CATEGORY.Photography  
## CATEGORY.Productivity CATEGORY.Productivity  
## CATEGORY.Racing CATEGORY.Racing  
## CATEGORY.Social CATEGORY.Social  
## CATEGORY.SportsGames CATEGORY.SportsGames  
## CATEGORY.Transportation CATEGORY.Transportation  
## CATEGORY.TravelandLocal CATEGORY.TravelandLocal  
## CATEGORY.Weather CATEGORY.Weather  
## MIN\_REQ\_ANDROID\_FIRST.3 MIN\_REQ\_ANDROID\_FIRST.3  
## MIN\_REQ\_ANDROID\_FIRST.4 MIN\_REQ\_ANDROID\_FIRST.4  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice  
## rel.inf  
## TOTAL\_REVIEWS 3.866802e+01  
## DOWNLOAD\_MIN 1.874854e+01  
## CATEGORY.Entertainment 1.490001e+01  
## PRICE 8.046166e+00  
## AVERAGE\_RATING 6.052037e+00  
## CONTENT\_RATING.LowMaturity 4.048534e+00  
## CONTENT\_RATING.Everyone 3.999759e+00  
## CONTENT\_RATING.MediumMaturity 3.292258e+00  
## CATEGORY.Education 6.902035e-01  
## CATEGORY.MediaandVideo 4.754408e-01  
## MIN\_REQ\_ANDROID\_FIRST.2 2.139837e-01  
## CATEGORY.MusicandAudio 2.125866e-01  
## CATEGORY.BrainandPuzzle 2.018015e-01  
## CATEGORY.ArcadeandAction 1.261566e-01  
## CATEGORY.Personalisation 1.138831e-01  
## CATEGORY.BooksandReference 7.279566e-02  
## CONTENT\_RATING.HighMaturity 4.770180e-02  
## CATEGORY.Business 3.415450e-02  
## CONTENT\_RATING.NotRated 2.348400e-02  
## MIN\_REQ\_ANDROID\_FIRST.1 9.999134e-03  
## CATEGORY.Shopping 7.325341e-03  
## CATEGORY.HealthandFitness 7.196382e-03  
## CATEGORY.Tools 4.471115e-03  
## CATEGORY.CardsandCasino 3.473400e-03  
## CATEGORY.Comics 2.141120e-05  
## CATEGORY.Sports 1.412949e-06  
## CATEGORY.Casual 0.000000e+00  
## CATEGORY.Communication 0.000000e+00  
## CATEGORY.Finance 0.000000e+00  
## CATEGORY.LibrariesandDemo 0.000000e+00  
## CATEGORY.Lifestyle 0.000000e+00  
## CATEGORY.Medical 0.000000e+00  
## CATEGORY.NewsandMagazines 0.000000e+00  
## CATEGORY.Photography 0.000000e+00  
## CATEGORY.Productivity 0.000000e+00  
## CATEGORY.Racing 0.000000e+00  
## CATEGORY.Social 0.000000e+00  
## CATEGORY.SportsGames 0.000000e+00  
## CATEGORY.Transportation 0.000000e+00  
## CATEGORY.TravelandLocal 0.000000e+00  
## CATEGORY.Weather 0.000000e+00  
## MIN\_REQ\_ANDROID\_FIRST.3 0.000000e+00  
## MIN\_REQ\_ANDROID\_FIRST.4 0.000000e+00  
## MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice 0.000000e+00

par(mfrow=c(1,2))  
  
confusion <- function(a, b){  
 tbl <- table(a, b)  
 mis <- 1 - sum(diag(tbl))/sum(tbl)  
 list(table = tbl, misclass.prob = mis)  
}  
gbm.perf(boost.boston)

## Using OOB method...

## Warning in gbm.perf(boost.boston): OOB generally underestimates the  
## optimal number of iterations although predictive performance is reasonably  
## competitive. Using cv.folds>0 when calling gbm usually results in improved  
## predictive performance.

## [1] 5000

################### Predicting the validation data #####################  
prediction <- predict.gbm(boost.boston, newdata = validation\_rf, n.trees = 5000, type = "response")   
  
CM1 <- table(prediction, validation\_rf$spam)  
mean((yhat.boost-boston.test)^2)

## [1] 1.105232

**KNN**

#Sampling Data  
data <- read.csv('app\_metadata\_cleaned\_removed\_min\_downloads\_above\_5m.csv')  
attach(data)  
  
  
data$CATEGORY <- as.factor(data$CATEGORY)  
data$PRICE <- as.numeric(data$PRICE)  
data$CONTENT\_RATING <- as.factor(data$CONTENT\_RATING)  
data$DOWNLOAD\_MIN <- as.numeric(data$DOWNLOAD\_MIN)  
data$DOWNLOAD\_MAX <- as.numeric(data$DOWNLOAD\_MAX)  
data$SIZE\_MEGABYTES <- as.numeric(data$SIZE\_MEGABYTES)  
data$MIN\_REQ\_ANDROID\_FIRST <- as.factor(data$MIN\_REQ\_ANDROID\_FIRST)  
data$TOTAL\_REVIEWS <- as.numeric(data$TOTAL\_REVIEWS)  
data$AVERAGE\_RATING <- as.numeric(data$AVERAGE\_RATING)  
data$X5RATING <- as.numeric(data$X5RATING)  
data$X4RATING <- as.numeric(data$X4RATING)  
data$X3RATING <- as.numeric(data$X3RATING)  
data$X2RATING <- as.numeric(data$X2RATING)  
data$X1RATING <- as.numeric(data$X1RATING)  
data$spam <- as.factor(data$spam)  
  
set.seed(12345)  
  
df <- data.frame(CATEGORY,PRICE, CONTENT\_RATING,DOWNLOAD\_MIN,MIN\_REQ\_ANDROID\_FIRST,TOTAL\_REVIEWS,AVERAGE\_RATING,spam)  
  
df$CATEGORY <- as.factor(df$CATEGORY)  
df$PRICE <- as.numeric(df$PRICE)  
df$CONTENT\_RATING <- as.factor(df$CONTENT\_RATING)  
df$DOWNLOAD\_MIN <- as.numeric(df$DOWNLOAD\_MIN)  
df$MIN\_REQ\_ANDROID\_FIRST <- as.factor(df$MIN\_REQ\_ANDROID\_FIRST)  
df$TOTAL\_REVIEWS <- as.numeric(df$TOTAL\_REVIEWS)  
df$AVERAGE\_RATING <- as.numeric(df$AVERAGE\_RATING)  
df$spam <- as.factor(df$spam)  
  
  
  
data <- df  
  
  
num.vars <- sapply(data, is.numeric)  
  
  
data[num.vars] <- lapply(data[num.vars], scale)  
  
Data <- data  
  
  
set.seed(123457)  
  
library(dummies)

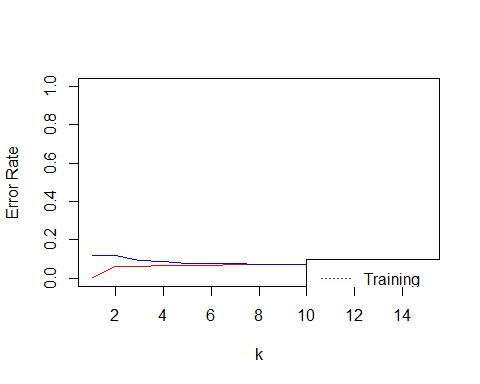
## dummies-1.5.6 provided by Decision Patterns

Data <- dummy.data.frame(Data, sep = ".", names = c("CATEGORY","CONTENT\_RATING","MIN\_REQ\_ANDROID\_FIRST"))  
  
colnames(Data)[which(names(Data) == "CATEGORY.Arcade and Action")] <- "CATEGORY.ArcadeandAction"  
colnames(Data)[which(names(Data) == "CATEGORY.Books & Reference")] <- "CATEGORY.BooksReference"  
colnames(Data)[which(names(Data) == "CATEGORY.Brain and Puzzle")] <- "CATEGORY.BrainandPuzzle"  
colnames(Data)[which(names(Data) == "CATEGORY.Cards and Casino")] <- "CATEGORY.CardsandCasino"  
colnames(Data)[which(names(Data) == "CATEGORY.Health & Fitness")] <- "CATEGORY.HealthFitness"  
colnames(Data)[which(names(Data) == "CATEGORY.Libraries & Demo")] <- "CATEGORY.LibrariesDemo"  
colnames(Data)[which(names(Data) == "CATEGORY.Media & Video")] <- "CATEGORY.MediaVideo"  
colnames(Data)[which(names(Data) == "CATEGORY.Music & Audio")] <- "CATEGORY.MusicAudio"  
colnames(Data)[which(names(Data) == "CATEGORY.News & Magazines")] <- "CATEGORY.NewsMAgazines"  
colnames(Data)[which(names(Data) == "CATEGORY.Sports Games")] <- "CATEGORY.SportsGames"  
colnames(Data)[which(names(Data) == "CATEGORY.Travel & Local")] <- "CATEGORY.TravelLocal"  
colnames(Data)[which(names(Data) == "CONTENT\_RATING.High Maturity")] <- "CONTENT\_RATING.HighMaturity"  
colnames(Data)[which(names(Data) == "CONTENT\_RATING.Low Maturity")] <- "CONTENT\_RATING.LowMaturity"  
colnames(Data)[which(names(Data) == "CONTENT\_RATING.Medium Maturity")] <- "CONTENT\_RATING.MediumMaturity"  
colnames(Data)[which(names(Data) == "CONTENT\_RATING.Not rated")] <- "CONTENT\_RATING.Notrated"  
colnames(Data)[which(names(Data) == "MIN\_REQ\_ANDROID\_FIRST.Varies with device")] <- "MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice"  
  
  
train <- sample(nrow(Data), .7 \* nrow(Data))  
df\_train <- Data[train,]  
df\_validation <- Data[-train, ]  
  
  
  
#Under Sampling Data  
#Taking all the observations with dependent variable = 1  
train\_under <- df\_train[df\_train$spam==1,]  
  
#Randomly select observations with dependent variable = 0  
zero\_spam <- df\_train[df\_train$spam==0,]  
  
set.seed(123457)  
rearrangedZero\_spams <- zero\_spam[sample(nrow(zero\_spam), length(train\_under$spam)),]  
  
train\_under <- rbind(train\_under, rearrangedZero\_spams)  
  
  
  
  
train\_over <- df\_train[df\_train$spam==1,]  
train\_1 <- train\_over  
for (i in seq(from=1, to=6, by=1)){  
 train\_over <- rbind(train\_over, train\_1)  
}  
train\_oversampling <- rbind(df\_train, train\_over)  
  
  
train\_input <- as.matrix(df\_train[,-45])  
train\_output <- as.vector(df\_train[,45])  
validate\_input <- as.matrix(df\_validation[,-45])  
  
  
  
library("caret")

## Loading required package: lattice

## Loading required package: ggplot2

library(class)  
kmax <- 15  
ER1 <- rep(0,kmax)  
ER2 <- rep(0,kmax)  
#  
for (i in 1:kmax){  
 prediction <- knn(train\_input, train\_input,train\_output, k=i)  
 prediction2 <- knn(train\_input, validate\_input, train\_output, k=i)  
 # The confusion matrix for training data is:  
 CM1 <- table(prediction, df\_train$spam)  
 # The training error rate is:  
 ER1[i] <- (CM1[1,2]+CM1[2,1])/sum(CM1)  
 # The confusion matrix for validation data is:   
 CM2 <- table(prediction2, df\_validation$spam)  
 ER2[i] <- (CM2[1,2]+CM2[2,1])/sum(CM2)  
}  
  
  
  
plot(c(1,kmax),c(0,1),type="n", xlab="k",ylab="Error Rate")  
lines(ER1,col="red")  
lines(ER2,col="blue")  
legend(10, 0.1, c("Training","Validation"),lty=c(3,1), col=c("red","blue"))



z <- which.min(ER2)  
cat("Minimum Validation Error k:", z)

## Minimum Validation Error k: 11

# Scoring at optimal k  
prediction <- knn(train\_input, train\_input,train\_output, k=3)  
prediction2 <- knn(train\_input, validate\_input,train\_output, k=3)  
  
#  
CM1 <- table( df\_train$spam, prediction)  
CM2 <- table( df\_validation$spam, prediction2)  
  
CM1

## prediction  
## 0 1  
## 0 4915 45  
## 1 286 102

CM2

## prediction2  
## 0 1  
## 0 2071 58  
## 1 152 11

ER3 <- (CM2[1,2]+CM2[2,1])/sum(CM2)  
ER3

## [1] 0.09162304

Accuracy <- (CM2[1,1]+CM2[2,2])/sum(CM2)  
  
Accuracy

## [1] 0.908377

#Under Sampling Data

data <- read.csv('app\_metadata\_cleaned\_removed\_min\_downloads\_above\_5m.csv')  
attach(data)  
  
  
data$CATEGORY <- as.factor(data$CATEGORY)  
data$PRICE <- as.numeric(data$PRICE)  
data$CONTENT\_RATING <- as.factor(data$CONTENT\_RATING)  
data$DOWNLOAD\_MIN <- as.numeric(data$DOWNLOAD\_MIN)  
data$DOWNLOAD\_MAX <- as.numeric(data$DOWNLOAD\_MAX)  
data$SIZE\_MEGABYTES <- as.numeric(data$SIZE\_MEGABYTES)  
data$MIN\_REQ\_ANDROID\_FIRST <- as.factor(data$MIN\_REQ\_ANDROID\_FIRST)  
data$TOTAL\_REVIEWS <- as.numeric(data$TOTAL\_REVIEWS)  
data$AVERAGE\_RATING <- as.numeric(data$AVERAGE\_RATING)  
data$X5RATING <- as.numeric(data$X5RATING)  
data$X4RATING <- as.numeric(data$X4RATING)  
data$X3RATING <- as.numeric(data$X3RATING)  
data$X2RATING <- as.numeric(data$X2RATING)  
data$X1RATING <- as.numeric(data$X1RATING)  
data$spam <- as.factor(data$spam)  
  
set.seed(12345)  
  
df <- data.frame(CATEGORY,PRICE, CONTENT\_RATING,DOWNLOAD\_MIN,MIN\_REQ\_ANDROID\_FIRST,TOTAL\_REVIEWS,AVERAGE\_RATING,spam)  
  
df$CATEGORY <- as.factor(df$CATEGORY)  
df$PRICE <- as.numeric(df$PRICE)  
df$CONTENT\_RATING <- as.factor(df$CONTENT\_RATING)  
df$DOWNLOAD\_MIN <- as.numeric(df$DOWNLOAD\_MIN)  
df$MIN\_REQ\_ANDROID\_FIRST <- as.factor(df$MIN\_REQ\_ANDROID\_FIRST)  
df$TOTAL\_REVIEWS <- as.numeric(df$TOTAL\_REVIEWS)  
df$AVERAGE\_RATING <- as.numeric(df$AVERAGE\_RATING)  
df$spam <- as.factor(df$spam)  
  
  
  
data <- df  
  
  
num.vars <- sapply(data, is.numeric)  
  
  
data[num.vars] <- lapply(data[num.vars], scale)  
  
Data <- data  
  
  
set.seed(123457)  
  
library(dummies)

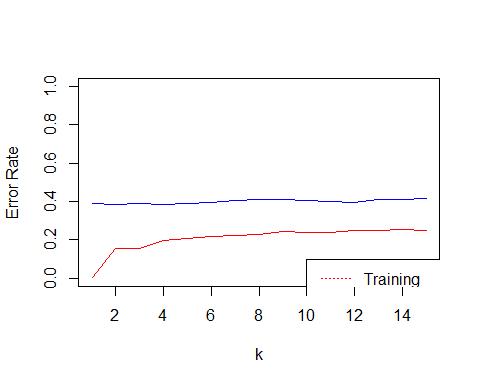
## dummies-1.5.6 provided by Decision Patterns

Data <- dummy.data.frame(Data, sep = ".", names = c("CATEGORY","CONTENT\_RATING","MIN\_REQ\_ANDROID\_FIRST"))  
  
colnames(Data)[which(names(Data) == "CATEGORY.Arcade and Action")] <- "CATEGORY.ArcadeandAction"  
colnames(Data)[which(names(Data) == "CATEGORY.Books & Reference")] <- "CATEGORY.BooksReference"  
colnames(Data)[which(names(Data) == "CATEGORY.Brain and Puzzle")] <- "CATEGORY.BrainandPuzzle"  
colnames(Data)[which(names(Data) == "CATEGORY.Cards and Casino")] <- "CATEGORY.CardsandCasino"  
colnames(Data)[which(names(Data) == "CATEGORY.Health & Fitness")] <- "CATEGORY.HealthFitness"  
colnames(Data)[which(names(Data) == "CATEGORY.Libraries & Demo")] <- "CATEGORY.LibrariesDemo"  
colnames(Data)[which(names(Data) == "CATEGORY.Media & Video")] <- "CATEGORY.MediaVideo"  
colnames(Data)[which(names(Data) == "CATEGORY.Music & Audio")] <- "CATEGORY.MusicAudio"  
colnames(Data)[which(names(Data) == "CATEGORY.News & Magazines")] <- "CATEGORY.NewsMAgazines"  
colnames(Data)[which(names(Data) == "CATEGORY.Sports Games")] <- "CATEGORY.SportsGames"  
colnames(Data)[which(names(Data) == "CATEGORY.Travel & Local")] <- "CATEGORY.TravelLocal"  
colnames(Data)[which(names(Data) == "CONTENT\_RATING.High Maturity")] <- "CONTENT\_RATING.HighMaturity"  
colnames(Data)[which(names(Data) == "CONTENT\_RATING.Low Maturity")] <- "CONTENT\_RATING.LowMaturity"  
colnames(Data)[which(names(Data) == "CONTENT\_RATING.Medium Maturity")] <- "CONTENT\_RATING.MediumMaturity"  
colnames(Data)[which(names(Data) == "CONTENT\_RATING.Not rated")] <- "CONTENT\_RATING.Notrated"  
colnames(Data)[which(names(Data) == "MIN\_REQ\_ANDROID\_FIRST.Varies with device")] <- "MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice"  
  
  
train <- sample(nrow(Data), .7 \* nrow(Data))  
df\_train <- Data[train,]  
df\_validation <- Data[-train, ]  
  
  
  
#Under Sampling Data  
#Taking all the observations with dependent variable = 1  
train\_under <- df\_train[df\_train$spam==1,]  
  
#Randomly select observations with dependent variable = 0  
zero\_spam <- df\_train[df\_train$spam==0,]  
  
set.seed(123457)  
rearrangedZero\_spams <- zero\_spam[sample(nrow(zero\_spam), length(train\_under$spam)),]  
  
train\_under <- rbind(train\_under, rearrangedZero\_spams)  
  
  
  
  
train\_over <- df\_train[df\_train$spam==1,]  
train\_1 <- train\_over  
for (i in seq(from=1, to=6, by=1)){  
 train\_over <- rbind(train\_over, train\_1)  
}  
train\_oversampling <- rbind(df\_train, train\_over)  
  
  
train\_input <- as.matrix(df\_train[,-45])  
train\_output <- as.vector(df\_train[,45])  
validate\_input <- as.matrix(df\_validation[,-45])  
  
  
###################################### under sampling ###########################  
  
  
  
  
train\_input <- as.matrix(train\_under[,-45])  
train\_output <- as.vector(train\_under[,45])  
validate\_input <- as.matrix(df\_validation[,-45])  
  
  
  
library("caret")

## Loading required package: lattice

## Loading required package: ggplot2

library(class)  
kmax <- 15  
ER1 <- rep(0,kmax)  
ER2 <- rep(0,kmax)  
#  
for (i in 1:kmax){  
 prediction <- knn(train\_input, train\_input,train\_output, k=i)  
 prediction2 <- knn(train\_input, validate\_input, train\_output, k=i)  
 # The confusion matrix for training data is:  
 CM1 <- table(prediction, train\_under$spam)  
 # The training error rate is:  
 ER1[i] <- (CM1[1,2]+CM1[2,1])/sum(CM1)  
 # The confusion matrix for validation data is:   
 CM2 <- table(prediction2, df\_validation$spam)  
 ER2[i] <- (CM2[1,2]+CM2[2,1])/sum(CM2)  
}  
  
  
  
plot(c(1,kmax),c(0,1),type="n", xlab="k",ylab="Error Rate")  
lines(ER1,col="red")  
lines(ER2,col="blue")  
legend(10, 0.1, c("Training","Validation"),lty=c(3,1), col=c("red","blue"))



z <- which.min(ER2)  
cat("Minimum Validation Error k:", z)

## Minimum Validation Error k: 4

# Scoring at optimal k  
prediction <- knn(train\_input, train\_input,train\_output, k=z)  
prediction2 <- knn(train\_input, validate\_input,train\_output, k=z)  
  
#  
CM1 <- table( train\_under$spam, prediction)  
CM2 <- table( df\_validation$spam, prediction2)  
  
CM1

## prediction  
## 0 1  
## 0 297 91  
## 1 58 330

CM2

## prediction2  
## 0 1  
## 0 1282 847  
## 1 49 114

ER3 <- (CM2[1,2]+CM2[2,1])/sum(CM2)  
ER3

## [1] 0.390925

Accuracy <- (CM2[1,1]+CM2[2,2])/sum(CM2)  
  
Accuracy

## [1] 0.609075

#Over Sampling Data

data <- read.csv('app\_metadata\_cleaned\_removed\_min\_downloads\_above\_5m.csv')  
attach(data)  
  
  
data$CATEGORY <- as.factor(data$CATEGORY)  
data$PRICE <- as.numeric(data$PRICE)  
data$CONTENT\_RATING <- as.factor(data$CONTENT\_RATING)  
data$DOWNLOAD\_MIN <- as.numeric(data$DOWNLOAD\_MIN)  
data$DOWNLOAD\_MAX <- as.numeric(data$DOWNLOAD\_MAX)  
data$SIZE\_MEGABYTES <- as.numeric(data$SIZE\_MEGABYTES)  
data$MIN\_REQ\_ANDROID\_FIRST <- as.factor(data$MIN\_REQ\_ANDROID\_FIRST)  
data$TOTAL\_REVIEWS <- as.numeric(data$TOTAL\_REVIEWS)  
data$AVERAGE\_RATING <- as.numeric(data$AVERAGE\_RATING)  
data$X5RATING <- as.numeric(data$X5RATING)  
data$X4RATING <- as.numeric(data$X4RATING)  
data$X3RATING <- as.numeric(data$X3RATING)  
data$X2RATING <- as.numeric(data$X2RATING)  
data$X1RATING <- as.numeric(data$X1RATING)  
data$spam <- as.factor(data$spam)  
  
set.seed(12345)  
  
df <- data.frame(CATEGORY,PRICE, CONTENT\_RATING,DOWNLOAD\_MIN,MIN\_REQ\_ANDROID\_FIRST,TOTAL\_REVIEWS,AVERAGE\_RATING,spam)  
  
df$CATEGORY <- as.factor(df$CATEGORY)  
df$PRICE <- as.numeric(df$PRICE)  
df$CONTENT\_RATING <- as.factor(df$CONTENT\_RATING)  
df$DOWNLOAD\_MIN <- as.numeric(df$DOWNLOAD\_MIN)  
df$MIN\_REQ\_ANDROID\_FIRST <- as.factor(df$MIN\_REQ\_ANDROID\_FIRST)  
df$TOTAL\_REVIEWS <- as.numeric(df$TOTAL\_REVIEWS)  
df$AVERAGE\_RATING <- as.numeric(df$AVERAGE\_RATING)  
df$spam <- as.factor(df$spam)  
  
  
  
data <- df  
  
  
num.vars <- sapply(data, is.numeric)  
  
  
data[num.vars] <- lapply(data[num.vars], scale)  
  
Data <- data  
  
  
set.seed(123457)  
  
library(dummies)

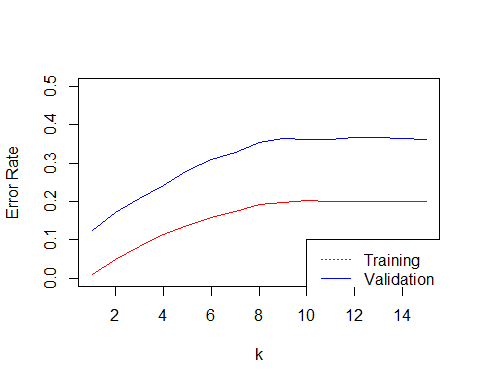
## dummies-1.5.6 provided by Decision Patterns

Data <- dummy.data.frame(Data, sep = ".", names = c("CATEGORY","CONTENT\_RATING","MIN\_REQ\_ANDROID\_FIRST"))  
  
colnames(Data)[which(names(Data) == "CATEGORY.Arcade and Action")] <- "CATEGORY.ArcadeandAction"  
colnames(Data)[which(names(Data) == "CATEGORY.Books & Reference")] <- "CATEGORY.BooksReference"  
colnames(Data)[which(names(Data) == "CATEGORY.Brain and Puzzle")] <- "CATEGORY.BrainandPuzzle"  
colnames(Data)[which(names(Data) == "CATEGORY.Cards and Casino")] <- "CATEGORY.CardsandCasino"  
colnames(Data)[which(names(Data) == "CATEGORY.Health & Fitness")] <- "CATEGORY.HealthFitness"  
colnames(Data)[which(names(Data) == "CATEGORY.Libraries & Demo")] <- "CATEGORY.LibrariesDemo"  
colnames(Data)[which(names(Data) == "CATEGORY.Media & Video")] <- "CATEGORY.MediaVideo"  
colnames(Data)[which(names(Data) == "CATEGORY.Music & Audio")] <- "CATEGORY.MusicAudio"  
colnames(Data)[which(names(Data) == "CATEGORY.News & Magazines")] <- "CATEGORY.NewsMAgazines"  
colnames(Data)[which(names(Data) == "CATEGORY.Sports Games")] <- "CATEGORY.SportsGames"  
colnames(Data)[which(names(Data) == "CATEGORY.Travel & Local")] <- "CATEGORY.TravelLocal"  
colnames(Data)[which(names(Data) == "CONTENT\_RATING.High Maturity")] <- "CONTENT\_RATING.HighMaturity"  
colnames(Data)[which(names(Data) == "CONTENT\_RATING.Low Maturity")] <- "CONTENT\_RATING.LowMaturity"  
colnames(Data)[which(names(Data) == "CONTENT\_RATING.Medium Maturity")] <- "CONTENT\_RATING.MediumMaturity"  
colnames(Data)[which(names(Data) == "CONTENT\_RATING.Not rated")] <- "CONTENT\_RATING.Notrated"  
colnames(Data)[which(names(Data) == "MIN\_REQ\_ANDROID\_FIRST.Varies with device")] <- "MIN\_REQ\_ANDROID\_FIRST.Varieswithdevice"  
  
  
train <- sample(nrow(Data), .7 \* nrow(Data))  
df\_train <- Data[train,]  
df\_validation <- Data[-train, ]  
  
  
  
#Under Sampling Data  
#Taking all the observations with dependent variable = 1  
train\_under <- df\_train[df\_train$spam==1,]  
  
#Randomly select observations with dependent variable = 0  
zero\_spam <- df\_train[df\_train$spam==0,]  
  
set.seed(123457)  
rearrangedZero\_spams <- zero\_spam[sample(nrow(zero\_spam), length(train\_under$spam)),]  
  
train\_under <- rbind(train\_under, rearrangedZero\_spams)  
  
  
  
  
train\_over <- df\_train[df\_train$spam==1,]  
train\_1 <- train\_over  
for (i in seq(from=1, to=6, by=1)){  
 train\_over <- rbind(train\_over, train\_1)  
}  
train\_oversampling <- rbind(df\_train, train\_over)  
  
  
train\_input <- as.matrix(df\_train[,-45])  
train\_output <- as.vector(df\_train[,45])  
validate\_input <- as.matrix(df\_validation[,-45])  
  
  
  
  
################################# Over sampling #####################################33  
  
  
  
  
train\_input <- as.matrix(train\_oversampling[,-45])  
train\_output <- as.vector(train\_oversampling[,45])  
validate\_input <- as.matrix(df\_validation[,-45])  
  
  
  
library("caret")

## Loading required package: lattice

## Loading required package: ggplot2

library(class)  
kmax <- 15  
ER1 <- rep(0,kmax)  
ER2 <- rep(0,kmax)  
#  
for (i in 1:kmax){  
 prediction <- knn(train\_input, train\_input,train\_output, k=i)  
 prediction2 <- knn(train\_input, validate\_input, train\_output, k=i)  
 # The confusion matrix for training data is:  
 CM1 <- table(prediction, train\_oversampling$spam)  
 # The training error rate is:  
 ER1[i] <- (CM1[1,2]+CM1[2,1])/sum(CM1)  
 # The confusion matrix for validation data is:   
 CM2 <- table(prediction2, df\_validation$spam)  
 ER2[i] <- (CM2[1,2]+CM2[2,1])/sum(CM2)  
}  
  
  
  
plot(c(1,kmax),c(0,0.5),type="n", xlab="k",ylab="Error Rate")  
lines(ER1,col="red")  
lines(ER2,col="blue")  
legend(10, 0.1, c("Training","Validation"),lty=c(3,1), col=c("red","blue"))



z <- which.min(ER2)  
cat("Minimum Validation Error k:", z)

## Minimum Validation Error k: 1

# Scoring at optimal k  
prediction <- knn(train\_input, train\_input,train\_output, k=z)  
prediction2 <- knn(train\_input, validate\_input,train\_output, k=z)  
  
#  
CM1 <- table( train\_oversampling$spam, prediction)  
CM2 <- table( df\_validation$spam, prediction2)  
  
CM1

## prediction  
## 0 1  
## 0 4886 74  
## 1 0 3104

CM2

## prediction2  
## 0 1  
## 0 1973 156  
## 1 131 32

ER3 <- (CM2[1,2]+CM2[2,1])/sum(CM2)  
ER3

## [1] 0.1252182

Accuracy <- (CM2[1,1]+CM2[2,2])/sum(CM2)  
  
Accuracy

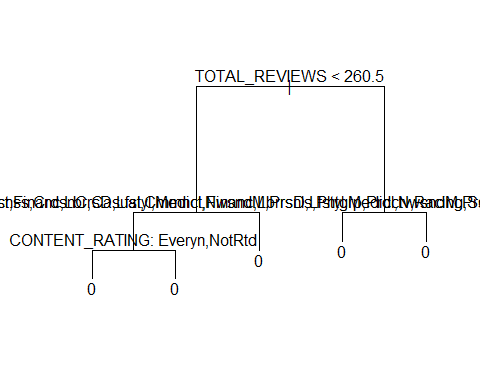
## [1] 0.8747818

**Trees**

# Uploading the file into R  
project <- read.csv("C:/Users/singhvi/Desktop/Data Mining/Project/DataMiningProject-master (1)/DataMiningProject-master/app\_metadata\_cleaned\_removed\_min\_downloads\_above\_5m.csv")  
# creating duplicate copy of project  
data <- project  
# creating factors of independent variables and changing variables from integer to numeric  
data$CATEGORY <- as.factor(data$CATEGORY)  
data$PRICE <- as.numeric(data$PRICE)  
data$CONTENT\_RATING <- as.factor(data$CONTENT\_RATING)  
data$DOWNLOAD\_MIN <- as.numeric(data$DOWNLOAD\_MIN)  
data$DOWNLOAD\_MAX <- as.numeric(data$DOWNLOAD\_MAX)  
data$SIZE\_MEGABYTES <- as.numeric(data$SIZE\_MEGABYTES)  
data$MIN\_REQ\_ANDROID\_FIRST <- as.factor(data$MIN\_REQ\_ANDROID\_FIRST)  
data$TOTAL\_REVIEWS <- as.numeric(data$TOTAL\_REVIEWS)  
data$AVERAGE\_RATING <- as.numeric(data$AVERAGE\_RATING)  
data$X5RATING <- as.numeric(data$X5RATING)  
data$X4RATING <- as.numeric(data$X4RATING)  
data$X3RATING <- as.numeric(data$X3RATING)  
data$X2RATING <- as.numeric(data$X2RATING)  
data$X1RATING <- as.numeric(data$X1RATING)  
data$spam <- as.factor(data$spam)  
  
#setting seed and dividing the dataset into Training and Test  
set.seed(12345)  
train <- sample(nrow(data),0.7\*nrow(data))  
project\_training <- data[train,]  
project\_Validation <- data[-train,]  
  
library(tree)  
library(ISLR)  
# creating the classification tree  
tree.project <- tree(spam~CATEGORY+PRICE+CONTENT\_RATING+DOWNLOAD\_MAX+DOWNLOAD\_MIN+TOTAL\_REVIEWS+AVERAGE\_RATING+X5RATING+X4RATING+X3RATING+X2RATING+X1RATING,data = project\_training)  
summary(tree.project)

##   
## Classification tree:  
## tree(formula = spam ~ CATEGORY + PRICE + CONTENT\_RATING + DOWNLOAD\_MAX +   
## DOWNLOAD\_MIN + TOTAL\_REVIEWS + AVERAGE\_RATING + X5RATING +   
## X4RATING + X3RATING + X2RATING + X1RATING, data = project\_training)  
## Variables actually used in tree construction:  
## [1] "TOTAL\_REVIEWS" "CATEGORY" "CONTENT\_RATING"  
## Number of terminal nodes: 5   
## Residual mean deviance: 0.4697 = 2510 / 5343   
## Misclassification error rate: 0.07386 = 395 / 5348

#plotting the tree  
plot(tree.project)  
text(tree.project, pretty=6)



# Creating confusion matrix of pridcted values in test data set   
tree.predict <- predict(tree.project, project\_Validation, type = "class")  
confmatrix <- table (project\_Validation$spam,tree.predict)  
confmatrix

## tree.predict  
## 0 1  
## 0 2136 0  
## 1 156 0

#calculating the accuracy  
accuracy <- (confmatrix[1,1]+confmatrix[2,2])/sum(confmatrix)  
accuracy

## [1] 0.9319372

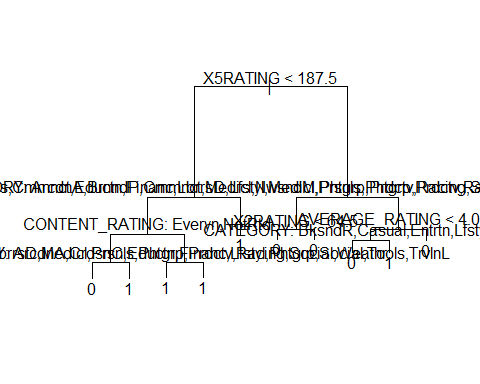
# calculating the sensitivity  
sensitivity <- confmatrix[2,2]/(confmatrix[2,2]+confmatrix[2,1])  
sensitivity

## [1] 0

#Under Sampling Data  
#Taking all the observations with dependent variable = 1  
train\_under <- project\_training[project\_training$spam==1,]  
  
#Randomly select observations with dependent variable = 0  
zeroObs <- project\_training[project\_training$spam==0,]  
set.seed(123457)  
rearrangedZeroObs <- zeroObs[sample(nrow(zeroObs), length(train\_under$spam)),]  
  
#Appending rows of randomly selected 0s in our undersampled data frame  
train\_under <- rbind(train\_under, rearrangedZeroObs)  
  
# creating the classification tree  
tree.project <- tree(spam~CATEGORY+PRICE+CONTENT\_RATING+DOWNLOAD\_MAX+DOWNLOAD\_MIN+TOTAL\_REVIEWS+AVERAGE\_RATING+X5RATING+X4RATING+X3RATING+X2RATING+X1RATING,data = train\_under)  
summary(tree.project)

##   
## Classification tree:  
## tree(formula = spam ~ CATEGORY + PRICE + CONTENT\_RATING + DOWNLOAD\_MAX +   
## DOWNLOAD\_MIN + TOTAL\_REVIEWS + AVERAGE\_RATING + X5RATING +   
## X4RATING + X3RATING + X2RATING + X1RATING, data = train\_under)  
## Variables actually used in tree construction:  
## [1] "X5RATING" "CATEGORY" "CONTENT\_RATING" "X2RATING"   
## [5] "AVERAGE\_RATING"  
## Number of terminal nodes: 10   
## Residual mean deviance: 1.028 = 801.6 / 780   
## Misclassification error rate: 0.2494 = 197 / 790

#plotting the tree  
plot(tree.project)  
text(tree.project, pretty=6)



# Creating confusion matrix of pridcted values in test data set   
tree.predict1 <- predict(tree.project, project\_Validation, type = "class")  
confmatrix1 <- table (project\_Validation$spam,tree.predict1)  
confmatrix1

## tree.predict1  
## 0 1  
## 0 1302 834  
## 1 42 114

#calculating the accuracy  
accuracy1 <- (confmatrix1[1,1]+confmatrix1[2,2])/sum(confmatrix1)  
accuracy1

## [1] 0.617801

# calculating the sensitivity  
sensitivity1 <- confmatrix1[2,2]/(confmatrix1[2,2]+confmatrix1[2,1])  
sensitivity1

## [1] 0.7307692

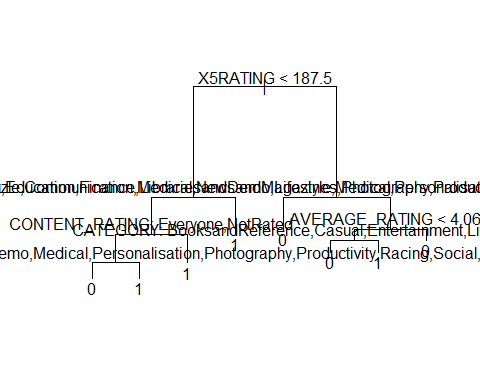
# Cross validation technique to find the best tree  
set.seed(123)  
cv.credit <- cv.tree(tree.project,FUN =prune.misclass, K=10)  
names(cv.credit)

## [1] "size" "dev" "k" "method"

cv.credit

## $size  
## [1] 10 8 7 5 4 2 1  
##   
## $dev  
## [1] 241 238 244 245 239 264 436  
##   
## $k  
## [1] -Inf 0.0 3.0 3.5 10.0 19.5 139.0  
##   
## $method  
## [1] "misclass"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

#pruning the tree with best no. of terminal nodes  
prune.credit <- prune.misclass(tree.project, best =8)  
plot(prune.credit)  
text(prune.credit, pretty=0)



summary(prune.credit)

##   
## Classification tree:  
## snip.tree(tree = tree.project, nodes = c(6L, 9L))  
## Variables actually used in tree construction:  
## [1] "X5RATING" "CATEGORY" "CONTENT\_RATING" "AVERAGE\_RATING"  
## Number of terminal nodes: 8   
## Residual mean deviance: 1.059 = 828.1 / 782   
## Misclassification error rate: 0.2494 = 197 / 790

# Creating confusion matrix of pridcted values in test data set with pruned tree  
tree.predict2 <- predict(prune.credit, project\_Validation, type = "class")  
confmatrix2 <- table (tree.predict2, project\_Validation$spam)  
confmatrix2

##   
## tree.predict2 0 1  
## 0 1302 42  
## 1 834 114

#calculating accuracy of pruned tree  
accuracy2 <- (confmatrix2[1,1]+confmatrix2[2,2])/sum(confmatrix2)  
accuracy2

## [1] 0.617801

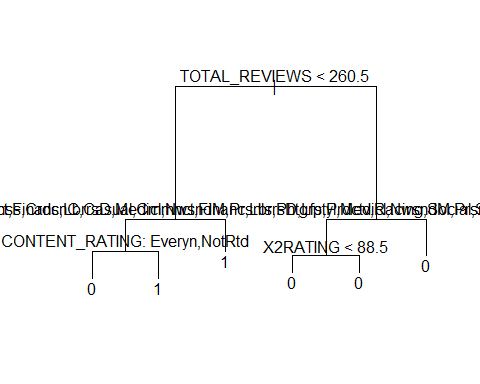
# calculating the sensitivity of pruned tree  
sensitivity2 <- confmatrix2[2,2]/(confmatrix2[1,2]+confmatrix2[2,2])  
sensitivity2

## [1] 0.7307692

#oversampling the data  
train\_over <- project\_training[project\_training$spam==1,]  
train\_1 <- train\_over  
for (i in seq(from=1, to=6, by=1)){  
 train\_over <- rbind(train\_over, train\_1)  
}  
train\_oversampling <- rbind(project\_training, train\_over)  
  
# running classification tree on oversampled data  
tree.project <- tree(spam ~ CATEGORY+PRICE+CONTENT\_RATING+DOWNLOAD\_MAX+DOWNLOAD\_MIN+TOTAL\_REVIEWS+AVERAGE\_RATING+X5RATING+X4RATING+X3RATING+X2RATING+X1RATING,data = train\_oversampling)  
summary(tree.project)

##   
## Classification tree:  
## tree(formula = spam ~ CATEGORY + PRICE + CONTENT\_RATING + DOWNLOAD\_MAX +   
## DOWNLOAD\_MIN + TOTAL\_REVIEWS + AVERAGE\_RATING + X5RATING +   
## X4RATING + X3RATING + X2RATING + X1RATING, data = train\_oversampling)  
## Variables actually used in tree construction:  
## [1] "TOTAL\_REVIEWS" "CATEGORY" "CONTENT\_RATING" "X2RATING"   
## Number of terminal nodes: 6   
## Residual mean deviance: 1.118 = 9060 / 8107   
## Misclassification error rate: 0.2966 = 2406 / 8113

#plotting the tree  
plot(tree.project)  
text(tree.project, pretty=6)



# Creating confusion matrix   
tree.predict3 <- predict(tree.project, project\_Validation, type = "class")  
confmatrix3 <- table (tree.predict3, project\_Validation$spam)  
confmatrix3

##   
## tree.predict3 0 1  
## 0 1489 57  
## 1 647 99

# calculating accuracy  
accuracy3 <- (confmatrix3[1,1]+confmatrix3[2,2])/sum(confmatrix3)  
accuracy3

## [1] 0.6928447

# calculating the sensitivity  
sensitivity3 <- confmatrix3[2,2]/(confmatrix3[1,2]+confmatrix3[2,2])  
sensitivity3

## [1] 0.6346154

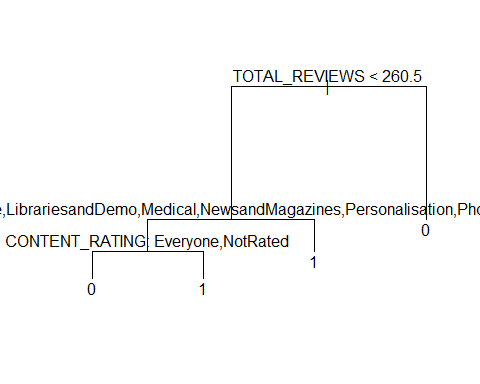
#running cross validation technique to fnd best tree  
set.seed(12345)  
cv.credit <- cv.tree(tree.project,FUN =prune.misclass, K=10)  
names(cv.credit)

## [1] "size" "dev" "k" "method"

cv.credit

## $size  
## [1] 6 4 3 1  
##   
## $dev  
## [1] 2644 2644 2850 3160  
##   
## $k  
## [1] -Inf 0 164 295  
##   
## $method  
## [1] "misclass"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

#pruning the tree  
prune.credit <- prune.misclass(tree.project, best =4)  
plot(prune.credit)  
text(prune.credit, pretty=0)



summary(prune.credit)

##   
## Classification tree:  
## snip.tree(tree = tree.project, nodes = 3L)  
## Variables actually used in tree construction:  
## [1] "TOTAL\_REVIEWS" "CATEGORY" "CONTENT\_RATING"  
## Number of terminal nodes: 4   
## Residual mean deviance: 1.165 = 9446 / 8109   
## Misclassification error rate: 0.2966 = 2406 / 8113

# calcuate confusion matrix  
tree.predict4 <- predict(prune.credit, project\_Validation, type = "class")  
confmatrix4 <- table (tree.predict4, project\_Validation$spam)  
confmatrix4

##   
## tree.predict4 0 1  
## 0 1489 57  
## 1 647 99

# calculating accuracy of pruned tree  
accuracy4 <- (confmatrix4[1,1]+confmatrix4[2,2])/sum(confmatrix4)  
accuracy4

## [1] 0.6928447

# calculating the sensitivity of pruned tree  
sensitivity4 <- confmatrix4[2,2]/(confmatrix4[1,2]+confmatrix4[2,2])  
sensitivity4

## [1] 0.6346154

## Including Plots

You can also embed plots, for example:



Note that the echo = FALSE parameter was added to the code chunk to prevent printing of the R code that generated the plot.

KNN

**Association Rules**

library(shiny)  
library(dummies)

## dummies-1.5.6 provided by Decision Patterns

#library(ggplot2)   
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

#library(MASS)   
#library(tree)   
library(ISLR)   
#library(cluster)   
#library(fpc)   
library(arules)

## Loading required package: Matrix

##   
## Attaching package: 'arules'

## The following objects are masked from 'package:base':  
##   
## abbreviate, write

library(arulesViz)

## Loading required package: grid

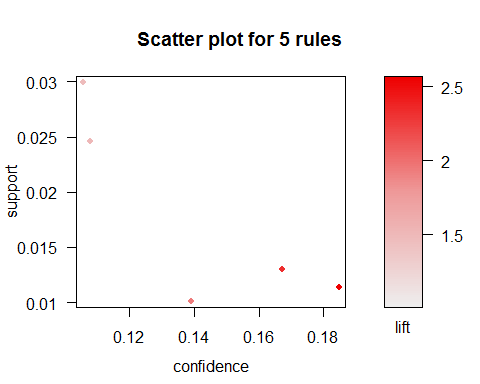
#library(GGally)  
#library(mice) #helps to fill values which are blank  
  
getwd();

## [1] "C:/Users/gupta/DataMiningProject/Akhil"

setwd("C:/Users/gupta/DataMiningProject")  
data<-read.csv(file="app\_metadata\_cleaned\_removed\_min\_downloads\_above\_5m.csv", header=TRUE) #Reading the dataset  
  
num.vars <- sapply(data, is.numeric)  
  
Data <- data  
  
  
#setting the variables to null so that they do not come in our analysis. These variables are useless for our computations.  
Data$APP\_ID <- NULL  
Data$APP\_NAME <- NULL  
Data$DOWNLOADS <- NULL  
Data$CURRENT\_VERSION <- NULL  
Data$LASTUPDATED <- NULL  
Data$DEVELOPER\_SITE <- NULL  
Data$DEVELOPER\_CONTACT <- NULL  
Data$DEVELOPER\_NAME <- NULL  
Data$MIN\_REQUIRED\_ANDROID <- NULL  
  
Data$MIN\_REQ\_ANDROID\_FIRST <- as.factor(Data$MIN\_REQ\_ANDROID\_FIRST) #declare variable as a factor  
Data1<-Data[c("CATEGORY","CONTENT\_RATING","MIN\_REQ\_ANDROID\_FIRST","spam")] #for the association rules we will consider the following  
Data2<-dummy.data.frame(Data1, names=c("")) #Create dummy variables  
Data3<-data.frame(sapply(Data1, as.numeric)) #use the variables as numericals  
Data1$MIN\_REQ\_ANDROID\_FIRST <- as.factor(Data$MIN\_REQ\_ANDROID\_FIRST)  
Data1$CATEGORY <- as.factor(Data$CATEGORY)  
Data1$CONTENT\_RATING <- as.factor(Data$CONTENT\_RATING)  
Data1$spam <- as.factor(Data$spam)  
###association rules applied ###  
rules<-apriori(Data1, parameter=list(supp=.01, conf=.1),appearance =list(default="lhs",rhs="spam=1"), control=list(verbose=F))  
  
rules<-sort(rules ,decreasing=TRUE,by="confidence")   
inspect(rules[1:5]) #inspect the top 5 rules which were sorted by confidence

## lhs rhs support confidence lift  
## [1] {CATEGORY=Entertainment,   
## MIN\_REQ\_ANDROID\_FIRST=2} => {spam=1} 0.01138743 0.1847134 2.561180  
## [2] {CATEGORY=Entertainment} => {spam=1} 0.01295812 0.1672297 2.318757  
## [3] {CATEGORY=Personalisation,   
## CONTENT\_RATING=LowMaturity} => {spam=1} 0.01007853 0.1392405 1.930667  
## [4] {CONTENT\_RATING=LowMaturity,   
## MIN\_REQ\_ANDROID\_FIRST=2} => {spam=1} 0.02460733 0.1081081 1.498994  
## [5] {CONTENT\_RATING=LowMaturity} => {spam=1} 0.02997382 0.1057248 1.465949

library(arulesViz)  
plot(rules,measure=c("confidence","support"),shading="lift") #Plot the cofidence vs support and shade the life to get a good idea of where the rules range is



plot(rules, method="matrix3D" ,measure="lift") #this is a 3d representtion of all the antecedents and consequents with lift shown

## Itemsets in Antecedent (LHS)  
## [1] "{CATEGORY=Entertainment,MIN\_REQ\_ANDROID\_FIRST=2}"   
## [2] "{CATEGORY=Entertainment}"   
## [3] "{CATEGORY=Personalisation,CONTENT\_RATING=LowMaturity}"  
## [4] "{CONTENT\_RATING=LowMaturity,MIN\_REQ\_ANDROID\_FIRST=2}"   
## [5] "{CONTENT\_RATING=LowMaturity}"   
## Itemsets in Consequent (RHS)  
## [1] "{spam=1}"

