Project ML Fall 2015. Appendix 2.1

Anastasiya Yarygina Udovenko, Manuel Aragonés Mora, Andrés Ponce de Leon

December 7, 2015

```
rm(list=ls())
setwd("~/Dropbox/MPP/ML/project")
```

II.1 Building Predictive Model for Poverty Threshold

```
# load required packages, set up parallel computation
```

```
library(recommenderlab)
library(ggplot2)
library(caret)
library(doParallel)
library(randomForest)
library(gbm)
require(ROCR)


cl <- makeCluster(detectCores() - 2)
clusterEvalQ(cl, library(foreach))
registerDoParallel(cl)   # register this cluster
```

Set seed

```
set.seed(99)

# read data
data<- read.csv("dataProject.csv", sep="," , header=TRUE)
dim(data)
```

```
## [1] 3146  633
```

Some data cleaning

```
# remove last three rows:
n<-dim(data)[1]
data<-data[1:(n-3),]

# get mean of Poverty.Percent:
summary(data$Poverty.Percent)[4]
```

```
##  Mean
## 17.22
```

```r
# generate new factor variable poverty
data$poverty <- ifelse(data$Poverty.Percent<= 17.22,
                       c("below"), c("above"))

data$poverty<- as.factor(data$poverty)

# remove regressors that we will not use: est variables
data2<- data[,-grep("est", colnames(data))]

# remove more regressors that we will not use
data3<- data2[, -which(names(data) %in% c("X.2", "X.1", "X", "NAME", "STATE_NAME",
                                 "STATE_FIPS", "CNTY_FIPS", "FIPS", "Poverty.Percent",
                                 "Median.Household.Income"))]
data4<- data3[,-grep("^X",colnames(data3))]
```

Our first outcome variable is "poverty threshold" it takes value 1 if county's poverty index is below national average and zero otherwise

```r
# define y as outcome variable
y<- data4$poverty
```

Some variables are imported in formats that are not suitable for our analysis. We transform them accordingly

```r
# convert socio-economic indicators into integers
cols <- data4[, -which(names(data4) %in% c("poverty"))]
cols<- cols[, -grep("rca", colnames(cols))]
cols <- data.frame(apply(cols, 2, as.integer))

# convert index of cometitiveness into factors
cols1<- data4[,grep("rca", colnames(data4))]
cols1<- data.frame(apply(cols1, 2, as.factor))


#bind socio-economic indicators and index of competitiveness
#into dataset of analysis:

data_new<- cbind(cols, cols1)

# add the dependent variable
data_new$y <- y
```

Models training and fitting

```r
# train and fit predictive models for the outcome "poverty" index:

# create data partition: %60 train and 40% test:
inTrain<- createDataPartition(y=data_new$y,
                              p=0.60, list=FALSE)
trainDf<- data_new[inTrain,]
testDf<- data_new[-inTrain,]

pnm="Counties"
```

We kate advantage of the R code provided with the lecture notes (week4), to construct "loss" and "lift" functions that we use to chose the best model specification fitting our dataset.

```
#deviance loss function
lossf = function(y,phat,wht=0.0000001) {
    #y should be 0/1
    #wht shrinks probs in phat towards .5, don't log 0!
    if(is.factor(y)) y = as.numeric(y)-1
    phat = (1-wht)*phat + wht*.5
    py = ifelse(y==1,phat,1-phat)
    return(-2*sum(log(py)))
}

#lift function
liftf = function(yl,phatl,dopl=TRUE) {
    if(is.factor(yl)) yl = as.numeric(yl)-1
    oo = order(-phatl)
    sy = cumsum(yl[oo])/sum(yl==1)
    if(dopl) {
        ii = (1:length(sy))/length(sy)
        plot(ii,sy,type='l',lwd=2,col='blue',xlab='% tried',ylab='% of successes',cex.lab=2)
        abline(0,1,lty=2)
    }
    return(sy)
}

# I initialize the list where we store the results
phatL = list()
```

1. Fit Logit Model

```
# set up
phatL$logit = matrix(0.0,nrow(testDf),1)

# fit logit
```

```
lgfit = glm(y~.,trainDf,family=binomial)
```

```
# predict
phat = predict(lgfit,testDf,type="response")
```

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

```
# store prediction
phatL$logit = matrix(phat,ncol=1)
```

2. Fit Random Forest Models

```r
# set up
p=ncol(trainDf)-1
mtryv = c(p,sqrt(p))
ntreev = c(500,1000)
setrf = expand.grid(mtryv,ntreev)
colnames(setrf)=c("mtry","ntree")
phatL$rf = matrix(0.0,nrow(testDf),nrow(setrf))

# train and fit
```

```r
for(i in 1:nrow(setrf)) {
   cat("on randomForest fit ",i,"\n")
   print(setrf[i,])


  frf = randomForest(y~.,data=trainDf,mtry=setrf[i,1],ntree=setrf[i,2])
   phat = predict(frf,newdata=testDf,type="prob")[,2]

   phatL$rf[,i]=phat # store results in matrix phalL
}
```

3. Fit Boosting Models

```r
idv = c(2,4)
ntv = c(1000,5000)
shv = c(.1,.01)
setboost = expand.grid(idv,ntv,shv)
colnames(setboost) = c("tdepth","ntree","shrink")
phatL$boost = matrix(0.0,nrow(testDf),nrow(setboost))

trainDfB = trainDf; trainDfB$y = as.numeric(trainDfB$y)-1
testDfB = testDf; testDfB$y = as.numeric(testDfB$y)-1

# train and fit
```

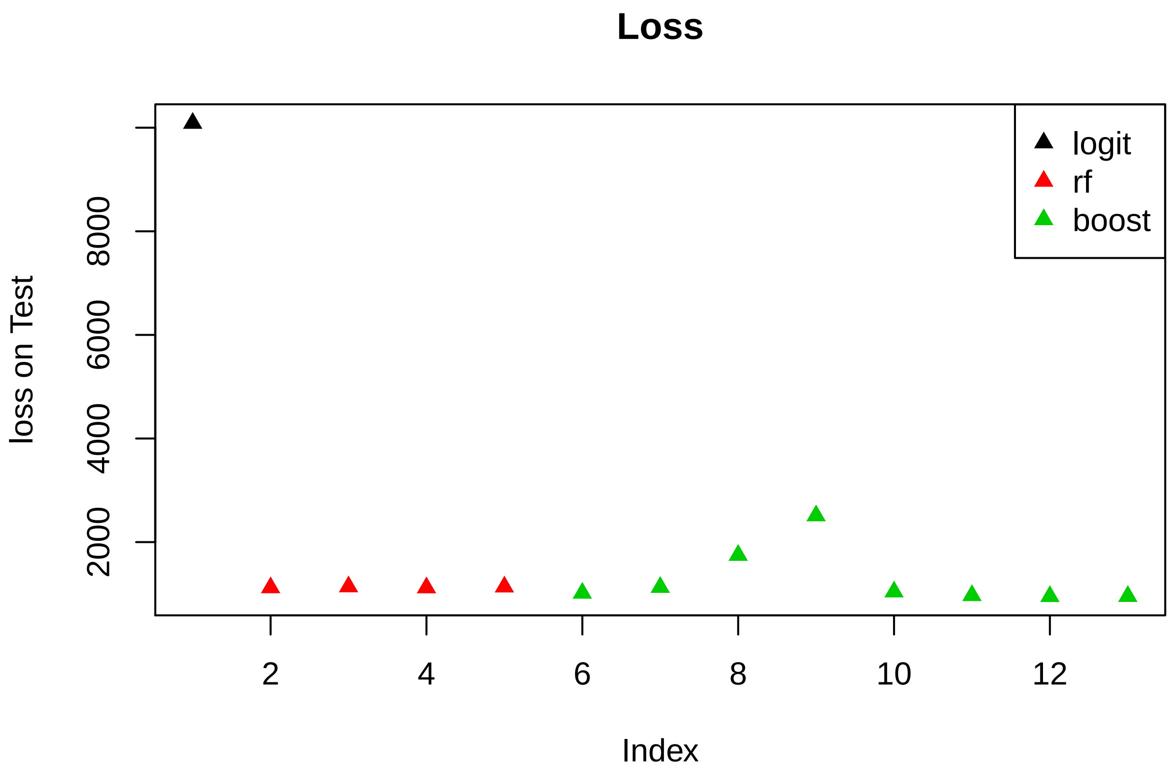
```r
for(i in 1:nrow(setboost)) {
   cat("on boosting fit ",i,"\n")
   print(setboost[i,])


   fboost = gbm(y~.,data=trainDfB,distribution="bernoulli",
                n.trees=setboost[i,2],interaction.depth=setboost[i,1],shrinkage=setboost[i,3])
   phat = predict(fboost,newdata=testDfB,n.trees=setboost[i,2],type="response")

   phatL$boost[,i] = phat # store results in the phatL matrix
}
```

4. Compute and plot deviance loss for Logit, RF and Boosting models

```r
lossL = list()
nmethod = length(phatL)
```

```r
for(i in 1:nmethod) {
   nrun = ncol(phatL[[i]])
   lvec = rep(0,nrun)
   print(nrun)
   for(j in 1:nrun) lvec[j] = lossf(testDf$y,phatL[[i]][,j])
   lossL[[i]]=lvec; names(lossL)[i] = names(phatL)[i]
}
```

```r
lossv = unlist(lossL)
par(mfrow=c(1,1))
plot(lossv,ylab="loss on Test",type="n", main = "Loss")
nloss=0
for(i in 1:nmethod) {
    ii = nloss + 1:ncol(phatL[[i]])
    points(ii,lossv[ii],col=i,pch=17)
    nloss = nloss + ncol(phatL[[i]])
}
legend("topright",legend=names(phatL),col=1:nmethod,pch=rep(17,nmethod))
```



From this figure we can see that Logit model has the highest loss. Tree-based models perform much better than logit.

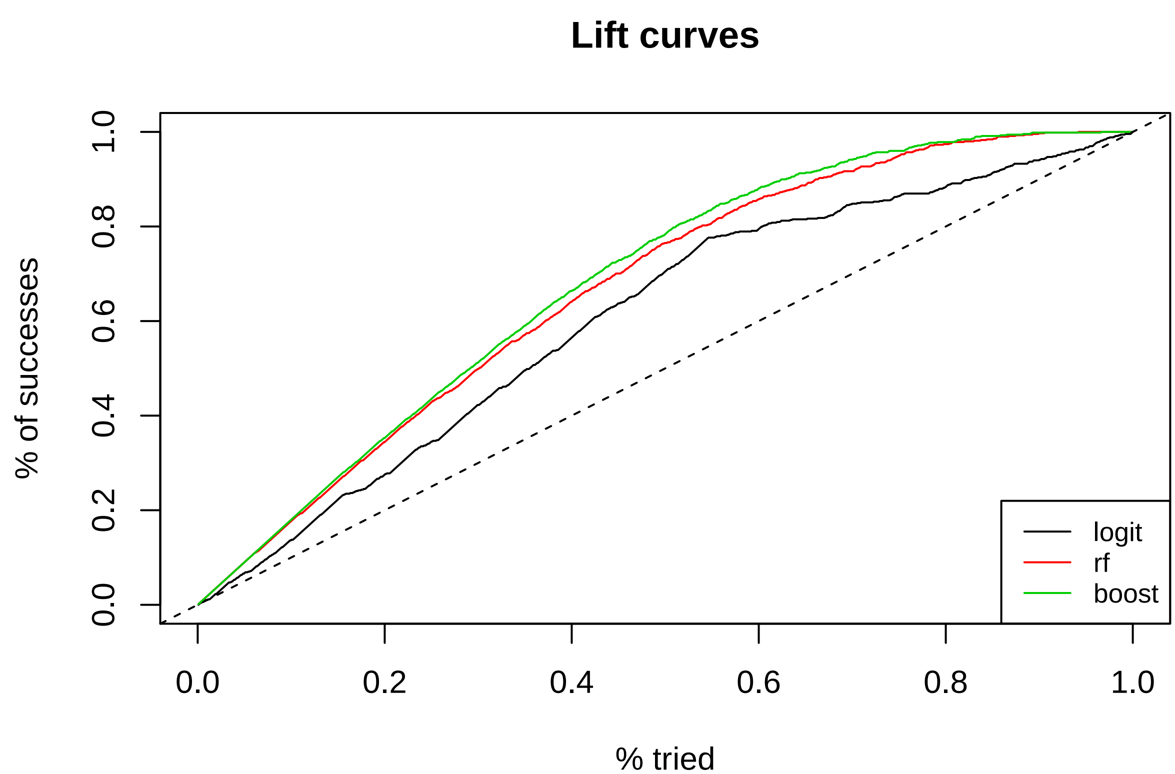
Picking the best specification

For each method's best spesicifation we build the lift curve

```r
#
```

```
nmethod = length(phatL)
phatBest = matrix(0.0,nrow(testDf),nmethod)
colnames(phatBest) = names(phatL)
for(i in 1:nmethod) {
   nrun = ncol(phatL[[i]])
   lvec = rep(0,nrun)
   print(nrun)
   for(j in 1:nrun) lvec[j] = lossf(testDf$y,phatL[[i]][,j])
   print(lvec)
   imin = which.min(lvec)
   cat("imin: ",imin,"\n")
   phatBest[,i] = phatL[[i]][,imin]
   phatBest[,i] = phatL[[i]][,1]
}
```

```
dfrac = (1:nrow(testDf))/nrow(testDf)
plot(c(0,1),c(0,1),xlab='% tried',ylab='% of successes',cex=2,type="n", main="Lift curves")
for(i in 1:ncol(phatBest)) {
    temp = liftf(testDf$y,phatBest[,i],dopl=FALSE)
    lines(dfrac,temp,type="l",col=i)
}
abline(0,1,lty=2)
legend("bottomright",legend=names(phatL),
       col=1:nmethod,lty=rep(1,nmethod), cex=0.8,)
```



So, the best methods and specifications are:

a) Random Forest specification 4 (mtry=18.27567,ntree=1000)

b) Boosting specification 8 (depth=4,n.trees=6000,shrinkage=.01)

Now, we fit models and plot ROC curves for the best rf, boosting and the unique logit model

```r
# rf
```

```r
rf_best = randomForest(y~.,data=trainDf,mtry=18.27567,ntree=1000)
```

```r
rf_pred_best = predict(rf_best,newdata=testDf,type="prob")[,2]

# gbm
```

```r
gbm_best = gbm(y~.,data=trainDfB,distribution="bernoulli",
               interaction.depth=4,n.trees=6000,shrinkage=.01)
```

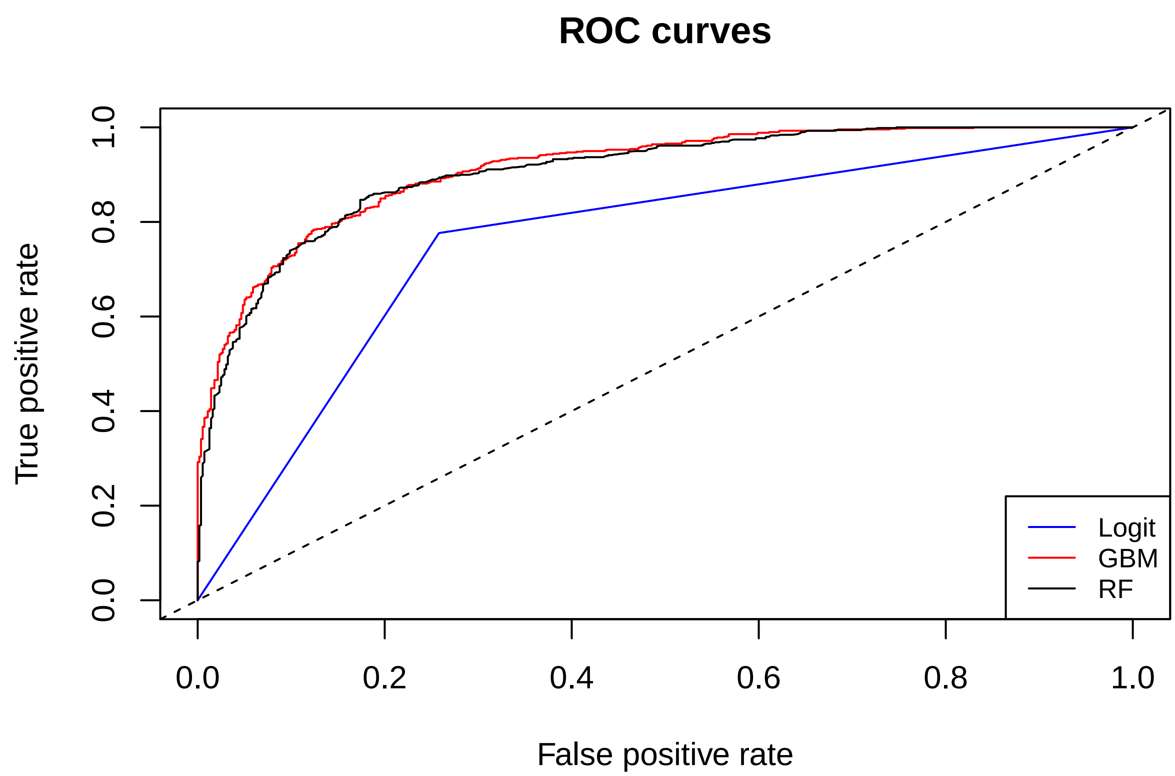
```r
gbm_pred_best = predict(gbm_best,newdata=testDfB, n.trees=6000,type="response")


#logit
```

```r
lgfit = glm(y~.,trainDf,family=binomial)
```

```r
phat = predict(lgfit,testDf,type="response")
```

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

```r
pred1 <- prediction(rf_pred_best, testDf$y)
pred9 <- prediction(phat, testDf$y)
pred4 <- prediction(gbm_pred_best, testDf$y)

perf1 <- performance(pred1,"tpr","fpr")
perf9 <- performance(pred9,"tpr","fpr")
perf4 <- performance(pred4,"tpr","fpr")

plot(perf9, col=4, main = "ROC curves")
lines(perf4@x.values[[1]], perf4@y.values[[1]], col = 2)
lines(perf1@x.values[[1]], perf1@y.values[[1]], col = 1)
abline(0,1,lty=2)
legend ("bottomright",
        legend=c("Logit", "GBM", "RF"),
        col=c("blue","red","black"), lty=c(1,1), cex=0.8)
```



Comparison of variable importance form different models

GLM

```
head(varImp(lgfit), n=20)
```

```
##                Overall
## POP2010       19655854
## POP10_SQMI    26336341
## POP2012       44093012
## POP12_SQMI    26234559
## WHITE         25475496
## BLACK         35977124
## AMERI_ES      61133435
## ASIAN          2861453
## HAWN_PI       58714781
## HISPANIC      40384000
## OTHER         26465806
## MALES        136529434
## AGE_UNDER5    23783434
## AGE_5_9       14516149
## AGE_10_14     10079067
## AGE_15_19     13107504
## AGE_20_24     43589069
## AGE_25_34     13792236
## AGE_35_44     16718007
## AGE_45_54     33068513
```

GBM

```r
head(summary(gbm_best, plotit=FALSE), n=20)
```

```
##                 var  rel.inf
## BLACK         BLACK 4.869648
## rca_2361   rca_2361 4.784303
## FHH_CHILD FHH_CHILD 3.867464
## AMERI_ES   AMERI_ES 3.817313
## rca_4529   rca_4529 3.775912
## MED_AGE_M MED_AGE_M 2.817762
## VACANT       VACANT 2.722560
## rca_2381   rca_2381 2.617724
## rca_5222   rca_5222 2.505626
## MARHH_CHD MARHH_CHD 2.465957
## rca_1133   rca_1133 2.340757
## rca_4461   rca_4461 1.963703
## rca_2383   rca_2383 1.898341
## rca_6241   rca_6241 1.819618
## rca_4451   rca_4451 1.757335
## ASIAN         ASIAN 1.645390
## PNTCNT_S   PNTCNT_S 1.551606
## rca_6214   rca_6214 1.424837
## AGE_85_UP AGE_85_UP 1.301156
## OTHER         OTHER 1.298451
```

RF

```r
head(importance(rf_best), n=20)
```

```
##            MeanDecreaseGini
## POP2010            8.533410
## POP10_SQMI        11.934628
## POP2012            8.528282
## POP12_SQMI        11.800887
## WHITE             12.934156
## BLACK             26.748412
## AMERI_ES          13.667884
## ASIAN             10.844283
## HAWN_PI            8.149513
## HISPANIC          11.313124
## OTHER             11.740289
## MULT_RACE         10.568144
## MALES              8.752840
## FEMALES            8.355474
## AGE_UNDER5         9.058235
## AGE_5_9            8.917370
## AGE_10_14          9.148620
## AGE_15_19          8.814471
## AGE_20_24         12.397916
## AGE_25_34          9.556742
```

We can observe that the varialbe importance varies across models. Interestingly, Boosting model, which is the model that has the best predictive capacity, has the largest number of predictors from the competitiveness index matrix.

```
stopCluster(cl)
```