Project ML Fall 2015. Appendix 2.2

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

December 7, 2015

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

II.2 Building Predictive Model for Median Household Income

```
# 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),]


# In this part of project we want to predict Median Houshold income
y<- data$Median.Household.Income


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

```
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", "poverty"))]
data4<- data3[,-grep("^X",colnames(data3))]
```

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

```
# convert socio-economic indicators into integers
cols<- data4[, -grep("rca", colnames(data4))]
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
data_new<- cbind(cols, cols1)


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

Model training and fitting

```
# 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,]
```

1 Fit using random forest

```
# mtry is the number of variables to try. We try 5 and 20
# ntree: we try 1000 and 500.

#mtry = 5, ntree=500
```

```
rffit1 = randomForest(y~.,data=trainDf,mtry=5,ntree=500)
```

```
rfpred1 = predict(rffit1,newdata=testDf)

# mtry=5, ntree=1000
```

```
rffit2 = randomForest(y~.,data=trainDf,mtry=5,ntree=1000)
```

```
rfpred2 = predict(rffit2,newdata=testDf)

# mtry = 20, ntree=500
```

```
rffit3 = randomForest(y~.,data=trainDf,mtry=20,ntree=500)
```

```
rfpred3 = predict(rffit3,newdata=testDf)

# mtry = 20, ntree=1000
```

```
rffit4 = randomForest(y~.,data=trainDf,mtry=20,ntree=1000)
```

```
rfpred4 = predict(rffit4,newdata=testDf)

rmserf1 = sqrt(mean((testDf$y-rfpred1)^2))
rmserf2 = sqrt(mean((testDf$y-rfpred2)^2))
rmserf3 = sqrt(mean((testDf$y-rfpred3)^2))
rmserf4 = sqrt(mean((testDf$y-rfpred4)^2))
```

```
## Random Forest, RMSE on validation dataset:
##  mtry = 5, ntree=500   is:  8047.782
##  mtry = 5, ntree=1000   is:  8032.396
##  mtry = 20, ntree=500   is:  7200.058
##  mtry = 20, ntree=1000  is   7194.155

## From RF model the best option is mtry = 20, ntree=1000
```

Fit using boosting

```
# depth 2, n.trees=1000, shrinkage =0.1
```

```
boostfit1 = gbm(y~.,data=trainDf,distribution="gaussian",
                interaction.depth=2,n.trees=1000,shrinkage=.1)
```

```
boostvalpred1=predict(boostfit1,newdata=testDf,n.trees=1000)

# depth 4, n.trees=1000,shrinkage =0.1
```

```
boostfit2 = gbm(y~.,data=trainDf,distribution="gaussian",
                interaction.depth=4,n.trees=1000,shrinkage=.1)
```

```
boostvalpred2=predict(boostfit2,newdata=testDf,n.trees=1000)


# depth 2, n.trees=5000,shrinkage =0.1
```

```
boostfit3 = gbm(y~.,data=trainDf,distribution="gaussian",
                interaction.depth=2,n.trees=5000,shrinkage=.1)
```

```
boostvalpred3=predict(boostfit3,newdata=testDf,n.trees=5000)

# depth 4, n.trees=5000,shrinkage =0.1
```

```
boostfit4 = gbm(y~.,data=trainDf,distribution="gaussian",
                interaction.depth=4,n.trees=5000,shrinkage=.1)
```

```
boostvalpred4=predict(boostfit4,newdata=testDf,n.trees=5000)


# depth 2, n.trees=1000, shrinkage =0.01
```

```
boostfit5 = gbm(y~.,data=trainDf,distribution="gaussian",
                interaction.depth=2,n.trees=1000,shrinkage=.01)
```

```
boostvalpred5=predict(boostfit5,newdata=testDf,n.trees=1000)

# depth 4, n.trees=1000,shrinkage =0.01
```

```
boostfit6 = gbm(y~.,data=trainDf,distribution="gaussian",
                interaction.depth=4,n.trees=1000,shrinkage=.01)
```

```
boostvalpred6=predict(boostfit6,newdata=testDf,n.trees=1000)


# depth 2, n.trees=5000,shrinkage =0.01
```

```
boostfit7 = gbm(y~.,data=trainDf,distribution="gaussian",
                interaction.depth=2,n.trees=5000,shrinkage=.01)
```

```
boostvalpred7=predict(boostfit7,newdata=testDf,n.trees=5000)

# depth 4, n.trees=5000,shrinkage =0.01
```

```
boostfit8 = gbm(y~.,data=trainDf,distribution="gaussian",
                interaction.depth=4,n.trees=5000,shrinkage=.01)
```

```
boostvalpred8=predict(boostfit8,newdata=testDf,n.trees=5000)
```

Get rmse on testing data

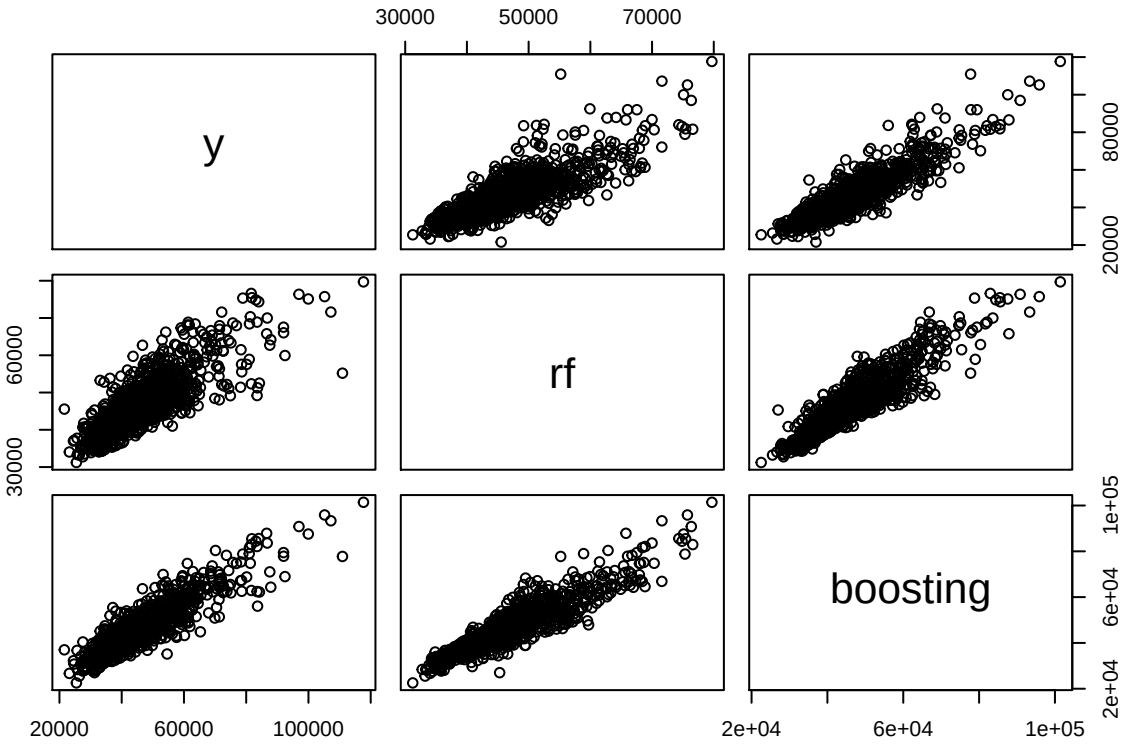
```
rmseboost1 = sqrt(mean((testDf$y-boostvalpred1)^2))
rmseboost2 = sqrt(mean((testDf$y-boostvalpred2)^2))
rmseboost3 = sqrt(mean((testDf$y-boostvalpred3)^2))
rmseboost4 = sqrt(mean((testDf$y-boostvalpred4)^2))
rmseboost5 = sqrt(mean((testDf$y-boostvalpred5)^2))
rmseboost6 = sqrt(mean((testDf$y-boostvalpred6)^2))
rmseboost7 = sqrt(mean((testDf$y-boostvalpred7)^2))
rmseboost8 = sqrt(mean((testDf$y-boostvalpred8)^2))
```

```
## Boosting, RMSE on testing dataset:
##  depth 2,   n.trees=1000, shrinkage =0.1  is:  5554.8
##  depth 4,   n.trees=1000, shrinkage =0.1  is:  5476.972
##  depth 2,   n.trees=5000, shrinkage =0.1  is:  5498.785
##  depth 4,   n.trees=5000, shrinkage =0.1  is:  5480.218
##  depth 2,   n.trees=1000, shrinkage =0.01 is:  6801.371
##  depth 4,   n.trees=1000, shrinkage =0.01 is:  6042.51
##  depth 2,   n.trees=5000, shrinkage =0.01 is:  5621.087
##  depth 4,   n.trees=5000, shrinkage =0.01 is:  5326.497

## The best fit from Boosting if for depth 4,  n.trees=5000, shrinkage =0.01
```

```
y<- testDf$y
rf<- rfpred4
boosting<- boostvalpred8

#plot the best fits from each group of models
pairs(cbind(y,rf,boosting))
```




```
print(cor(cbind(y,rf,boosting)))
```

```
##                   y        rf  boosting
## y         1.0000000 0.8129388 0.8911592
## rf        0.8129388 1.0000000 0.9161324
## boosting 0.8911592 0.9161324 1.0000000
```

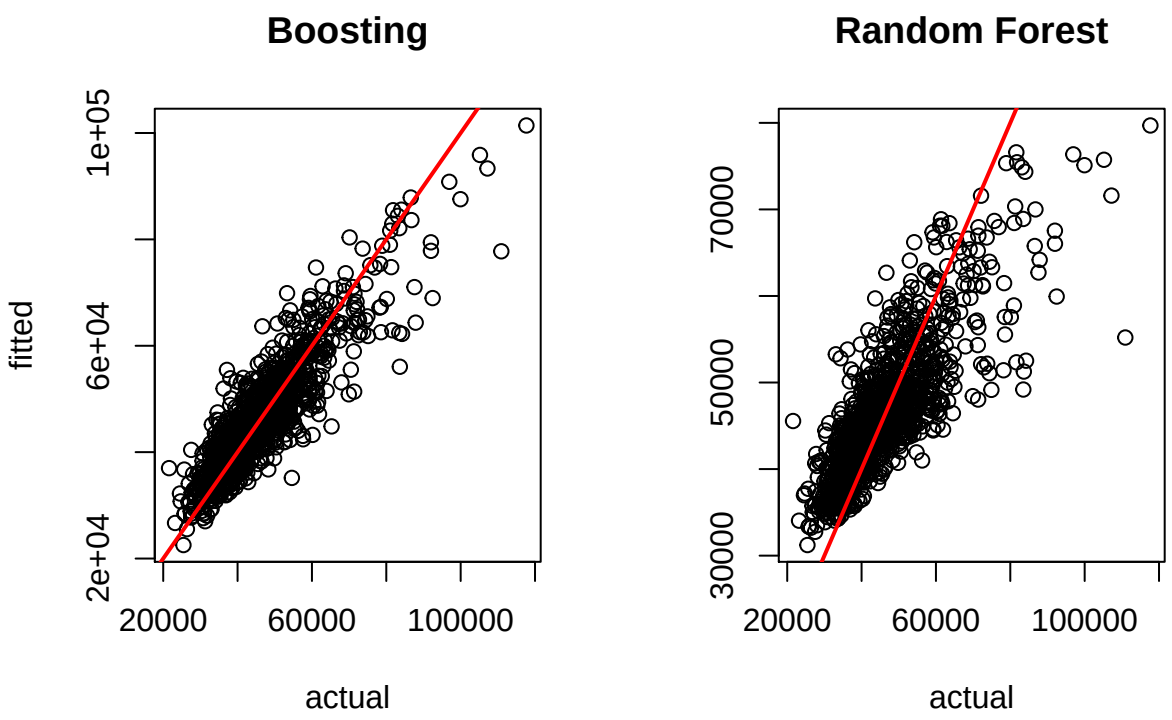
```
# var importance boosting
head(summary(boostfit8, plotit=FALSE), n=20)
```

```
##                   var  rel.inf
## rca_5415     rca_5415 8.726229
## rca_4529     rca_4529 5.472310
## ASIAN           ASIAN 5.168479
## VACANT         VACANT 3.834971
## rca_4471     rca_4471 3.798647
## rca_6116     rca_6116 3.214426
## MARHH_CHD   MARHH_CHD 3.214350
## POP10_SQMI POP10_SQMI 2.841777
## rca_5416     rca_5416 2.352813
## rca_4461     rca_4461 1.912648
## rca_2381     rca_2381 1.852743
## FHH_CHILD   FHH_CHILD 1.803999
## rca_6241     rca_6241 1.783765
## POP12_SQMI POP12_SQMI 1.598140
## rca_2361     rca_2361 1.583794
## rca_1133     rca_1133 1.548054
## rca_7224     rca_7224 1.412911
## rca_4451     rca_4451 1.362849
## WHITE           WHITE 1.293358
## MED_AGE_M   MED_AGE_M 1.292223
```

```
head(importance(rffit4), n=20)
```

```
##            IncNodePurity
## POP2010       2077449787
## POP10_SQMI    5082578184
## POP2012       2639942405
## POP12_SQMI    6022587267
## WHITE         4073373880
## BLACK         3261925293
## AMERI_ES      1786009218
## ASIAN         6824962260
## HAWN_PI       2112224355
## HISPANIC      1848020293
## OTHER         1831589607
## MULT_RACE     2183968384
## MALES         2142423342
## FEMALES       2054066958
## AGE_UNDER5    2312553982
## AGE_5_9       3285406954
## AGE_10_14     3462768960
## AGE_15_19     1873626495
## AGE_20_24     1939506253
## AGE_25_34     1995387114
```

```
# plot fitted vs actual y
par(mfrow=c(1,2),oma=c(0,0,1.5,0)) # 4 plot frames
plot(testDf$y,boostvalpred8, main="Boosting", xlab="actual",ylab="fitted")
abline(0,1,col="red",lwd=2)
plot(testDf$y,rfpred4, main="Random Forest", xlab="actual",ylab="")
abline(0,1,col="red",lwd=2)
title("Fitted and Actual y from 2 models", outer=TRUE)
```




```
stopCluster(cl)
```