# Random Forests Worksheet - solutions

### 

### Cleaning data and using the randomForest package in R

1. Given the following toy data, consider a single tree within a randomForest.

Give an example of which rows in the toy data could be used for

1) creating the tree and

2) the OOB sample for that tree.

3) In addition, give an example of the set of variables that could be used at a single split in this tree given mtry= 2 and we are predicting 'Species'.

Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
 5.5 3.5 1.3 0.2 setosa  
 5.2 3.4 1.4 0.2 setosa  
 6.3 2.9 5.6 1.8 virginica  
 7.7 2.8 6.7 2.0 virginica  
 5.1 3.8 1.5 0.3 setosa  
 6.5 2.8 4.6 1.5 versicolor  
 4.9 2.5 4.5 1.7 virginica  
 4.6 3.1 1.5 0.2 setosa  
 6.0 2.9 4.5 1.5 versicolor

1. Consider variable j. The Variable Importance Score =

OOB error with permuted j - OOB error without permuted j

What does a high variable importance score imply? One sentence only.

**Aim: Clean income.csv dataset and use randomForest algorithm to predict income class**

1. Set your working directory (where you have stored your datafile) using either setwd() or the 'Session' menu

setwd("C:/Users/Tim/OneDrive/SSA\_bigdata\_course/trees")

1. Install relevant packages: randomForest

install.packages('randomForest')

1. Load packages:

library('randomForest')

library('ggplot2')  
library('tidyr') #we will use gather()

1. Load dataset and save as 'a'. Delete first column at the same time

a<- read.csv('income.csv')[,-1]

1. We should do some cleaning: neaten up some of the variables Let's look at table summaries for three variables: occupation, education, marital status, workclass

table(a$education, a$class)

table(a$occupation, a$class)

table(a$marital.status, a$class)

table(a$workclass, a$class)

Are any of the category levels representing a similar idea and have similar stats with respect to the class we're predicting?

1. Creating clear distinct levels within a variable helps with the final prediction. Before we can combine levels, first convert factor variables to character variables - this allows us to change the names of the levels

a$occupation = as.character(a$occupation)  
a$education = as.character(a$education)  
a$marital.status = as.character(a$marital.status)  
a$workclass = as.character(a$workclass)

1. Replace variable levels with other labels. Target levels with low numbers or similar meanings/stats

We'll be using the function gsub, look it up under help.

?gsub

**Occupation:**

table(a$occupation, a$class)

##   
## <=50K >50K  
## Adm-clerical 2991 458  
## Armed-Forces 8 1  
## Craft-repair 2825 860  
## Exec-managerial 1917 1818  
## Farming-fishing 768 111  
## Handlers-cleaners 1116 73  
## Machine-op-inspct 1463 224  
## Other-service 2671 106  
## Priv-house-serv 89 1  
## Prof-specialty 2043 1650  
## Protective-serv 403 203  
## Sales 2436 928  
## Tech-support 593 257  
## Transport-moving 1186 305

#combine Priv-house-serv and Protective-Serv  
a$occupation<- gsub("Priv-house-serv", "Service", a$occupation)  
a$occupation<- gsub("Other-service", "Service", a$occupation)

**Education:**

table(a$education, a$class)

##   
## <=50K >50K  
## 10th 695 57  
## 11th 902 55  
## 12th 305 26  
## 1st-4th 38 1  
## 5th-6th 75 3  
## 7th-8th 408 29  
## 9th 328 22  
## Assoc-acdm 698 241  
## Assoc-voc 910 323  
## Bachelors 2643 1975  
## Doctorate 75 239  
## HS-grad 7668 1541  
## Masters 635 849  
## Preschool 15 0  
## Prof-school 121 367  
## Some-college 4993 1267

#create highschool dropout level - "HS.dropout", check for similar stats  
# | means 'or'  
a$education<- gsub("10th|11th|12th|9th", "HS.dropout", a$education)  
  
#create primary school dropout level and include preschool - "PS.dropout"  
a$education<- gsub("1st-4th|5th-6th|7th-8th|Preschool", "PS.dropout", a$education)  
  
#meld 'some college' with HS-grad  
a$education<- gsub("Some-college", "HS-grad", a$education)  
  
#merge Assoc-acdm and Assoc-voc. Similar stats  
a$education<- gsub("Assoc-acdm|Assoc-voc", "Associates", a$education)

**marital.status:**

table(a$marital.status, a$class)

##   
## <=50K >50K  
## Divorced 3564 429  
## Married-AF-spouse 11 10  
## Married-civ-spouse 6839 5959  
## Married-spouse-absent 208 25  
## Never-married 8435 441  
## Separated 765 60  
## Widowed 687 71

#combine married-AF-spouse (tiny) and Married-civ-spouse.   
a$marital.status<- gsub("Married-AF-spouse|Married-civ-spouse", "Married", a$marital.status)  
  
# combine Divorced, Married-spouse-absent and Separated b/c similar meaning  
a$marital.status<- gsub("Divorced|Separated|Married-spouse-absent","Prev.Married", a$marital.status)

**workclass:**

table(a$workclass, a$class)

##   
## <=50K >50K  
## Federal-gov 550 336  
## Local-gov 1381 575  
## Private 15594 4541  
## Self-emp-inc 433 558  
## Self-emp-not-inc 1654 659  
## State-gov 884 326  
## Without-pay 13 0

#remove class 'Without-pay' - class is obvious  
a<- a[!(a$workclass == 'Without-pay'),]  
  
# combine non-fed government  
a$workclass<- gsub("Local-gov|State-gov","lower.gov", a$workclass)

Convert back to factor:

a$occupation = as.factor(a$occupation)  
a$education = as.factor(a$education)  
a$marital.status = as.factor(a$marital.status)  
a$workclass = as.factor(a$workclass)

View cleaned data:

summary(a)

1. Note the ratio of cases which are <=50K and >50k. What could this mean for the predictive power of the model?

We won't worry about doing anything about it in this worksheet.

1. Create test and train set.

set.seed(100)  
sample<- sample(1:nrow(a), 2/3\*nrow(a), replace=F)  
train<- a[sample,]  
test<- a[-sample,]

1. Build a first-run random forest model. We will tune mtry next, set at sqrt(number of variables) for now. Set number of trees = 1000.

# look up which arguments you can use in randomForest  
?randomForest  
#examples are at the bottom of the help file

We will use the arguments: formula, data, mtry, ntree, importance, xtest, ytest

rf.fit<- randomForest(class~., data=train, mtry=3, ntree=1000, xtest=test[,1:9],  
 ytest=test[,10])  
  
#view results  
rf.fit

##   
## Call:  
## randomForest(formula = class ~ ., data = train, mtry = 3, ntree = 1000, xtest = test[, 1:9], ytest = test[, 10])   
## Type of random forest: classification  
## Number of trees: 1000  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 17.29%  
## Confusion matrix:  
## <=50K >50K class.error  
## <=50K 12394 1249 0.09154878  
## >50K 1920 2764 0.40990606  
## Test set error rate: 16.95%  
## Confusion matrix:  
## <=50K >50K class.error  
## <=50K 6183 670 0.0977674  
## >50K 883 1428 0.3820857

# OOB error for classification is err.rate of all trees up to i-th tree. Which row are we most interested in?  
rf.fit$err.rate[1000]

## [1] 0.1729143

1. Manually tune mtry parameter with a for-loop. Build 6 trees using mtry = 1,2,3,4,5 and plot OOB error and test error vs. mtry. Similar to tuning cp in single trees.

#set up two NULL vectors to contain the OOB error rate and test error rate  
oob.err<-double(5)  
test.err<-double(5)  
  
for(i in 1:5){  
 rf.fit<-randomForest(class~., data=train, mtry=i, ntree=1000, xtest=test[,1:9],  
 ytest=test[,10])  
 oob.err[i]<-rf.fit$err.rate[1000]  
 test.err[i]<- rf.fit$test$err.rate[1000]  
   
 #print progress of each i while computing  
 cat(i," ")  
}

## 1 2 3 4 5

#view two original vectors  
oob.err

## [1] 0.1764610 0.1672396 0.1725323 0.1796257 0.1858460

test.err

## [1] 0.1759057 0.1623745 0.1695766 0.1772152 0.1823440

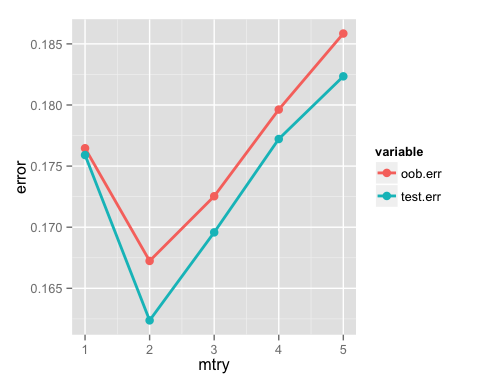
#plot error vectors against mtry  
  
#first create data frame with all data  
plot.df<- data.frame('mtry' = c(1:5), 'oob.err'=oob.err, 'test.err' = test.err)  
print(plot.df)

## mtry oob.err test.err  
## 1 1 0.1764610 0.1759057  
## 2 2 0.1672396 0.1623745  
## 3 3 0.1725323 0.1695766  
## 4 4 0.1796257 0.1772152  
## 5 5 0.1858460 0.1823440

plot.df<- gather(plot.df, key=variable, value =value, -mtry)  
print(plot.df)

## mtry variable value  
## 1 1 oob.err 0.1764610  
## 2 2 oob.err 0.1672396  
## 3 3 oob.err 0.1725323  
## 4 4 oob.err 0.1796257  
## 5 5 oob.err 0.1858460  
## 6 1 test.err 0.1759057  
## 7 2 test.err 0.1623745  
## 8 3 test.err 0.1695766  
## 9 4 test.err 0.1772152  
## 10 5 test.err 0.1823440

#gathering is often required to work well with ggplot 'grammar'  
  
#plot using ggplot  
ggplot(data= plot.df, aes(x=mtry, y=value, colour=variable)) +  
 geom\_line(size =1) +  
 geom\_point(size=3)+  
 labs(y='error')



Which mtry is optimal? Copy and paste plot below:

1. Make final forest and look at test set confusion matrix, print below. Better than single tree?

#set importance = TRUE  
rf.fit<- randomForest(class~., data=train, mtry=2, ntree=1000, xtest=test[,1:9],  
 ytest=test[,10], importance = TRUE)  
  
#view results  
rf.fit

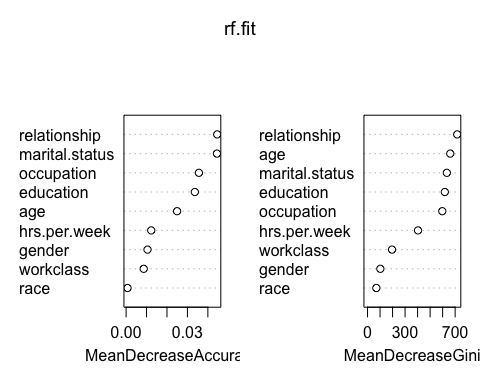
##   
## Call:  
## randomForest(formula = class ~ ., data = train, mtry = 2, ntree = 1000, xtest = test[, 1:9], ytest = test[, 10], importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 1000  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 16.67%  
## Confusion matrix:  
## <=50K >50K class.error  
## <=50K 12575 1068 0.0782819  
## >50K 1987 2697 0.4242101  
## Test set error rate: 16.29%  
## Confusion matrix:  
## <=50K >50K class.error  
## <=50K 6286 567 0.08273749  
## >50K 926 1385 0.40069234

1. Look at variable importances

importance(rf.fit)

## <=50K >50K MeanDecreaseAccuracy MeanDecreaseGini  
## age 5.468809 114.309099 119.95939 661.20337  
## workclass 58.005314 28.751758 67.84548 195.77359  
## education 58.550284 74.580578 94.75745 617.53401  
## marital.status 57.240472 32.556218 57.49031 633.53646  
## occupation 67.446757 65.159183 101.16920 597.39196  
## relationship 32.971742 53.688537 58.63921 716.58305  
## race 9.347648 8.124646 14.40828 70.25577  
## gender 35.985941 10.647283 56.19570 102.30177  
## hrs.per.week 15.791909 69.671489 74.94692 403.49590

varImpPlot(rf.fit, scale=FALSE)



1. Are the variable importances different for the two different methods?

Note: in reality, randomForest variable importances are biased esp. towards correlated variables and variables with many categories. Should use cforest() in 'party' package for purpose of variable importance.

Do you think we would have correlated variables?

1. Optional: tune mtry parameter using tuneRF() function. Google to help if needed.

?tuneRF

## End