ml\_course\_project\_v3.R

nkravitz

Fri Dec 18 11:53:47 2015

#start

cat("Init");

## Init

setwd("Q:\\dev\\coursera\\ml");  
  
#libraries  
library(caret)

## Loading required package: lattice  
## Loading required package: ggplot2

library(kernlab)  
library(ggplot2)  
library(lattice)  
library(corrplot)  
library(tree)

## Warning: package 'tree' was built under R version 3.2.3

library(rattle)

## Warning: package 'rattle' was built under R version 3.2.3

## Rattle: A free graphical interface for data mining with R.  
## Version 4.0.5 Copyright (c) 2006-2015 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(rpart)  
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.2.3

require(randomForest)

## Loading required package: randomForest  
## randomForest 4.6-12  
## Type rfNews() to see new features/changes/bug fixes.

#tuning and settings  
OUTPUT\_EXPLORATORY\_PLOTS <- FALSE  
TEST\_ROWS <- 1000 #if > -1, test code with a subset of observations  
TRAINING\_PCT <- 0.8  
CORREL\_THRESHHOLD <- 0.95  
  
#seed  
set.seed(33833)  
  
#load data  
pml\_training = read.csv("pml-training.csv", header = TRUE, stringsAsFactors = FALSE, na.strings=c("", "NA", "NULL"))  
pml\_testing = read.csv("pml-testing.csv", header = TRUE, stringsAsFactors = FALSE, na.strings=c("", "NA", "NULL"))  
if (TEST\_ROWS > -1) {  
 pml\_training <- pml\_training[sample(nrow(pml\_training), TEST\_ROWS), ]  
}  
   
dim(pml\_training)

## [1] 1000 160

#summary(pml\_training)  
#str(pml\_training)  
  
#tidying of training set  
pml\_training\_tidy <- pml\_training  
#change classe into factor  
pml\_training\_tidy$classe = as.factor(pml\_training\_tidy$classe)  
#Some sparse fields are incorrectly interpreted as characters, turn them explicitly into numerics  
transform\_fields <- grep("^(kurtosis|max|min|skewness)", colnames(pml\_training\_tidy))  
pml\_training\_tidy[transform\_fields] <- lapply(pml\_training\_tidy[transform\_fields], as.numeric)

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

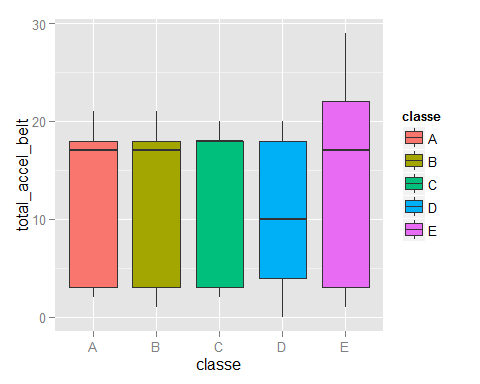
## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

## Warning in lapply(pml\_training\_tidy[transform\_fields], as.numeric): NAs  
## introduced by coercion

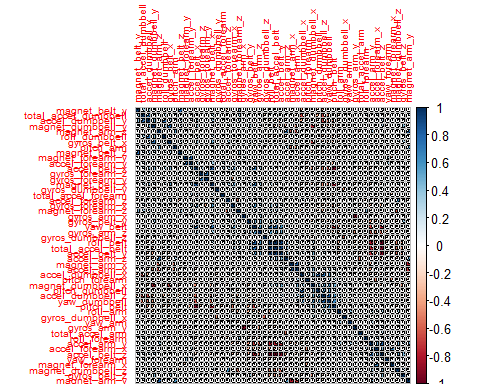
#remove mostly NA fields  
pml\_training\_tidy <- pml\_training[ , colSums(is.na(pml\_training)) == 0]  
#remove non-data fields  
nondata\_fields = c('X', 'user\_name', 'raw\_timestamp\_part\_1', 'raw\_timestamp\_part\_2', 'cvtd\_timestamp', 'new\_window', 'num\_window')  
pml\_training\_ind <- pml\_training\_tidy[, -which(names(pml\_training\_tidy) %in% nondata\_fields)]  
  
#exploratory analysis - output boxplots for all variables  
if (OUTPUT\_EXPLORATORY\_PLOTS) {  
 for (ind in 1:(ncol(pml\_training\_ind) - 1)) {  
 png(paste0("img/box\_", colnames(pml\_training\_ind)[ind], ".png"))  
 p <- ggplot(  
 pml\_training\_ind,   
 aes\_string(x = "classe", y = colnames(pml\_training\_ind)[ind], fill = "classe")  
 ) + geom\_boxplot() #+ ylim(-5, 5)  
 print(p)  
 dev.off()   
 }  
}  
  
#yaw\_belt has good explanatory power. Intuitively, We expect this to show up in analysis  
p <- ggplot(  
 pml\_training\_ind,   
 aes(x = classe, y = total\_accel\_belt, fill = classe)  
) + geom\_boxplot() #+ ylim(-5, 5)  
print(p)



#the following variables are also good indicators.   
primary\_indep\_vars <- c(  
 "accel\_arm\_x",   
 "accel\_belt\_y",   
 "accel\_belt\_z",  
 "amplitude\_pitch\_belt",  
 "amplitude\_pitch\_forearm",  
 "avg\_roll\_belt",  
 "avg\_roll\_dumbbell",  
 "avg\_yaw\_belt",  
 "magnet\_arm\_x",  
 "max\_picth\_arm",  
 "max\_picth\_belt",  
 "max\_picth\_dumbbell",  
 "max\_roll\_belt",  
 "max\_roll\_dumbbell",  
 "min\_pitch\_belt",  
 "min\_pitch\_forearm",  
 "min\_roll\_belt",  
 "min\_roll\_forearm",  
 "roll\_belt",  
 "roll\_forearm",  
 "stddev\_pitch\_belt",  
 "stddev\_roll\_belt",  
 "stddev\_roll\_forearm",   
 "stddev\_yaw\_forearm",  
 "total\_accel\_belt",  
 "var\_pitch\_dumbbell",  
 "var\_roll\_belt",  
 "var\_roll\_forearm",  
 "var\_total\_accel\_belt",  
 "var\_yaw\_forearm"  
)  
primary\_indep\_vars

## [1] "accel\_arm\_x" "accel\_belt\_y"   
## [3] "accel\_belt\_z" "amplitude\_pitch\_belt"   
## [5] "amplitude\_pitch\_forearm" "avg\_roll\_belt"   
## [7] "avg\_roll\_dumbbell" "avg\_yaw\_belt"   
## [9] "magnet\_arm\_x" "max\_picth\_arm"   
## [11] "max\_picth\_belt" "max\_picth\_dumbbell"   
## [13] "max\_roll\_belt" "max\_roll\_dumbbell"   
## [15] "min\_pitch\_belt" "min\_pitch\_forearm"   
## [17] "min\_roll\_belt" "min\_roll\_forearm"   
## [19] "roll\_belt" "roll\_forearm"   
## [21] "stddev\_pitch\_belt" "stddev\_roll\_belt"   
## [23] "stddev\_roll\_forearm" "stddev\_yaw\_forearm"   
## [25] "total\_accel\_belt" "var\_pitch\_dumbbell"   
## [27] "var\_roll\_belt" "var\_roll\_forearm"   
## [29] "var\_total\_accel\_belt" "var\_yaw\_forearm"

#study correlation of covariates  
corrMatrix <- cor(na.omit(pml\_training\_ind[sapply(pml\_training\_ind, is.numeric)]))  
corrplot(corrMatrix, order="AOE", method="pie", tl.cex=0.6)



#and remove highly correlated  
remove\_highcorrelation <- findCorrelation(corrMatrix, CORREL\_THRESHHOLD, verbose=TRUE)

## Compare row 10 and column 1 with corr 0.991   
## Means: 0.272 vs 0.172 so flagging column 10   
## Compare row 1 and column 4 with corr 0.975   
## Means: 0.252 vs 0.168 so flagging column 1   
## Compare row 8 and column 2 with corr 0.956   
## Means: 0.253 vs 0.165 so flagging column 8   
## All correlations <= 0.95

pml\_training\_independent = pml\_training\_ind[,-remove\_highcorrelation]  
  
#align training/testing sets  
pml\_testing\_independent <- pml\_testing[, which(names(pml\_testing) %in% names(pml\_training\_independent))]  
  
#how many fields do we have left?  
dim(pml\_training\_independent)

## [1] 1000 50

dim(pml\_testing\_independent)

## [1] 20 49

#split original training set into test / training for CV  
inTrain <- createDataPartition(y=pml\_training\_independent$classe, p=TRAINING\_PCT, list=FALSE)  
training <- pml\_training\_independent[inTrain,]  
testing <- pml\_training\_independent[-inTrain,]  
  
#use several ML methods   
ml\_methods <- c("rf", "gbm", "lda")  
ensemble\_pred <- data.frame(NA, NA, NA, classe=testing$classe)  
pml\_testing\_pred <- data.frame(rep(NA, nrow(pml\_testing\_independent)), NA, NA, NA)  
colnames(ensemble\_pred) = c(ml\_methods, "classe")  
colnames(pml\_testing\_pred) = colnames(ensemble\_pred)  
  
#for each method  
for (ml\_method in ml\_methods) {  
 cat ("Running ML Algorithm", ml\_method, ":\n")  
   
 #train   
 fitML <- train (  
 classe ~ .,   
 data=training,   
 method=ml\_method,  
 metric = "Accuracy",  
 #preProcess=c("pca"),   
 trControl = trainControl(method = "cv"),  
 verbose = FALSE  
 )  
   
 #output variable importance - for some reason this doesn't work in lda  
 if (ml\_method != "lda") {  
 p <- plot(varImp(fitML, scale = FALSE), top=10)  
 print(p)  
 }  
   
 #predict  
 pred <- predict(fitML, testing)  
 ensemble\_pred[ml\_method] <- pred  
  
 #accuracy  
 cm <- confusionMatrix(pred, testing$classe)  
 cat ("Accuracy:", cm$overall[1], "\n")  
  
 #apply model to final testing set  
 pred <- predict(fitML, pml\_testing\_independent)  
 pml\_testing\_pred[ml\_method] <- pred  
}

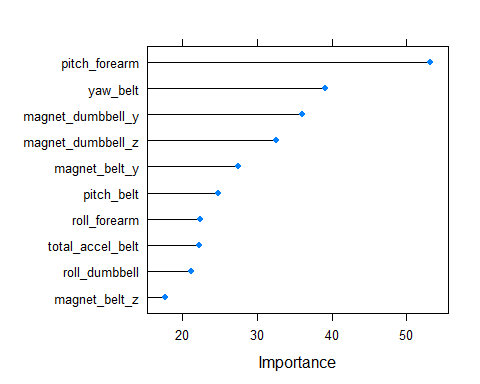
## Running ML Algorithm rf :

## Accuracy: 0.8984772   
## Running ML Algorithm gbm :

## Loading required package: gbm

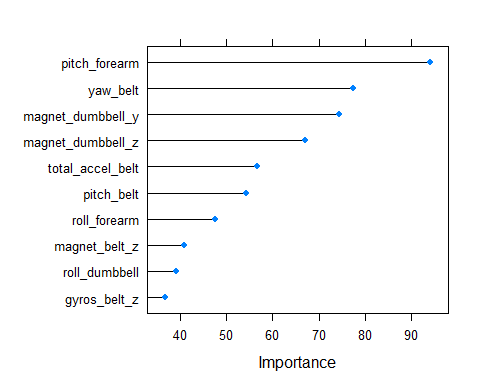
## Warning: package 'gbm' was built under R version 3.2.3

## Loading required package: survival  
##   
## Attaching package: 'survival'  
##   
## The following object is masked from 'package:caret':  
##   
## cluster  
##   
## Loading required package: splines  
## Loading required package: parallel  
## Loaded gbm 2.1.1  
## Loading required package: plyr



## Accuracy: 0.8274112   
## Running ML Algorithm lda :

## Loading required package: MASS



## Accuracy: 0.6548223

#stack (ensemble) models  
final\_fit <- train(classe ~., data=ensemble\_pred, method="rf")  
predFit <- predict(final\_fit, ensemble\_pred)  
cm <- confusionMatrix(predFit, ensemble\_pred$classe)  
cat ("Ensemble Accuracy:", cm$overall[1], "\n")

## Ensemble Accuracy: 0.9137056

# predict final training  
pml\_testing\_pred$classe <- predict(final\_fit, pml\_testing\_pred)  
pml\_testing\_pred

## rf gbm lda classe  
## 1 C C C C  
## 2 A A B A  
## 3 A B A A  
## 4 A A A A  
## 5 A A A A  
## 6 E E C E  
## 7 D D D D  
## 8 D B D D  
## 9 A A A A  
## 10 A A A A  
## 11 B B D B  
## 12 C C A C  
## 13 B B B B  
## 14 A A A A  
## 15 E E E E  
## 16 E A A E  
## 17 A A A A  
## 18 A B B B  
## 19 A A B A  
## 20 B B B B

```