HAR Project for Practical Machine Learning Class

Igor Volfman

Friday, July 17, 2015

Executive summary

The project aims to predict correct human body position during dumbbell excersize. For group of sensors are positioned on on arm, forearm, elbow and on a dumbbell. Each group collects 38 data streams of time during excersizes.

The training data will be used to traing classifiers using different models.
The model with highest accuracy will be selected by predicting cross validation set. Then the winnging model will be applied to predict outcome of 20 test cases.

The project will divided into followinf parts: 1. Loading training data 2. Data cleaing and exploration a. Removing irrelevant features. b. Cleaning data based on number of complete cases c. Inspection variations 3. Dividing training data into training and cross-validation sets with 80/20 ratio. 4. Run different models on the training sets a. GBM, LDA, CART, Random forest and SVM b. Predict outcome on CV set c. Aggeregate prediction models using random forest 5. Load test data set 6. Predict outcome

Loading libraries and setting appropriate folders

```
library(knitr)
library(ggplot2)
library(stats)
library(plyr)
library(reshape2)
library(caret)
```

```
## Loading required package: lattice
```

```
library(e1071)
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
setwd( file.path("~", "Coursera Data Science Track",
                 "Class8-PracitcalMachineLearning", "Project" ) )
```

1.Download training and test data sets.

```
# download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
#               "pml-training.csv")
# download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
#               "pml-testing.csv")
```

Loading files and removing empty strings and “#DIV/0!”

```
pml.testing <- read.csv( "pml-testing.csv", na.strings=c("",".","#DIV/0!", "NA") )
pml.training <- read.csv( "pml-training.csv", na.strings=c("",".","#DIV/0!", "NA") )
```

2. Cleaning and exploring data.

Outcome classes are evenly distributed

```
count(pml.training, 'classe')
```

```
##   classe freq
## 1      A 5580
## 2      B 3797
## 3      C 3422
## 4      D 3216
## 5      E 3607
```

There are 38 measurements in each group of sensors attached to person’s belt, arm forearmp and dumbbell.

```
groups <- c("belt", "arm", "dumbbell", "forearm")
col.group <- list()

cnames <- colnames(pml.training)

for (i in 1:length( groups )) {
        col.group[[i]] <- grep( paste("_",  groups[i], "_{0,1}", sep =""), cnames)

        writeLines( paste( "Sensor group: ", groups[i] ) )
        print ("----------------------")
        print ( cnames[ col.group[[i]] ] )
        print ("----------------------")
}
```

```
## Sensor group:  belt
## [1] "----------------------"
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
```

```
##  [4] "total_accel_belt"     "kurtosis_roll_belt"   "kurtosis_picth_belt"
##  [7] "kurtosis_yaw_belt"    "skewness_roll_belt"   "skewness_roll_belt.1"
## [10] "skewness_yaw_belt"    "max_roll_belt"        "max_picth_belt"
## [13] "max_yaw_belt"         "min_roll_belt"        "min_pitch_belt"
## [16] "min_yaw_belt"         "amplitude_roll_belt"  "amplitude_pitch_belt"
## [19] "amplitude_yaw_belt"   "var_total_accel_belt" "avg_roll_belt"
## [22] "stddev_roll_belt"     "var_roll_belt"        "avg_pitch_belt"
## [25] "stddev_pitch_belt"    "var_pitch_belt"       "avg_yaw_belt"
## [28] "stddev_yaw_belt"      "var_yaw_belt"         "gyros_belt_x"
## [31] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [34] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [37] "magnet_belt_y"        "magnet_belt_z"
## [1] "---------------------"
## Sensor group:  arm
## [1] "---------------------"
##  [1] "roll_arm"            "pitch_arm"           "yaw_arm"
##  [4] "total_accel_arm"     "var_accel_arm"       "avg_roll_arm"
##  [7] "stddev_roll_arm"     "var_roll_arm"        "avg_pitch_arm"
## [10] "stddev_pitch_arm"    "var_pitch_arm"       "avg_yaw_arm"
## [13] "stddev_yaw_arm"      "var_yaw_arm"         "gyros_arm_x"
## [16] "gyros_arm_y"         "gyros_arm_z"         "accel_arm_x"
## [19] "accel_arm_y"         "accel_arm_z"         "magnet_arm_x"
## [22] "magnet_arm_y"        "magnet_arm_z"        "kurtosis_roll_arm"
## [25] "kurtosis_picth_arm"  "kurtosis_yaw_arm"    "skewness_roll_arm"
## [28] "skewness_pitch_arm"  "skewness_yaw_arm"    "max_roll_arm"
## [31] "max_picth_arm"       "max_yaw_arm"         "min_roll_arm"
## [34] "min_pitch_arm"       "min_yaw_arm"         "amplitude_roll_arm"
## [37] "amplitude_pitch_arm" "amplitude_yaw_arm"
## [1] "---------------------"
## Sensor group:  dumbbell
## [1] "---------------------"
##  [1] "roll_dumbbell"            "pitch_dumbbell"
##  [3] "yaw_dumbbell"             "kurtosis_roll_dumbbell"
##  [5] "kurtosis_picth_dumbbell"  "kurtosis_yaw_dumbbell"
##  [7] "skewness_roll_dumbbell"   "skewness_pitch_dumbbell"
##  [9] "skewness_yaw_dumbbell"    "max_roll_dumbbell"
## [11] "max_picth_dumbbell"       "max_yaw_dumbbell"
## [13] "min_roll_dumbbell"        "min_pitch_dumbbell"
## [15] "min_yaw_dumbbell"         "amplitude_roll_dumbbell"
## [17] "amplitude_pitch_dumbbell" "amplitude_yaw_dumbbell"
## [19] "total_accel_dumbbell"     "var_accel_dumbbell"
## [21] "avg_roll_dumbbell"        "stddev_roll_dumbbell"
## [23] "var_roll_dumbbell"        "avg_pitch_dumbbell"
## [25] "stddev_pitch_dumbbell"    "var_pitch_dumbbell"
## [27] "avg_yaw_dumbbell"         "stddev_yaw_dumbbell"
## [29] "var_yaw_dumbbell"         "gyros_dumbbell_x"
## [31] "gyros_dumbbell_y"         "gyros_dumbbell_z"
## [33] "accel_dumbbell_x"         "accel_dumbbell_y"
## [35] "accel_dumbbell_z"         "magnet_dumbbell_x"
## [37] "magnet_dumbbell_y"        "magnet_dumbbell_z"
```

```
## [1] "---------------------"
## Sensor group:  forearm
## [1] "---------------------"
##  [1] "roll_forearm"            "pitch_forearm"
##  [3] "yaw_forearm"             "kurtosis_roll_forearm"
##  [5] "kurtosis_picth_forearm"  "kurtosis_yaw_forearm"
##  [7] "skewness_roll_forearm"   "skewness_pitch_forearm"
##  [9] "skewness_yaw_forearm"    "max_roll_forearm"
## [11] "max_picth_forearm"       "max_yaw_forearm"
## [13] "min_roll_forearm"        "min_pitch_forearm"
## [15] "min_yaw_forearm"         "amplitude_roll_forearm"
## [17] "amplitude_pitch_forearm" "amplitude_yaw_forearm"
## [19] "total_accel_forearm"     "var_accel_forearm"
## [21] "avg_roll_forearm"        "stddev_roll_forearm"
## [23] "var_roll_forearm"        "avg_pitch_forearm"
## [25] "stddev_pitch_forearm"    "var_pitch_forearm"
## [27] "avg_yaw_forearm"         "stddev_yaw_forearm"
## [29] "var_yaw_forearm"         "gyros_forearm_x"
## [31] "gyros_forearm_y"         "gyros_forearm_z"
## [33] "accel_forearm_x"         "accel_forearm_y"
## [35] "accel_forearm_z"         "magnet_forearm_x"
## [37] "magnet_forearm_y"        "magnet_forearm_z"
## [1] "---------------------"
```

```
# NUmber of variables in each sensor group
unlist( lapply( col.group, length) )
```

```
## [1] 38 38 38 38
```

All other vaiables are irrelevant to the predcition model.They are removed from training set.

```
valid.columns = list()
valid.columns[[1]] <- c( unlist(col.group), which(cnames == "classe") )
str( pml.training[-valid.columns[[1]]] )
```

```
## 'data.frame':    19622 obs. of  7 variables:
##  $ X                    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name            : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2
2 ...
##  $ raw_timestamp_part_1: int  1323084231 1323084231 1323084231 1323084232 1323084232 1
323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2: int  788290 808298 820366 120339 196328 304277 368296 440390
484323 484434 ...
##  $ cvtd_timestamp       : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9
9 ...
##  $ new_window           : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window           : int  11 11 11 12 12 12 12 12 12 12 ...
```

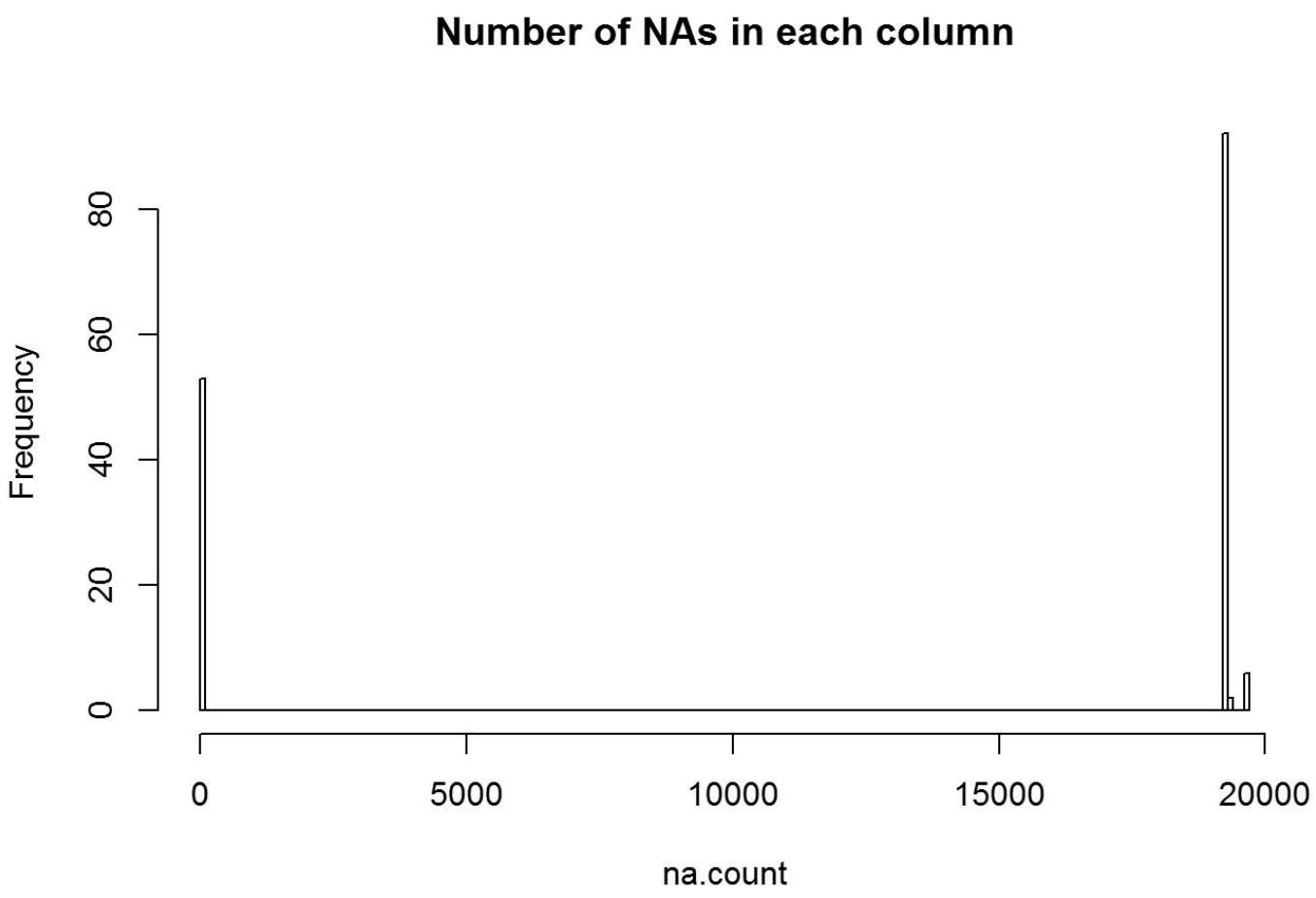
```
pml.training <- pml.training[, valid.columns[[1]]]
```

Most of variables carry very little information as one can see from the histogram. In fact all but 53 of them have more than 19,000 missing values. These variabes are discarded as well.

```
sum( complete.cases( pml.training ) )
```

```
## [1] 0
```

```
na.count <- sapply(pml.training, function(x) sum(length(which(is.na(x)))))
hist( na.count, breaks = 200, main = "Number of NAs in each column" )
```




```
sum( na.count < 15000 )
```

```
## [1] 53
```

```
valid.columns[[2]] <- which( na.count < 15000 )
sum( complete.cases( pml.training[, na.count < 15000 ] ) )
```

```
## [1] 19622
```

```
pml.training <- pml.training [, na.count < 15000 ]
```

3. Dividing training set into training and cross-validation groups.

One would be used for training different models and another (cross- validation set) would be used to select highest accuracy model.

```
trainIndex <- createDataPartition(pml.training$classe, p = 0.80,list=FALSE)
training <- pml.training[trainIndex,]
cv <- pml.training[-trainIndex,]
```

Random forest model.

```
# Seed is set anew before run of each model
# Training run-time is monitored

set.seed(54321)
rf.time <- system.time(
        mod.rf <- randomForest( training[,1:52], training$classe, ntree = 200)
        )
prediction.rf <- predict(mod.rf, newdata = cv)
accuracy.rf <- confusionMatrix( prediction.rf, cv$classe)$overall[1]
print(accuracy.rf)
```

```
##  Accuracy
## 0.9974509
```

```
print( rf.time[[1]])
```

```
## [1] 24.52
```

Support vector machine

It runs faster but is less accurate compared to randon forest

```
set.seed(54321)
svm.time <- system.time(
        mod.svm <- svm( classe ~ ., data = training)
        )
prediction.svm <- predict(mod.svm, newdata = cv)
accuracy.svm <- confusionMatrix( prediction.svm, cv$classe)$overall[1]
print(accuracy.svm)
```

```
##  Accuracy
## 0.9525873
```

```
print( svm.time[[1]])
```

```
## [1] 69.04
```

LDA model

```
set.seed(5431)
lda.time <- system.time(
        mod.lda <- train( classe ~., data = training, method = "lda")
        )
```

```
## Loading required package: MASS
```

```
prediction.lda <- predict(mod.lda, newdata = cv)
accuracy.lda <- confusionMatrix( prediction.lda, cv$classe)$overall[1]
print(accuracy.lda)
```

```
##  Accuracy
## 0.7099159
```

```
print( lda.time[[1]])
```

```
## [1] 14.3
```

5. Selecting best model

Comparing accuracy and run times of all three models, the random forest is most accurate and has comparable run time.

```
models <- data.frame( lda = c( accuracy.lda, lda.time[1] ),
                      rf = c( accuracy.rf, rf.time[1] ),
                      svm = c( accuracy.svm, svm.time[1]) )
rownames(models) <- c("Accuracy", "Run-time")
```

Out of sample error rate of random forest model is very low - 0.5%

```
error.rate <- 1 - accuracy.rf
print(error.rate)
```

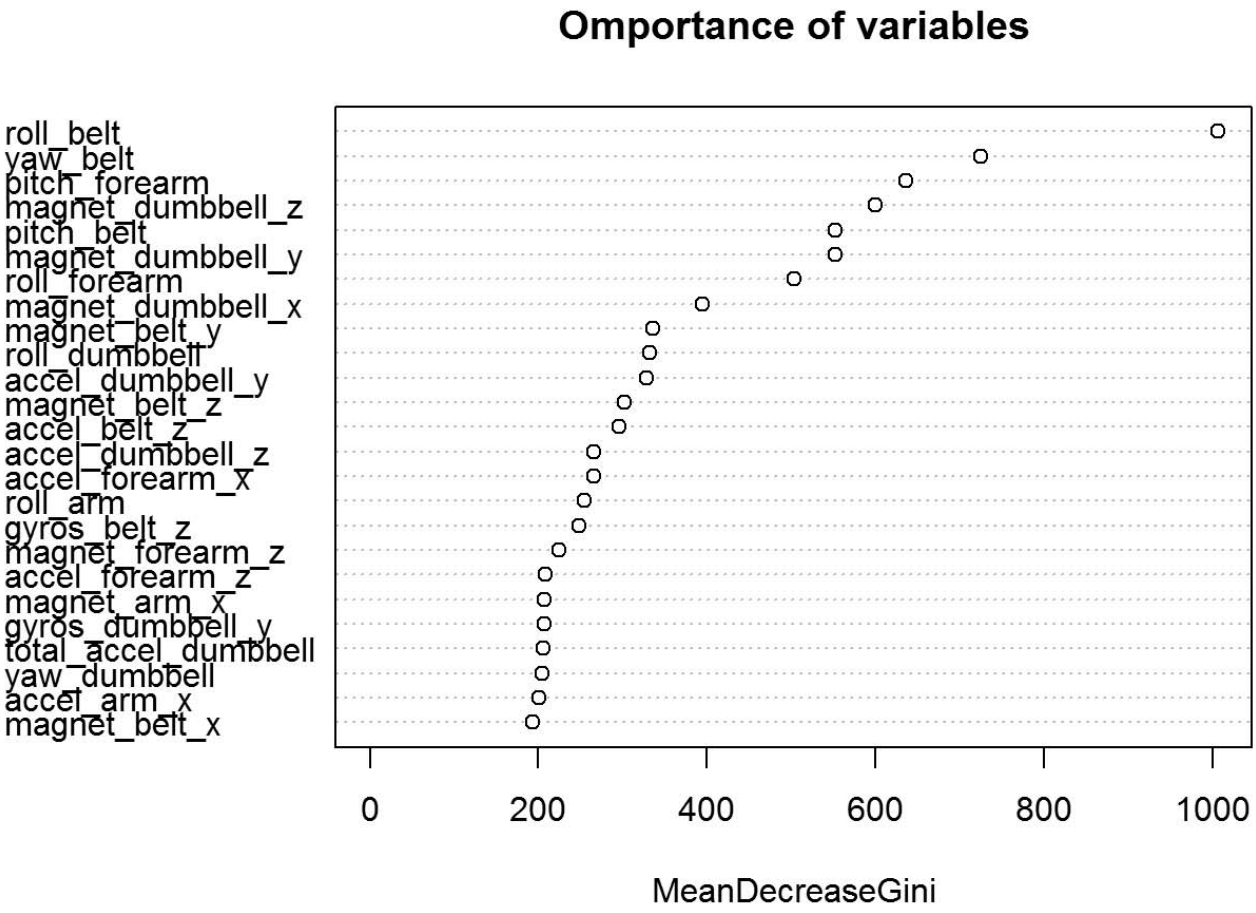
```
##   Accuracy
## 0.00254907
```

Confusion matrix

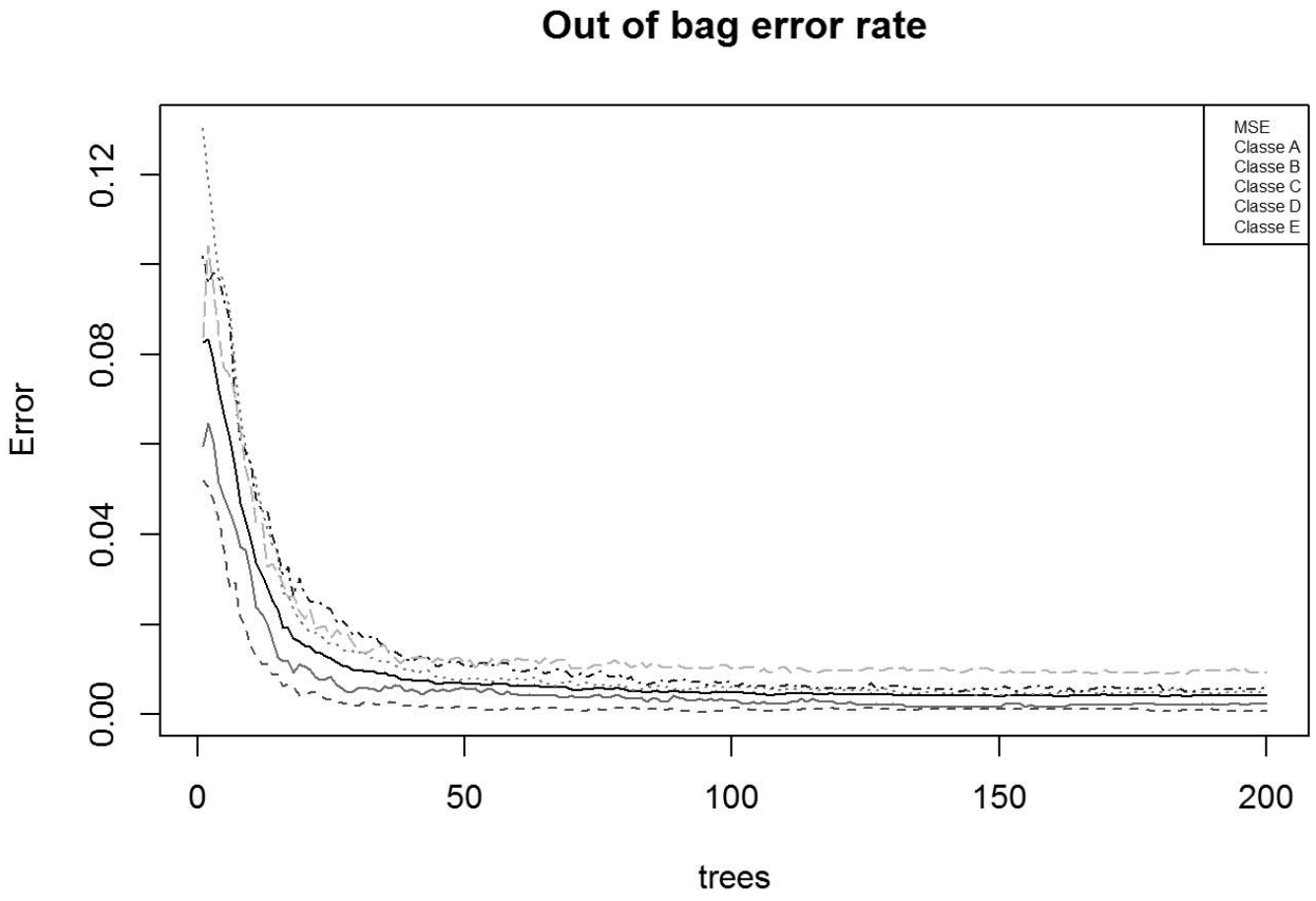
```
confusionMatrix( prediction.rf, cv$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1114    1    0    0    0
##          B    1  757    2    0    0
##          C    0    1  682    2    0
##          D    0    0    0  641    2
##          E    1    0    0    0  719
##
## Overall Statistics
##
##                Accuracy : 0.9975
##                  95% CI : (0.9953, 0.9988)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9968
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9974   0.9971   0.9969   0.9972
## Specificity            0.9996   0.9991   0.9991   0.9994   0.9997
## Pos Pred Value         0.9991   0.9961   0.9956   0.9969   0.9986
## Neg Pred Value         0.9993   0.9994   0.9994   0.9994   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1930   0.1738   0.1634   0.1833
## Detection Prevalence   0.2842   0.1937   0.1746   0.1639   0.1835
## Balanced Accuracy      0.9989   0.9982   0.9981   0.9981   0.9985
```

Ranking 25 most important vaiables of random forest model based on Gini index



It is interesting to see how fast random forest model converges



6. Predicting outcome of 20 test cases

```
# Reading test data set and removing columns not used for training
pml.testing <- read.csv( "pml-testing.csv", na.strings=c("",".","#DIV/0!", "NA") )
testing <- pml.testing [, valid.columns[[1]] ]
testing <- testing[, valid.columns[[2]] ]

colnames(testing)[53] <- "classe"
testing$classe <- factor( testing$classe )
answers = predict( mod.rf, newdata = testing)
print( answers )
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

Writing data files for prject submission