Practical Machine Learning Prediction Assignment

user12345678890

Saturday, March 21, 2015

Problem Specification

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

What you should submit

The goal of your project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

1. Your submission should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).

2. You should also apply your machine learning algorithm to the 20 test cases available in the test data above. Please submit your predictions in appropriate format to the programming assignment for automated grading. See the programming assignment for additional details.

Reproducibility

Due to security concerns with the exchange of R code, your code will not be run during the evaluation by your classmates. Please be sure that if they download the repo, they will be able to view the compiled HTML version of your analysis.

High Level Outcomes

Feature selection easily reduced 160 features to a tractable 53. Further reductions might produce more predictive power, but sufficient accuracy and a perfect prediction were made without further effort .

Model Training Several models were trained using diferent techiques, pre-processing, and degrees of cross validation.

Comparative Analysis In sample and out of sample errors were compared, as were the results of the given test set of 20 samples

Model Selection One model had outstanding and superior accuracy in sample (**99.3%**), and out of sample (**99.8%**).

This model was selected.

Perfect Score

Predictive Power The selected model produced a perfect prediction, on the first try, on the original test set of 20 samples when submitted

Prepare the workspace:

```
###############################################
##
## CLEAR WORKSPACE
##
###############################################

rm(list = ls(.GlobalEnv), envir = .GlobalEnv)
ls()
```

```
## character(0)
```

```
###############################################
##
## CONSTANTS
##
###############################################

doCalc <- FALSE
setwd("h:\\dev\\Johns_Hopkins\\Data_Science_Specialization\\8_Practical_Machine_Learning")

###############################################
##
## LOAD LIBRARIES
##
###############################################

library(knitr)      # PUBLISHING
library(caret)      # MACHINE LEARNING
library(rattle)     # NICE GRAPHICS
library(corrplot)   # CORRELATION GRAPHICS
library(parallel)   # PARALLEL PROCESSING
library(doParallel) # PARALLEL PROCESSING
```

Load the given data:

```
###############################################
##
## LOAD DATA
##
###############################################
```

```
setwd("h:\\dev\\Johns_Hopkins\\Data_Science_Specialization\\8_Practical_Machine_Learning")
training     <- read.csv("pml-training.csv", na.strings=c("", "NA", "NULL"))
testing      <- read.csv("pml-testing.csv" , na.strings=c("", "NA", "NULL"))
origTraining <- training
origTesting  <- testing
```

Feature Selection:

The data was examined for feature selection subject to a number of critera. The original width included 160 features, but 100 of those contained NAs which were excluded to reduce the dimensionality to a much more manageable 60 features. 7 of those 60 were uninformative, e.g. time stamps, further reducing the feature set to 53. An examination of variance did not flag any features as near-zero. A correlation study revealed 4 of the remaining 1,326 feature pairs are correlated by more than 90% but, due to time constraints and a small enough universe of features, they were left in. Further analysis would include items like skew and kurtosis which could be corrected for by various transformations, e.g. Box-Cox. A more parsimonious feature set, and centering and scaling, might produce more predictive power and a more tractable model, but time did not allow.

```
##############################################
##
## FEATURE SELECTION
##
##############################################

# RECORD ORIGINAL DIMENSIONS
origDim <- dim(training)
print(sprintf("There are %d columns in the training data.", origDim[2]))
```

```
## [1] "There are 160 columns in the training data."
```

```
# REMOVE SPARSE DATA
training <- training[ , colSums(is.na(training)) == 0]
newDim <- dim(training)
print(sprintf("There are %d columns after removing columns with NAs.", newDim [2]))
```

```
## [1] "There are 60 columns after removing columns with NAs."
```

```
# REMOVE UNINFORMATIVE COLUMNS
training <- training[, -which(names(training) %in% c('X', 'user_name', 'raw_timestamp_part_1',
  'raw_timestamp_part_2', 'cvtd_timestamp', 'new_window', 'num_window'))]
newDim <- dim(training)
print(sprintf("There are %d columns after removing uninformative columns.", newDim [2]))
```

```
## [1] "There are 53 columns after removing uninformative columns."
```

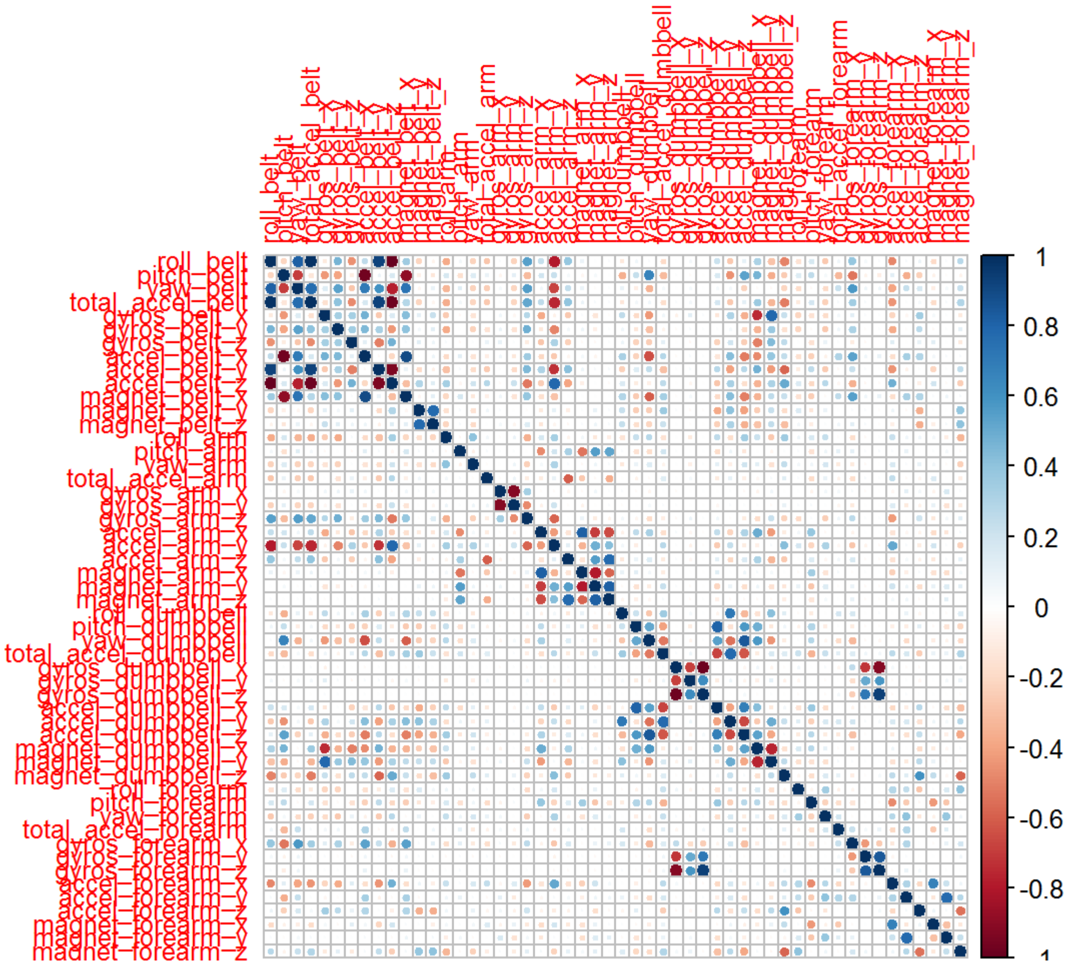
```
# CHECK FOR NEAR ZERO VARIANCE
nzv <- nearZeroVar(training,saveMetrics=TRUE)
print(sprintf("%d near zero values found.", sum(nzv$nzv)))
```

```
## [1] "0 near zero values found."
```

```
# CORRELATION
nums <- sapply(training, is.numeric)
corTraining <- cor(training[, nums])

corTraining2 <- corTraining
diag(corTraining2)<-0
print(sprintf("%d of the remaining %d feature pairs are highly correlated.",
              sum(corTraining2 >.9)/2, sum(abs(corTraining2) >0)/2))
```

```
## [1] "4 of the remaining 1326 feature pairs are highly correlated."
```

```
corrplot(corTraining, tl.cex = 0.8)
```



Data partitioning:

The original “training” data was partition arbitrarily with 70% in the training set, and 30% in the testing set. The original “testing” data was not used in the partition. Do not confuse the testing partition with the original given testing data. You can see in the table that the original data set was 19,622 samples, each having 160 features. The number of features were reduced to 53 and 13,737 (70%) of the samples were used for training. The remaining 30% of the data was used for out of sample testing.

```
###############################################
##
## PARTITION
##
###############################################

inTrain  <- createDataPartition(y=training$classe, p=0.7, list=FALSE)
training <- training[ inTrain,]
testing  <- training[ -inTrain,]
rbind("original dataset" = dim(origTraining ),"training set" = dim(training))
```

```
##                   [,1] [,2]
## original dataset 19622  160
## training set     13737   53
```

Model training:

Several models were trained using the new partitioned training data. Had there been more time, an exhaustive combination of models and parameters would have been tried. These models used cross validation as required by the problem specification. Parallel processing was used to speed up the calcuation and the resulting models were saved to disk for re-use. Random Forest without PCA was cross validated 20 times as well as the 5 times CV for the others.

```
###############################################
##
## TRAIN MODELS
##
###############################################

setwd("h:\\dev\\Johns_Hopkins\\Data_Science_Specialization\\8_Practical_Machine_Learning")

## NB WITH 5-Times cross validATION
if (doCalc){
    cl <- makeCluster(detectCores())
    registerDoParallel(cl)
    set.seed(12345)
    modelFitNB <- train(classe ~ ., data=training, method="nb",
        trControl = trainControl(method = "cv", number = 5),
        allowParallel = TRUE)
    stopCluster(cl)
    save(modelFitNB , file="modelFitNB.RData")
} else load("modelFitNB.RData")

## RANDOM FOREST WITH PCA & 5-Times cross validATION
if (doCalc){
    cl <- makeCluster(detectCores())
    registerDoParallel(cl)
    set.seed(12345)
    modelFit99PCA <- train(classe ~ ., preProcess="pca", data=training, method="rf",
        thresh = 0.99, trControl = trainControl(method = "cv", number = 5),
        prox = TRUE, importance = TRUE, allowParallel = TRUE)
    stopCluster(cl)
    save(modelFit99PCA , file="modelFit99PCA.RData")
} else load("modelFit99PCA.RData")

## RANDOM FOREST WITH & 5-Times cross validATION (NO PCA)
if (doCalc){
    cl <- makeCluster(detectCores())
    registerDoParallel(cl)
    set.seed(12345)
    modelFit <- train(classe ~ ., data=training, method="rf",
        thresh = 0.99, trControl = trainControl(method = "cv", number = 5),
        prox = TRUE, importance = TRUE, allowParallel = TRUE)
    stopCluster(cl)
    save(modelFit , file="modelFit.RData")
} else load("modelFit.RData")

## RANDOM FOREST WITH & 5-Times cross validATION (NO PCA)
if (doCalc){
  cl <- makeCluster(detectCores())
    registerDoParallel(cl)
    set.seed(12345)
    modelFitRF20Times <- train(classe ~ ., data=training, method="rf",
        thresh = 0.99, trControl = trainControl(method = "cv", number = 20),
        prox = TRUE, importance = TRUE, allowParallel = TRUE)
    stopCluster(cl)
    save(modelFitRF20Times , file="modelFitRF20Times.RData")
} else load("modelFitRF20Times.RData")
```

Comparative analysis, answers:

It is crucial to note that, regardless of how small the difference in accuracy might be for a given method vs. another, the predicted answers are different and so the most accurate method must be chosen, even if the computational cost is high. You can see in column 3, RF+PCA predicts “3” and RF predicts “2.” There is also a difference in column 6. The 5x and 20x cross validations produces the same answers in the original test data, though the computational cost was much higher for the extra CVs.

```
##############################################
##
## COMPARE ANSWERS
##
##############################################

set.seed(12345)
answersNB    <- predict(modelFitNB,      origTesting)
answers99PCA <- predict(modelFit99PCA,   origTesting)
answers99    <- predict(modelFit,        origTesting)
answers20F   <- predict(modelFitRF20Times,origTesting)

rbind(
  "NB     5xCV"  = answersNB,
  "RF+PCA 5xCV"  = answers99PCA,
  "RF     5xCV"  = answers99,
  "RF    20xCV"  = answers20F)
```

```
##               [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
## NB      5xCV    1    1    1    1    1    5    4    5    1     1     1     1
## RF+PCA 5xCV    2    1    3    1    1    2    4    2    1     1     2     3
## RF      5xCV    2    1    2    1    1    5    4    2    1     1     2     3
## RF    20xCV    2    1    2    1    1    5    4    2    1     1     2     3
##               [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
## NB      5xCV     2     1     5     1     1     2     1     2
## RF+PCA 5xCV     2     1     5     5     1     2     2     2
## RF      5xCV     2     1     5     5     1     2     2     2
## RF    20xCV     2     1     5     5     1     2     2     2
```

Comparative analysis, in sample and out of sample error:

The in sample and out of sample errors were examined for the various models. The accuracy of Random Forest without Principal Component Analysis (RF) is far better than the other two models. Increasing cross validations from 5x to 20x greatly increased both in and out of sample accuracy. In sample error was reduced to 0.7% and out of sample errror was reduced to 0.2%, which is exeedingly small! Note that it is unusual that the out of sample error is less than the in sample error but, because of the smaller data set, the out of sample error will have more variability which may explain this discrepancy. Examining the distribution of errors over different trials would be one way to examine this.

```
#############################################
##
## IN SAMPLE ERROR
##
#############################################

errISNB        <- 1-max(      modelFitNB$results$Accuracy)
errIS99PCA     <- 1-max(   modelFit99PCA$results$Accuracy)
errIS99        <- 1-max(        modelFit$results$Accuracy)
errIS20F       <- 1-max(modelFitRF20Times$results$Accuracy)

#############################################
##
## OUT OF SAMPLE ERROR
##
#############################################

answersTestNB    <- predict(modelFitNB      , testing[-match("classe", colnames(testing))])
answersTest99PCA <- predict(modelFit99PCA   , testing[-match("classe", colnames(testing))])
answersTest99    <- predict(modelFit        , testing[-match("classe", colnames(testing))])
answersTest20F   <- predict(modelFitRF20Times, testing[-match("classe", colnames(testing))])

confusionNB      =  confusionMatrix(testing$classe, answersTestNB)
confusion99PCA   =  confusionMatrix(testing$classe, answersTest99PCA)
confusion99      =  confusionMatrix(testing$classe, answersTest99)
confusion20F     =  confusionMatrix(testing$classe, answersTest20F)

rbind(
  "NB     5xCV"  = c(errISNB   , 1-   confusionNB$overall[[1]]),
  "RF+PCA 5xCV"  = c(errIS99PCA, 1-confusion99PCA$overall[[1]]),
  "RF     5xCV"  = c(errIS99   , 1-   confusion99$overall[[1]]),
  "RF    20xCV"  = c(errIS20F  , 1-  confusion20F$overall[[1]]))
```

```
##                   [,1]        [,2]
## NB     5xCV 0.26221090 0.256813160
## RF+PCA 5xCV 0.04553410 0.019641542
## RF     5xCV 0.01445107 0.009084213
## RF    20xCV 0.00706022 0.002455193
```

Model selection:

The Random Forest model with no Principal Component Analysis and 20x cross validation was chosen because of its superior accuracy. This may come at the cost of overfitting. All the following applies to that model only.

Out of sample confusion Matrix:

```
# NOTE: IF PCA PRE-PROCESSING IS APPLIED TO THE TRAINING SET, IT IS ALSO AUTOMATICALLY
# APPLIED WHEN USING PREDICT, SO LONG AS THE MODEL IS SPECIFIED (NOT THE FINAL MODEL
confusionMatrix(testing$classe, answersTest20F)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1166    0    0    0    0
##          B    5  798    0    0    0
##          C    0    1  729    0    0
##          D    0    0    3  652    0
##          E    0    0    0    1  718
##
## Overall Statistics
##
##                Accuracy : 0.9975
##                  95% CI : (0.9955, 0.9988)
##     No Information Rate : 0.2875
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9969
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9957   0.9987   0.9959   0.9985   1.0000
## Specificity            1.0000   0.9985   0.9997   0.9991   0.9997
## Pos Pred Value         1.0000   0.9938   0.9986   0.9954   0.9986
## Neg Pred Value         0.9983   0.9997   0.9991   0.9997   1.0000
## Prevalence             0.2875   0.1962   0.1797   0.1603   0.1763
## Detection Rate         0.2863   0.1959   0.1790   0.1601   0.1763
## Detection Prevalence   0.2863   0.1972   0.1792   0.1608   0.1765
## Balanced Accuracy      0.9979   0.9986   0.9978   0.9988   0.9999
```

In sample model analysis:

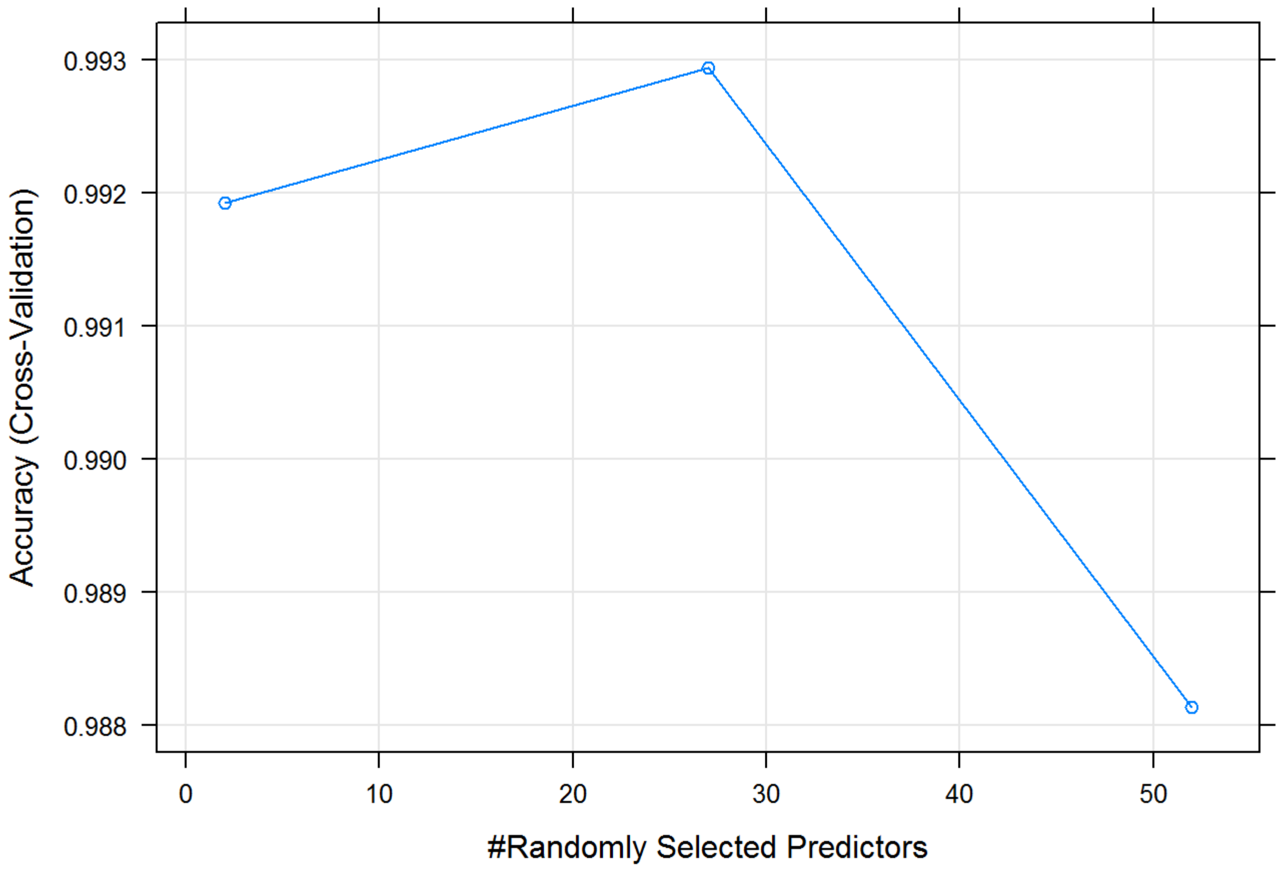
```
#############################################
##
## TABLES & CHARTS
##
#############################################

# PRINT THE MODEL
print(modelFitRF20Times)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (20 fold)
##
## Summary of sample sizes: 13049, 13050, 13049, 13050, 13050, 13050, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9919204  0.9897787  0.003848821  0.004869490
##   27    0.9929398  0.9910689  0.003872249  0.004898344
##   52    0.9881349  0.9849897  0.004800483  0.006074521
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

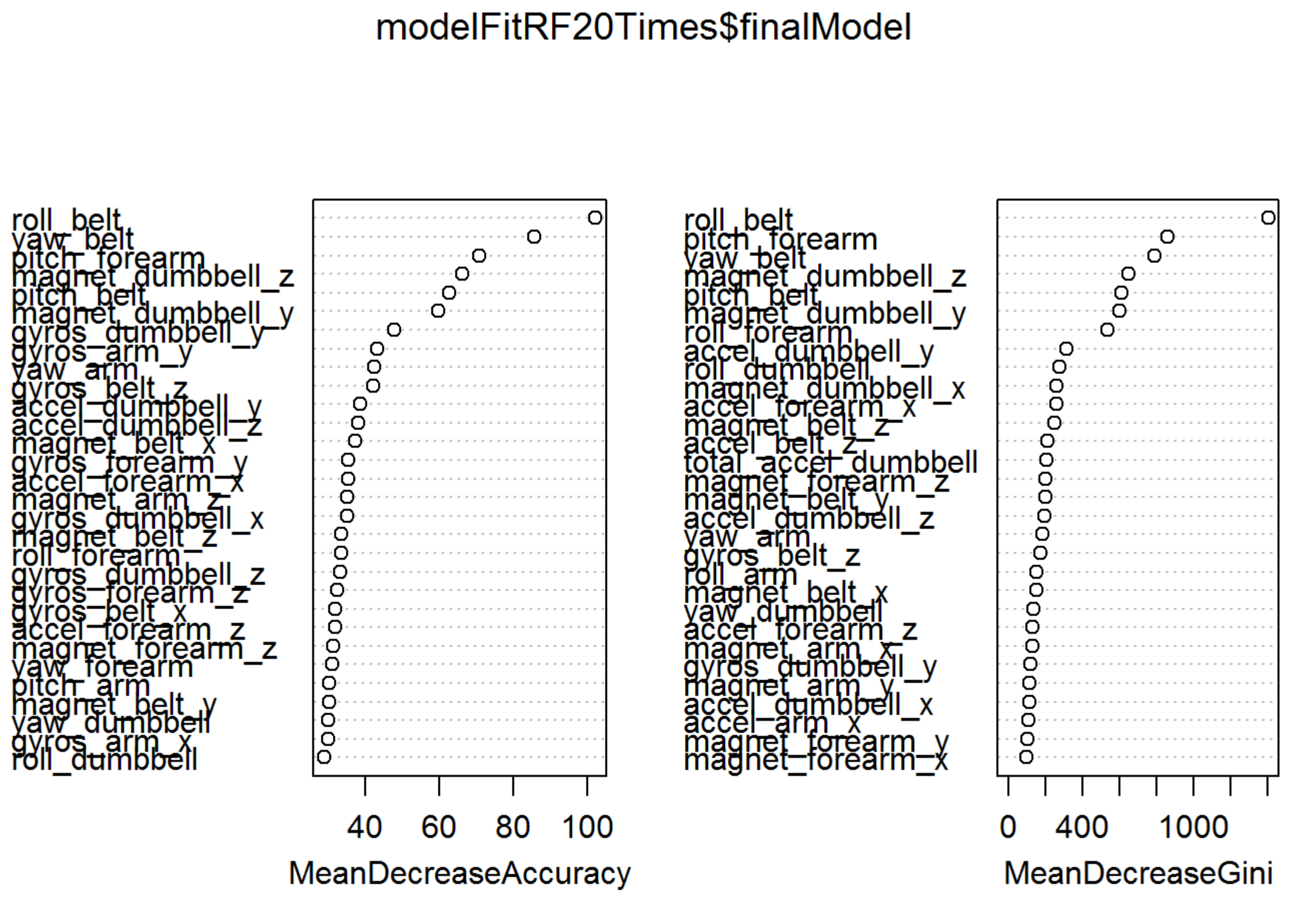
```
print(modelFitRF20Times$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE,      proximity = TRUE, t
hresh = 0.99, allowParallel = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.67%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    3    1    0    0 0.001024066
## B   19 2632    6    1    0 0.009781791
## C     0   14 2371   11    0 0.010434057
## D     0    1   23 2226    2 0.011545293
## E     0    0    4    7 2514 0.004356436
```

```
# ERROR VS NUM TREES
plot(modelFitRF20Times, log="y", tl.cex = 0.8)
```



```
# VARIABLE IMPORTANCE
varImpPlot(modelFitRF20Times$finalModel)
```



Build submission files:

```
#######################
##
## ASSIGNMENT SUBMISSION FILES
##
#######################

setwd("h:\\dev\\Johns_Hopkins\\Data_Science_Specialization\\8_Practical_Machine_Learning")

n = length(answers20F)
for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(answers20F[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
}
```

Perfect Score

The selected model produced a **perfect prediction**, on the first try, on the original test set of 20 samples when submitted