Human Activity Recognition | Adam J. Christopherson

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, we 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). We will use this data to predict the class of activity.

Reading and Cleaning Data

We are provided with a training data set, and a set of 20 observations that we are to predict. We will read in the data, and

```r
library(caret)

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

Let us further clean the data by removing variables that are mostly missing, removing the non-predictive variables and then split the training into a true training set, and another for cross validation.

```r
NAs <- sapply(training, function(x) mean(is.na(x))) > .9
training <- training[, NAs==FALSE]

trainingTrain <- training[,-(1:6)]
str(trainingTrain)

set.seed(12345)
inTrain <- createDataPartition(y=trainingTrain$classe, p=0.75, list=FALSE)
mytraining <- trainingTrain[inTrain,]
mytesting <- trainingTrain[-inTrain,]
```

Building the Model

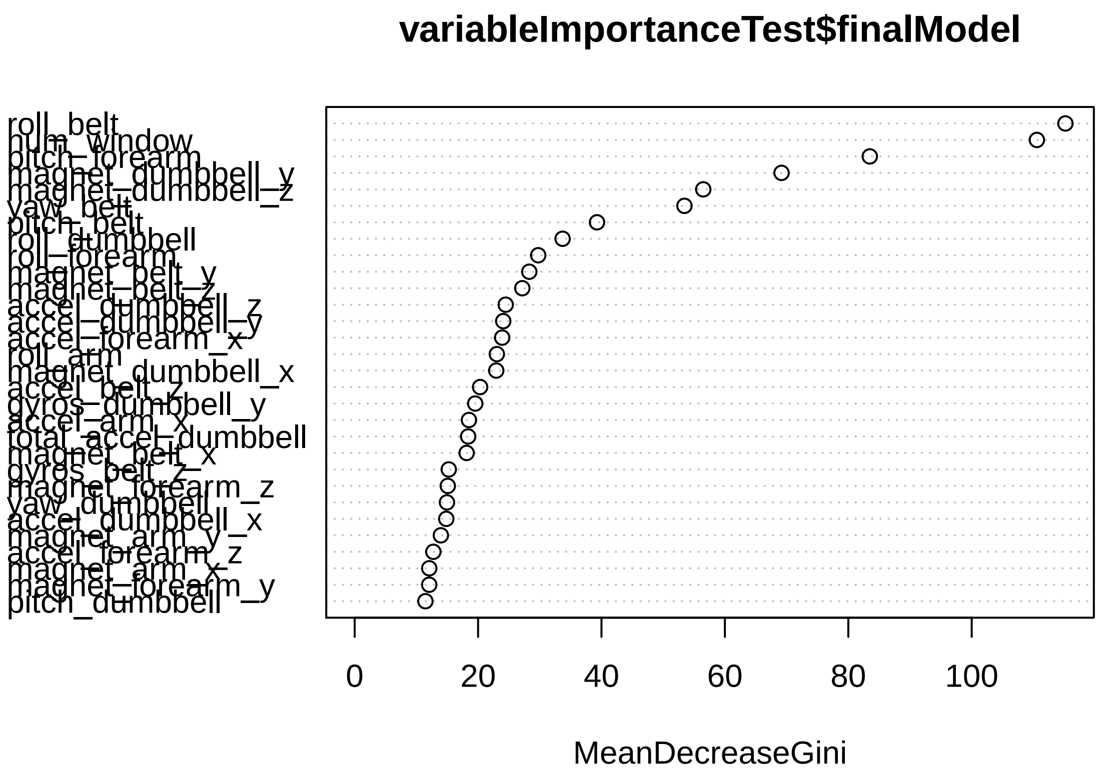
We will build a random forest to predict the class of exercise (variable classe). The data set is large, with many variables, so let us first take a smaller set of the training data to build a model and estimate which variables are most important.

```r
inTrain2 <- createDataPartition(y=mytraining$classe, p=0.1, list=FALSE)
mytrainsmall <- mytraining[inTrain2,]

variableImportanceTest <- train(classe~., method="rf", data=mytrainsmall, prox=TRUE)
```

We can plot the variable importances

```
varImpPlot(variableImportanceTest$finalModel)
```



From the plot we can see that using a subset of the variables would allow for an accurate prediction. We will choose 20 of the variables, since this will be less computationally expensive to grow the random forest. We will now build a model on the remaining training data.

```
VI <- data.frame(varImp(variableImportanceTest)$importance)
VI$vars <- row.names(VI)
row.names(VI) <- NULL

preds <- head(VI[order(VI$Overall, decreasing=TRUE),], n=20)
predNames <- preds$vars
predNames2 <- paste(predNames, collapse=" + ")
predNamesForm <- as.formula(paste"classe ~ ", predNames2)

mytrainlarger <- mytraining[-inTrain2,]
modFitForest <- train(predNamesForm, method="rf", data=mytrainlarger, prox=TRUE)
```

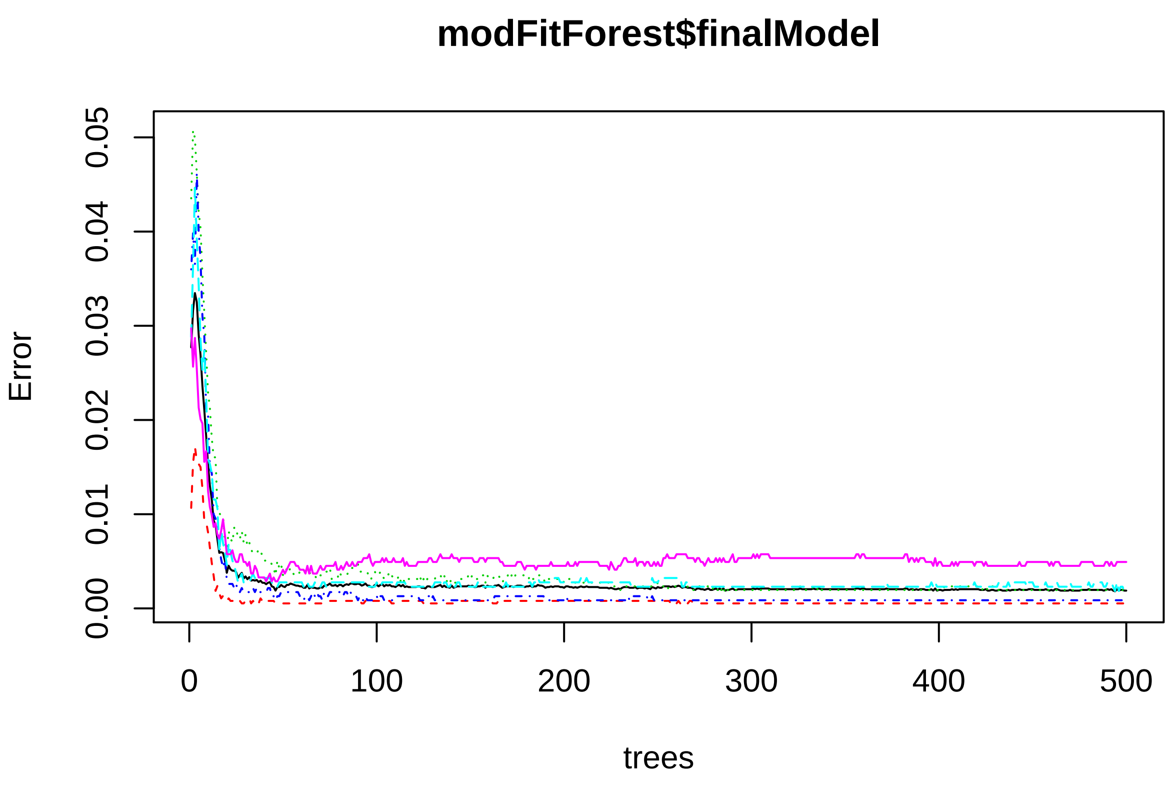
This produces the following random forest

```
modFitForest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 11
```

```
##
##         OOB estimate of  error rate: 0.19%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3764    1    0    0    1 0.0005310674
## B    3 2558    2    0    0 0.0019508389
## C    0    2 2308    0    0 0.0008658009
## D    0    0    4 2166    0 0.0018433180
## E    0    3    0    9 2423 0.0049281314
```

```
plot(modFitForest$finalModel)
```



Predictions

The out of box error is estimated at 0.19%. We can estimate this expressly through our cross validation data set

```
predForest <- predict(modFitForest, newdata=mytesting)
confusionMatrix(predForest, mytesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    3    0    0    0
##          B    0  943    2    0    0
##          C    0    3  852    4    0
##          D    0    0    1  800    3
##          E    0    0    0    0  898
##
## Overall Statistics
##
##                Accuracy : 0.9967
##                  95% CI : (0.9947, 0.9981)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9959
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9937   0.9965   0.9950   0.9967
## Specificity            0.9991   0.9995   0.9983   0.9990   1.0000
## Pos Pred Value         0.9979   0.9979   0.9919   0.9950   1.0000
## Neg Pred Value         1.0000   0.9985   0.9993   0.9990   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1923   0.1737   0.1631   0.1831
## Detection Prevalence   0.2851   0.1927   0.1752   0.1639   0.1831
## Balanced Accuracy      0.9996   0.9966   0.9974   0.9970   0.9983
```

giving an estimated accuracy of 99.7%. Finally, we obtain the following predictions for the given testing data set:

```
predict(modFitForest, newdata=testing)
```

```
##  [1] 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
```