Practical Machine Learning Course Project

Jeffrey A Vance

13 February 2016

Assignment

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 (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here

The test data are available here

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.(see Velloso et al. 2013).

Data Processing

We cache the data as many of the models can take a tremendous amount of time to run.

```
library(caret)
```

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

```
## Loading required package: ggplot2
```

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(rpart)
library(rpart.plot)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(doMC)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
library(iterators)
registerDoMC(cores=6)


set.seed(1971)

complete_trainingset <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
final_testset <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", na.stri
```

Feature Selection (stage 1)

We first remove off the timestamp and username parameters stored in columns 1-5.

The data also has a significant number of columns composed entirely of NAs. In order to be able to test, we will remove the rows in the training and testing set that are all NA in the testing set. (see "How to Delete Columns with NA in R")

Finally we remove any columns showing near zero variance.

The resulting structures are shown in Appendix 1.

```
working.training<-complete_trainingset[,-c(1:5)]
working.finaltest<-final_testset[,-c(1:5)]

working.training<-working.training[,colSums(is.na(working.finaltest)) != nrow(working.finaltest)]
working.finaltest<-working.finaltest[,colSums(is.na(working.finaltest)) != nrow(working.finaltest)]

indices.zerovariance <-nearZeroVar(working.training)
working.training<-working.training[,-indices.zerovariance]
working.finaltest<-working.finaltest[,-indices.zerovariance]
```

Training and Cross-Validation Subsetting of Training Set

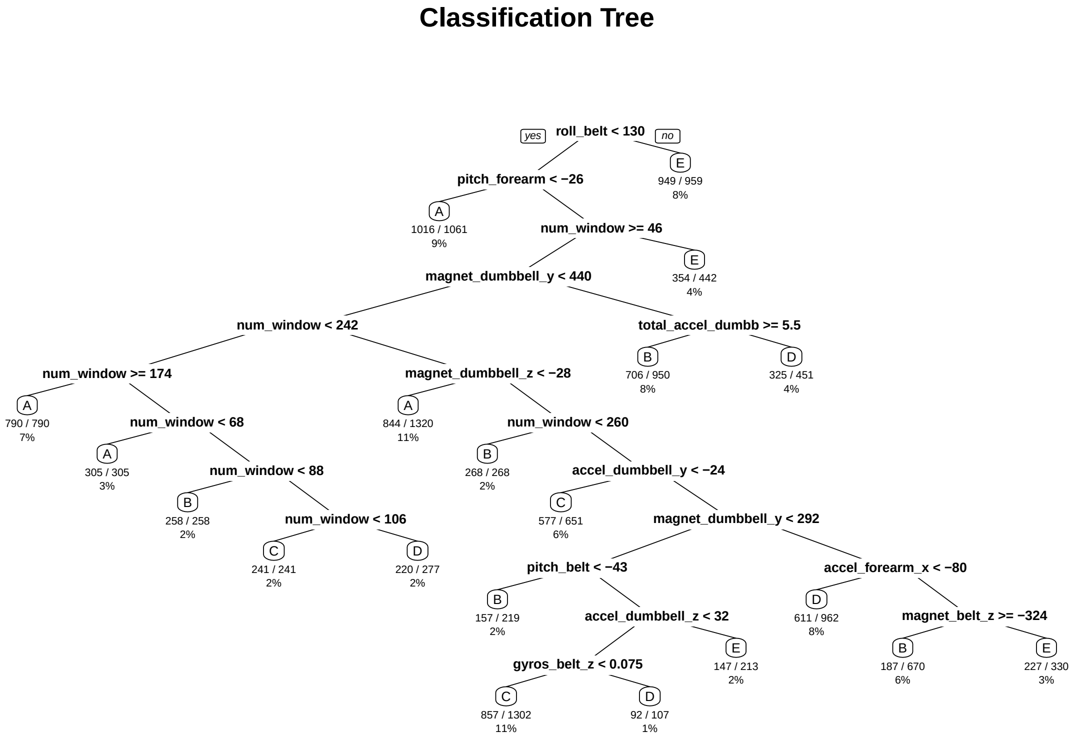
We further partition the training set to give us a working training & validation set.

```r
indices.training<-createDataPartition(working.training$classe, p=0.6, list=FALSE)
trainingset<-working.training[indices.training,]
crossvalset<-working.training[-indices.training,]
```

Analysis

We look first at a classification tree. This is intended to give us a sense of which variables are involved and identify significant issues (such as the initial 5 columns). It is not used in the evaluation.

```r
classmodel <- rpart(classe ~ ., data=trainingset, method="class")
classpredict <- predict(classmodel, crossvalset)
rpart.plot(classmodel, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```



We also define a misclassification function to determine out of sample error.

```r
misclassification = function(values, prediction) {
    sum(prediction!=values)/length(values)
}
```

We first look at a random forest (rf) model.

```r
rfmodel <- suppressMessages(train(classe~., data=trainingset, method="rf"))
rfpredict <- predict(rfmodel, crossvalset)
rfcm<-confusionMatrix(rfpredict, crossvalset$classe)
```

```
rfaccuracy <- rfcm$overall['Accuracy']
rferror <- misclassification(crossvalset$classe, rfpredict)
rfresults <- c("Random Forest", rfaccuracy, rferror)
```

We then look at a boosted trees (gba) model.

```
gbmmodel <- suppressMessages(train(classe~., data=trainingset, method="gbm"))
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2388
##      2        1.4560             nan     0.1000    0.1562
##      3        1.3562             nan     0.1000    0.1314
##      4        1.2734             nan     0.1000    0.1029
##      5        1.2069             nan     0.1000    0.1028
##      6        1.1429             nan     0.1000    0.0819
##      7        1.0916             nan     0.1000    0.0728
##      8        1.0470             nan     0.1000    0.0576
##      9        1.0108             nan     0.1000    0.0618
##     10        0.9728             nan     0.1000    0.0676
##     20        0.6984             nan     0.1000    0.0288
##     40        0.4587             nan     0.1000    0.0126
##     60        0.3339             nan     0.1000    0.0061
##     80        0.2525             nan     0.1000    0.0029
##    100        0.1974             nan     0.1000    0.0018
##    120        0.1547             nan     0.1000    0.0017
##    140        0.1247             nan     0.1000    0.0028
##    150        0.1102             nan     0.1000    0.0019
```

```
gbmpredict <- predict(gbmmodel, crossvalset)
gbmcm<-confusionMatrix(gbmpredict, crossvalset$classe)
gbmaccuracy <- gbmcm$overall['Accuracy']
gbmerror <- misclassification(crossvalset$classe, gbmpredict)
gbmresults <- c("GBM", gbmaccuracy, gbmerror)
```

We finally look at a linear discriminant analysis (lda) model.

```
ldamodel <- suppressMessages(train(classe~., data=trainingset, method="lda"))
ldapredict <- predict(ldamodel, crossvalset)
ldacm<-confusionMatrix(ldapredict, crossvalset$classe)
ldaaccuracy <- gbmcm$overall['Accuracy']
ldaerror <- misclassification(crossvalset$classe, ldapredict)
ldaresults <- c("LDA", ldaaccuracy, ldaerror)
```

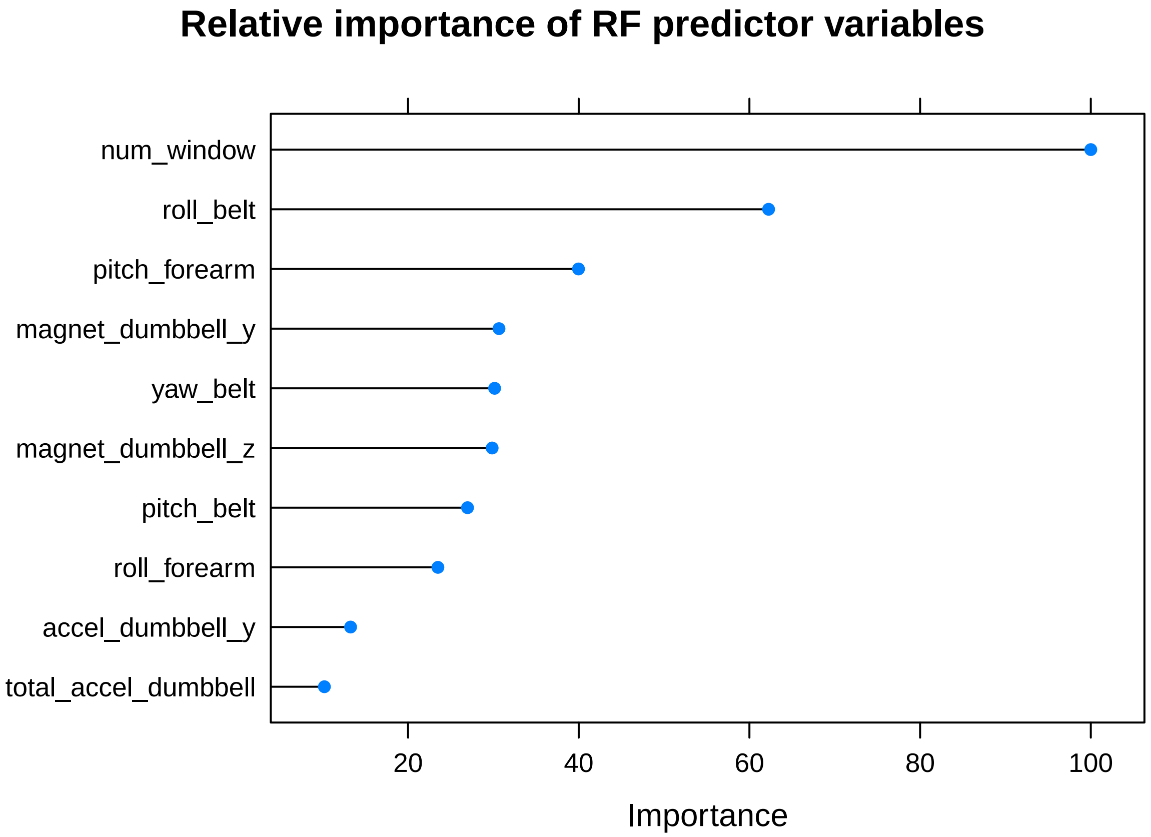
Evaluating the relative outcomes of the models:

```
df<-rbind.data.frame(rfresults, gbmresults, ldaresults)
colnames(df) <- c("Model Type", "Accuracy", "Error")
df
```

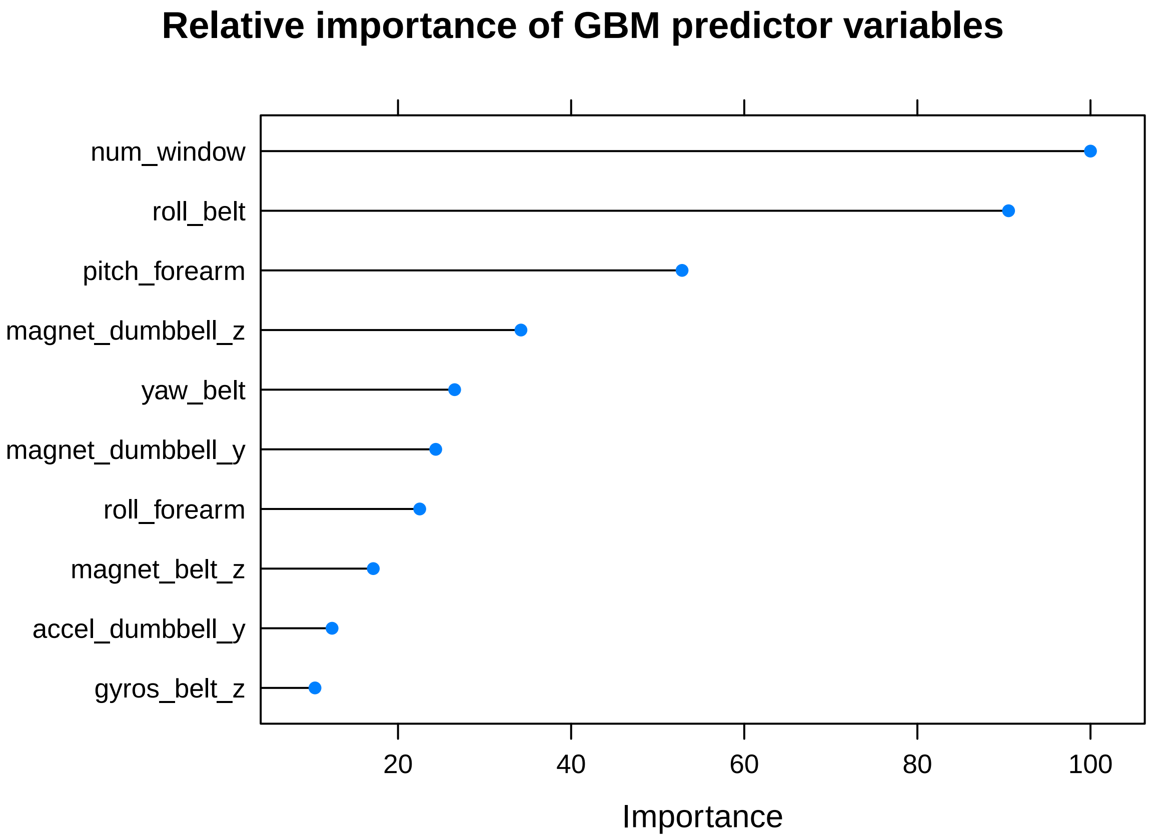
```
##      Model Type          Accuracy              Error
## 1 Random Forest   0.9974509304104 0.0025490695895998
## 2           GBM 0.984960489421361 0.0150395105786388
## 3           LDA 0.984960489421361  0.282946724445577
```

Looking at the variable importance of our models, we see consistency.

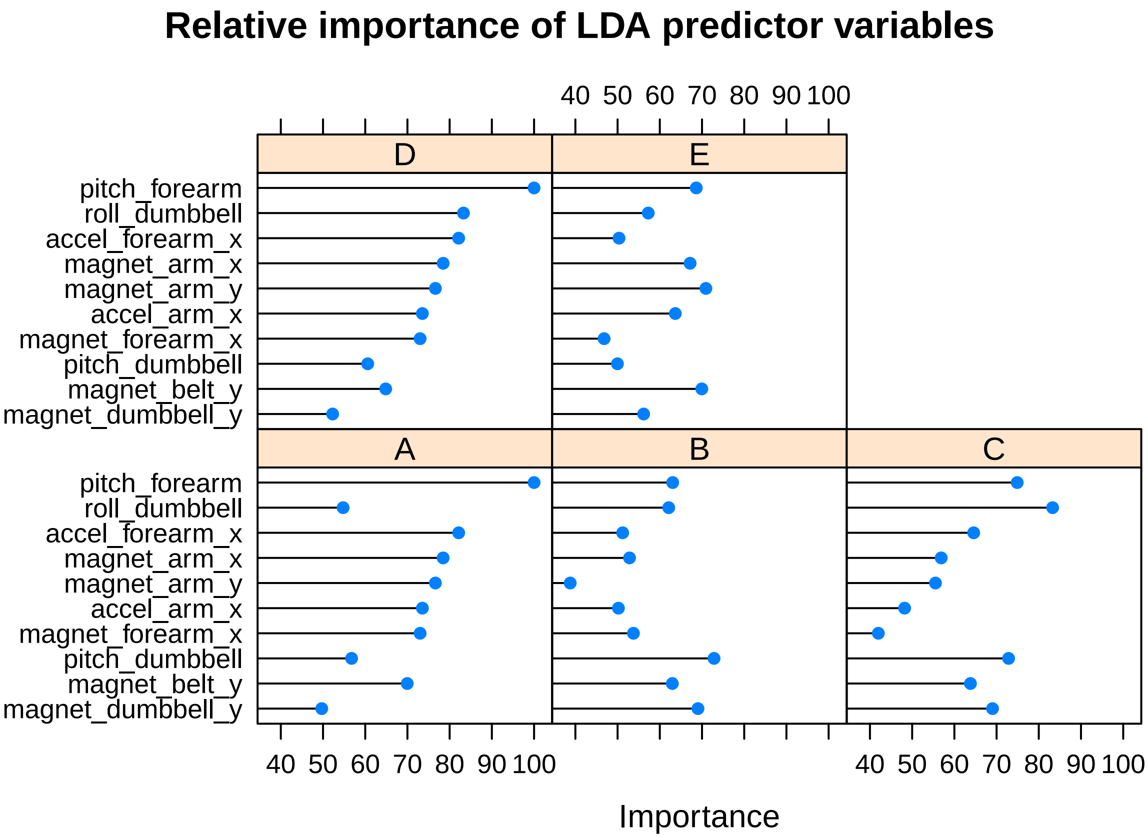
```
plot(varImp(rfmodel), main = "Relative importance of RF predictor variables", top=10)
```



```
plot(varImp(gbmmodel), main = "Relative importance of GBM predictor variables", top=10)
```



```r
plot(varImp(ldamodel), main = "Relative importance of LDA predictor variables", top=10)
```



Results

We then apply the same transformations to the final test set and generate predictions. As our best model, we use random forest as our predictor.

```
bestmodel<-rfmodel
testset<-working.finaltest

answers<-predict(bestmodel,testset)
answers
```

```
##  [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
```

Appendices

Appendix 1

```
colnames(working.finaltest)
```

```
##  [1] "num_window"           "roll_belt"            "pitch_belt"
##  [4] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
##  [7] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [10] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [13] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [16] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [19] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [22] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [25] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [28] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [31] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [34] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [37] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [40] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [43] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [46] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [49] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [52] "magnet_forearm_y"     "magnet_forearm_z"     "problem_id"
```

References

"How to Delete Columns with NA in R." http://stackoverflow/questions/15968494/how-to-delete-columns-with-na-in-r.

Velloso, E., A. Bulling, W. Gellersen, and H. Fuks. 2013. "Qualitative Activity Recognition of Weight Lifting Exercises." *Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmentation Human '13)*. ACM SIGCHI. http://groupware.les.inf.puc-rio.br/har#wle_paper_section#ixzz400Qta47M.