Practical Machine Learning Course Project

Anna Berman

April 18, 2017

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 and sourcing

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. My special thanks to the above mentioned authors for being so generous in allowing their data to be used for this kind of assignment.

Setting up the environment

First, we download the datasets and necessary analysis libraries.

```
knitr::opts_chunk$set(echo = TRUE)
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(corrplot)
library(rpart)
```

```r
library(rpart.plot)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
set.seed(125)

trainData <- read.csv("~/pml-training.csv")
testData <- read.csv("~/pml-testing.csv")
```

Next we partition the training dataset further into a training and testing set.

```r
inTrain <- createDataPartition(y = trainData$classe, p = .7, list = FALSE)
training <- trainData[inTrain,]
testing <- trainData[-inTrain,]
dim(training)
```

```
## [1] 13737   160
```

```r
dim(testing)
```

```
## [1] 5885  160
```

Cleaning the data

Before we build our model we remove variables that are mostly NA, have near zero variance (NZV), and those that are ID variables.

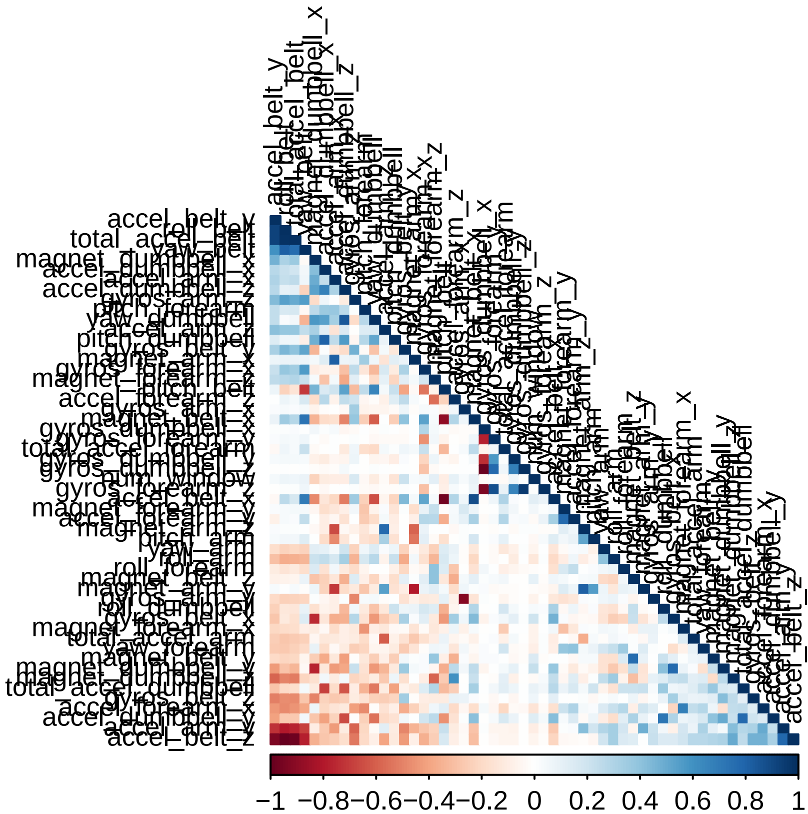
```r
#removing variables with near zero variance
nzv <- nearZeroVar(training)
training <- training[, -nzv]
testing <- testing[, -nzv]

#removing variables that are more than 90% NA
mostlyNA <- sapply(training, function(x) mean(is.na(x))) > 0.9
training <- training[, mostlyNA == FALSE]
testing <- testing[, mostlyNA == FALSE]

# remove identification only variables (columns 1 to 5)
training <- training [, -(1:5)]
testing <- testing[, -(1:5)]
```

The cleaning process narrows down the number of variables in our training and testing datasets from 180 to 54.

Exploring the data

Before we build our model, we need to be sure that there isn't a strong correlation between many of the variables.

```r
corMatrix <- cor(training[, -54])
corrplot(corMatrix, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```

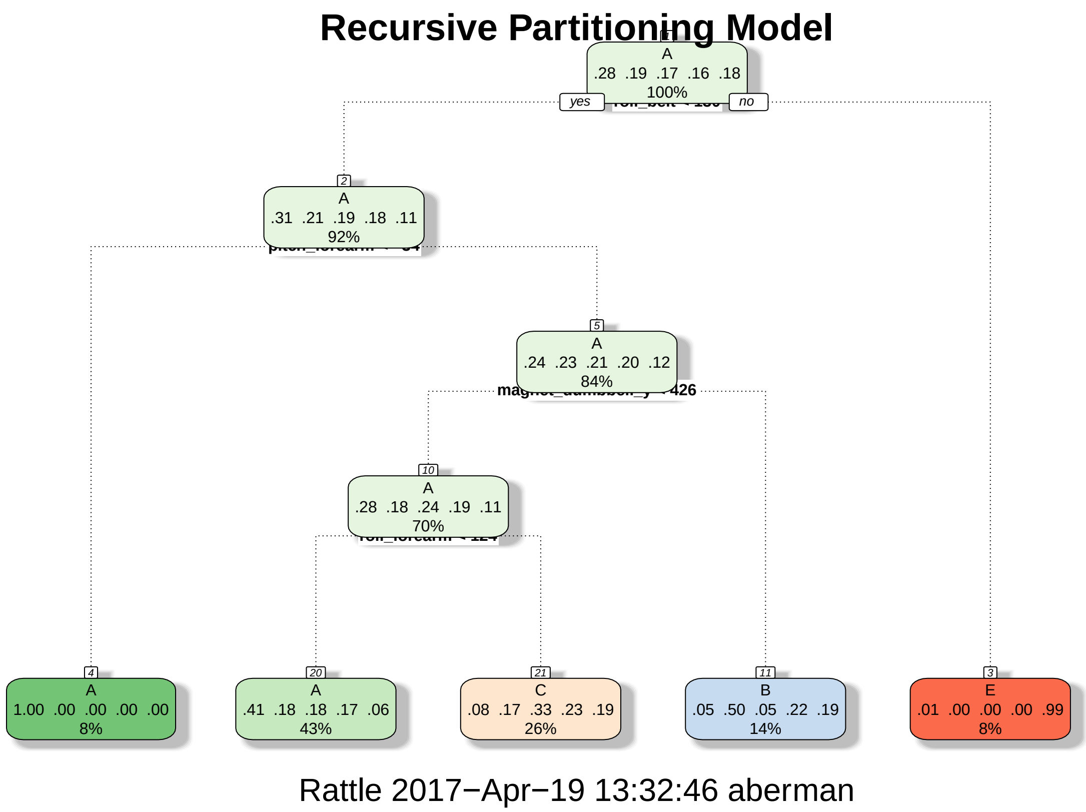


The highly correlated variables are shown in dark colors in the chart above. Principal Components Analysis (PCA) is a way to pre-process the data to make a more compact analysis. Given the lack of strong correlation between most of the variable, we will not perform PCA in the following analysis.

Fitting a model

We are ready to fit a model to our training dataset. Below we train with three methods, recurrsive paritioning, linear discriminant analysis, and random forests. The outputs show the 20 most predictive variables in each model.

```r
set.seed(125)
#Recursive partitioning regression tree modeling
modfit1 <- train(classe ~ ., method = "rpart", data = training)
fancyRpartPlot(modfit1$finalModel, cex = .5, main = "Recursive Partitioning Model")
```




```r
varImp(modfit1)
```

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 53)
##
##                      Overall
## pitch_forearm         100.00
## roll_forearm           73.02
## roll_belt              69.38
## magnet_dumbbell_y      49.85
## accel_belt_z           42.18
## yaw_belt               40.56
## num_window             40.48
## magnet_belt_y          40.35
## total_accel_belt       34.67
## magnet_arm_x           27.83
## accel_arm_x            26.27
## roll_dumbbell          18.71
## magnet_dumbbell_z      17.97
## roll_arm               16.02
## magnet_forearm_x        0.00
## total_accel_arm         0.00
## gyros_belt_z            0.00
## gyros_dumbbell_x        0.00
## accel_arm_y             0.00
```

```
## total_accel_dumbbell    0.00
```

```r
#Linear discriminant analysis modeling
modfit2 <- train(classe ~ ., method = "lda", data = training)
```

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

```r
varImp(modfit2)
```

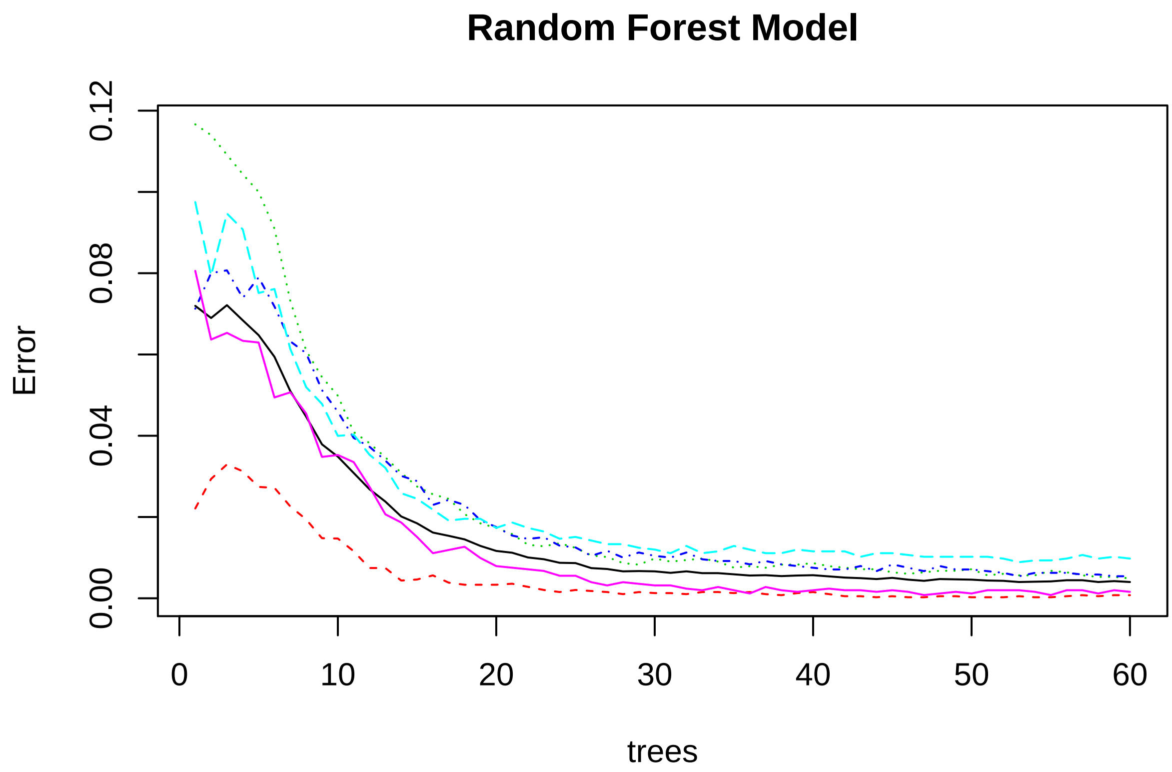
```
## ROC curve variable importance
##
##   variables are sorted by maximum importance across the classes
##   only 20 most important variables shown (out of 53)
##
##                          A       B     C     D      E
## pitch_forearm        76.27 100.000 77.49 76.27 100.00
## magnet_arm_x         69.91  86.112 78.75 69.91  86.11
## accel_arm_x          68.19  82.551 75.59 68.19  82.55
## magnet_dumbbell_x    77.46  77.455 77.46 77.46  66.80
## accel_dumbbell_x     72.84  72.843 72.84 72.84  65.83
## magnet_dumbbell_z    71.22  58.513 70.66 49.43  71.22
## num_window           70.23  70.235 70.23 70.23  65.25
## pitch_dumbbell       69.82  69.823 69.82 69.82  63.22
## roll_dumbbell        55.17  67.647 55.17 55.17  67.65
## magnet_dumbbell_y    65.01  65.013 65.01 65.01  49.47
## roll_belt            42.09  41.123 63.29 41.12  42.09
## accel_dumbbell_z     63.05  63.054 63.05 63.05  45.98
## magnet_arm_z         22.55   6.125  0.00 60.20  22.55
## roll_arm             57.07  57.071 57.07 57.07  49.85
## roll_forearm         56.76  39.073 43.71 50.10  56.76
## total_accel_forearm  50.73  52.585 56.25 50.73  52.58
## accel_forearm_y      55.19  33.273 36.03 51.10  55.19
## accel_arm_z          53.63  53.590 43.72 50.42  53.63
## magnet_belt_x        31.27  20.628 36.01 51.85  31.27
## magnet_forearm_y     39.22  20.879 20.88 50.65  39.22
```

```r
#Random forest modeling
modfit3 <- randomForest(classe ~ ., training, ntree = 60)
varImp(modfit3)
```

```
##                        Overall
## num_window           965.25412
## roll_belt            709.48353
## pitch_belt           480.94923
## yaw_belt             594.83243
## total_accel_belt     166.22215
## gyros_belt_x          66.52578
## gyros_belt_y          76.52201
## gyros_belt_z         187.16216
## accel_belt_x          81.26072
## accel_belt_y          93.98751
## accel_belt_z         269.58829
## magnet_belt_x        178.55285
## magnet_belt_y        234.07157
## magnet_belt_z        278.47825
## roll_arm             214.71419
```

```
## pitch_arm            100.86818
## yaw_arm              138.04486
## total_accel_arm       66.35632
## gyros_arm_x           80.19223
## gyros_arm_y           78.22871
## gyros_arm_z           37.62842
## accel_arm_x          170.98364
## accel_arm_y           77.98139
## accel_arm_z           77.77134
## magnet_arm_x         164.98584
## magnet_arm_y         138.60404
## magnet_arm_z         120.00393
## roll_dumbbell        257.90373
## pitch_dumbbell       110.40798
## yaw_dumbbell         174.74619
## total_accel_dumbbell 187.53614
## gyros_dumbbell_x      66.17493
## gyros_dumbbell_y     139.57767
## gyros_dumbbell_z      49.24366
## accel_dumbbell_x     152.22882
## accel_dumbbell_y     329.61951
## accel_dumbbell_z     228.92581
## magnet_dumbbell_x    344.67423
## magnet_dumbbell_y    456.15434
## magnet_dumbbell_z    495.49534
## roll_forearm         369.21796
## pitch_forearm        443.33536
## yaw_forearm          102.65203
## total_accel_forearm   57.29891
## gyros_forearm_x       40.18802
## gyros_forearm_y       67.65006
## gyros_forearm_z       52.64923
## accel_forearm_x      209.88131
## accel_forearm_y       86.18293
## accel_forearm_z      157.15392
## magnet_forearm_x     131.93394
## magnet_forearm_y     137.17799
## magnet_forearm_z     163.98528
```

```r
plot(modfit3, main = "Random Forest Model")
```



From our models, no consistent set of variables appears to achieve high predictability.

Assessing model accuracy

To select our final model, we use confusion matrices to determine the prediction accuracy of each model on our oaritioned testing test.

```
set.seed(233)
#compare classificiation accuracy on our testing set
#Recurrive partitioning regression tree model
confusionMatrix(predict(modfit1, testing), testing$classe)$overall['Accuracy']
```

```
##  Accuracy
## 0.4975361
```

```
#Linear discriminant analysis model
confusionMatrix(predict(modfit2, testing), testing$classe)$overall['Accuracy']
```

```
##  Accuracy
## 0.7119796
```

```
#Random forest model
confusionMatrix(predict(modfit3, testing), testing$classe)$overall['Accuracy']
```

```
##  Accuracy
## 0.9972812
```

Our analysis shows that our random forest model has the hightest prediction accuracy with 99.75%. With our random forest model, our expected out of sample error rate is 0.25%. We select the random forest model as our final model.

Predicting final test set

Finally, we apply our random forest model to our final test set.

```
set.seed(255)
predict(modfit3, testData)
```

```
##  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
```