Practical Machine Learning

Course Project

Paul Vonck

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, our goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants who were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The test set data contains the accelerometer data along with the grade of the movement known as the classe. We will build machine learning models to predict the outcome classe based on test accelerometer data.

Load Libraries

```
library(caret)
```

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

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

```
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

Data download

```
training <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),header=
dim(training)
```

```
## [1] 19622   160
```

```
testing <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),header=T
dim(testing)
```

```
## [1]  20 160
```

Clean Data

- Remove meta data columns
- Remove predictors with NAs
- Remove predictors with near zero variance

```
trainingCleaned<- training[, colSums(is.na(training)) == 0]
testingCleaned <- testing[, colSums(is.na(training)) == 0]
trainingCleaned <- trainingCleaned[, -c(1:7)]
testingCleaned <- testingCleaned[, -c(1:7)]
NZV <- nearZeroVar(trainingCleaned)
trainingCleaned <- trainingCleaned[, -NZV]
testingCleaned  <- testingCleaned[, -NZV]
dim(testingCleaned)
```

```
## [1] 20 53
```

```
dim(trainingCleaned)
```

```
## [1] 19622    53
```

Show cleaned data structure

- Predictors are numeric
- Outcome (classe) character factor

```
str(trainingCleaned)
```

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y        : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x        : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y        : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z        : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x       : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y       : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z       : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x         : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y         : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z         : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x        : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y        : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z        : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
```

```
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : chr  "A" "A" "A" "A" ...
```

Model Fitting

In order to limit the effects of overfitting, and improve the efficicency of the models, we will use 5 fold cross-validation. 10 fold had higher run times with no increased accuracy. We will build models with random forest and generalized boosted models.

```r
library(caret)

set.seed(257)

inTrain1 <- createDataPartition(trainingCleaned$classe, p=0.50, list=FALSE)
finalTrain <- trainingCleaned[inTrain1,]
finalTest <- trainingCleaned[-inTrain1,]

trControl <- trainControl(method="cv", number=5)
model_RF <- train(classe~., data=trainingCleaned, method="rf", trControl=trControl, verbose=FALSE)
model_GBM <- train(classe~., data=trainingCleaned, method="gbm", trControl=trControl, verbose=FALSE)
```

Model Prediction

- Predict with random forest model and display confusion matrix and accuracy
- Predict with GBM mode and display confusion matrix and accuracy

---

```r
trainpred <- predict(model_RF,newdata=finalTest)
trainpredgbm <- predict(model_GBM,newdata=finalTest)
finalTest$classe = as.factor(finalTest$classe)
confMatRF <- confusionMatrix(finalTest$classe,trainpred)
confMatRF$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2790    0    0    0    0
##          B    0 1898    0    0    0
##          C    0    0 1711    0    0
##          D    0    0    0 1608    0
##          E    0    0    0    0 1803
```

```r
confMatRF$overall[1]
```

```
## Accuracy
##        1
```

```r
confMatGBM <- confusionMatrix(finalTest$classe,trainpredgbm)
confMatGBM$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2760   24    2    3    1
##          B   40 1811   45    2    0
##          C    0   43 1648   19    1
##          D    2    3   45 1550    8
##          E    1   12    9   28 1753
```

```r
confMatGBM$overall[1]
```

```
##  Accuracy
## 0.9706422
```

Random Forest Model Characteristics

The Random Forest model was more accurate with the test prediction and we show some of the model characteristics below.

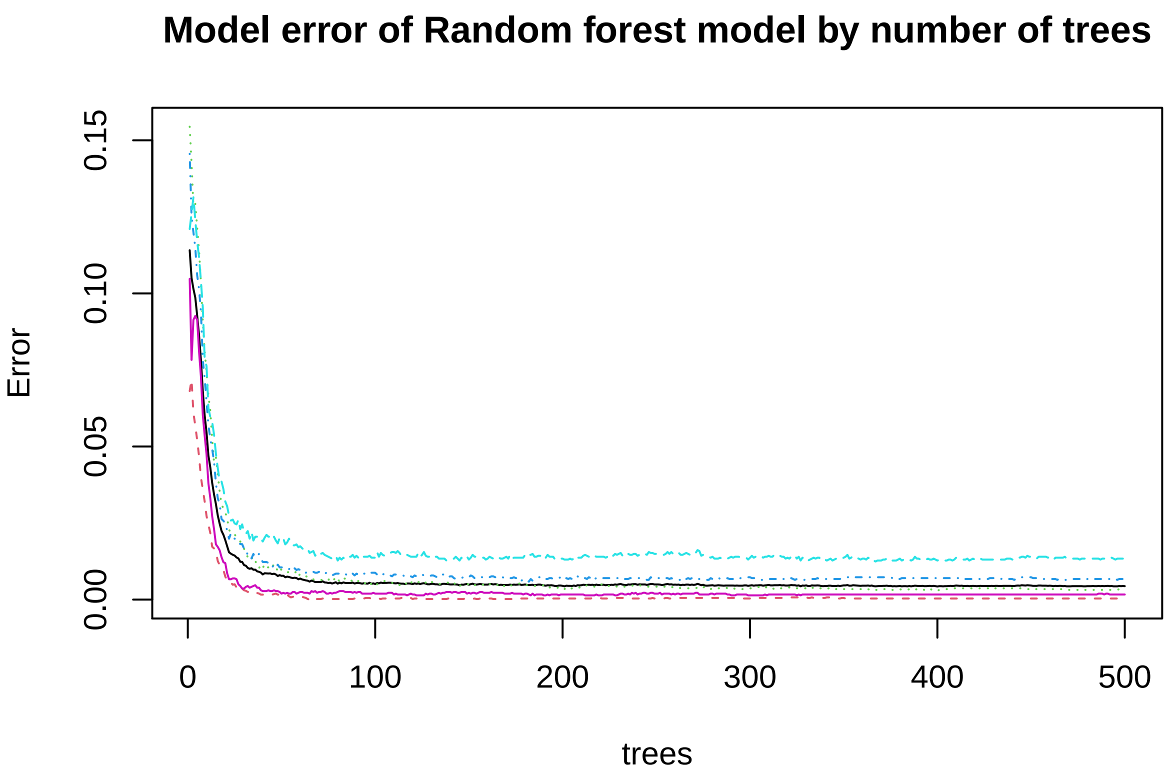
```r
print(model_RF)
```

```
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 15698, 15698, 15699, 15697, 15696
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9942924  0.9927800
##   27    0.9936807  0.9920065
##   52    0.9884317  0.9853659
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

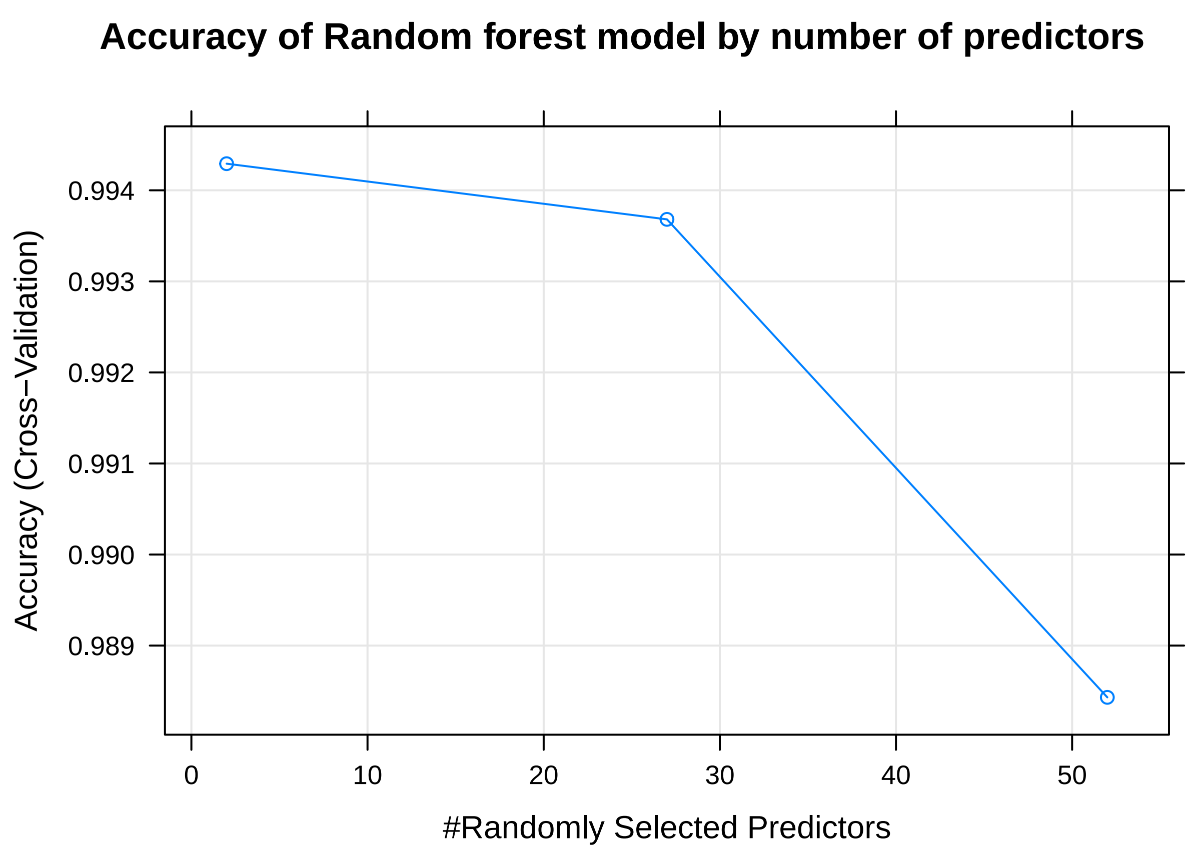
```r
model_RF$finalModel$classes
```

```
## [1] "A" "B" "C" "D" "E"
```

```r
plot(model_RF$finalModel,main="Model error of Random forest model by number of trees")
```




```r
plot(model_RF,main="Accuracy of Random forest model by number of predictors")
```




```r
MostImpVars <- varImp(model_RF)
MostImpVars
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                     Overall
## roll_belt            100.00
## yaw_belt              84.68
## magnet_dumbbell_z     73.67
## pitch_belt            63.31
## magnet_dumbbell_y     63.23
## pitch_forearm         62.52
## roll_forearm          55.84
## magnet_dumbbell_x     52.84
## accel_belt_z          48.48
## accel_dumbbell_y      46.49
## magnet_belt_z         45.80
## roll_dumbbell         44.75
## magnet_belt_y         42.75
## roll_arm              39.18
## accel_dumbbell_z      39.12
## accel_forearm_x       34.31
## gyros_belt_z          32.20
## accel_dumbbell_x      31.05
## yaw_dumbbell          30.67
```

```
## gyros_dumbbell_y    29.71
```

Conclusion and Test Set Prediction

We will use the random forest model to predict the values of classe for the test data set. We will also predict with the GBM model and compare with the random forest model prediction.

```r
predfinalRF <- predict(model_RF,newdata=testingCleaned)
predfinalGBM <- predict(model_RF,newdata=testingCleaned)
predfinalGBM==predfinalRF
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE
```

```r
predfinalRF
```

```
##  [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
```