Practical Machine Learning Final Project

Ruoshi Li

3/9/2020

Executive Summary

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. 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. I will be predicting the manner in which they did the exercise.

```r
# Set 'working directory'
wdir <- "/Users/Ruoshi/Documents/Study/Data Science/Coursera_Johns_Hopkins/Class 8/Final Project"
setwd(wdir)
# load necessary libraries
library(caret)
```

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

```r
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```r
library(randomForest)
```

```
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Data Exploration

```r
# import training and testing datasets
training_data <- read.csv("./pml-training.csv", header = T)
testing_data <- read.csv("./pml-testing.csv", header = T)
# check dataset dimensions
dim(training_data)
```

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

```r
dim(testing_data)
```

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

```r
# remove near zero variance variables
nzv <- nearZeroVar(training_data)
training_data <- training_data[ ,-nzv]
testing_data <- testing_data[ ,-nzv]
dim(training_data)
```

```
## [1] 19622   100
```

```r
dim(testing_data)
```

```
## [1]  20 100
```

```r
str(training_data)
```

```
## 'data.frame':    19622 obs. of  100 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 4844
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ 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 ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_dumbbell: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ total_accel_dumbbell    : int  37 37 37 37 37 37 37 37 37 37 ...
##  $ var_accel_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_dumbbell    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_dumbbell   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_dumbbell     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 ...
##  $ max_picth_forearm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_forearm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_forearm : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ total_accel_forearm     : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ var_accel_forearm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 ...
##   [list output truncated]
```

We can see from the output that there are many NA cols in the dataset. Also, the first 6 cols are not related to the variable we are trying to predict. Will clean the dataset by removing those columns.

```r
# remove cols that more than 90% of their values are NAs from training dataset and testing dataset
training_remove <- which(colSums(is.na(training_data) |training_data=="")>0.9*dim(training_data)[1])
training_data <- training_data[ , -training_remove]
testing_data <- testing_data[ , -training_remove]
# remove first 6 unrelated cols
training_data <- training_data[ , -c(1:6)]
testing_data <- testing_data[ , -c(1:6)]
dim(training_data)
```

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

```r
# split training dataset into training and testing parts
set.seed(123)
inTrain <- createDataPartition(training_data$classe, p = 0.7, list = FALSE)
training <- training_data[inTrain, ]
testing <- training_data[-inTrain, ]
```

Model Building

We will try three models in our model selection process: classification tree, gradient boosting and random forest. And we will compare their accuracy to pick the best one for our prediction.

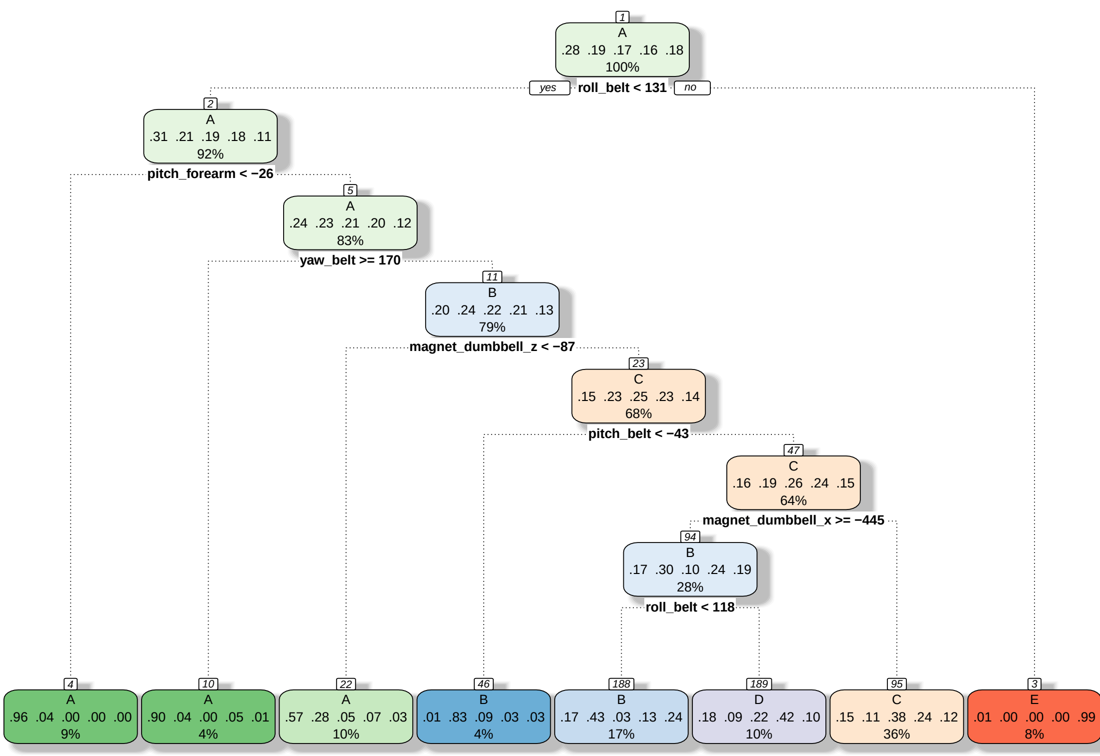
Prediction with Classification Tree

```r
set.seed(1234)
fitControl <- trainControl(method = "cv", number = 5, verboseIter = F)
mod_tree <- train(classe ~ ., data = training, method = "rpart", trControl = fitControl)
mod_tree$finalModel
```

```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
```

```
##       * denotes terminal node
##
##    1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##      2) roll_belt< 130.5 12580 8681 A (0.31 0.21 0.19 0.18 0.11)
##        4) pitch_forearm< -26.45 1238   45 A (0.96 0.036 0 0 0) *
##        5) pitch_forearm>=-26.45 11342 8636 A (0.24 0.23 0.21 0.2 0.12)
##         10) yaw_belt>=169.5 550   56 A (0.9 0.044 0 0.049 0.0091) *
##         11) yaw_belt< 169.5 10792 8203 B (0.2 0.24 0.22 0.21 0.13)
##           22) magnet_dumbbell_z< -87.5 1420  615 A (0.57 0.28 0.051 0.075 0.025) *
##           23) magnet_dumbbell_z>=-87.5 9372 7048 C (0.15 0.23 0.25 0.23 0.14)
##             46) pitch_belt< -42.95 576   96 B (0.014 0.83 0.092 0.033 0.028) *
##             47) pitch_belt>=-42.95 8796 6525 C (0.16 0.19 0.26 0.24 0.15)
##               94) magnet_dumbbell_x>=-445.5 3783 2642 B (0.17 0.3 0.097 0.24 0.19)
##                188) roll_belt< 117.5 2357 1340 B (0.17 0.43 0.025 0.13 0.24) *
##                189) roll_belt>=117.5 1426  830 D (0.18 0.087 0.22 0.42 0.097) *
##               95) magnet_dumbbell_x< -445.5 5013 3110 C (0.15 0.11 0.38 0.24 0.12) *
##      3) roll_belt>=130.5 1157    7 E (0.0061 0 0 0 0.99) *
```

```r
suppressMessages(library(rattle))
fancyRpartPlot(mod_tree$finalModel)
```



Rattle 2020−Mar−09 17:45:22 Ruoshi

```r
tree_pred <- predict(mod_tree, newdata = testing)
tree_confMatrix <- confusionMatrix(testing$classe, tree_pred)
tree_confMatrix$overall[1]
```

```
##  Accuracy
## 0.5522515
```

The accuracy for classification tree model is 55.23%, which is not good. We will then try different methods.

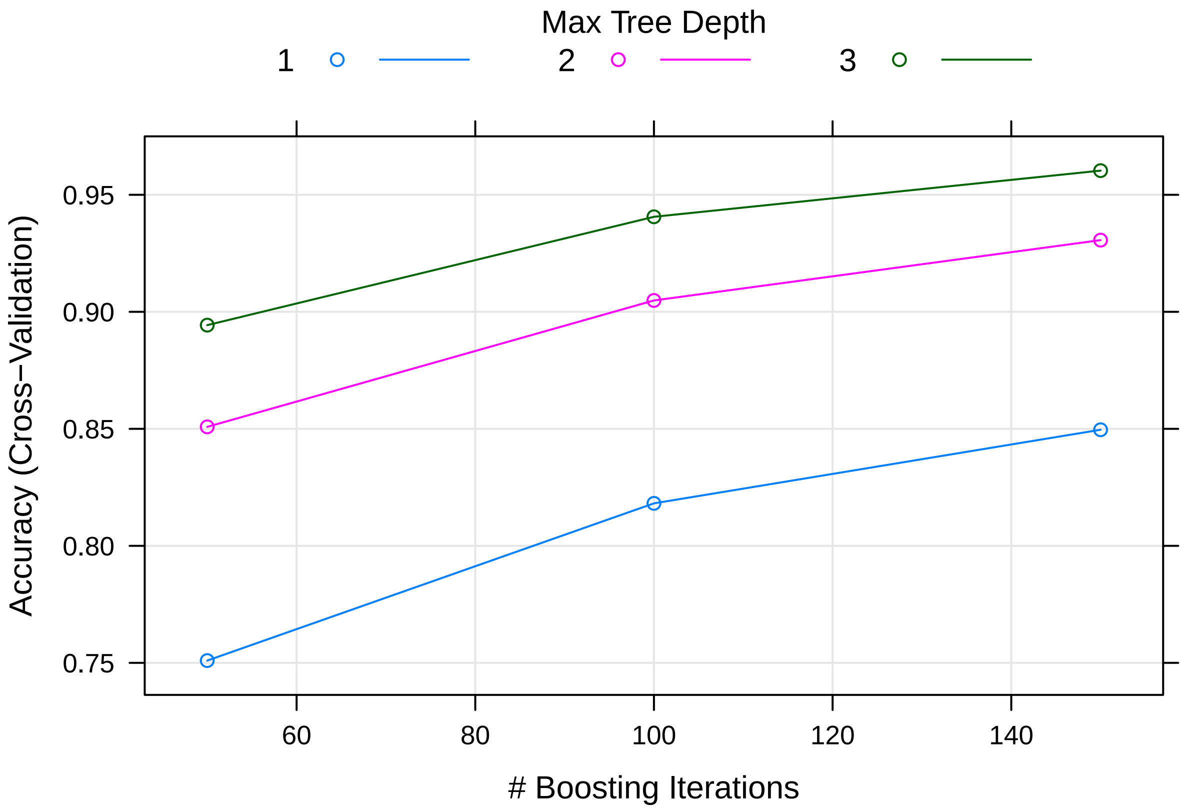
Prediction with Gradient Boosting Method

```
mod_gbm <- train(classe ~ ., data = training, method = "gbm", trControl = fitControl)
```

```
mod_gbm$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
```

```
plot(mod_gbm)
```




```
gbm_pred <- predict(mod_gbm, newdata = testing)
gbm_confMatrix <- confusionMatrix(testing$classe, gbm_pred)
gbm_confMatrix$overall[1]
```

```
##  Accuracy
## 0.9575191
```

The accuracy for gradient boosting model is 95.84%. It is improved a lot from the previous classification model. We will now try the last method.

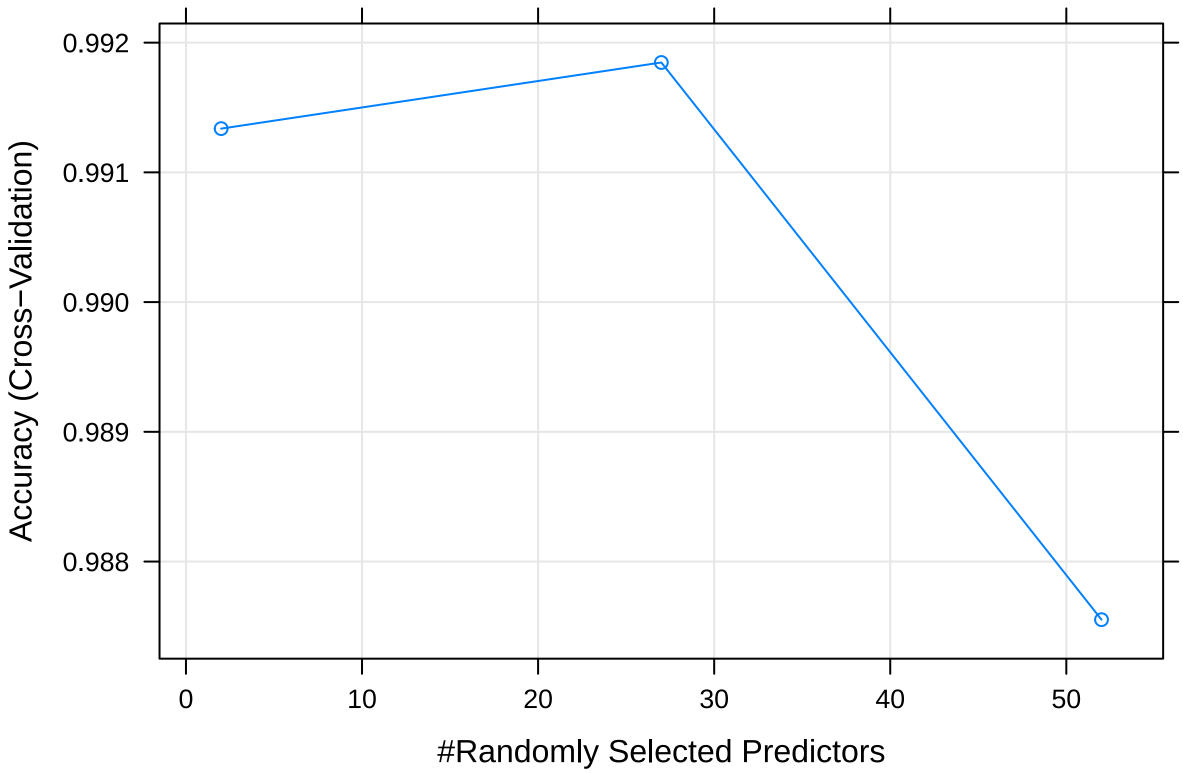
Prediction with Random Forest

```
mod_rf <- train(classe ~ ., data = training, method = "rf", trControl = fitControl)
mod_rf$finalModel
```

```
##
## Call:
```

```
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.66%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3901    3    1    0    1 0.001280082
## B   21 2629    7    1    0 0.010910459
## C    0   12 2379    5    0 0.007095159
## D    0    0   26 2225    1 0.011989343
## E    0    1    4    7 2513 0.004752475
```

```r
plot(mod_rf)
```




```r
rf_pred <- predict(mod_rf, newdata = testing)
rf_confMatrix <- confusionMatrix(testing$classe, rf_pred)
rf_confMatrix$overall[1]
```

```
##  Accuracy
## 0.9925234
```

The accuracy for random forest model is 99.2%, which is very ideal. However, we will need to check the out of sample error to see if we have overfitting issue.

Prediction with test data

```r
# use random forest model to predict since this is the model with highest accuracy
test_pred <- predict(mod_rf, newdata = testing_data)
test_pred
```

```
##  [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
```

Conclusion

We can see from the above analysis that random forest model has the highest accuracy of in predicting classe