Machine Learning to Predict Type of Exercise

by Darryl Ma

Executive Summary

We set out to create a prediction model that would be able to predict if a barbell lift was being performed properly given measurements from accelerometers placed on the belt, forearm, arm, and dumbbell. We managed to create a random forest model that achieved 100% accuracy on both our validation and testing datasets. Additionally, we determined that in order to perform a barbell lift correctly, it is most important that the subject maintain his/her hips in the proper position as well as complete the full forearm motion.

Introduction

The goal of this project is to predict the manner in which a subject performed barbell lifts based on input from accelerometers placed on the belt, forearm, arm, and dumbbell. Six participants of age between 20-28 were asked to perform barbell lifts correctly and incorrectly in 5 different ways:

1. Class A: exactly according to the specification
2. Class B: throwing the elbows to the front
3. Class C: lifting the dumbbell only halfway
4. Class D: lowering the dumbbell only halfway
5. Class E: throwing the hips to the front

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. The datasets provided contain various quantitative measurements as each of these participants performed different types of barbell lifts. The aim is to categorize/predict how well an activity was performed based on these measurements.

Loading Data

Two datasets were provided, one training set and one testing set. The code below downloads and loads the datasets into R:

```
training <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),
                    header = TRUE, na.strings = c("NA", ""))
testing <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),
                    header = TRUE, na.strings = c("NA", ""))
```

The training set contains 19622 observations and 160 variables, and the testing set contains 20 observations and 160 variables.

```
str(training)
```

```
## 'data.frame':    19622 obs. of  160 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 13230
84232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2     : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484434
...
##  $ cvtd_timestamp           : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window               : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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 ...
##  $ kurtosis_roll_belt       : Factor w/ 396 levels "-0.016850","-0.021024",..: NA NA NA NA NA NA NA NA NA
NA ...
##  $ kurtosis_picth_belt      : Factor w/ 316 levels "-0.021887","-0.060755",..: NA NA NA NA NA NA NA NA NA
NA ...
##  $ kurtosis_yaw_belt        : Factor w/ 1 level "#DIV/0!": NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt       : Factor w/ 394 levels "-0.003095","-0.010002",..: NA NA NA NA NA NA NA NA NA
NA ...
##  $ skewness_roll_belt.1     : Factor w/ 337 levels "-0.005928","-0.005960",..: NA NA NA NA NA NA NA NA NA
NA ...
##  $ skewness_yaw_belt        : Factor w/ 1 level "#DIV/0!": NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 ...
##  $ max_yaw_belt             : Factor w/ 67 levels "-0.1","-0.2",..: 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 ...
##  $ min_yaw_belt            : Factor w/ 67 levels "-0.1","-0.2",..: 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 ...
##  $ amplitude_yaw_belt      : Factor w/ 3 levels "#DIV/0!","0.00",..: 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 ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_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 ...
##  $ kurtosis_roll_arm       : Factor w/ 329 levels "-0.02438","-0.04190",..: NA NA NA NA NA NA NA NA NA NA
...
##  $ kurtosis_picth_arm      : Factor w/ 327 levels "-0.00484","-0.01311",..: NA NA NA NA NA NA NA NA NA NA
...
##  $ kurtosis_yaw_arm        : Factor w/ 394 levels "-0.01548","-0.01749",..: NA NA NA NA NA NA NA NA NA NA
...
##  $ skewness_roll_arm       : Factor w/ 330 levels "-0.00051","-0.00696",..: NA NA NA NA NA NA NA NA NA NA
...
##  $ skewness_pitch_arm      : Factor w/ 327 levels "-0.00184","-0.01185",..: NA NA NA NA NA NA NA NA NA NA
...
##  $ skewness_yaw_arm        : Factor w/ 394 levels "-0.00311","-0.00562",..: NA NA NA NA NA NA NA NA NA NA
...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  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_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  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 ...
##  $ kurtosis_roll_dumbbell  : Factor w/ 397 levels "-0.0035","-0.0073",..: NA NA NA NA NA NA NA NA NA NA .
..
```

```
##  $ kurtosis_picth_dumbbell : Factor w/ 400 levels "-0.0163","-0.0233",..: NA NA NA NA NA NA NA NA NA NA .
..
##  $ kurtosis_yaw_dumbbell   : Factor w/ 1 level "#DIV/0!": NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : Factor w/ 400 levels "-0.0082","-0.0096",..: NA NA NA NA NA NA NA NA NA NA .
..
##  $ skewness_pitch_dumbbell : Factor w/ 401 levels "-0.0053","-0.0084",..: NA NA NA NA NA NA NA NA NA NA .
..
##  $ skewness_yaw_dumbbell   : Factor w/ 1 level "#DIV/0!": NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 ...
##  $ max_yaw_dumbbell        : Factor w/ 72 levels "-0.1","-0.2",..: 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 ...
##  $ min_yaw_dumbbell        : Factor w/ 72 levels "-0.1","-0.2",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

A quick glance at the list of variables below shows that a lot of the variables have NA values. Therefore our first objective is to remove any unnecessary variables that will not have an impact in the classification of the barbell lift class.

Cleaning the Data

Firstly we selected to only keep variables which did not contain any NA values:

```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
```

This reduced the number of variables to 60 and 60 in the training and testing set, respectively.

```
training <- training[, -c(1:7)]
testing <- testing[, -c(1:7)]
```

Next, we noticed that the first 7 variables: X, username, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window and num_window, would probably not have a great impact on how we would classify different types of barbell lifts. Consequently, these first 7 variables were removed from the training and testing sets, further reducing the variable count to 53 and 53 in the training and testing set, respectively.

```
str(training)
```

```
## '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              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

The final list of features above all appear to be possible factors in determining the final output variable, classe.

Data Splitting

The training dataset was further split into a cross validation dataset to measure the effectiveness of each prediction model

```
set.seed(1234)
inTrain <- createDataPartition(training$classe, p=0.6, list=FALSE)
training <- training[inTrain,]
validation <- training[-inTrain,]
```

Prediction Models

A few prediction models were chosen to try and solve the classification problem: decision tree, random forest and gradient boosting method.

Decision Tree

The following code uses the training set to fit a decision tree model, predicts the classe outcome on the validation set, and reports the confusion matrix and accuracy of the prediction model.

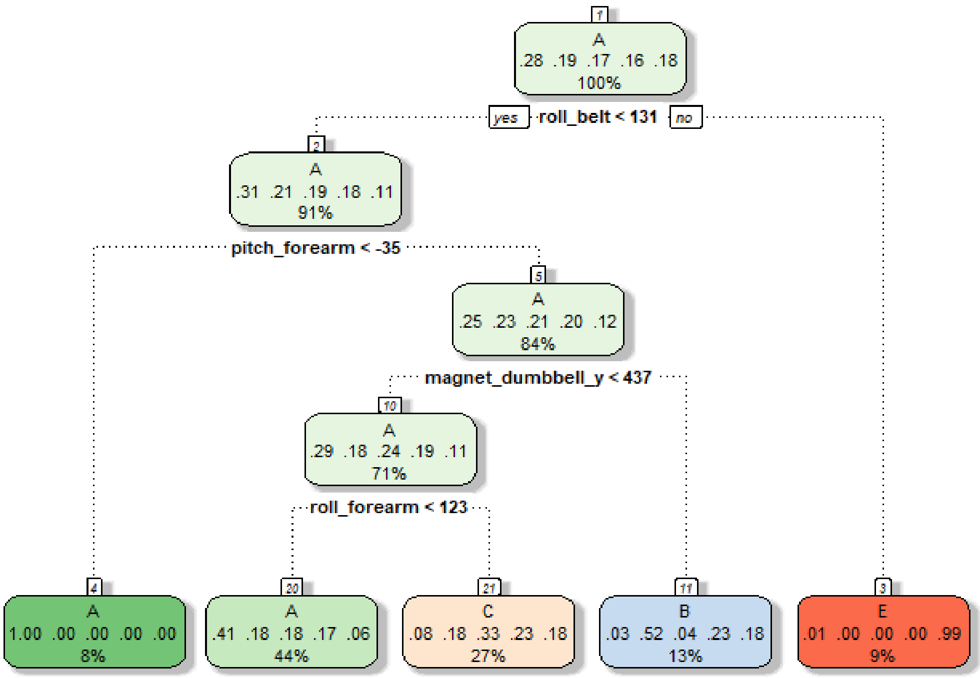
```
rpartFit <- train(classe ~ ., data=training, method="rpart")
rpartPred <- predict(rpartFit, validation)
confusionMatrix(rpartPred, validation$classe)$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1219  360  363  339  134
##          B   20  314   24  144   94
##          C  102  230  427  291  245
##          D    0    0    0    0    0
##          E    5    0    0    0  381
```

```
rpart_acc <- confusionMatrix(rpartPred, validation$classe)$overall[1]
rpart_acc
```

```
##  Accuracy
## 0.4989344
```

```
fancyRpartPlot(rpartFit$finalModel)
```



Rattle 2018-Aug-24 00:05:25 Darryl

From the output above, we see that the decision tree does not accurately predict the outcome, classe, very well only achieving 49.9% accuracy. The confusion matrix further shows us that this model was particularly very poor at classifying class D, which is corroborated by the dendrogram, where there is no node that classifies any observation as class D.

Random Forest

The following code uses the training set to fit a random forest model, predicts the classe outcome on the validation set, and reports the confusion matrix and accuracy of the prediction model.

```
rfFit <- train(classe ~ ., data=training, method="rf")
rfPred <- predict(rfFit, validation)
confusionMatrix(rfPred, validation$classe)$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1346    0    0    0    0
##          B    0  904    0    0    0
##          C    0    0  814    0    0
##          D    0    0    0  774    0
##          E    0    0    0    0  854
```

```
rf_acc <- confusionMatrix(rfPred, validation$classe)$overall[1]
rf_acc
```

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

From the output above, we see that the random forest accurately predicts the outcome, classe, very well, achieving 100% accuracy on the validation set.

Gradient Boost Model

The following code uses the training set to fit a gradient boost model, predicts the classe outcome on the validation set, and reports the confusion matrix and accuracy of the prediction model.

```
gbmFit <- train(classe ~ ., data=training, method="gbm", verbose=FALSE)
gbmPred <- predict(gbmFit, validation)
confusionMatrix(gbmPred, validation$classe)$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1332   16    0    2    1
##          B    7  874   20    1    6
##          C    3   13  791   27    9
##          D    4    1    2  742   11
##          E    0    0    1    2  827
```

```
gbm_acc <- confusionMatrix(gbmPred, validation$classe)$overall[1]
gbm_acc
```

```
##  Accuracy
## 0.9731458
```

From the output above, we see that the gradient boost model accurately predicts the outcome, classe, very well, achieving 97.3% accuracy on the validation set.

Most Influential Variables

```
imp <- data.frame(varImp(rfFit)[1])
top_ten_imp <- rownames(imp)[order(imp$Overall, decreasing=TRUE)[1:10]]
top_ten_imp
```

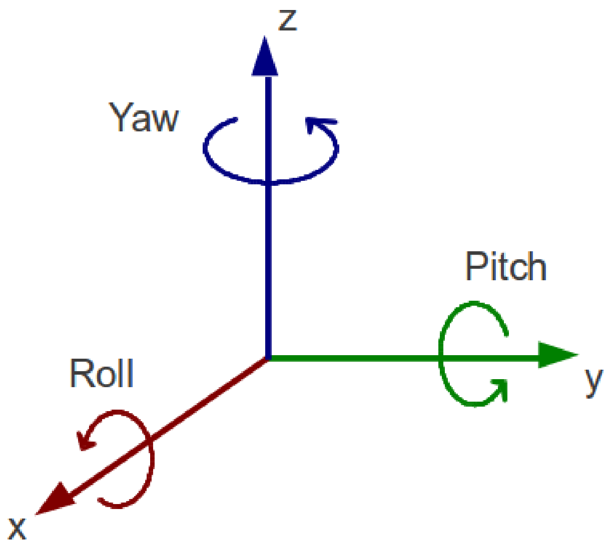
```
##  [1] "roll_belt"         "yaw_belt"          "magnet_dumbbell_z"
##  [4] "magnet_dumbbell_y" "pitch_belt"        "pitch_forearm"
##  [7] "magnet_dumbbell_x" "roll_forearm"      "magnet_belt_z"
## [10] "roll_dumbbell"
```

```
avg_by_classe <- aggregate(. ~ classe, training, mean)
avg_by_classe[,c("classe", top_ten_imp[1:3])]
```

```
##   classe roll_belt   yaw_belt magnet_dumbbell_z
## 1      A  60.20939  -9.869695          12.60024
## 2      B  64.38249 -15.486705          46.96928
## 3      C  64.58860  -7.676091          62.65871
## 4      D  60.03787 -19.164720          56.52850
## 5      E  75.17748  -4.698296          73.64619
```

The top 10 variables that influenced the random forest model in determining the classe outcome are printed above. As you can see roll_belt, yaw_belt, and magnet_dumbbell_z were the top 3 most influential variables (refer to image below for a visual definition of pitch, yaw and roll). In general, we can conclude that being able to perform a barbell lift correctly has a lot to do with maintaining your hips in the proper position as well as making sure you complete full forearm motion. More specifically, when performing a barbell lift, you should be aiming for the following roll_belt, yaw_belt, and magnet_dumbbell_z values:

- roll_belt: 60.21
- yaw_belt: -9.87
- magnet_dumbbell_z: 12.6



Graphical Depiction of Pitch, Yaw and Roll

Conclusion

```
testPred <- predict(rfFit, testing)
testPred
```

```
##  [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
```

Of all the models, we found that the random forest achieved the highest accuracy, 100%. The accuracies achieved by the other models are listed below:

1. Decision Tree: 49.9%
2. Gradient Boost Model: 97.3%
3. Random Forest: 100%

When the random forest model was used to predict the classe outcome for the testing set, it again achieved 100% accuracy, predicting the classe outcome correctly for all 20 observations. Additionally, we determined that in order to perform a barbell lift correctly, it is most important that the subject maintain his/her hips in the proper position as well as complete the full forearm motion.