Predicting Exersize Performance

Mike Pennell

June 12, 2016

Executive Summary

This study attempts to predict how well users execute weight-lifting exercises and detect mistakes using sensors attached to participants and equipment. Study participant were asked to perform barbell lifts correctly and incorrectly in 6 different ways while their movements were measured. A random forest model was trained to predict whether the exercise was performed correctly or the class of mistake made. The model achieved accuracy exceeding .985 indicating there is high confidence the model will predict accurately for these same users and exercises. However, validation of the model by leaving the data from 1 participant out of the training set and validating against the data from the participant left out showed poor accuracy. Therefore, the model is inaccurate at predicting whether any other user performing these exercises are doing so correctly. This was the intent of the study, but was not included in the requirements for this project as the test data did not support this usage.

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. The purpose of the study used in this analysis was to quantify how well users did a particular activity and detect mistakes in weight-lifting exercises through activity recognition techniques using wearable sensors and machine learning to classify each mistake. The data is from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. Each participant was asked to perform barbell lifts correctly and incorrectly in 5 different ways.

Inertial measurement units (IMU) provided three-axes acceleration, gyroscope and magnetometer data at a joint sampling rate of 45 Hz. Each IMU also featured a Bluetooth module to stream the recorded data to a notebook/ Participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in the following fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years with little weight lifting experience using a relatively light dumbbell (1.25kg).

Discovery and Preparation

The training and test data were reviewed to assess best options for modeling.

Training and Test Data

- Training: 19,622 sets of time series sensor readings from 6 participants performing exercises in 1 correct method (class = A) and 4 incorrect methods (class = B, C, D, E)
- Testing: 20 selected individual sensors readings collected from each of the 6 participant. Class is unknown.

```r
nrow(train) # Training Data
```

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

```r
summary(train$user_name) # Participant Data
```

```
##   adelmo carlitos  charles   eurico   jeremy    pedro
##     3892     3112     3536     3070     3402     2610
```

```r
summary(train$classe) # Class of exercise performance
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```r
# Naive Class Prediction; Majority Class: A
summary(train$classe)/length(train$classe)
```

```
##         A         B         C         D         E
## 0.2843747 0.1935073 0.1743961 0.1638977 0.1838243
```

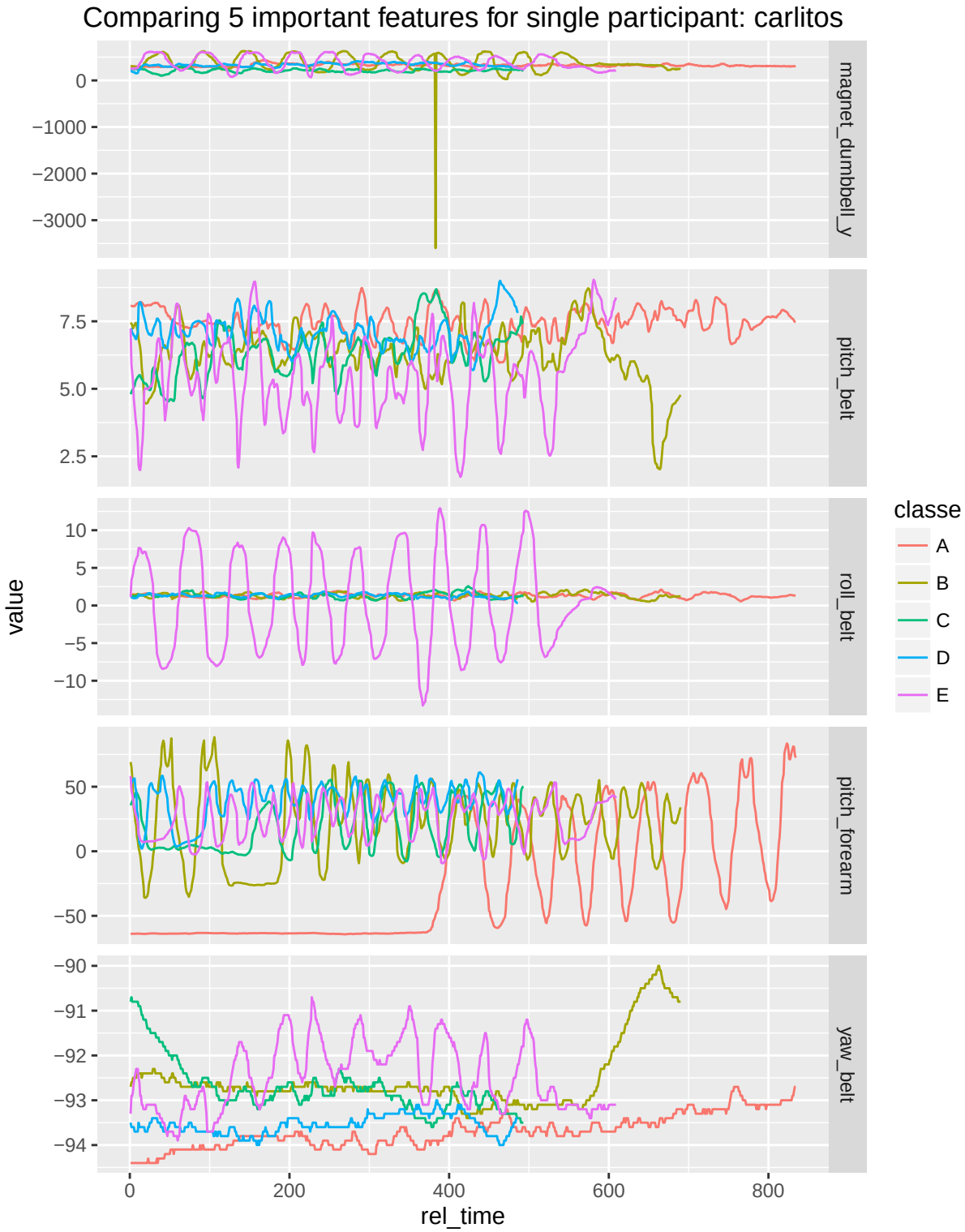
```r
nzvTrain <- nearZeroVar(train, saveMetrics = TRUE)
nzvTest <- nearZeroVar(test, saveMetrics = TRUE)
trAvail <- train[,!nzvTest$zeroVar]

trAvail$full_time <- trAvail$raw_timestamp_part_1 * 1000000 + trAvail$raw_timestamp_part_2
users <- unique(trAvail$user_name)
classes <- unique(trAvail$classe)
trAvail <- arrange(trAvail, user_name, classe, full_time)
trAvail$rel_time <- 0
for (user in users) {
  for (class in classes) {
    rl <- nrow(trAvail[trAvail$user_name == user & trAvail$classe == class,])
    trAvail[trAvail$user_name == user & trAvail$classe == class, "rel_time"] = seq(1:rl)
  }
}
```

Time Series

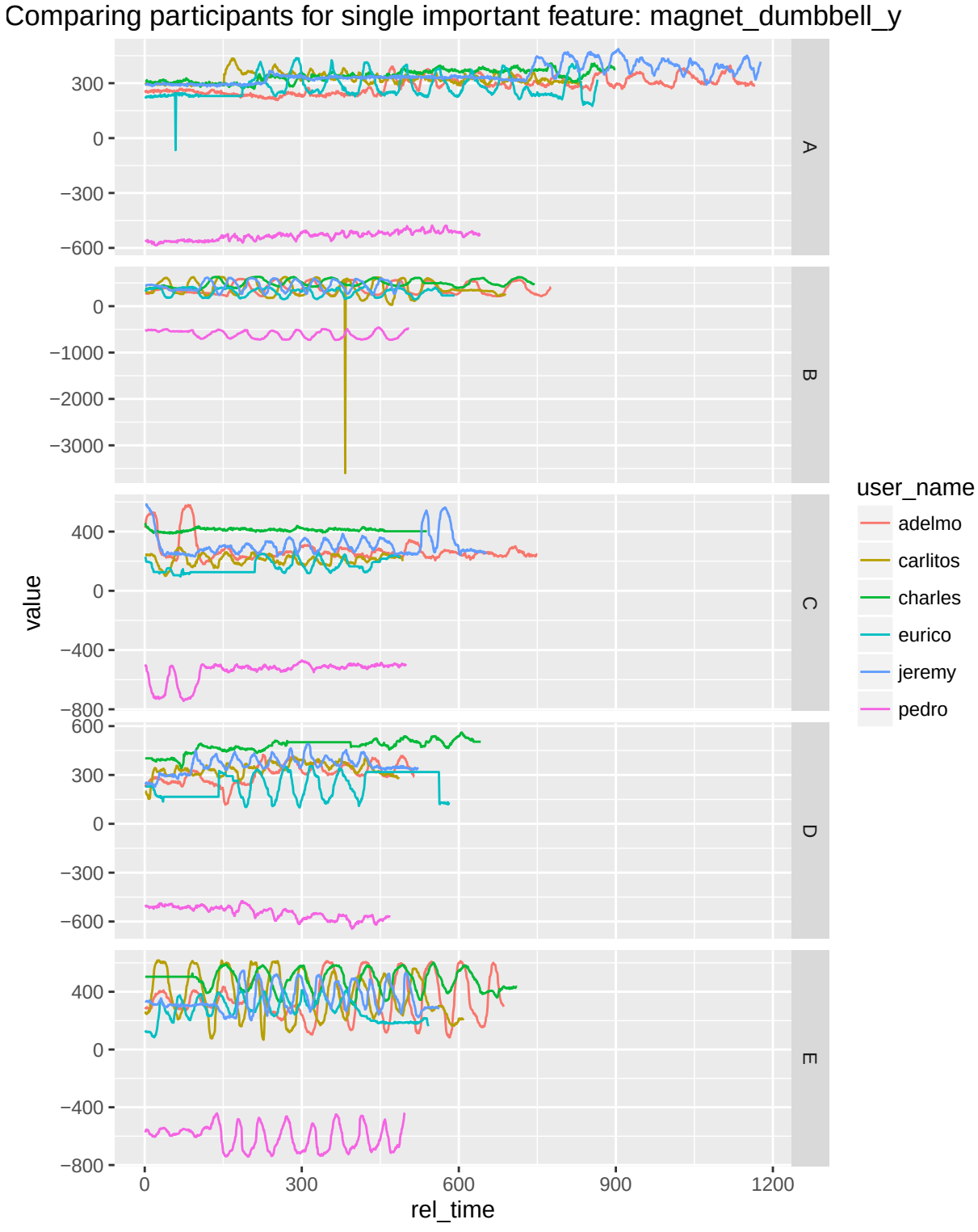
The data is a time series data set and the exercises are completed in a series of repetitive motions. Ideally, the data should be modeled as a time series and features extracted based on patterns in sequential readings of each sensor. As such a relative timer was derived for each set of 10 repetitions to enable visualization and comparison of the different exercises by user and sensor feature. The following visualizations illustrate the patterns showing 1) 5 important sensor readings comparing each class of exercise (A-E) and 2) a single important sensor reading (magnet_dumbbell_y) for each user. It is evident that the individual sensor readings vary significantly at different time points in the exercise and that readings from individual users vary significantly from other participants performing the exercise in the same manner (same class).

```r
# Monitoring time series readings for 1 user comparing different classes (exersize mistakes)
userMelt <- melt(trAvail[trAvail$user_name==users[1],], id.vars = c("classe", "user_name", "rel_time"),
userPlot <- ggplot(data = userMelt, aes(x = rel_time, y = value)) +
    geom_line(aes(colour = classe)) +
    facet_grid(feature ~ ., scales = "free_y") +
    labs(title= paste("Comparing 5 important features for single participant:", users[1]))
userPlot
```




```
# Monitoring time series readings for all users for single important feature by class)
sensorMelt <- melt(trAvail, id.vars = c("classe", "user_name", "rel_time"), measure.vars = shortlist[2]
featurePlot <- ggplot(data = sensorMelt, aes(x = rel_time, y = value)) +
    geom_line(aes(colour = user_name)) +
    facet_grid(classe ~ ., scales = "free_y") +
```

```
    labs(title= paste("Comparing participants for single important feature:", shortlist[2]))
featurePlot
```



Test Data Limitations

The test data is not a time series, but is instead a set of 20 discontinuous sensor readings from each of the participants. Therefore, it is not feasible to use a time series model to predict the test data.

Also, the training data includes window based aggregates that provide an indicator as to the variance and trending of the data over time. This aggregate data is not available in the test data and needs to be excluded from the model to enable accurate prediction. As a result, the following features were excluded:

```
names(test)[nzvTest$zeroVar]
```

```
##   [1] "new_window"               "kurtosis_roll_belt"
##   [3] "kurtosis_picth_belt"      "kurtosis_yaw_belt"
##   [5] "skewness_roll_belt"       "skewness_roll_belt.1"
##   [7] "skewness_yaw_belt"        "max_roll_belt"
##   [9] "max_picth_belt"           "max_yaw_belt"
##  [11] "min_roll_belt"            "min_pitch_belt"
##  [13] "min_yaw_belt"             "amplitude_roll_belt"
##  [15] "amplitude_pitch_belt"     "amplitude_yaw_belt"
##  [17] "var_total_accel_belt"     "avg_roll_belt"
##  [19] "stddev_roll_belt"         "var_roll_belt"
##  [21] "avg_pitch_belt"           "stddev_pitch_belt"
##  [23] "var_pitch_belt"           "avg_yaw_belt"
##  [25] "stddev_yaw_belt"          "var_yaw_belt"
##  [27] "var_accel_arm"            "avg_roll_arm"
##  [29] "stddev_roll_arm"          "var_roll_arm"
##  [31] "avg_pitch_arm"            "stddev_pitch_arm"
##  [33] "var_pitch_arm"            "avg_yaw_arm"
##  [35] "stddev_yaw_arm"           "var_yaw_arm"
##  [37] "kurtosis_roll_arm"        "kurtosis_picth_arm"
##  [39] "kurtosis_yaw_arm"         "skewness_roll_arm"
##  [41] "skewness_pitch_arm"       "skewness_yaw_arm"
##  [43] "max_roll_arm"             "max_picth_arm"
##  [45] "max_yaw_arm"              "min_roll_arm"
##  [47] "min_pitch_arm"            "min_yaw_arm"
##  [49] "amplitude_roll_arm"       "amplitude_pitch_arm"
##  [51] "amplitude_yaw_arm"        "kurtosis_roll_dumbbell"
##  [53] "kurtosis_picth_dumbbell"  "kurtosis_yaw_dumbbell"
##  [55] "skewness_roll_dumbbell"   "skewness_pitch_dumbbell"
##  [57] "skewness_yaw_dumbbell"    "max_roll_dumbbell"
##  [59] "max_picth_dumbbell"       "max_yaw_dumbbell"
##  [61] "min_roll_dumbbell"        "min_pitch_dumbbell"
##  [63] "min_yaw_dumbbell"         "amplitude_roll_dumbbell"
##  [65] "amplitude_pitch_dumbbell" "amplitude_yaw_dumbbell"
##  [67] "var_accel_dumbbell"       "avg_roll_dumbbell"
##  [69] "stddev_roll_dumbbell"     "var_roll_dumbbell"
##  [71] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
##  [73] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
##  [75] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
##  [77] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
##  [79] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
##  [81] "skewness_pitch_forearm"   "skewness_yaw_forearm"
##  [83] "max_roll_forearm"         "max_picth_forearm"
##  [85] "max_yaw_forearm"          "min_roll_forearm"
```

```
##  [87] "min_pitch_forearm"         "min_yaw_forearm"
##  [89] "amplitude_roll_forearm"    "amplitude_pitch_forearm"
##  [91] "amplitude_yaw_forearm"     "var_accel_forearm"
##  [93] "avg_roll_forearm"          "stddev_roll_forearm"
##  [95] "var_roll_forearm"          "avg_pitch_forearm"
##  [97] "stddev_pitch_forearm"      "var_pitch_forearm"
##  [99] "avg_yaw_forearm"           "stddev_yaw_forearm"
## [101] "var_yaw_forearm"
```

Assumed Modeling Objective

Based on the above limitations of the test data, it is assumed that the objective of this analysis is to determine if a model can classify exercise performance (class) based on sensor readings at any specific point in time. This is consistent with the test data.

Note: Given the obvious time series nature of the data, it seems unrealistic to expect a model to be predictive given just data from a single point in time, but that is what the test data requires so that is modeled in this analysis.

Leakage

Additionally, the model should eliminate other data which would be unavailable at the time of model execution and is available only as a result of the experimental procedures and data collection. This is called leakage. A random forest model was generated based on the training data, including all predictors provided in the test data (but not the test data). The most important features were reviewed to identify potential leakage. The 20 most important features were:

```
# md <- train(classe ~ .,method="rf", data=trAvail, importance = TRUE)
mdOrder <- order(with(varImp(md)$importance, pmax(A,B,C,D,E)),decreasing=TRUE)
row.names(varImp(md)$importance[mdOrder,])[1:20]
```

```
##  [1] "X"                              "roll_belt"
##  [3] "full_time"                      "pitch_forearm"
##  [5] "raw_timestamp_part_1"           "accel_belt_z"
##  [7] "rel_time"                       "cvtd_timestamp02/12/2011 14:57"
##  [9] "roll_dumbbell"                  "num_window"
## [11] "cvtd_timestamp30/11/2011 17:12" "magnet_belt_y"
## [13] "magnet_dumbbell_y"              "magnet_belt_z"
## [15] "cvtd_timestamp05/12/2011 14:23" "accel_forearm_x"
## [17] "cvtd_timestamp30/11/2011 17:11" "pitch_arm"
## [19] "total_accel_belt"               "yaw_belt"
```

Given the exercises were done by users in a specific series, the order (X) and time of the sensor relative to other test results became important in the model and yet would not be relevant to a new user performing activity for the first time. Similarly, the identity of the user would not be known unless the model is going to be retrained for each new user (i.e. each new user would have to perform each exercise correctly and performing each type of mistake which is impractical). So the following features were excluded from the model.

```
remove <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp", "full_t
trModel <- trAvail[,setdiff(names(trAvail),remove)]
```

The remaining features were the sensor readings taken from the various monitoring devices along with the target class. These are the relevant features available for the model:

```
names(trModel)
```

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

Note that these exclusions were based on the expected use of the model to provide insight and feedback for new users performing an exercise at some future time. This was the intent of the original research.

Modeling

This is a multiclass classification problem. As such there are a variety of algorithms that could be applied including logistic regression, decision trees, random forests and others. Given a multiclass classification problem with a large set of predictors, the random forest model was chosen for this application as it has typically performed best under these circumstances. Decision tree and logistic regression models were tested, but the results were significantly less predictive.

Using 500 trees (default) and including all available predictors, the random forest model achieved a .992 accuracy based on out of bag (oob) error using 27 predictors (mtry). However this model required several hours to run and included leakage. (not shown)

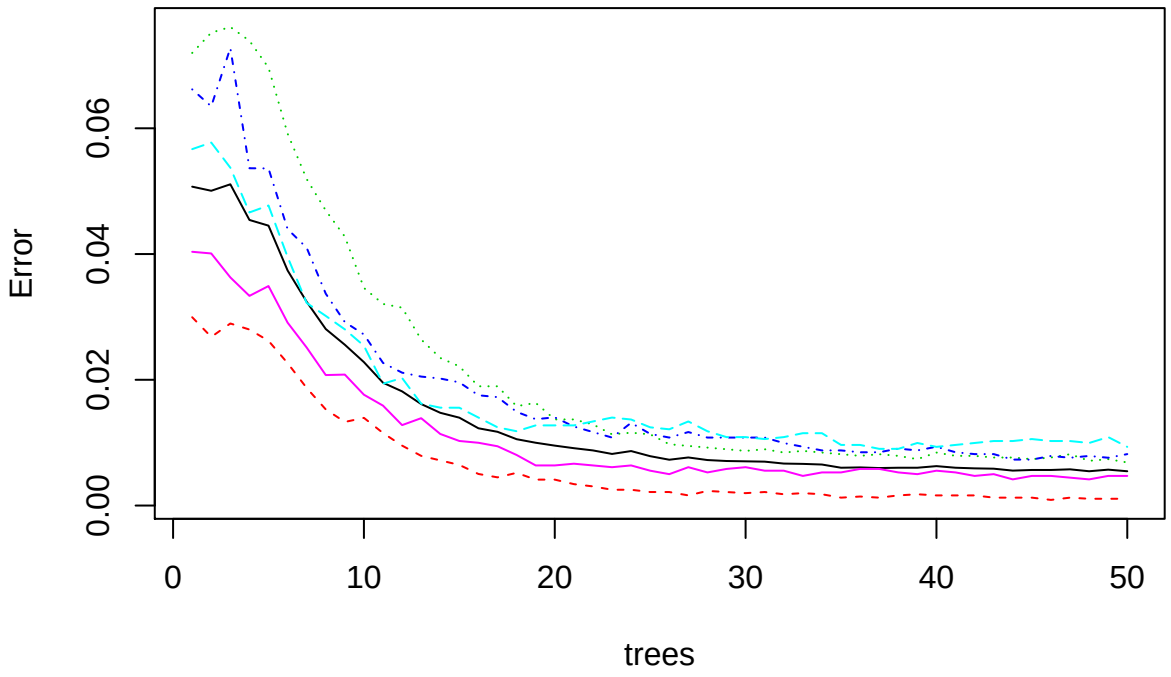
Leakage predictors were removed. The random forest cross validation (rfcf) was run (see below) to determine that a near .99 accuracy could be achieved with just 10 predictors.

```
# rfmt <- rfcv(trModel[,-53], trModel[,53])
rfmt$error.cv
```

```
##          52          26          13           6           3           1
## 0.003873204 0.004892468 0.007185812 0.038732035 0.107481398 0.594689634
```

The number of trees can be significantly reduced. The error rate stabilizes after 50 trees.

```
par(mfrow=c(1,1))
plot(mt$finalModel)
```



Subsequently the 10 most important predictors (shortlist) were chosen based on information gain/loss.

```
iOrder <- order(with(varImp(mt)$importance, pmax(A,B,C,D,E)),decreasing=TRUE)
shortlist <- append("classe",row.names(varImp(mt)$importance[iOrder,])[1:10])
shortlist
```

```
##  [1] "classe"            "magnet_dumbbell_y" "pitch_belt"
##  [4] "roll_belt"         "pitch_forearm"     "yaw_belt"
##  [7] "accel_forearm_x"   "magnet_dumbbell_z" "roll_forearm"
## [10] "gyros_belt_z"      "gyros_dumbbell_y"
```

The final model achieved a .989 accuracy based on oob error with just 50 trees and 2 predictors sampled with each split. This model was parsimonious, computed in a a few minutes and achieved very similar accuracy.

```
# mt10 <- train(classe ~ .,method="rf", data=trModel[,shortlist], importance = TRUE, ntree=50)
mt10
```

```
## Random Forest
##
## 19622 samples
##    10 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 19622, 19622, 19622, 19622, 19622, 19622, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
```

```
##    2    0.9893743  0.9865586  0.001246109  0.001575071
##    6    0.9872153  0.9838283  0.001489960  0.001884900
##   10    0.9803041  0.9750849  0.002949302  0.003734539
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Cross Validation

The random forest accuracy is calculated using out of bag error which inherently estimates test set accuracy. To further assess likely test set accuracy, an additional k-fold cross validation was conducted using 5 folds and the accuracy had a mean of .992. This is within the .95 confidence of of the accuracy estimate based on oob error.

```
folds <- createFolds(trModel$classe, k=5)
crossTrain <- function(fold) {train(classe ~ .,method="rf", data=trModel[-fold,shortlist], ntree=50)}
# crossModel <- lapply(folds, crossTrain)
crossValidate <- function(i) {
  confusionMatrix(predict(crossModel[[i]],newdata = trModel[folds[[i]],shortlist]),trModel[folds[[i]],"
}
# crossValidated <- lapply(1:length(folds),crossValidate)
crossAcc <- function(cV) {cV$overall[1]}
crossAccuracy <- sapply(crossValidated, crossAcc)
crossAccuracy
```

```
##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.9910828 0.9933741 0.9913376 0.9931193 0.9910805
```

```
mean(crossAccuracy)
```

```
## [1] 0.9919989
```

Hence there is high confidence that the model will be very accurate (accuracy $>$ .987 with 97.5% confidence) for the testing data since the testing data is from the same users performing the same exercises as the training data. The model accurately predicts whether one of these participants is performing the exercise correctly or incorrectly exactly as he did in the experiment.

Cross Validation of New Users/Exercises

The primary objective of the experiment is to detect when a user is performing an exercise incorrectly. It can be implied based on these objectives that the users of these devices would not necessarily be one of the 6 users who participated in the test nor that the the exercise would be executed in the same manner. The appropriate validation is whether the model can detect if a new user is performing the exercise in similar (but not identical) manner as the training classes.

Therefore, the model was further validated by leaving each participant out of the training set, training the model and testing against the data set from the participant that was left out. The accuracy results were much worse.

```
userFolds <- split(1:nrow(trAvail), trAvail$user_name)
userCrossTrain <- function(userFold) {train(classe ~ .,method="rf", data=trModel[-userFold,shortlist], r
# userCrossModel <- lapply(userFolds, userCrossTrain)
userCrossValidate <- function(i) {
  confusionMatrix(predict(userCrossModel[[i]], newdata = trModel[userFolds[[i]],shortlist]),
                  trModel[userFolds[[i]],"classe"])
}
# userCrossValidated <- lapply(1:length(users),userCrossValidate)
userCrossAcc <- function(ucV) {ucV$overall[1]}
userCrossAccuracy <- sapply(userCrossValidated, userCrossAcc)
names(userCrossAccuracy) = users
userCrossAccuracy
```

```
##  carlitos     pedro    adelmo   charles    eurico    jeremy
## 0.1788284 0.4742931 0.5517534 0.2814332 0.4215168 0.4275862
```

```
mean(userCrossAccuracy)
```

```
## [1] 0.3892352
```

The mean accuracy is .39 and in the case of carlitos, the accuracy is worse that the majority class prediction (Class = A: 28%). Clearly the model is inaccurate at predicting exercise errors for other users performing the exercise even when the user is attempting to perform the errors in a consistent manner as instructed. There are no test cases for users executing the exercise in any other manner than that specified, but it can be reasonably assumed that the model prediction accuracy would be worse.

Conclusion

The final random forest model accurately predicts whether one of these participants is performing the exercise correctly or incorrectly as specified in the experiment. The model was parsimonious using only 50 trees and sampling just 2 predictors at a time, and was able to achieve accuracy exceeding .985. There is high confidence that the test set will perform with similar accuracy based on cross validation.

However, validation of the model by leaving the data from 1 participant out of the training set and validating against the data from the participant left out showed very poor accuracy. Therefore, the model is inaccurate at predicting whether any other user performing these exercises are doing so correctly. This was the intent of the study, but was not included in the requirements for this project.

To achieve the objectives of the experiment, the model should incorporate the time series nature of the data and extract features from the patterns of data over the series of movements and exercises. This was beyond the scope of this project since the test data did not provide the necessary data to execute such a test.

Final Test Results

Below are the results from executing the model against the test data. These were all correctly classified.

```
# test execution
testResults <- data.frame(predict(mt10, test))
print(testResults)
```

```
##    predict.mt10..test.
## 1                    B
## 2                    A
## 3                    B
## 4                    A
## 5                    A
## 6                    E
## 7                    D
## 8                    B
## 9                    A
## 10                   A
## 11                   B
## 12                   C
## 13                   B
## 14                   A
## 15                   E
## 16                   E
## 17                   A
## 18                   B
## 19                   B
## 20                   B
```