Machine Learning project: Developing a Prediction Model

Peter Toyinbo

February 11, 2018

Executive Summary

The purpose of this project was to predict how well a weight lifting exercise is performed using secondary data from accelerometers.

Two classifiers were considered: Random Forest and Generalized Boosted Regression Modeling. Both models showed very high prediction accuracy on a test sample. The final choice, Random Forest, performed better with a higher 99% accuracy and lower expected out-of-sample error.

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, the goal is 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).

General Approach

1. Prepare the labeled data (includes target variable)
2. Split the labeled data into Training and Testing
3. Use the training set for feature selection via cross-validation after pre-processing
4. Apply prediction model to testing set once and obtain its out-of-sample error rate
5. Apply prediction model to the unlabeled data (with no target variable) to classify 20 different test cases

Data Preparation

Data description

Six young health male participants aged between 20-28 years, with little weight lifting experience, were asked to perform one set of 10 repetitions of a weight lifting excercise under the supervision of an experienced weight lifter.

The exercice, the Unilateral Dumbbell Biceps Curl, was performed repeatedly in five different 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.

Source: http://groupware.les.inf.puc-rio.br/har#weight_lifting_exercises#ixzz56AlaSYsJ

File retrieval and reading into R:

Set up the urls from where to download the two files

```
urlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
urlTest <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

Download the files *pml-training.csv* and *pml-testing.csv* and read them into R as *PMLtrain* and *PMLtest* respectively.

```
pml_training <- "pml-training.csv"

if (file.exists(pml_training)) {
    PMLtrain <- read_csv(pml_training)
} else {
    download.file(urlTrain, pml_training)
    PMLtrain <- read_csv(pml_training)
}


pml_testing <- "pml-testing.csv"

if (file.exists(pml_testing)) {
    PMLtest <- read_csv(pml_testing)
} else {
    download.file(urlTest, pml_testing)
    PMLtest <- read_csv(pml_testing)
}
```

Missing name for column 1 was automatically filled in as X1.

Review the datasets starting with the *PMLtest*.

```
# str(PMLtest) # Not run due to space consideration
dim(PMLtest)
```

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

The 20 x 160 *PMLtest* dataset contains 20 test cases for which the classification status (how well a weight lifting exercise is performed) is not available. There are 159 feature variables ("problem_id" in column 160 is row index). This dataset will be put aside untill the very end when the final developed model will be applied on it.

Next, review the *PMLtrain* dataset

```
# str(PMLtrain) # Not run due to space consideration

dim(PMLtrain)
```

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

```
names(PMLtrain)
```

```
##   [1] "X1"                       "user_name"
```

```
##   [3] "raw_timestamp_part_1"     "raw_timestamp_part_2"
##   [5] "cvtd_timestamp"           "new_window"
##   [7] "num_window"               "roll_belt"
##   [9] "pitch_belt"               "yaw_belt"
##  [11] "total_accel_belt"         "kurtosis_roll_belt"
##  [13] "kurtosis_picth_belt"      "kurtosis_yaw_belt"
##  [15] "skewness_roll_belt"       "skewness_roll_belt.1"
##  [17] "skewness_yaw_belt"        "max_roll_belt"
##  [19] "max_picth_belt"           "max_yaw_belt"
##  [21] "min_roll_belt"            "min_pitch_belt"
##  [23] "min_yaw_belt"             "amplitude_roll_belt"
##  [25] "amplitude_pitch_belt"     "amplitude_yaw_belt"
##  [27] "var_total_accel_belt"     "avg_roll_belt"
##  [29] "stddev_roll_belt"         "var_roll_belt"
##  [31] "avg_pitch_belt"           "stddev_pitch_belt"
##  [33] "var_pitch_belt"           "avg_yaw_belt"
##  [35] "stddev_yaw_belt"          "var_yaw_belt"
##  [37] "gyros_belt_x"             "gyros_belt_y"
##  [39] "gyros_belt_z"             "accel_belt_x"
##  [41] "accel_belt_y"             "accel_belt_z"
##  [43] "magnet_belt_x"            "magnet_belt_y"
##  [45] "magnet_belt_z"            "roll_arm"
##  [47] "pitch_arm"                "yaw_arm"
##  [49] "total_accel_arm"          "var_accel_arm"
##  [51] "avg_roll_arm"             "stddev_roll_arm"
##  [53] "var_roll_arm"             "avg_pitch_arm"
##  [55] "stddev_pitch_arm"         "var_pitch_arm"
##  [57] "avg_yaw_arm"              "stddev_yaw_arm"
##  [59] "var_yaw_arm"              "gyros_arm_x"
##  [61] "gyros_arm_y"              "gyros_arm_z"
##  [63] "accel_arm_x"              "accel_arm_y"
##  [65] "accel_arm_z"              "magnet_arm_x"
##  [67] "magnet_arm_y"             "magnet_arm_z"
##  [69] "kurtosis_roll_arm"        "kurtosis_picth_arm"
##  [71] "kurtosis_yaw_arm"         "skewness_roll_arm"
##  [73] "skewness_pitch_arm"       "skewness_yaw_arm"
##  [75] "max_roll_arm"             "max_picth_arm"
##  [77] "max_yaw_arm"              "min_roll_arm"
##  [79] "min_pitch_arm"            "min_yaw_arm"
##  [81] "amplitude_roll_arm"       "amplitude_pitch_arm"
##  [83] "amplitude_yaw_arm"        "roll_dumbbell"
##  [85] "pitch_dumbbell"           "yaw_dumbbell"
##  [87] "kurtosis_roll_dumbbell"   "kurtosis_picth_dumbbell"
##  [89] "kurtosis_yaw_dumbbell"    "skewness_roll_dumbbell"
##  [91] "skewness_pitch_dumbbell"  "skewness_yaw_dumbbell"
##  [93] "max_roll_dumbbell"        "max_picth_dumbbell"
##  [95] "max_yaw_dumbbell"         "min_roll_dumbbell"
##  [97] "min_pitch_dumbbell"       "min_yaw_dumbbell"
##  [99] "amplitude_roll_dumbbell"  "amplitude_pitch_dumbbell"
## [101] "amplitude_yaw_dumbbell"   "total_accel_dumbbell"
## [103] "var_accel_dumbbell"       "avg_roll_dumbbell"
## [105] "stddev_roll_dumbbell"     "var_roll_dumbbell"
## [107] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
## [109] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
```

```
## [111] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
## [113] "gyros_dumbbell_x"         "gyros_dumbbell_y"
## [115] "gyros_dumbbell_z"         "accel_dumbbell_x"
## [117] "accel_dumbbell_y"         "accel_dumbbell_z"
## [119] "magnet_dumbbell_x"        "magnet_dumbbell_y"
## [121] "magnet_dumbbell_z"        "roll_forearm"
## [123] "pitch_forearm"            "yaw_forearm"
## [125] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
## [127] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
## [129] "skewness_pitch_forearm"   "skewness_yaw_forearm"
## [131] "max_roll_forearm"         "max_picth_forearm"
## [133] "max_yaw_forearm"          "min_roll_forearm"
## [135] "min_pitch_forearm"        "min_yaw_forearm"
## [137] "amplitude_roll_forearm"   "amplitude_pitch_forearm"
## [139] "amplitude_yaw_forearm"    "total_accel_forearm"
## [141] "var_accel_forearm"        "avg_roll_forearm"
## [143] "stddev_roll_forearm"      "var_roll_forearm"
## [145] "avg_pitch_forearm"        "stddev_pitch_forearm"
## [147] "var_pitch_forearm"        "avg_yaw_forearm"
## [149] "stddev_yaw_forearm"       "var_yaw_forearm"
## [151] "gyros_forearm_x"          "gyros_forearm_y"
## [153] "gyros_forearm_z"          "accel_forearm_x"
## [155] "accel_forearm_y"          "accel_forearm_z"
## [157] "magnet_forearm_x"         "magnet_forearm_y"
## [159] "magnet_forearm_z"         "classe"
```

The *PMLtrain* dataset consists of 19622 observations of the same 159 independent variables (features) plus the target variable *classe* with 5-category levels. This dataset will be used to develop the predictive model.

The first seven columns are not directly relevant to our prediction in the current study and are dropped. One of these, "X1", is the row index but the column name is missing from the raw data.

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

Missing data

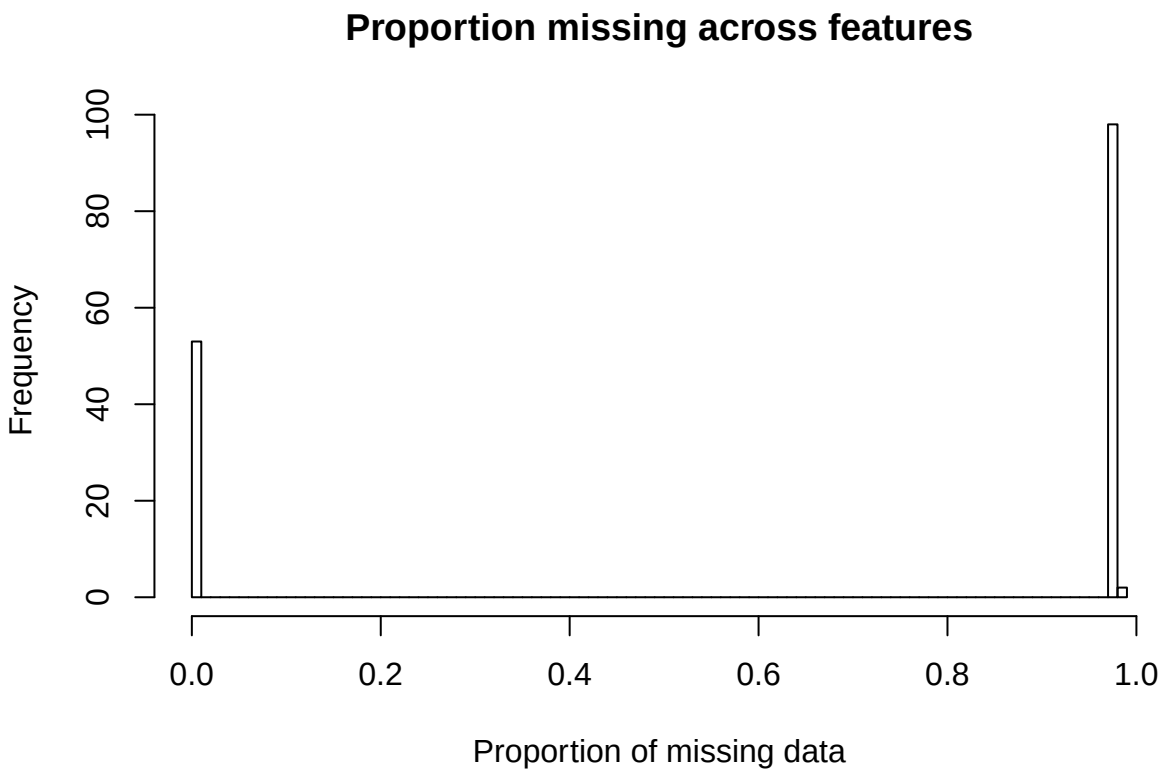
Dataset was checked for missing data.

```
mean(is.na(PMLtrain))
```

```
## [1] 0.6401328
```

There was 64% missing overall in the data. Next the missing pattern was explored using a histogram.

```
propNA <- colSums(is.na(PMLtrain))/nrow(PMLtrain)
hist(propNA, nclass = 100, main = "Proportion missing across features", xlab = "Proportion of missing da
```



The features reveal two patterns: either have close to zero missing or in the upper 90% missing.

Exclude the variables with missing values from the analysis.

```
hiNAvar <- which(propNA > 0)

PMLtrain <- PMLtrain[, -hiNAvar]

str(PMLtrain)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    19622 obs. of  50 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 ...
##  $ 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 ...
##  $ classe              : chr  "A" "A" "A" "A" ...
```

The final number of the variables to be used in analysis has reduced to 50 where the 50th column is the target variable.

Model building

The cleaned *PMLtrain* dataset is now ready to be used for building the predition model.

Split into *training* (to train/develop the model) and *testing* (to test the model and compute out-of-sample error). Use split ratio 60:40.

```
inTrain <- createDataPartition(PMLtrain$classe, p = 0.6, list = FALSE)
training <- PMLtrain[inTrain, ]
testing <- PMLtrain[-inTrain, ]

dim(training)
```

```
## [1] 11776    50
```

```
dim(testing)
```

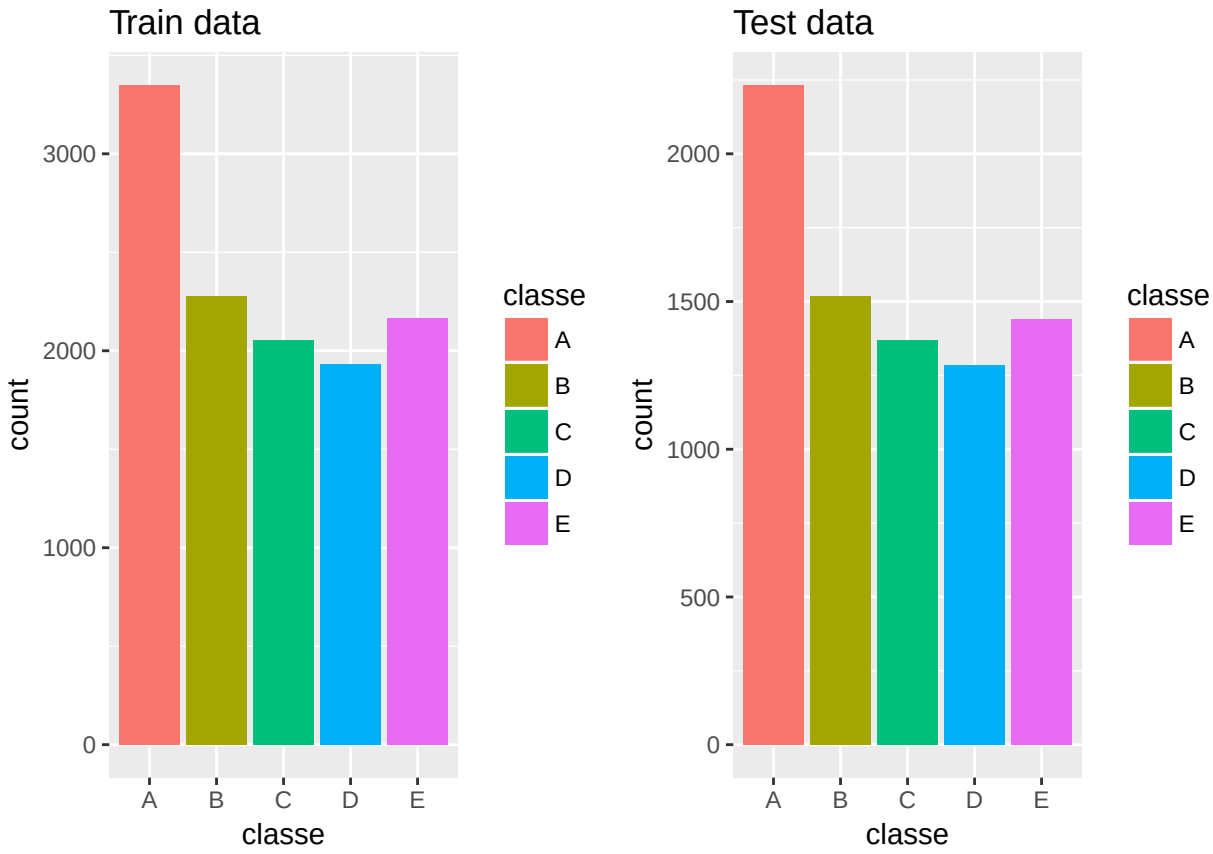
```
## [1] 7846   50
```

Check to ensure comparable distributions of the target variable *classe* between both training and testing samples.

```
g1 <- ggplot(training, aes(x = classe, fill = classe)) + geom_bar() + labs(title = "Train data")

g2 <- ggplot(testing, aes(x = classe, fill = classe)) + geom_bar() + labs(title = "Test data")

grid.arrange(g1, g2, ncol = 2)
```



The variable *classe* shows very similar distribution between both training and testing sets after split.

Next, the specification of the candidate models will include pre-processing and cross-validation parameters as follows.

Data pre-processing

To enhance numerical stability of the models the *train* set will be pre-processed by performing centering and scaling of the predictor variables. Also the "nzv" filter will be applied to exclude zero- or near zero-variance predictors.

Later, when the trained model is applied to *test* data to predict a new sample, the *train* set pre-process information will be used. That is, both training and testing data will be pre-processed the same way.

Cross-validation

To minimize overfitting and subsequently minize out of sample errors, the tuning parameters for the modeling procedure were set to perform a 5-fold cross-validation.

Specify the model parameters for cross-validation.

```
ctrl <- trainControl(classProbs = TRUE, savePredictions = TRUE, method = "cv",
    number = 5)
```

Candidate models

Two of the commonly used predictive modelling and machine learning technique for multi-classification problems are Random Forest (RF), and Generalized Boosted Regression Modeling (GBM) with a multinomial method.

Train the RF on the *train* data and display model fit results

```
# RF

set.seed(5555)
trainMod.rf <- train(classe ~ ., data = training, method = "rf", trControl = ctrl,
    preProcess = c("center", "scale", "nzv"))

print(trainMod.rf)
```

```
## Random Forest
##
## 11776 samples
##    49 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered (49), scaled (49)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9421, 9420, 9422, 9421, 9420
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9867525  0.9832397
##   25    0.9887056  0.9857115
##   49    0.9840349  0.9798022
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 25.
```

Train the GBM on the *train* data and display model fit results.

```
# GBM

set.seed(5555)
trainMod.gbm <- train(classe ~ ., data = training, method = "gbm", trControl = ctrl,
    verbose = FALSE, preProcess = c("center", "scale", "nzv"))

print(trainMod.gbm)
```

```
## Stochastic Gradient Boosting
##
## 11776 samples
##    49 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
```

```
## Pre-processing: centered (49), scaled (49)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9421, 9420, 9422, 9421, 9420
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7445648  0.6758944
##   1                  100      0.8137738  0.7641604
##   1                  150      0.8490156  0.8089429
##   2                   50      0.8434110  0.8017136
##   2                  100      0.8996272  0.8729657
##   2                  150      0.9228089  0.9023229
##   3                   50      0.8928325  0.8643309
##   3                  100      0.9352912  0.9181220
##   3                  150      0.9512558  0.9383302
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

Apply the two trained models separately on the *test* data to test their prediction.

```
set.seed(5555)
pred.rf <- predict(trainMod.rf, testing, class = "class")

set.seed(5555)
pred.gbm <- predict(trainMod.gbm, testing, class = "class")
```

Next, compare their prediction performance.

Display the confusion matrix for RF

```
cM.rf <- confusionMatrix(pred.rf, testing$classe)
cM.rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2227   11    0    0    0
##          B    4 1503    5    0    1
##          C    0    4 1359   12    5
##          D    0    0    4 1272    2
##          E    1    0    0    2 1434
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9915, 0.9952)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9918
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9901   0.9934   0.9891   0.9945
## Specificity            0.9980   0.9984   0.9968   0.9991   0.9995
## Pos Pred Value         0.9951   0.9934   0.9848   0.9953   0.9979
## Neg Pred Value         0.9991   0.9976   0.9986   0.9979   0.9988
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2838   0.1916   0.1732   0.1621   0.1828
## Detection Prevalence   0.2852   0.1928   0.1759   0.1629   0.1832
## Balanced Accuracy      0.9979   0.9943   0.9951   0.9941   0.9970
```

Display the confusion matrix for GBM

```
cM.gbm <- confusionMatrix(pred.gbm, testing$classe)
cM.gbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2183   65    1    4    2
##          B   21 1419   37    3   15
##          C   12   32 1315   51   27
##          D    9    1   12 1224   18
##          E    7    1    3    4 1380
##
## Overall Statistics
##
##                Accuracy : 0.9586
##                  95% CI : (0.9539, 0.9629)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9476
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9780   0.9348   0.9613   0.9518   0.9570
## Specificity            0.9872   0.9880   0.9812   0.9939   0.9977
## Pos Pred Value         0.9681   0.9492   0.9151   0.9684   0.9892
## Neg Pred Value         0.9912   0.9844   0.9917   0.9906   0.9904
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2782   0.1809   0.1676   0.1560   0.1759
## Detection Prevalence   0.2874   0.1905   0.1832   0.1611   0.1778
## Balanced Accuracy      0.9826   0.9614   0.9712   0.9728   0.9773
```

The overall prediction accuracy is greater for the RF model (99%) compared to GBM (95%). The balanced accuracy statistics across the predicted classes A to E are also better for RF model.

Out-of-sample error

Compute the expected out-of-sample error rate for each model as: 1-Accuracy

```
DF <- data.frame(rbind(cM.rf$overall, cM.gbm$overall))
DF$Error <- 1 - DF$Accuracy
row.names(DF) <- c("RF", "GBM")
DF[, c(1, 2, 8)]
```

```
##      Accuracy     Kappa       Error
## RF  0.9934999 0.9917774 0.006500127
## GBM 0.9585776 0.9475904 0.041422381
```

The out-of-sample error rate of the Random Forest classifier is about five times smaller than GBM's.

The Random Forest classifier is chosen as the final model.

Predicting new cases

Use the final prediction model, Random Forest, to classfy each of 20 different test cases in the unlabled *PMLtest* data into one of A, B, C, D, and E activity classes.

```
predict(trainMod.rf, PMLtest)
```

```
##  [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
```

Reference

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Read more: http://groupware.les.inf.puc-rio.br/har#weight_lifting_exercises#ixzz56AmUAJMg