Machine Learning Assignment

Danielle Boucher

3/25/2017

Predicting Proper Exercise: Machine Learning Final Project

Introduction and Pre-Processing

This report uses the Weight Lifting Data Set (WLE) and attempts to create a model that can predict *how well* a user is performing dumbbell exercises. Full details of the data set can be found here (http://groupware.les.inf.puc-rio.br/har#dataset).

First, download and clean the WLE dataset. The original dataset has 160 columns - it is useful to remove those with NA values, and those that are not valuable for prediction (timestamps, entry IDs, and user data). We're left with 52 predictors, and the *classe* outcome variable.

The original data comes with a training set, and a test set of 20 entries. Since the provided test set does not include the true *classe* variable for validation, we will subdivide the training set into a training (70%) and validation (30%) in order to assess out-of-sample error and model accuracy.

```
# Set up parallel processing for speed
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)

# Load WLE data sets
traindata <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.
csv", na.strings=c("NA", "#DIV/0!"))
testing <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.cs
v", na.strings=c("NA", "#DIV/0!"))

# Remove unneeded columns
col.na <- colSums(sapply(traindata, is.na))
col.na.test <- colSums(sapply(testing, is.na))
traindata <- traindata[,col.na == 0 & col.na.test == 0]
traindata <- traindata[,-1:-7]
testing <- testing[,col.na == 0 & col.na.test == 0]
testing <- testing[,-1:-7]

# Create a validation data set
set.seed(6497)
inTrain <- createDataPartition(y = traindata$classe, p=0.7, list=FALSE)
training <- traindata[inTrain,]
validation <- traindata[-inTrain,]
```

```
dim(training); dim(validation); dim(testing)
```

```
## [1] 13737    53
```

```
## [1] 5885   53
```

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

```
str(training)
```

```
## 'data.frame':    13737 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.42 1.48 1.42 1.45 1.42 1.42 1.48 1.51 1.55 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.09 8.18 8.2 8.21 8.15 8.12 8.08 ...
##  $ 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 0.02 0.02 0.03 0.02 0.02 0 0 0 ...
##  $ gyros_belt_y        : num  0 0 0.02 0 0 0 0 0 0 0.02 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.02 -0.02 0 -0.02 0 -0.02 0 ...
##  $ accel_belt_x        : int  -21 -20 -21 -22 -21 -22 -22 -21 -21 -21 ...
##  $ accel_belt_y        : int  4 5 2 3 2 4 4 4 4 5 ...
##  $ accel_belt_z        : int  22 23 24 21 23 21 21 23 22 21 ...
##  $ magnet_belt_x       : int  -3 -2 -6 -4 -5 -3 -8 0 -6 1 ...
##  $ magnet_belt_y       : int  599 600 600 599 596 606 598 592 598 600 ...
##  $ magnet_belt_z       : int  -313 -305 -302 -311 -317 -309 -310 -305 -317 -316 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -129 -129 -129 ...
##  $ pitch_arm           : num  22.5 22.5 22.1 21.9 21.5 21.4 21.4 21.3 21.3 21.2 ...
##  $ 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 0 0.02 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.03 -0.03 -0.03 -0.02 0 0 0 -0.02 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 0 0 0 -0.02 -0.03 -0.03 -0.02 -0.03 ...
##  $ accel_arm_x         : int  -288 -289 -289 -289 -290 -287 -288 -289 -289 -288 ...
##  $ accel_arm_y         : int  109 110 111 111 110 111 111 109 110 108 ...
##  $ accel_arm_z         : int  -123 -126 -123 -125 -123 -124 -124 -121 -122 -124 ...
##  $ magnet_arm_x        : int  -368 -368 -374 -373 -366 -372 -371 -367 -371 -373 ...
##  $ magnet_arm_y        : int  337 344 337 336 339 338 331 340 337 336 ...
##  $ magnet_arm_z        : int  516 513 506 509 509 509 523 509 512 510 ...
##  $ roll_dumbbell       : num  13.1 12.9 13.4 13.1 13.1 ...
##  $ pitch_dumbbell      : num  -70.5 -70.3 -70.4 -70.2 -70.6 ...
##  $ yaw_dumbbell        : num  -84.9 -85.1 -84.9 -85.1 -84.7 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 36 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0.02 0 0 0.02 ...
##  $ 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 0 -0.02 -0.02 0 0 -0.02 ...
##  $ accel_dumbbell_x    : int  -234 -232 -233 -232 -233 -234 -234 -233 -233 -231 ...
##  $ accel_dumbbell_y    : int  47 46 48 47 47 48 48 48 47 47 ...
##  $ accel_dumbbell_z    : int  -271 -270 -270 -270 -269 -269 -268 -271 -272 -268 ...
##  $ magnet_dumbbell_x   : int  -559 -561 -554 -551 -564 -552 -554 -554 -551 -557 ...
##  $ magnet_dumbbell_y   : int  293 298 292 295 299 302 295 297 296 292 ...
##  $ magnet_dumbbell_z   : num  -65 -63 -68 -70 -64 -69 -68 -73 -56 -62 ...
##  $ roll_forearm        : num  28.4 28.3 28 27.9 27.6 27.2 27.2 27.1 27.1 27 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.8 -63.9 -63.9 -64 -64 -64
 ...
##  $ yaw_forearm         : num  -153 -152 -152 -152 -152 -151 -151 -151 -151 -151 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.03 0.02 0.02 0.02 0 0 0.02 0.02 0.02 ...
##  $ gyros_forearm_y     : num  0 -0.02 0 0 -0.02 0 -0.02 0 -0.02 0 ...
##  $ gyros_forearm_z     : num  -0.02 0 -0.02 -0.02 -0.02 -0.03 -0.03 0 0 -0.02 ...
##  $ accel_forearm_x     : int  192 196 189 195 193 193 193 194 192 192 ...
##  $ accel_forearm_y     : int  203 204 206 205 205 205 202 204 204 206 ...
##  $ accel_forearm_z     : int  -215 -213 -214 -215 -214 -215 -214 -215 -213 -216 ...
```

```
##  $ magnet_forearm_x    : int  -17 -18 -17 -18 -17 -15 -14 -13 -13 -16 ...
##  $ magnet_forearm_y    : num  654 658 655 659 657 655 659 656 653 653 ...
##  $ magnet_forearm_z    : num  476 469 473 470 465 472 478 471 481 472 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1
 ...
```

Model Creation

Using the *caret* package, we begin by creating a random forest model.

```
# Generate a random forest model on the Testing set:
fitControl <- trainControl(method = "cv", number = 10, allowParallel = TRUE)
randomforestmodel <- train(classe~., method="rf", data=training, trControl = fitControl)
# Make classe predictions using the Validation data set and generate the confusion matri
x.
rfprediction <- predict(randomforestmodel, validation)
confusionMatrix(rfprediction, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672   11    0    0    0
##          B    1 1124    3    0    1
##          C    1    3 1020    9    3
##          D    0    0    3  954    5
##          E    0    1    0    1 1073
##
## Overall Statistics
##
##                Accuracy : 0.9929
##                  95% CI : (0.9904, 0.9949)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.991
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9868   0.9942   0.9896   0.9917
## Specificity            0.9974   0.9989   0.9967   0.9984   0.9996
## Pos Pred Value         0.9935   0.9956   0.9846   0.9917   0.9981
## Neg Pred Value         0.9995   0.9968   0.9988   0.9980   0.9981
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1910   0.1733   0.1621   0.1823
## Detection Prevalence   0.2860   0.1918   0.1760   0.1635   0.1827
## Balanced Accuracy      0.9981   0.9929   0.9954   0.9940   0.9956
```

The confusion matrix reveals that the model accuracy of this random forest method is over 99%. In fact, in the case of identifying Class A (dumbbell exercises done correctly), only 2 of 1674 examples were misclassified, yielding an accuracy of 99.88%. We therefore conclude that the random forest model is sufficient for predicting how well a user is performing these dumbbell exercises, and move on to evaluate against the testing set.

Model Assessment

A highly accurate model has been created, with an estimated out of sample error of 0.8%. Next we will predict and display the results of the twenty test cases in the testing set.

```
rfpredictionfinal <- predict(randomforestmodel, testing)
rfpredictionfinal
```

```
##  [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
```