PML-Assignment-Writeup

Jonathan Isernhagen

December 26, 2015

Synopsis

We have been asked to analyze a data set of metrics related to the performance of weight-lifting physical exercises, with each corresponding to: A) a properly-performed exercise, or; B-E) an exercise improperly-performed in a specific way. Our task is to use any and all useful metrics to ascertain which category "bucket" of correct or incorrect exercise each observation belongs to, and predict the A-E category for a set of twenty new observations with the highest possible degree of accuracy. Our preference in this context is for accuracy over interpretability or scalability.

Executive Summary

After deleting all variables which contained few or no values in order to concentrate the algorithms' efforts on useful data of manageable size, and creating a simple random forest model with no preprocessing of the data and a simple 60/40 split of the data into testing/training (with no cross-validation, boosting or bootstrapping) we discovered a model which gave us very high accuracy on both the training and testing data sets and perfect (100%) accuracy on the validation set.

Analyses

Exploratory Analysis:

We first installed the caret package, ggplot2 for charting and dplyr for data manipulation, then imported the "training" (which we labeled as training.and.testing) and "testing" (which we labeled "validation") data sets.

```
require(caret); require(randomForest); require(ggplot2); require(dplyr)
training.and.testing<-read.csv("pml-training.csv")
validation<-read.csv("pml-testing.csv")
```

....and used str to observe their characteristics (Appendix A). The training.and.testing set includes 19,622 observations of 160 variables, qualifying it as a "large" data set. It was apparent that many columns of both data sets contained data for only a small minority of observations. Attempting to fit a random forest model against the un-edited data set took > 1 hour, so we decided to delete all non-exercise-performance-related (e.g. timestamp) and all sparsely-populated columns, and then make other changes necessary to align the two data sets:

```
training.and.testing.cut<-select(training.and.testing, new_window:total_accel_belt,
gyros_belt_x:total_accel_arm, gyros_arm_x:magnet_arm_z, roll_dumbbell:yaw_dumbbell,
total_accel_dumbbell, gyros_dumbbell_x:yaw_forearm, total_accel_forearm,
gyros_forearm_x:magnet_forearm_z, classe)
validation.cut<-select(validation, new_window:total_accel_belt,
gyros_belt_x:total_accel_arm, gyros_arm_x:magnet_arm_z, roll_dumbbell:yaw_dumbbell,
total_accel_dumbbell, gyros_dumbbell_x:yaw_forearm, total_accel_forearm,
gyros_forearm_x:magnet_forearm_z)
validation.cut$magnet_forearm_y<-as.numeric(validation.cut$magnet_forearm_y)
validation.cut$magnet_forearm_z<-as.numeric(validation.cut$magnet_forearm_z)
levels(validation.cut$new_window) <- c("no", "yes")
```

We then divided the "training.and.testing" data set into "training" and "testing" partitions.

```
inTrain<-createDataPartition(y=training.and.testing.cut$classe, p = 0.6, list = FALSE)
training<-training.and.testing.cut[inTrain,]
testing<-training.and.testing.cut[-inTrain,]
```

Model #1: random forest, no preprocessing, 60/40 split: Our first model was a simple random forest. Evoking the random forest from within caret (modelFit<-train(classe ~ ., data = training, method="rf")) took too long, so we did it directly.

```
set.seed(1)
modelFit<-randomForest(classe ~ ., data = training)
modelFit
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.42%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3348    0    0    0    0 0.000000000
## B    6 2271    2    0    0 0.003510312
## C    0   11 2041    2    0 0.006329114
## D    0    0   21 1906    3 0.012435233
## E    0    0    0    5 2160 0.002309469
```

Diagnostics: The model's error OOB error rate is 0.35%, which is much lower than we expected. When applied to the testing data set, the confusion matrix appears as follows:

```
predictions<-predict(modelFit, newdata=testing)
confusionMatrix(predictions, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    2    0    0    0
##          B    0 1515    6    0    0
##          C    0    1 1359   14    0
##          D    0    0    3 1271    4
##          E    0    0    0    1 1438
##
## Overall Statistics
##
##                Accuracy : 0.996
##                  95% CI : (0.9944, 0.9973)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.995
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9980   0.9934   0.9883   0.9972
## Specificity            0.9996   0.9991   0.9977   0.9989   0.9998
## Pos Pred Value         0.9991   0.9961   0.9891   0.9945   0.9993
## Neg Pred Value         1.0000   0.9995   0.9986   0.9977   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1931   0.1732   0.1620   0.1833
## Detection Prevalence   0.2847   0.1939   0.1751   0.1629   0.1834
## Balanced Accuracy      0.9998   0.9985   0.9956   0.9936   0.9985
```

...which showed an encouragingly very low error rate (we expected the testing error rate to be significantly higher than the training error rate because of overfitting), and gave us the confidence to proceed directly to the validation phase and use one of our two submission "bullets" to see if the model was in fact highly predictive.

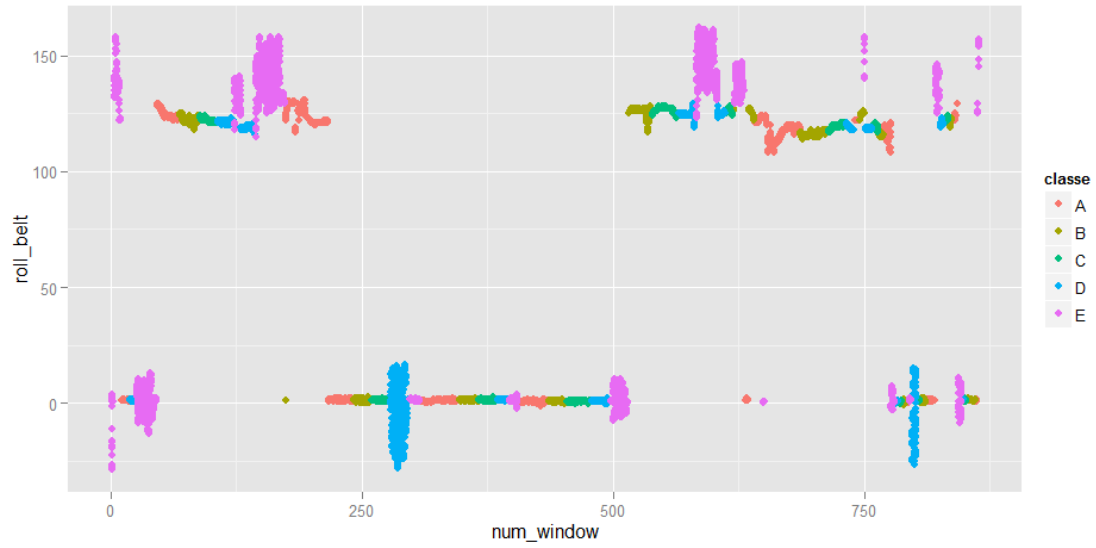
The predictions achieved when applying the model to the validation set were as follows:

```
predictions<-predict(modelFit, newdata=validation.cut)
predictions
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

...which turned out to be 100% correct. At this point we decided not to construct other models, but rather to try to figure out what is happening within this model that makes it so effective. We used varImp to ascertain which variables were most important to the model we'd built (Appendix B), and then charted the two most important variables against each other while coloring by classe

```
qplot(num_window,roll_belt, colour = classe, data=training)
```



It is readily apparent why these two predictors are so valuable: the classe categories stand apart from each other in sharp relief when they are charted against one another, much more sharply than in any of the charts used in the lecture examples in class.

Conclusions

We came to the exercise prepared to test a wide variety of models and methods to achieve high predictive accuracy, but the very first model we tried, with minimal, common-sense shaping of the data, provided a model sufficiently accurate for us to score 100% of the validation observations successfully, so we stopped the exercise and are submitting these findings.

Appendices:

Appendix A: str(Training) and str(Validation)

```
str(training)
```

```
## 'data.frame':    11776 obs. of  55 variables:
##  $ 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 1.42 1.45 1.43 1.42 1.45 1.48 1.59 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.09 8.18 8.18 8.21 8.2 8.15 8.07 ...
##  $ 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 0.03 0.02 0.02 0 0 0.02 ...
##  $ gyros_belt_y        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 0 0 -0.02 ...
##  $ accel_belt_x        : int  -21 -22 -20 -22 -21 -22 -22 -21 -21 -22 ...
##  $ accel_belt_y        : int  4 4 5 3 2 2 4 2 4 5 ...
##  $ accel_belt_z        : int  22 22 23 21 23 23 21 22 23 22 ...
##  $ magnet_belt_x       : int  -3 -7 -2 -4 -5 -2 -8 -1 0 -1 ...
##  $ magnet_belt_y       : int  599 608 600 599 596 602 598 597 592 604 ...
##  $ magnet_belt_z       : int  -313 -311 -305 -311 -317 -319 -310 -310 -305 -314 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -129 -129 -129 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 21.9 21.5 21.5 21.4 21.4 21.3 21.1 ...
##  $ 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 0.02 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 0 0 0 -0.02 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0 0 0 -0.03 -0.03 -0.03 -0.02 ...
##  $ accel_arm_x         : int  -288 -290 -289 -289 -290 -288 -288 -289 -289 -289 ...
##  $ accel_arm_y         : int  109 110 110 111 110 111 111 111 109 109 ...
##  $ accel_arm_z         : int  -123 -125 -126 -125 -123 -123 -124 -124 -121 -125 ...
##  $ magnet_arm_x        : int  -368 -369 -368 -373 -366 -363 -371 -374 -367 -373 ...
##  $ magnet_arm_y        : int  337 337 344 336 339 343 331 342 340 335 ...
##  $ magnet_arm_z        : int  516 513 513 509 509 520 523 510 509 514 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.1 13.1 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.2 -70.6 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -85.1 -84.7 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0.02 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 0 0 -0.02 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -233 -234 -234 -233 -234 ...
##  $ accel_dumbbell_y    : int  47 47 46 47 47 47 48 47 48 46 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -270 -269 -270 -268 -270 -271 -272 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -551 -564 -554 -554 -554 -554 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 295 299 291 295 294 297 302 ...
```

```
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -70 -64 -65 -68 -63 -73 -66 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 27.9 27.6 27.5 27.2 27.2 27.1 26.9 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.9 -63.9 -64 -64
...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -151 -151 -151 -151 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0 0 0.02 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 0 -0.02 0.02 -0.02 -0.02 0 -0.02 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 -0.02 -0.02 -0.03 -0.03 -0.02 0 0 ...
##  $ accel_forearm_x     : int  192 192 196 195 193 191 193 192 194 193 ...
##  $ accel_forearm_y     : int  203 203 204 205 205 203 202 201 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -215 -214 -215 -214 -214 -215 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -18 -17 -11 -14 -16 -13 -9 ...
##  $ magnet_forearm_y    : num  654 661 658 659 657 657 659 656 656 657 ...
##  $ magnet_forearm_z    : num  476 473 469 470 465 478 478 472 471 480 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1
...
```

```
str(testing)
```

```
## 'data.frame':    7846 obs. of  55 variables:
##  $ new_window          : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window          : int  12 12 12 12 12 12 12 12 12 12 ...
##  $ roll_belt           : num  1.48 1.48 1.45 1.42 1.43 1.45 1.42 1.51 1.55 1.57 ...
##  $ pitch_belt          : num  8.05 8.07 8.06 8.13 8.16 8.17 8.2 8.12 8.08 8.06 ...
##  $ 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  0.02 0.02 0.02 0.02 0.02 0.03 0.02 0 0 0 ...
##  $ gyros_belt_y        : num  0 0.02 0 0 0 0 0 0 0.02 0 ...
##  $ gyros_belt_z        : num  -0.03 -0.02 -0.02 -0.02 -0.02 0 0 -0.02 0 -0.02 ...
##  $ accel_belt_x        : int  -22 -21 -21 -22 -20 -21 -22 -21 -21 -20 ...
##  $ accel_belt_y        : int  3 2 4 4 2 4 4 4 5 5 ...
##  $ accel_belt_z        : int  21 24 21 21 24 22 21 22 21 21 ...
##  $ magnet_belt_x       : int  -6 -6 0 -2 1 -3 -3 -6 1 -3 ...
##  $ magnet_belt_y       : int  604 600 603 603 602 609 606 598 600 603 ...
##  $ magnet_belt_z       : int  -310 -302 -312 -313 -312 -308 -309 -317 -316 -313 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -129 -129 -129 ...
##  $ pitch_arm           : num  22.1 22.1 22 21.8 21.7 21.6 21.4 21.3 21.2 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  0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 -0.02 0 -0.02 -0.02
...
##  $ gyros_arm_z         : num  0.02 0 0 0 -0.02 -0.02 -0.02 -0.02 -0.03 -0.02 ...
##  $ accel_arm_x         : int  -289 -289 -289 -289 -288 -288 -287 -289 -288 -289 ...
##  $ accel_arm_y         : int  111 111 111 111 109 110 111 110 108 109 ...
##  $ accel_arm_z         : int  -123 -123 -122 -124 -122 -124 -124 -122 -124 -122 ...
##  $ magnet_arm_x        : int  -372 -374 -369 -372 -369 -376 -372 -371 -373 -369 ...
##  $ magnet_arm_y        : int  344 337 342 338 341 334 338 337 336 340 ...
##  $ magnet_arm_z        : int  512 506 513 510 518 516 509 512 510 509 ...
##  $ roll_dumbbell       : num  13.4 13.4 13.4 12.8 13.2 ...
##  $ pitch_dumbbell      : num  -70.4 -70.4 -70.8 -70.3 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.9 -84.5 -85.1 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 36 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0.02 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  -0.02 0 0 0 0 0 -0.02 0 -0.02 -0.02 ...
##  $ accel_dumbbell_x    : int  -232 -233 -234 -234 -232 -235 -234 -233 -231 -233 ...
##  $ accel_dumbbell_y    : int  48 48 48 46 47 48 48 47 47 47 ...
##  $ accel_dumbbell_z    : int  -269 -270 -269 -272 -269 -270 -269 -272 -268 -271 ...
##  $ magnet_dumbbell_x   : int  -552 -554 -558 -555 -549 -558 -552 -551 -557 -559 ...
##  $ magnet_dumbbell_y   : int  303 292 294 300 292 291 302 296 292 295 ...
##  $ magnet_dumbbell_z   : num  -60 -68 -66 -74 -65 -69 -69 -56 -62 -74 ...
##  $ roll_forearm        : num  28.1 28 27.9 27.8 27.7 27.7 27.2 27.1 27 26.9 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 -63.9 -64 -64 -64
...
##  $ yaw_forearm         : num  -152 -152 -152 -152 -152 -152 -151 -151 -151 -151 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.02 0.02 0.02 0.02 0.03 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_forearm_y     : num  -0.02 0 -0.02 -0.02 0 0 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  0 -0.02 -0.03 0 -0.02 -0.02 -0.03 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  189 189 193 193 193 190 193 192 192 192 ...
##  $ accel_forearm_y     : int  206 206 203 205 204 205 205 204 206 203 ...
##  $ accel_forearm_z     : int  -214 -214 -215 -213 -214 -215 -215 -213 -216 -216 ...
##  $ magnet_forearm_x    : int  -16 -17 -9 -9 -16 -22 -15 -13 -16 -10 ...
##  $ magnet_forearm_y    : num  658 655 660 660 653 656 655 653 653 657 ...
##  $ magnet_forearm_z    : num  469 473 478 474 476 473 472 481 472 466 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1
...
```

Appendix B: Most important variables

```
varImp(modelFit)
```

```
##                          Overall
## new_window             0.3615052
## num_window           787.5226448
## roll_belt            661.7996731
## pitch_belt           388.8794362
## yaw_belt             462.2021637
## total_accel_belt     120.2506082
## gyros_belt_x          52.4653577
## gyros_belt_y          66.5763104
## gyros_belt_z         170.5780998
## accel_belt_x          73.4141030
## accel_belt_y          78.7792309
## accel_belt_z         214.9313132
## magnet_belt_x        141.8007849
## magnet_belt_y        217.6904066
## magnet_belt_z        226.7694693
## roll_arm             169.9893468
## pitch_arm             94.8554194
## yaw_arm              123.5579924
## total_accel_arm       57.1040062
## gyros_arm_x           68.9851372
## gyros_arm_y           68.3052602
## gyros_arm_z           33.2202625
## accel_arm_x          148.7301199
## accel_arm_y           78.7339524
## accel_arm_z           69.8148217
```

```
## magnet_arm_x         146.2637543
## magnet_arm_y         125.6169189
## magnet_arm_z          92.8075040
## roll_dumbbell        252.5117155
## pitch_dumbbell       112.3521790
## yaw_dumbbell         151.6591392
## total_accel_dumbbell 158.8365646
## gyros_dumbbell_x      70.7949637
## gyros_dumbbell_y     136.8116038
## gyros_dumbbell_z      44.2442475
## accel_dumbbell_x     147.2577720
## accel_dumbbell_y     237.2481359
## accel_dumbbell_z     189.2282602
## magnet_dumbbell_x    265.6219163
## magnet_dumbbell_y    392.5635479
## magnet_dumbbell_z    420.6919915
## roll_forearm         311.5279965
## pitch_forearm        416.2794323
## yaw_forearm           86.1757423
## total_accel_forearm   59.2392643
## gyros_forearm_x       38.8871025
## gyros_forearm_y       65.6512097
## gyros_forearm_z       45.5522479
## accel_forearm_x      177.6136919
## accel_forearm_y       77.1516278
## accel_forearm_z      140.4636115
## magnet_forearm_x     114.5870141
## magnet_forearm_y     110.9179709
## magnet_forearm_z     143.8677890
```