Machine Learning Course Project

Brett E Shelton

01/02/2020

Executive Summary

Synopsis - Overview

The basic goal of this report is to explore the prediction power of machine learning as applied to an exercise dataset, so that we might predict which exercise is being completed (classe in the original data). "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, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants."
To complete this report, it will include:
* "a description of how the prediction model was built"
* "how cross validation was used"
* "what the expected out-of-sample error is"
* "why the choices were made"
* "use of the prediction model to predict 20 test cases"
Background and data:
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, your goal will be 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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).
Data
The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

Cleaning and Tidying the Data

Here we load the libraries.

```
library(dplyr); library(datasets); library(knitr); library(ggplot2);
library(rattle); library(caret); library(randomForest); library(rpart); library(rpart.plot); library(kl
set.seed(1411)
```

Here we load the data and put it into the format desirable for our analysis.

```r
if (file.exists("./pml-training.csv") == FALSE) {
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv","./pml-training
}
if (file.exists("./pml-testing.csv") == FALSE) {
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv","./pml-testing.
}
pml.training <- read.csv("./pml-training.csv",sep = ",",header = TRUE)
pml.testing <- read.csv("./pml-testing.csv",sep = ",",header = TRUE)
```

Check the names and the unique users of training and testing data.

```r
dim(pml.training); dim(pml.testing)
```

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

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

```r
unique(pml.training$user_name)
```

```
## [1] carlitos pedro    adelmo   charles  eurico   jeremy
## Levels: adelmo carlitos charles eurico jeremy pedro
```

```r
colnames(pml.training)
```

```
##   [1] "X"                        "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"
```

Select data that may be useful and eliminate variables with too many NA.

Since we can’t test the model on any data that’s not in the test dataset, we’ll clean the test data and use those variables to pull the training set. So, if the NA data constitutes greater than 80% of the variable, we remove it. We can also remove index value, timestamps and window info, as well as the last variable which doesn’t match the training set.

```
pml.training.c <- pml.training[,c(2,8:159)]
pml.testing.c <- pml.testing[,c(2,8:159)]

#here we take out the variables with NA data
empty.vars <- colnames(pml.testing.c[colSums(is.na(pml.testing.c)) > (.8*(nrow(pml.testing.c)))])
keepers <- !(colnames(pml.testing.c)  %in% empty.vars)
keep.vars <- colnames(pml.testing.c[(keepers == TRUE)])
testing <- pml.testing.c[keep.vars]
training <- pml.training.c[keep.vars]

#add the classe variable back to training set
training <- mutate(training, classe = pml.training$classe)

colnames(training)
```

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

Random Forest

The outcome we are trying to predict is the final “class” of exercise with a dumbell, based on data gathered from 6 male participants.
* Class A is the correct use of the dumbell.
* Class B is with elbows too far forward.
* Class C is a half-way lift.
* Class D is a half-way drop.
* Class E is with hips too far forward.

If we use a method like random forest for our prediction, we achieve cross validation that avoids overfitting due to the split of training and test sets, and due to randomization of variable selection over each tree split.

That is, the "train" function in the "caret" package takes care of the cross validation for us. For further clarification, we could use the "rfcv" function. That is,
rfcv(trainx = rfTraining[,-54], trainy = rfTraining[,54])
should give us the number of predictors and the error rate associated with each as we reduce the number of predictors in the rf model.
Out-of-sampling error in random forests is a technique to verify the performance of the bootstrapping method, but since we have a validation set (the second split of our our original training set), that error is already addressed by comparing accuracies.

Random forest prediction

First we split the designated testing data into 2 partitions to train and test the random forest model.

```
inTrain <- createDataPartition(y=training$classe,p=0.7,list = FALSE)
rfTraining <- training[inTrain,]
rfTesting <- training[-inTrain,]
modFit.rf <- train(classe ~ .,data=rfTraining,method="rf",trControl=trainControl(method="cv",number = 3
# different method same result
# modFit.rf <- randomForest(classe ~ ., data = rfTraining)
prediction.rf <- predict(modFit.rf,rfTesting)
cm.rf <- confusionMatrix(prediction.rf, rfTesting$classe)
cm.rf
```

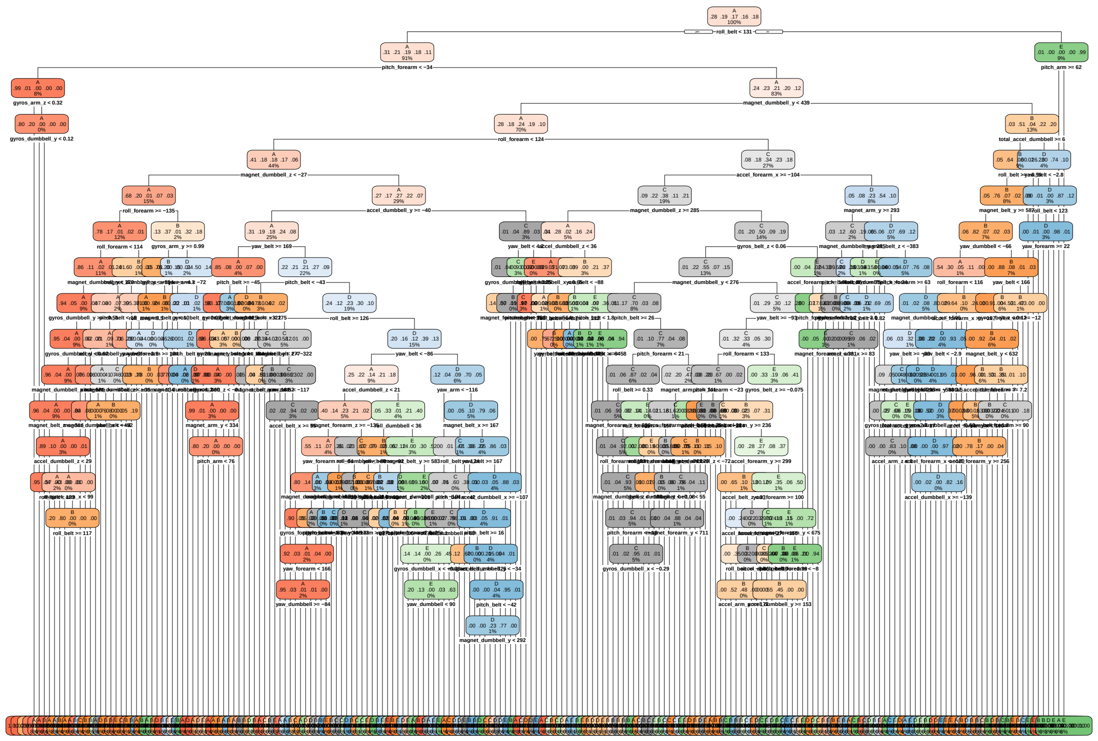
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670   13    0    0    0
##          B    2 1124   10    0    0
##          C    1    2 1010   15    0
##          D    0    0    6  946    3
##          E    1    0    0    3 1079
##
## Overall Statistics
##
##                Accuracy : 0.9905
##                  95% CI : (0.9877, 0.9928)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.988
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9868   0.9844   0.9813   0.9972
## Specificity            0.9969   0.9975   0.9963   0.9982   0.9992
## Pos Pred Value         0.9923   0.9894   0.9825   0.9906   0.9963
## Neg Pred Value         0.9990   0.9968   0.9967   0.9963   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2838   0.1910   0.1716   0.1607   0.1833
## Detection Prevalence   0.2860   0.1930   0.1747   0.1623   0.1840
## Balanced Accuracy      0.9973   0.9922   0.9904   0.9897   0.9982
```

As seen in the confusion matrix, accuracy of the random forest attempt resulted in 99.29% on the subsetted test data.

Can we improve with a different model?

Let's see if a decision tree (CART) is more accurate, using cross validation (xval).

```
fitControl <- rpart.control(cp=0.0001,xval = 10)
modFit.dt <- rpart(classe ~ ., method="class",data=rfTraining,control=fitControl)
rpart.plot(modFit.dt)
```



```
prediction.dt <- predict(modFit.dt, rfTesting, type = "class")
cm.dt <- confusionMatrix(prediction.dt, rfTesting$classe)
cm.dt
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1598   48   10    8    3
##          B   32  997   53   19   38
##          C   17   43  931   29   23
##          D   13   33   19  887   13
##          E   14   18   13   21 1005
##
## Overall Statistics
##
##                Accuracy : 0.9206
##                  95% CI : (0.9134, 0.9274)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8996
##
##  Mcnemar's Test P-Value : 0.0003974
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9546   0.8753   0.9074   0.9201   0.9288
## Specificity            0.9836   0.9701   0.9769   0.9841   0.9863
## Pos Pred Value         0.9586   0.8753   0.8926   0.9192   0.9384
## Neg Pred Value         0.9820   0.9701   0.9804   0.9843   0.9840
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2715   0.1694   0.1582   0.1507   0.1708
## Detection Prevalence   0.2833   0.1935   0.1772   0.1640   0.1820
## Balanced Accuracy      0.9691   0.9227   0.9422   0.9521   0.9575
```

As suspected, the CART decision tree was less accurate on the subsetted test data, resulting in 92.06%. Not bad, but not as good.

What about a linear discriminant analysis model?

```
modFit.lda <- train(classe ~ ., data=rfTraining,method="lda")
prediction.lda <- predict(modFit.lda, rfTesting)
cm.lda <- confusionMatrix(prediction.lda, rfTesting$classe)
cm.lda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1414  186  115   74   35
##          B   42  704   88   33  133
##          C   98  147  674  114   86
##          D  119   43  129  734   94
##          E    1   59   20    9  734
##
## Overall Statistics
##
##                Accuracy : 0.7239
##                  95% CI : (0.7123, 0.7353)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                    Kappa : 0.65
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8447   0.6181   0.6569   0.7614   0.6784
## Specificity            0.9026   0.9376   0.9084   0.9218   0.9815
## Pos Pred Value         0.7752   0.7040   0.6023   0.6559   0.8919
## Neg Pred Value         0.9360   0.9110   0.9261   0.9517   0.9313
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2403   0.1196   0.1145   0.1247   0.1247
## Detection Prevalence   0.3099   0.1699   0.1901   0.1901   0.1398
## Balanced Accuracy      0.8737   0.7779   0.7827   0.8416   0.8299
```

Here, accuracy results in 72.39%, so our original random forest is still a better choice.

Predicting the Final Test Dataset

Because the random forest model had by far the best accuracy on the testing validation set, we apply it to the final test set to turn in for the assignment.

```
prediction.final <- predict(modFit.rf, testing)
prediction.final
```

```
##  [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
```