Practical Machine Learning Course Project

Sean Fields

May 2, 2017

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, 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://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

Assignment

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Your submission should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to $<$ 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders.

You should also apply your machine learning algorithm to the 20 test cases available in the test data above. Please submit your predictions in appropriate format to the programming assignment for automated grading. See the programming assignment for additional details.

Reproducible Work

To ensure reproducibility, a seed has been set at 1234 for this analysis. The packages used are all shown being loaded below.

The Model

The variable we are trying to predict is the "classe" variable. This variable is used to identify in which way participants performed one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl. There are 5 levels of this variable:

- Exactly according the 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)
- Throwing the hips to the front (Class E)

After cleaning our data to remove irrelevant and mostly NA variables, all of the remaining variables will be used for prediction. Decision tree and random forst algorithms will be used. The model with a higher accuracy will be the model that is used on the testing data.

Cross Validation

To cross validate, we will split our training data into to sets; one will contain 70% of the data and the other will contain 30%. The models we use will be fitted on the new training set (70%) and tested on the new testing set (30%).

Expected out-of-sample error

The expected out-of-sample error will be equal to 1 minus the accuracy in the cross-validation data. Accuracy is the proportion of correct classified observations over the total sample size of the new testing data. Expected accuracy is the expected accuracy of the original testing data. Thus, the expected value of the out-of-sample error will correspond to the expected number of misclassified observations divided by the total number of observations in the test set.

Loading packages and setting the seed.

```
library(caret)
library(randomForest)
library(rpart)
library(rpart.plot)
library(rattle)

set.seed(1234)
```

Reading in the data

```
pml.testing <- read.csv("pml-testing.csv", na.strings = c("NA", "#DIV/0!",""))
pml.training <- read.csv("pml-training.csv", na.strings = c("NA", "#DIV/0!",""))
```

Splitting our data into two sets

```
inTrain <- createDataPartition(y = pml.training$classe, p = 0.70, list = FALSE)
training2 <- pml.training[inTrain,]
testing2 <- pml.training[-inTrain,]
```

Cleanining the training and new testing data

The following code will remove the identifier, timestamp, and window columns as they cannot be used for prediction. It will also remove all columns that have NA rates above 95%.

```
excvars <- grep("name|timestamp|window|X", colnames(pml.training), value = FALSE)

training3 <- training2[,-excvars]
NArate <- apply(training3, 2, function(x) sum(is.na(x)))/nrow(training3)
training3 <- training3[!(NArate > 0.95)]
dim(training3)
```

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

```
testing3 <- testing2[,-excvars]
NArate2 <- apply(testing3, 2, function(x) sum(is.na(x)))/nrow(testing3)
testing3 <- testing3[!(NArate > 0.95)]
dim(testing3)
```

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

```
summary(training3) #this would show a summary of training3
summary(testing3) #testing 3
```

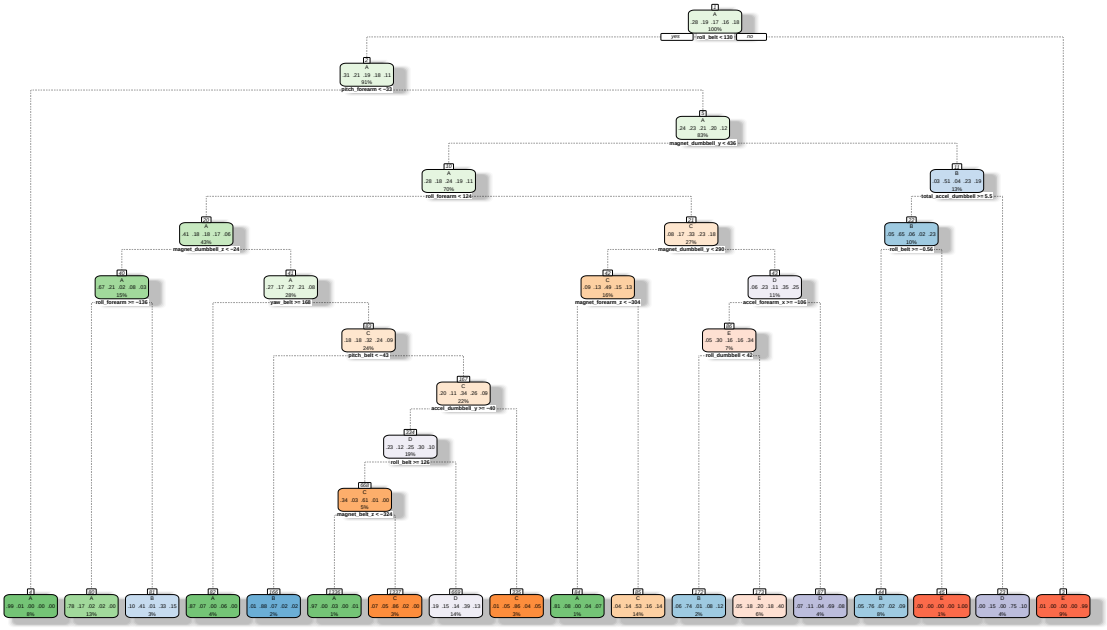
Decision Tree

This portion will be applying the decision tree method.

```
mod_DT <- rpart(classe ~ ., data = training3, method = "class")
```

Viewing the decision tree:

```
fancyRpartPlot(mod_DT)
```



Rattle 2017−May−03 04:32:30 Sean

Predicting using decision tree

```
pred_DT <- predict(mod_DT, testing3, type = "class")
```

The confusion matrix below will show our models accuracy.

```
confusionMatrix(pred_DT, testing3$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1364  169   24   48   16
##          B   60  581   46   79   74
##          C   52  137  765  129  145
##          D  183  194  125  650  159
##          E   15   58   66   58  688
##
## Overall Statistics
##
##                Accuracy : 0.6879
##                  95% CI : (0.6758, 0.6997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.6066
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8148  0.51010   0.7456   0.6743   0.6359
## Specificity            0.9390  0.94543   0.9047   0.8657   0.9590
## Pos Pred Value         0.8415  0.69167   0.6230   0.4958   0.7774
## Neg Pred Value         0.9273  0.88940   0.9440   0.9314   0.9212
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2318  0.09873   0.1300   0.1105   0.1169
## Detection Prevalence   0.2754  0.14274   0.2087   0.2228   0.1504
## Balanced Accuracy      0.8769  0.72776   0.8252   0.7700   0.7974
```

The accuracy shown in the matrix is 0.6878505

Random Forest

This portion will be applying the random forest method.

```
mod_RF <- randomForest(classe ~ ., data = training3)
mod_RF
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training3)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.51%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3903    2    0    0    1 0.0007680492
## B   10 2643    5    0    0 0.0056433409
## C    0   14 2379    3    0 0.0070951586
## D    0    0   25 2226    1 0.0115452931
## E    0    0    2    7 2516 0.0035643564
```

Predicting using random forests

```
pred_RF <- predict(mod_RF, testing3, type = "class")
```

Viewing results in a confusion matrix.

```
confusionMatrix(pred_RF, testing3$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    5    0    0    0
```

```
##          B    0 1133    5    0    0
##          C    0    1 1021    6    0
##          D    0    0    0  957    1
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9968
##                  95% CI : (0.995, 0.9981)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9959
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9951   0.9927   0.9991
## Specificity            0.9988   0.9989   0.9986   0.9998   0.9998
## Pos Pred Value         0.9970   0.9956   0.9932   0.9990   0.9991
## Neg Pred Value         1.0000   0.9987   0.9990   0.9986   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1925   0.1735   0.1626   0.1837
## Detection Prevalence   0.2853   0.1934   0.1747   0.1628   0.1839
## Balanced Accuracy      0.9994   0.9968   0.9968   0.9963   0.9994
```

The accuracy for this confusion matrix is 0.9967715

Decision

The random forest method performed significantly better than the decision tree, so it will be used for the original testing set. With such a high accuracy, we expect very few, if any, of our test set to be misclassified.