Practical Machine Learning Week 4 Assignment

Jason Miller

23 August 2016

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, our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbbell 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).

It is thus the focus of this project to predict whether an exercise is done correctly from the other variables available.

How the model was built

Our outcome variable is 'classe', a factor variable with 5 levels. For this data set, participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 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)
- 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. Prediction evaluations will be based on maximizing the accuracy and minimizing the out-of-sample error. All other available variables after cleaning will be used for prediction. Two models will be tested using Decision Tree and Random Forest algorithms. The model with the highest accuracy will be chosen as our final model.

Loading the Data

```
rm(list=ls())
pml_train <- read.csv("C:/Users/J Miller/Desktop/Data Science Course/Practical Machine Learni
ng/Assignment/pml-training.csv")
Testing <- read.csv("C:/Users/J Miller/Desktop/Data Science Course/Practical Machine Learnin
g/Assignment/pml-testing.csv")

#install.packages("caret")
library(caret); library(rpart); library(randomForest); library(rattle);
```

Exploratory Analysis & Data Cleaning

To clean the data, we will remove irrelevant variables first. Beyond that, it makes sense to review variables that will not be able to contribute significantly to the model in that they do not capture enough responses to relay meaningful information. Below we remove any variables in which 66% or more of observations are missing or NA.

```
# Perform exploratory analysis
# dim(pml_train); head(pml_train); summary(pml_train); str(pml_train)

# Delete variables that are irrelevant to our current project: user_name, raw_timestamp_part_
1, raw_timestamp_part_,2 cvtd_timestamp, new_window, and  num_window (columns 1 to 7).
pml_train2 <- pml_train[,-c(1:7)]
Testing <- Testing[,-c(1:7)]

# Delete columns with more than 66% missing or NA values
pml_train2 <- pml_train2[,colMeans(is.na(pml_train2) | pml_train2=="") < .66]
keepNZV <- names(Testing) %in% names(pml_train2)
Testing <- Testing[,keepNZV]
```

Cross Validation & Model Fitting

There is often a tendency to split off 25% of the training set into a Validation set and using the remaining 75% to train the model. Because this data set is quite large and the models we are planning to fit are quite system intensive, we will split 60% into a Training set and 40% into a Validation set. we then fit a Decision Tree and Random Forest to be compared thereafter.

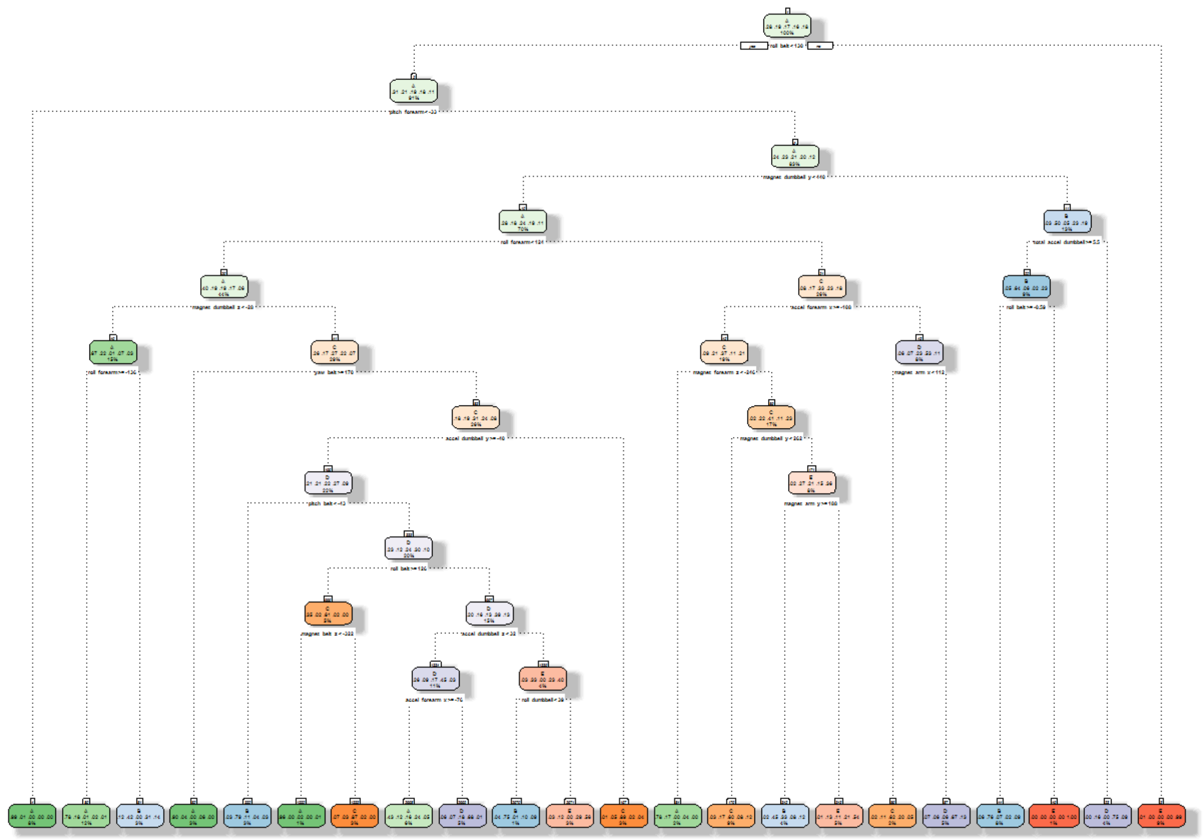
```
# partition the data so that 60% of the training dataset into training and the remaining 40%
 to validation
part <- createDataPartition(y=pml_train2$classe, p=0.6, list=FALSE)
Training <- pml_train2[part, ]
Validation <- pml_train2[-part, ]

model1 <- rpart(classe ~ ., data=Training, method="class")
prediction1 <- predict(model1, Validation, type = "class")

model2 <- randomForest(classe ~ ., data=Training, method="class")
prediction2 <- predict(model2, Validation, type = "class")
```

Having created both a Decision Tree model and a Random Forests model, we are able to compare their predictive power on the Validation set.

```
# Plot the Decision Tree
fancyRpartPlot(model1)
```



Rattle 2016-Aug-26 10:38:49 J Miller

```
confusionMatrix(prediction1, Validation$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2048  274   78  169   37
##          B   71  913  145  129  146
##          C   43  164  967  111  108
##          D   58   97  114  748  106
##          E   12   70   64  129 1045
## 
## Overall Statistics
## 
##                Accuracy : 0.7292
##                  95% CI : (0.7192, 0.739)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.6552
##  Mcnemar's Test P-Value : < 2.2e-16
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9176   0.6014   0.7069  0.58165   0.7247
## Specificity            0.9006   0.9224   0.9342  0.94284   0.9571
## Pos Pred Value         0.7859   0.6503   0.6942  0.66607   0.7917
## Neg Pred Value         0.9649   0.9061   0.9379  0.91998   0.9392
## Prevalence             0.2845   0.1935   0.1744  0.16391   0.1838
## Detection Rate         0.2610   0.1164   0.1232  0.09534   0.1332
## Detection Prevalence   0.3321   0.1789   0.1775  0.14313   0.1682
## Balanced Accuracy      0.9091   0.7619   0.8206  0.76224   0.8409
```

```
# Test results on Validation data set:
confusionMatrix(prediction2, Validation$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2229    7    0    0    0
##          B    2 1507   10    0    0
##          C    0    4 1357   22    1
##          D    0    0    1 1262    7
##          E    1    0    0    2 1434
## 
## Overall Statistics
## 
##                Accuracy : 0.9927          
##                  95% CI : (0.9906, 0.9945)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
##                   Kappa : 0.9908          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9928   0.9920   0.9813   0.9945
## Specificity            0.9988   0.9981   0.9958   0.9988   0.9995
## Pos Pred Value         0.9969   0.9921   0.9805   0.9937   0.9979
## Neg Pred Value         0.9995   0.9983   0.9983   0.9964   0.9988
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1921   0.1730   0.1608   0.1828
## Detection Prevalence   0.2850   0.1936   0.1764   0.1619   0.1832
## Balanced Accuracy      0.9987   0.9954   0.9939   0.9901   0.9970
```

Conclussion

From the above, we see that the Random Forest model does significantly better with an accuracy of 0.9939 compared to the 0.7702 accuracy from the Decision Tree model. This, it must be said, was fairly predictable because Random Forests is just successive Decision Trees fit to increase accuracy, so what we have displayed here is literally why Random Forests was created.

None the less, we take our findings and thus use model2, the Random Forests, to predict classe for the Test set. Further, we note that the expected out-of-sample error is estimated at 0.0061, or 0.61%.

Predicting on Test set

```
predictfinal <- predict(model2, Testing, type="class")
predictfinal
```

```
##  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
```