Practical Machine Learning Course Project

Michael Drobish

June 2, 2016

Executive Summary

The goal of this project is to predict the manner in which a person performed excercise based on the data collected while a person excercised wearing devices such as Jawbone Up, Nike FuelBand, and Fitbit. A person would be classified as class A, B, C, D, or E. Class A means that the excercise was performed correctly, while other classifications indicate the type of errors. (See Appendix A for more information.)

I compared two prediction models, decision tree and random forest. Both used 3-fold cross validation for the fit control. As expected, the random forest model produced much better accuracy of 99.1% compared to just 50% of the decision tree. The down side is that it took much longer to process the random forest.

I re-ran the random forest with PCA preprocess. PCA process lost only 0.9% of accuracy. However, I chose to use random forest model without PCA preprocess.

Preparation

```
library(caret)
library(rpart)
library(randomForest)
```

```
raw.training <- read.csv("./pml-training.csv", head=TRUE, sep=",", na.strings=c("NA", "", "#DIV/0!"))
raw.testing <- read.csv("./pml-testing.csv",head=TRUE, sep=",")
data.training <- raw.training[, colSums(is.na(raw.training)) == 0]
data.testing <- raw.testing[, colSums(is.na(raw.training)) == 0]
data.training <- data.training[, -c(1:7)]
data.testing <- data.testing[, -c(1:7)]
```

See Appendix B - Cleaning Data for more information on how I cleaned data.

Create sub-training and test data from the training set.

```
set.seed(23678)
#Build sub-train and sub-test data from the training data
data.training.inTrain <- createDataPartition(y=data.training$classe, p=0.7, list=FALSE)
data.training.train <- data.training[data.training.inTrain,]
data.training.test <- data.training[-data.training.inTrain,]
```

Building Models

Let's use 3-fold cross validation when building models

```
fitControl.cv3 <- trainControl(method="cv", number=3, verboseIter = FALSE)
```

Model 1: Decision Tree Model with 3-fold cross validation

```
tree_model.train <- train(classe ~ ., data=data.training.train, method="rpart", trControl=fitControl.cv3
tree_model.predict <- predict(tree_model.train, newdata=data.training.test)
tree_model.matrix <- confusionMatrix(data.training.test$classe, tree_model.predict)
```

See Appendix C - Building Models: Outputs of Decision Tree Model with 3-fold cross validation

Model 2: Random Forest with 3-fold cross validation

```
rf_model.train <- train(classe ~ ., data=data.training.train, method="rf", trControl=fitControl.cv3)
rf_model.predict <- predict(rf_model.train, newdata=data.training.test)
rf_model.matrix <- confusionMatrix(data.training.test$classe, rf_model.predict)

#print the final model
#rf_model.fit$finalModel
```

See Appendix D - Building Models: Outputs of Randon Forest Model with 3-fold cross validation

Compare Two Models

```
data.frame(tree_model.matrix$overall, rf_model.matrix$overall)
```

```
##                tree_model.matrix.overall rf_model.matrix.overall
## Accuracy                       0.5007647               0.9916737
## Kappa                          0.3479743               0.9894677
## AccuracyLower                  0.4879075               0.9890070
## AccuracyUpper                  0.5136211               0.9938340
## AccuracyNull                   0.5145285               0.2851317
## AccuracyPValue                 0.9832209               0.0000000
## McnemarPValue                        NaN                     NaN
```

The random forest model has much better accuracy of 99.1% compared to just 50% of the decision tree model. So, we will use the random forest model.

Random Forest with PCA preprocess

For fun, let's preprocess the data before running the random forest training

```
preProc <- preProcess(data.training.train[,-53], method="pca")
trainPCA <- predict(preProc, data.training.train[,-53])
trainPCA$classe <- data.training.train$classe
```

```
testPCA <- predict(preProc, data.training.test[,-53])
testPCA$classe <- data.training.test$classe

rf_pca.train <-train(classe ~ ., data= trainPCA, method="rf")
rf_pcs.predict <- predict(rf_pca.train, testPCA)
rf_pcs.matrix <- confusionMatrix(rf_pcs.predict, testPCA$classe)
rf_pcs.matrix$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.809686e-01   9.759226e-01   9.771447e-01   9.843044e-01   2.844520e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   8.201720e-06
```

Submission

```
final_prediction <- predict(rf_model.train, data.testing)
final_prediction
```

```
##  [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
```

Appendix

Apendix A - About Data

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions:

1. Class A - exactly according to the specification
2. Class B - throwing the elbows to the front
3. Class C - lifting the dumbbell only halfway
4. Class D - lowering the dumbbell only halfway
5. Class E - throwing the hips to the front

The data for this project can be downloaded:

1. Training Data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
2. Testing Data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Appendix B - Cleaning Data

1. The raw training data contains 19,622 observations and 160 variables.
2. 100 of those variables had NA values in all observations. Therefore they were removed.
3. Following variables are also removed because they are irrelevant (X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window, and num_window)

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

data.training <- training[, colSums(is.na(training)) == 0]
data.testing <- testing[, colSums(is.na(training)) == 0]

data.training <- data.training[, -c(1:7)]
data.testing <- data.testing[, -c(1:7)]
```

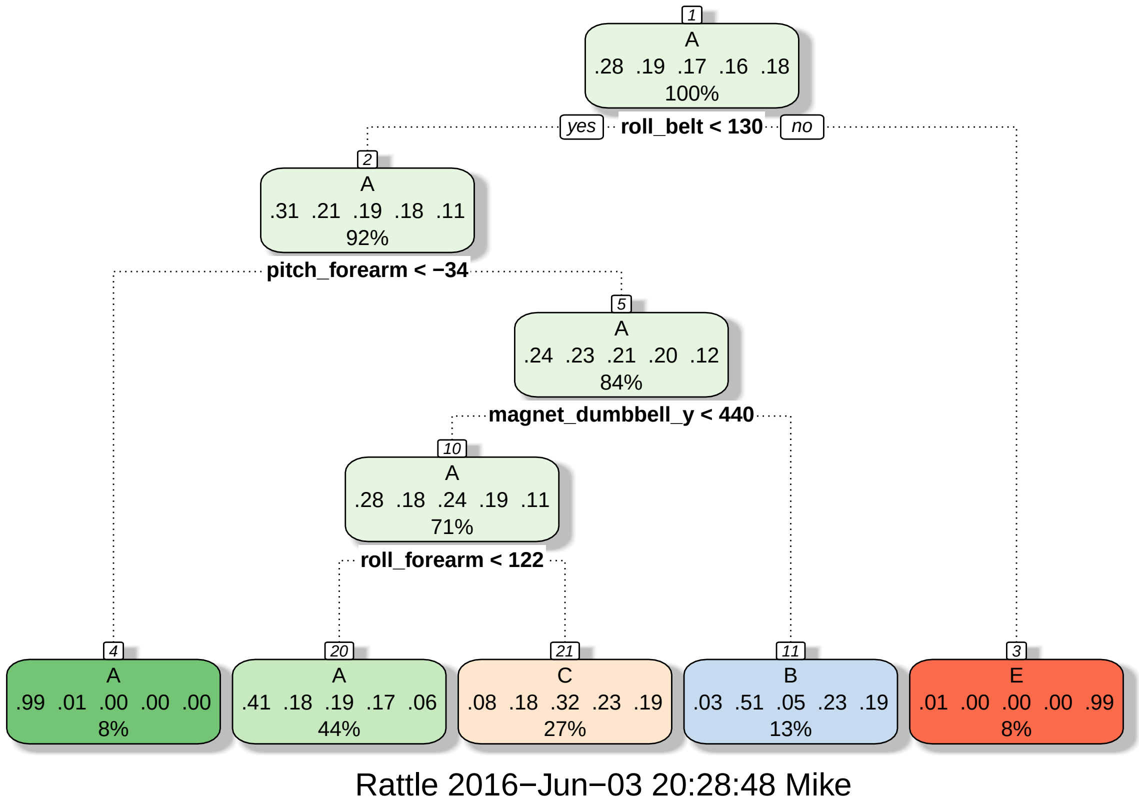
Appendix C - Building Models: Outputs of Decision Tree Model with 3-fold cross validation

```r
library(rattle)
```

```r
#print the training of the model
print(tree_model.train$finalModel)
```

```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 12607 8709 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1124    9 A (0.99 0.008 0 0 0) *
##      5) pitch_forearm>=-33.95 11483 8700 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 439.5 9721 6990 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 121.5 6002 3566 A (0.41 0.18 0.19 0.17 0.062) *
##         21) roll_forearm>=121.5 3719 2516 C (0.079 0.18 0.32 0.23 0.19) *
##       11) magnet_dumbbell_y>=439.5 1762  870 B (0.03 0.51 0.045 0.23 0.19) *
##    3) roll_belt>=130.5 1130    8 E (0.0071 0 0 0 0.99) *
```

```r
fancyRpartPlot(tree_model.train$finalModel)
```




```r
#print the confusion Matrix of the prediction
tree_model.matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1517   29  122    0    6
##          B  464  394  281    0    0
##          C  471   28  527    0    0
##          D  428  159  377    0    0
##          E  148  157  268    0  509
##
## Overall Statistics
##
##                Accuracy : 0.5008
##                  95% CI : (0.4879, 0.5136)
##     No Information Rate : 0.5145
##     P-Value [Acc > NIR] : 0.9832
##
##                   Kappa : 0.348
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.5010  0.51369  0.33460      NA  0.98835
## Specificity            0.9450  0.85444  0.88422  0.8362  0.89330
## Pos Pred Value         0.9062  0.34592  0.51365      NA  0.47043
## Neg Pred Value         0.6412  0.92141  0.78432      NA  0.99875
## Prevalence             0.5145  0.13033  0.26763  0.0000  0.08751
## Detection Rate         0.2578  0.06695  0.08955  0.0000  0.08649
## Detection Prevalence   0.2845  0.19354  0.17434  0.1638  0.18386
## Balanced Accuracy      0.7230  0.68406  0.60941      NA  0.94082
```

Appendix D - Building Models: Outputs of Random Forest Model with 3-fold cross validation

```r
#print the training of the model
print(rf_model.train$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.71%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3898    5    2    0    1 0.002048131
## B   22 2630    6    0    0 0.010534236
## C    0    9 2379    8    0 0.007095159
## D    0    2   29 2219    2 0.014653641
## E    0    1    3    8 2513 0.004752475
```

```r
#print the confusion Matrix of the prediction
rf_model.matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    5 1129    5    0    0
##          C    0    8 1006   12    0
##          D    0    1    6  954    3
##          E    0    0    3    5 1074
##
## Overall Statistics
##
##                Accuracy : 0.9917
##                  95% CI : (0.989, 0.9938)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9895
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9912   0.9863   0.9825   0.9972
## Specificity            0.9998   0.9979   0.9959   0.9980   0.9983
## Pos Pred Value         0.9994   0.9912   0.9805   0.9896   0.9926
## Neg Pred Value         0.9988   0.9979   0.9971   0.9965   0.9994
## Prevalence             0.2851   0.1935   0.1733   0.1650   0.1830
## Detection Rate         0.2843   0.1918   0.1709   0.1621   0.1825
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9984   0.9946   0.9911   0.9902   0.9978
```