Machine Learning Course Project: Predicting Exercise Quality

Albert Stanton

January 31, 2016

Executive Summary

The objective of this project was to build a model to classify how an exercise was performed, specifically a dumbbell bicep curl. There are five different potential classes, A, B, C, D, and E. Class A corresponds to correct execution of the exercise, while the other classes indicate different deficiencies in execution. Using the random forest machine learning method, a model was built on the training data that correctly classified 20 out of 20 exercises provided in the testing data.

Exploratory Analysis

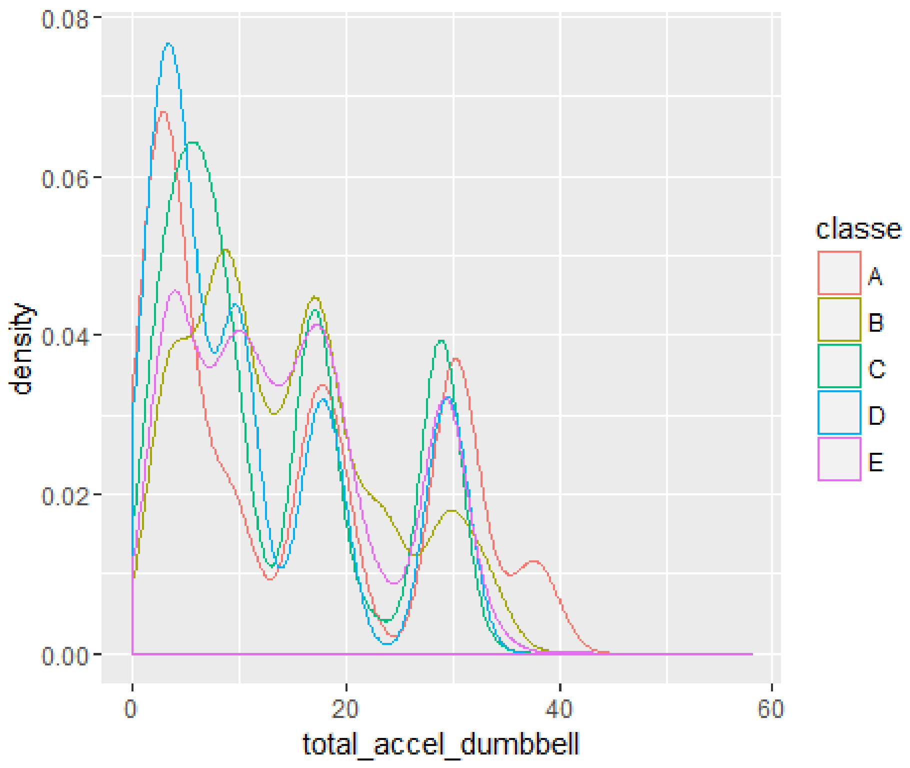
First we load in the training and testing data, and see what sort of data we are working with.

```r
library(caret)
library(randomForest)

training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")

str(training); str(testing)
head(training); head(testing)
```

The output of the above code has been hidden, as it is too long to include in the report. The result of this inspection, however, was seeing that there are some variables that are not related to the exercise movements, and some variables that contain mostly NA's.

There are many variables, but just to get a feel for the data, let's plot one of the variables against classe (the outcome variable) and see what it looks like.

```r
qplot(total_accel_dumbbell, colour = classe, data = training, geom =
'density')
```



So no obvious patterns in the data. Exploration of other variables in relation to classe yield similar findings.

Data Cleaning

Based on the exploratory analysis, there are many variables that should be excluded from the analysis. Below, we remove those variables.

```
removeList <- c()
for (i in 1:ncol(training)) {
        if (is.na(training[1,i]) | training[1,i] == "" | names(training)[i]
== "" |
            names(training)[i] == "num_window") {
                removeList <- c(removeList, i)
        }
}
training <- training[,-removeList]; training <- training[,-c(1:6)]

removeList <- c()
for (i in 1:ncol(testing)) {
        if (is.na(testing[1,i]) | testing[1,i] == "" | names(testing)[i] ==
"" |
            names(testing)[i] == "num_window") {
                removeList <- c(removeList, i)
        }
}
```

```
testing <- testing[,-removeList]; testing <- testing[,-c(1:6)]; testing <-
testing[,-c(53)]
```

This cleaning procedure results in a 19622 by 53 data frame for the training data, and a 20 by 52 data frame for the testing data. Model building can now commence.

Model Building

To begin, we set the seed so that the results are reproducible. Then, using the randomForest package, we build a random forest model using the training data set. The random forest method was chosen as it is well known for having excellent prediction accuracy, and simple classification trees were not powerful enough. Bagging and boosting were other options, though the random forest method will be shown to have been an excellent choice.

```
set.seed(2357)

modelFit <- randomForest(classe~., data = training)
print(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: 0.27%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5577    3    0    0    0 0.0005376344
## B   10 3784    3    0    0 0.0034237556
## C    0    9 3411    2    0 0.0032144944
## D    0    0   17 3197    2 0.0059079602
## E    0    0    1    5 3601 0.0016634322
```

Inspecting this model, we see it has a very low error rate. But we would like to be able to estimate the out-of-sample error as well.

Cross Validation

To estimate the out-of-sample error, we perform 5-fold cross validation on the training data.

```
folds <- createFolds(y=training$classe, k=5, list=TRUE, returnTrain = TRUE)

errorRates <- c()
for (i in 1:5) {
        tempTrain <- training[folds[[i]],]
        tempTest <- training[-folds[[i]],]
```

```r
        tempModel <- randomForest(classe~., data = tempTrain)
        tempPrediction <- predict(tempModel, tempTest)
        errorRate <- 1 - (sum(tempPrediction == tempTest$classe) /
nrow(tempTest))
        errorRates <- c(errorRates, errorRate)
}
print(errorRates)

## [1] 0.005605096 0.004077472 0.004841998 0.005095541 0.003058104

print(mean(errorRates))

## [1] 0.004535642
```

As we saw in the model building section, the in-sample error was 0.27%. We would expect the out-of-sample error to be higher, and the result of our cross-validation yields an estimate of the out-of-sample error of 0.45%. This is still quite low, and so we can expect our model to perform quite well on the real testing data.

Results

Now for the final test, we apply our original random forest model to the original testing data set.

```r
prediction <- predict(modelFit, testing)
print(prediction)

##  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
```

This prediction can be input to the Week 4 Quiz, and it turns out that this prediction is 100% accurate.

Conclusion

Using the training data, we were able to build a highly-successful random forest model. Our cross validation told us our model was quite powerful, predicting the test data perfectly. The random forest method was indeed an excellent choice for predicting the classes of exercise for this project.