Machine Learning Project

DJSherwood

## Background

Researchers[1] between the UK, Germany, and Brazil used machine learning to predict the manner in which a *Unilateral Dumbell Bisceps Curl* was conducted. Particpants were asked to perform the activity:

* Class A - Correctly
* Class B - Throwing elbows to the front
* Class C - Lifting the dumbell halfway
* Class D - Lowering the dumbell halfway
* Class E - Throwing the hips to the front

The gloves, armband, lumbar belt, and dumbell were outfitted with sensors[1]. The data was captured and features were derived from the sensor data. The researchers identified 17 features of interest. Ten *random forests* were created using 10 *trees* each[1]. Their random forest was highly accurate for each class:

* Class A - 97.6%
* Class B - 97.3%
* Class C - 98.2%
* Class D - 98.1%
* Class E - 99.1%

The goal for this work is to demonstrate undestanding of the material presented in the *Machine Learning* course in *Coursera's Data Scientist Specilization* track. Data from the weight lifting study[1] was used to train a machine learning algorithm using the **Caret** library in R. The results were tested and predictions were made on a separate dataset.

## The Data

The .csv files were loaded into memory. The file "pml-training.csv" was read in as the variable "trn" and the file "pml-testing.csv" was read in as the variable "prd." The variable "trn" contains measured and derived features, and also the class of activity performed. The variable "prd" contains 20 observations of the same variables, but lacks the corresponding class of activity, thus this was the data used to make predictions on. The prediction data was evaluated only after the training method was deemed able to control for the out-of-sample error.

The first seven columns were removed because they did not seem to be direct measurments of interest. That is, they were date measurements, subject names, etc.

## Remove prod id, username, etc.  
train\_data <- trn[,-c(1,2,3,4,5,6,7)]  
predn\_data <- prd[,-c(1,2,3,4,5,6,7)]

### Cleaning Data and Feature Selection

A cursory inspection of the .csv files showed columns upon columns filled with NAs. Further, upon using the str() command, several more columns were counted as type "factor" and were mostly empty. Columns which fit either of these observations were removed in hopes that sources of unwanted noise would be eliminated. The "classe" column--containing the class of activity for each observation--was not removed from the training set. This approach left 52 predictors and 1 response within the training and prediction data frames.

## Find bad columns  
bad\_seq <- NULL  
for (i in 1:(dim(train\_data)[2]-1)) {  
   
 ## Determine how "filled" a column is  
 percent\_na <- sum(is.na(train\_data[,i]))/(dim(train\_data)[1])  
   
 ## Boole  
 boole <- is.factor(train\_data[,i])  
   
 ## Add NA columns to bad list  
 if (percent\_na > 0.30) {  
 bad\_seq <- c(bad\_seq,i)  
 }  
   
 ## Add factor columns to bad list  
 else if (boole == TRUE) {  
 bad\_seq <- c(bad\_seq,i)  
 }  
   
}  
  
## Remove "bad" columns  
unique\_bad\_seq <- unique(bad\_seq)  
real\_train\_data <- train\_data[,-unique\_bad\_seq]  
real\_predn\_data <- predn\_data[,-unique\_bad\_seq]

## Fitting a Model

### Creating Training and Testing Sets

Because the prediction data does not have a "classe" column it is impossible to evaluate the out-of-sample-error. Thus, the training data from the "pml-training.csv" was divided into a "training" set and a "testing" set. The model was trained on the "training" set, and the out-of-sample error was estimated with this "testing" set. The fit from the satisfactory model was used to generate predictions from the prediction set.

## Divide the training data into training and testing  
inTrain <- createDataPartition(y=real\_train\_data$classe, p=0.75, list=FALSE)  
training <- real\_train\_data[inTrain,]  
testing <- real\_train\_data[-inTrain,]

### Preprocessing and Fitting

Although many different options were explored, the "random forest" method was used because:

* It is useful in data with high noise[1]
* It had the highest accuracy of any other attempted method (qda,lda,RRF,gmp, etc.)

It is especially important to perform cross-validation in conjunction with a *random forest*. "Repeated cross-validation" was used five times, and repeated 5 times. The data was normalized via the "center, scale" pre-processing option in the train() function.

## Control for the train function  
fitControl <- trainControl(method="repeatedcv",  
 number = 5,   
 repeats = 5)  
  
## Produce a model fit  
ptm <- proc.time()  
set.seed(1)  
modelFit <- train(training$classe ~.,   
 data=training,   
 method="rf",  
 preProcess = c("center","scale"),  
 trControl=fitControl)  
proc.time() - ptm

## user system elapsed   
## 2695.95 8.03 2707.23

### Out-of-Sample-Error and Accuracy

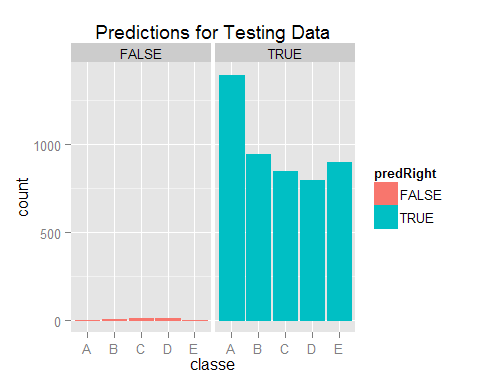
To evaluate the out-of-sample-error, the "testing" set was used to generate predictions. The final column, "classe," was removed from this operation. The predictions were then compared against activity classifcations in the "classe" column in the "testing" set to evaluate the specificity, sensitivity, and overall accuracy of the random forest method on the data. The output of the confusionMatrix() function shows low out-of-sample error. The sensitivity was high for all classes, as was the specificity.

## Generate predictions from "testing data"  
predictions <- predict(modelFit,newdata=testing[,-53])  
  
## Confustion Matrix  
confusionMatrix(predictions,testing$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1394 4 0 0 0  
## B 1 943 7 0 0  
## C 0 2 846 11 0  
## D 0 0 2 793 2  
## E 0 0 0 0 899  
##   
## Overall Statistics  
##   
## Accuracy : 0.994   
## 95% CI : (0.992, 0.996)  
## No Information Rate : 0.284   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.993   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.999 0.994 0.989 0.986 0.998  
## Specificity 0.999 0.998 0.997 0.999 1.000  
## Pos Pred Value 0.997 0.992 0.985 0.995 1.000  
## Neg Pred Value 1.000 0.998 0.998 0.997 1.000  
## Prevalence 0.284 0.194 0.174 0.164 0.184  
## Detection Rate 0.284 0.192 0.173 0.162 0.183  
## Detection Prevalence 0.285 0.194 0.175 0.163 0.183  
## Balanced Accuracy 0.999 0.996 0.993 0.993 0.999

As a visual example of how few incorrect classifications were made, consider a plot of true/false predictions using the testing set.

## This example was take from the lecture notes  
testing$predRight <- predictions==testing$classe  
  
## Plot  
qplot(classe, color=predRight,data=testing,fill=predRight,main="Predictions for Testing Data")+facet\_grid(.~predRight)



### Predictions

The model fit generated by the *random forest* was used with the predict() funtion to generate grades of activity for 20 samples of new data. These predictions were graded as correct when submitted, that is, the error in making new predictions was essentially zero.

## Generate predictions from "prediction" set  
predict(modelFit,newdata=real\_predn\_data)

## [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

## Conclusion

It was nearly possible to replicate the findings of the previous research using the Caret library in R. The random forest method was higly accurate, but the run-time was quite long. Potentially, if I had focused on finding only highly relevant features then the run-time would have been shorter.

## References

1. Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H.: Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13). Stuttgart, Germany: ACM SIGCHI, 2013