Machine Learning - Final Project

Carlos Calderon

November 27, 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, 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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Goal

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.

Data Processing

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

```
# Load packages needed for the project

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

```
# Download the raw data from the web

set.seed(12345)

training_raw_data <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-tr
aining.csv"
testing_raw_data <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-tes
ting.csv"

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

dim(training)
```

```
## [1] 19622   160
```

```
#head(training)
dim(testing)
```

```
## [1]  20 160
```

```
#head(training)

exclude_training <- c('X','user_name','raw_timestamp_part_1','raw_timestamp_par
t_2','cvtd_timestamp','new_window')
```

As per practical machine learning best practice we split the training data set into two, training and testing.

```
inTrain <- createDataPartition(training$classe, p=0.7, list=FALSE)
training_dataset <- training[inTrain, ]
testing_dataset <- training[-inTrain, ]
dim(training_dataset); dim(testing_dataset)
```

```
## [1] 13737   160
## [1] 5885  160
```

As we saw from the raw data download our data has a lot of ‘NAs’ and ‘Zeros’ that we will remove. We assume our model will still provide a good enough prediction which is to be confirmed later on.

```
# We clean the data by removing NA, Zeroes and Variables not needed for the mod
el.

cleantraining <- training_dataset[,!(names(training_dataset) %in% exclude_train
ing)]

nzvtraining <- nearZeroVar(cleantraining)
cleantraining <- cleantraining[,-nzvtraining]

filterData <- function(cleaned) {
    noNA <- !sapply(cleaned, function(x) any(is.na(x)))
    cleaned <- cleaned[, noNA]
    noBlank <- !sapply(cleaned, function(x) any(x==""))
    cleaned <- cleaned[, noBlank]
}

cleantraining <- filterData(cleantraining)
finalcleantraining <- cleantraining
dim(finalcleantraining)
```

```
## [1] 13737    54
```

```
# Same cleaning process for the testing data set

cleantesting <- testing_dataset[,!(names(testing_dataset) %in% exclude_trainin
g)]

nzvtesting <- nearZeroVar(cleantesting)
cleantesting <- cleantesting[,-nzvtesting]
cleantesting <- filterData(cleantesting)
finalcleantesting <- cleantesting
dim(finalcleantesting)
```

```
## [1] 5885   54
```

Model Selection

From lecture and class notes we have a good idea to start with boosting and or random forests as both are some of the most widely used techniques. Therefore, we will test these 2 models first. We will start with Boosting and follow with Random Forest. For completeness we will add a classification tree to compare against a third model. We are looking for accuracies greater than 90% using confustion matrix function.

Boosting

```
# Generate model by using the final clean training dataset
fitControl <- trainControl(method = "repeatedcv",number = 5, repeats = 1)
modFitBoost <- train(classe ~ ., data=finalcleantraining, method = "gbm", trCon
trol = fitControl, verbose = FALSE)

# Apply model to final clean testing data set
predictionBoost <- predict(modFitBoost, newdata=finalcleantesting)
CMBoost <- confusionMatrix(predictionBoost, finalcleantesting$classe)
CMBoost
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670   13    0    3    0
##          B    3 1111   22    7    1
##          C    0   10  998   15    3
##          D    1    5    6  939   13
##          E    0    0    0    0 1065
##
## Overall Statistics
##
##                Accuracy : 0.9827
##                  95% CI : (0.979, 0.9858)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9781
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9754   0.9727   0.9741   0.9843
## Specificity            0.9962   0.9930   0.9942   0.9949   1.0000
## Pos Pred Value         0.9905   0.9712   0.9727   0.9741   1.0000
## Neg Pred Value         0.9990   0.9941   0.9942   0.9949   0.9965
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2838   0.1888   0.1696   0.1596   0.1810
## Detection Prevalence   0.2865   0.1944   0.1743   0.1638   0.1810
## Balanced Accuracy      0.9969   0.9842   0.9835   0.9845   0.9921
```

Random Forest

```
# Generate model by using the final clean training dataset
modFitRF <- randomForest(classe ~ ., data=finalcleantraining)

# Apply model to final clean testing data set
predictionRF <- predict(modFitRF, finalcleantesting, type = "class")
CMRF <- confusionMatrix(predictionRF, finalcleantesting$classe)
CMRF
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    5    0    0    0
##          B    0 1133    4    0    0
##          C    0    1 1022   12    0
##          D    0    0    0  952    3
##          E    0    0    0    0 1079
## 
## Overall Statistics
## 
##                Accuracy : 0.9958          
##                  95% CI : (0.9937, 0.9972)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
##                   Kappa : 0.9946          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9961   0.9876   0.9972
## Specificity            0.9988   0.9992   0.9973   0.9994   1.0000
## Pos Pred Value         0.9970   0.9965   0.9874   0.9969   1.0000
## Neg Pred Value         1.0000   0.9987   0.9992   0.9976   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1925   0.1737   0.1618   0.1833
## Detection Prevalence   0.2853   0.1932   0.1759   0.1623   0.1833
## Balanced Accuracy      0.9994   0.9969   0.9967   0.9935   0.9986
```

Classification Tree

```
# Generate model by using the final clean training dataset
modFitTree <- rpart(classe ~ ., data=finalcleantraining, method="class")

# Apply model to final clean testing data set
predictionTree <- predict(modFitTree, finalcleantesting, type="class")
CMtree <- confusionMatrix(predictionTree, finalcleantesting$classe)
CMtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530  269   51   79   16
##          B   35  575   31   25   68
##          C   17   73  743   68   84
##          D   39  146  130  702  128
##          E   53   76   71   90  786
##
## Overall Statistics
##
##                Accuracy : 0.7368
##                  95% CI : (0.7253, 0.748)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6656
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140  0.50483   0.7242   0.7282   0.7264
## Specificity            0.9014  0.96650   0.9502   0.9100   0.9396
## Pos Pred Value         0.7866  0.78338   0.7543   0.6131   0.7305
## Neg Pred Value         0.9635  0.89051   0.9422   0.9447   0.9384
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2600  0.09771   0.1263   0.1193   0.1336
## Detection Prevalence   0.3305  0.12472   0.1674   0.1946   0.1828
## Balanced Accuracy      0.9077  0.73566   0.8372   0.8191   0.8330
```

Conclusion

Out of the three models Random Forest has the best accuracy at 99.5%.

```
# We calculate our out of sample error using 1 minus accuracy

outofsample_error <- 1 - CMRF$overall[1]
outofsample_error
```

```
##    Accuracy
## 0.004248088
```

```
print(paste("Out of Error Sample is: ", outofsample_error * 100, "%"))
```

```
## [1] "Out of Error Sample is:  0.424808836023793 %"
```

Therefore, we will use the Random Forest model on the original testing raw data

```
# Now we fit our Random Forest model to the original testing raw data

prediction_originaltesting_RF <- predict(modFitRF, testing, type = "class")

# Write the results to a text file for submission
pml_write_files = function(x){
    n = length(x)
    for(i in 1:n){
        filename = paste0("problem_id_",i,".txt")
        write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FA
LSE)
    }
}

pml_write_files(prediction_originaltesting_RF)
```