Practical Machine Learning Project

This document was produced for the course project of 'Practical Machine Learning', the 8th course in the 'Data Science Specialization', a MOOC on Coursera. The code in this report has been developed and tested on a machine running Ubuntu 14.04 and R 3.2.2.

Background

This report outlines the exploratory data analysis conducted on the 'Weight Lifting Exercises' dataset in R, details of which can be found here. The manual notes that '*Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five 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) and 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.*'

The goal is to predict if the exercise was perfomred correctly, i.e. predict which of the five aforementioned classes a particular routine correspinds to using the information provided. Multiple measurements were made using accelerometers attached to the fore arm, upper arm, waist and dumb-bells. These measured values constitute the predictors.

Data Sources

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Getting the data

First, I assign the URL's provided earlier to variables that I can access in R.

```
trUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
teUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

In an exploratory data analysis (not reported here) it was found that the missing data in this dataset can take one of three values. They are appropriately identified while inputing the data

```
train <- read.csv(url(trUrl), na.strings=c("NA","#DIV/0!",""))
test <- read.csv(url(teUrl), na.strings=c("NA","#DIV/0!",""))
```

```
dim(train)
```

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

```
summary(train)
```

A summary of the dataset (output suppressed here for brevity) shows that in additon to some columns with entirely missing data, a large number of columns contain ∼ 19000 NA's (19216, to be precise). These predictors are there unusable as missing values vastly outnumber the non-missing or provided data, i.e. imputing these values isn't feasible. All these columns are excluded

```
train <- train[,colSums(is.na(train)) < 19216]
```

Finally, the first column (containing ID's) and the fifth column (containing timestamps) are excluded as predictors

```
train <- train[,-c(1,5)]
```

Now, using the package *caret*, the dataset is divided into *testing* and *training* sets, following the 40/60 split suggested in the lectures.

```
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
inTrain <- createDataPartition(y=train$classe, p=0.6, list=FALSE)
myTrain <- train[inTrain, ]; myTest <- train[-inTrain, ]
dim(myTrain); dim(myTest)
```

```
## [1] 11776    58
```

```
## [1] 7846   58
```

Machine Learning Models

Now that the data had been formatted, machine learning models are developed to be able to predict the *classe* variable from the 57 predictors that have been short listed. Given the non-linear nature of the data, decision tree based algorithms are used to construct the models. The machine learning algorithms are trained on the *training* set. In the next section, their predictions on the *testing* set are compared.

Decision Trees

```
library(rpart)
set.seed(1234)
DTfit <- rpart(classe ~ ., data=myTrain, method="class")
```

Decision trees have an advantage in that you can view the results directly using the following commands (code for plot commented out).

```
library(rattle)
```

```
## Loading required package: RGtk2
## Rattle: A free graphical interface for data mining with R.
## Version 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(rpart.plot)
library(RColorBrewer)
#fancyRpartPlot(DTfit)
```

Random Forests

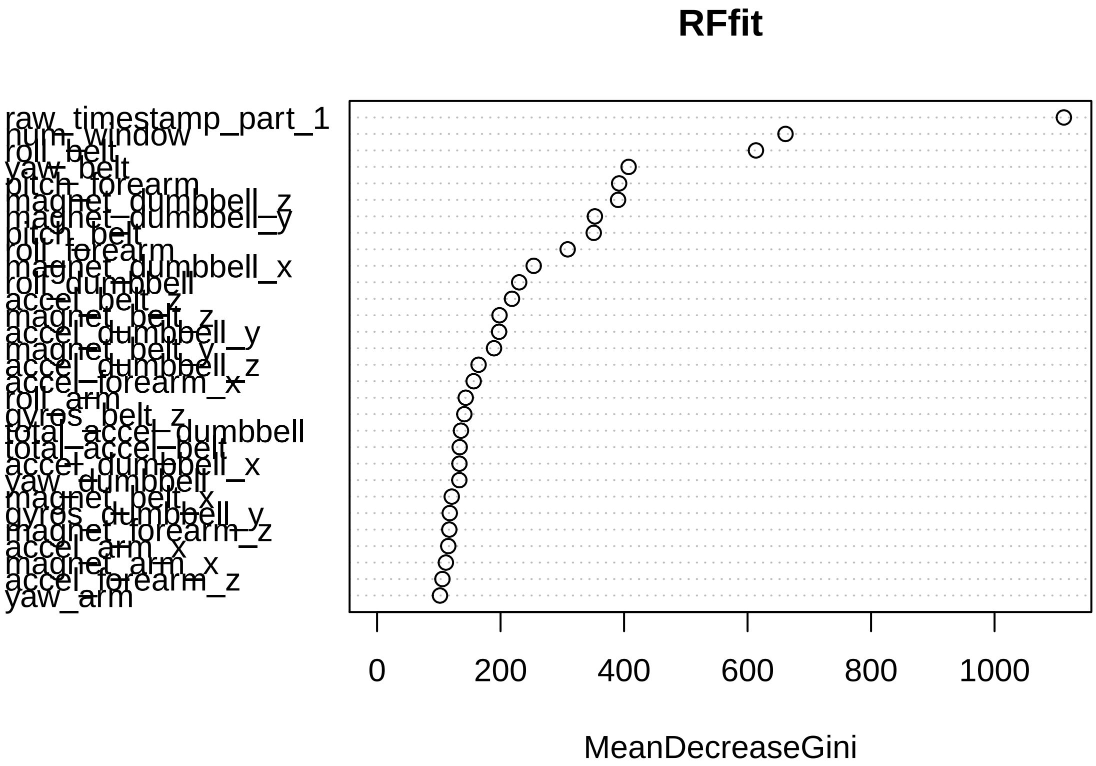
```r
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```r
RFfit <- randomForest(classe ~ ., data=myTrain, method="class")
```

The importance of the predictors to the accuracy of the model can be visualized using *varImpPlot* command.

```r
varImpPlot(RFfit)
```



This indicates that a far simpler model could be formaulated by using just the first few predictors listed here. Here, simplification of the model is not pursued as it isn't required for this assignemnt.

Comparison of model performance

To evaluate the accuracy of the models, their performance on the *test* set is computed

```r
predictDT <- predict(DTfit, myTest, type = "class")
predictRF <- predict(RFfit, myTest, type = "class")
```

Now we look at the respective confusion matrices for the two models

```r
confusionMatrix(predictDT, myTest$classe)$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2082  159   16   23   20
##          B   71 1096   51   10   90
##          C    5   61 1172   85  107
##          D   58  120   39 1022  115
##          E   16   82   90  146 1110
```

```
confusionMatrix(predictRF, myTest$classe)$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    3    0    0    0
##          B    0 1515    9    0    0
##          C    0    0 1359    1    0
##          D    0    0    0 1285    3
##          E    0    0    0    0 1439
```

The Random Forest algorithm appears to be far more accurate, when used to evaluate out of sample data. This can be confirmed:

```
confusionMatrix(predictDT, myTest$classe)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.261535e-01   7.800021e-01   8.175834e-01   8.344805e-01   2.844762e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   1.321672e-31
```

```
confusionMatrix(predictRF, myTest$classe)$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9979607      0.9974205      0.9966905      0.9988340      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

The out of sample error can be readily calculated.

```
error <- 1 - confusionMatrix(predictRF, myTest$classe)$overall[1]
names(error) <- "Out of sample error"
error
```

```
## Out of sample error
##         0.002039256
```

Conclusions

Two models were developed to predict the qualitative exercise behaviour of 6 individuals. The Random Forest algorithm was shown to be far superior, as measured by its accuracy on the test data.

Appendix

In addition the model here is also used to predict the results for 20 test cases. This requires modifying the levels in one of the predictors

```
levels(test$new_window) <- levels(train$new_window)
answers <- predict(RFfit, newdata = test)
answers
```

```
##  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
```

Now the predictions can be converted into individual files using the code provided on the project submission page.