Practical Machine Learning - Prediction Assignment Write Up

Genevieve Beart

13 January 2020

Executive Summary

In this project, we look at dumbbell bicep curl data from various senors to try to the predict different ways the curls were undertaken. Using a training data set, we built an algorithm to predict the way the exercise was undertaken. A random forest algorithm was applied as it is generally suitable when you have a large number of variable, is not overally influenced by outliers and selects the most important variables to use. To cross validate the algorithm, we applied this model a partitioned component of the data set to test the accuracy, highlighting the algorithm was 99.66% accurate. Finally, we applied the identified algorithm to the testing data set.

Data Preparation

We start by initiating relevant packages, loading and reviewing the training and testing dataset:-

```
library(knitr)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(e1071)
library(rpart)
library(rpart.plot)
##load the training dataset and identify NAs:
training <- read.csv("pml-training.csv", header = TRUE,sep = ",",na.strings = c("#DIV/0!","NA", ""))
 ##load the testing dataset and identify NAs:
testing <- read.csv("pml-testing.csv", header = TRUE,sep = ",",na.strings = c("#DIV/0!","NA", ""))

#Review the training data:
dim(training)
```

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

```
table(training$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
#Reviewing the testing data:
dim(testing)
```

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

In this data set, we can see that we have 19622 observations of 160 variables. In this assignment we want to use a combination of these variables to predict the manner in which each exercise was undertaken, i.e., the **classe** variable. In contrast, the testing data set has only 20 observations with 160 variables. The aim is to use the testing data set to build and validate a model that can project the **classe** of the 20 observations in the testing data set.

classe may equal A, B, C, D or E. Based on the notes from the Human Activity Webpage, these factors correspond to different ways the dumbbell bicep curls were undertaken: * A = exactly according to the specification; * B = throwing the elbows to the front; * C = lifting the dumbbell only halfway; * D = lowering the dumbbell only halfway; and * E = throwing the hips to the front.

As shown by the table above, we have several thousand observations for each of the five **classe** options in the testing set.

Before building a predictive model, we can drop data which does not help the predicition (i.e., the first seven columns) and remove columns which only include NAs - we will do this on both data sets:

```
## Clean the testing and training data
# Remove first 7 columns
training1<-training[,-c(1,2,3,4,5,6,7)]
testing1<-testing[,-c(1,2,3,4,5,6,7)]
#Remove columns with only NAs
training2<-training1[,colSums(is.na(training1)) == 0]
testing2<-testing1[,colSums(is.na(testing1)) == 0]
#Resummarise the number of dimensions in each data set:
dim(training2)
```

```
## [1] 19622    53
```

```
dim(testing2)
```

```
## [1] 20 53
```

As can be seen, this cleaning has reduced the number of variables from 160 to 53 in both data sets while maintaining all of the observations.

In order to build and test potential models, we start with the training data **training2** and separate this data into a training and validation data set. We will use 70% of the data to train the algorithm and the remaining 30% to test it. This remaining 30% of the data is what will be used as cross-validation of the identified algorithm.

```
## Split the data into a training and validation data set
set.seed(1973) #set the seed so analysis can be replicated
training3 <- createDataPartition(training2$classe, p=0.7, list=FALSE)
train_data <- training2[training3,] ## the final training data set to be used
val_data <- training2[-training3,] ## the final validation data set to be used for cross-validation
```

Model Training

We will use a **random forest** algorithm to fit a model to the test data. The random forest algorithm has been selected as it is generally suitable when you have a large number of variables (53 in our case), is not overally influenced by outliers and selects the most important variables to use.

```
## Apply random forest algorithm to the partitioned training data set
training_modelRF <- randomForest(classe~., data=train_data)
training_modelRF
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = train_data)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.58%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    3    1    0    0 0.001024066
## B   14 2638    6    0    0 0.007524454
## C    0   16 2375    5    0 0.008764608
## D    0    0   25 2224    3 0.012433393
## E    0    0    1    6 2518 0.002772277
```

As can be seen from the output of fitting this model against the test dat, there is an estimated out of sample (OOS) error rate of 0.58% using the testing data set.

The decision tree of the random forest model used is shown in appendix **A.1 - Decision Tree Visualisation**.

Model Cross-Validation

We can now cross-validate this model using the split validation data set **val_data**. We apply the algorithm to **val_data** and view the results:

```
## Apply algorithm to the partitioned validation data set
val_output <- predict(training_modelRF, newdata=val_data)
confusionMatrix(val_data$classe, val_output)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    3    0    0    0
##          B    6 1133    0    0    0
##          C    0    5 1020    1    0
##          D    0    0    3  961    0
##          E    0    0    0    2 1080
##
## Overall Statistics
##
##                Accuracy : 0.9966
##                  95% CI : (0.9948, 0.9979)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9957
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9930   0.9971   0.9969   1.0000
## Specificity            0.9993   0.9987   0.9988   0.9994   0.9996
## Pos Pred Value         0.9982   0.9947   0.9942   0.9969   0.9982
## Neg Pred Value         0.9986   0.9983   0.9994   0.9994   1.0000
## Prevalence             0.2850   0.1939   0.1738   0.1638   0.1835
## Detection Rate         0.2839   0.1925   0.1733   0.1633   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9979   0.9959   0.9979   0.9981   0.9998
```

As can be seen from the outputs above, the accuracy of this algorithm on the validation data set if 99.66%.

Run the Model on the Test Data

Finally, we need to run the algorithm on the test data so we can get the outputs for the quiz.

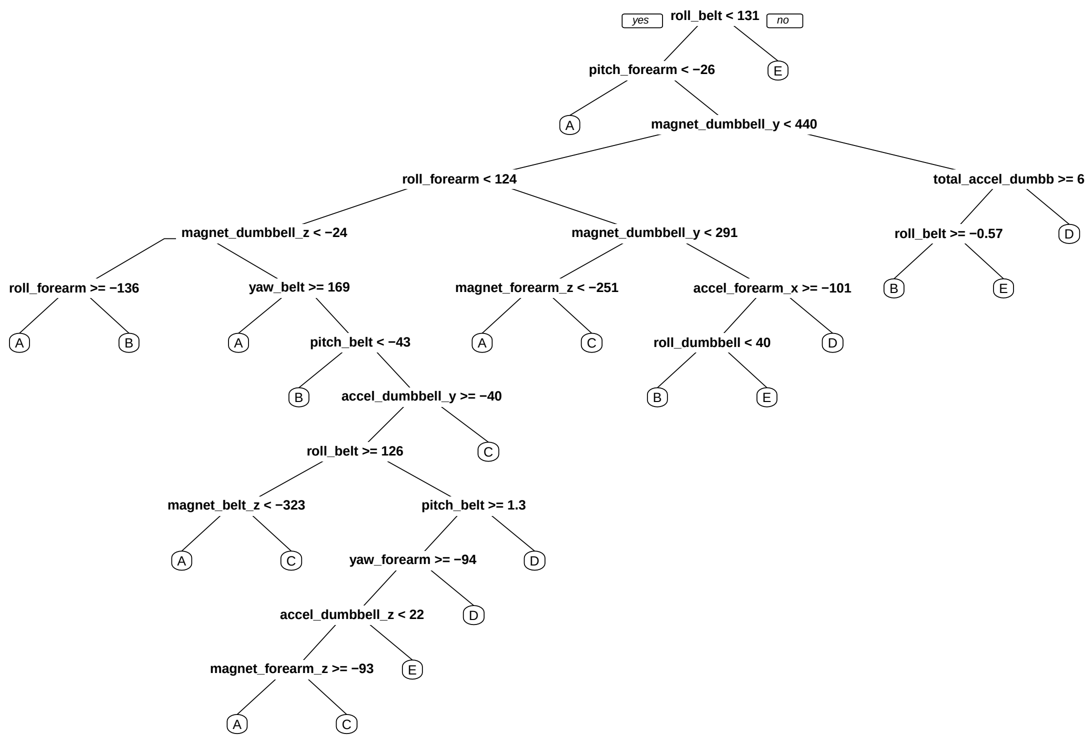
```
## Apply algorithm to the test data set
test_output <- predict(training_modelRF, newdata=testing2)
test_output
```

```
##  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
```

Appendix

A.1 - Decision Tree Visualisation

```
tree_output <- rpart(classe ~ ., data=train_data, method="class")
prp(tree_output) #Plot the decision tree
```



yes
roll_belt < 131
no
pitch_forearm < −26
E
A
magnet_dumbbell_y < 440
roll_forearm < 124
total_accel_dumbb >= 6
magnet_dumbbell_z < −24
magnet_dumbbell_y < 291
roll_belt >= −0.57
D
roll_forearm >= −136
yaw_belt >= 169
magnet_forearm_z < −251
accel_forearm_x >= −101
B
E
A
B
A
pitch_belt < −43
A
C
roll_dumbbell < 40
D
B
accel_dumbbell_y >= −40
B
E
roll_belt >= 126
C
magnet_belt_z < −323
pitch_belt >= 1.3
A
C
yaw_forearm >= −94
D
accel_dumbbell_z < 22
D
magnet_forearm_z >= −93
E
A
C