Analysis of Fitness Trackers with Barbell Movement

Introduction

This analysis uses data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants performing barbell lifts correctly and incorrectly in 5 different ways. The data is available from http://groupware.les.inf.puc-rio.br/har. The analysis will use the data from the sensors to predict whether the barbell lifts were perfomred correctly or not.

Reading the Data

There are 19622 cases in the training dataset with 160 variables while the testing dataset has 20 cases with the same 160 variables.

```
training <- read.csv("pml-training.csv")
dim(training)
```

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

```
testing <- read.csv("pml-testing.csv")
dim(testing)
```

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

Data Preparation

There are many variables with blanks in the dataset which are set to NA.

```
training[training==""] <- NA
```

There are 100 variables with 19216 NA values which are of no use for the analysis and are removed.

```
training <- training[,colSums(is.na(training))==0]
dim(training)
```

```
## [1] 19622    60
```

The non-informative columns X, username, the 3 timestamp columns and the 2 window columns are removed.

```
training <- training[,-c(1:7)]
dim(training)
```

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

Splitting the Data

The data is now split into two parts, the training data and the test data to be used to measure the out of sample error.

```r
library(caret)
set.seed(1)
inTrain <- createDataPartition(training$classe, p=0.60, list=F)
trainData <- training[inTrain, ]
testData <- training[-inTrain, ]
```

Model Discussion

This problem is a classification problem predicting 5 classes in the data. Classification problems are best dealt with trees, bagged trees or random forests. Since these models are good with non-linear and non-parametric data there is no obvious need for transforming or pre-processing the data - the missing values are also so extreme that an imputation will not improve the situation.

Classification Tree

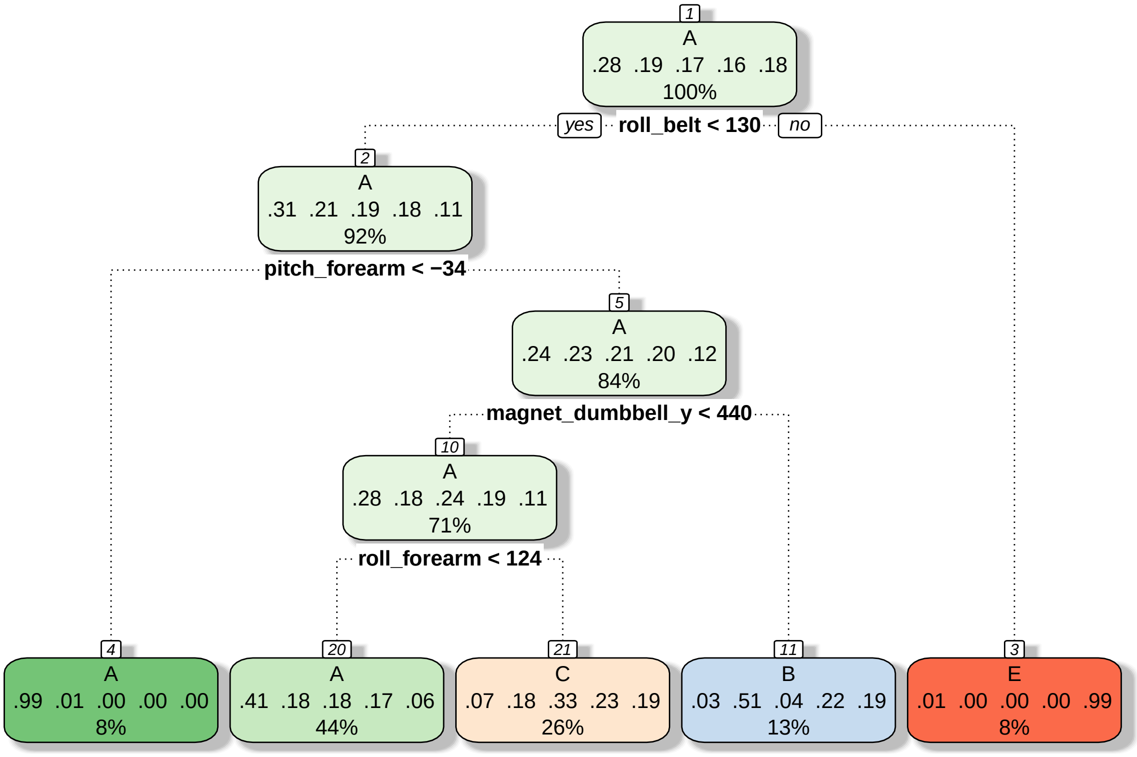
As a first step a classification tree is fit with an accuracy of 0.52.

```r
set.seed(1)
mod1 <- train(classe ~ ., data=trainData, method="rpart")
mod1
```

```
## CART
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa       Accuracy SD  Kappa SD
##   0.03500237  0.5173990  0.37434487  0.02527923   0.04117744
##   0.05991932  0.4329676  0.23719982  0.06627214   0.10965036
##   0.11568581  0.3227539  0.05924185  0.04209128   0.06297990
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03500237.
```

The tree shows that `roll_belt`, `pitch_Forearm`, `magnet_dumbbell_y` and `roll_forearm` are the key variables but is still a very simplicstic model.

```r
library(rattle)
par(mar=c(1,1,1,1))
fancyRpartPlot(mod1$finalModel)
```



Bagged Tree

A bagged tree with the default of 10 bootstraps takes much longer to compute but also has a much higher accuracy of 0.97.

```
set.seed(1)
mod2 <- train(classe ~ ., data=trainData, method="treebag")
mod2
```

```
## Bagged CART
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results
##
##   Accuracy   Kappa      Accuracy SD  Kappa SD
##   0.9732414  0.9661433  0.006100222  0.007718177
##
##
```

The Prediction table is as follows:

```r
table(trainData$classe,predict(mod2, trainData))
```

```
##      
##        A    B    C    D    E
##   A 3348    0    0    0    0
##   B    0 2279    0    0    0
##   C    0    0 2054    0    0
##   D    0    0    1 1929    0
##   E    0    0    0    0 2165
```

Random Forest

A random forest is the most accurate with 0.99. Here 10-fold corssvalidation was chosen since the sample is large enough and 100 trees were made as this already takes quite some time.

```r
set.seed(1)
mod3 <- train(classe ~ ., data=trainData, method="rf",
                 trControl=trainControl(method="cv", 10), ntree=100)
mod3
```

```
## Random Forest 
## 
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E' 
## 
## No pre-processing
## Resampling: Cross-Validated (10 fold) 
## Summary of sample sizes: 10599, 10598, 10599, 10599, 10597, 10599, ... 
## Resampling results across tuning parameters:
## 
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD   
##    2    0.9878582  0.9846383  0.004270313  0.005405540
##   27    0.9896408  0.9868949  0.002551376  0.003228863
##   52    0.9814870  0.9765802  0.005173595  0.006548168
## 
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```r
table(trainData$classe,predict(mod3, trainData))
```

```
##      
##        A    B    C    D    E
##   A 3348    0    0    0    0
##   B    0 2279    0    0    0
##   C    0    0 2054    0    0
##   D    0    0    0 1930    0
##   E    0    0    0    0 2165
```

Accuracy Measures

So far the accuracy measures have been on the training data. The idea is to check the accuracy andout of sample error on the test dataset. The model that will be used is the random forest as this seems to predict extremely well.

```
set.seed(1)
predict_mod3 <- predict(mod3, testData)
confusionMatrix(testData$classe, predict_mod3)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2224    3    4    0    1
##          B    9 1502    7    0    0
##          C    0    5 1358    5    0
##          D    0    0   18 1268    0
##          E    0    0    2    7 1433
##
## Overall Statistics
##
##                Accuracy : 0.9922
##                  95% CI : (0.99, 0.994)
##     No Information Rate : 0.2846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9902
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9960   0.9947   0.9777   0.9906   0.9993
## Specificity            0.9986   0.9975   0.9985   0.9973   0.9986
## Pos Pred Value         0.9964   0.9895   0.9927   0.9860   0.9938
## Neg Pred Value         0.9984   0.9987   0.9952   0.9982   0.9998
## Prevalence             0.2846   0.1925   0.1770   0.1631   0.1828
## Detection Rate         0.2835   0.1914   0.1731   0.1616   0.1826
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9973   0.9961   0.9881   0.9939   0.9989
```

The accuracy on the testing data is **0.9922** and the out of sample error is thus estimated at **0.0078**.

Predicting for Test Data Set

Now, we apply the model to the original testing data set downloaded from the data source.

```
predict_20 <- predict(mod3, testing[,colnames(trainData)[-53]])
predict_20
```

```
##  [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
```