Practical Machine Learning: Assignment

Synopsis

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 (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) are available from this website.

Dataset

The training data (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) and the test data (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv) from the provided links and the data (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) for this project come from the provided source.

Loading the data and the packages

We load the data as below:

```r
#Set the URL for the downloads
UrlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
UrlTest  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
#Download the datasets
training <- read.csv(url(UrlTrain))
testing  <- read.csv(url(UrlTest))
```

The packages *knitr*, *caret*, *rpart*, *rpart.plot*, *rattle*, *randomForest*, *e1071*, *gridExtra* and *gbm* have to be loaded.

Data partition and cleaning

Regarding the course Practical Machine Learning, the data will be divided into a training data set and a testing data set with a rate 70%/30%.

```
#Make the research reproducible
set.seed(12345)
partition  <- createDataPartition(training$classe, p=0.7, list=FALSE)
training_set <- training[partition, ]
test_set  <- training[-partition, ]
```

These variables have a lot of NA, that can be removed as below:

```
#Variables with Nearly Zero Variance are removed
NZV <- nearZeroVar(training_set)
training_set <- training_set[, -NZV]
test_set  <- test_set[, -NZV]
remove_NA    <- sapply(training_set, function(x) mean(is.na(x))) > 0.95
training_set <- training_set[, remove_NA==FALSE]
test_set  <- test_set[, remove_NA==FALSE]

#ID variables (from 1 to 5) are removed
training_set <- training_set[, -(1:5)]
test_set  <- test_set[, -(1:5)]
dim(training_set)
```

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

```
dim(test_set)
```

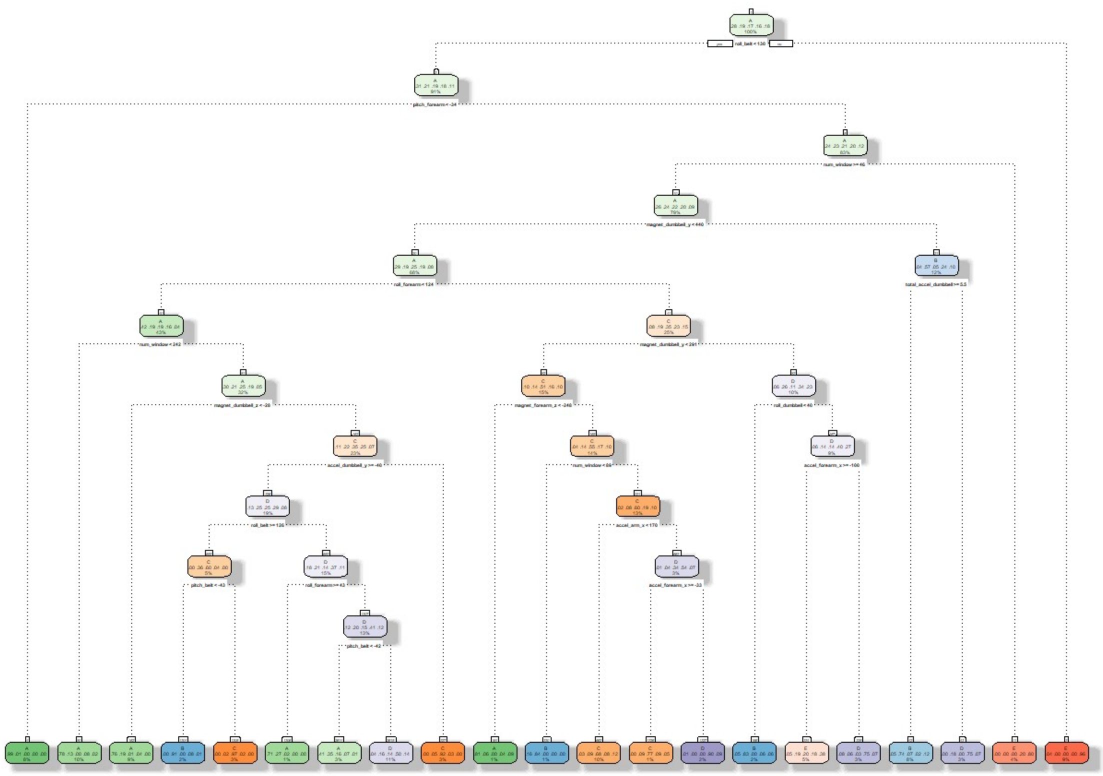
```
## [1] 5885   54
```

The number of variables for the analysis has been reduced to 54 only.

Building prediction models

1. Decision Tree model prediction

```
model_decision_tree <- rpart(classe ~ ., data = training_set, method="class", contro
l = rpart.control(method = "cv", number = 10))
fancyRpartPlot(model_decision_tree)
```



Rattle 2019-avr.-24 19:15:43 maxim

We do not expect a very high accuracy.

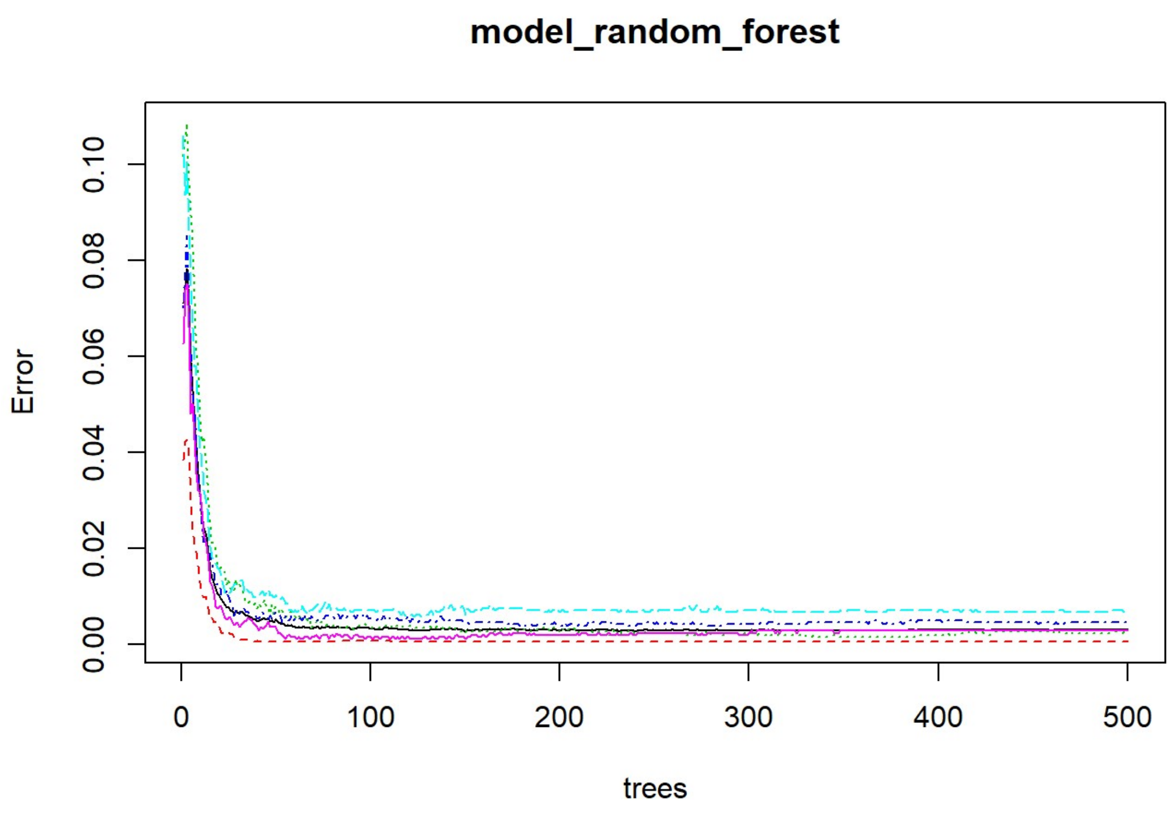
```
prediction_decision_tree <- predict(model_decision_tree, test_set, type = "class")
confusion_matrix_DT <- confusionMatrix(prediction_decision_tree, test_set$classe)
confusion_matrix_DT
```

```
## 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
```

The accuracy reaches 73.68%.

2. Random Forest model prediction

```
model_random_forest <- randomForest(classe ~ ., data = training_set, method = "rf",
importance = T, trControl = trainControl(method = "cv", classProbs=TRUE,savePredicti
ons=TRUE,allowParallel=TRUE, number = 10))
plot(model_random_forest)
```



Using random forest, the out of sample error is small as it is expected.

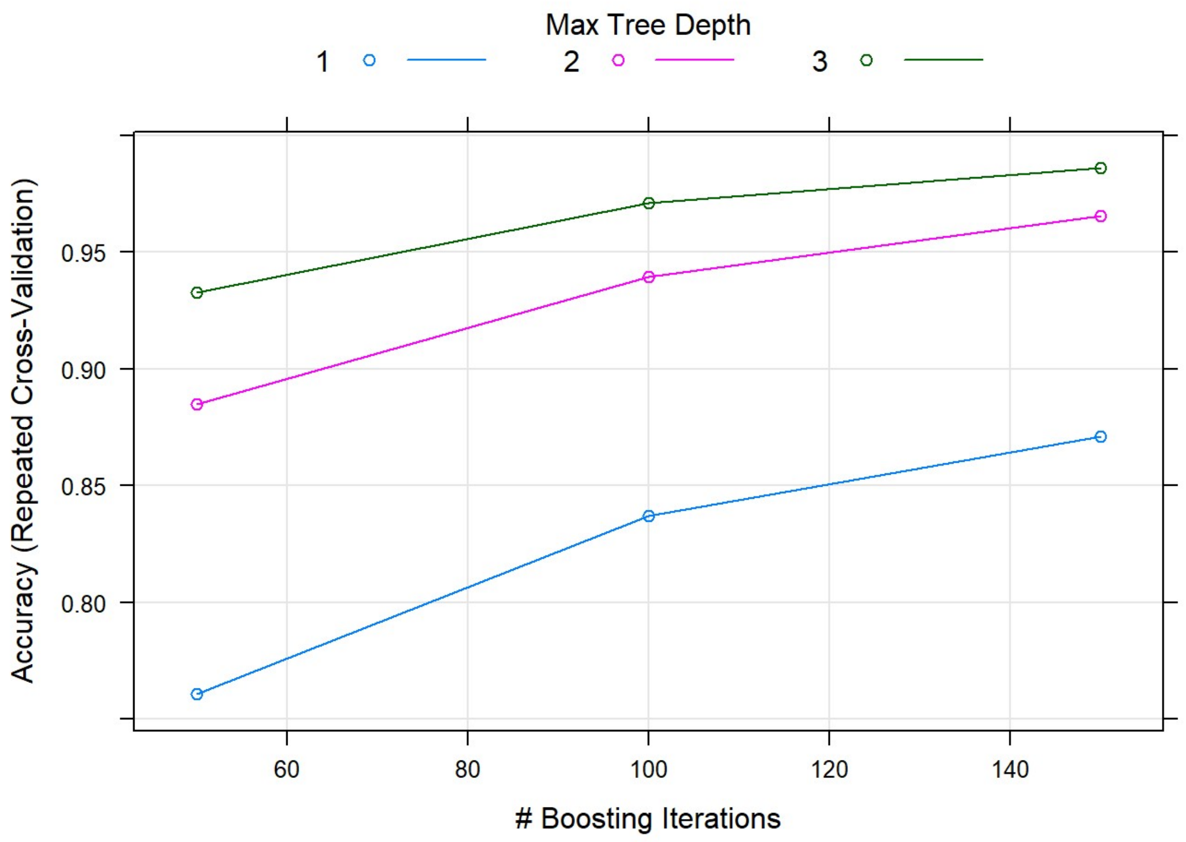
```
prediction_random_forest <- predict(model_random_forest, newdata=test_set)
confusion_matrix_RF <- confusionMatrix(prediction_random_forest, test_set$classe)
confusion_matrix_RF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    6    0    0    0
##          B    0 1133    6    0    0
##          C    0    0 1020   12    0
##          D    0    0    0  952    4
##          E    0    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.9952
##                  95% CI : (0.9931, 0.9968)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.994
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9942   0.9876   0.9963
## Specificity            0.9986   0.9987   0.9975   0.9992   1.0000
## Pos Pred Value         0.9964   0.9947   0.9884   0.9958   1.0000
## Neg Pred Value         1.0000   0.9987   0.9988   0.9976   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1925   0.1733   0.1618   0.1832
## Detection Prevalence   0.2855   0.1935   0.1754   0.1624   0.1832
## Balanced Accuracy      0.9993   0.9967   0.9958   0.9934   0.9982
```

The accuracy reaches 99.52%.

3. Boosting model prediction

```
control_boosting <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
model_boosting <- train(classe ~ ., data=training_set, method = "gbm", trControl = c
ontrol_boosting, verbose = FALSE)
plot(model_boosting)
```



```
prediction_boosting <- predict(model_boosting, test_set)
confusion_matrix_boosting <- confusionMatrix(prediction_boosting, test_set$classe)
confusion_matrix_boosting
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674   16    0    3    0
##          B    0 1103   16    7    5
##          C    0   17 1005   14    4
##          D    0    3    5  939   15
##          E    0    0    0    1 1058
##
## Overall Statistics
##
##                Accuracy : 0.982
##                  95% CI : (0.9783, 0.9852)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9772
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9684   0.9795   0.9741   0.9778
## Specificity            0.9955   0.9941   0.9928   0.9953   0.9998
## Pos Pred Value         0.9888   0.9752   0.9663   0.9761   0.9991
## Neg Pred Value         1.0000   0.9924   0.9957   0.9949   0.9950
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1874   0.1708   0.1596   0.1798
## Detection Prevalence   0.2877   0.1922   0.1767   0.1635   0.1799
## Balanced Accuracy      0.9977   0.9812   0.9862   0.9847   0.9888
```

The accuracy reaches 98.2%

Prediction with the best model

The Random Forest model provides the best accuracy. Hence we expect to get a perfect prediction as below:

```
prediction_test <- predict(model_random_forest, newdata=testing)
prediction_test
```

```
##  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
```

Finally, we got a perfect score in the quiz.