Practical Machine Learning Project - Human Activity Recognition Report by Florin Toth

Introduction

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, our 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).

The goal of this project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set.

Data Cleaning

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)

```
trainData <- read.csv("pml-training.csv",header=T)
testData <- read.csv("pml-testing.csv",header=T)
nrow(trainData);ncol(trainData)
```

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

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

```
nrow(testData);ncol(testData)
```

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

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

The training data set contains 19622 observations and 160 variables, while the testing data set contains 20 observations and 160 variables.

Using *summary(trainData)* and *summary(testData)* for data exploration we notice there are many missing obeservations we need to remove.

```
trainData <- trainData[, colSums(is.na(trainData)) == 0]
testData <- testData[, colSums(is.na(testData)) == 0]
nrow(trainData);ncol(trainData)
```

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

```
## [1] 93
```

```
nrow(testData);ncol(testData)
```

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

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

The training data set now contains 19622 observations and 93 variables, while the testing data set now contains 20 observations and 60 variables.

We also notice some variables that have no influence to the accelerometer measurements. These happen to be the first 7 variables that we can remove from both train and test data sets.

```
trainData <- trainData[,-c(1:7)]
testData <- testData[,-c(1:7)]
nrow(trainData);ncol(trainData)
```

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

```
## [1] 86
```

```
nrow(testData);ncol(testData)
```

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

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

The training data set now contains 19622 observations and 86 variables, while the testing data set now contains 20 observations and 53 variables.

In order to simplify the model we also remove all variables that are not numeric except for the “classe” variable from the training data set.

```
classe <- trainData$classe # saving the "classe" variable
trainData <- trainData[, sapply(trainData, is.numeric)]
trainData$classe <- classe # adding it back
nrow(trainData);ncol(trainData)
```

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

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

```
nrow(testData);ncol(testData)
```

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

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

Now the training data set contains 19622 observations and 53 variables, while the testing data set contains 20 observations and 53 variables with the 5-levels “classe” factor variable being kept in the training data set.

Data slicing

Now we have a clean data set for training but we also need a subset of this data set for validation. Therefore we split the training dataset using the usual 70%/30% ratios to get a validation data set.

```
library(caret)
```

```
set.seed(12345)
inTrain <- createDataPartition(trainData$classe, p=0.70, list=FALSE)
training <- trainData[inTrain, ]
testing <- trainData[-inTrain, ]
```

Data Modelling

We will fit a predictive model using *random forest* algorithm on all variables since it gives the most accurate results even though it is pretty slow and quite difficult to describe. We will also use *k-fold cross validation* with 5 folds, based on the 5-levels “classe” factor variable we use. We also cap the number of proccessed trees to 100.

```
library(randomForest)
```

```
modelRF <- train(classe ~ ., data=training, method="rf",
                 trControl=trainControl(method="cv", 5), ntree=100)
modelRF
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 10989, 10990, 10990, 10989, 10990
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD   Kappa SD
##    2    0.9905366  0.9880278  0.0025602854  0.0032397549
##   27    0.9898814  0.9871993  0.0018446597  0.0023334534
##   52    0.9855136  0.9816729  0.0006991841  0.0008851263
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

We get an accuracy of: 0.9905

```
modelRF$results[1,2]
```

```
## [1] 0.9905366
```

and an in sample error of: 0.009463

```
1-modelRF$results[1,2]
```

```
## [1] 0.009463414
```

Based on this model we test its predictive performance on the validation data set.

```
predictRF <- predict(modelRF, testing)
confusionMatrix(testing$classe, predictRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    3    0    0    0
##          B   13 1122    4    0    0
##          C    0   16 1007    3    0
##          D    0    0   25  939    0
##          E    0    0    0    3 1079
##
## Overall Statistics
##
##                Accuracy : 0.9886
##                  95% CI : (0.9856, 0.9912)
##     No Information Rate : 0.2862
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9856
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9923   0.9833   0.9720   0.9937   1.0000
## Specificity            0.9993   0.9964   0.9961   0.9949   0.9994
## Pos Pred Value         0.9982   0.9851   0.9815   0.9741   0.9972
## Neg Pred Value         0.9969   0.9960   0.9940   0.9988   1.0000
## Prevalence             0.2862   0.1939   0.1760   0.1606   0.1833
## Detection Rate         0.2839   0.1907   0.1711   0.1596   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9958   0.9899   0.9840   0.9943   0.9997
```

We get an accuracy of: 0.9886

```
confusionMatrix(testing$classe, predictRF)$overall[1]
```

```
##  Accuracy
## 0.9886151
```

and an out of sample error of: 0.011385

```
1-as.numeric(confusionMatrix(testing$classe, predictRF)$overall[1])
```

```
## [1] 0.01138488
```

We notice that the out of sample error is slightly higher than the in sample error. This indicates a possible over-fitting model with very high accuracy that captures both the signal and the noise. This is one of the drawbacks of a random forrest model but since the sample error difference is not so big we conclude that overall the model is very good.

Predicting on the provided dataset

We apply the model to testData, the cleaned provided testing dataset. We noticed the testing dataset has a last column named “problem_id” which we need to remove since it is not related to our analysis.

```
prediction <- predict(modelRF, testData[, -length(testData)])
prediction
```

```
##  [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
```