Practical Machine Learning Course Project

A.Nikitenko

August 4, 2016

Introduction

This document is a course project report in "Practical Machine Learning" provided by J.Hopkins University using Coursera on-line learning tool. The report presents results of prediction model developed and the reasoning behind it. The problem and the date has been taken from http://groupware.les.inf.puc-rio.br/har.

Data

Data represents human activity of 6 young people performing physical exercises. The description of the data provided by the researchers conducting the data acquisition is as follows:

"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)."

For the project purposes data is split into training and test data sets being stored in the same folders as the R code executing the data analysis.

Data reading R code:

```
rawTraining <- read.csv("pml-training.csv", header = TRUE, sep = ",", quote = "\"", dec = ".", na.strings = "NA")
rawTesting <- read.csv("pml-testing.csv", header = TRUE, sep = ",", quote = "\"", dec = ".", na.strings = "NA")

dim(rawTraining)

## [1] 19622   160

dim(rawTesting)

## [1]  20 160

names(rawTraining)

##   [1] "X"                       "user_name"
##   [3] "raw_timestamp_part_1"    "raw_timestamp_part_2"
##   [5] "cvtd_timestamp"          "new_window"
##   [7] "num_window"              "roll_belt"
##   [9] "pitch_belt"              "yaw_belt"
##  [11] "total_accel_belt"        "kurtosis_roll_belt"
##  [13] "kurtosis_picth_belt"     "kurtosis_yaw_belt"
##  [15] "skewness_roll_belt"      "skewness_roll_belt.1"
##  [17] "skewness_yaw_belt"       "max_roll_belt"
##  [19] "max_picth_belt"          "max_yaw_belt"
##  [21] "min_roll_belt"           "min_pitch_belt"
##  [23] "min_yaw_belt"            "amplitude_roll_belt"
##  [25] "amplitude_pitch_belt"    "amplitude_yaw_belt"
##  [27] "var_total_accel_belt"    "avg_roll_belt"
##  [29] "stddev_roll_belt"        "var_roll_belt"
##  [31] "avg_pitch_belt"          "stddev_pitch_belt"
##  [33] "var_pitch_belt"          "avg_yaw_belt"
##  [35] "stddev_yaw_belt"         "var_yaw_belt"
##  [37] "gyros_belt_x"            "gyros_belt_y"
##  [39] "gyros_belt_z"            "accel_belt_x"
##  [41] "accel_belt_y"            "accel_belt_z"
##  [43] "magnet_belt_x"           "magnet_belt_y"
##  [45] "magnet_belt_z"           "roll_arm"
##  [47] "pitch_arm"               "yaw_arm"
```

```
##  [49] "total_accel_arm"          "var_accel_arm"
##  [51] "avg_roll_arm"             "stddev_roll_arm"
##  [53] "var_roll_arm"             "avg_pitch_arm"
##  [55] "stddev_pitch_arm"         "var_pitch_arm"
##  [57] "avg_yaw_arm"              "stddev_yaw_arm"
##  [59] "var_yaw_arm"              "gyros_arm_x"
##  [61] "gyros_arm_y"              "gyros_arm_z"
##  [63] "accel_arm_x"              "accel_arm_y"
##  [65] "accel_arm_z"              "magnet_arm_x"
##  [67] "magnet_arm_y"             "magnet_arm_z"
##  [69] "kurtosis_roll_arm"        "kurtosis_picth_arm"
##  [71] "kurtosis_yaw_arm"         "skewness_roll_arm"
##  [73] "skewness_pitch_arm"       "skewness_yaw_arm"
##  [75] "max_roll_arm"             "max_picth_arm"
##  [77] "max_yaw_arm"              "min_roll_arm"
##  [79] "min_pitch_arm"            "min_yaw_arm"
##  [81] "amplitude_roll_arm"       "amplitude_pitch_arm"
##  [83] "amplitude_yaw_arm"        "roll_dumbbell"
##  [85] "pitch_dumbbell"           "yaw_dumbbell"
##  [87] "kurtosis_roll_dumbbell"   "kurtosis_picth_dumbbell"
##  [89] "kurtosis_yaw_dumbbell"    "skewness_roll_dumbbell"
##  [91] "skewness_pitch_dumbbell"  "skewness_yaw_dumbbell"
##  [93] "max_roll_dumbbell"        "max_picth_dumbbell"
##  [95] "max_yaw_dumbbell"         "min_roll_dumbbell"
##  [97] "min_pitch_dumbbell"       "min_yaw_dumbbell"
##  [99] "amplitude_roll_dumbbell"  "amplitude_pitch_dumbbell"
## [101] "amplitude_yaw_dumbbell"   "total_accel_dumbbell"
## [103] "var_accel_dumbbell"       "avg_roll_dumbbell"
## [105] "stddev_roll_dumbbell"     "var_roll_dumbbell"
## [107] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
## [109] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
## [111] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
## [113] "gyros_dumbbell_x"         "gyros_dumbbell_y"
## [115] "gyros_dumbbell_z"         "accel_dumbbell_x"
## [117] "accel_dumbbell_y"         "accel_dumbbell_z"
## [119] "magnet_dumbbell_x"        "magnet_dumbbell_y"
## [121] "magnet_dumbbell_z"        "roll_forearm"
## [123] "pitch_forearm"            "yaw_forearm"
## [125] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
## [127] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
## [129] "skewness_pitch_forearm"   "skewness_yaw_forearm"
## [131] "max_roll_forearm"         "max_picth_forearm"
## [133] "max_yaw_forearm"          "min_roll_forearm"
## [135] "min_pitch_forearm"        "min_yaw_forearm"
## [137] "amplitude_roll_forearm"   "amplitude_pitch_forearm"
## [139] "amplitude_yaw_forearm"    "total_accel_forearm"
## [141] "var_accel_forearm"        "avg_roll_forearm"
## [143] "stddev_roll_forearm"      "var_roll_forearm"
## [145] "avg_pitch_forearm"        "stddev_pitch_forearm"
## [147] "var_pitch_forearm"        "avg_yaw_forearm"
## [149] "stddev_yaw_forearm"       "var_yaw_forearm"
## [151] "gyros_forearm_x"          "gyros_forearm_y"
## [153] "gyros_forearm_z"          "accel_forearm_x"
## [155] "accel_forearm_y"          "accel_forearm_z"
## [157] "magnet_forearm_x"         "magnet_forearm_y"
## [159] "magnet_forearm_z"         "classe"
```

Due to report size limitations the data summary is not presented here, but few thing have to be emphasized:

1) The data set has 160 parameters including record number and class index - "classe";
2) Some of the parameters include many NAs;
3) A significant part of the parameters are not necessary for the questions being answered by this exercise. Therefore the data has to bee cleaned before application of machine learning methods.

Cleaning data

Since many parameters are aggregates of the others, they can be removed providing only the raw sensor data. The same can be done with parameters including mostly NAs. The necessary parameters are those starting with "gyros" - gyroscope readings, "accel" - accelerometer readings, "magnet" - magnetometer readings, "roll", "pitch" and "yaw" providing calculated static angles around axes. An finally it is necessary to add class field "classe"

```
library(PerformanceAnalytics)
```

```
initNames <- grepl("^accel|^gyros|^magnet|^roll|^pitch|^yaw|^classe",names(rawTraining))
initialData <- rawTraining[,initNames]
ValidationData <- rawTesting[, initNames]
```

Splitting data

The acquired training data has to split into training and testing part, which is done by the following code:

```
library(caret)

inTraining <- createDataPartition(initialData$classe, p = 0.7, list = FALSE)
training <- initialData[inTraining,]
testing <- initialData[-inTraining,]
```

Building prediction models

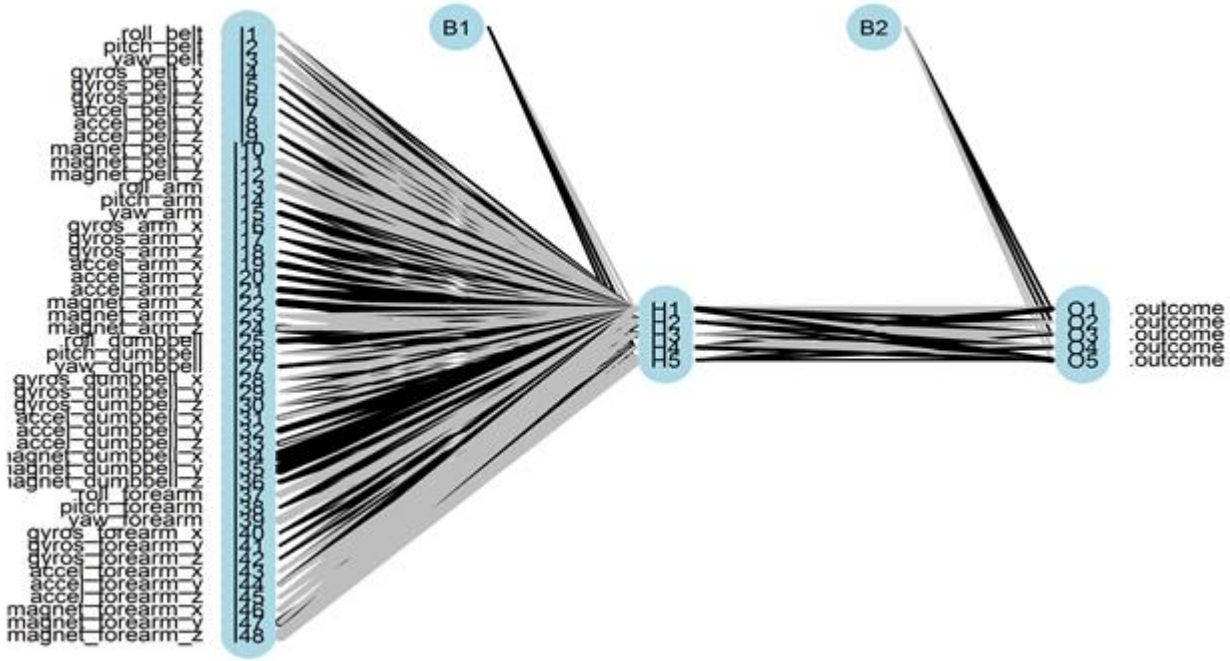
The prediction models are built using "caret" package using "decision trees", "random forest" and boosting techniques. The following code in addition check for saved models to reduce script running time if repeated.

```
set.seed(1234321)

if(file.exists("modFitDT.rda"))
{
    load("modFitDT.rda")
} else
{
   fitControl <- trainControl(method = 'cv', number=5)
   modFitDT <- train(classe ~ ., data = training, method = 'rpart', trControl = fitControl)
   save(modFitDT,file = "modFitDT.rda")
}
#modFitDT

if(file.exists("modFitRF.rda"))
{
    load("modFitRF.rda")
} else
{
   fitControl <- trainControl(method = 'cv', number=5)
   modFitRF <- train(classe ~ ., data = training, method = 'rf', trControl = fitControl, ntree = 100)
   save(modFitRF,file = "modFitRF.rda")
}
#modFitRF

if(file.exists("modFitBoosted.rda"))
{
    load("modFitBoosted.rda")
} else
{
    fitControl <- trainControl(method = 'cv', number=5)
    modFitBoosted <- train(classe ~ ., data = training, method = "gbm", trControl = fitControl, verbose=FALSE)
    save(modFitBoosted,file = "modFitBoosted.rda")
}
#modFitBoosted

if(file.exists("modFitNNET.rda"))
{
    load("modFitNNET.rda")
} else
{
    fitControl <- trainControl(method = 'cv', number=5)
    modFitNNET<- train(classe ~ ., data = training, method = "nnet", trControl = fitControl, verbose=FALSE)
    save(modFitNNET,file = "modFitNNET.rda")
}
#modFitNNET
```

Since the Artificial neuron networks has not bee a part of course topics, it is interesting to visualize the found model:

```
library(devtools)
source_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet_plot_update.r')
```

```
plot.nnet(modFitNNET)
```



Building predictions

This sections reports on the predictions done by each model. First it is necessary to collect predictions into separate data structures.

```
predDT <- predict(modFitDT, newdata = testing)

predRF <- predict(modFitRF, newdata = testing)

predBoost <- predict(modFitBoosted, newdata = testing)

predNNET <- predict(modFitNNET, newdata = testing)
```

Confusion matrix comparision

```
cmDT <- confusionMatrix(predDT, testing$classe)
cmRF <-confusionMatrix(predRF, testing$classe)
cmBoost <- confusionMatrix(predBoost, testing$classe)
cmNNET <- confusionMatrix(predNNET,testing$classe)

cmDT$table

##           Reference
## Prediction    A    B    C    D    E
##          A 1534  475  460  416  156
##          B   28  369   29  184  171
##          C  107  295  537  364  285
##          D    0    0    0    0    0
##          E    5    0    0    0  470

cmDT$overall['Accuracy']

##  Accuracy
## 0.4944775

cmRF$table

##           Reference
## Prediction    A    B    C    D    E
##          A 1674    2    0    0    0
##          B    0 1135    4    0    0
##          C    0    2 1020    5    0
```

```
##          D    0    0    2  959    1
##          E    0    0    0    0 1081

cmRF$overall['Accuracy']

##  Accuracy
## 0.9972812

cmBoost$table

##           Reference
## Prediction    A    B    C    D    E
##          A 1651   33    0    1    3
##          B   18 1071   24    2    9
##          C    3   30  989   23    2
##          D    1    4   12  935   11
##          E    1    1    1    3 1057

cmBoost$overall['Accuracy']

##  Accuracy
## 0.9690739

cmNNET$table

##           Reference
## Prediction    A    B    C    D    E
##          A 1440  380  911  542  475
##          B  161  685  101  257  451
##          C    0    3    7    0    6
##          D   72   67    7  165  127
##          E    1    4    0    0   23

cmNNET$overall['Accuracy']

##  Accuracy
## 0.3942226
```

Resampling

To ground the results a resampling is done and visualized.

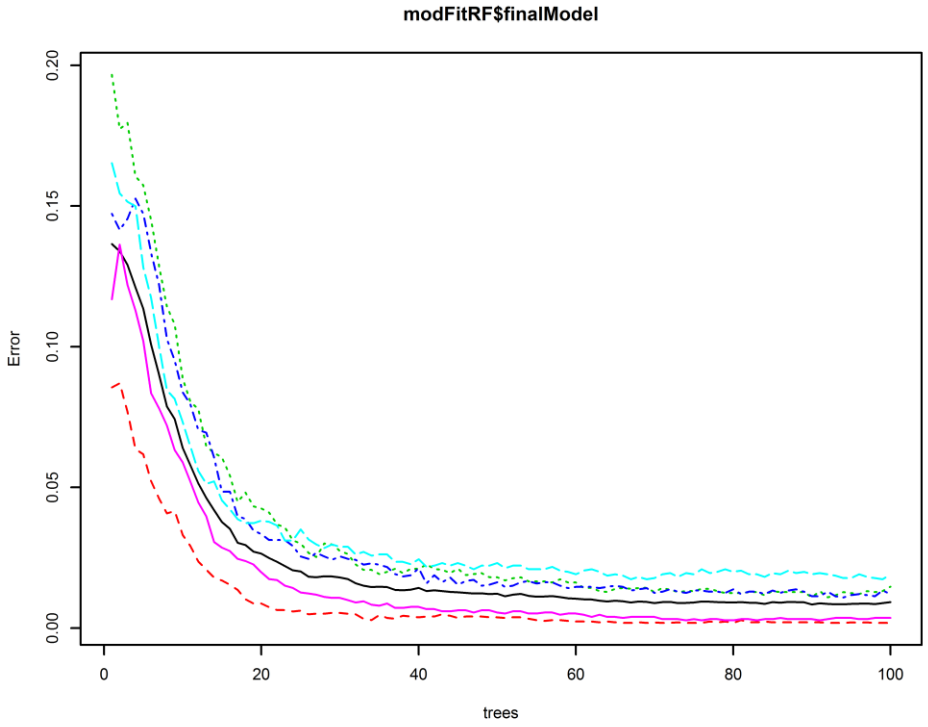
```
cvValues <- resamples(list(DT = modFitDT, RF = modFitRF, ANNet = modFitNNET))
summary(cvValues)

##
## Call:
## summary.resamples(object = cvValues)
##
## Models: DT, RF, ANNet
## Number of resamples: 5
##
## Accuracy
##         Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## DT    0.4863  0.4913 0.5033 0.5015  0.5078 0.5187    0
## RF    0.9873  0.9884 0.9905 0.9905  0.9913 0.9953    0
## ANNet 0.3481  0.4139 0.4318 0.4149  0.4378 0.4430    0
##
## Kappa
##         Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## DT    0.3289  0.3341 0.3526 0.3488  0.3567 0.3715    0
## RF    0.9839  0.9853 0.9880 0.9880  0.9889 0.9940    0
## ANNet 0.1614  0.2683 0.2765 0.2598  0.2883 0.3044    0
```

Since the random forest outperforms the other used models in terms of accuracy, which is justified by the confusion matrix and the estimated accuracy on the testing data, there is no need for further analysis of the remaining two models. Using other models like Support vector machines, general linear models or nonlinear regression because the accuracy of the Random Forest model is 99,8%, which is close to the maximum possible.

It is interesting to note that artificial neuron network's performance is significantly lower than performance of the other models. It might be related to overfitting.

Thereby the further analysis is done only by Random Forest model.

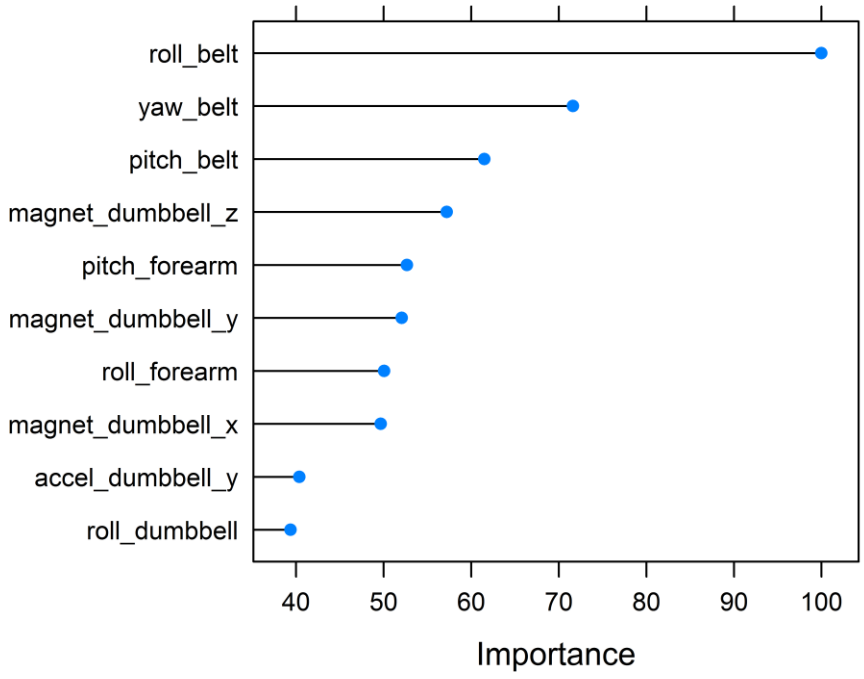
```
plot(modFitRF$finalModel)
```



Feature importance

For better description of the model a feature importance can be analyzed:

```
finalmodel <- modFitRF$finalModel
plot(varImp(modFitRF, scale = TRUE), top = 10)
```



Prediction

This section presents prediction results on the validation data set consisting of 20 samples, where each sample is indexes by problem_id field value.

```
predValid <- predict(modFitRF, newdata = ValidationData)
result <- data.frame(Problem_ID = ValidationData$problem_id, classe = predValid)
result
```

```
##    Problem_ID classe
## 1           1      B
## 2           2      A
## 3           3      B
## 4           4      A
## 5           5      A
## 6           6      E
## 7           7      D
## 8           8      B
## 9           9      A
## 10         10      A
## 11         11      B
## 12         12      C
## 13         13      B
## 14         14      A
## 15         15      E
## 16         16      E
## 17         17      A
## 18         18      B
## 19         19      B
## 20         20      B
```

Conclusion

The course project examined 4 different prediction models including artificial neuron networks. The models were built using cross validation and resampling. Unfortunately the initial data set included fields without meaningful content, what emphasizes the importance of data exploration before applying any further analysis. The fitted models differed in their accuracy significantly starting from 40% up to 99% with the lowest rate for artificial neuron networks, which seems to be interesting because the technique is used for highly non-linear problems. As it was noted during the course Random Forests has a good performance in many machine learning problems. The course project supports this thesis.