Title: Qualitative activity recognition of the unilateral dumbbell biceps curl, a course project of Practical Machine Learning, June 2014.

Author: Dmitry Borisoglebsky
Github: https://github.com/dborisog/pml_qsm

Introduction

This report finalizes a course project of Practical Machine Learning June 2014 [1], a course delivered by staff of John Hopkins University Jeff Leek, Brian Caffo, and Roger D. Peng via Coursera. This course project was inspired by a human activity recognition (HAR) project [2 , 3].

This HAR project was researching qualitative physical activity recognition, in different words how well an activity was executed. Six young adults were executing ten repetition of the unilateral dumbbell biceps curl in five manners: (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, (E) throwing the hips to the front, where B – E represent common mistakes. These executions were recorded via four sensors on the belt, arm, forearm, and dumbbell.

Data from this HAR project was delivered to the course students in two datasets, a training dataset of 19622 observations and 158 variables plus a manner 'classe' variable, and a testing dataset of 20 observations and 158 variables without 'classe' variable. The course project's objective is to predict the manner 'classe' of the testing dataset.

Methods

Specifics of the objective and the course limit methods could be applied by the author. The major part of the design of experiment was defined by the HAR project researchers, the HAR project's objective was set, exercise and types of mistakes were chosen, means to collect the data were selected, and data was collected. This course project also de facto narrows down the analytical means leaving methods that have been mentioned within the course: data preprocessing, modeling and prediction, analysis organization and sharing.

Data pre-processing. Training dataset is relatively large to split into training and evaluation datasets in proportion 75% and 25%, respectively. The number of variables is relatively large to assume that some variables might be excluded without loosing on variability and accuracy, near to zero variable and non-principle components were identified and filtered out.

Modeling and prediction. Having the manners 'classe' variable within the original training dataset and the qualitative nature of 'classe' variable focuses the author on classification and algorithms. Random forest is one of the most accurate classification algorithm, the author assumed that this algorithm would allow to create a model for the data on his desktop PC, and thereby selected random forests for modeling and prediction.

Reproduction. The course project assumes that the predicted results and report would be submitted on Coursera website for further evaluation, and the project's files would be available online via Github. The file organization structure proposed by the teaching personal of this and other data analysis MOOC from John Hopkins University was used in this project.

R scripting language and 'caret' package were major major preprocessing and analysis tools used in this project.

Results

Data preprocessing are possible to find in the appendix and '\Rcode\script.R' file of the github repository.

'confusionMatrix' function of 'caret' package produces the following output for predicted and actual values of 'classe' variable of evaluation dataset, see Frame 1.

```
> print(confusionMatrix(evalPredict, evaluation$classe))
Confusion Matrix and Statistics
Reference
 Prediction      A        B         C        D         E
           A  1388        8         2        1         0
           B     1      934         8        3         2
           C     1        5       841       15         0
           D     1        2         2      783         3
           E     4        0         2        2       896
Overall Statistics
                     Accuracy:  0.9874
                       95% CI:  (0.9838, 0.9903)
          No Information Rate:  0.2845
          P-Value [Acc > NIR]:  < 2.2e-16
                        Kappa:  0.984
     Mcnemar's Test P-Value:  0.005715
Statistics by Class:
                           Class: A   Class: B   Class: C   Class: D   Class: E
             Sensitivity   0.9950     0.9842     0.9836     0.9739     0.9945
             Specificity   0.9969     0.9965     0.9948     0.9980     0.9980
          Pos Pred Value   0.9921     0.9852     0.9756     0.9899     0.9912
          Neg Pred Value   0.9980     0.9962     0.9965     0.9949     0.9987
              Prevalence   0.2845     0.1935     0.1743     0.1639     0.1837
          Detection Rate   0.2830     0.1905     0.1715     0.1597     0.1827
    Detection Prevalence   0.2853     0.1933     0.1758     0.1613     0.1843
       Balanced Accuracy   0.9959     0.9903     0.9892     0.9860     0.9962
```

Frame 1: Prediction quality on evaluation dataset.

The prediction for testing dataset are shown in the following script, see Frame 2.

```
> predict(modelFit,testPC)
 [1] B A A A A B A B A A B A B A B A A B B B
Levels: A B C D E
```

Frame 2: Predicted results from testing dataset.

Discussion and conclusions

Frame 1 shows that the developed classification model delivers accurate results, even better results than in the original research [3], see confusion matrix and other parameters. Assuming that training, evaluation, and testing datasets share the same bias, it is possible to conclude that predictions (see Frame 2) are accurate as well.

References

1 https://www.coursera.org/course/predmachlearn, accessed 2014-06-22.

2 http://groupware.les.inf.puc-rio.br/har, accessed 2014-06-22.

3 Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th Augmented Human (AH) International Conference in cooperation with ACM SIGCHI (Augmented Human'13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Appendix

```
#load data
library('caret', 'kernlab', 'RANN')
# set working directory
setwd('C:/Users/user/gitrep/pml_qsm/')
df.training <- read.csv('./Data/pml-training.csv')
df.testing <- read.csv('./Data/pml-testing.csv')
set.seed(579245)
# divide df.training dataset to a training and evaluation datasets
trainIn <- createDataPartition(df.training$classe, list=FALSE, p=0.75)
training = df.training[trainIn,]
evaluation = df.training[-trainIn,]
# remove nearZeroVar
nzvIn <- nearZeroVar(training)
training <- training[-nzvIn]
evaluation <- evaluation[-nzvIn]
df.testing <- df.testing[-nzvIn]
training[is.na(training)] <- 0
evaluation[is.na(evaluation)] <- 0
df.testing[is.na(df.testing)] <- 0
# remove remaining factor variables, these changes should not affect the analysis,
# names are tranformed to numeric IDs, and these dates are irrelevant to the analysis
training[,2] <- as.numeric(training[,2])
training$cvtd_timestamp <- NULL
evaluation[,2] <- as.numeric(evaluation[,2])
evaluation$cvtd_timestamp <- NULL
df.testing[,2] <- as.numeric(df.testing[,2])
df.testing$cvtd_timestamp <- NULL
# run PCA and prepare data
preProc <- preProcess(training[,-104], method='pca')
trainPC <- predict(preProc,training[,-104])
evalPC <- predict(preProc,evaluation[,-104])
testPC <- predict(preProc,df.testing[,-104])
# create random forest model
# train random forest is a heavy operation, so use the model from file
if (file.exists('./RCode/randomForest_PCA.RDS')) {
  modelFit <- readRDS('./RCode/randomForest_PCA.RDS')
} else {
  modelFit <- train(training$classe ~ ., method='rf',data=trainPC)
  saveRDS(modelFit, './RCode/randomForest_PCA.RDS')
}
# predict for evaluation dataset and testing (submission) dataset
evalPredict <- predict(modelFit,evalPC)
print(confusionMatrix(evalPredict, evaluation$classe))
testPredict <- predict(modelFit,testPC)
# prepare to submit ansers
# uncomment the following lines and run separately
# pml_write_files = function(x){
#   n = length(x)
#   for(i in 1:n){
#     filename = paste0("./Data/","problem_id_",i,".txt")
#     write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
#   }
# }
#
# pml_write_files(as.character(testPredict))
```