Coursera - Practical Machine Learning - Course Assignment

Iñigo Fernández del Amo

1/2/2021

Executive Summary

This report explains a procedure to predict the *classe* of 20 different test cases belonging to the Weight Lifting Exercise data set. Among various ML algorithms trained using k-fold cross-validation ($k = 10$), Random Forests obtained the lowest out-of-sample error (0.7). Thus, making it the most accurate algorithm to predict the abovementioned outcomes.

1. Introduction

The Weight Lifting Exercise data set comprises information collected from various sensors (e.g., accelerometers) in wearable devices (e.g., Jawbone Up, Fitbit, etc.) regarding the movements of people when performing Weight Lifting exercises. This data set, and the different variables within it, can be utilized to predict how well wearable devices' users perform certain Weight Lifting exercises.

The Weight Lifting Exercise data set has been presented in **"Qualitative Activity Recognition of Weight Lifting Exercises"** by *Velloso, Bulling, Gellersen, Ugulino and Fuks (2013)*. For more information on it, please visit **http://groupware.les.inf.puc-rio.br/har#weight_lifting_exercises**.

2. Data preparation

The following are the R libraries utilized for this assignment.

```r
library(caret) # For training models and predicting outcomes
library(randomForest) # For applying random forests' algorithms
set.seed(234332) # For reproducibility purposes
```

The training and test data sets can be downloaded directly from the Internet.

```r
trainURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
training <- read.csv(url(trainURL), na.strings = c("NA", "", "#DIV/0!"))
training$classe <- as.factor(training$classe) # For classe to be treated as factor
dim(training)
```

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

```r
testURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
testing <- read.csv(url(testURL), na.strings = c("NA", "", "#DIV/0!"))
dim(testing)
```

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

For each data set a total of 160 variables are loaded, with 19622 observations for the training set and 20 observations for the test set. Due to the large number of observations in the training data set, this can be further split for cross-validation purposes (using a 70/30 ratio).

```r
inTrain <- createDataPartition(training$classe, p = 0.7, list = FALSE)
trainSet <- training[inTrain, ]
valSet <- training[-inTrain, ]
```

A brief exploration to the training data set shows that many of those variables have a large amount of NA values, while some others have very small variances. Besides, the first seven columns are identifiers with no relevance as predictors. All the variables can therefore be removed.

```r
allNA <- sapply(trainSet, function(x) mean(is.na(x))) > 0.95 # remove vars with NAs > 95%
trainSet <- trainSet[, allNA == FALSE]
valSet <- valSet[, allNA == FALSE]
nzv <- nearZeroVar(trainSet) # remove variables with near-zero variance
trainSet <- trainSet[, -nzv]
valSet <- valSet[, -nzv]
idVARS <- c(1:7)
trainSet <- trainSet[, -idVARS]
valSet <- valSet[, -idVARS]
dim(trainSet)
```

```
## [1] 13737    52
```

```r
dim(valSet)
```

```
## [1] 5885   52
```

These data tidying operations leave a total of 52 variables for *classe* prediction.

3. Prediction modelling with ML algorithms

Due to the number of variables included in the study, there are several methods that can be applied to obtain predictions on the outcome *classe*: (a) Naive Bayes, (b) Linear discriminant analysis, (c) Decision trees, (d) Random forests and (e) Stochastic Gradient Boosting (generalized boosted models).

The results of training these algorithms with the *trainSet* and evaluating them with the *valSet* are shown below. The coding employed to run those is show in **Appendix A**. The results of models' training and evaluation are fully displayed in **Appendix B** and **Appendix C**, respectively. It is worthy to note that k-fold cross-validation ($k = 10$) was employed to train these models.

```r
# List ML algorithms to be applied, train them with trainSet and evaluate their accuracies with valSet
#mlMethods <- c("nb", "lda", "rpart", "gbm", "rf")
#mlModels <- lapply(mlMethods, mlTraining)
#mlResults <- lapply(mlModels, mlPredicting)
# Visualize confusion matrices and accuracies
#mlPlots <- lapply(mlResults, mlPlotting)
#layout(matrix(c(1,3,5,2,4,5), nrow = 3, ncol = 2))
#mlPlots
# Visualize in-sample and out-of-sample errors
#mlErrors <- data.frame(do.call(rbind,(lapply(mlResults, mlEvaluating))))
#mlErrors
```

These results show the in-sample and out-of-sample errors obtained by the algorithms for both, the training set (*trainSet*) and the cross-validation set (*valSet*). Based on them, it is possible to say that Random Forests is the algorithm obtaining the lowest out-of-sample error (~0.7%). This is closely followed by Stochastic Gradient Boosting (~4%), while Naive Bayes (~26%), Linear Discriminant Analysis (~32%) and Decision Trees - CART (~52%) obtain less accurate results. Therefore, it seems reasonable to use Random Forests to predict the *classe* outcome on the *testing* data set.

4. Data prediction

Based on previous results, the Random Forests' model is utilized to predict the 20-quiz results (*testing* data set).

```
#predict(mlModels[[5]], newdata = testing)
```

5. Conclusions

This report shows the application of different ML algorithms to predict the *classe* outcome on the Weight Lifting Exercise data set. Due to the large number of NA values in some variables, and the low variances in others, only 52 variables out of 160 were used to predict the outcome. Besides, different ML algorithms (Naive Bayes, LDA, CART, GBM and RF) were trained (using 10-fold cross-validation) and evaluated with cross-validation data sets (*valSet*) to compare them according to their out-of-sample errors ($1 - Accuracy$). Random Forests (RF) was the algorithm with the lowest out-of-sample error (0.7) and so, it was utilized to predict the results on the *testing* data set.

Appendices

Appendix A

Functions to train ML algorithms and evaluate and plot confusion matrices

```
# Functions to train, evaluate and plot each ML algorithm sequentially
# Training control parameters
control <- trainControl(    preProcOptions = list(thresh = 0.8),
                      allowParallel=T,
                      savePredictions=T,
                      method = "cv",
                      number = 10)
# Training function for trainSet
mlTraining <- function (mlMethod) {
  modelFit <- train(classe ~ ., data = trainSet,
                    method = mlMethod, trControl = control)
  return(modelFit)
}
# Confusion matrix for valSet
mlPredicting <- function(modelFit) {
  modelPrediction <- predict(modelFit, newdata = valSet)
  modelCM <- confusionMatrix(modelPrediction, valSet$classe)
  modelCM$methodName <- modelFit$modelInfo$label
  modelCM$methodAccuracy <- max(modelFit$results$Accuracy)
  return(modelCM)
}
# Plot on confusion matrix results
mlPlotting <- function(modelResult) {
  mlPlot <- plot(modelResult$table, col = modelResult$byClass,
                 main = paste(modelResult$methodName, " - Accuracy = ",
                              round(modelResult$overall["Accuracy"], 4)))
  return(mlPlot)
}
# Calculate errors
mlEvaluating <- function(modelResult) {
  return(data.frame(Algorithm = modelResult$methodName,
                    In_Sample_Error = round(1 - modelResult$methodAccuracy, 4),
                    Out_Of_Sample_Error = round(1 - modelResult$overall["Accuracy"], 4)))
}
```

Appendix B

Results of ML algorithms training

```
#mlModels
```

Appendix C

Results of ML algorithms evaluation

```
#mlResults
```