Practical Machine Learning - Course Project Report

Yuan Liao

10/10/2020

1 Introduction

This is the report of modelling and predicting how well a group of people did barbell lifting. For more details on the data used, see HAR Dataset http://groupware.les.inf.puc-rio.br/har.

The prediction target is the variable "classe", corresponding to the 5 fashions of the Unilateral Dumbbell Biceps Curl: A, B, C, D, and E where A is the correct one according to the specification while the rest are the ones with different mistakes.

The predictors are potentially computed from the rest of the dataset that contain 159 variables, some of which are from the sensors mounted at arm, belt, forearm, and dumbbell.

```
library(dplyr)
library(Hmisc)
library(caret)
```

2 Descriptive analysis

Load training and testing data and take a look at the variables.

```
training <- read.csv('data/pml-training.csv')
testing <- read.csv('data/pml-testing.csv')

# Remove those columns that have ALL NAs
testing <- testing[,colSums(is.na(testing))<nrow(testing)]
colnames(testing)
```

```
##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
##  [7] "num_window"           "roll_belt"            "pitch_belt"
## [10] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
## [13] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [16] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [19] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [22] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [25] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [28] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [31] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [34] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [37] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [40] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
```

```
## [43] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [46] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [49] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [52] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [55] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [58] "magnet_forearm_y"     "magnet_forearm_z"     "problem_id"
```

To predict the testing set, its non-NA columns are selected except for the irrelevant ones of predicting classe: X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window, and num_window.

Correspondingly, the training set is also preprocessed to be in line with the testing set.

```
testing <- testing %>%
  select(!c(X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window, num_w


training <- training %>%
  select(!c(X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window, num_w
  select(c(colnames(select(testing, !problem_id)), 'classe'))
```

3 Modelling

The selected algorithm is Random forests.

After the screening of predictors in Section 2, the number of remaining predictors is still large. To reduce the dimension of feature set, PCA analysis is implemented in this section. And the 5-fold cross-validation is implemented to the training process to avoid overfitting.

```
set.seed(666)

lb <- createDataPartition(training$classe, p=0.7, list=FALSE)
training_train <- training[lb, ]
training_test <- training[-lb, ]

train_control <- trainControl(method = "cv", number = 5)
model_rf <- train(as.factor(classe) ~ .,
                  trControl = train_control,
                  preProcess = "pca",
                  method = "rf",
                  data = training_train,
                  prx=TRUE)
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
```

```
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range
```

```
print(model_rf)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: principal component signal extraction (52), centered
##  (52), scaled (52)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10991, 10989, 10988, 10990, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9696429  0.9615913
##   27    0.9542836  0.9421593
##   52    0.9539197  0.9416993
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Check the performance on the training set.

```
confusionMatrix(predict(model_rf, training_test[, -53]), as.factor(training_test$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1660   20    1    0    0
##          B    3 1089   21    0    3
```

```
##          C    7   23  991   52   10
##          D    3    5   11  909   11
##          E    1    2    2    3 1058
##
## Overall Statistics
##
##                Accuracy : 0.9698
##                  95% CI : (0.9651, 0.974)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9617
##
##  Mcnemar's Test P-Value : 1.001e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9916   0.9561   0.9659   0.9429   0.9778
## Specificity            0.9950   0.9943   0.9811   0.9939   0.9983
## Pos Pred Value         0.9875   0.9758   0.9151   0.9681   0.9925
## Neg Pred Value         0.9967   0.9895   0.9927   0.9889   0.9950
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2821   0.1850   0.1684   0.1545   0.1798
## Detection Prevalence   0.2856   0.1896   0.1840   0.1596   0.1811
## Balanced Accuracy      0.9933   0.9752   0.9735   0.9684   0.9881
```

4 Predicting

This section demonstrates the application of the trained model on the testing set.

```
predictions_rf <- predict(model_rf, testing[, -53])
names(predictions_rf) <- testing$problem_id
predictions_rf
```

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