Project for machine learning with the Weight Lifting Exercise Dataset

Wei Li

6/25/2017

Summary

The aim of this project is to train a machine learning model with the training set in order to predict which category the execution quality of each of 20 participant in the test data belongs to. The training and the test data come from the source http://groupware.les.inf.puc-rio.br/har. The project first cleaned and selected the variables, and then trained models with cross validation strategy and examined the out-of-sample error rate. Finally Chose the model with a better accuracy to predict 20 different test cases.

Cleaned and selected variables for the training process

The original training and the testing data sets from the source have 19622 and 20 observations respectively for 160 variables.

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

```
## 'data.frame':    20 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 6 5 5 1 4 5 5 5 2 3 ...
##   [list output truncated]
```

As there was no observation for many variables, removing them would simplify the training procedure. Also the names of participant, time and window related variables had been removed too. The total final prediction variable number was 53 with 19622 and 20 observations for training and validation sets, as listed as bellow:

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##   [list output truncated]
```

```
## 'data.frame':    20 obs. of  53 variables:
##  $ roll_belt           : num  123 1.02 0.87 125 1.35 -5.92 1.2 0.43 0.93 114 ...
##  $ pitch_belt          : num  27 4.87 1.82 -41.6 3.33 1.59 4.44 4.15 6.72 22.4 ...
```

```
##   [list output truncated]
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

Cross validation strategy

The goal to set up a model from the training dataset with machine learning here is to predict on the new data where the outcome is unknown, so the models should not over-fit the training data while generalizing well at the same time. The strategy for dealing with this issue in machine learning is cross validation. Basically there are three common ways to do cross validation: (a) simply split the data into the training and validation set; (b) k-fold cross validation; (c) leave-p-out or leave-one-out cross validation. The third way is usually called exhaustive cross-validation, so it wouldn't be addressed in this project.

Divide the dataset into training and validation sets

The cleaned training data was divided into the final training and validation data sets with 3/4 and 1/4 fractions respectively, as listed as below:

```
inTrain = createDataPartition(data_pml_train$classe, p = 3/4)[[1]]
training = data_pml_train[ inTrain,]
validation = data_pml_train[-inTrain,]

str(training, list.len = 2)
```

```
## 'data.frame':    14718 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.45 1.43 1.42 ...
##  $ pitch_belt          : num  8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.18 8.18 8.2 ...
##   [list output truncated]
```

```
str(validation, list.len = 2)
```

```
## 'data.frame':    4904 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.43 1.45 1.45 1.57 1.6 1.53 1.41 1.41 1.39 ...
##  $ pitch_belt          : num  8.07 8.16 8.17 8.2 8.06 8.1 8.14 8.18 8.11 8.05 ...
##   [list output truncated]
```

Train models

1: The "train" function in "caret" can do cross-validation. I tried folder number be 10 with trControl argument and "rf" (random forest model).

As the training time with k-fold with "train" function was very time consuming and needed hours, which is not convenient with Knit, so I just listed them as bellows:

```
# set.seed(12345)
# modelfit_rf10k= train(classe ~. , training, method = "rf",prox = TRUE,
#                       trControl = trainControl(method = "cv", number =10, verboseIter = TRUE))

# > modelfit_rf10k
# Random Forest
#
# 14718 samples
# 52 predictor
# 5 classes: 'A', 'B', 'C', 'D', 'E'
#
# No pre-processing
# Resampling: Cross-Validated (10 fold)
# Summary of sample sizes: 13247, 13246, 13245, 13247, 13246, 13248, ...
# Resampling results across tuning parameters:
#
# mtry  Accuracy   Kappa
# 2     0.9929344  0.9910613
# 27    0.9933432  0.9915794
# 52    0.9912366  0.9889140
#
# Accuracy was used to select the optimal model using  the largest value.
# The final value used for the model was mtry = 27.
```

2: Use randomForest function with randomForest package

Here the model was trained on the split training set without do further k-fold cross-validation.

```
set.seed(12345)
modelfit_rf <- randomForest::randomForest(classe ~ ., data = training)
```

In-sample error and out-of-sample error

1: Let's first take a look at in-sample error for the models, that is, the error with the training set itself.

This in-sample error is from modelfit_rf10k:

```
# pred1Train10k <- predict(modelfit_rf10k, newdata = training)
# confusionMatrix(table(pred1Train10k, training$classe))

# Confusion Matrix and Statistics
#
# pred1Train10k    A    B    C    D    E
# A 4185    0    0    0    0
# B    0 2848    0    0    0
# C    0    0 2567    0    0
# D    0    0    0 2412    0
# E    0    0    0    0 2706
#
# Overall Statistics
```

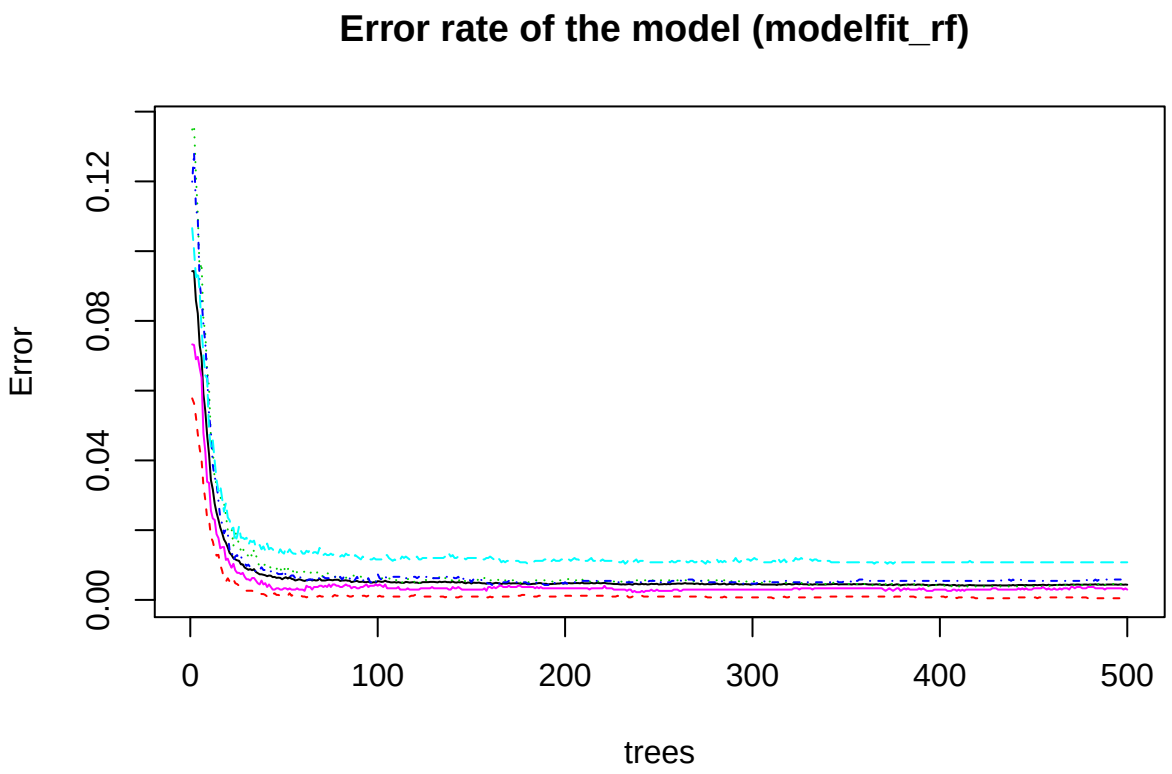
```
#
# Accuracy : 1
# 95% CI : (0.9997, 1)
# No Information Rate : 0.2843
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 1
# Mcnemar's Test P-Value : NA
```

The following in-sample error comes from modelfit_rf

```
pred_train_rf <- predict(modelfit_rf, newdata = training)
confusionMatrix(table(pred_train_rf, training$classe))
```

```
## Confusion Matrix and Statistics
##
##
## pred_train_rf    A    B    C    D    E
##             A 4185    0    0    0    0
##             B    0 2848    0    0    0
##             C    0    0 2567    0    0
##             D    0    0    0 2412    0
##             E    0    0    0    0 2706
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
plot(modelfit_rf, main = "Error rate of the model (modelfit_rf) ")
```



The above figure shows us the error rate of the model with function randomForest.

2: Examine the out-of-sample error with the training models

The out-of-sample error from modelfit_rf10k is:

```
# pred1Test10k <- predict(modelfit_rf10k, newdata = validation)
# confusionMatrix(table(pred1Test10k, validation$classe))
#
# Confusion Matrix and Statistics
#
#
# pred1Test10k    A    B    C    D    E
# A 1394    6    0    0    0
# B    1  938    3    0    0
# C    0    5  848    8    2
# D    0    0    4  796    5
# E    0    0    0    0  894
#
# Overall Statistics
#
# Accuracy : 0.9931
# 95% CI : (0.9903, 0.9952)
# No Information Rate : 0.2845
# P-Value [Acc > NIR] : < 2.2e-16
#
# Kappa : 0.9912
```

Here is the out_of_sample error from modelfit_rf

```
pred_valid_rf <- predict(modelfit_rf, newdata = validation)
confusionMatrix(table(pred_valid_rf, validation$classe))
```

```
## Confusion Matrix and Statistics
##
##
## pred_valid_rf    A    B    C    D    E
##             A 1395    6    0    0    0
##             B    0  940    3    0    0
##             C    0    3  850    6    1
##             D    0    0    2  798    4
##             E    0    0    0    0  896
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9925, 0.9967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9905   0.9942   0.9925   0.9945
## Specificity            0.9983   0.9992   0.9975   0.9985   1.0000
## Pos Pred Value         0.9957   0.9968   0.9884   0.9925   1.0000
## Neg Pred Value         1.0000   0.9977   0.9988   0.9985   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1917   0.1733   0.1627   0.1827
## Detection Prevalence   0.2857   0.1923   0.1754   0.1639   0.1827
## Balanced Accuracy      0.9991   0.9949   0.9958   0.9955   0.9972
```

From the above two models, we can see their performances are very similar, the out-of-sample errors are 0.69%=1-0.9931 and 0.51%=1-0.9949 for modelfit_rf10k and modelfit_rf respectively. So both models fitted the training data and generalized well. Because the model (modelfit_rf10k) trained with "train" function using trControl argument and "rf" algorithm was too time consuming for the training process, so the model (modelfit_rf) with randomForest function was selected for the prediction.

Predict the outcome for the original test dataset

```
pred_test_rf <- predict(modelfit_rf, newdata = data_pml_test)
pred_test_rf
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

Conclussion

The outcome of the test data was predicted using modelfit_rf, the error rate for this prediction will be expected around 0.51% based on the out-of-sample error obtained from the validation dataset.