Prediction Assignment Writeup

Sanyam Sharma

28 November 2020

Synopsis

One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. Our goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and to predict the manner in which they did the exercise. This is the "classe" variable in the training set. This report describes how data was cleaned, how I split "pml-training.csv" into train set and test set, and some of models are investigated.

Basic Setting

```
knitr::opts_chunk$set(echo = TRUE, warning = TRUE)
```

Loading Package and Set Seed

```
set.seed(12345)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.3
```

```
## Warning: package 'ggplot2' was built under R version 4.0.3
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.0.3
```

Download Training Data and Testing Data

```
#training
url_train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(url_train, destfile = "./pml-training.csv")
#testing
url_submit <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(url_submit, destfile = "./pml-testing.csv")
```

```
rawdata <- read.csv("./pml-training.csv", na.strings = c("", "NA"))
submit_data <- read.csv("./pml-testing.csv", na.strings = c("", "NA"))
```

Cleaning Data

We should delete the column that contains NA to avoid the error. In addition, in order to make accurate predictions, columns that is not related exercise must also be deleted. In particular "X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp", "new_window", "num_window" are deleted.

```
colname <- colnames(rawdata)[!colSums(is.na(rawdata)) > 0]
colname
```

```
##  [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"     "classe"
```

```
#Slice data related with exercise
colname <- colname[8: length(colname)]
df_wo_NA <- rawdata[colname]
```

```
#Check the colnames of df_wo_NA is in submit_data.
#The last colname is "classe"
is.element(colname, colnames(submit_data))
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [49]  TRUE  TRUE  TRUE  TRUE FALSE
```

Split Data

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

Random Forest

It takes a very long time for training, but it has a high accuracy.

```
model_rf <- train(classe ~ ., data = training, method = "rf")
pred_rf <- predict(model_rf, testing)
confusionMatrix(testing$classe, pred_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    9  934    6    0    0
##          C    0    3  848    4    0
##          D    0    0    3  801    0
##          E    0    0    1    1  899
##
## Overall Statistics
##
##                Accuracy : 0.9945
##                  95% CI : (0.992, 0.9964)
##     No Information Rate : 0.2863
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.993
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9936   0.9968   0.9883   0.9938   1.0000
## Specificity            1.0000   0.9962   0.9983   0.9993   0.9995
## Pos Pred Value         1.0000   0.9842   0.9918   0.9963   0.9978
## Neg Pred Value         0.9974   0.9992   0.9975   0.9988   1.0000
## Prevalence             0.2863   0.1911   0.1750   0.1644   0.1833
## Detection Rate         0.2845   0.1905   0.1729   0.1633   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9968   0.9965   0.9933   0.9965   0.9998
```

Liner Discriminant Analysis

It takes a short time but poor accuracy.

```
model_lda <- train(classe ~ ., data = training, method = "lda")
pred_lda <- predict(model_lda, testing)
confusionMatrix(testing$classe, pred_lda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1150   29  103  103   10
##          B  159  589  125   34   42
##          C  100   71  558  101   25
##          D   34   40   89  609   32
##          E   31  151   84   83  552
##
## Overall Statistics
##
##                Accuracy : 0.7051
##                  95% CI : (0.6922, 0.7179)
##     No Information Rate : 0.3006
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6267
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7802   0.6693   0.5819   0.6548   0.8351
## Specificity            0.9286   0.9105   0.9247   0.9509   0.9177
## Pos Pred Value         0.8244   0.6207   0.6526   0.7575   0.6127
## Neg Pred Value         0.9077   0.9264   0.9010   0.9217   0.9728
## Prevalence             0.3006   0.1794   0.1956   0.1896   0.1348
## Detection Rate         0.2345   0.1201   0.1138   0.1242   0.1126
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.8544   0.7899   0.7533   0.8029   0.8764
```

Recursive Partitioning and Regression Trees

The results can be confirmed visually, but poor accuracy.

```
model_rpart <- train(classe ~ ., data = training, method = "rpart")
pred_rpart<- predict(model_rpart, testing)
confusionMatrix(testing$classe, pred_rpart)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 859 215 203 109   9
##          B 151 434 166 197   1
##          C  27 117 442 269   0
##          D  43  65 131 492  73
##          E  14 143 111 133 500
##
## Overall Statistics
##
##                Accuracy : 0.5561
```

```
##                  95% CI : (0.542, 0.57)
##     No Information Rate : 0.2447
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4442
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7852   0.4456  0.41975   0.4100   0.8576
## Specificity            0.8593   0.8690  0.89276   0.9158   0.9072
## Pos Pred Value         0.6158   0.4573  0.51696   0.6119   0.5549
## Neg Pred Value         0.9330   0.8635  0.84910   0.8273   0.9793
## Prevalence             0.2231   0.1986  0.21472   0.2447   0.1189
## Detection Rate         0.1752   0.0885  0.09013   0.1003   0.1020
## Detection Prevalence   0.2845   0.1935  0.17435   0.1639   0.1837
## Balanced Accuracy      0.8223   0.6573  0.65625   0.6629   0.8824
```

Submit data with Random Forest

We can use the high accuracy model to submit data. In this report the Random Forest accuracy has the highest value 99.45. We can show the prediction.

```
submit_rf <- predict(model_rf, submit_data)
submit_rf
```

```
##  [1] 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
```