Prediction of Human Activity by Applying Machine Learning Algorithms

Jiachang (Ernest) Xu

6/23/2017

Sectiong 1: Synopsis

The objective of this project is to predict human activity by applying **machine learning** algorithms.

Section 2: Data Loading

First of all, before we do anything, we shall set the seed to 1024 for the purpose of reproducibility. Then, we shall download the training and testing datasets to the **./data** folder.

```
## set the seed for reproducibility
set.seed(22)
## download training data
if (!file.exists("./data/training.csv")) {
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
                  destfile = "./data/training.csv")
}

## download testing data
if (!file.exists("./data/testing.csv")) {
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
                  destfile = "./data/testing.csv")
}
```

After the training and testing datasets are downloaded, we shall read the datasets into R ready for data cleaning.

```
## load the full training data reday for data slicing
data <- read.csv(file = "./data/training.csv")
## loading 20 cases of testing data for validation
validation <- read.csv(file = "./data/testing.csv")
```

Section 3: Data Cleaning

Before actually starting the process of data cleaning, let's take a close look at **data** first.

```
class(data$classe)
```

```
## [1] "factor"
```

```
levels(data$classe)
```

```
## [1] "A" "B" "C" "D" "E"
```

```
dim(data)
```

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

We can see from the output above that there exist a lot of empty spaces and NA values. Let's identify the level of NA value in **data**.

```
## identify NA level
NA.levels <- unique(apply(data, 2, function(x) {sum(is.na(x))} ))
NA.number <- dim(data)[1]-NA.levels[2]
NA.non <- NA.number/dim(data)[1]
sprintf("%1.2f%%", 100*NA.non)
```

```
## [1] "2.07%"
```

Then, we can replace empty spaces and div0 to NA

```
data[data == ""] <- NA
data[data == "#DIV/0!"] <- NA
data[data == "<NA>"] <- NA
```

Now, there are no empty spaces or irregular values in **data**, we shall spitt **data** to **train.data** and **test.data**

```
set.seed(22)
traindex <- createDataPartition(data$classe,p = 0.8,list = FALSE)
train <- data[traindex,]
test <- data[-traindex,]
```

We split **train.data** to old window rows (non-aggregated).

```
## select non-aggregated sensor data
train_raw <- train[which(train$new_window == "no"),]
## sensor data without NA columns (summary data)
train_raw <- train[!colSums(is.na(train)) > 0]
## test NA purity
sum(is.na(train_raw))
```

```
## [1] 0
```

We split **train.data** and **test.data** to new window rows (aggregated), and remove NA columns and rows from the new training and testing data frames.

```
#Splitting data to new window rows (aggregated data)
train_sum <- train[which(train$new_window == "yes"),]
test_sum <- test[which(test$new_window == "yes"),]

#Removing full NA columns
train_sum_clean <- subset(train_sum, select=-c(kurtosis_picth_belt,kurtosis_yaw_belt,kurtosis_picth_arm

test_sum_clean <- subset(test_sum, select=-c(kurtosis_picth_belt,kurtosis_yaw_belt,kurtosis_picth_arm,ku

#Removing NA rows
train_done <- train_sum_clean[complete.cases(train_sum_clean),]
sum(is.na(train_done))
```

```
## [1] 0
```

```
test_done <- test_sum_clean[complete.cases(test_sum_clean),]
sum(is.na(test_done))
```

```
## [1] 0
```

Section 4: Machine Learning

```r
model1 <- randomForest(classe ~. , data=train_raw[,-c(1:7)], method="class")
model1
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = train_raw[, -c(1:7)],      method = "class")
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.43%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4461    1    0    1    1 0.000672043
## B   13 3019    6    0    0 0.006254115
## C    0   11 2725    2    0 0.004747991
## D    0    0   27 2544    2 0.011270890
## E    0    0    1    3 2882 0.001386001
```

```r
pred_test1 <- predict(model1, test)
pred_train1 <- predict(model1, train)
confusionMatrix(pred_test1, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1115    4    0    0    0
##          B    1  755    3    0    0
##          C    0    0  681    2    0
##          D    0    0    0  640    3
##          E    0    0    0    1  718
##
## Overall Statistics
##
##                Accuracy : 0.9964
##                  95% CI : (0.994, 0.998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9955
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9947   0.9956   0.9953   0.9958
## Specificity            0.9986   0.9987   0.9994   0.9991   0.9997
## Pos Pred Value         0.9964   0.9947   0.9971   0.9953   0.9986
## Neg Pred Value         0.9996   0.9987   0.9991   0.9991   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1925   0.1736   0.1631   0.1830
## Detection Prevalence   0.2852   0.1935   0.1741   0.1639   0.1833
## Balanced Accuracy      0.9988   0.9967   0.9975   0.9972   0.9978
```

```
confusionMatrix(pred_train1, train$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4464    0    0    0    0
##          B    0 3038    0    0    0
##          C    0    0 2738    0    0
##          D    0    0    0 2573    0
##          E    0    0    0    0 2886
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 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   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Section 5: Conclusion

```
predict(model1,validation)
```

```
##  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
```