Practical Machine Learning Course Project

INTRODUCTION

The goal of this project is to predict the manner in which people did the exercise. This is the “classe” variable in the training set.

EXPLORE AND PREPROCESS THE TRAINING DATASET

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

After download the data set to local dir, using read.csv() to load data.

```
setwd("D:\\code\\predmachlearn")
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.0.3
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 3.0.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.0.3
```

```
set.seed(107)
training <- read.csv("data\\pml-training.csv", na.strings = c("NA",
""))
testing <- read.csv("data\\pml-testing.csv", na.strings = c("NA",
""))
ncol(training)
```

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

```
nrow(training)
```

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

The training data set is quite large. It contains 19622 samples with 160 features. Exclude the “classe” variable, there are 159 variables would be used as predictor.

The “classe” is the outcome variable with five levels. There are sitting-down, standing-up, standing, walking, and sitting.

```
table(training$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

Then, remove some metadata to record, such as X (row number), user_name and cvtd_timestamp.

```
mindex <- grep("X|user_name|cvtd_timestamp", names(training))
training <- training[, -mindex]
testing <- testing[, -mindex]
```

Explore the data, it is easy to find that there are a lot of values NA or useless or empty. Remove these variables will increase training speed.

```
NASum <- apply(training, 2, function(x) {
    sum(is.na(x))
})
training <- training[, which(NASum == 0)]
testing <- testing[, which(NASum == 0)]
ncol(training)
```

```
## [1] 57
```

BUILD MODEL

Divide the data into 2 sets, one for training, the other for validate.

```
trainingIndex <- createDataPartition(training$classe, p = 0.5, list
= FALSE)
training.train <- training[trainingIndex, ]
training.test <- training[-trainingIndex, ]
rm(training)
```

There are five groups to identify. Tree-based methods and LDA are adopted to analysis this data set.

```
# CART
system.time(modRPart <- train(classe ~ ., data = training.train,
method = "rpart"))
```

```
## Loading required package: rpart
```

```
## Warning: package 'e1071' was built under R version 3.0.3
```

```
##    user  system elapsed
##   82.44    0.36   82.99
```

```
# Random Forest
trControl <- trainControl(method = "cv", number = 4)
system.time(modRF <- train(training.train$classe ~ ., method =
"rf", trControl = trControl,
    training.train))
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 3.0.3
```

```
## randomForest 4.6-7
## Type rfNews() to see new features/changes/bug fixes.
```

```
##    user  system elapsed
##  497.01    2.11  500.34
```

```
# Boosting tree system.time(modGBM <- train(classe ~ .,
data=training.train,
# method='gbm')) LDA
system.time(modLDA <- train(classe ~ ., data = training.train,
method = "lda"))
```

```
## Loading required package: MASS
```

```
## Warning: package 'MASS' was built under R version 3.0.3
```

```
##    user  system elapsed
##   37.50    0.62   38.22
```

According to system.time method, boosting tree took a long time to train the model (Removed when prepared this report).

```
predRPart <- predict(modRPart, training.test)
predRF <- predict(modRF, training.test)
# Removed when prepared this report predGBM <- predict(modGBM,
# training.test)
predLDA <- predict(modLDA, training.test)
confusionMatrix(predRPart, training.test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2533  750  796  697  269
##          B   48  655   55  288  249
##          C  201  493  860  623  479
##          D    0    0    0    0    0
##          E    8    0    0    0  806
##
## Overall Statistics
##
##                Accuracy : 0.495
##                  95% CI : (0.485, 0.505)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.34
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class:
E
## Sensitivity            0.908   0.3451   0.5026    0.000
0.4470
## Specificity            0.642   0.9191   0.7782    1.000
0.9990
## Pos Pred Value         0.502   0.5058   0.3238      NaN
0.9902
## Neg Pred Value         0.946   0.8540   0.8810    0.836
0.8892
## Prevalence             0.284   0.1935   0.1744    0.164
0.1838
## Detection Rate         0.258   0.0668   0.0877    0.000
0.0822
## Detection Prevalence   0.514   0.1320   0.2707    0.000
0.0830
## Balanced Accuracy      0.775   0.6321   0.6404    0.500
0.7230
```

```
confusionMatrix(predRF, training.test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2790    6    0    0    0
##          B    0 1891   10    0    0
##          C    0    1 1701    4    0
##          D    0    0    0 1603    0
##          E    0    0    0    1 1803
##
## Overall Statistics
##
##                Accuracy : 0.998
##                  95% CI : (0.997, 0.999)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.997
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class:
E
## Sensitivity            1.000    0.996    0.994    0.997
1.000
## Specificity            0.999    0.999    0.999    1.000
1.000
## Pos Pred Value         0.998    0.995    0.997    1.000
0.999
## Neg Pred Value         1.000    0.999    0.999    0.999
1.000
## Prevalence             0.284    0.193    0.174    0.164
0.184
## Detection Rate         0.284    0.193    0.173    0.163
0.184
## Detection Prevalence   0.285    0.194    0.174    0.163
0.184
## Balanced Accuracy      1.000    0.998    0.997    0.998
1.000
```

```
# Removed when prepared this report confusionMatrix(predGBM,
# training.test$classe)
confusionMatrix(predLDA, training.test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2383  272  172   94   76
##          B   77 1231  152   61  238
##          C  148  231 1141  197  139
##          D  177   72  190 1208  198
##          E    5   92   56   48 1152
##
## Overall Statistics
##
##                Accuracy : 0.725
##                  95% CI : (0.716, 0.734)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.652
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class:
E
## Sensitivity            0.854    0.649    0.667    0.751
0.639
## Specificity            0.913    0.933    0.912    0.922
0.975
## Pos Pred Value         0.795    0.700    0.615    0.655
0.851
## Neg Pred Value         0.940    0.917    0.928    0.950
0.923
## Prevalence             0.284    0.193    0.174    0.164
0.184
## Detection Rate         0.243    0.125    0.116    0.123
0.117
## Detection Prevalence   0.306    0.179    0.189    0.188
0.138
## Balanced Accuracy      0.883    0.791    0.789    0.837
0.807
```

Using training.test to test result, obviously, Boosting tree and random forest are the best, their accuracies are great than 99%.

CONCLUSION

Random Forest is a good method for this dataset for it's accuracy and training time cost.

20 TEST CASES

Now use the modRF to predict the second part of the assignment.

```
pml_write_files = function(x) {
    n = length(x)
    for (i in 1:n) {
        filename = paste0("problem_id_", i, ".txt")
        write.table(x[i], file = filename, quote = FALSE, row.names
= FALSE,
            col.names = FALSE)
    }
}
predTestRPart <- predict(modRF, testing)
pml_write_files(predTestRPart)
```