Practical Machine Learning Project

freestander

Sunday, September 27, 2015

Introduction

This project analyzes the personal activity data collected from body movement. The purpose is to quantify how well people perform on a particular activity by using a predictive model to classify correct and incorrect movements.

Data Sources

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

More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data Processing

First we download the data from sources and load it into R studio.

```r
# remove all the previous data in the environment
rm(list = ls(all = TRUE))

# set up the working directory
setwd("C://Users//Qi-Desktop//Documents//RProjects//Coursera//Practical_Machine_Learning")

# load the data set for training and testing
training <- read.csv(file="./Data/pml-training.csv", header=TRUE, as.is = TRUE, stringsAsFactors = FALSE
testing_ds <- read.csv(file="./Data/pml-testing.csv", header=TRUE, as.is = TRUE, stringsAsFactors = FALS

# convert the target variable into factor variable
training$classe <- as.factor(training$classe)
```

First we check the dimension of the training data set. The original training data contains 19622 records and 160 variables.

```r
dim(training)
```

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

Next we check the dimension of the testing data set. The testing data contains 20 records and 160 variables.

```r
dim(testing_ds)
```

```
## [1]  20 160
```

Next we do some data processing to prepare for the modeling dataset. First we load "caret" package and split the training data into training set (70%) and validation set (30%).

```
## Warning: package 'caret' was built under R version 3.2.2
```

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

Variable Selection

By inspecting the variables in the data sets, we can see a lot variables with NA values or near zero variance. Also some descriptive variables should not be included in the prediction model.

```
# function to check the number of missing values from a variable
na_size <- sapply(training_ds, function(x)
  {
  sum(is.na(x))
  })
# variables with more than 90% missing values
na_var <- names(na_size[na_size >= 0.9 * dim(training_ds)[1]])
# exclude variables with more than 90% missing values
training_ds <- training_ds[, !names(training_ds) %in% na_var]

# variables with descriptive features
descr_var <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp", "new
# exclude variables with descriptive features
training_ds <- training_ds[, !names(training_ds) %in% descr_var]

# variables near zero variance
no_variance_var <- nearZeroVar(training_ds)
# exclude variables with near zero variance
training_ds <- training_ds[, !names(training_ds) %in% no_variance_var]
```

We can check the dimension of the training set after the split and variable selection. The training set now only contains 13737 records and 53 variables.

```
dim(training_ds)
```

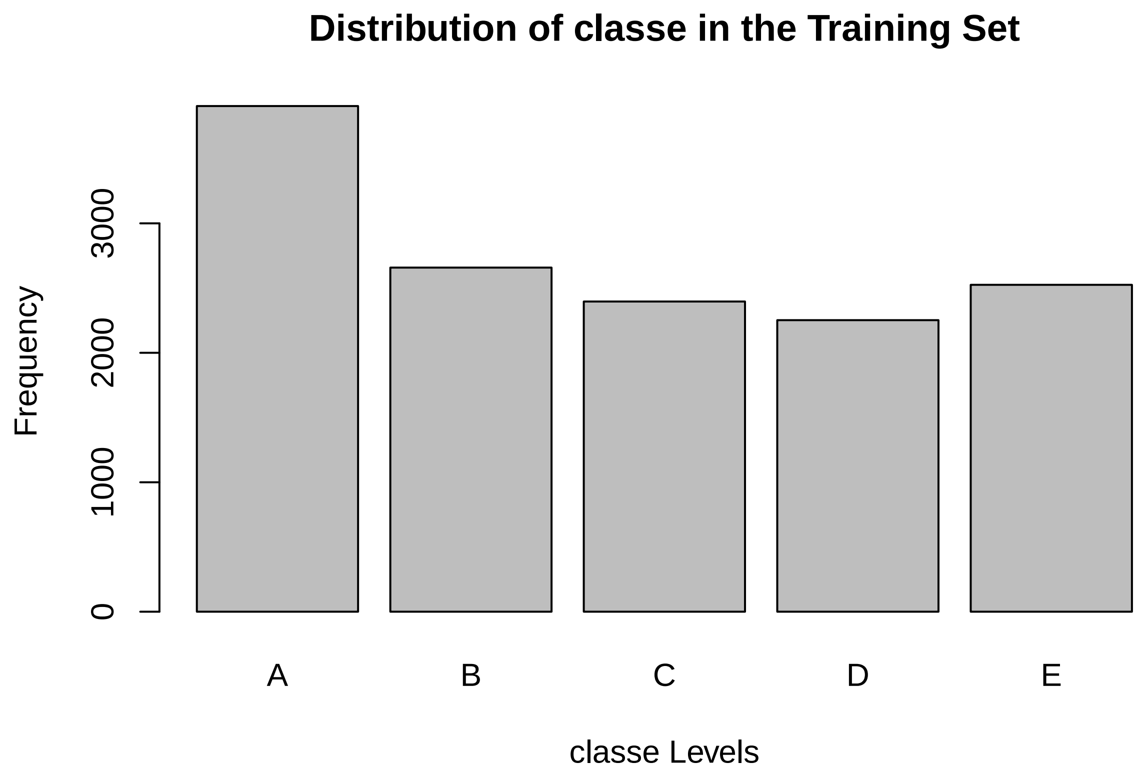
```
## [1] 13737    53
```

By visualizing the distribution of the target variable, we found no rare "classe" that needs special processing, e.g., stratified sampling or weighted regression.

```
# frequency table of the target variable
table(training_ds$classe)
```

```
##
##    A    B    C    D    E
## 3906 2658 2396 2252 2525
```

```
# frequency plot of the target variable
plot(training_ds$classe, main="Distribution of classe in the Training Set", xlab="classe Levels", ylab=
```



Predictive Model Training

We are going to train the predictive model using two popular algorithms “rpart” (decision trees) and “rf” (random forest).

First we train the model using “rpart” algorithm, and use confusion matrix to check prediction accurancy. As from the output below, the prediction accuracy is 74.03%.

```
set.seed(123)
library(rpart)
model_rpart <- rpart(classe ~ ., data=training_ds, method="class")
# use confusion matrix to check accuracy
rpart_pred_traiing_ds <- predict(model_rpart, training_ds, type="class")
confusionMatrix(training_ds$classe, rpart_pred_traiing_ds)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 3356   68   77  349   56
##          B  393 1570  269  258  168
##          C   97  151 1622  461   65
##          D  114  121  105 1766  146
##          E   50  145  126  349 1855
##
## Overall Statistics
##
##                Accuracy : 0.7403
##                  95% CI : (0.7328, 0.7476)
##     No Information Rate : 0.2919
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6718
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8369   0.7640   0.7376   0.5548   0.8100
## Specificity            0.9435   0.9069   0.9329   0.9540   0.9415
## Pos Pred Value         0.8592   0.5907   0.6770   0.7842   0.7347
## Neg Pred Value         0.9335   0.9562   0.9491   0.8766   0.9612
## Prevalence             0.2919   0.1496   0.1601   0.2317   0.1667
## Detection Rate         0.2443   0.1143   0.1181   0.1286   0.1350
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.8902   0.8354   0.8353   0.7544   0.8758
```

Next we training the model using “rf” algorithm, and use confusion matrix to check prediction accurancy. As from the output below, the prediction accuracy is 100%. Note than we only use 10 trees to avoid excessive computational time.

```r
set.seed(123)
# train the model using "rf" algorithm
# model_rf <- train(classe ~ ., method="rf", data=training_ds, verbose=F)
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```r
model_rf <- randomForest(classe ~ ., data=training_ds, importance=TRUE, ntrees=10)
# use confusion matrix to check accuracy
rf_pred_training_ds <- predict(model_rf, training_ds)
confusionMatrix(training_ds$classe, rf_pred_training_ds)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
```

```
##
## 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   0.1838
## 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
```

Model Selection

Prediction result from the training set is not enough to tell which model is better since the model may be overfitting. Next We check the prediction results of the two models using the validation set.

First we check prediction accurancy on the validation set using "rpart" algorithm. As from the output below, the prediction accuracy is 73.15%, smaller than 74.03% from the training set.

```
# use confusion matrix to check accuracy
library(rpart)
rpart_pred_validation_ds <- predict(model_rpart, validation_ds, type="class")
confusionMatrix(validation_ds$classe, rpart_pred_validation_ds)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1424   26   33  167   24
##          B  184  667  107  106   75
##          C   52   54  701  193   26
##          D   56   48   53  735   72
##          E   25   70   52  157  778
##
## Overall Statistics
##
##                Accuracy : 0.7315
##                  95% CI : (0.72, 0.7428)
##     No Information Rate : 0.2958
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.6606
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8179   0.7711   0.7410   0.5412   0.7979
## Specificity            0.9397   0.9060   0.9342   0.9494   0.9381
## Pos Pred Value         0.8507   0.5856   0.6832   0.7624   0.7190
## Neg Pred Value         0.9247   0.9583   0.9496   0.8734   0.9590
## Prevalence             0.2958   0.1470   0.1607   0.2308   0.1657
## Detection Rate         0.2420   0.1133   0.1191   0.1249   0.1322
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.8788   0.8385   0.8376   0.7453   0.8680
```

Next we check prediction accurancy on the validation set using "rf" algorithm. As from the output below, the prediction accuracy is 99.46%, smaller than 100% from the training set.

```
# use confusion matrix to check accuracy
library(randomForest)
rf_pred_validation_ds <- predict(model_rf, validation_ds)
confusionMatrix(validation_ds$classe, rf_pred_validation_ds)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    5 1134    0    0    0
##          C    0   10 1016    0    0
##          D    0    0   15  948    1
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9946
##                  95% CI : (0.9923, 0.9963)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                  Kappa : 0.9931
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9904   0.9855   1.0000   0.9991
## Specificity            0.9998   0.9989   0.9979   0.9968   1.0000
## Pos Pred Value         0.9994   0.9956   0.9903   0.9834   1.0000
## Neg Pred Value         0.9988   0.9977   0.9969   1.0000   0.9998
## Prevalence             0.2851   0.1946   0.1752   0.1611   0.1840
## Detection Rate         0.2843   0.1927   0.1726   0.1611   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9984   0.9947   0.9917   0.9984   0.9995
```

As a result, “rf” model has better prediction accuracy on the validation set, and we will use it to predict on the final testing set.

Test Set Prediction Result

Finally, we use “rf” model to predict on the final testing set.

```
##  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
```

We then save the output to text files according to instructions and post it to the submission page.