Project for Practical Machine Learning

12/2015

Background

Using devices such as Jawbone Up, Nike FuelBand and Fitbit, it is now possible to collect a large amount of data about personal activities. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or simply because they are tech geeks. It is common for people to quantify how much of a particular activity they do. However, it is not common for them to quantify how well they do it.

This project invovles predicting the activities that people do using data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants, who were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website: http://groupware.les.inf.puc-rio.br/har.

Loading the data

The data was downloaded from the course website. It consists of two files, training set, which is used to build the predicting model, and testing set, which is to be tested on.

The dimension of the training data which is used to build the model is 19622×160, which means that it has 160 variables and 19622 measurements.

```
# Load the library
library(caret)
```

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

```
# Set working directory
setwd("D:\\Online\\DataScienceSpecialization\\PracticalMachineLearning\\Project\\temp")
# Load the training set
data <- read.csv('pml-training.csv', na.strings = c("NA", "#DIV/0!", ""))
dim(data)
```

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

A quick inspection reveals that the data has a large portion of missing values (NA). Therefore, we need to preprocess and clean the data before doing the machine learning study.

```
# inspect data
names(data)
head(data)
summary(data)
```

Preprocessing the data

Data preprocessing in this project includes three steps - removing the non-meaningful variables in predicting, removing the variables with nearly zero variance, and removing the variables which have lots of missing values.

Removing the non-meaningful variables

There are 160 variables in total in the raw data. Some of them, such as "X", "user_name" and "raw_timestamp_part_1", do not make sense in building the machine learning model, and need to be removed.

```
# Removing non-meaningful variables
data <- data[, -(1 : 5)]
```

Removing the variables with nearly zero variance

The next step is to removing the variables with nearly zero variance. The number of the variables now reduces to 119.

```
# Removing variables with nearly zero variance
nzv <- nearZeroVar(data)
data <- data[, -nzv]
dim(data)
```

```
## [1] 19622   119
```

Removing the variables which have lots of missing values

Variables which have lots of missing values make it difficult to build the model and predict on new data. We need to remove them. The criteria I use is that a variable with more than 90% missing values should be removed. The number of the variables now reduces to 54.

```
# Removing variables which have lots of missing values
removeNA <- sapply(data, function(x) mean(is.na(x))) > 0.9
data <- data[, removeNA == F]
dim(data)
```

```
## [1] 19622    54
```

Building the model

We are now ready to build the model. First, I split the data into training and validation sets. Validation set is used to compute the out of sample error. 75% data goes to the training set, while the rest goes to validation set.

```
inTrain <- createDataPartition(y = data$classe, p = 0.75, list = F)
training <- data[inTrain, ]
validation <- data[-inTrain, ]
```

Two predicting methods are tried, decision trees and random forests. These methods are then compared by the predicting accuracies.

Predicting with trees

The first model I use it the decision trees model. I build the model, test the model on the validation set, and then calculate the confusion matrix.

```
# Build the decision tree model
model_tree <- train(classe ~ ., method = "rpart", data = training)
```

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

```
# Predict on the validation set
predict_tree <- predict(model_tree, newdata = validation)
# Compute the confusion matrix
confusionMatrix(validation$classe, predict_tree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1254   18  100    0   23
##          B  387  324  238    0    0
##          C  388   30  437    0    0
##          D  348  154  302    0    0
##          E  125  129  227    0  420
##
## Overall Statistics
##
##                Accuracy : 0.4965
##                  95% CI : (0.4824, 0.5106)
##     No Information Rate : 0.5102
##     P-Value [Acc > NIR] : 0.9731
##
##                   Kappa : 0.3428
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5012  0.49466  0.33512       NA  0.94808
## Specificity            0.9413  0.85291  0.88389   0.8361  0.89218
## Pos Pred Value         0.8989  0.34141  0.51111       NA  0.46615
## Neg Pred Value         0.6443  0.91631  0.78587       NA  0.99425
## Prevalence             0.5102  0.13356  0.26591   0.0000  0.09033
## Detection Rate         0.2557  0.06607  0.08911   0.0000  0.08564
## Detection Prevalence   0.2845  0.19352  0.17435   0.1639  0.18373
## Balanced Accuracy      0.7212  0.67378  0.60951       NA  0.92013
```

The accuracy with decision trees is only 0.4986, which is really bad. Thus, I switch to a more accurate method, random forests.

Predicting with random forests

Random forests method in R has a built in cross-validation component. In this project, I use 5-fold cross-validation to select the optimized parameters for the model. Again, I build the model, test the model on the

validation set, and then calculate the confusion matrix.

```
# Predicting with random forests
# use 5-fold cross-validation
control_forests <- trainControl(method = "cv", number = 5, verboseIter = F)
# Build the random forests model
model_forests <- train(classe ~ ., data = training, method = "rf", trControl = control_forests)
```

```
## Loading required package: randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
# Predict on the validation set
predict_forests <- predict(model_forests, newdata = validation)
# Compute the confusion matrix
confusionMatrix(validation$classe, predict_forests)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    3  945    1    0    0
##          C    0    1  854    0    0
##          D    0    0    3  801    0
##          E    0    0    0    1  900
##
## Overall Statistics
##
##                Accuracy : 0.9982
##                  95% CI : (0.9965, 0.9992)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9977
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9979   0.9989   0.9953   0.9988   1.0000
## Specificity            1.0000   0.9990   0.9998   0.9993   0.9998
## Pos Pred Value         1.0000   0.9958   0.9988   0.9963   0.9989
## Neg Pred Value         0.9991   0.9997   0.9990   0.9998   1.0000
## Prevalence             0.2851   0.1929   0.1750   0.1635   0.1835
## Detection Rate         0.2845   0.1927   0.1741   0.1633   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9989   0.9990   0.9975   0.9990   0.9999
```

The accuracy for the random forests model is 0.9978, which is much better than the result with the decision trees model. Therefore, I decide to proceed with this model to predict the testing data. The out of sample error is 0.0022.

Predicting on testing data

To predict on the testing data, we need to load the testing set first. We also need to preprocess and clean the data, as we did for the training set.

```
# Load the testing set
testing <- read.csv('pml-testing.csv', na.strings = c("NA", "#DIV/0!", ""))
# Removing non-meaningful variables
testing <- testing[, -(1 : 5)]
# Removing variables with nearly zero variance
nzv <- nearZeroVar(testing)
testing <- testing[, -nzv]
# Removing variables which have lots of missing values
removeNA <- sapply(testing, function(x) mean(is.na(x))) > 0.9
testing <- testing[, removeNA == F]
dim(testing)
```

```
## [1] 20 54
```

After the testing set is cleaned, I apply the random forests model to predict the result.

```
# Predict on the testing set
predict_testing <- predict(model_forests, newdata = testing)
```

Using the code provided by the course website, the predicting results are saved into a bunch of .txt files.

```
# Output the predicting results
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)
  }
}
```

```
# Call the function to output
pml_write_files(predict_testing)
```