Weight Lifting Exercises Analysis

Abhijeet Sharma

Sunday, June 21, 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 activity relatively inexpensively. These type of de
vices are part of the quantified self movement - a group of enthusiasts who take measurem
ents about themselves regularly to improve their health, to find patterns in their behavi
or, or because they are tech geeks. One thing that people regularly do is quantify how mu
ch of a particular activity they do, but they rarely quantify how well they do it. In thi
s project, your goal will be to use data from accelerometers on the belt, forearm, arm, a
nd dumbell of 6 participants. They were asked to perform barbell lifts correctly and inco
rrectly in 5 different ways. More information is available from the website here: htt
p://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Data
set).
```

```
Data

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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.
If you use the document you create for this class for any purpose please cite them as the
y have been very generous in allowing their data to be used for this kind of assignment.
```

Step 1: Read data from CSV files

```
train_doc <- "pml-training.csv"
test_doc <- "pml-testing.csv"
training <- read.table(train_doc, header=TRUE, sep=",", quote="\"",
                       na.strings=c("NA", "<NA>", "", "#DIV/0!"), fill=TRUE, comment.cha
r="")
testing <- read.table(test_doc, header=TRUE, sep=",", quote="\"",
                       na.strings=c("NA", "<NA>", "", "#DIV/0!"), fill=TRUE, comment.cha
r="")
```

Step 2: Preprocessing: Eliminating unnecessary Columns and columns with more than 90% NA data.

```
# Identifying columns not required for the model
unnecessary_cols <-  c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2",
                       "cvtd_timestamp", "new_window", "num_window")
# Identifying Columns with more than 90% NA data are not relevant
na_cols <- names(which(colMeans(is.na(training)) > 0.90))
remove_cols <- c(unnecessary_cols, na_cols)
# Selecting only the relevant columns
colSel <- -which(names(training) %in% remove_cols)
# Subset the training dataset with only useful columns
training <- training[,colSel]
# Convert the outcome variable to a factor
training$classe <- as.factor(training$classe)

# Subsetting the testing dataset
testing <- testing[,colSel] # select only useful columns
testing <- testing[, -53] # remove the "problem_id" column

# convert all integer columns to numeric so that training and testing datasets
# are in sync
for (i in 1:52){
    training[, i] <- as.numeric(training[, i])
    testing[, i] <- as.numeric(testing[, i])
}
```

Step 3: Divide the original training set into 3 parts: actual training set, validation set for picking classifier and a test set for calculating out-of-sample errors(not to be confused with the “testing” dataset.)

```
set.seed(123)
# We allocate 20% data for training data (for improved performance)
inTrain <- createDataPartition(y=training$classe, p = 0.2, list=FALSE)
actualTrain <- training[inTrain,]
restTrain <- training[-inTrain,]

set.seed(123)
inTest <- createDataPartition(y=restTrain$classe, p = 0.5, list=FALSE)
# 40% data for validation set to select best classifier
actualClassifierValidation <- restTrain[inTest,]
# 40% data for test set to calculate out-of-sample errors
actualTest <- restTrain[-inTest,]
# Please note:
# in-sample accuracy: accuracy on seen data(using training set)
# out-of-sample accuracy: accuracy on unseen data(using actualTest set)
```

Step 4: Pick the best classifier among “rpart” and “rf” methods by training on actualTrain and predicting on actualClassifierValidation.

```
set.seed(123)
fitRpart <- train(classe ~ ., data=actualTrain, method="rpart")
```

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

```
predRpart <- predict(fitRpart, newdata=actualClassifierValidation[,1:52])
confRpart <- confusionMatrix(predRpart, actualClassifierValidation$classe)

set.seed(123)
fitRf <- train(classe ~ ., data=actualTrain, method="rf")
```

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

```
predRf <- predict(fitRf, newdata=actualClassifierValidation[,1:52])
confRf <- confusionMatrix(predRf, actualClassifierValidation$classe)
data.frame(Accuracy = c(confRpart$overall[1], confRf$overall[1]), row.names = c("rpart",
"rf"))
```

```
##        Accuracy
## rpart 0.5711556
## rf    0.9773220
```

Since Accuracy of Random Forest is higher, we select this model for our final predictions.

Step 5: Using the best classifier, predict the actualTest data and display the out-of-sample error.

```
set.seed(123)
predictions <- predict(fitRf, newdata=actualTest[,1:52])
confidence <- confusionMatrix(predictions, actualTest$classe)
confidence
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2209   46    3    0    0
##          B   10 1447   13    0    2
##          C    2   22 1327   31    8
##          D    3    3   25 1251    7
##          E    8    0    0    4 1425
##
## Overall Statistics
##
##                Accuracy : 0.9762
##                  95% CI : (0.9725, 0.9794)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9698
##  Mcnemar's Test P-Value : 1.661e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9897   0.9532   0.9700   0.9728   0.9882
## Specificity            0.9913   0.9960   0.9903   0.9942   0.9981
## Pos Pred Value         0.9783   0.9830   0.9547   0.9705   0.9916
## Neg Pred Value         0.9959   0.9889   0.9936   0.9947   0.9973
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2815   0.1844   0.1691   0.1594   0.1816
## Detection Prevalence   0.2878   0.1876   0.1772   0.1643   0.1832
## Balanced Accuracy      0.9905   0.9746   0.9802   0.9835   0.9932
```

The Out-of-sample Accuracy is 0.9763

Step 6: Calculate the predictions for the testing dataset (Assignment Submission)

```
test_pred <- predict(fitRf, newdata=testing)
answers <- as.character(test_pred)
```

Step 7: Write the predictions on seprate files

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

pml_write_files(answers)
```

Summary
We analysed the data and built a model to predict the "classe" variable.
We found that the Random forest classifier gave the best output.We trained the model using the default caret parameters.
We also found the below two observations:
1. We observed the cross validation error to be 0.9773
2. We observed the out of sample error to be 0.9763