Prediction Assignment - Predicting effectiveness of activity

Siva Sethu

Executive Summary

The goal of our analysis explained in this write-up is to create a machine learning algorithm to accurately predict how effectively (how well) the activity was performed.In this project, we are using data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. These participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways (which corresponds to the Classe variable in our data set). These classes are (as outlined in http://groupware.les.inf.puc-rio.br/har)
- Class A - exactly according to the specification
- Class B - throwing the elbows to the front
- Class C - lifting the dumbbell only halfway
- Class D - lowering the dumbbell only halfway
- Class E - throwing the hips to the front

Data Processing

The below steps are required to download the training & test datasets from the URL provided and assign to respective data frames. While populating the dataframe, we are ensuring all missing values are updated with "NA" values for subsequent processing

```
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile = "pml-t
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile = "pml-te

training_df <- read.csv("pml-training.csv", na.strings = c("NA", "","#DIV/0!"))

dim(training_df)
```

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

```
testing_df <- read.csv("pml-testing.csv", na.strings = c("NA", "","#DIV/0!"))
dim(testing_df)
```

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

Next, we load the required files required for our analysis

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

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

```
## Warning: package 'lattice' was built under R version 3.1.3
```

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

```
## Warning: package 'ggplot2' was built under R version 3.1.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.3
```

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

To ensure reproduceablity of the code, we will set the seed , so that the code produces the same result for each run.

To avoid overfitting and to include only predictors that have an impact on the outcome, we will remove columns with all missing value and also the first 7 columns which are primarily columns having only metadata about the individual observations and will not have any impact on the outcome

```
set.seed(12345)
training_df <- training_df[,colSums(is.na(training_df)) == 0]
training_df <- training_df[,-c(1:7)]

testing_df <- testing_df[,colSums(is.na(testing_df)) == 0]
testing_df <- testing_df[,-c(1:7)]
```

Creating training & validation datasets for training & cross-validation

Since the test dataset only has 20 observations, while the training dataset has 19622 observations, we will create a validation dataset from the training set, so that we can perform cross-validation of the model developed using the training dataset.

```
trn_index <- createDataPartition(y=training_df$classe, p=0.75, list=FALSE)
newTrainingData <- training_df[trn_index, ]
dim(newTrainingData)
```

```
## [1] 14718    53
```

```
newValidationData <- training_df[-trn_index, ]
dim(newValidationData)
```

```
## [1] 4904   53
```

Selecting the correct Prediction Model

It is important to identify the correct algorithm to build our prediction model. We will use two of the algorithms - classification tree(decision tree) and random Forest algorithms and check the accuracy of the models. We will then train our dataset with the most accurate algorithm to build our prediction model

Prediction Model and cross-validation using decision tree algorithm

```
mod_rpart <- train(classe ~ ., data = newTrainingData, method = "rpart")
```

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

```
pred_rpart <- predict(mod_rpart,newValidationData)

confusionMatrix(pred_rpart,newValidationData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 870 162  29  46  16
##          B 159 530  36 136 221
##          C 273 214 674 429 224
##          D  88  43 116 193  51
##          E   5   0   0   0 389
##
## Overall Statistics
##
##                Accuracy : 0.5416
##                  95% CI : (0.5275, 0.5556)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4245
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6237   0.5585   0.7883  0.24005  0.43174
## Specificity            0.9279   0.8604   0.7184  0.92732  0.99875
## Pos Pred Value         0.7747   0.4898   0.3716  0.39308  0.98731
## Neg Pred Value         0.8611   0.8904   0.9414  0.86155  0.88647
## Prevalence             0.2845   0.1935   0.1743  0.16395  0.18373
## Detection Rate         0.1774   0.1081   0.1374  0.03936  0.07932
## Detection Prevalence   0.2290   0.2206   0.3699  0.10012  0.08034
## Balanced Accuracy      0.7758   0.7095   0.7534  0.58368  0.71525
```

Prediction Model and cross-validation using Random Forest algorithm

```
mod_rf <- randomForest(classe ~ ., data = newTrainingData, method = "class")
pred_rf <- predict(mod_rf,newValidationData, type = "class")

confusionMatrix(pred_rf,newValidationData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    5    0    0    0
##          B    0  939    3    0    0
##          C    0    5  851    7    1
##          D    0    0    1  797    4
##          E    0    0    0    0  896
##
## Overall Statistics
##
##                Accuracy : 0.9947
##                  95% CI : (0.9922, 0.9965)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9933
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9895   0.9953   0.9913   0.9945
## Specificity            0.9986   0.9992   0.9968   0.9988   1.0000
## Pos Pred Value         0.9964   0.9968   0.9850   0.9938   1.0000
## Neg Pred Value         1.0000   0.9975   0.9990   0.9983   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1915   0.1735   0.1625   0.1827
## Detection Prevalence   0.2855   0.1921   0.1762   0.1635   0.1827
## Balanced Accuracy      0.9993   0.9944   0.9961   0.9950   0.9972
```

Finalizing the correct prediction model & Out of sample error estimation

As could be evidenced in the outputs above, the prediction model built using Random Forest algorithm is 99.57% (accuracy of 0.9957) accurate when applied to the cross-validation dataset. The expected out of Sample error will be 1 - accuracy, which is equal to 0.0043 or 0.43%. With 99.57% accuracy, we can confidently apply this model on testing dataset.

Applying the model to predict outcome on testing dataset

```
pred_test <- predict(mod_rf,testing_df, type = "class")
pred_test
```

```
##  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
```