Prediction Assignment Writeup

Author: Thomas Lee Wai Siong

Date: Friday, December 25, 2015

Source: PDF, Markdown, Html are available at rpubs

Executive Summary

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 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 because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the **belt**, **forearm**, **arm**, and **dumbell** of **6** participants. They were asked to perform barbell lifts correctly and incorrectly in **5** different ways. More information is available from the website here (see the section on the Weight Lifting Exercise Dataset).

Dataset

The data for this project come from this source.

- The training data for this project are available here
- The test data are available here

Data Loading From CSV

```
TrainData<-read.csv("pml-training.csv", header=T, na.strings=c("NA", "#DIV/0!"))
TestData<-read.csv("pml-testing.csv", header=T, na.string=c("NA", "#DIV/0!"))
```

Basic Data Exploration

```
dim(TrainData)
dim(TestData)

head(TrainData)
head(TestData)

summary(TrainData)
summary(TestData)

str(TrainData)
str(TestData)
```

Cleaning The Data

From above, we can see that the data contains mostly numerical features. However, many of them contain **nonstandard coded missing values**, standard **NA**, empty strings **""**, and error expressions **"#DIV/0!"**.

All variables with at least one **NA** have to be excluded from the analysis.

```
## before clean
dim(TrainData)
```

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

```
NoNATrainData<-TrainData[, apply(TrainData, 2, function(x) !any(is.na(x)))]
## after clean
dim(NoNATrainData)
```

```
## [1] 19622    60
```

Next, variables related to time and user information also have to be excluded from the analysis.

```
## before clean
dim(NoNATrainData)
```

```
## [1] 19622    60
```

```
CleanTrainData<-NoNATrainData[,-c(1:8)]
## after clean
dim(CleanTrainData)
```

```
## [1] 19622    52
```

Same variables will be maintained in the test data set in order to be used for predicting the 20 test cases provided.

```
## before clean
dim(TestData)
```

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

```
CleanTestData<-TestData[,names(CleanTrainData[,-52])]
## after clean
dim(CleanTestData)
```

```
## [1] 20 51
```

Besides, the outcome variable classe with values are:

- exactly according to the specification (Class A),
- throwing the elbows to the front (Class B),
- lifting the dumbbell only halfway (Class C),
- lowering the dumbbell only halfway (Class D)
- throwing the hips to the front (Class E).

Data Partitioning and Prediction Process

In order to have a better model validation and performance, dataset will be further seperated into 2 sub dataset which are:

- a training set containing 60% of the data
- a validation set containing 40% of the data (Please take note that this validation set will be held out at the end and the testing set will be used for all model selection only.)

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.2.3
```

```
library(lattice)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.3
```

```
inTrain<-createDataPartition(y=CleanTrainData$classe, p=0.60,list=F)
train_set <- CleanTrainData[inTrain, ]
test_set <- CleanTrainData[-inTrain, ]
#Training and test set dimensions
dim(train_set)
```

```
## [1] 11776    52
```

```
dim(test_set)
```

```
## [1] 7846   52
```

Exploratory Analysis

We will focus on the traning set at this level only.

Firstly, we have excluded missing values during data cleansing. This allow us to only consider any variable that contain no missing values for the model that we are going to apply later.

Model Selection

The models we try are:

- Random Forest
- Gradient Boosted Machines

Cross-validation

To assess model performance, we perform 5-fold cross-validation. This gives a good estimate for the out-of-sample accuracy.

Model Selection 1 - Random Forest

We show the procedure Random Forest.

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.3
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(13333)
fitControl2<-trainControl(method="cv", number=5, allowParallel=T, verbose=T)
rffit<-train(classe~.,data=train_set, method="rf", trControl=fitControl2, verbose=F)
```

```
## + Fold1: mtry= 2
## - Fold1: mtry= 2
## + Fold1: mtry=26
## - Fold1: mtry=26
## + Fold1: mtry=51
## - Fold1: mtry=51
## + Fold2: mtry= 2
## - Fold2: mtry= 2
## + Fold2: mtry=26
## - Fold2: mtry=26
## + Fold2: mtry=51
## - Fold2: mtry=51
## + Fold3: mtry= 2
## - Fold3: mtry= 2
## + Fold3: mtry=26
## - Fold3: mtry=26
## + Fold3: mtry=51
## - Fold3: mtry=51
## + Fold4: mtry= 2
## - Fold4: mtry= 2
## + Fold4: mtry=26
## - Fold4: mtry=26
## + Fold4: mtry=51
## - Fold4: mtry=51
## + Fold5: mtry= 2
```

```
## - Fold5: mtry= 2
## + Fold5: mtry=26
## - Fold5: mtry=26
## + Fold5: mtry=51
## - Fold5: mtry=51
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 26 on full training set
```

```
rffit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, verbose = ..1)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 26
##
##         OOB estimate of  error rate: 0.82%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3342    4    1    0    1 0.001792115
## B   20 2247   12    0    0 0.014041246
## C    0   12 2036    5    1 0.008763389
## D    0    0   27 1901    2 0.015025907
## E    0    1    5    6 2153 0.005542725
```

```
class(rffit)
```

```
## [1] "train"         "train.formula"
```

```
predict_rf<-predict(rffit, newdata=test_set)
#we can see some statistics of the fit here:
confusionMatrix(predict_rf, test_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    7    0    0    0
##          B    2 1506    7    0    0
##          C    0    4 1354   17    0
##          D    0    0    7 1267    9
##          E    0    1    0    2 1433
##
## Overall Statistics
##
##                Accuracy : 0.9929
##                  95% CI : (0.9907, 0.9946)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.991
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9921   0.9898   0.9852   0.9938
## Specificity            0.9988   0.9986   0.9968   0.9976   0.9995
## Pos Pred Value         0.9969   0.9941   0.9847   0.9875   0.9979
## Neg Pred Value         0.9996   0.9981   0.9978   0.9971   0.9986
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1919   0.1726   0.1615   0.1826
## Detection Prevalence   0.2851   0.1931   0.1752   0.1635   0.1830
## Balanced Accuracy      0.9989   0.9953   0.9933   0.9914   0.9966
```

Generated algorithm above examined the accuracy and estimated error of prediction in the partition of training set. It gaves 5-fold an accuracy of 99% with a 95% CI [0.9925, 0.9959] was achieved accompanied by a Kappa value of 0.99. Random Forest classifier has given extremely high accuracy estimate of 99%

```
predict_20<-predict(rffit, newdata=CleanTestData)
# Output for the prediction of the 20 cases provided
predict_20
```

```
##  [1] 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
```

Model Selection 2 - Gradient Boosted Machines

```
fitControl2<-trainControl(method="cv", number=5, allowParallel=T, verbose=T)
gmbfit<-train(classe~.,data=train_set, method="gbm", trControl=fitControl2, verbose=F)
```

```
## Loading required package: gbm
```

```
## Warning: package 'gbm' was built under R version 3.2.3
```

```
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1
## Loading required package: plyr
```

```
## Warning: package 'plyr' was built under R version 3.2.1
```

```
## + Fold1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold1: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold2: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold2: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold3: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold3: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold4: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold4: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=1, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=2, n.minobsinnode=10, n.trees=150
## + Fold5: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## - Fold5: shrinkage=0.1, interaction.depth=3, n.minobsinnode=10, n.trees=150
## Aggregating results
## Selecting tuning parameters
## Fitting n.trees = 150, interaction.depth = 3, shrinkage = 0.1, n.minobsinnode = 10 on full training s
```

```
gmbfit$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 51 predictors of which 44 had non-zero influence.
```

```
class(gmbfit)
```

```
## [1] "train"         "train.formula"
```

```
predict_gmb<-predict(gmbfit, newdata=test_set)
#we can see some statistics of the fit here:
confusionMatrix(predict_gmb, test_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 2205   44    0    1    3
##          B   18 1431   39    2   14
##          C    5   34 1312   46    7
##          D    2    4   14 1223   20
##          E    2    5    3   14 1398
##
## Overall Statistics
##
##                Accuracy : 0.9647
##                  95% CI : (0.9604, 0.9687)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9553
##  Mcnemar's Test P-Value : 9.451e-06
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9879   0.9427   0.9591   0.9510   0.9695
## Specificity            0.9914   0.9885   0.9858   0.9939   0.9963
## Pos Pred Value         0.9787   0.9515   0.9345   0.9683   0.9831
## Neg Pred Value         0.9952   0.9863   0.9913   0.9904   0.9932
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2810   0.1824   0.1672   0.1559   0.1782
## Detection Prevalence   0.2872   0.1917   0.1789   0.1610   0.1812
## Balanced Accuracy      0.9897   0.9656   0.9724   0.9725   0.9829
```

```
predict_train<-predict(gmbfit, newdata=train_set)
#we can see some statistics of the fit here:
confusionMatrix(predict_train, train_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3310   46    0    0    4
##          B   26 2194   44    3   13
##          C    8   36 1990   64   21
##          D    2    0   18 1856   24
##          E    2    3    2    7 2103
##
## Overall Statistics
##
##                Accuracy : 0.9726
##                  95% CI : (0.9695, 0.9754)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9653
##  Mcnemar's Test P-Value : 1.858e-12
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9886   0.9627   0.9688   0.9617   0.9714
## Specificity            0.9941   0.9909   0.9867   0.9955   0.9985
## Pos Pred Value         0.9851   0.9623   0.9391   0.9768   0.9934
## Neg Pred Value         0.9955   0.9910   0.9934   0.9925   0.9936
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2811   0.1863   0.1690   0.1576   0.1786
## Detection Prevalence   0.2853   0.1936   0.1799   0.1613   0.1798
## Balanced Accuracy      0.9914   0.9768   0.9778   0.9786   0.9850
```

Boosted algorithm above was also run to confirm and be able to compare predictions. It gaves 5-fold an accuracy of 96% with a 95% CI [0.9547, 0.9636] was achieved accompanied by a Kappa value of 0.95. Gradient Boosted Machines approach presented less accuracy (95%) However, the predictions for the 20 test cases were compared match was same for both ran algorimths above.

```
predict_20<-predict(gmbfit, newdata=CleanTestData)
# Output for the prediction of the 20 cases provided
predict_20
```

```
##  [1] 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
```

Applying the final model

Finally, we apply the model to the 20 unlabeled assignment cases. The script below was used to obtain single text files to be uploaded to the courses web site to comply with the submission assigment. 20 out of 20 hits also confirmed the accuracy of the obtained models.

```
getwd()
```

```
## [1] "C:/Users/Lee/Documents/R/8. Machine Learning/Prediction Assignment Writeup"
```

```
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)
  }
}
pml_write_files(predict_20)
```