Assignment Writeup - Prediction of how was Exercise Performed

Human activity recognition research has traditionally focused on discriminating between different activities. However, the “how (well)” investigation has only received little attention so far, even though it potentially provides useful information for a large variety of applications,such as sports training (http://groupware.les.inf.puc-rio.br/har).

For the prediction of how welll individuals performed the assigned exercise six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: 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) and throwing the hips to the front (Class E).

This report aims to use machine learning algoritmhs to predict the class of exercise the individuals was performing by using meaurements available from devices such as Jawbone Up, Nike FuelBand, and Fitbit.

Data Cleaning

The data for this project come was obteined from http://groupware.les.inf.puc-rio.br/har. Two data set were available a training set and a test set for which 20 individuals without any classification for the class of exercise was available.

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

Training data was partitioned and preprocessed using the code described below. In brief, all variables with at least one “NA” were excluded from the analysis. Variables related to time and user information were excluded for a total of 51 variables and 19622 class measurements. Same variables were maintained in the test data set (Validation dataset) to be used for predicting the 20 test cases provided.

```
## NA exclusion for all available variables
noNApmlTrain<-pmlTrain[, apply(pmlTrain, 2, function(x) !any(is.na(x)))]
dim(noNApmlTrain)

## [1] 19622    60

## variables with user information, time and undefined
cleanpmlTrain<-noNApmlTrain[,-c(1:8)]
dim(cleanpmlTrain)

## [1] 19622    52
## 20 test cases provided clean info - Validation data set
cleanpmltest<-pmlTest[,names(cleanpmlTrain[,-52])]
dim(cleanpmltest)

## [1] 20 51
```

Data Partitioning and Prediction Process

The cleaned downloaded data set was subset in order to generate a test set independent from the 20 cases provided set. Partitioning was performed to obtain a 75% training set and a 25% test set.

```
#data cleaning
library(caret)

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

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

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

inTrain<-createDataPartition(y=cleanpmlTrain$classe, p=0.75,list=F)
training<-cleanpmlTrain[inTrain,]
test<-cleanpmlTrain[-inTrain,]
#Training and test set dimensions
dim(training)

## [1] 14718    52

dim(test)

## [1] 4904   52
```

Results and Conclusions

Random forest trees were generated for the training dataset using cross-validation. Then the generated algorithm was examnined under the partitioned training set to examine the accuracy and estimated error of prediction. By using 51 predictors for five classes using cross-validation at a 5-fold an accuracy of 99.2% with a 95% CI [0.989-0.994] was achieved accompanied by a Kappa value of 0.99.

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

## Loading required package: randomForest

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

```
## + 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
```

```
predrf<-predict(rffit, newdata=test)
confusionMatrix(predrf, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1388    4    0    0    0
##          B    7  941    3    0    0
##          C    0    4  850    5    1
##          D    0    0    2  799    5
##          E    0    0    0    0  895
##
## Overall Statistics
##
##                Accuracy : 0.9937
##                  95% CI : (0.991, 0.9957)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.992
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9950   0.9916   0.9942   0.9938   0.9933
## Specificity            0.9989   0.9975   0.9975   0.9983   1.0000
## Pos Pred Value         0.9971   0.9895   0.9884   0.9913   1.0000
## Neg Pred Value         0.9980   0.9980   0.9988   0.9988   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2830   0.1919   0.1733   0.1629   0.1825
## Detection Prevalence   0.2838   0.1939   0.1754   0.1644   0.1825
## Balanced Accuracy      0.9969   0.9945   0.9958   0.9960   0.9967
```

```
pred20<-predict(rffit, newdata=cleanpmltest)
# Output for the prediction of the 20 cases provided
pred20
```

```
## [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
```

A boosting algorithm was also run to confirm and be able to compare predictions. Data is not shown but the boosting approach presented less accuracy (96%) (Data not shown). However, when the predictions for the 20 test cases were compared match was same for both ran algorithms.

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

```
## Aggregating results
## Selecting tuning parameters
## Fitting n.trees = 150, interaction.depth = 3, shrinkage = 0.1, n.minobsinnode = 10 on full training set
```

```
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"
```

```
predgmb<-predict(gmbfit, newdata=test)
confusionMatrix(predgmb, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1368   25    0    1    1
##          B   15  890   23    2    8
##          C    4   29  824   24   12
##          D    5    2    7  767   12
##          E    3    3    1   10  868
##
## Overall Statistics
##
##                Accuracy : 0.9619
##                  95% CI : (0.9561, 0.9671)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9518
##  Mcnemar's Test P-Value : 0.0004092
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9806   0.9378   0.9637   0.9540   0.9634
## Specificity            0.9923   0.9879   0.9830   0.9937   0.9958
## Pos Pred Value         0.9806   0.9488   0.9227   0.9672   0.9808
## Neg Pred Value         0.9923   0.9851   0.9923   0.9910   0.9918
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2790   0.1815   0.1680   0.1564   0.1770
## Detection Prevalence   0.2845   0.1913   0.1821   0.1617   0.1805
## Balanced Accuracy      0.9865   0.9628   0.9734   0.9738   0.9796
```

```
predtrain<-predict(gmbfit, newdata=training)
confusionMatrix(predtrain, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4145   80    0    1    2
##          B   27 2709   53   11   13
##          C   10   53 2479   61   24
##          D    1    1   30 2330   28
##          E    2    5    5    9 2639
##
## Overall Statistics
##
##                Accuracy : 0.9717
##                  95% CI : (0.9689, 0.9744)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9642
##  Mcnemar's Test P-Value : 3.334e-13
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9904   0.9512   0.9657   0.9660   0.9752
## Specificity            0.9921   0.9912   0.9878   0.9951   0.9983
## Pos Pred Value         0.9804   0.9630   0.9437   0.9749   0.9921
## Neg Pred Value         0.9962   0.9883   0.9927   0.9933   0.9944
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2816   0.1841   0.1684   0.1583   0.1793
## Detection Prevalence   0.2873   0.1911   0.1785   0.1624   0.1807
## Balanced Accuracy      0.9913   0.9712   0.9768   0.9806   0.9867
```

```
predtrain<-predict(gmbfit, newdata=training)
confusionMatrix(predtrain, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4145   80    0    1    2
##          B   27 2709   53   11   13
##          C   10   53 2479   61   24
##          D    1    1   30 2330   28
##          E    2    5    5    9 2639
##
## Overall Statistics
##
##                Accuracy : 0.9717
##                  95% CI : (0.9689, 0.9744)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9642
##  Mcnemar's Test P-Value : 3.334e-13
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9904   0.9512   0.9657   0.9660   0.9752
## Specificity            0.9921   0.9912   0.9878   0.9951   0.9983
## Pos Pred Value         0.9804   0.9630   0.9437   0.9749   0.9921
## Neg Pred Value         0.9962   0.9883   0.9927   0.9933   0.9944
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2816   0.1841   0.1684   0.1583   0.1793
## Detection Prevalence   0.2873   0.1911   0.1785   0.1624   0.1807
## Balanced Accuracy      0.9913   0.9712   0.9768   0.9806   0.9867
```