Prediction Assignment

Synopsis

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. This project will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The goal is to use that data to predict the manner in which the participants did the exercise:

```
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
```

For more information, see http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har)

Prepare Data

Read data and see number of rows and columns

```
alltrain = read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
testing = read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")

dim(alltrain)
```

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

```
dim(testing)
```

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

Slice training into training and validation

```
library(caret)
```

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

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

```
set.seed(1342)
inTrain <- createDataPartition(y=alltrain$classe,p=0.7, list=FALSE)
training <- alltrain[inTrain,];
validation <- alltrain[-inTrain,]
```

Take a quick look at training data

```
head(training)
```

The first seven columns are identifiers such as names and time stamps, so I will remove them before training. I will also remove columns that don't have numeric values for all rows.

```
library(caret)
classe <- training$classe
training <- training[,-seq(1:7)]
training <- training[, colSums(is.na(training)) == FALSE]
training <- training[, sapply(training, is.numeric)]
training$classe <- classe
dim(training)
```

```
## [1] 13737    53
```

```
summary(training$classe)
```

```
##    A    B    C    D    E
## 3906 2658 2396 2252 2525
```

```
head(training)
```

Fit a Model

Just as in the original paper, I will use Random Forest algorithm

```
library(ggplot2)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
set.seed(1837)
modFit <- train(classe ~ ., method="rf", data=training, ntree = 50)
```

```
## Loading required package: randomForest
```

```
## 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
```

```
modFit
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9854040  0.9815343  0.002157943  0.002726887
##   27    0.9868301  0.9833396  0.002092796  0.002644573
##   52    0.9772142  0.9711747  0.004728724  0.005978407
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
# fancyRpartPlot(modFit$finalModel)
```

Model Performance

With Validation set

The accuracy is estimated at 99.17%

```
valdPred <- predict(modFit, validation)
confusionMatrix(valdPred, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669    4    0    0    0
##          B    3 1131    3    0    1
##          C    2    0 1018   11    0
##          D    0    4    5  952    5
##          E    0    0    0    1 1076
##
## Overall Statistics
##
##                Accuracy : 0.9934
##                  95% CI : (0.991, 0.9953)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9916
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9930   0.9922   0.9876   0.9945
## Specificity            0.9991   0.9985   0.9973   0.9972   0.9998
## Pos Pred Value         0.9976   0.9938   0.9874   0.9855   0.9991
## Neg Pred Value         0.9988   0.9983   0.9984   0.9976   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2836   0.1922   0.1730   0.1618   0.1828
## Detection Prevalence   0.2843   0.1934   0.1752   0.1641   0.1830
## Balanced Accuracy      0.9980   0.9958   0.9948   0.9924   0.9971
```

With Testing set

When this model was used against the testing data, it scored 20 out of 20 correct.

```
predict(modFit,newdata=testing)
```

```
##  [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
```