Peer-graded Assignment: Prediction Assignment Writeup

Project 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 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: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

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://web.archive.org/web/20161224072740/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 they have been very generous in allowing their data to be used for this kind of assignment.

```
> trainRaw <- read.csv("C:/Desktop/pml-training.csv")
> testRaw <- read.csv("C:/Desktop/pml-testing.csv")
> dim(trainRaw)
[1] 19622   160
> dim(testRaw)
[1]  20 160
> sum(complete.cases(trainRaw))
[1] 406
##data cleaning and modeling process
> trainRaw <- trainRaw[, colSums(is.na(trainRaw)) == 0]
> testRaw <- testRaw[, colSums(is.na(testRaw)) == 0]
```

```
> classe <- trainRaw$classe
> trainRemove <- grepl("^X|timestamp|window", names(trainRaw))

> trainRaw <- trainRaw[, !trainRemove]
> trainCleaned <- trainRaw[, sapply(trainRaw, is.numeric)]
> trainCleaned$classe <- classe
> testRemove <- grepl("^X|timestamp|window", names(testRaw))

> testRaw <- testRaw[, !testRemove]
> testCleaned <- testRaw[, sapply(testRaw, is.numeric)]
> set.seed(22519) # For reproducibile purpose
> inTrain <- createDataPartition(trainCleaned$classe, p=0.70, list=F)

> trainData <- trainCleaned[inTrain, ]
> testData <- trainCleaned[-inTrain, ]

> controlRf <- trainControl(method="cv", 5)
> modelRf <- train(classe ~ ., data=trainData, method="rf", trControl=
controlRf, ntree=250)
> modelRf
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 10989, 10989, 10991, 10990, 10989
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD   Kappa SD
##    2    0.9904636  0.9879361  0.0006534224  0.0008263772
##   27    0.9913374  0.9890405  0.0015731292  0.0019917940
##   52    0.9850766  0.9811193  0.0027732182  0.0035098533
##
## Accuracy was used to select the optimal model using  the largest val
ue.
## The final value used for the model was mtry = 27.
> predictRf <- predict(modelRf, testData)
> confusionMatrix(testData$classe, predictRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    1    0    0    1
##          B    8 1127    4    0    0
##          C    0    1 1020    5    0
##          D    0    0   14  949    1
##          E    0    0    0    6 1076
##
## Overall Statistics
##
##                Accuracy : 0.993
##                  95% CI : (0.9906, 0.995)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9912
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9952   0.9982   0.9827   0.9885   0.9981
## Specificity            0.9995   0.9975   0.9988   0.9970   0.9988
## Pos Pred Value         0.9988   0.9895   0.9942   0.9844   0.9945
## Neg Pred Value         0.9981   0.9996   0.9963   0.9978   0.9996
## Prevalence             0.2855   0.1918   0.1764   0.1631   0.1832
## Detection Rate         0.2841   0.1915   0.1733   0.1613   0.1828
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9974   0.9979   0.9907   0.9927   0.9984
> accuracy <- postResample(predictRf, testData$classe)
> accuracy
##  Accuracy     Kappa
## 0.9930331 0.9911870
> oose <- 1 - as.numeric(confusionMatrix(testData$classe, predictRf)
$overall[1])
> oose
[1] 0.006966865
##predict data test
> result <- predict(modelRf, testCleaned[, -length(names(testCleane
d))])
> result
```