Prediction Assignment Writeup

Instructions

In this project, we use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants that 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).

The goal is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Summary of Data

```
rm(list = ls())
library(lattice);
library(ggplot2);
library(caret);
library(randomForest);

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

library(rpart);
library(rpart.plot);
library(rattle)

## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'

## The following object is masked from 'package:randomForest':
##
##     importance
```

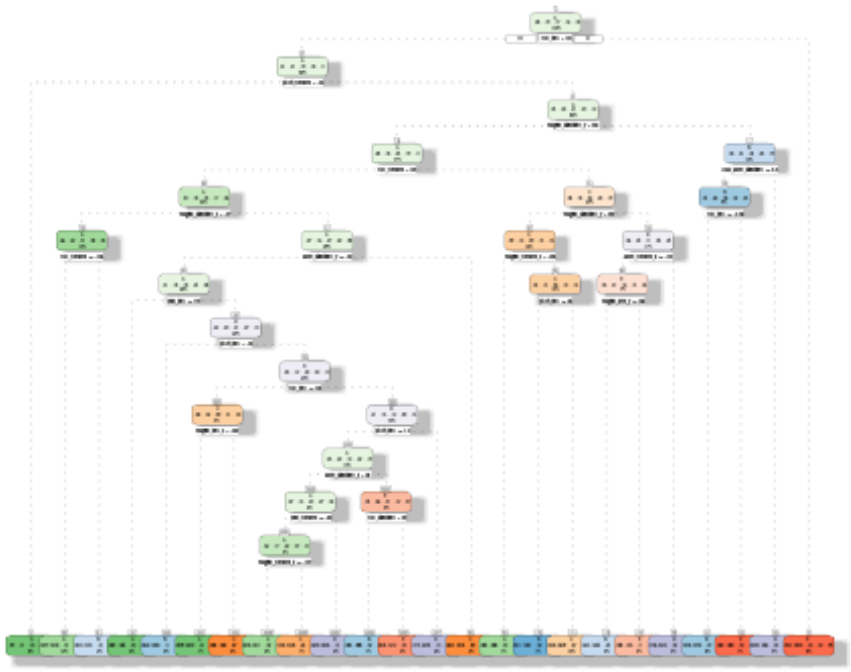
Load and partition data

```
set.seed(4321)

# data load and clean up
trainfileUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
testfileUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"

trainfilename<-"./pml-training.csv"
if(!file.exists(trainfilename)) {
  download.file(trainfileURL, trainfilename, method = "curl")
}
testfilename<-"./pml-testing.csv"
if(!file.exists(testfilename)) {
  download.file(testfileURL, testfilename, method = "curl")
}

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

# Perform exploratory analysis -
dim(training);

## [1] 19622   160

# str(training);
# head(training);

# Remove variables with near zero variance
training <- training[,colSums(is.na(training)) == 0]
testing <- testing[,colSums(is.na(testing)) == 0]

# Remove columns that are not predictors.
training <- training[,-c(1:7)]
testing <- testing[,-c(1:7)]

#str(training)
dim(training)

## [1] 19622    53
```

```
training$classe = factor(training$classe)
#testing$classe = factor(testing$classe)

# Create a training set and validation data set
inTrain <- createDataPartition(y=training$classe, p=0.7, list=FALSE)
trainSet <- training[inTrain, ]
validSet <- training[-inTrain, ]
```

Prediction 1: Recursive Partitioning and Regression Trees

```
modDT <- rpart(classe ~ ., data=trainSet, method="class")

# Plot the Decision Tree
fancyRpartPlot(modDT)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```




```
predDT <- predict(modDT, validSet, type = "class")

confusionMatrix(predDT, validSet$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1476  160   15   45   13
##          B   57  731   86   87   85
```

```
##          C   44  125  820  154  129
##          D   71   73   78  605   72
##          E   26   50   27   73  783
##
## Overall Statistics
##
##                Accuracy : 0.7502
##                  95% CI : (0.7389, 0.7612)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6839
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8817   0.6418   0.7992   0.6276   0.7237
## Specificity            0.9447   0.9336   0.9070   0.9403   0.9634
## Pos Pred Value         0.8637   0.6989   0.6447   0.6730   0.8165
## Neg Pred Value         0.9526   0.9157   0.9553   0.9280   0.9393
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2508   0.1242   0.1393   0.1028   0.1331
## Detection Prevalence   0.2904   0.1777   0.2161   0.1528   0.1630
## Balanced Accuracy      0.9132   0.7877   0.8531   0.7839   0.8435
```

Prediction 2: Random Forest

```
modRF <- randomForest(classe ~. , data=trainSet, method="class")

predRF <- predict(modRF, validSet, type = "class")

confusionMatrix(predRF, validSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    3    0    0    0
##          B    2 1135    4    0    0
##          C    0    1 1021    7    0
##          D    0    0    1  956    1
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9966
##                  95% CI : (0.9948, 0.9979)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9957
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9965   0.9951   0.9917   0.9991
## Specificity            0.9993   0.9987   0.9984   0.9996   0.9998
## Pos Pred Value         0.9982   0.9947   0.9922   0.9979   0.9991
## Neg Pred Value         0.9995   0.9992   0.9990   0.9984   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1929   0.1735   0.1624   0.1837
## Detection Prevalence   0.2846   0.1939   0.1749   0.1628   0.1839
## Balanced Accuracy      0.9990   0.9976   0.9967   0.9956   0.9994
```

From the confusion matrix, the Random Forest (99%) perform better than Decision Tree (75%) in accuarcy using the validation set. Hence, we will use the Random Forest on the testing set

Perform prediction on testing set

```
predTest <- predict(modRF, testing, type="class")
predTest
```

```
##  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
```