Prediction Assignment Writeup

Background

We will use data collected by four accelerometers placed on the belt, forearm, arm, and dumbell of 6 young healthy participant. There are 10 different correct and incorrect ways to lift barbells. Each accelerometer The websource of the article and more information can be found at http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset). The task is to "predict the manner in which they did the exercises." The prediction model will be used to predict 20 different test cases. The following paper details the WLE dataset and the study.

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Getting and Cleaning Data

The training and testing data can be downloaded from the links provided below.

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

```
library(caret)
library(rpart)
library(e1071)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)
library(AppliedPredictiveModeling)
library(knitr)
```

```
set.seed(222)
#setwd("./doc/data-science/data-science-track/8-ds-machine/project")
test1 = read.csv("./pml-testing.csv", header=T)
train1 = read.csv("./pml-training.csv", header=T)

dim(test1)
```

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

```
dim(train1)
```

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

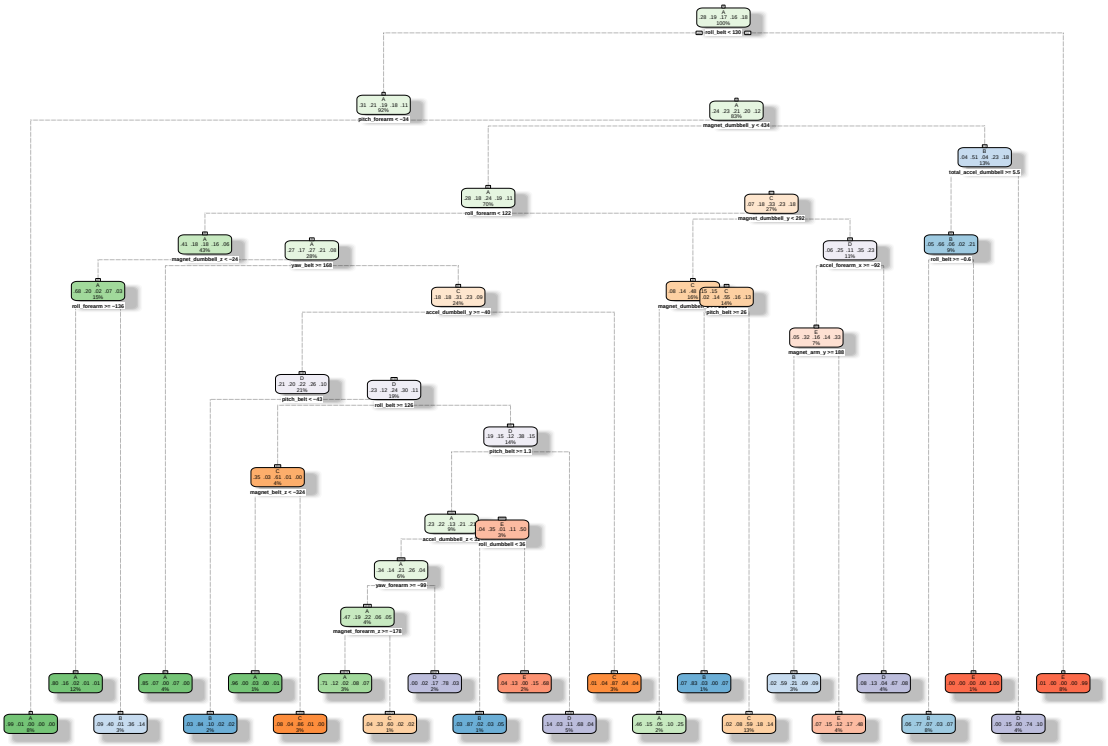
```
#From test data: Remove the first 7 variables as they are not relevant, remove columns with NA. Get the
varNames1 = names(test1[,colSums(is.na(test1)) == 0])[8:160]
varNames1 = varNames1[1:53]

varNames2 = names(train1[,colSums(is.na(train1)) == 0])[8:160]
```

```
test2 = test1[,c(varNames1)]

# Clean the training data to be similar to the testing data.
train2 = train1[,c(varNames1[1:52],"classe")]

#Partition train data into 2 sets: 60% for training, 40% for testing.
Train60 = createDataPartition(train2$classe, p=0.6, list=FALSE)
Testing = train2[-Train60, ]
Training = train2[Train60, ]
```

Training Using Decision Trees and Random Forest

```
#modFit = train(classe ~., method="rpart",data=Training )
modFitTrees = rpart(classe ~ ., method="class",data=Training)
fancyRpartPlot(modFitTrees)
```

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



Rattle 2016−Aug−08 02:50:42 tom

```
modFitForest = randomForest(classe ~ .,data=Training)
```

Confusion Matrix and Statistics

```
predictTrees = predict(modFitTrees,Testing,type="class")
predictForest = predict(modFitForest,Testing,type="class")

CMTrees = confusionMatrix(predictTrees,Testing$classe)
CMForest = confusionMatrix(predictForest,Testing$classe)

CMTrees
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1996  262   37   87   50
##          B   93  965  140  115  157
##          C   43  125 1080  184  150
##          D   63   96   82  824   75
##          E   37   70   29   76 1010
##
## Overall Statistics
##
##                Accuracy : 0.7488
##                  95% CI : (0.739, 0.7584)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6812
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8943   0.6357   0.7895   0.6407   0.7004
## Specificity            0.9223   0.9202   0.9225   0.9518   0.9669
## Pos Pred Value         0.8207   0.6565   0.6827   0.7228   0.8265
## Neg Pred Value         0.9564   0.9133   0.9540   0.9311   0.9348
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2544   0.1230   0.1376   0.1050   0.1287
## Detection Prevalence   0.3100   0.1874   0.2016   0.1453   0.1557
## Balanced Accuracy      0.9083   0.7780   0.8560   0.7963   0.8337
```

```
CMForest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228   16    0    0    0
##          B    4 1498    9    0    0
##          C    0    4 1359   11    3
##          D    0    0    0 1273    5
```

```
##          E    0    0    0    2 1434
##
## Overall Statistics
##
##                Accuracy : 0.9931
##                  95% CI : (0.991, 0.9948)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9913
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9868   0.9934   0.9899   0.9945
## Specificity            0.9971   0.9979   0.9972   0.9992   0.9997
## Pos Pred Value         0.9929   0.9914   0.9869   0.9961   0.9986
## Neg Pred Value         0.9993   0.9968   0.9986   0.9980   0.9988
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1909   0.1732   0.1622   0.1828
## Detection Prevalence   0.2860   0.1926   0.1755   0.1629   0.1830
## Balanced Accuracy      0.9977   0.9924   0.9953   0.9946   0.9971
```

Final Results with the Test Data

The random forest fit has a better accuracy at 99.31% than the decision trees fit with accuracy at 74.88%. The out-of-sample error for random forest = 1 - 0.9931 = 0.0069. The out-of-sample error for decision trees = 1 - 0.7488 = 0.2512. I will use the random forest to predict the 20 cases in the test data as shown below. The prediction results were submitted to the course project prediction quiz.

```
prediction = predict(modFitForest,test2,type="class")
prediction
```

```
##  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
```