PRACTICAL MACHINE LEARNING ASSIGNMENT

~~~~~~

Yogesh Rampariya

4 September, 2019

Introduction

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://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data Source

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://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.

Loading the Dataset

Download the data files from the Internet and load them into two data frames. We ended up with a training dataset and a 20 observations testing dataset that will be submitted to Coursera

```
library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```
~~~~~~

```
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:rattle':
##
##     importance

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

```
library(ipred)
training <- read.csv("training.csv")
```

Cleaning Data

Identify all column having No variation which doesnot affect outcome and doesnot participate actively in deciding final outcome

```
trainnzv <- nearZeroVar(training)
trainnzv
```

```
##  [1]   6  12  13  14  15  16  17  20  23  26  51  52  53  54  55  56  57
## [18]  58  59  69  70  71  72  73  74  75  78  79  81  82  87  88  89  90
## [35]  91  92  95  98 101 125 126 127 128 129 130 131 133 134 136 137 139
## [52] 142 143 144 145 146 147 148 149 150
```

```
## Remove all unwanted column

newdata <- training[, - trainnzv]
ncol(newdata)
```

```
## [1] 100
```

Case 1 (Remove all NA from data)

```
any(is.na(newdata))
```

```
## [1] TRUE
```

```r
newdata1 <- complete.cases(newdata)
table(as.factor(newdata1))
```

```
##
## FALSE  TRUE
## 19216   406
```

```r
## Considering only active rows
newdata2 <- na.omit(newdata)

## Removing 1st and 2nd column as it doesnot affect outcome classe variable
newdata2 <- newdata2[, -c(1,2)]

nrow(newdata2)
```

```
## [1] 406
```

```r
ncol(newdata2)
```

```
## [1] 98
```

Case 1 Data Partition and Model Building

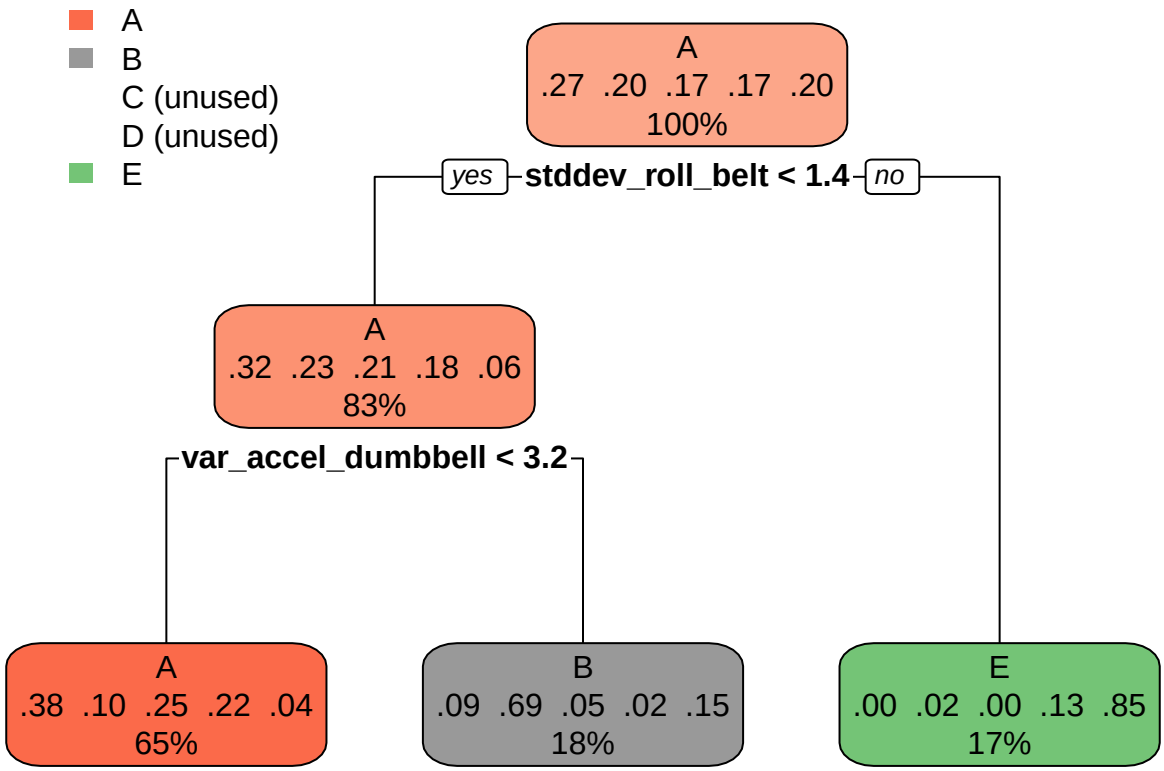
```r
datapart <- createDataPartition(newdata2$classe, p = 0.75, list = FALSE)
traindata <- newdata2[datapart,]
testdata <- newdata2[- datapart,]
## Decision Tree Model
model1 <- train(classe ~ ., data = traindata, method="rpart")
model1
```

```
## CART
##
## 307 samples
##  97 predictor
##   5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 307, 307, 307, 307, 307, 307, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.09333333  0.5072458  0.36513326
##   0.14666667  0.4246038  0.23857886
##   0.19555556  0.3042458  0.06903736
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.09333333.
```

```r
prediction1 <- predict(model1, testdata)
confusionMatrix(prediction1, testdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E
##          A 23 11 17 13  2
##          B  3  8  0  0  0
##          C  0  0  0  0  0
##          D  0  0  0  0  0
##          E  1  0  0  4 17
##
## Overall Statistics
##
##                Accuracy : 0.4848
##                  95% CI : (0.3832, 0.5875)
##     No Information Rate : 0.2727
##     P-Value [Acc > NIR] : 5.696e-06
##
##                   Kappa : 0.317
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8519  0.42105   0.0000   0.0000   0.8947
## Specificity            0.4028  0.96250   1.0000   1.0000   0.9375
## Pos Pred Value         0.3485  0.72727      NaN      NaN   0.7727
## Neg Pred Value         0.8788  0.87500   0.8283   0.8283   0.9740
## Prevalence             0.2727  0.19192   0.1717   0.1717   0.1919
## Detection Rate         0.2323  0.08081   0.0000   0.0000   0.1717
## Detection Prevalence   0.6667  0.11111   0.0000   0.0000   0.2222
## Balanced Accuracy      0.6273  0.69178   0.5000   0.5000   0.9161
```

```r
rpart.plot(model1$finalModel, roundint=FALSE)
```




```
## Very low accuracy

## Bagging Model

bagdata <- bagging(classe ~ ., data = traindata)
prediction2 <- predict(bagdata, testdata)
confusionMatrix(prediction2, testdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E
##          A 25  2  0  0  0
##          B  1 14  1  0  0
##          C  1  3 16  1  1
##          D  0  0  0 14  1
##          E  0  0  0  2 17
##
## Overall Statistics
##
##                Accuracy : 0.8687
##                  95% CI : (0.7859, 0.9282)
##     No Information Rate : 0.2727
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8345
```

```
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9259   0.7368   0.9412   0.8235   0.8947
## Specificity            0.9722   0.9750   0.9268   0.9878   0.9750
## Pos Pred Value         0.9259   0.8750   0.7273   0.9333   0.8947
## Neg Pred Value         0.9722   0.9398   0.9870   0.9643   0.9750
## Prevalence             0.2727   0.1919   0.1717   0.1717   0.1919
## Detection Rate         0.2525   0.1414   0.1616   0.1414   0.1717
## Detection Prevalence   0.2727   0.1616   0.2222   0.1515   0.1919
## Balanced Accuracy      0.9491   0.8559   0.9340   0.9057   0.9349
```

```r
## Quite Improvement but still not upto mark

## Random Forrest Model
model <- train(classe ~ ., data = traindata[, -1], method = "rf", ntree = 100)
prediction <- predict(model, testdata[, -1])
confusionMatrix(prediction, testdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C  D  E
##          A 24  4  0  1  0
##          B  1 11  1  1  0
##          C  1  3 16  0  1
##          D  1  0  0 14  1
##          E  0  1  0  1 17
##
## Overall Statistics
##
##                Accuracy : 0.8283
##                  95% CI : (0.7394, 0.8967)
##     No Information Rate : 0.2727
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7832
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8889   0.5789   0.9412   0.8235   0.8947
## Specificity            0.9306   0.9625   0.9390   0.9756   0.9750
## Pos Pred Value         0.8276   0.7857   0.7619   0.8750   0.8947
## Neg Pred Value         0.9571   0.9059   0.9872   0.9639   0.9750
## Prevalence             0.2727   0.1919   0.1717   0.1717   0.1919
## Detection Rate         0.2424   0.1111   0.1616   0.1414   0.1717
## Detection Prevalence   0.2929   0.1414   0.2121   0.1616   0.1919
## Balanced Accuracy      0.9097   0.7707   0.9401   0.8996   0.9349
```

```
## Unexpected result shows there is flow in analysis and need to consider more columns
```

Case 2 Eliminating Unnecessary column

```
removena <- sapply(newdata, function(x) mean(is.na(x))) > 0.95
newnadata <- newdata[,removena == FALSE]

ncol(newnadata)
```

```
## [1] 59
```

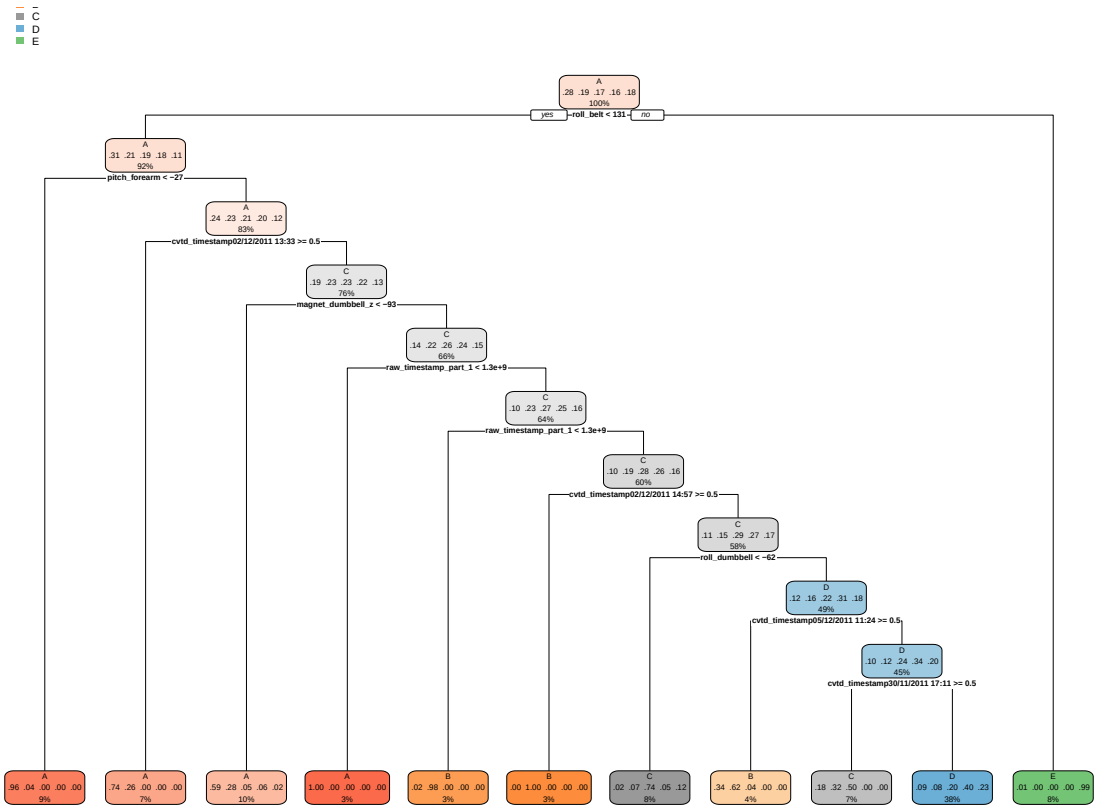
```
datana <- createDataPartition(newnadata$classe, p = 0.75, list = FALSE)
newtraindata <- newnadata[datana, ]
newtestdata <- newnadata[- datana,]

## Decision tree Model
## Eliminating first column as Name doesnot play any role in decisiding Outcome
model3 <- train(classe ~ ., data = newtraindata[,-1], method = "rpart")
prediction3 <- predict(model3, newtestdata[, -1])
confusionMatrix(prediction3, newtestdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1043  237   19   24   11
##          B   88  403   11    0    0
##          C   76  157  458   23   55
##          D  187  152  367  757  428
##          E    1    0    0    0  407
##
## Overall Statistics
##
##                Accuracy : 0.6256
##                  95% CI : (0.6119, 0.6392)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5302
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7477  0.42466  0.53567   0.9415  0.45172
## Specificity            0.9171  0.97497  0.92319   0.7234  0.99975
## Pos Pred Value         0.7819  0.80279  0.59558   0.4003  0.99755
## Neg Pred Value         0.9014  0.87597  0.90399   0.9844  0.89012
## Prevalence             0.2845  0.19352  0.17435   0.1639  0.18373
```

```
## Detection Rate         0.2127  0.08218  0.09339   0.1544  0.08299
## Detection Prevalence   0.2720  0.10237  0.15681   0.3856  0.08320
## Balanced Accuracy      0.8324  0.69981  0.72943   0.8325  0.72574
```

```
## Not Quite Accurate
rpart.plot(model3$finalModel, roundint=FALSE)
```




```
## Bagging Model

bagdata1 <- bagging(classe ~ ., data = newtraindata[, -1])
prediction4 <- predict(bagdata1, newtestdata[,-1])
confusionMatrix(prediction4, newtestdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  946    4    0    0
##          C    0    3  850    4    0
##          D    0    0    1  794    4
##          E    0    0    0    6  897
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9955
##                  95% CI : (0.9932, 0.9972)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9943
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9968   0.9942   0.9876   0.9956
## Specificity            1.0000   0.9990   0.9983   0.9988   0.9985
## Pos Pred Value         1.0000   0.9958   0.9918   0.9937   0.9934
## Neg Pred Value         1.0000   0.9992   0.9988   0.9976   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1929   0.1733   0.1619   0.1829
## Detection Prevalence   0.2845   0.1937   0.1748   0.1629   0.1841
## Balanced Accuracy      1.0000   0.9979   0.9962   0.9932   0.9970
```

```
## Random Forrest Model
model5 <- train(classe ~ ., data = newtraindata[, -1], method = "rf", ntree = 100)
prediction5 <- predict(model5, newtestdata[, -1])
confusionMatrix(prediction5, newtestdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  947    1    0    0
##          C    0    2  854    2    0
##          D    0    0    0  801    0
##          E    0    0    0    1  901
##
## Overall Statistics
##
##                Accuracy : 0.9988
##                  95% CI : (0.9973, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9985
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9979   0.9988   0.9963   1.0000
## Specificity            1.0000   0.9997   0.9990   1.0000   0.9998
## Pos Pred Value         1.0000   0.9989   0.9953   1.0000   0.9989
## Neg Pred Value         1.0000   0.9995   0.9998   0.9993   1.0000
```

```
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1931   0.1741   0.1633   0.1837
## Detection Prevalence   0.2845   0.1933   0.1750   0.1633   0.1839
## Balanced Accuracy      1.0000   0.9988   0.9989   0.9981   0.9999
```

Conclusion

- Based on result obtain Random forrest Model provides best possible predicting model for Case 2.
- As we can we from the result, the random forest algorithem far outperforms the decision tree in terms of accuracy. We are getting 99.99% in sample accuracy, while the decision tree gives us only nearly 50% in sample accuracy