Practical Machine Learning Week 4 Project

Udaya K Tejwani

December 7, 2019

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

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).

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 Data

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.3
```

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

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

```r
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.5.3
```

```r
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.5.3
```

```r
library(RColorBrewer)
library(RGtk2)
```

```
## Warning: package 'RGtk2' was built under R version 3.5.3
```

```r
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.5.3
```

```
## 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.
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.3
```

```
## 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
```

```r
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.5.3
```

```
## Loaded gbm 2.1.5
```

```r
training_data <- read.csv(url("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"))
testing_data <- read.csv(url("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"))

dim(training_data)
```

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

```r
dim(testing_data)
```

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

Data Cleanup

Removing variables, 95% of which have NA values

```
na_val_col <- sapply(training_data, function(x) mean(is.na(x))) > 0.95

training_data <- training_data[,na_val_col == FALSE]
testing_data <- testing_data[,na_val_col == FALSE]

dim(training_data)
```

```
## [1] 19622    93
```

```
dim(testing_data)
```

```
## [1] 20 93
```

Recoding missing values

```
######colSums(is.na(training_data))
######colSums(is.na(testing_data))
testing_data[is.na(testing_data)] <- mean(testing_data, na.rm = TRUE)
```

```
## Warning in mean.default(testing_data, na.rm = TRUE): argument is not
## numeric or logical: returning NA
```

Remvoing variables that have nearly zero variance

```
non_zero_var <- nearZeroVar(training_data)


training_data <- training_data[,-non_zero_var]
testing_data <- testing_data[,-non_zero_var]

dim(training_data)
```

```
## [1] 19622    59
```

```
dim(testing_data)
```

```
## [1] 20 59
```

```
######head(testing_data)
```

Removing first 6 variables since these will not contribute to the model

```
training_data <- training_data[,7:59]
testing_data <- testing_data[,7:59]

dim(training_data)
```

```
## [1] 19622    53
```

```
dim(testing_data)
```

```
## [1] 20 53
```

Data partitioning

The training dataset, training_data, is being partitioned into training and testing data in the ratio of 80%:20%

```r
inTrain <- createDataPartition(training_data$classe, p=0.8, list=FALSE)
training <- training_data[inTrain,]
testing <- training_data[-inTrain,]

dim(training)
```

```
## [1] 15699    53
```

```r
dim(testing)
```

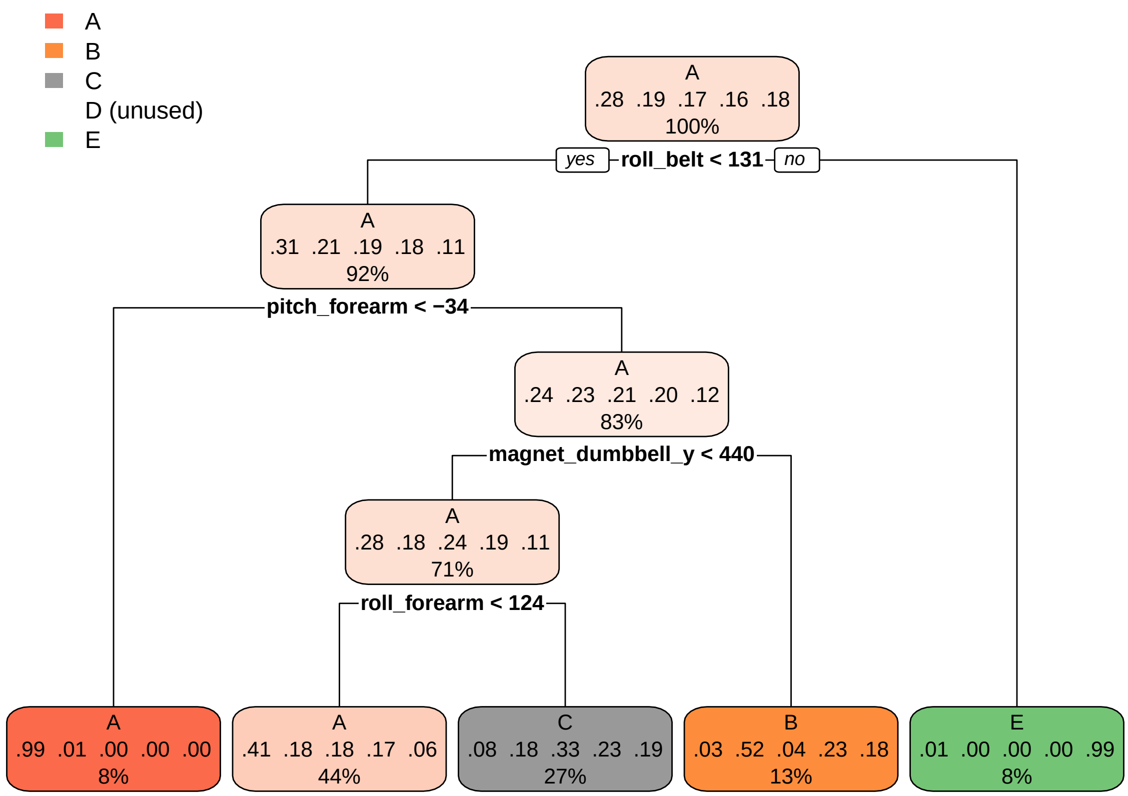
```
## [1] 3923   53
```

Decision Tree Model

```r
set.seed(12345)
mod_DT <- train(classe ~ ., data = training, method="rpart")
pred_DT <- predict(mod_DT, testing)
cmDT <- confusionMatrix(pred_DT, as.factor(testing$classe))
cmDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1012  335  315  293  112
##          B   16  241   20  110  113
##          C   82  183  349  240  187
##          D    0    0    0    0    0
##          E    6    0    0    0  309
##
## Overall Statistics
##
##                Accuracy : 0.4871
##                  95% CI : (0.4714, 0.5029)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3291
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9068  0.31752  0.51023   0.0000  0.42857
## Specificity            0.6242  0.91814  0.78635   1.0000  0.99813
## Pos Pred Value         0.4896  0.48200  0.33525      NaN  0.98095
## Neg Pred Value         0.9440  0.84867  0.88376   0.8361  0.88581
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2580  0.06143  0.08896   0.0000  0.07877
## Detection Prevalence   0.5269  0.12745  0.26536   0.0000  0.08030
## Balanced Accuracy      0.7655  0.61783  0.64829   0.5000  0.71335
```

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



The decision tree model has a very low accuracy of 51%. The model accuracy is not satisfactory. This model needs to be verified with cross validation from other models.

Random Forest Model

```
set.seed(23456)
mod_RF <- train(classe ~. , data=training, method= "rf", ntree=100)
pred_RF <- predict(mod_RF, testing)
cmRFM <- confusionMatrix(pred_RF, testing$classe)
cmRFM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    9    0    0    0
##          B    0  750    2    0    0
##          C    0    0  680    4    0
##          D    0    0    2  639    3
##          E    0    0    0    0  718
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9921, 0.9969)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.9935
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9881   0.9942   0.9938   0.9958
## Specificity            0.9968   0.9994   0.9988   0.9985   1.0000
## Pos Pred Value         0.9920   0.9973   0.9942   0.9922   1.0000
## Neg Pred Value         1.0000   0.9972   0.9988   0.9988   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1912   0.1733   0.1629   0.1830
## Detection Prevalence   0.2868   0.1917   0.1744   0.1642   0.1830
## Balanced Accuracy      0.9984   0.9938   0.9965   0.9961   0.9979
```

The random forest model has accuracy of 99%. Although this is an impressive model accuracy, it could also imply over-fitting. Next, we will cross validate the above models with Gradient Boosting model.

Gradient Boosting Model

```r
set.seed(34567)
mod_gbm <- train(classe~., data=training, method="gbm", verbose= FALSE)
mod_gbm$finalmodel
```

```
## NULL
```

```r
pred_gbm <- predict(mod_gbm, testing)
cmGBM <- confusionMatrix(pred_gbm, testing$classe)
cmGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1101   30    0    1    0
##          B   11  712   23    5   13
##          C    2   14  656   25    5
##          D    2    2    5  607    5
##          E    0    1    0    5  698
##
## Overall Statistics
##
##                Accuracy : 0.962
##                  95% CI : (0.9556, 0.9678)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9519
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9866   0.9381   0.9591   0.9440   0.9681
## Specificity            0.9890   0.9836   0.9858   0.9957   0.9981
## Pos Pred Value         0.9726   0.9319   0.9345   0.9775   0.9915
## Neg Pred Value         0.9946   0.9851   0.9913   0.9891   0.9929
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2807   0.1815   0.1672   0.1547   0.1779
## Detection Prevalence   0.2886   0.1947   0.1789   0.1583   0.1795
## Balanced Accuracy      0.9878   0.9608   0.9724   0.9699   0.9831
```

The gradient boosting model has accuracy of 96%.

```
cmDT$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   4.871272e-01   3.290767e-01   4.713732e-01   5.029003e-01   2.844762e-01
## AccuracyPValue  McnemarPValue
##  7.974835e-158            NaN
```

```
cmRFM$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9949019      0.9935498      0.9921373      0.9968832      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
cmGBM$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9620189      0.9519291      0.9555558      0.9677815      0.2844762
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Model of Choice

The Random Forest Model is the model of choice since it has the highest accuracy. Grdient Boosting Model is also good but it is a close second model.

Test Prediction

```
Test_RF_prediction <- predict(mod_RF, testing_data )
Test_RF_prediction
```

```
##  [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
```