Coursera - Practical Machine Learning - Assignment

Kushan Sarathchandra

30/05/2021

Prediction Assignment Writeup

1. Overview & Background

This document summarise the work done for the Prediction Assignment Writeup project for the Coursera Practical Machine Learning course. It's created using the functionalities of the knitr package in RStudio using the actual analysis code. 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).

2. Exploratory Analysis

- Environment preparation

This section will load necessary package for this analysis.

```
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)
```

- Data Loading and Cleaning

```
# Download the training & testing data and load the dataset
download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile = "
dt_training <- read.csv("./pml-training.csv", na.strings=c("NA","#DIV/0!",""))
download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile = "
dt_testing <- read.csv("./pml-testing.csv", na.strings=c("NA","#DIV/0!",""))
```

- Cleaning the Data

In this section, we will remove all columns that contains NA and remove features that are not in the testing dataset. The features containing NA are the variance, mean and standard devition (SD) within each window for each feature. Since the testing dataset has no time-dependence, these values are useless and can be disregarded. We will also remove the first 7 features since they are related to the time-series or are not numeric.

```
features <- names(dt_testing[,colSums(is.na(dt_testing)) == 0])[8:59]

# Only use features used in testing cases.
dt_training <- dt_training[,c(features,"classe")]
dt_testing <- dt_testing[,c(features,"problem_id")]

dim(dt_training); dim(dt_testing);
```

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

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

- Partitioning the Dataset

Following the recommendation in the course Practical Machine Learning, we will split our data into a training data set (60% of the total cases) and a testing data set (40% of the total cases; the latter should not be confused with the data in the pml-testing.csv file). This will allow us to estimate the out of sample error of our predictor.

```
set.seed(1234)

inTrain <- createDataPartition(dt_training$classe, p=0.6, list=FALSE)
training <- dt_training[inTrain,]
testing <- dt_training[-inTrain,]

dim(training); dim(testing);
```

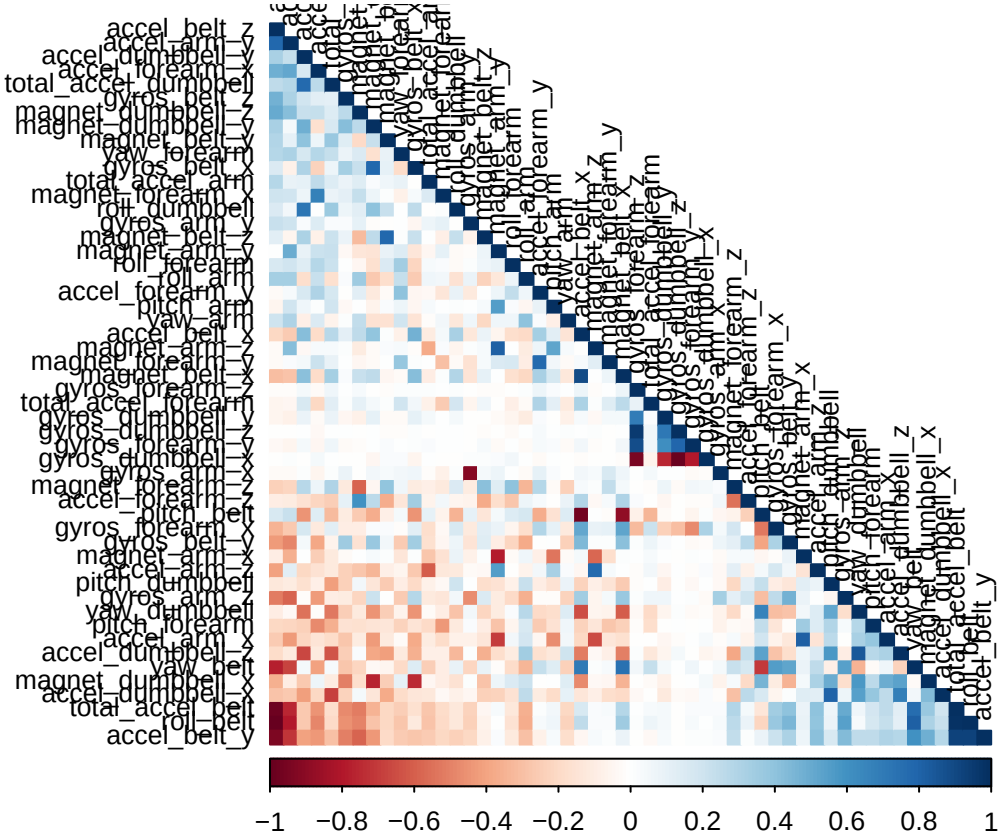
```
## [1] 11776    53
```

```
## [1] 7846   53
```

- Data Correlation Analysis

A correlation among variables is analysed before proceeding to the modelling procedures.

```
library(corrplot)
corMatrix <- cor(training[, -53])
corrplot(corMatrix, order = "FPC", method = "color", type = "lower", tl.cex = 0.8, tl.col = rgb(0, 0, 0
```

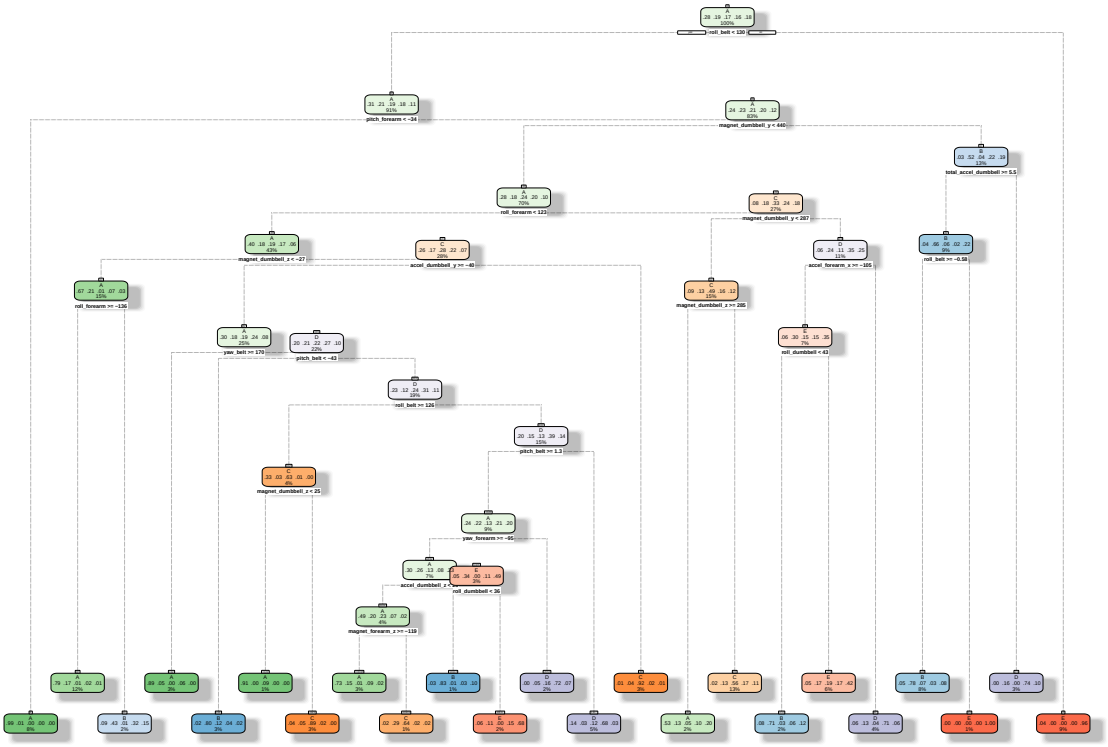


3. Prediction Model Building

- Building the Decision Tree Model

Using Decision Tree, we should not expect the accuracy to be high. In fact, anything around 80% would be acceptable.

```
modFitDT <- rpart(classe ~ ., data = training, method="class")
fancyRpartPlot(modFitDT)
```



- Predicting with the Decision Tree Model

```
set.seed(1234)

prediction <- predict(modFitDT, testing, type = "class")
confusionMatrix(prediction, as.factor(testing$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1930  231   47   81   53
##          B   82  866   67  103  121
##          C   59  189 1062  173  134
##          D   94  118   85  821   70
##          E   67  114  107  108 1064
##
## Overall Statistics
##
##                Accuracy : 0.732
##                  95% CI : (0.722, 0.7417)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.6605
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8647   0.5705   0.7763   0.6384   0.7379
## Specificity            0.9266   0.9411   0.9143   0.9441   0.9382
## Pos Pred Value         0.8241   0.6990   0.6568   0.6911   0.7288
## Neg Pred Value         0.9451   0.9013   0.9509   0.9302   0.9408
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2460   0.1104   0.1354   0.1046   0.1356
## Detection Prevalence   0.2985   0.1579   0.2061   0.1514   0.1861
## Balanced Accuracy      0.8957   0.7558   0.8453   0.7912   0.8380
```

- Building the Random Forest Model

Using random forest, the out of sample error should be small. The error will be estimated using the 40% testing sample. We should expect an error estimate of $< 3\%$.

```
set.seed(1234)
training$classe <- as.factor(training$classe)
modFitRF <- randomForest(classe ~ ., data = training, ntree = 1000)
```

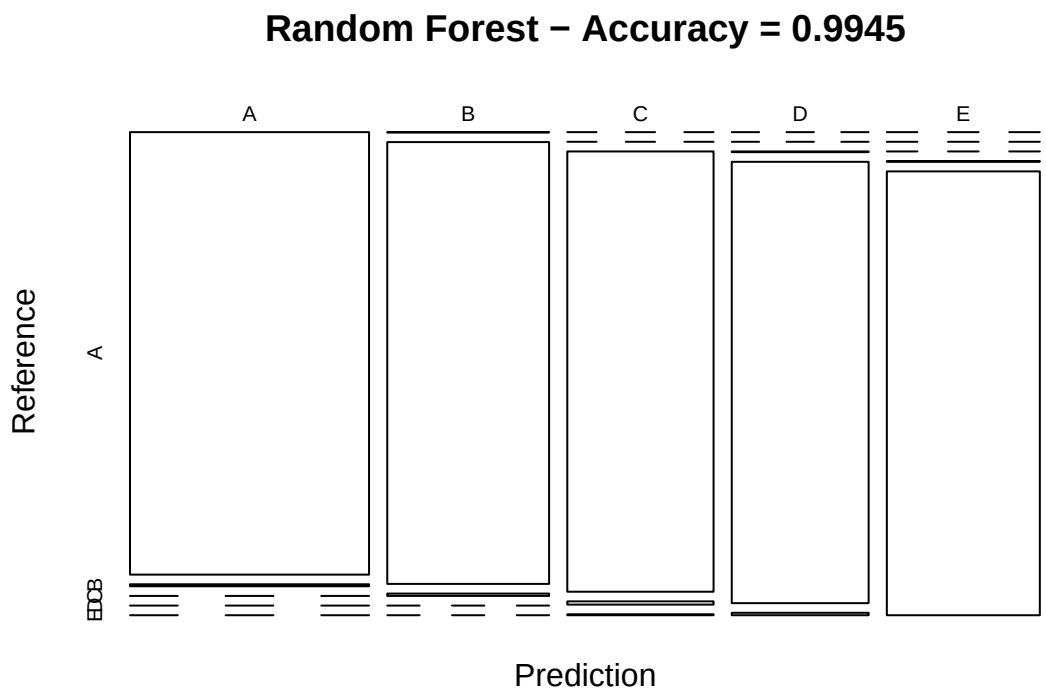
- Predicting with the Random Forest Model

```
prediction <- predict(modFitRF, testing, type = "class")
confusionMatrix <- confusionMatrix(prediction, as.factor(testing$classe))
confusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231   10    0    0    0
##          B    1 1508    8    0    0
##          C    0    0 1358   10    3
##          D    0    0    2 1274    7
##          E    0    0    0    2 1432
##
## Overall Statistics
##
##                Accuracy : 0.9945
##                  95% CI : (0.9926, 0.996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9931
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9934   0.9927   0.9907   0.9931
## Specificity            0.9982   0.9986   0.9980   0.9986   0.9997
## Pos Pred Value         0.9955   0.9941   0.9905   0.9930   0.9986
## Neg Pred Value         0.9998   0.9984   0.9985   0.9982   0.9984
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1922   0.1731   0.1624   0.1825
## Detection Prevalence   0.2856   0.1933   0.1747   0.1635   0.1828
## Balanced Accuracy      0.9989   0.9960   0.9953   0.9946   0.9964
```

```r
#Plotting matrix results
plot(confusionMatrix$table, col = confusionMatrix$byClass,
        main = paste("Random Forest - Accuracy =", round(confusionMatrix$overall['Accuracy'], 4)))
```



4. Predicting on the Testing Data

- Decision Tree Prediction

```r
predictionDT <- predict(modFitDT, dt_testing, type = "class")
predictionDT
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  C  A  E  D  A  C  D  A  A  A  C  E  A  A  E  E  A  B  B  B
## Levels: A B C D E
```

- Random Forest Prediction

```
predictionRF <- predict(modFitRF, dt_testing, type = "class")
predictionRF
```

```
##  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
```

5. Submission

As can be seen from the confusion matrix the Random Forest model is very accurate, about 99%. Because of that we could expect nearly all of the submitted test cases to be correct. It turned out they were all correct.