Practical Machine Learning Assignment

Zahiruddin Zahidanishah

06/20/2020

Introduction

Background

Using smart 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.

This report will used data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

More information is available from the following website: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available on the following link:

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

The test data are available on the following link:

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

The data for this project come from this source: http://web.archive.org/web/20161224072740/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.

Human Activity Recognition

Human Activity Recognition (HAR) has emerged as a key research area in the last years and is gaining increasing attention by the pervasive computing research community, especially for the development of context-aware systems. There are many potential applications for HAR, such as elderly monitoring, life log systems for monitoring energy expenditure and for supporting weight-loss programs, and digital assistants for weight lifting exercises.

Based on the authors website, the description of the data set contents are as detail below:-

1. Class A - Throwing the elbows to the front
2. Class B - Lifting the dumbbell only halfway
3. Class C - Lowering the dumbbell only halfway
4. Class D - Throwing the hips to the front

Setting up

The following r packages shall be loaded for the purposes of this report:-

1. knitr
2. caret
3. rpart
4. rpart.plot
5. rattle
6. randomForest
7. corrplot

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

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

```
## 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.
```

```
## 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
```

```
## corrplot 0.84 loaded
```

As given in the assignment, the following data set is loaded in r for the purposes of this report including the training and test set:-

```
UrlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
UrlTest  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(UrlTrain))
testing  <- read.csv(url(UrlTest))
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
TrainSet <- training[inTrain, ]
TestSet  <- training[-inTrain, ]
```

Based on the training and test sets, shows that both consists of 160 variable (column).

```
dim(TrainSet)
```

```
## [1] 13737   160
```

```
dim(TestSet)
```

```
## [1] 5885  160
```

Data cleaning process

The data set consist of unnecessary items such as zero values and NA. This items need to be removed before continue with the analysis works.

```
NZV <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -NZV]
TestSet  <- TestSet[, -NZV]
AllNA    <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet <- TrainSet[, AllNA==FALSE]
TestSet  <- TestSet[, AllNA==FALSE]
TrainSet <- TrainSet[, -(1:5)]
TestSet  <- TestSet[, -(1:5)]
dim(TrainSet)
```

```
## [1] 13737    54
```

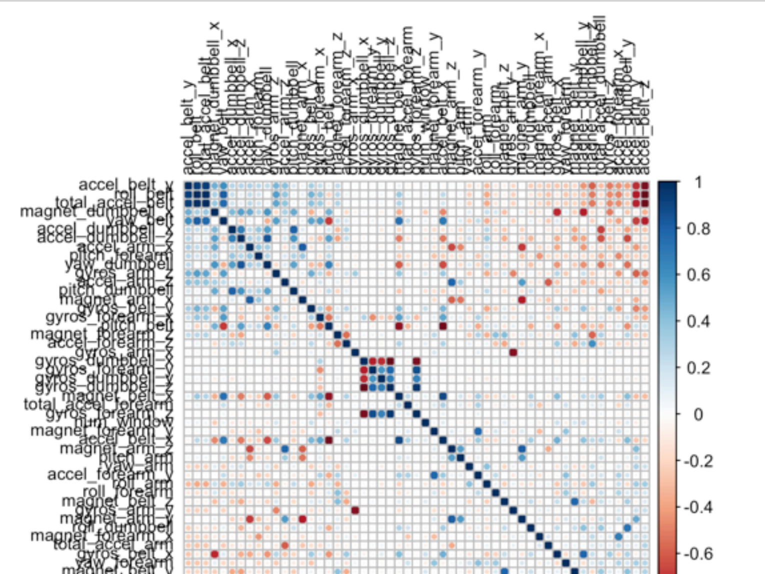
```
dim(TestSet)
```

```
## [1] 5885   54
```

Cross Validation

To carry out the cross validation on the data set variable, a corrplot package is used in order to get a graphical dispaly of the correlation matrix, confidence interval of the data set.

```
corMatrix <- cor(TrainSet[, -54])
corrplot(corMatrix, order = "FPC", method = "circle", type = "full", tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



From the above plot shows that the highly correlated variables as shown in the dark colors.

Model Prediction

In order to identify the pattern between sample in training set, three methods are selected to carry out to find out the suitable modeling methods.

1.0 Random Forest Method

Based on the *Random Forest Method* modeling, shows that the error rate is **0.23%** and the accuracy is **99.9%**. The details are shows below.

```
set.seed(12345)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=TrainSet, method="rf", trControl=controlRF)
modFitRandForest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.23%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    2    0    0    0 0.0005120328
## B    6 2647    4    1    0 0.0041384500
## C    0    5 2391    0    0 0.0020868114
## D    0    0    9 2243    0 0.0039964476
## E    0    0    0    5 2520 0.0019801980
```

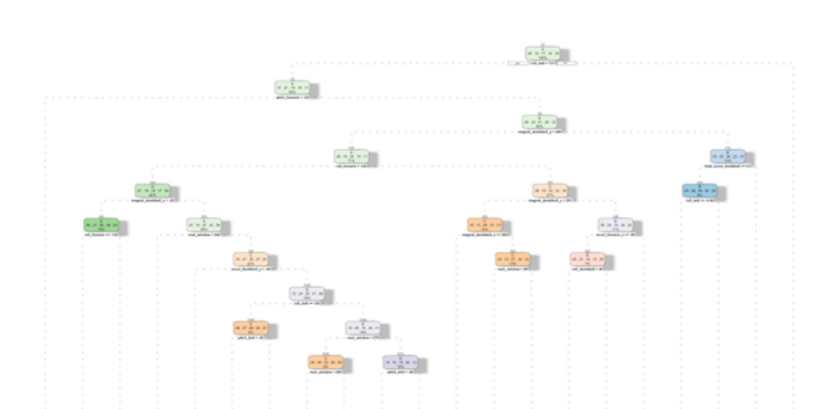
```
predictRandForest <- predict(modFitRandForest, newdata=TestSet)
confMatRandForest <- confusionMatrix(predictRandForest, as.factor(TestSet$classe))
confMatRandForest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    2    0    0
##          C    0    0 1024    2    0
##          D    0    0    0  962    1
##          E    0    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.9978, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9987
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9981   0.9979   0.9991
## Specificity            0.9998   0.9996   0.9996   0.9998   1.0000
## Pos Pred Value         0.9994   0.9982   0.9981   0.9990   1.0000
## Neg Pred Value         1.0000   0.9998   0.9996   0.9996   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1740   0.1635   0.1837
## Detection Prevalence   0.2846   0.1937   0.1743   0.1636   0.1837
## Balanced Accuracy      0.9999   0.9994   0.9988   0.9989   0.9995
```

2.0 Predicting with decision trees

Based on the *Decision Trees Method* modeling, shows that the accuracy is **73.4%**. The details are shows below.

```
set.seed(12345)
modFitDecTree <- rpart(classe ~ ., data=TrainSet, method="class")
fancyRpartPlot(modFitDecTree)
```



Rattle 2020-Jun-21 10:09:36 zzahir1978

```
predictDecTree <- predict(modFitDecTree, newdata=TestSet, type="class")
confMatDecTree <- confusionMatrix(predictDecTree, as.factor(TestSet$classe))
confMatDecTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1502  201   59   66   74
##          B   58  660   37   64  114
##          C    4   66  815  129   72
##          D   90  148   54  648  126
##          E   20   64   61   57  696
##
## Overall Statistics
##
##                Accuracy : 0.7342
##                  95% CI : (0.7228, 0.7455)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6625
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8973   0.5795   0.7943   0.6722   0.6433
## Specificity            0.9050   0.9425   0.9442   0.9151   0.9579
## Pos Pred Value         0.7897   0.7074   0.7505   0.6079   0.7751
## Neg Pred Value         0.9568   0.9033   0.9560   0.9344   0.9226
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2552   0.1121   0.1385   0.1101   0.1183
## Detection Prevalence   0.3232   0.1585   0.1845   0.1811   0.1526
## Balanced Accuracy      0.9011   0.7610   0.8693   0.7936   0.8006
```

3.0 Generalized Boosted Model

Based on the *Generalizied Boosted Method* modeling, shows that the accuracy is **98.7%**. The details are shows below.

```
set.seed(12345)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitGBM  <- train(classe ~ ., data=TrainSet, method = "gbm", trControl = controlGBM, verbose = FALSE)
modFitGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```
predictGBM <- predict(modFitGBM, newdata=TestSet)
confMatGBM <- confusionMatrix(predictGBM, as.factor(TestSet$classe))
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1668   12    0    1    0
##          B    6 1115   12    1    3
##          C    0   12 1012   21    0
##          D    0    0    2  941    6
##          E    0    0    0    0 1073
##
## Overall Statistics
##
##                Accuracy : 0.9871
##                  95% CI : (0.9839, 0.9898)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9837
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9789   0.9864   0.9761   0.9917
## Specificity            0.9969   0.9954   0.9932   0.9984   1.0000
## Pos Pred Value         0.9923   0.9807   0.9684   0.9916   1.0000
## Neg Pred Value         0.9986   0.9949   0.9971   0.9953   0.9981
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2834   0.1895   0.1720   0.1599   0.1823
## Detection Prevalence   0.2856   0.1932   0.1776   0.1613   0.1823
## Balanced Accuracy      0.9967   0.9871   0.9898   0.9873   0.9958
```

Model Prediction

Based on the prediction modeling shows that **Random Forest Method** shows the highest accuracy equal to **99.9%** while the **Decision Tree Method** has the lowest accuracy equal to **73.4%**.

Therefore, based on the Random Forest Method prediction model, the predict model for the 20 cases are summarized below:-

1. Class A - 7 cases
2. Class B - 8 cases
3. Class C - 1 case
4. Class D - 1 case
5. Class E - 3 cases

```
predictTEST <- predict(modFitRandForest, newdata=testing)
predictTEST
```

```
##  [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
```