Practical Machine Learning : Prediction Assignment Writeup

Johnson Kamireddy

May 29, 2019

OVERVIEW:

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, we will 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.

The goal of the project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set.

This report describe how the model was built, how cross validation was used, what expected out of sample error was, and why/ how the specific choices were made. We used prediction model to predict 20 different test cases.

Loading and Cleaning data:

Loading the required R libraries to perform the analysis.

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

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

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
## 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.
```

```
## 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
```

The datasets are downloaded into the system so the next step involves loading the datasets into Rstudio. The dataset was partioned into two. 70% of the data is considered to be training set to train the algorithms and remaining 30% as test set to test the trained models.

```
training_data <- read.csv(file = "C:/Users/johns/Documents/pml-training.csv", header = TRUE, sep = ",")
test_data <- read.csv(file = "C:/Users/johns/Documents/pml-testing.csv", header = TRUE, sep = ",")

# Creating the data partitions:

train_partition <- createDataPartition(training_data$classe, p = 0.7, list = FALSE)
traindata <- training_data[train_partition, ]
testdata <- training_data[-train_partition, ]

# Dimensions of the training datatset:
dim(traindata)
```

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

```
# Dimensions of the test datatset:
dim(testdata)
```

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

Both the datasets have 160 variables and there is a possibility that there are plenty NA values present in the dataset.In addition to that the near zero variance variables (NZV) are also removed and the identification variables as well.

```
zerovariables <- nearZeroVar(traindata)
traindata <- traindata[, -zerovariables]
testdata <- testdata[, -zerovariables]

dim(traindata)
```

```
## [1] 13737   102
```

```
dim(testdata)
```

```
## [1] 5885  102
```

Removing the variables that have most NA values

```
navariables <- sapply(traindata, function(x) mean(is.na(x))) > 0.95
traindata <- traindata[, navariables == FALSE]
testdata <- testdata[, navariables == FALSE]

dim(traindata)
```

```
## [1] 13737    59
```

```
dim(testdata)
```

```
## [1] 5885   59
```

Removing the identification variables which are columns 1 to 5

```
traindata <- traindata[, -(1:5)]
testdata <- testdata[, -(1:5)]

dim(traindata)
```

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

```
dim(testdata)
```

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

After the data wrangling we have 54 variables available with 13737 and 5885 records in traindata and testdata datasets.

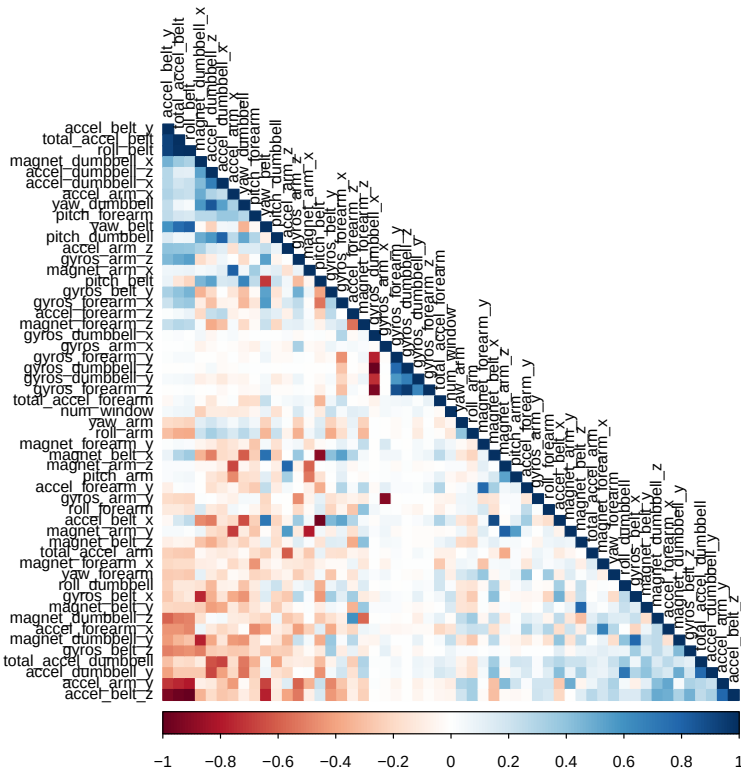
Exploratory Data Analysis:

Correlation Analysis

The correlation analysis is performed on the train dataset before modeling it with machine learning techniques.

```
cor_matrix <- cor(traindata[, -54])

corrplot(cor_matrix, order = "FPC", method = "color", type = "lower", tl.cex = 0.8, tl.col = rgb(0, 0, 
```


accel_belt_y
total_accel_belt
roll_belt
magnet_dumbbell_x
accel_dumbbell_z
accel_dumbbell_x
accel_arm_x
yaw_dumbbell
pitch_forearm
yaw_belt
pitch_dumbbell
accel_arm_z
gyros_arm_z
magnet_arm_x
pitch_belt
gyros_belt_y
gyros_forearm_x
accel_forearm_z
magnet_forearm_z
gyros_dumbbell_x
gyros_arm_x
gyros_forearm_y
gyros_dumbbell_z
gyros_dumbbell_y
gyros_forearm_z
total_accel_forearm
num_window
yaw_arm
roll_arm
magnet_forearm_y
magnet_belt_x
magnet_arm_z
pitch_arm
accel_forearm_y
gyros_arm_y
roll_forearm
accel_belt_x
magnet_arm_y
magnet_belt_z
total_accel_arm
magnet_forearm_x
yaw_forearm
roll_dumbbell
gyros_belt_x
magnet_belt_y
magnet_dumbbell_z
accel_forearm_x
magnet_dumbbell_y
accel_dumbbell_y
accel_arm_y
accel_belt_z
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1

Prediction Modeling:

Three methods are applied to the train dataset to model the regressions and the accurate model with the higher accuracy when applied to test data will be employed for further results or predictions. The methods employed are Random forests, Decision tree and generalized boosted model.

Random Forests:

```
set.seed(12345)
RFcontrol <- trainControl(method = "cv", number = 3, verboseIter = FALSE)
RFmodfit <- train(classe ~ ., data = traindata, method="rf", trControl=RFcontrol)

RFmodfit$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: 0.2%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    1    0 0.0005120328
## B    7 2646    4    1    0 0.0045146727
## C    0    5 2391    0    0 0.0020868114
## D    0    0    6 2245    1 0.0031083481
## E    0    0    0    2 2523 0.0007920792
```

Using the trained model on the test dataset to check the accuracy of the model.

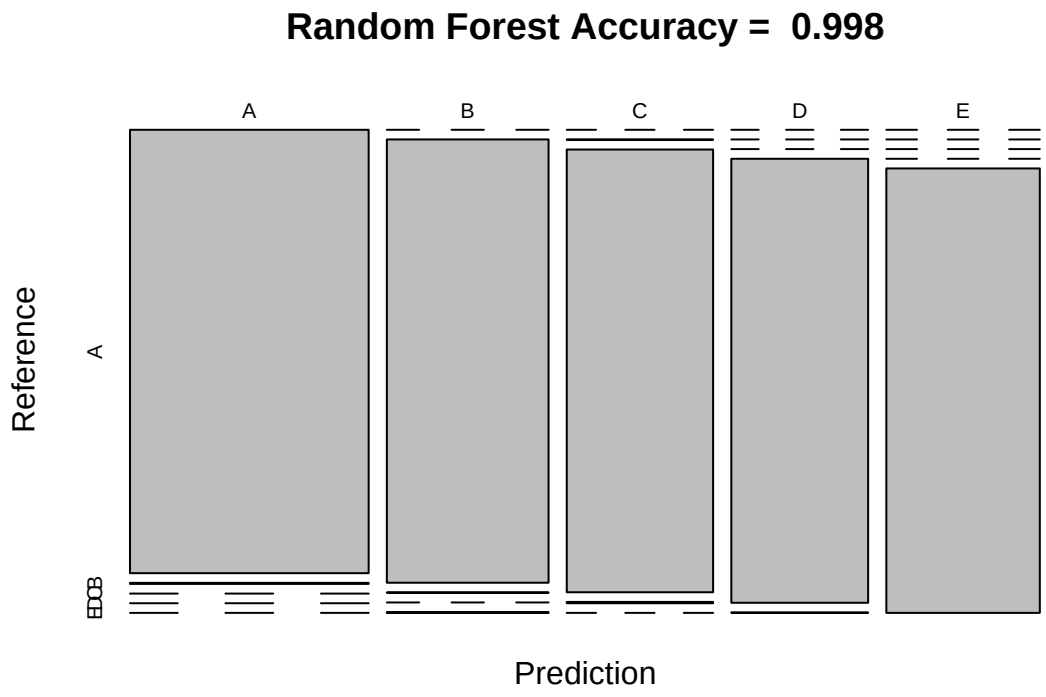
```
predict_RF <- predict(RFmodfit, newdata = testdata)
confusion_RF <- confusionMatrix(predict_RF, testdata$classe)
confusion_RF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    4    0    0    0
##          B    0 1134    1    0    2
##          C    0    1 1025    3    0
##          D    0    0    0  961    1
##          E    0    0    0    0 1079
##
## Overall Statistics
##
##                Accuracy : 0.998
##                  95% CI : (0.9964, 0.9989)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9974
##
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9956   0.9990   0.9969   0.9972
## Specificity            0.9991   0.9994   0.9992   0.9998   1.0000
## Pos Pred Value         0.9976   0.9974   0.9961   0.9990   1.0000
## Neg Pred Value         1.0000   0.9989   0.9998   0.9994   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1927   0.1742   0.1633   0.1833
## Detection Prevalence   0.2851   0.1932   0.1749   0.1635   0.1833
## Balanced Accuracy      0.9995   0.9975   0.9991   0.9983   0.9986
```

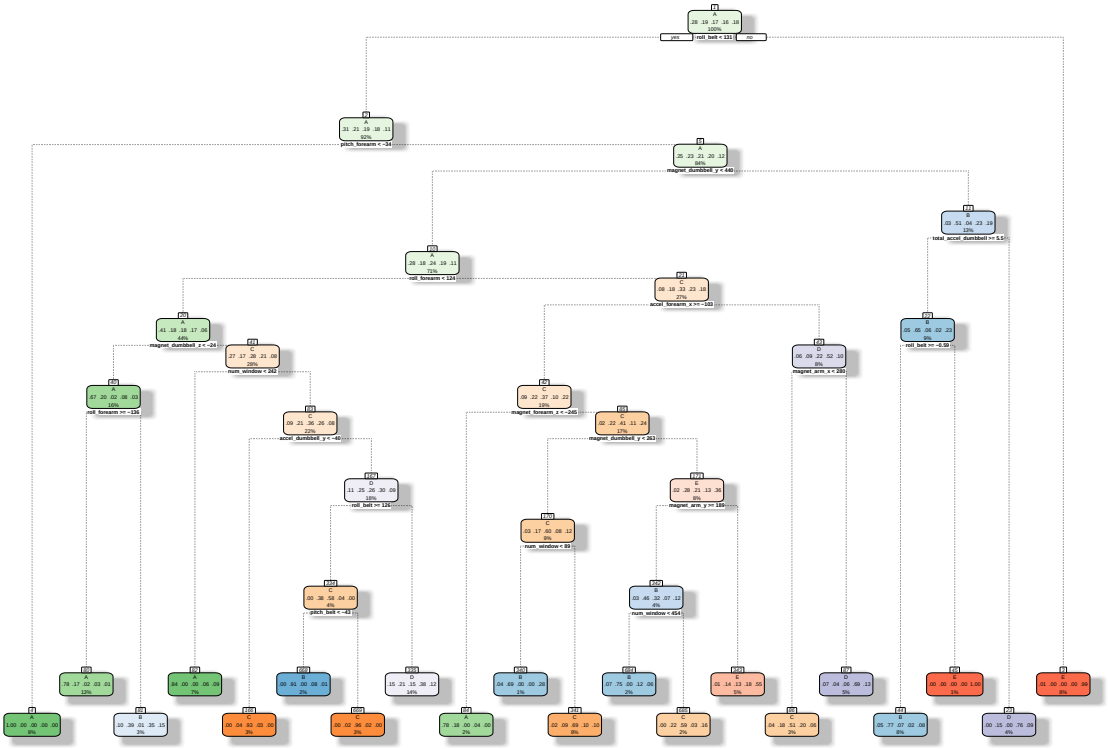
Plotting the results of the consufusion matrix to explain it more visually.

```
plot(confusion_RF$table, col = confusion_RF$byclass, main = paste("Random Forest Accuracy = ", round(co
```



Decision Trees:

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



Using the trained model on the test dataset to check the accuracy of the model.

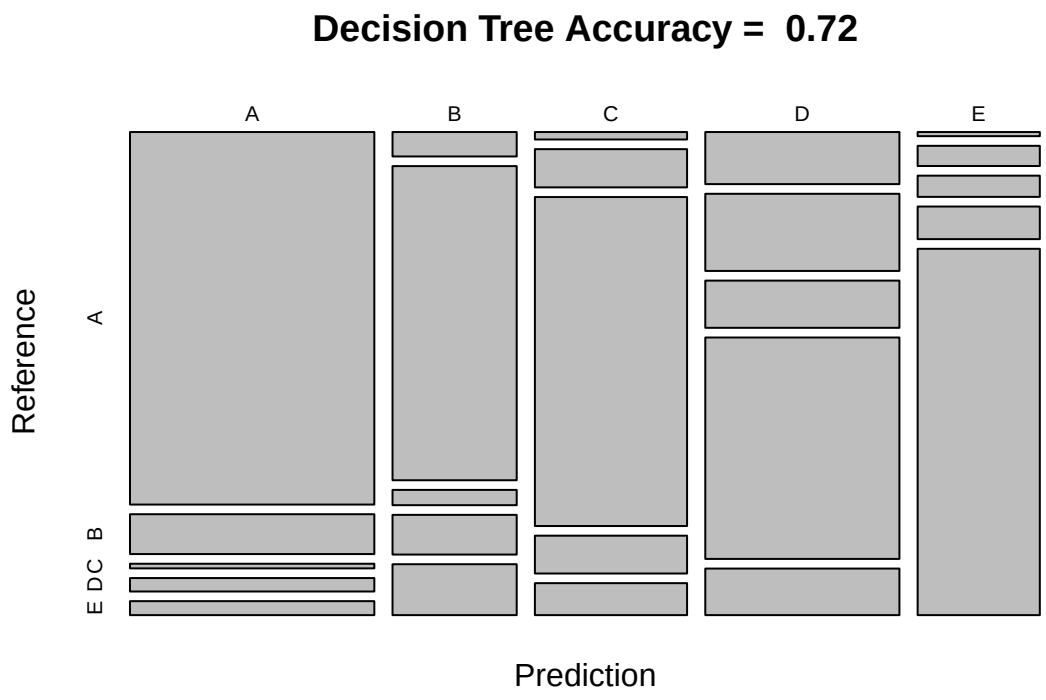
```
predict_DT <- predict(DTmodfit, newdata = testdata, type = "class")
confusion_DT <- confusionMatrix(predict_DT, testdata$classe)
confusion_DT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1440  154   18   52   54
##          B   48  617   30   78  100
##          C   18   92  792   91   77
##          D  160  237  145  680  143
##          E    8   39   41   63  708
##
## Overall Statistics
##
##                Accuracy : 0.72
##                  95% CI : (0.7083, 0.7314)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6463
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8602   0.5417   0.7719   0.7054   0.6543
## Specificity            0.9340   0.9461   0.9428   0.8608   0.9686
## Pos Pred Value         0.8382   0.7068   0.7402   0.4982   0.8242
## Neg Pred Value         0.9438   0.8958   0.9514   0.9372   0.9256
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2447   0.1048   0.1346   0.1155   0.1203
## Detection Prevalence   0.2919   0.1483   0.1818   0.2319   0.1460
## Balanced Accuracy      0.8971   0.7439   0.8574   0.7831   0.8115
```

Plotting the results of the consufusion matrix to explain it more visually.

```
plot(confusion_DT$table, col = confusion_DT$byclass, main = paste("Decision Tree Accuracy = ", round(co
```



Generalized Boosted Model:

```
set.seed(12345)
GBMcontrol <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
GBMmodfit <- train(classe ~ ., data = traindata, method = "gbm", trControl = GBMcontrol, verbose = FALS

GBMmodfit$finalModel
```

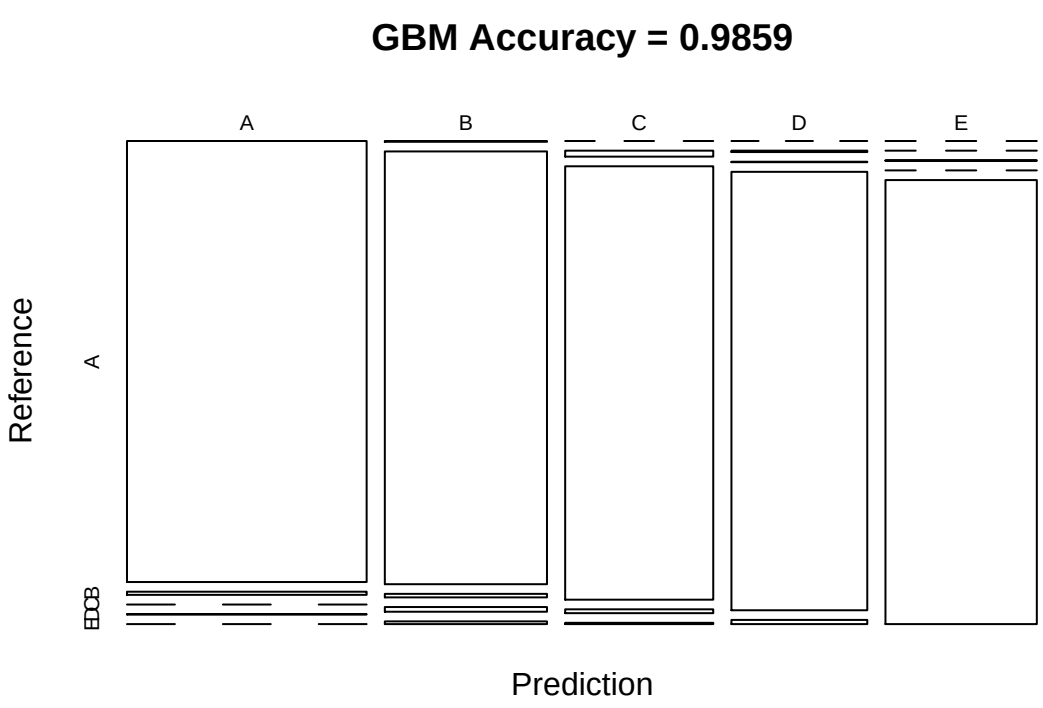
```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```
predict_GBM <- predict(GBMmodfit, newdata = testdata)
confusion_GBM <- confusionMatrix(predict_GBM, testdata$classe)
confusion_GBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672   12    0    1    0
##          B    2 1110    9   12    7
##          C    0   14 1015    9    3
##          D    0    3    1  942    9
##          E    0    0    1    0 1063
##
## Overall Statistics
##
##                Accuracy : 0.9859
##                  95% CI : (0.9825, 0.9888)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9822
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9745   0.9893   0.9772   0.9824
## Specificity            0.9969   0.9937   0.9946   0.9974   0.9998
## Pos Pred Value         0.9923   0.9737   0.9750   0.9864   0.9991
## Neg Pred Value         0.9995   0.9939   0.9977   0.9955   0.9961
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1886   0.1725   0.1601   0.1806
## Detection Prevalence   0.2863   0.1937   0.1769   0.1623   0.1808
## Balanced Accuracy      0.9979   0.9841   0.9920   0.9873   0.9911
```

Plotting the results of the consufusion matrix to explain it more visually.

```
plot(confusion_GBM$table, col = confusion_GBM$byClass, main = paste("GBM Accuracy =", round(confusion_GB
```



Employing the efficient model to the test data:

The accuracy ratings of the 3 models used are as follows: I. Random Forest - 0.99 II. Decision Tree - 0.73 III. Generalized Boosted Model - 0.98

As the accuracy of Random forest model is more efficient and accurate we will be employing this on the test data set.

```
predict_test <- predict(RFmodfit, newdata = test_data)
predict_test
```

```
##  [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
```