

Prediction Assignment Writeup

Akinyemi

2/29/2020

1. Overview

This document is the final report of the Peer Assessment project from Coursera's course Practical Machine Learning, as part of the Specialization in Data Science. It was built in RStudio, using its knitr functions, meant to be published in html format. This analysis meant to be the basis for the course quiz and a prediction assignment writeup. The main goal of the project is to predict the manner in which 6 participants performed some exercise as described below. This is the "classe" variable in the training set. The machine learning algorithm described here is applied to the 20 test cases available in the test data and the predictions are submitted in appropriate format to the Course Project Prediction Quiz for automated grading.

2. Data Loading and Exploratory Analysis

a. Environment Setup

```
setwd("~/Coursera Coureses/Machine Learning")
library(knitr)
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 5.3.0 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(corrplot)
```

```
## corrplot 0.84 loaded
```

```
set.seed(301)
```

b. Data Loading and Cleaning

```
testing <- read.csv("pml-testing.csv")
training <- read.csv("pml-training.csv")

# create a partition using caret with the training dataset on 70,30 ratio
inTrain <- createDataPartition(training$classe, p=0.7, list=FALSE)

TrainSet <- training[inTrain, ]
TestSet  <- training[-inTrain, ]

dim(TrainSet)
```

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

```
dim(TestSet)
```

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

Both created datasets have 160 variables. Let's clean NA, The Near Zero variance (NZV) variables and the ID variables as well.

```
# remove variables with Nearly Zero Variance
NZV <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -NZV]
TestSet  <- TestSet[, -NZV]
dim(TestSet)
```

```
## [1] 5885  105
```

```
dim(TrainSet)
```

```
## [1] 13737  105
```

```
# remove variables that are mostly NA
AllNA <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet <- TrainSet[, AllNA==FALSE]
TestSet <- TestSet[, AllNA==FALSE]
dim(TestSet)
```

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

```
dim(TrainSet)
```

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

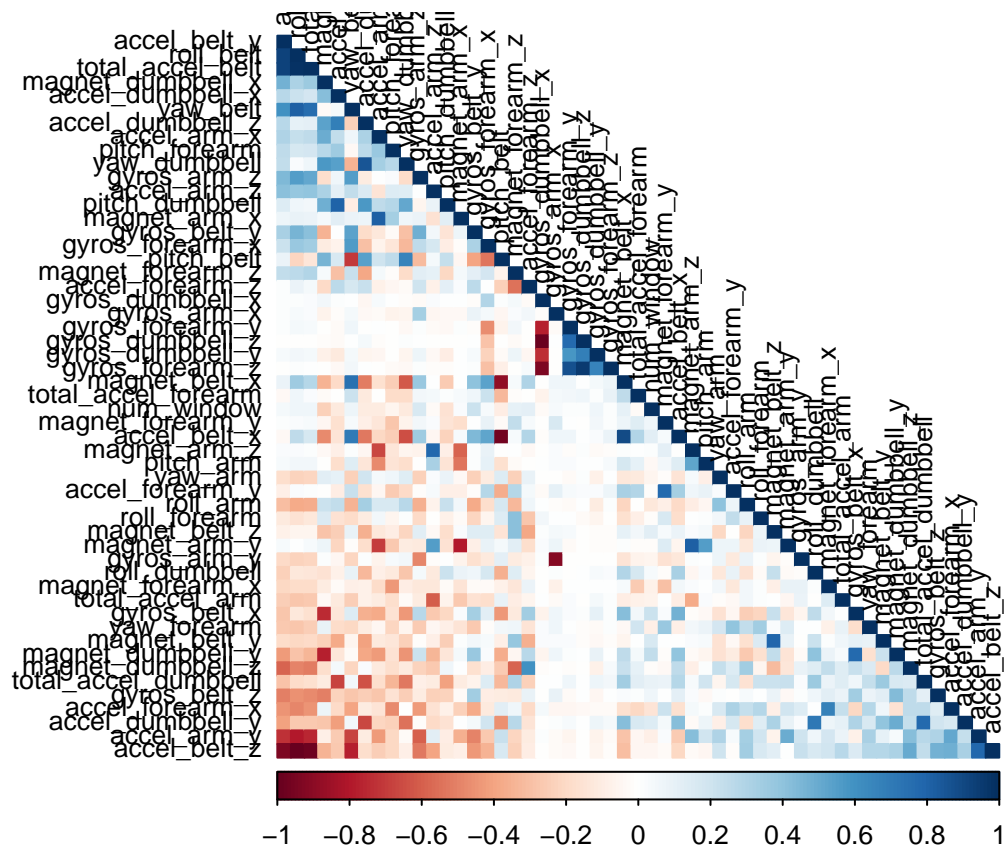
```
# remove identification only variables (columns 1 to 5)
TrainSet <- TrainSet[, -(1:5)]
TestSet <- TestSet[, -(1:5)]
dim(TrainSet)
```

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

correlation Analysis

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

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



The highly correlated variables are shown in dark colors in the graph above. To make an even more compact analysis, a PCA (Principal Components Analysis) could be performed as pre-processing step to the datasets. Nevertheless, as the correlations are quite few, this step will not be applied for this assignment.

3. Prediction Model Building

Three popular methods will be applied to model the regressions (in the Train dataset) and the best one (with higher accuracy when applied to the Test dataset) will be used for the quiz predictions. The methods are: Random Forests, Decision Tree and Generalized Boosted Model, as described below. A Confusion Matrix is plotted at the end of each analysis to better visualize the accuracy of the models. *a. Random Forests*

```
# model fit
set.seed(301)
library(e1071)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=TrainSet, method="rf",
                           trControl=controlRF)
modFitRandForest$finalModel
```

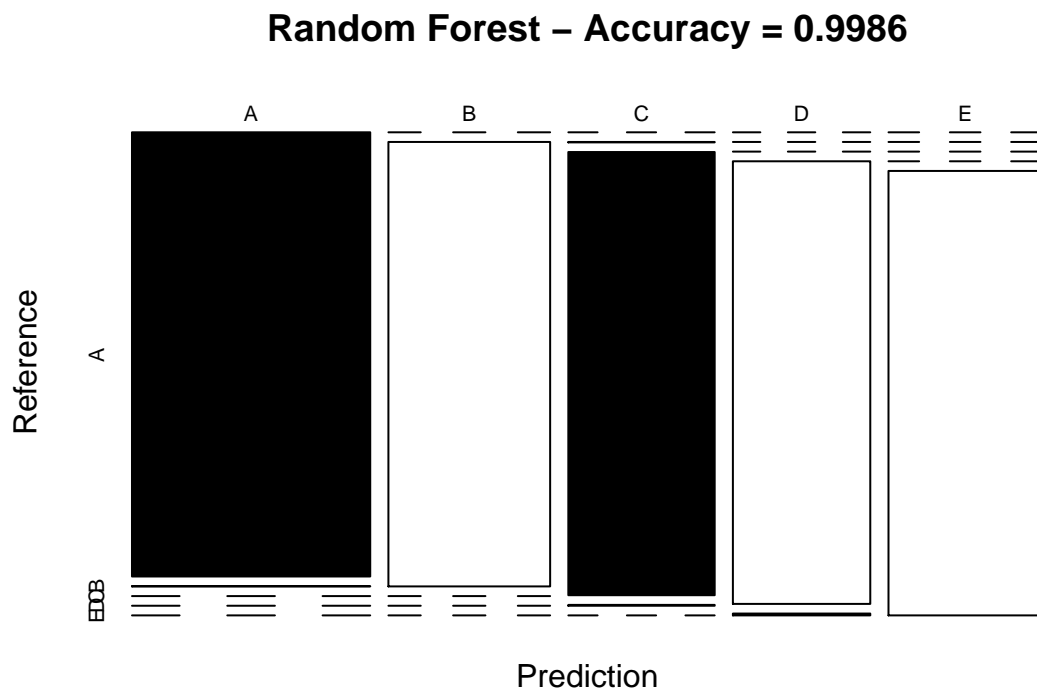
```
##
## Call:
## randomForest(x = x, y = y, mtry = param$mtry)
##               Type of random forest: classification
##               Number of trees: 500
## No. of variables tried at each split: 27
##
##               OOB estimate of  error rate: 0.22%
## Confusion matrix:
##      A      B      C      D      E  class.error
## A 3904      1      0      0      1 0.0005120328
## B      6 2648      3      1      0 0.0037622272
## C      0      4 2392      0      0 0.0016694491
## D      0      0      8 2243      1 0.0039964476
## E      0      1      0      4 2520 0.0019801980
```

```
# prediction on Test dataset
predictRandForest <- predict(modFitRandForest, newdata=TestSet)
confMatRandForest <- confusionMatrix(predictRandForest, TestSet$classe)
confMatRandForest
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction      A      B      C      D      E
##      A 1674      1      0      0      0
##      B      0 1137      0      0      0
##      C      0      1 1026      2      0
##      D      0      0      0 962      4
##      E      0      0      0      0 1078
##
## Overall Statistics
##
##               Accuracy : 0.9986
##               95% CI : (0.9973, 0.9994)
```

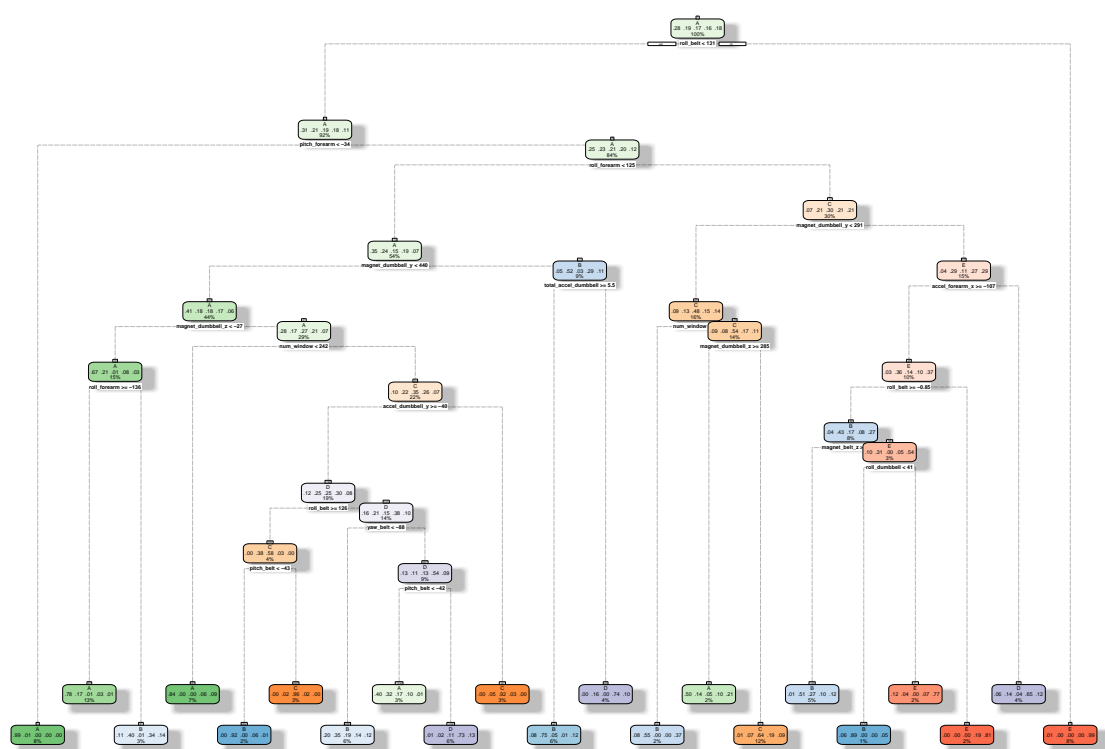
```
##      No Information Rate : 0.2845
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.9983
##
##      McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##              Class: A Class: B Class: C Class: D Class: E
## Sensitivity          1.0000   0.9982   1.0000   0.9979   0.9963
## Specificity          0.9998   1.0000   0.9994   0.9992   1.0000
## Pos Pred Value       0.9994   1.0000   0.9971   0.9959   1.0000
## Neg Pred Value       1.0000   0.9996   1.0000   0.9996   0.9992
## Prevalence           0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate       0.2845   0.1932   0.1743   0.1635   0.1832
## Detection Prevalence 0.2846   0.1932   0.1749   0.1641   0.1832
## Balanced Accuracy    0.9999   0.9991   0.9997   0.9986   0.9982
```

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



b. Decision Tree

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```

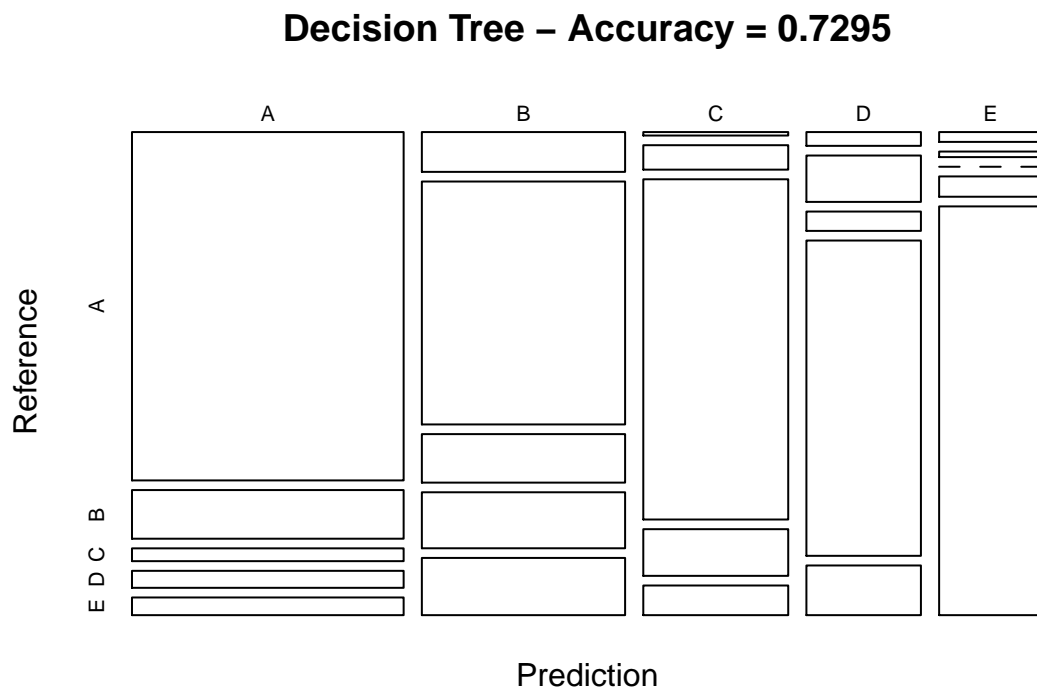


```
# prediction on Test dataset
predictDecTree <- predict(modFitDecTree, newdata=TestSet, type="class")
confMatDecTree <- confusionMatrix(predictDecTree, TestSet$classe)
confMatDecTree
```

6

```
##               Accuracy : 0.7295
##               95% CI   : (0.7179, 0.7408)
##      No Information Rate : 0.2845
##      P-Value [Acc > NIR] : < 2.2e-16
##
##               Kappa : 0.6556
##
##  McNemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##               Class: A Class: B Class: C Class: D Class: E
## Sensitivity      0.8943   0.6857   0.7602   0.59232  0.6137
## Specificity      0.9019   0.8635   0.9508   0.95245  0.9879
## Pos Pred Value   0.7838   0.5465   0.7655   0.70932  0.9197
## Neg Pred Value   0.9555   0.9197   0.9494   0.92264  0.9190
## Prevalence       0.2845   0.1935   0.1743   0.16381  0.1839
## Detection Rate   0.2544   0.1327   0.1325   0.09703  0.1128
## Detection Prevalence 0.3246 0.2428 0.1732 0.13679 0.1227
## Balanced Accuracy 0.8981   0.7746   0.8555   0.77239  0.8008
```

```
# plot matrix results
plot(confMatDecTree$table, col = confMatDecTree$byClass,
     main = paste("Decision Tree - Accuracy =",
                  round(confMatDecTree$overall['Accuracy'], 4)))
```



c. Generalized Boosted Model (GBM)

```
# model fit
set.seed(301)
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.
```

```
#prediction on Test dataset
predictGBM <- predict(modFitGBM, newdata=TestSet)
confMatGBM <- confusionMatrix(predictGBM, TestSet$classe)
confMatGBM
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   A    B    C    D    E
##           A 1671   11    0    2    1
##           B    3 1113    3    4    4
##           C    0   14 1020   12    1
##           D    0    1    2  945   11
##           E    0    0    1    1 1065
```

```
## Overall Statistics
```

```
##
##           Accuracy : 0.9879
##           95% CI : (0.9848, 0.9906)
##           No Information Rate : 0.2845
##           P-Value [Acc > NIR] : < 2.2e-16
```

```
##           Kappa : 0.9847
```

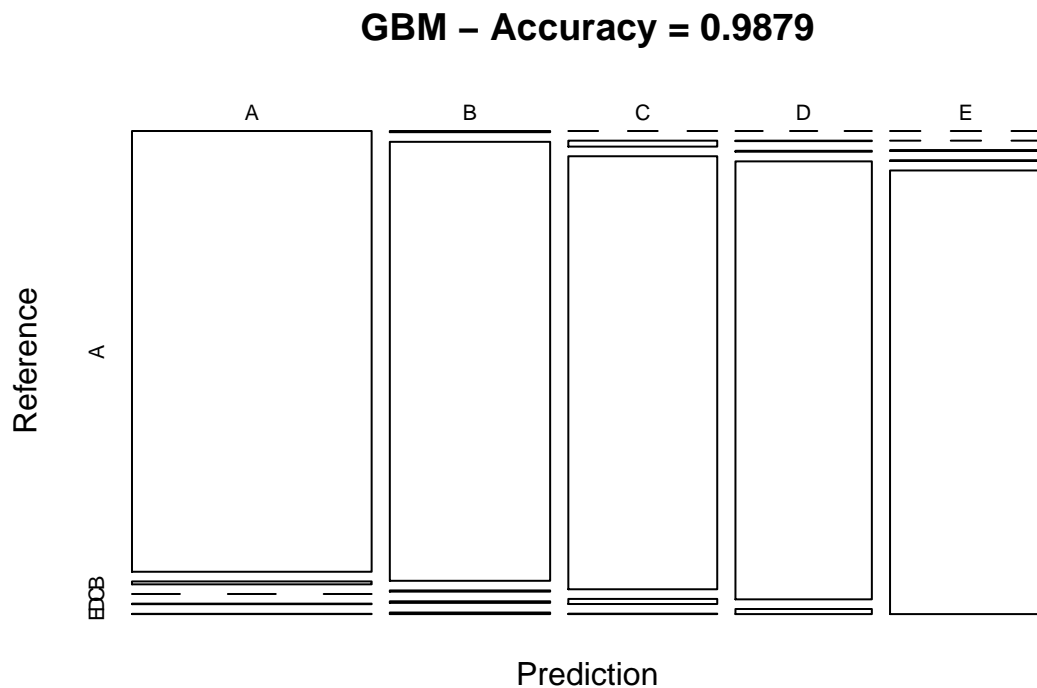
```
## Mcnemar's Test P-Value : NA
```

```
## Statistics by Class:
```

```
##
##           Class: A Class: B Class: C Class: D Class: E
## Sensitivity      0.9982  0.9772  0.9942  0.9803  0.9843
## Specificity      0.9967  0.9971  0.9944  0.9972  0.9996
## Pos Pred Value   0.9917  0.9876  0.9742  0.9854  0.9981
## Neg Pred Value   0.9993  0.9945  0.9988  0.9961  0.9965
## Prevalence       0.2845  0.1935  0.1743  0.1638  0.1839
## Detection Rate   0.2839  0.1891  0.1733  0.1606  0.1810
## Detection Prevalence 0.2863  0.1915  0.1779  0.1630  0.1813
## Balanced Accuracy 0.9974  0.9871  0.9943  0.9887  0.9919
```



```
# plot matrix results
plot(confMatGBM$table, col = confMatGBM$byClass,
     main = paste("GBM - Accuracy =", round(confMatGBM$overall['Accuracy'], 4)))
```



5. Applying the selected Model to the Test Data

The accuracy of the 3 regression modeling methods above are:

Random Forest : 0.9968 Decision Tree : 0.8291 GBM : 0.9884 In that case, the Random Forest model will be applied to predict the 20 quiz results (testing dataset) as shown below.

```
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
```