title: “Practical Machine Learning Project” author: “Mohamed Osama” date: “7/25/2020” —

Practical Machine Learning Project : Prediction Assignment

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 up 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. Background

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).

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz3xsbS5bVX

3. Data Loading and Exploratory Data Analysis

a) Dataset Overview

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 http://groupware.les.inf.puc-rio.br/har. Full source:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. “Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human ’13)”. Stuttgart, Germany: ACM SIGCHI, 2013.

My special thanks to the above mentioned authors for being so generous in allowing their data to be used for this kind of assignment.

A short description of the datasets content from the authors’ website:

“Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg)."

b) Environment Preparation

We first upload the R libraries that are necessary for the complete analysis.

```
rm(list=ls())                # free up memory for the download of the data sets



library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
library(rattle)
```

```
## 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.
```

```
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
```

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
set.seed(12345)
```

c) Data Loading and Cleaning

The next step is loading the dataset from the URL provided above. The training dataset is then partinioned in 2 to create a Training set (70% of the data) for the modeling process and a Test set (with the remaining 30%) for the validations. The testing dataset is not changed and will only be used for the quiz results generation.

```r
# set the URL for the download
UrlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
UrlTest  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# download the datasets
training <- read.csv(url(UrlTrain))
testing  <- read.csv(url(UrlTest))

# create a partition with the training dataset
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
TrainSet <- training[inTrain, ]
TestSet  <- training[-inTrain, ]
dim(TrainSet)
```

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

```r
dim(TestSet)
```

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

Both created datasets have 160 variables. Those variables have plenty of NA, that can be removed with the cleaning procedures below. The Near Zero variance (NZV) variables are also removed and the ID variables as well.

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

```
## [1] 13737   104
```

```r
dim(TestSet)
```

```
## [1] 5885  104
```

```
# 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(TrainSet)
```

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

```
dim(TestSet)
```

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

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

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

```
dim(TestSet)
```

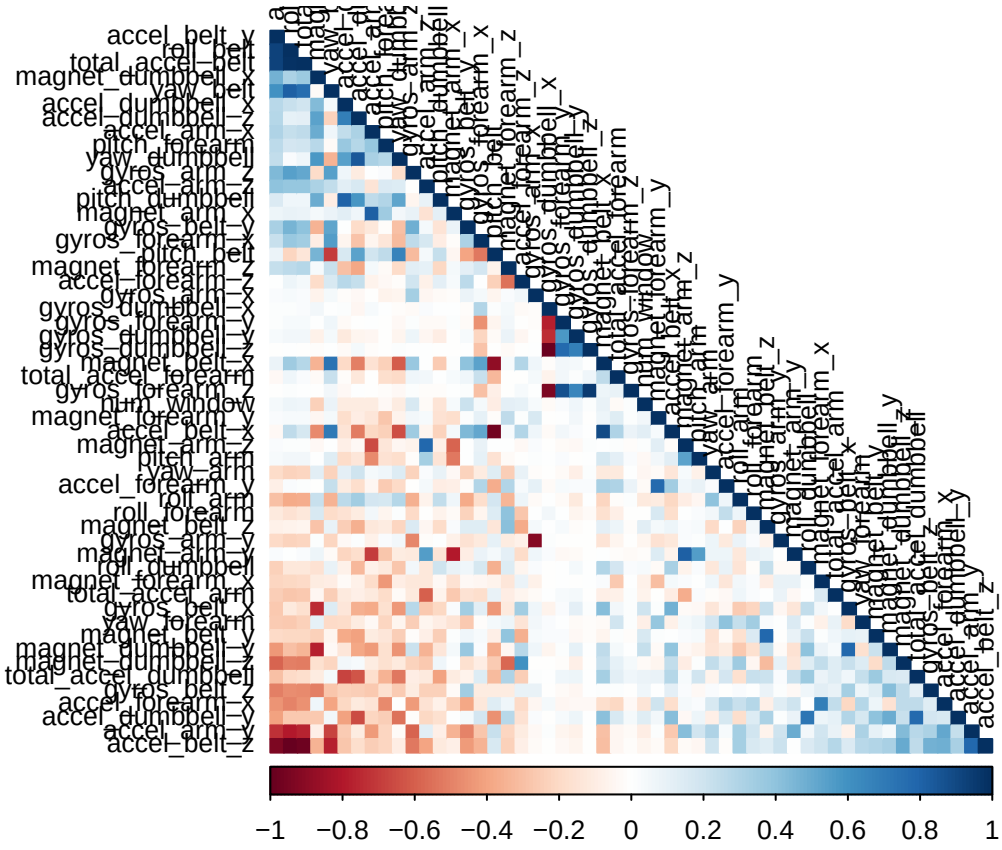
```
## [1] 5885   54
```

With the cleaning process above, the number of variables for the analysis has been reduced to 54 only.

d) 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))
```

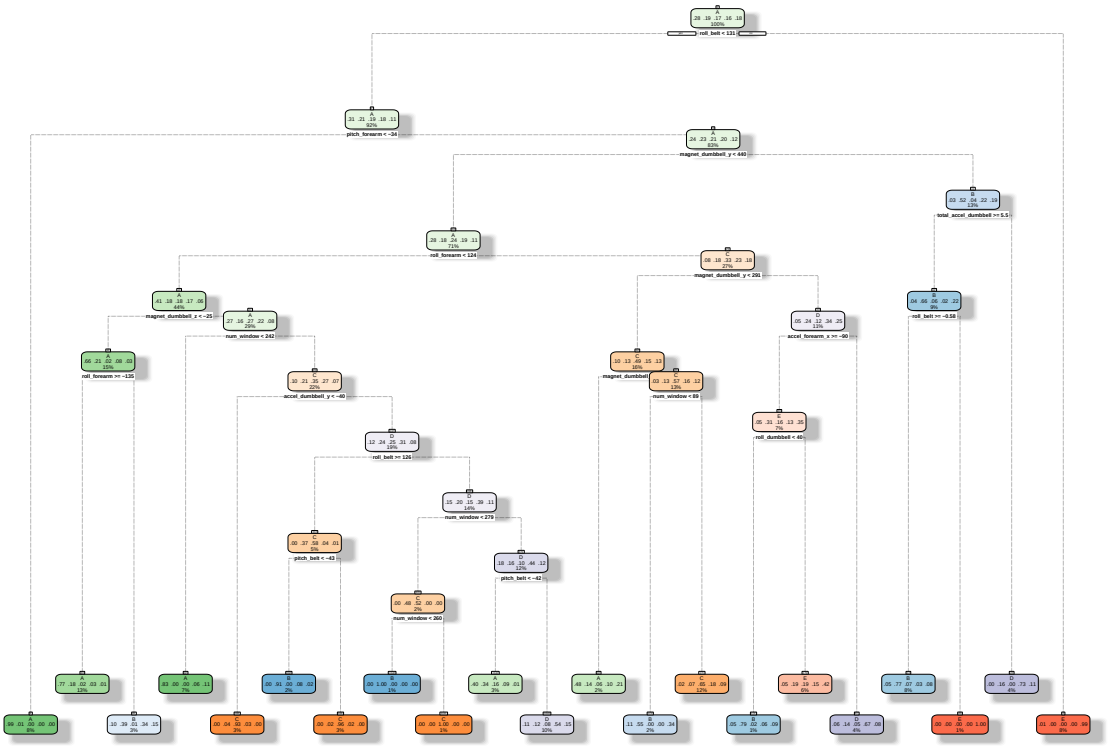


The highly correlated variables are shown in dark colors in the graph above. To make an evem more compact analysis, a PCA (Principal Components Analysis) could be performed as pre-processing step to the datasets. Nevertheless, as the correlations are quite few, this step will not be applied for this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

4.1. Decision Tree Model

```
# model fit
set.seed(1813)
fit_decision_tree <- rpart(classe ~ ., data = TrainSet, method="class")
fancyRpartPlot(fit_decision_tree)
```

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



Predictions of the decision tree model on TestSet.

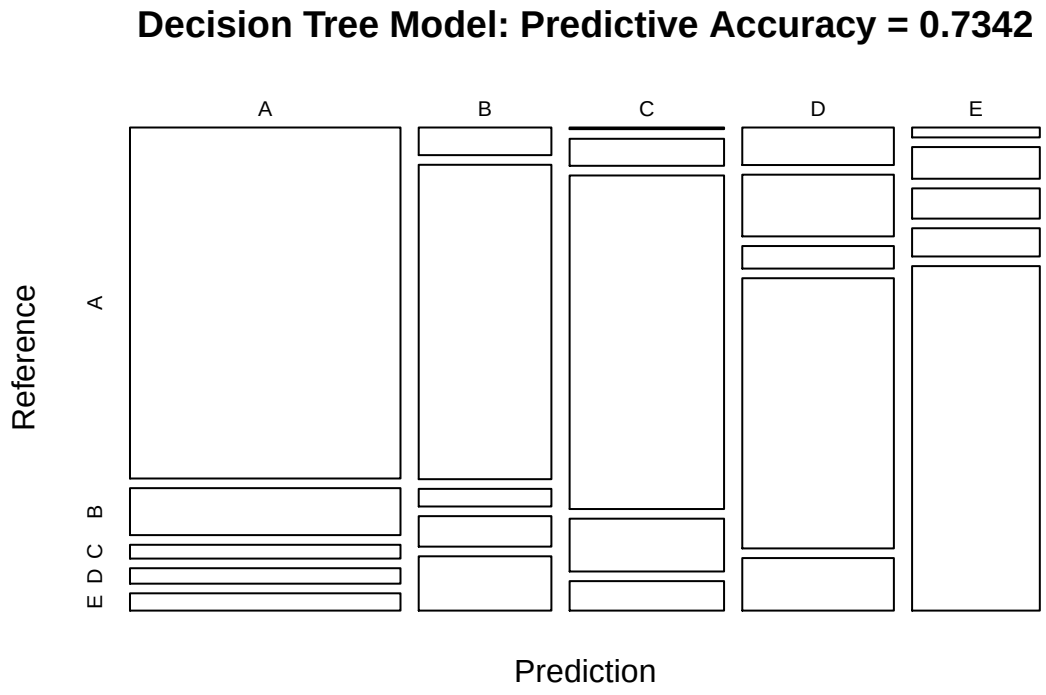
```
# prediction on Test dataset
predict_decision_tree <- predict(fit_decision_tree, newdata = TestSet, type="class")
conf_matrix_decision_tree <- confusionMatrix(predict_decision_tree,factor(TestSet$classe))
conf_matrix_decision_tree
```

```
## 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
```

The predictive accuracy of the decision tree model is relatively low at 73.4 %.
Plot the predictive accuracy of the decision tree model.

```
plot(conf_matrix_decision_tree$table, col = conf_matrix_decision_tree$byClass,
     main = paste("Decision Tree Model: Predictive Accuracy =",
                  round(conf_matrix_decision_tree$overall['Accuracy'], 4)))
```



4.2. Generalized Boosted Model (GBM)

```r
set.seed(1813)
ctrl_GBM <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
fit_GBM  <- train(classe ~ ., data = TrainSet, method = "gbm",
                  trControl = ctrl_GBM, verbose = FALSE)
fit_GBM$finalModel
```

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

```r
predict_GBM <- predict(fit_GBM, newdata = TestSet)
conf_matrix_GBM <- confusionMatrix(predict_GBM, factor(TestSet$classe))
conf_matrix_GBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669   11    0    0    0
##          B    4 1115   10    5    4
##          C    0   10 1014   18    0
##          D    1    3    2  941    4
##          E    0    0    0    0 1074
##
## Overall Statistics
##
##                Accuracy : 0.9878
##                  95% CI : (0.9846, 0.9904)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9845
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9789   0.9883   0.9761   0.9926
## Specificity            0.9974   0.9952   0.9942   0.9980   1.0000
## Pos Pred Value         0.9935   0.9798   0.9731   0.9895   1.0000
## Neg Pred Value         0.9988   0.9949   0.9975   0.9953   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2836   0.1895   0.1723   0.1599   0.1825
## Detection Prevalence   0.2855   0.1934   0.1771   0.1616   0.1825
## Balanced Accuracy      0.9972   0.9870   0.9913   0.9871   0.9963
```

The predictive accuracy of the GBM is relatively high at 98.78 %.

4.3. Random Forest Model

```
set.seed(1813)
ctrl_RF <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
fit_RF  <- train(classe ~ ., data = TrainSet, method = "rf",
                 trControl = ctrl_RF, verbose = FALSE)
fit_RF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, verbose = FALSE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.17%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    7 2648    2    1    0 0.0037622272
## C    0    4 2391    1    0 0.0020868114
## D    0    0    5 2247    0 0.0022202487
## E    0    0    0    2 2523 0.0007920792
```

Predictions of the Random Forest model on TestSet.

```
predict_RF <- predict(fit_RF, newdata = TestSet)
conf_matrix_RF <- confusionMatrix(predict_RF, factor(TestSet$classe))
conf_matrix_RF
```

```
## 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    1    0
##          D    0    0    0  963    1
##          E    0    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.9992
##                  95% CI : (0.998, 0.9997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9989
##
##  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   0.9990   0.9991
## Specificity            0.9998   0.9996   0.9998   0.9998   1.0000
## Pos Pred Value         0.9994   0.9982   0.9990   0.9990   1.0000
## Neg Pred Value         1.0000   0.9998   0.9996   0.9998   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1740   0.1636   0.1837
## Detection Prevalence   0.2846   0.1937   0.1742   0.1638   0.1837
## Balanced Accuracy      0.9999   0.9994   0.9989   0.9994   0.9995
```

The predictive accuracy of the Random Forest model is excellent at 99.9 %.
5. Applying the Best Predictive Model to the Test Data

To summarize, the predictive accuracy of the three models evaluated is as follows:

Decision Tree Model: 73.42 %
Generalized Boosted Model: 98.87 %
Random Forest Model: 99.92 %

The Random Forest model is selected and applied to make predictions on the 20 data points from the original testing dataset (data_quiz).

```
predict_quiz <- predict(fit_RF, newdata = testing)
predict_quiz
```

```
##  [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
```