Practical Machine Learning Course Project

N Touheed

8/29/2020

Introduction

We are thankful to the authors of the paper:

“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.”

for allowing us to use the WLE dataset.

We first downloaded the Weight Lifting Exercises training data (pml-training.csv) for this project into the working directory form the website:

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

Likewise, we also downloaded the Weight Lifting Exercises testing data (pml-testing.csv) for this project into the working directory form the website:

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

We, next define and populate two data frames from these two csv files:

```
trainDf <- read.csv("E:\\Coursera\\08 Practical Machine Learning\\Project\\pml-training.csv",
header = TRUE)
# Given testing data will be used for validation purpose
validationDf <- read.csv("E:\\Coursera\\08 Practical Machine Learning\\Project\\pml-testing.cs
v", header = TRUE)
```

Cleaning Data

We first clean the dataset and exclude those attributes which have missing values. We also remove few meaningless variables. We begin with cleaning the Near Zero Variance Variables.

```
library(caret)
```

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

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

```
nearZV <- nearZeroVar(trainDf, saveMetrics = TRUE)
```

```
trainDf01 <- trainDf[, !nearZV$nzv]
validationDf01 <- validationDf[, !nearZV$nzv]
dim(trainDf01); dim(validationDf01)
```

```
## [1] 19622   100
```

```
## [1]  20 100
```

Removing those attributes of the dataset that contribute very little to the accelerometer measurements.

```
toRemove <- grepl("^X|timestamp|user_name", names(trainDf01))
trainDf <- trainDf01[, !toRemove]
validationDf <- validationDf01[, !toRemove]
rm(toRemove); rm(trainDf01); rm(validationDf01)
dim(trainDf); dim(validationDf)
```

```
## [1] 19622    95
```

```
## [1] 20 95
```

In the final step of cleaning, we remove attributes that contain NA's.

```
removeCond <- (colSums(is.na(trainDf)) == 0)
trainDf <- trainDf[, removeCond]
validationDf <- validationDf[, removeCond]
rm(removeCond); dim(trainDf); dim(validationDf)
```

```
## [1] 19622    54
```

```
## [1] 20 54
```

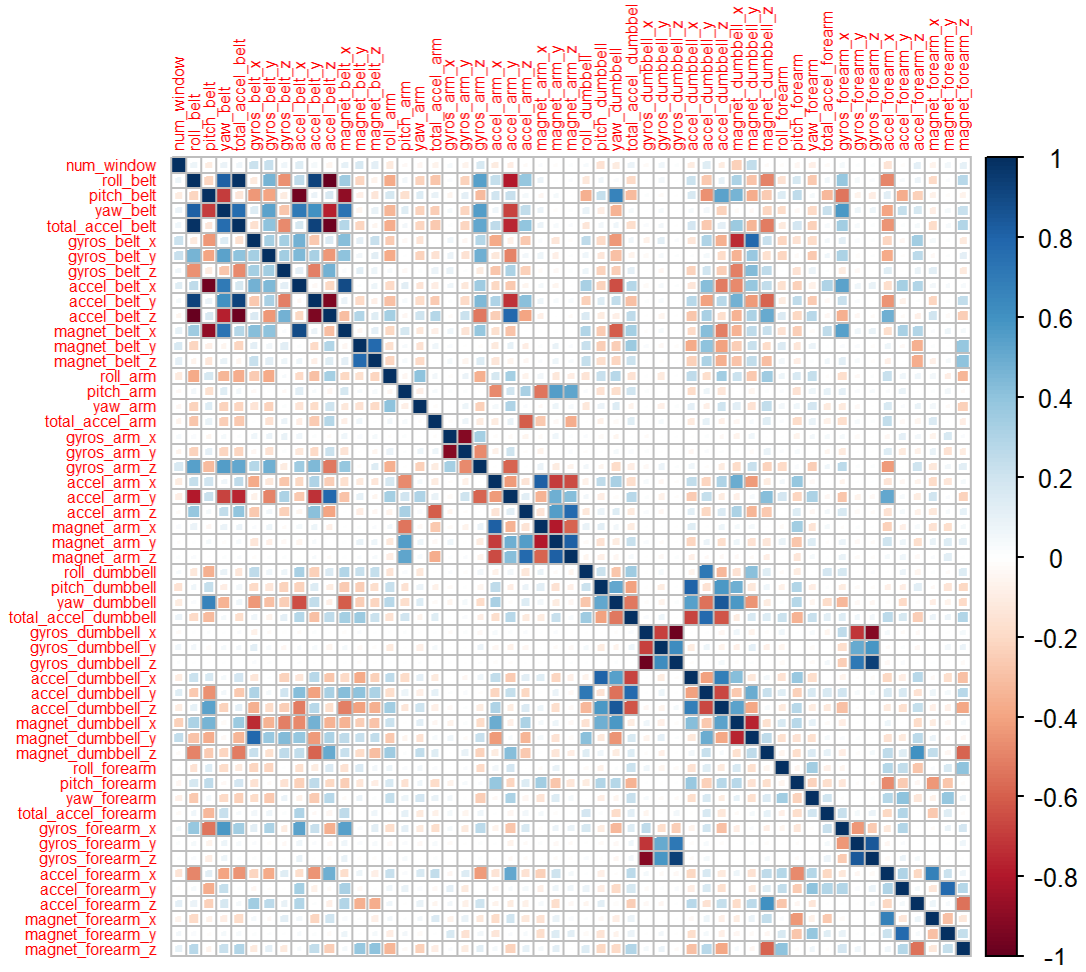
So, we endup with train data set containing 19622 observations and 54 attributes, while the validation data set contains 20 observations and same 54 attributes.

Let us have a Correlation Matrix of attributes in the train data set:

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
corrplot(cor(trainDf[, -length(names(trainDf))]), method = "square", tl.cex = 0.5)
```



Datasets for prediction

Let us Prepare the data for prediction by splitting the training data into 70% as train data and 30% as test data. This splitting will also be used to compute the out-of-sample errors.

It may please be noted that the original test data saved under validationDf will stay as is and will be used later to test the prediction algorithm on the 20 cases.

```
set.seed(135)
train <- createDataPartition(trainDf$classe, p = 0.7, list = FALSE)
trainDf0 <- trainDf
trainDf <- trainDf0[train, ]
testDf <- trainDf0[-train, ]
rm(trainDf0); dim(trainDf); dim(testDf)
```

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

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

Model 1 - Random Forest Algorithm

```
set.seed(1234)
rfControl <- trainControl(method="cv", number=3, verboseIter=FALSE)
```

```
rfModFit <- train(classe ~ ., data=trainDf, 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: 0.25%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    0    0    0    1 0.0002560164
## B    9 2646    2    1    0 0.0045146727
## C    0    4 2391    1    0 0.0020868114
## D    0    0   10 2241    1 0.0048845471
## E    0    0    0    5 2520 0.0019801980
```

Now, we estimate the performance of the model on the testing data set.

```
rfPredict <- predict(rfModFit, newdata=testDf)
rfConfMat <- confusionMatrix(factor(rfPredict), factor(testDf$classe))
rfConfMat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    9    0    0    0
##          B    0 1128    0    0    2
##          C    0    2 1023    4    0
##          D    0    0    3  960    6
##          E    0    0    0    0 1074
##
## Overall Statistics
##
##                Accuracy : 0.9956
##                  95% CI : (0.9935, 0.9971)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9944
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9903   0.9971   0.9959   0.9926
```

```
## Specificity            0.9979   0.9996   0.9988   0.9982   1.0000
## Pos Pred Value         0.9947   0.9982   0.9942   0.9907   1.0000
## Neg Pred Value         1.0000   0.9977   0.9994   0.9992   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1917   0.1738   0.1631   0.1825
## Detection Prevalence   0.2860   0.1920   0.1749   0.1647   0.1825
## Balanced Accuracy      0.9989   0.9950   0.9979   0.9970   0.9963
```

```
accuracy <- postResample(rfPredict, factor(testDf$classe))
outOfSampleError <- 1 - as.numeric((rfConfMat)$overall[1])
accuracy; outOfSampleError
```

```
## Accuracy    Kappa
## 0.995582 0.994411
```

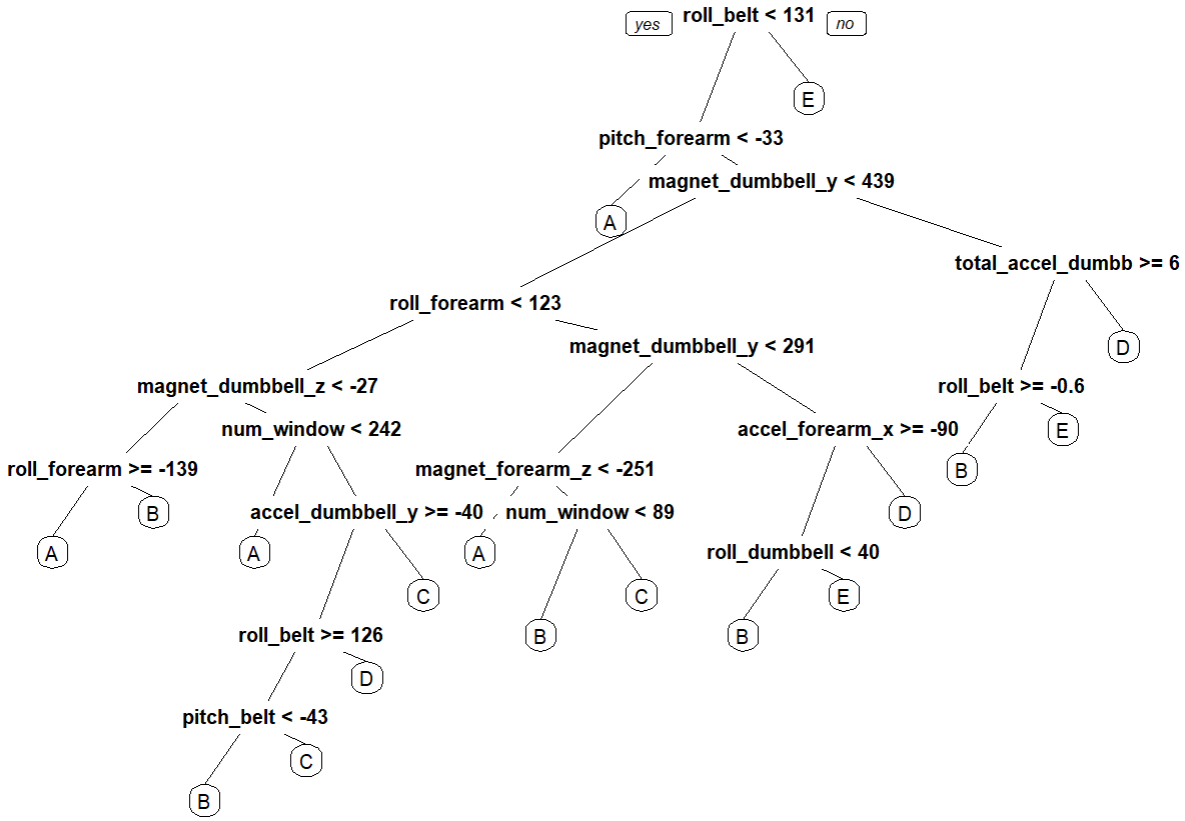
```
## [1] 0.004418012
```

The Estimated Accuracy of the Random Forest Model is 99.5582% and the Estimated Out-of-Sample Error is 0.4418012%. We will compare these values with other algorithms. Hopefully, Random Forests will yielded better Results, as happens in majority of the cases!

Model 2 - Decision Tree

We next fit a predictive model for activity recognition using Decision Tree algorithm.

```
library(rpart)
library(rpart.plot)
decisionTree <- rpart(classe ~ ., data = trainDf, method = "class")
prp(decisionTree)
```



Next, we estimate the performance of the decision tree model on the testing data set.

```
predictTree <- predict(decisionTree, testDf, type = "class")
dtConfMat <- confusionMatrix(factor(testDf$classe), factor(predictTree))
dtConfMat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1433   52   25  142   22
##          B  148  593   85  255   58
##          C   10   38  782  141   55
##          D   49   58  159  652   46
##          E   57  109   95  135  686
##
## Overall Statistics
##
##                Accuracy : 0.7045
##                  95% CI : (0.6927, 0.7161)
##     No Information Rate : 0.2884
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6269
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8444   0.6976   0.6824   0.4921   0.7912
## Specificity            0.9425   0.8916   0.9485   0.9316   0.9211
## Pos Pred Value         0.8560   0.5206   0.7622   0.6763   0.6340
## Neg Pred Value         0.9373   0.9458   0.9251   0.8632   0.9623
## Prevalence             0.2884   0.1444   0.1947   0.2251   0.1473
## Detection Rate         0.2435   0.1008   0.1329   0.1108   0.1166
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.8934   0.7946   0.8154   0.7118   0.8562
```

```
accuracy <- postResample(predictTree, factor(testDf$classe))
outOfSampleError <- 1 - as.numeric((dtConfMat)$overall[1])
accuracy; outOfSampleError
```

```
##  Accuracy     Kappa
## 0.7045030 0.6269466
```

```
## [1] 0.295497
```

The Estimated Accuracy of the Decision Tree Model is 70.45030% and the Estimated Out-of-Sample Error is 29.5497%. We can compare this with Random Forest Model. Decision Tree Model is less accurate and Out-of-Sample Error is also large.

Model 3 - At the end we like to have Prediction with Classification Trees.

Let us first obtain the model, and then use the fancyRpartPlot() function to plot the classification tree as a dendogram.

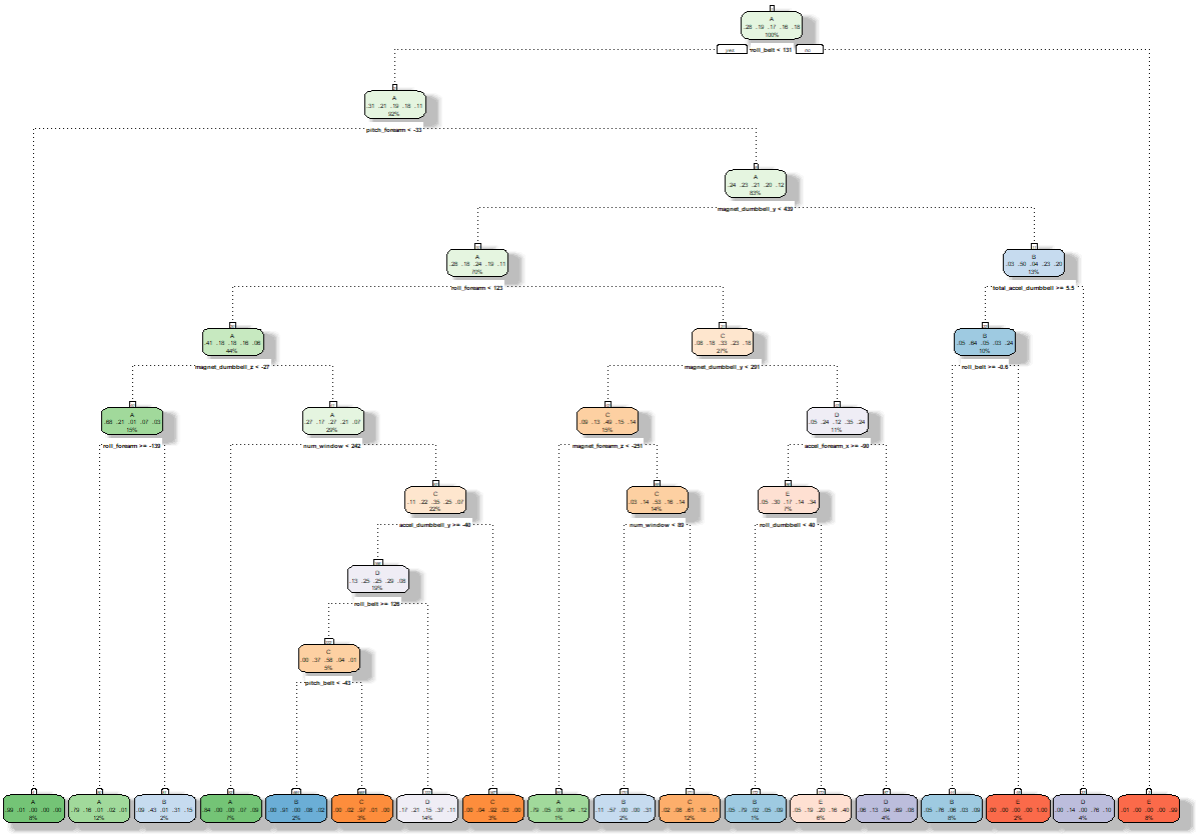
```
library(rattle)
```

```
## 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.
```

```
set.seed(135)
cTModFit <- rpart(classe ~ ., data=trainDf, method="class")
fancyRpartPlot(cTModFit)
```



Rattle 2020-Aug-29 16:29:09 ntouheed

Let us now validate the “cTModFit” model on the testDf to find out how well it performs by looking at the accuracy variable.

```
cTModFitPrediction <- predict(cTModFit, testDf, type = "class")
cTConfMat <- confusionMatrix(cTModFitPrediction, factor(testDf$classe))
cTConfMat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1433  148   10   49   57
##          B   52  593   38   58  109
##          C   25   85  782  159   95
##          D  142  255  141  652  135
##          E   22   58   55   46  686
##
## Overall Statistics
##
##                Accuracy : 0.7045
##                  95% CI : (0.6927, 0.7161)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6269
##
```

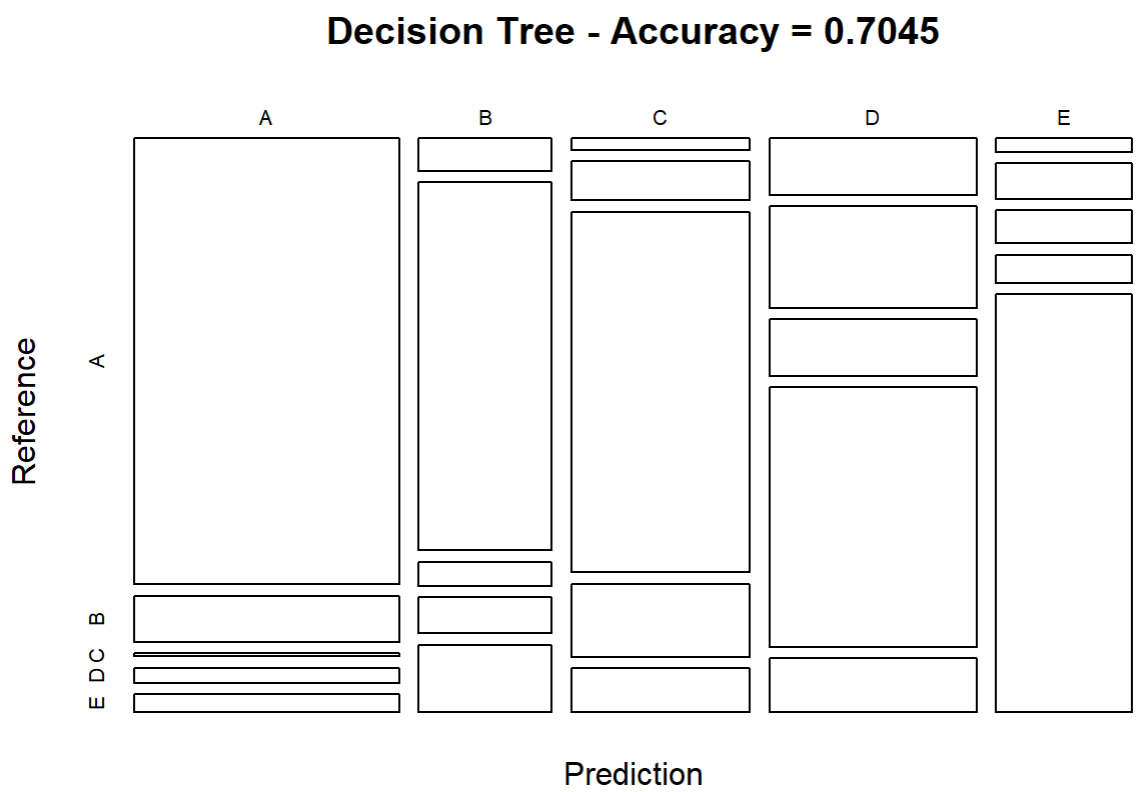
```
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8560   0.5206   0.7622   0.6763   0.6340
## Specificity            0.9373   0.9458   0.9251   0.8632   0.9623
## Pos Pred Value         0.8444   0.6976   0.6824   0.4921   0.7912
## Neg Pred Value         0.9425   0.8916   0.9485   0.9316   0.9211
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2435   0.1008   0.1329   0.1108   0.1166
## Detection Prevalence   0.2884   0.1444   0.1947   0.2251   0.1473
## Balanced Accuracy      0.8967   0.7332   0.8436   0.7698   0.7982
```

```
outOfSampleError <- 1 - as.numeric((cTConfMat)$overall[1])
outOfSampleError
```

```
## [1] 0.295497
```

Plot matrix results

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



We can see that the accuracy rate of the model is low: 0.7045 and therefore the out-of-sample-error is about 0.295497 which is considerable.

Conclusion.

Out of three predictions model considered here, the Random Forest Model is the best model.

Finally, we pay attention to the Course Project Prediction Quiz, i.e. Predicting the Manner of Exercise for the original testing data. We apply the Random Forest model to the original testing data set downloaded from the data source and named as validationDf. We remove the problem_id column first.

```
predict(rfModFit, validationDf[, -length(names(validationDf))])
```

```
##  [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
```