Practical Machine Learning (Prediction Assignment Writeup)

Ehsan Pourhadi

August 21, 2018

The aim of the project is to fit a Machine Learning model in order to predict the way some weight exercises were performed given data about the movement of subjects. For full information about the data, please visit http://groupware.les.inf.puc-rio.br/har

Source of Data

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 this source: 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.

Loading the Dataset

Set the directory where your source file is

```
setwd("F:/coursera.org/Course 8-Practical Machine Learning/week 4/course project")
```

We now load the dataset and understanting concerning with the data.

```
library(lattice)
library(ggplot2)
library(caret)
library(rpart)
library(rpart.plot)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(rattle)
```

```
## 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.
```

```
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(RColorBrewer)
```

```r
set.seed(1813)
```

Data Cleaning

```r
training <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testing  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
dt_train <- read.csv(url(training), strip.white = TRUE, na.strings = c("NA",""))
dt_test  <- read.csv(url(testing),  strip.white = TRUE, na.strings = c("NA",""))
```

```r
dim(dt_train)
```

```
## [1] 19622   160
```

```r
dim(dt_test)
```

```
## [1]  20 160
```

we Create two partitions (75 % and 25 %) within the initiall training dataset.

```r
in_train  <- createDataPartition(dt_train$classe, p=0.75, list=FALSE)
train_set <- dt_train[ in_train, ]
test_set  <- dt_train[-in_train, ]
```

```r
dim(train_set)
```

```
## [1] 14718   160
```

```r
dim(test_set)
```

```
## [1] 4904  160
```

The two datasets (train_set and test_set) have a large number of NA values as well as near-zero-variance (Non_Z.V.) variables. Both will be removed together with their ID variables.

```r
nonzv_var <- nearZeroVar(train_set)
train_set <- train_set[ , -nonzv_var]
test_set  <- test_set [ , -nonzv_var]
dim(train_set)
```

```
## [1] 14718   121
```

```r
dim(test_set)
```

```
## [1] 4904  121
```

Remove variables that are mostly NA. A threshlod of 95 % is taken.

```r
na_var <- sapply(train_set, function(x) mean(is.na(x))) > 0.95
train_set <- train_set[ , na_var == FALSE]
test_set  <- test_set [ , na_var == FALSE]
```

```r
dim(train_set)
```

```
## [1] 14718    59
```

```r
dim(test_set)
```

```
## [1] 4904   59
```

As we know that the columns 1 to 5 are identification variables only, they would be removed too.

```r
train_set <- train_set[ , -(1:5)]
test_set  <- test_set [ , -(1:5)]
dim(train_set)
```

```
## [1] 14718    54
```

```r
dim(test_set)
```

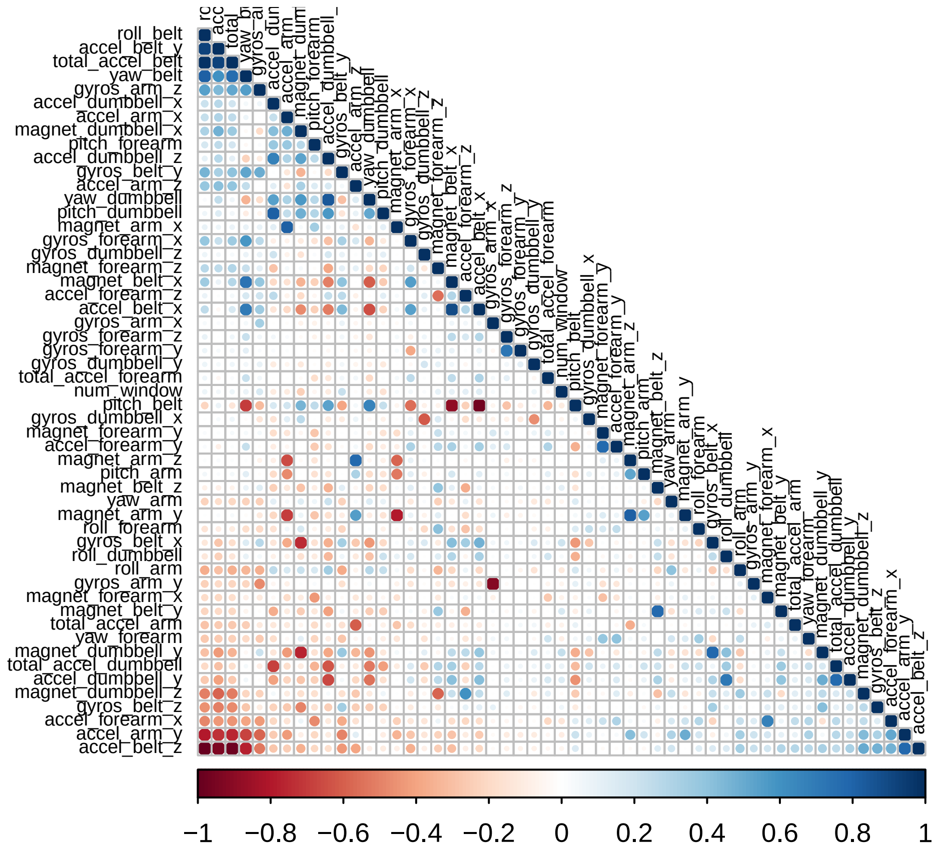
```
## [1] 4904   54
```

Hence, for the analysis, the number of variables has been decreased from the original 160 down to 54.

Analysis of Correlation of variables

Before the the prediction modeling, we Perform a correlation analysis between the variables. Pick "FPC" for the first principal component order.

```r
corr_matrix <- cor(train_set[ , -54])
corrplot(corr_matrix, order = "FPC", method = "circle", type = "lower",
         tl.cex = 0.6, tl.col = rgb(0, 0, 0))
```



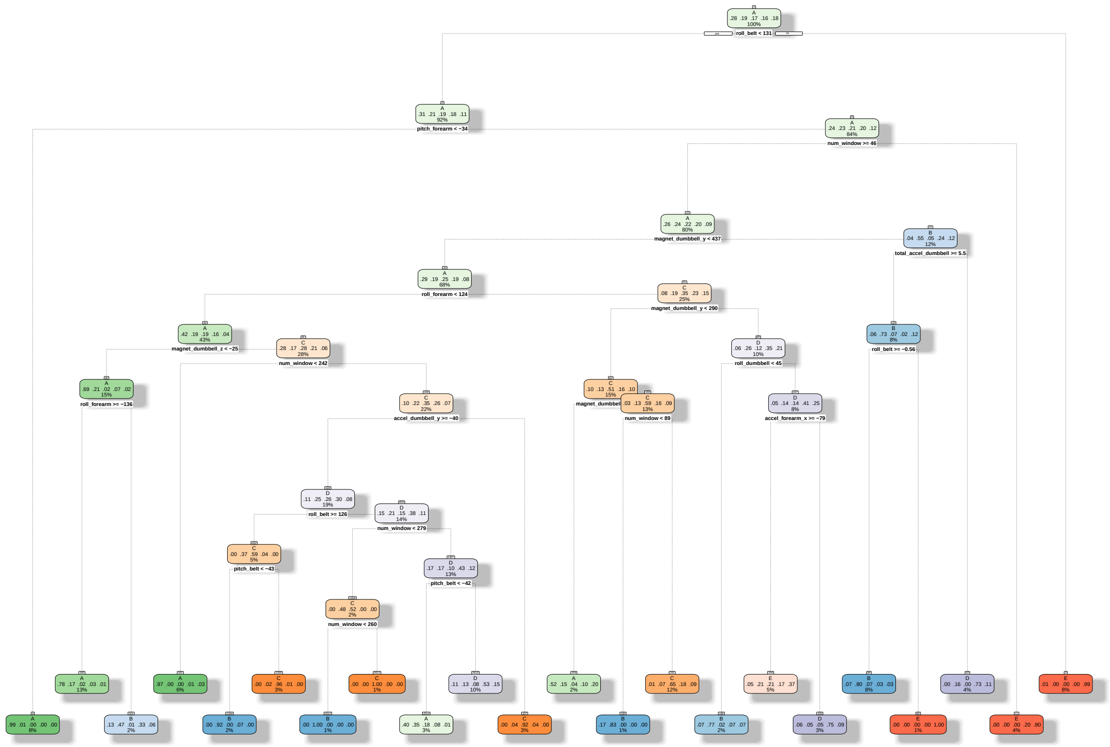
If two variables are highly correlated their colors are either dark blue (for a positive correlation) or dark red (for a negative corraltions). To further reduce the number of variables, a Principal Components Analysis (PCA) could be performed as the next step. However, since there are only very few strong correlations among the input variables, the PCA will not be performed. In fact, a few different prediction models will be made next.

Models of Prediction

Decision Tree Model

```r
set.seed(1813)
fit_decision_tree <- rpart(classe ~ ., data = train_set, method="class")
fancyRpartPlot(fit_decision_tree)
```

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



Rattle 2018−Aug−21 23:12:10 epour

Predictions of the decision tree model on test_set.

```
predict_decision_tree <- predict(fit_decision_tree, newdata = test_set, type="class")
conf_matrix_decision_tree <- confusionMatrix(predict_decision_tree, test_set$classe)
conf_matrix_decision_tree
```

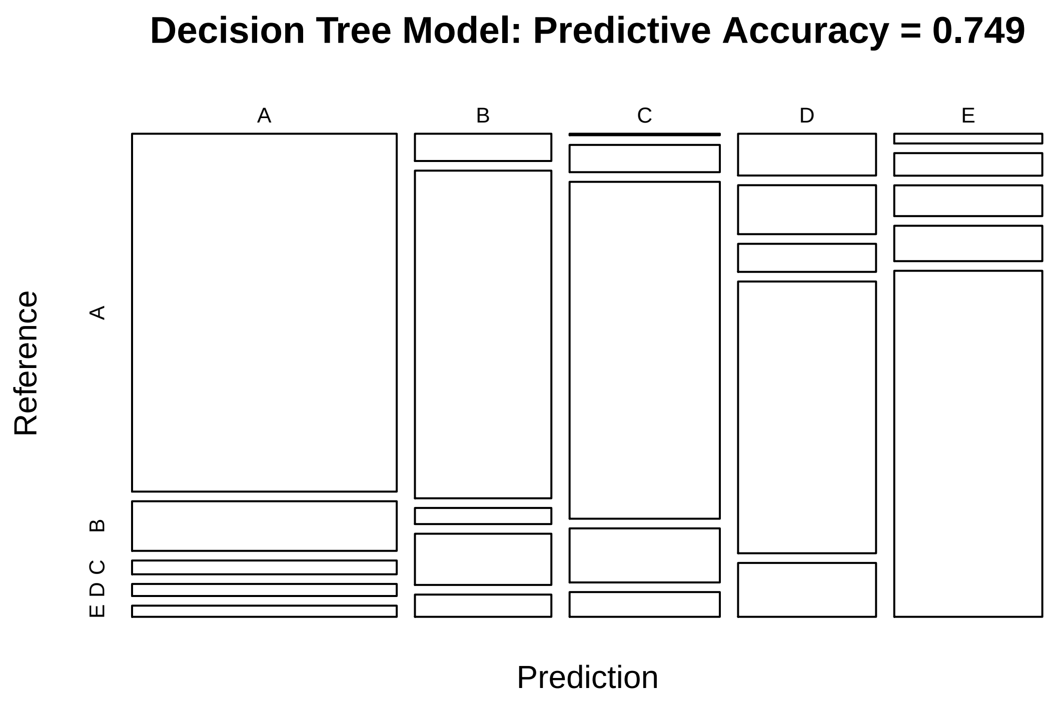
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1248  173   48   42   39
##          B   49  589   29   92   40
##          C    3   54  667  107   49
##          D   76   89   51  494   98
##          E   19   44   60   69  675
##
## Overall Statistics
##
##                Accuracy : 0.749
##                  95% CI : (0.7366, 0.7611)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6814
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8946   0.6207   0.7801   0.6144   0.7492
## Specificity            0.9139   0.9469   0.9474   0.9234   0.9520
## Pos Pred Value         0.8052   0.7372   0.7580   0.6114   0.7785
## Neg Pred Value         0.9562   0.9123   0.9533   0.9243   0.9440
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2545   0.1201   0.1360   0.1007   0.1376
## Detection Prevalence   0.3161   0.1629   0.1794   0.1648   0.1768
## Balanced Accuracy      0.9043   0.7838   0.8638   0.7689   0.8506
```

The predictive accuracy of the decision tree model is relatively low at 74.9 %.

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)))
```



Here, the algorithm which will be applied for the predictive model is Random Forest.

```
set.seed(1813)
modFitRF <- randomForest(classe ~ ., data = train_set, ntree = 100)
```

Predicting on the Testing Data

```r
predictionDT <- predict(fit_decision_tree, dt_test, type = "class")
predictionDT
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  E  A  A  C  D  D  A  A  C  E  A  A  E  D  A  B  B  B
## Levels: A B C D E
```

Predicting Random Forest

```r
predictionRF <- predict(modFitRF, dt_test, type = "class")
predictionRF
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

```r
predict_RF <- predict(modFitRF, newdata = test_set)
conf_matrix_RF <- confusionMatrix(predict_RF, test_set$classe)
conf_matrix_RF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    1    0    0    0
##          B    0  947    5    0    0
##          C    0    1  850    8    0
##          D    0    0    0  796    2
##          E    0    0    0    0  899
##
## Overall Statistics
##
##                Accuracy : 0.9965
##                  95% CI : (0.9945, 0.998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9956
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9979   0.9942   0.9900   0.9978
## Specificity            0.9997   0.9987   0.9978   0.9995   1.0000
## Pos Pred Value         0.9993   0.9947   0.9895   0.9975   1.0000
## Neg Pred Value         1.0000   0.9995   0.9988   0.9981   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1931   0.1733   0.1623   0.1833
## Detection Prevalence   0.2847   0.1941   0.1752   0.1627   0.1833
## Balanced Accuracy      0.9999   0.9983   0.9960   0.9948   0.9989
```

Conclusion

We observe from the confusion matrix that the Random Forest model is very accurate, approximately 99%. For this reason, we could expect nearly all of the submitted test cases to be correct. It turned out they were all correct.