Practical Machine Learning - Course Project

andre_manente

31/10/2019

Practical Machine Learning - Prediction Assignment Writeup

1. Summary

This document is the final report of the Peer Assessment project from the Practical Machine Learning course, which is a part of the Data Science Specialization. It was written and coded in RStudio, using its knitr functions and published in the html format. The purpose of this analysis is to predict the manner in which the six participants performed the exercises described below and to answer the questions of the associated course quiz. The machine learning algorithm, which uses the classe variable in the training set, is applied to the 20 test cases available in the test data. The predictions are submitted to the Course Project Prediction Quiz for grading.

2.Introduction

Devices such as Jawbone Up, Nike FuelBand, and Fitbit can enable collecting a large amount of data about someone's physical activity. These devices are used by the enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. However, even though these enthusiasts regularly quantify how much of a particular activity they do, they rarely quantify how well they do it. In this project, the goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of six participants. They were asked to perform barbell lifts correctly and incorrectly in five different ways.

More information is available from the following website: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

3. Source of Data

The data for this project can be found on the following website:

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

The training data for this project:

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

The test data for this project:

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

4. Data Loading and Cleaning

Load the required R packages and set a seed.

```
library(knitr)
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
```

```
library(RColorBrewer)
library(simEd)
```

```
## Loading required package: rstream

##
## Attaching package: 'simEd'

## The following objects are masked from 'package:base':
##
##     sample, set.seed
```

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

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

Load the training and test datasets.

```
data_train <- read.csv("pml-training.csv", strip.white = TRUE, na.strings = c("NA",""))
data_quiz  <- read.csv("pml-testing.csv",  strip.white = TRUE, na.strings = c("NA",""))
dim(data_train)
```

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

```
dim(data_quiz)
```

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

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

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

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

```
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 (NZV) variables. Both will be removed together with their ID variables.

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

```
## [1] 14718   123
```

```
dim(test_set)
```

```
## [1] 4904  123
```

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

```
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]
dim(train_set)
```

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

```r
dim(test_set)
```

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

Since columns 1 to 5 are identification variables only, they will be removed as well.

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

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

```r
dim(test_set)
```

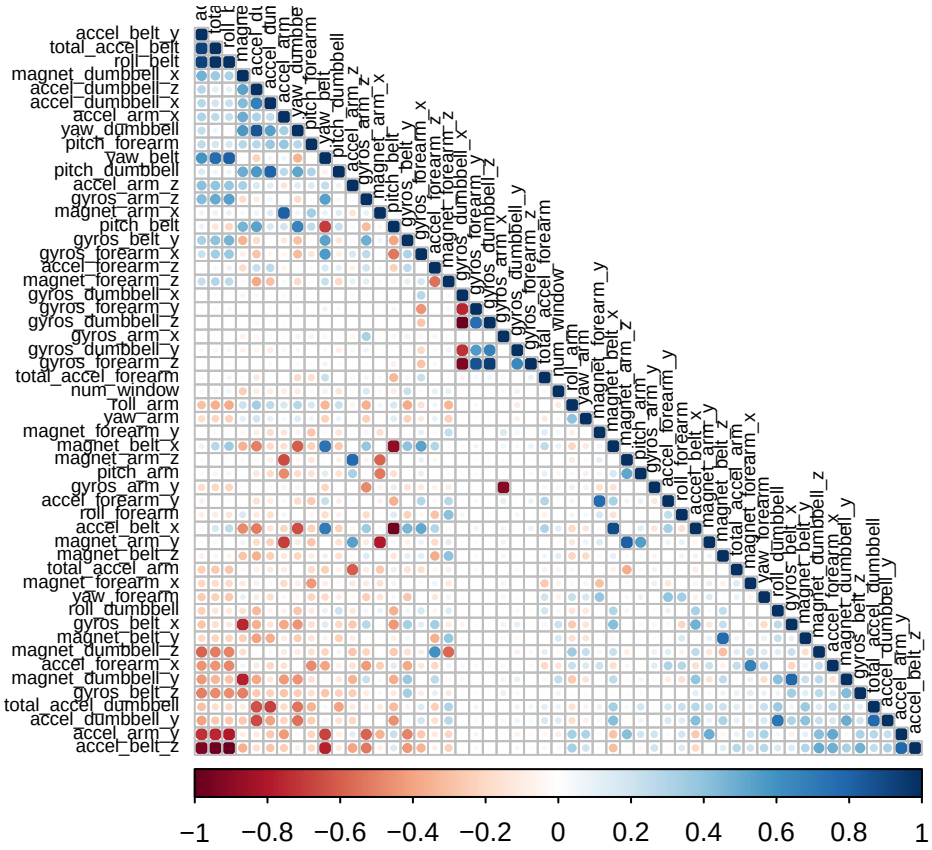
```
## [1] 4904   54
```

The number of variables for the analysis has been reduced from the original 160 down to 54.

5. Correlation Analysis

Perform a correlation analysis between the variables before the modeling work itself is done. Select "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))
```



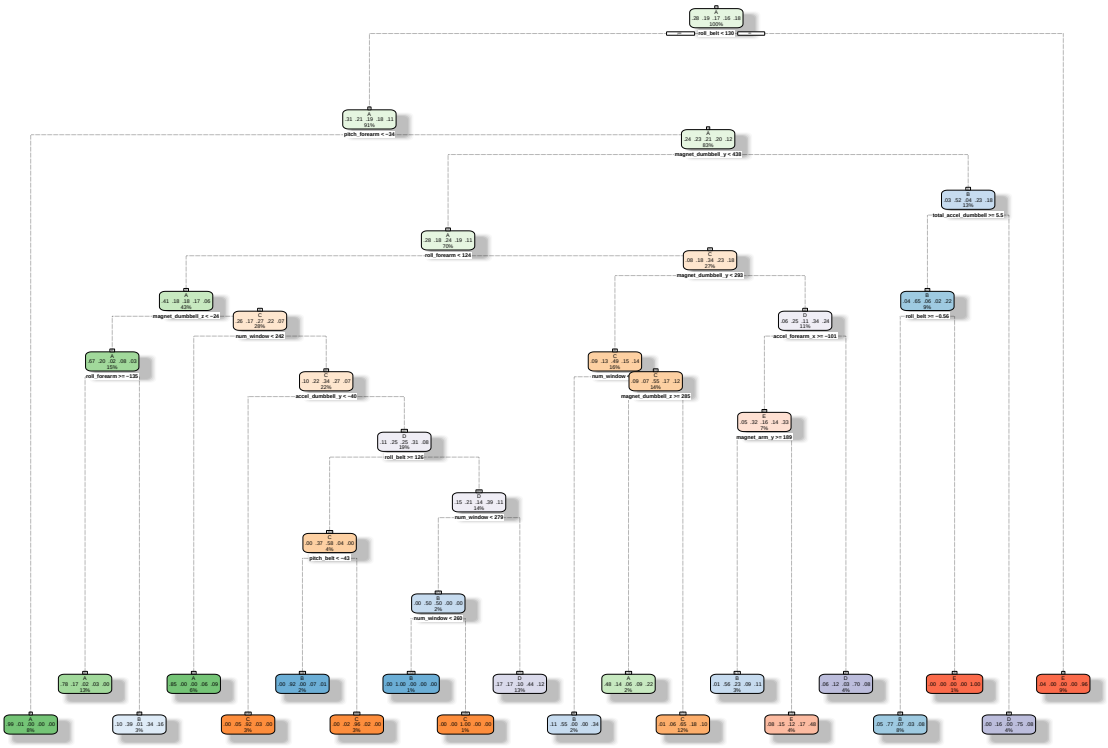
If two variables are highly correlated their colors are either dark blue (for a positive correlation) or dark red (for a negative corraltions). To further reduce the number of variables, a Principal Components Analysis (PCA) could be performed as the next step. However, since there are only very few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

6. Prediction Models

6.1. Decision Tree Model

```
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 2019−Oct−31 16:20:59 andre.silva

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
```

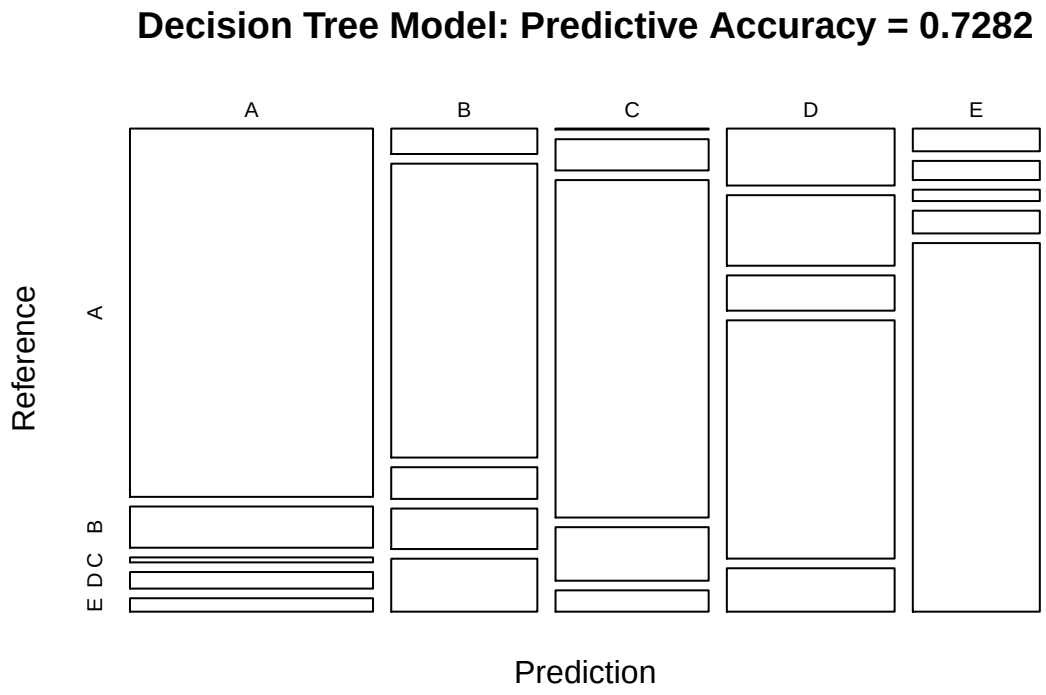
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1180  132   16   53   43
##          B   49  566   61   78  102
##          C    2   63  681  108   43
##          D  126  156   78  527   96
##          E   38   32   19   38  617
##
## Overall Statistics
##
##                Accuracy : 0.7282
##                  95% CI : (0.7155, 0.7406)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6564
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8459   0.5964   0.7965   0.6555   0.6848
## Specificity            0.9305   0.9267   0.9467   0.8888   0.9683
## Pos Pred Value         0.8287   0.6612   0.7592   0.5361   0.8293
## Neg Pred Value         0.9382   0.9054   0.9566   0.9294   0.9317
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2406   0.1154   0.1389   0.1075   0.1258
## Detection Prevalence   0.2904   0.1746   0.1829   0.2004   0.1517
## Balanced Accuracy      0.8882   0.7615   0.8716   0.7721   0.8265
```

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)))
```



6.2. Generalized Boosted Model (GBM)

```r
set.seed(12345)
ctrl_GBM <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
fit_GBM  <- train(classe ~ ., data = train_set, 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.
```

Predictions of the GBM on test_set.

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1391    6    0    0    0
##          B    4  932    4    0    4
##          C    0   10  849   19    2
##          D    0    1    2  781    5
##          E    0    0    0    4  890
##
## Overall Statistics
##
##                Accuracy : 0.9876
##                  95% CI : (0.9841, 0.9905)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9843
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9971   0.9821   0.9930   0.9714   0.9878
## Specificity            0.9983   0.9970   0.9923   0.9980   0.9990
## Pos Pred Value         0.9957   0.9873   0.9648   0.9899   0.9955
## Neg Pred Value         0.9989   0.9957   0.9985   0.9944   0.9973
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2836   0.1900   0.1731   0.1593   0.1815
## Detection Prevalence   0.2849   0.1925   0.1794   0.1609   0.1823
## Balanced Accuracy      0.9977   0.9895   0.9927   0.9847   0.9934
```

The predictive accuracy of the GBM is relatively high at 98.45 %.

6.3. Random Forest Model

```
set.seed(12345)
ctrl_RF <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
fit_RF  <- train(classe ~ ., data = train_set, 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: 0.2%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4183    1    0    0    1 0.0004778973
## B    4 2839    5    0    0 0.0031601124
## C    0    5 2562    0    0 0.0019477990
## D    0    0    8 2404    0 0.0033167496
## E    0    1    0    5 2700 0.0022172949
```

Predictions of the Random Forest model on test_set.

```
predict_RF <- predict(fit_RF, 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  948    1    0    0
##          C    0    0  854    3    0
##          D    0    0    0  800    0
##          E    0    0    0    1  901
##
## Overall Statistics
##
##                Accuracy : 0.9988
##                  95% CI : (0.9973, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9985
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9989   0.9988   0.9950   1.0000
## Specificity            0.9997   0.9997   0.9993   1.0000   0.9998
## Pos Pred Value         0.9993   0.9989   0.9965   1.0000   0.9989
## Neg Pred Value         1.0000   0.9997   0.9998   0.9990   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1933   0.1741   0.1631   0.1837
## Detection Prevalence   0.2847   0.1935   0.1748   0.1631   0.1839
## Balanced Accuracy      0.9999   0.9993   0.9990   0.9975   0.9999
```

The predictive accuracy of the Random Forest model is excellent at 99.8 %.

7. 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: 74.90 % Generalized Boosted Model: 98.45 % Random Forest Model: 99.80 % 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 = data_quiz)
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
```