Practical Machine Learning - Prediction Assignment Writeup

Maria Lyasheva

26 January 2019

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).

Aim

Apply a machine learning algorithm to the 20 test cases available in the test data above and submit the predictions in appropriate format to the Course Project Prediction Quiz for automated grading.

Data loading

The pml_training and pml_testing datasets were downloaded from the following websites: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv Original data can be found on the following website: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

The packages, reqired for the analysis, were loaded.

```
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 4.6-12

## 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)
```

The dateset pml_training were read and the information about this data set was obtained.

```
setwd("~/Desktop/Coursera Machine Learning")
url_train <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
data_train <- read.csv(url(url_train), strip.white = TRUE, na.strings =
c("NA",""))
View(data_train)
dim(data_train)

## [1] 19622   160
```

Similarly, the dateset pml_testing was read and information about this data set was extracted.

```
url_quiz  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
data_quiz  <- read.csv(url(url_quiz),  strip.white = TRUE, na.strings =
c("NA",""))
View(data_quiz)
dim(data_quiz)

## [1]  20 160
```

Both datasets were found to have the same number of variables - 160.

Data Loading and Cleaning

Data Partitioning: To do a data partitioning two partitions (75 % and 25 %) within the original training dataset (data_train) were created. This was used for cross validation purpose.

```
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
```

There were many NA values and near-zero-variance variables in two mentioned datasets (train_set and test_set). Thus, these values and variables as well as ID variables were be removed for the further analysis.

```
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
```

Folowing this, the variables that were close to NA were excluded via selecting a threshlod of 95%.

```
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

dim(test_set)

## [1] 4904   59
```

Lastly, colums 1, 2, 3, 4 and 5 were also removed from the datasets because they contained only identification variables.

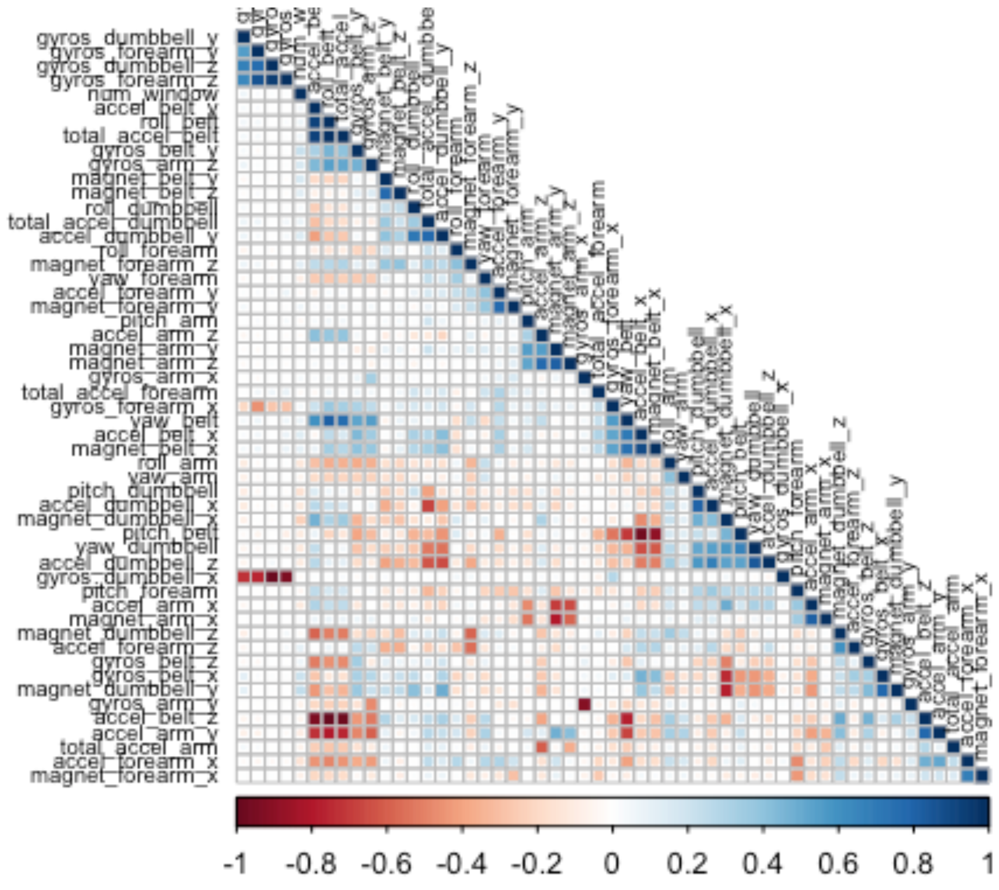
```
train_set <- train_set[ , -(1:5)]
test_set  <- test_set [ , -(1:5)]

dim(train_set)

## [1] 14718    54

dim(test_set)

## [1] 4904   54
```

As a result, the original number of variables (160) was reduced to 54.

Correlation Analysis

A correlation analysis was useful to highlight how much variables were correlated between each other. Thus, this analysis was performed before the modeling work itself. “Hclust” order was selected to organise the data. Figure 1: Correlation matrix visualization

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



The correlation coefficients were coloured according to the value (red is =-1 (negative corraltions), blue is =1 (positive correlation)) so it was easier to understand what was correlating with what and how much. A Principal Components Analysis (PCA) could be further applied in order to reduce the number of variables but there were not many variables that had strong correlations and, thus, PCA was not applied.

Prediction Models

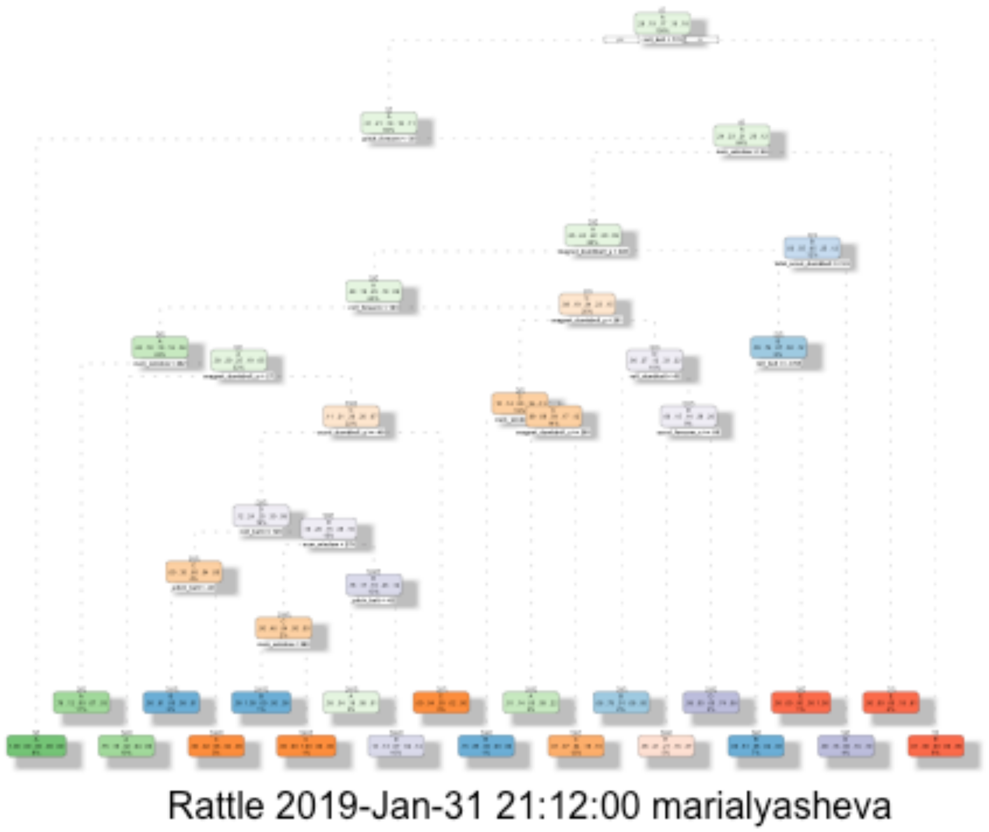
The models that were probed in this coursework were Decision Tree Model, Generalized Boosted Model (GBM) and Random Forest Model. This probing helped to choose the best prediction model.

1. Decision Tree Model

```r
set.seed(1834)
fit_decision_tree <- rpart(classe ~ ., data = train_set, method="class")
```

Figure 2: Decision Tree Plot

```r
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 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 1263  214   34   65   45
##          B   35  534   25   25   24
##          C   10   56  689  118   46
##          D   67   99   57  511  105
##          E   20   46   50   85  681
##
## Overall Statistics
##
##                Accuracy : 0.75
##                  95% CI : (0.7376, 0.7621)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6823
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9054   0.5627   0.8058   0.6356   0.7558
## Specificity            0.8980   0.9724   0.9432   0.9200   0.9498
## Pos Pred Value         0.7791   0.8305   0.7497   0.6091   0.7721
## Neg Pred Value         0.9598   0.9026   0.9583   0.9279   0.9453
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2575   0.1089   0.1405   0.1042   0.1389
## Detection Prevalence   0.3305   0.1311   0.1874   0.1711   0.1799
## Balanced Accuracy      0.9017   0.7676   0.8745   0.7778   0.8528
```

The predictive accuracy of the Decision Tree Model was low, 72.92%.

2. GBM

```
set.seed(1834)
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 42 had non-zero influence.
```

Predictions of the GBM on test_set.

```
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 1394   11    0    0    0
##          B    0  926    6    1    3
##          C    0   12  849   13    1
##          D    1    0    0  789    7
##          E    0    0    0    1  890
##
## Overall Statistics
##
##                Accuracy : 0.9886
##                  95% CI : (0.9852, 0.9914)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9856
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9758   0.9930   0.9813   0.9878
## Specificity            0.9969   0.9975   0.9936   0.9980   0.9998
## Pos Pred Value         0.9922   0.9893   0.9703   0.9900   0.9989
## Neg Pred Value         0.9997   0.9942   0.9985   0.9963   0.9973
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1888   0.1731   0.1609   0.1815
## Detection Prevalence   0.2865   0.1909   0.1784   0.1625   0.1817
## Balanced Accuracy      0.9981   0.9866   0.9933   0.9897   0.9938
```

The predictive accuracy of the GBM was high, 98.63%.

3. Random Forest Model

```
set.seed(1834)
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: 0.25%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4183    1    0    0    1 0.0004778973
## B    9 2834    4    1    0 0.0049157303
## C    0    4 2562    1    0 0.0019477990
## D    0    0    8 2403    1 0.0037313433
## E    0    0    0    7 2699 0.0025868441
```

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    0    0    0    0
##          B    0  949    1    0    0
##          C    0    0  854    5    0
##          D    0    0    0  799    0
##          E    0    0    0    0  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   1.0000   0.9988   0.9938   1.0000
## Specificity            1.0000   0.9997   0.9988   1.0000   1.0000
## Pos Pred Value         1.0000   0.9989   0.9942   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   0.9998   0.9988   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1935   0.1741   0.1629   0.1837
## Detection Prevalence   0.2845   0.1937   0.1752   0.1629   0.1837
## Balanced Accuracy      1.0000   0.9999   0.9988   0.9969   1.0000
```

The predictive accuracy of the Random Forest Model was very high, 99.82%.

Applying the Best Predictive Model to the Test Data

To summarise, Decision Tree Model had the lowest accuracy (72.92%), whereas Generalized Boosted Model and Random Forest Model had higher accuracy, 98.63% and 99.82% respectively. The Random Forest model was found to be the most accurate model and was 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
```