Practical Machine Learning Course Prediction Assignment

Quantified Self Movement Data Analysis Report

Barbara Henning

3/27/2020

Background and data available

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

The training data for this project are available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

What you should submit

The goal of this project is to predict the manner in which the participants did the exercise, which is measured by the “classe” variable in the training set. Any of the other variables can be used to predict with. This report should describe:
1. how the model was built;
2. how the cross validation was performed;
3. what is the expected out of sample error;
4. why each choice was made.
Later, the prediction model will also be used to predict 20 different test cases.

The submission for the Peer Review portion should consist of a link to a Github repo with your R markdown and compiled HTML file describing the analysis. The ext should be constrained to:
A. up to 2000 words of the writeup;
B. less than 5 figures.

Course Project Prediction Quiz Portion

To apply the final 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

```
# load required lbraries and setting set
library(GGally)
```

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

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(caret)
```

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

```
library(rpart)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
set.seed(3433)
```

```
#trainning data
trainUrl = "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
# testing data
testUrl = "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

trainData = read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""), row.names = "X")
testData = read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""), row.names = "X")
```

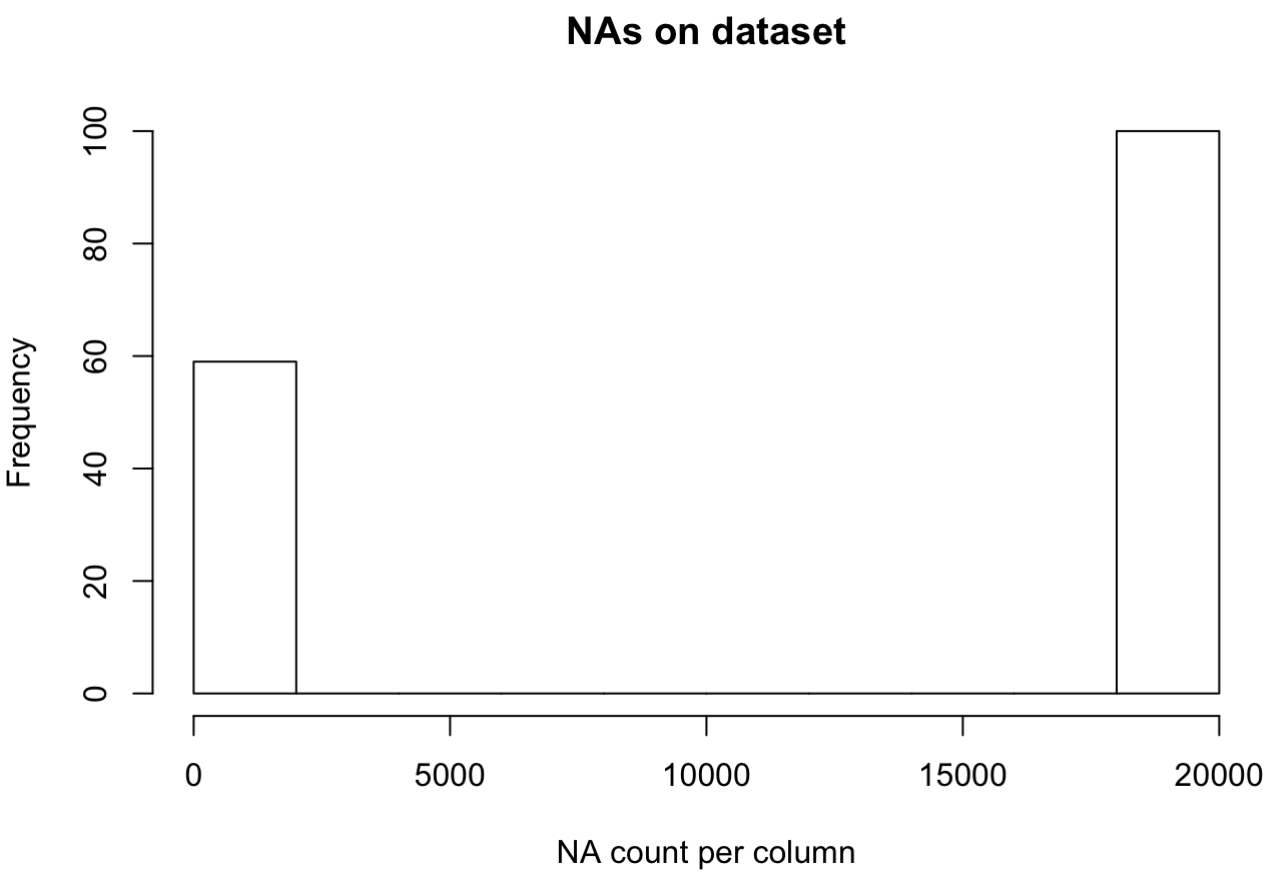
Data exploration and data cleaning

The dataset has 19622 observations. The following histogram shows that around 60 variables has almost zero NA values (NA count per column is close to zero) and around 100 variables have almost all observations as NA values (NA count per column is over 15000). Therefore, these variables with high number of NA values can be removed from the dataset because they are not informative for the model.

```
# checking if variables are in the same sequence in both train and test sets
colnames(trainData)[colnames(trainData)!=colnames(testData)]
```

```
## [1] "classe"
```

```
# checking the presence of NAs
hist(colSums(is.na(trainData)), main = "NAs on dataset", xlab = "NA count per column"
)
```




```
# because there are columns (variables) with almost no data, they can be removed from traing set
trainData = trainData[, colSums(is.na(trainData)) == 0]
testData = testData[, colSums(is.na(testData)) == 0]

# check variblese sequence again
colnames(trainData)[colnames(trainData)!=colnames(testData)]
```

```
## [1] "classe"
```

Training data paritioning

For the model training and validation step, the initial training data is split into two data sets, the training set with 60% of the observations and the validation set with 40% of the observations:

```
inTrain = createDataPartition(y = trainData$classe, p = 0.6, list = FALSE)
trainSet <- trainData[inTrain, ]
validSet <- trainData[-inTrain, ]
```

Machine learning model selection

Random forest and decision tree models are highly intepretable and robust classification models. Therefore, here I am fitting these two models to the training set and then comparing their performance for predction on the validation set.

Random forest

I will use **5-fold cross validation** when applying the algorithm.

```
modRF = train(classe ~ ., data = trainSet, method = "rf",
              trControl = trainControl(method = "cv", 5), ntree = 250)
modRF
```

```
## Random Forest
##
## 11776 samples
##    58 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9420, 9421, 9422, 9420, 9421
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9860735  0.9823807
##   41    0.9986413  0.9982815
##   80    0.9975375  0.9968853
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 41.
```

```
predictRF = predict(modRF, validSet)
confusionMatrix(validSet$classe, predictRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    0    0    0    0
##          B    0 1518    0    0    0
##          C    0    3 1365    0    0
##          D    0    0    0 1286    0
##          E    0    0    0    1 1441
##
## Overall Statistics
##
##                Accuracy : 0.9995
##                  95% CI : (0.9987, 0.9999)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9994
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9980   1.0000   0.9992   1.0000
## Specificity            1.0000   1.0000   0.9995   1.0000   0.9998
## Pos Pred Value         1.0000   1.0000   0.9978   1.0000   0.9993
## Neg Pred Value         1.0000   0.9995   1.0000   0.9998   1.0000
## Prevalence             0.2845   0.1939   0.1740   0.1640   0.1837
## Detection Rate         0.2845   0.1935   0.1740   0.1639   0.1837
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   0.9990   0.9998   0.9996   0.9999
```

```
accuracy = postResample(predictRF, validSet$classe)
ose = 1 - as.numeric(confusionMatrix(validSet$classe, predictRF)$overall[1])
```

The Random forest estimated accuracy was 99.9490186% and the estimated Out-of-Sample Error was 0.0509814%.

Decision Tree

```
modDT <- rpart(classe ~ ., data = trainSet, method = "class")

predictDT <- predict(modDT, validSet, type = "class")
confusionMatrix(validSet$classe, predictDT)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2135   77   20    0    0
##          B   62 1254  198    4    0
##          C   10   88 1242   28    0
##          D    2   63  145 1010   66
##          E    0    0   56  172 1214
##
## Overall Statistics
##
##                Accuracy : 0.8737
##                  95% CI : (0.8661, 0.881)
##     No Information Rate : 0.2815
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8403
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9665   0.8462   0.7477   0.8320   0.9484
## Specificity            0.9828   0.9585   0.9796   0.9584   0.9653
## Pos Pred Value         0.9565   0.8261   0.9079   0.7854   0.8419
## Neg Pred Value         0.9868   0.9640   0.9353   0.9689   0.9897
## Prevalence             0.2815   0.1889   0.2117   0.1547   0.1631
## Detection Rate         0.2721   0.1598   0.1583   0.1287   0.1547
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9746   0.9023   0.8637   0.8952   0.9569
```

```
accuracyDT <- postResample(predictDT, validSet$classe)
oseDT <- 1 - as.numeric(confusionMatrix(validSet$classe, predictDT)$overall[1])
```

The Random forest estimated accuracy was 87.3693602% and the estimated Out-of-Sample Error was 12.6306398%.

Finally, the conclusion is that the Random Forests algorithm yielded better performance.

Application of the selected model to the testing set

```
testPred = predict(modRF, testData)
testPred
```

```
##  [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
```