Practical Machine Learning Peer-graded Assignment: Prediction Assignment Writeup

U.Esparza

8/12/2020

Synopsis

The aim of this study is to predict the manner (“classe”) in which some healthy subjects performed a weight lifting exercise.

The subjects carried out the excercise in different fashions (some correct and some wrong). Their movements were monitorized using devices equipped with accelerometers and stored in datasets that are available in the “WayBack Machine” website: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har.

Data download and required package loading

```
library(AppliedPredictiveModeling)
```

```
## Warning: package 'AppliedPredictiveModeling' was built under R version 4.0.2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.2
```

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

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

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.2
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
## 
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
## 
##     combine
```

```
## The following object is masked from 'package:ggplot2':
## 
##     margin
```

```
urlTrain <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
urlTest <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

download.file(urlTrain, destfile =  "./pml-training.csv")
download.file(urlTest, destfile =  "./pml-testing.csv")

training <- read.csv("pml-training.csv", na.strings=c("", "NA"))
testing <- read.csv("pml-testing.csv", na.strings=c("", "NA"))
unique(training$classe)
```

```
## [1] "A" "B" "C" "D" "E"
```

Data Exploratory Analysis

The str() and table() functions are used to understand the basic structure of the dataset. Due to the high number of columns (160), the result is subsetted:

```
ncol(training)
```

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

```
str(training[,1:10]) # fist 10 columns. The first variables are not actual predictors
```

```
## 'data.frame':    19622 obs. of  10 variables:
##  $ X                   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name           : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1: int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084
232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2: int  788290 808298 820366 120339 196328 304277 368296 440390 484323
484434 ...
##  $ cvtd_timestamp      : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23" "05/1
2/2011 11:23" ...
##  $ new_window          : chr  "no" "no" "no" "no" ...
##  $ num_window          : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4
...
```

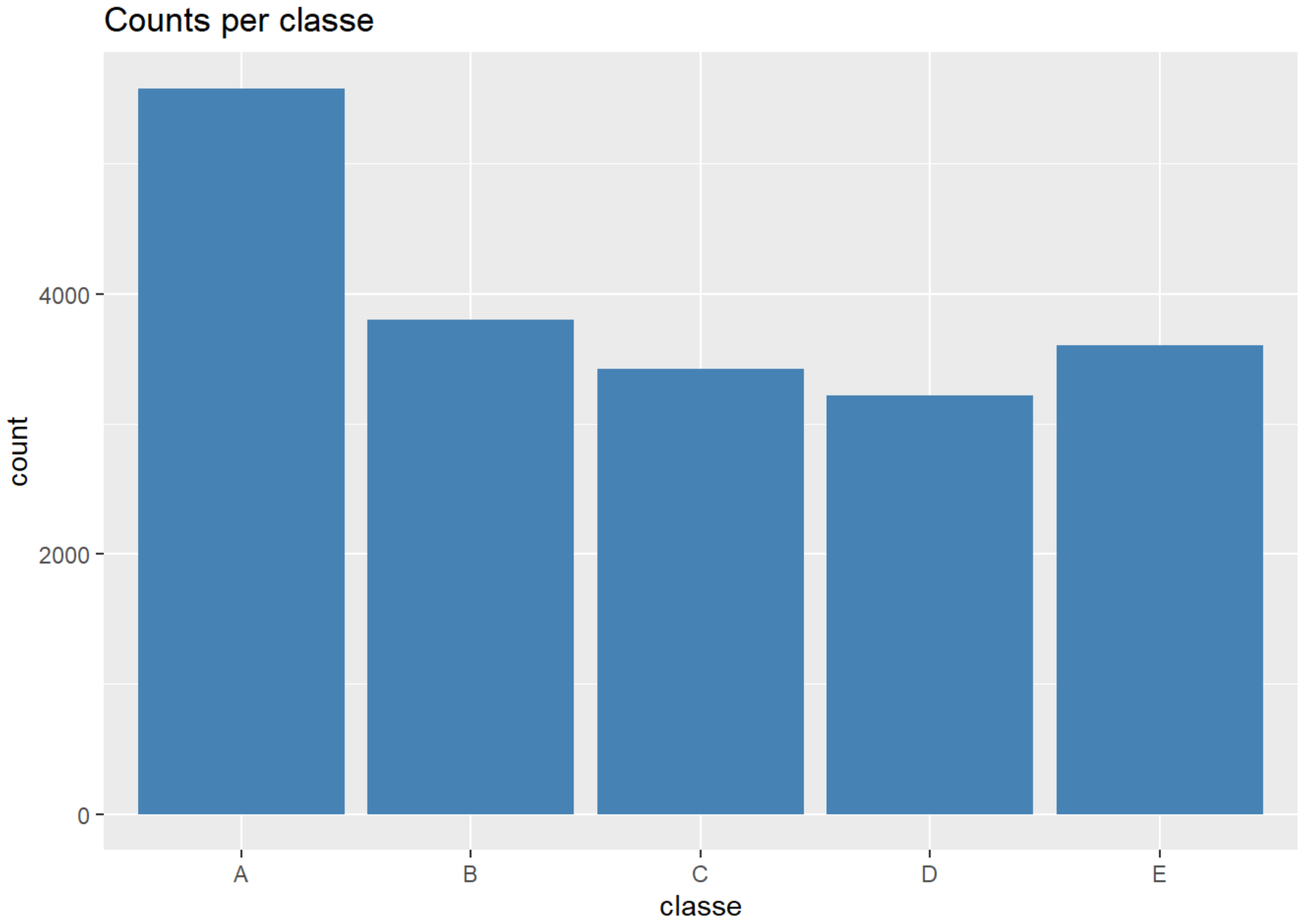
```
str(training[,149:160]) # last 12 columns. The outcome Classe appears at the end. Some columns
 appear to have plenty of NAs.
```

```
## 'data.frame':    19622 obs. of  12 variables:
##  $ stddev_yaw_forearm: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_forearm   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_forearm_x   : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y   : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z   : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x   : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y   : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z   : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x  : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y  : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z  : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe            : chr  "A" "A" "A" "A" ...
```

```
table(training$classe,training$user_name) # number of observations per "user_name" and per "cla
sse"
```

```
## 
##     adelmo carlitos charles eurico jeremy pedro
##   A   1165      834     899    865   1177   640
##   B    776      690     745    592    489   505
##   C    750      493     539    489    652   499
##   D    515      486     642    582    522   469
##   E    686      609     711    542    562   497
```

```
ggplot(training, aes(classe)) + geom_bar(fill = "steelblue") + ggtitle("Counts per classe")
```



The plot shows that there is a relatively balanced distribution of observations among "classe" types.

Data pre-processing

The outcome "classe" must be converted into a factor variable. Additionally, there are many columns which do not provide any relevant information, because they either have plenty of NAs or because they are not actual predictors obtained from accelerator measurements. Those columns will be removed:

```
training$classe <- as.factor(training$classe) # classe is converted into a factor variable.

trainingPrep <- training %>% select(8:160) # Non-predictors are removed.

trainingPrep <- trainingPrep %>% select_if(colSums(is.na(trainingPrep)) < 19000) # Only the col
umns with LESS than 19000 NAs are left (total nr. of obs. is 19622)

ncol(trainingPrep) # The resulting amount of columns in the dataset is 53.
```

```
## [1] 53
```

Create Data Partition

This dataset is further divided into train (75%) and test (25%) parts for cross-validation:

```
inTrain = createDataPartition(trainingPrep$classe, p = 3/4)[[1]]
trainPart = trainingPrep[ inTrain,]
testPart = trainingPrep[-inTrain,]
```

Model training

A couple of models will be trained and tested with cross validation to find out which of them has the highest accuracy level. More precisely, a random forest model and an LDA model will be tested:

```
set.seed(1234)
modfitrf <- randomForest(classe~., method = "class", data = trainPart)
predrf <- predict(modfitrf, newdata = testPart, type = "class")
confusionMatrix(predrf, testPart$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    3    0    0    0
##          B    0  944    6    0    0
##          C    0    2  849    7    0
##          D    0    0    0  796    3
##          E    0    0    0    1  898
##
## Overall Statistics
##
##                Accuracy : 0.9955
##                  95% CI : (0.9932, 0.9972)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9943
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9930   0.9900   0.9967
## Specificity            0.9991   0.9985   0.9978   0.9993   0.9998
## Pos Pred Value         0.9979   0.9937   0.9895   0.9962   0.9989
## Neg Pred Value         1.0000   0.9987   0.9985   0.9981   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1925   0.1731   0.1623   0.1831
## Detection Prevalence   0.2851   0.1937   0.1750   0.1629   0.1833
## Balanced Accuracy      0.9996   0.9966   0.9954   0.9947   0.9982
```

```
set.seed(1234)
modfitlda <- train(classe ~ ., method = "lda", data = trainPart)
predlda <- predict(modfitlda, newdata = testPart)
confusionMatrix(predlda, testPart$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1110  151   85   51   42
##          B   34  614  101   40  158
##          C  129  102  538   84   75
##          D  116   38  112  590   88
##          E    6   44   19   39  538
## 
## Overall Statistics
## 
##                Accuracy : 0.6913          
##                  95% CI : (0.6781, 0.7042)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
##                   Kappa : 0.6094          
## 
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7957   0.6470   0.6292   0.7338   0.5971
## Specificity            0.9062   0.9158   0.9037   0.9137   0.9730
## Pos Pred Value         0.7714   0.6484   0.5797   0.6250   0.8328
## Neg Pred Value         0.9177   0.9153   0.9203   0.9460   0.9147
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2263   0.1252   0.1097   0.1203   0.1097
## Detection Prevalence   0.2934   0.1931   0.1892   0.1925   0.1317
## Balanced Accuracy      0.8510   0.7814   0.7665   0.8237   0.7851
```

Model selection

The accuracy level of the random forest model (higher than 99%) is clearly higher than that of the LDA model (close to 70%). Therefore, the random forest model is selected.

Cross validation and expected out of sample error

The out of sample error (calculated as 1 - Accuracy Level) is below 1%, therefore very low.

Prediction on 20 test cases

```
predrf20 <- predict(modfitrf, newdata = testing, type = "class")
print(predrf20)
```

```
##  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
```

This is the prediction achieved with the selected model (random forest) for the 20 test cases.