Practical Machine Learning Assignment

Yves Görgen

8/12/2022

1. Executive summary

2. Project instructions

2.1 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).

2.2 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.

3. Set-up and Reproducability

To ensure reproducability download the required packages and set the same seed as shown below.

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(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(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.7     v purrr   0.3.4
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(rattle)
```

```
## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(kernlab)
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:purrr':
##
##     cross

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

```r
set.seed(123)

#setwd("~Data Science_Statistics and ML/Machine Learning/Course project")
har_train <- read.csv('pml-training.csv', header = T)
har_test <- read.csv('pml-testing.csv', header = T)
```

4. Data analysis

4.1 Cleaning the data

First we clean the data by applying the following steps:

- We find several numerical variables with > 90% missing values and delete them
- We remove the first variables from 'X' to 'num_window' as they provide no value for the prediction
- We remove the remaining character variables as they almost only empty cells

```
har_train %>%
  gather(col, value) %>%
  group_by(col) %>%
  dplyr::summarize(missing_share = mean(is.na(value))) %>%
  arrange(desc(missing_share)) %>%
  print(n = 5)
```

```
## # A tibble: 160 x 2
##   col                         missing_share
##   <chr>                               <dbl>
## 1 amplitude_pitch_arm                 0.979
## 2 amplitude_pitch_belt                0.979
## 3 amplitude_pitch_dumbbell            0.979
## 4 amplitude_pitch_forearm             0.979
## 5 amplitude_roll_arm                  0.979
## # ... with 155 more rows
```

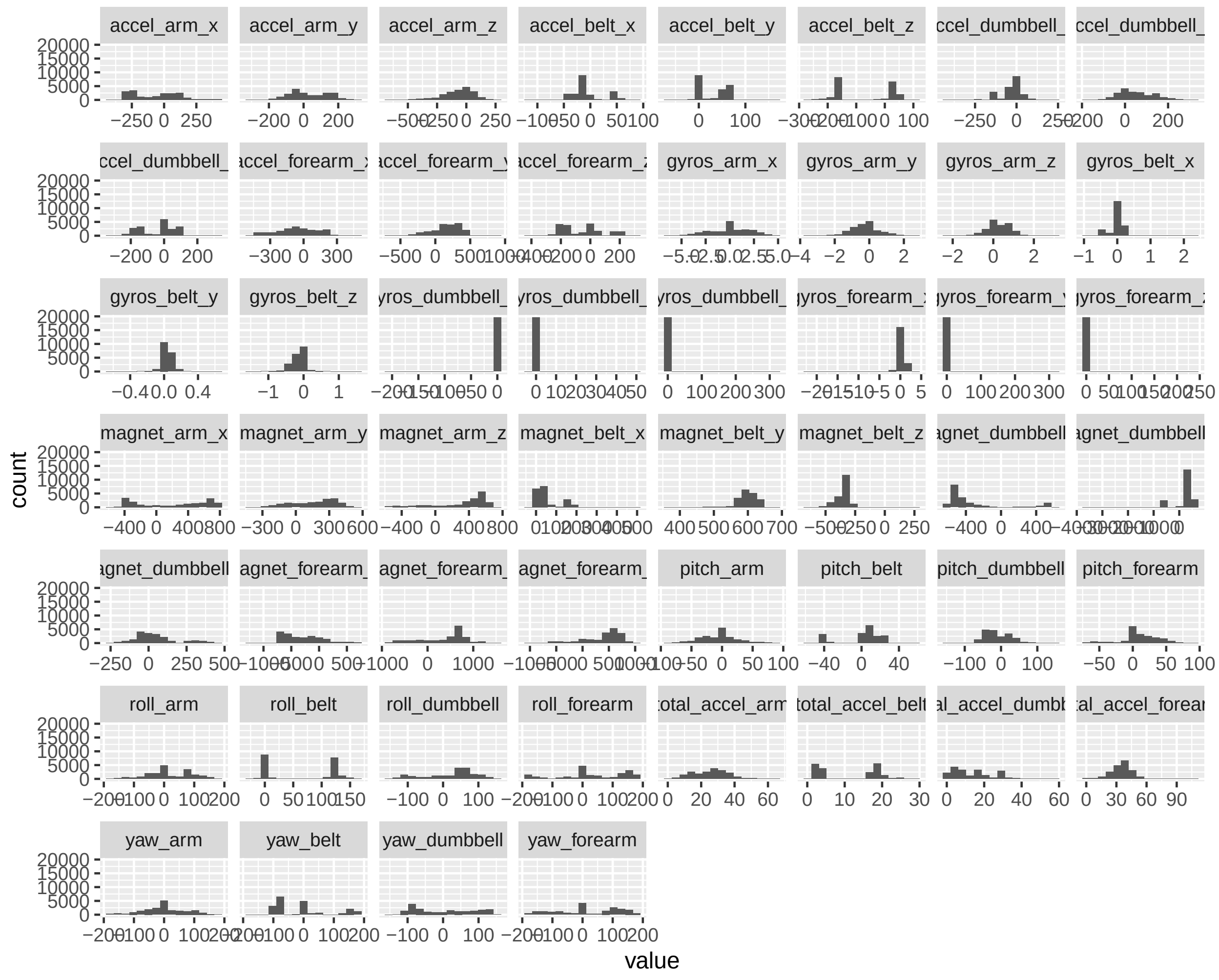
```
har_train_cl <- har_train[, colMeans(is.na(har_train)) < 0.9]
har_train_cl$classe <- as.factor(har_train_cl$classe)
har_train_cl <- har_train_cl[, !sapply(har_train_cl, is.character)]
har_train_cl <- dplyr::select(har_train_cl, -c(1:4))
```

4.2 Exploring the data

Histograms:

- We find that the variables 'gyros_dumbbell_x', 'gyros_dumbbell_y', 'gyros_dumbbell_z', 'gyros_forearm_y', 'gyros_forearm_z' are highly skewed and contain extreme outliers
- We initially keep them, but we might transform these variables dependent on the prediction results
- The remaining variables do not contain any extreme outliers

```
plot_vars <- dplyr::select(har_train_cl, -classe)
ggplot(gather(plot_vars), aes(x = value)) +
  geom_histogram(bins = 15) +
  facet_wrap(~key,scales = 'free_x')
```



Correlations

- The correlations between the classe and the explanatory variables are quite low (max: 0.34)
- However, the relationship might be non-linear

```
cormatrix <- cor(plot_vars, as.numeric(har_train_cl$classe), method = 'pearson')

cors <- tibble(Variable = rownames(cormatrix), cor = cormatrix[,1])
cors %>% arrange(desc(abs(cor))) %>% print(n = 4)
```

```
## # A tibble: 52 x 2
##   Variable         cor
##   <chr>          <dbl>
## 1 pitch_forearm  0.344
## 2 magnet_arm_x   0.296
## 3 magnet_belt_y -0.290
## 4 magnet_arm_y  -0.257
## # ... with 48 more rows
```

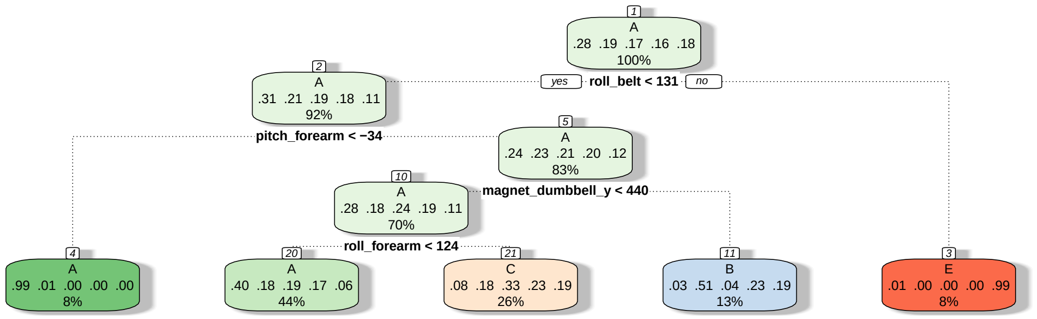
5. Model building

5.1 Create train / validation split

5.2 Build first model

As out initial model we choose the decision tree:

- Suitable model for classification
- Easy to interpret
- Better performance in non-linear settings (as we might have)

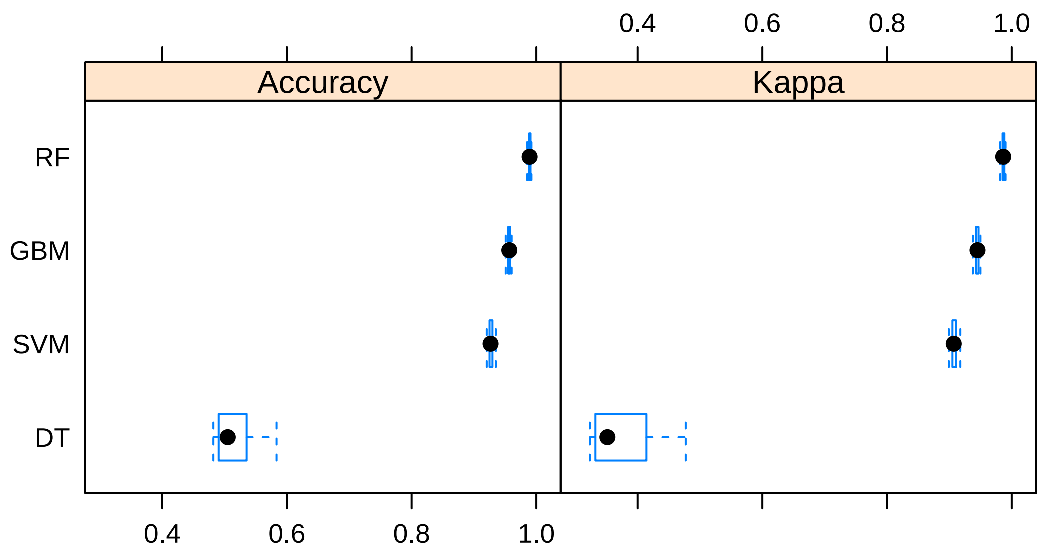


Rattle 2022−Aug−27 20:57:02 Yves.Gorgen

5.3 Build alternatives

- Since the decisions tree performs poorly, we build 3 alternative models and compare the training results
- All 3 models (Random forest (RF), Gradient boosting (GDM) and Support vector machine (SVM)) perform significantly better in terms of accuracy than the tree model
- We decide to continue with RF (accuracy of almost 1) and SVM (Fastest model with good accuracy) for the validation set

```
model_results <- resamples(list(DT = mod_tree, RF = mod_rf, GBM = mod_gbm, SVM = mod_svm))
bwplot(model_results)
```



5.4 Test on validation set

Random forest

```
options(max.print = 1000)
pred_rf <- predict(mod_rf, validation)
pred_svm <- predict(mod_svm, validation)
confusionMatrix(pred_rf, factor(validation$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    4    0    0    0
##          B    0  945    7    0    0
##          C    0    0  847    6    3
##          D    0    0    1  798    2
##          E    0    0    0    0  896
##
## Overall Statistics
##
##                Accuracy : 0.9953
##                  95% CI : (0.993, 0.997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9941
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9958   0.9906   0.9925   0.9945
## Specificity            0.9989   0.9982   0.9978   0.9993   1.0000
## Pos Pred Value         0.9971   0.9926   0.9895   0.9963   1.0000
## Neg Pred Value         1.0000   0.9990   0.9980   0.9985   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1927   0.1727   0.1627   0.1827
## Detection Prevalence   0.2853   0.1941   0.1746   0.1633   0.1827
## Balanced Accuracy      0.9994   0.9970   0.9942   0.9959   0.9972
```

Support Vector Machine

```
confusionMatrix(pred_svm, factor(validation$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1382   75    2    6    1
##          B   10  834   38    0   11
##          C    3   36  805   91   43
##          D    0    1   10  704   26
##          E    0    3    0    3  820
##
## Overall Statistics
##
##                Accuracy : 0.9268
##                  95% CI : (0.9191, 0.9339)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9072
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9907   0.8788   0.9415   0.8756   0.9101
## Specificity            0.9761   0.9851   0.9573   0.9910   0.9985
## Pos Pred Value         0.9427   0.9339   0.8231   0.9501   0.9927
## Neg Pred Value         0.9962   0.9713   0.9873   0.9760   0.9801
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2818   0.1701   0.1642   0.1436   0.1672
## Detection Prevalence   0.2989   0.1821   0.1994   0.1511   0.1684
## Balanced Accuracy      0.9834   0.9320   0.9494   0.9333   0.9543
```

6. Predicting on test data

We now disregard the timing issue and use RF for the test prediction due to the higher accuracy.

Preprocess the testing data

```
testing <- har_test[, colMeans(is.na(har_test)) < 0.9]
testing <- testing[, !sapply(testing, is.character)]
testing <- select(testing, -c(1:4))
testing <- select(testing, -problem_id)
```

Perform predictions

```
predict(mod_rf, testing)
```

```
##  [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
```