Practical Machine Learning Course Project - Classifying quality of exercise using monitor data

Aleksander Zawada

26 Sep 2015

I. Overview

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).
The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Libraries to load:
- **data.table** - a package used to extend the data.frame class and perform fast operations on it
- **dplyr** - a package used to add some extra grammar of data manipulation
- **caret** - a package which contains a set of functions that attempt to streamline the process for creating predictive models
- **corrplot** - a package which provides a graphical display of a correlation
- **rattle** - a package which provides a Gnome based interface to R functionality for data mining

```
# Load library
library(data.table)
library(dplyr)
library(caret)
library(corrplot)
library(rattle)
```

II. Download the data files

Training and testing data sets are downloaded from the http://groupware.les.inf.puc-rio.br/har website.

```
# Download the data file
data.dirname          <- "./data"
output.files.dirname  <- "./output"
pml.training.filepath <- paste0(data.dirname, "/", "pml-training.csv")
pml.testing.filepath  <- paste0(data.dirname, "/", "pml-testing.csv")

file.pml.training.url <-
    "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
```

```
file.pml.testing.url <-
    "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

if(!file.exists(data.dirname)) {
    dir.create(data.dirname)
}

if(!file.exists(output.files.dirname)) {
    dir.create(output.files.dirname)
}

if (!file.exists(pml.training.filepath)) {
    download.file(file.pml.training.url, destfile = pml.training.filepath)
}

if (!file.exists(pml.testing.filepath)) {
    download.file(file.pml.testing.url, destfile = pml.testing.filepath)
}
```

II. Load and clean the data

The training and test CSV data files are loaded. The values: "", "NA", "NULL", "#DIV/0!" are treated as missing values.

```
# Load the CSV data files
pml.training.dt <-
    data.table(
        read.csv(
            pml.training.filepath, na.strings=c("", "NA", "NULL", "#DIV/0!")))

pml.testing.dt <-
    data.table(
        read.csv(
            pml.testing.filepath, na.strings=c("", "NA", "NULL", "#DIV/0!")))

print(dim(pml.training.dt))
```

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

Remove unrelevant variables which are unlikely to be related to the predicted variable.

```
# Remove unrelevant variables - which are unlikely to be related to the
# predicted variable
pml.training.dt <- pml.training.dt %>%
    select(-user_name, -raw_timestamp_part_1, -raw_timestamp_part_2,
           -cvtd_timestamp, -X, -new_window, -num_window)

print(dim(pml.training.dt))
```

```
## [1] 19622   153
```

Remove variables which have more than 10% missing values in the training data set.

```
# Remove variables which have more than 10% missing values in the training data
# set
not.na.col.index <-
    which(
        !as.vector(
            colSums(is.na(pml.training.dt)) > 0.1*dim(pml.training.dt)[2]))

pml.training.dt <- pml.training.dt %>% select(not.na.col.index)

print(dim(pml.training.dt))
```

```
## [1] 19622    53
```

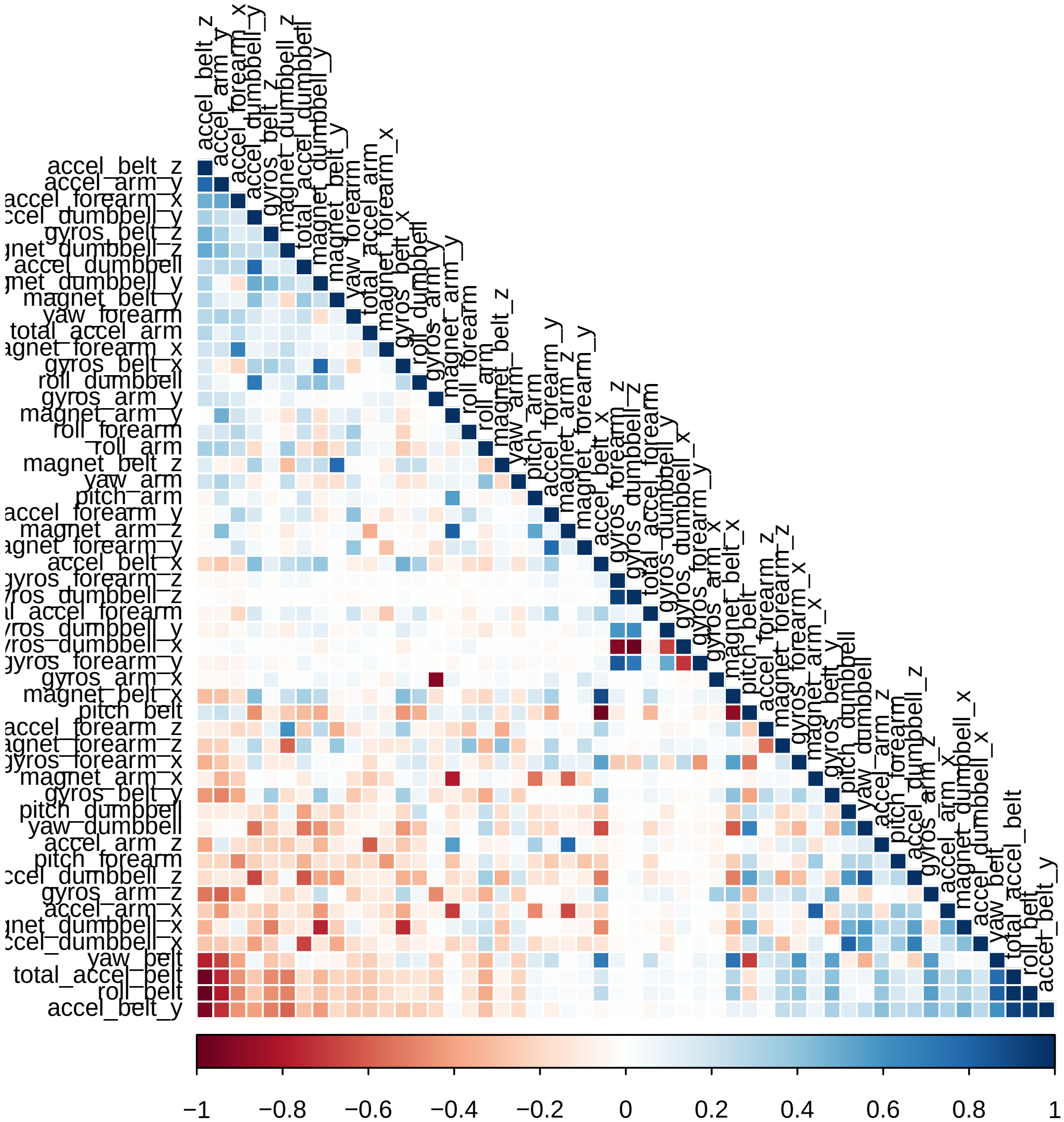
Remove variables that have very low variance in the training data set (only numeric data).

```
# Remove variables that have very low variance in the training data set
# (only numeric data)
near.zero.var <-
    nearZeroVar(pml.training.dt %>% select(-classe), saveMetrics = TRUE)

pml.training.dt <-
    pml.training.dt %>% select(which(!near.zero.var[["nzv"]]), classe)

print(dim(pml.training.dt))
```

```
## [1] 19622    53
```

Remove variables that are highly correlated to one another -> 95% (only numeric data).

```
# Remove variables that are highly correlated -> 95%
# (only numeric data)
correlation.matrix <- cor(pml.training.dt %>% select(-classe))
corrplot(correlation.matrix, order = "FPC", method = "color", type = "lower",
         tl.col = "black", tl.cex = 0.8)
```



```
corr.remove.index = findCorrelation(correlation.matrix, cutoff = .95,
                                    verbose = FALSE)

pml.training.dt <- pml.training.dt %>% select(-corr.remove.index, classe)

print(dim(pml.training.dt))
```

```
## [1] 19622    49
```

Split the training data set to training and testing part for cross validation.

```r
# Split the training data set to training and testing part for cross validation
pml.training.dt.training.index <-
    createDataPartition(y = pml.training.dt$classe, p = 0.7, list = FALSE)

pml.training.dt.training <-
    pml.training.dt %>% filter(pml.training.dt.training.index)

pml.training.dt.cv <-
    pml.training.dt %>% filter(-pml.training.dt.training.index)

print(dim(pml.training.dt.training))
```

```
## [1] 13737    49
```

```r
print(dim(pml.training.dt.cv))
```

```
## [1] 5885   49
```

III. Analysis

The following algorithms are used to predict the class variable:
- Classification Tree (rpart) - Random Forests

```r
set.seed(20150927)
```

Classification Tree

```r
# Classification Tree
classification.tree.mod.fit <-
    train(classe ~ ., data = pml.training.dt.training, method = "rpart")
```

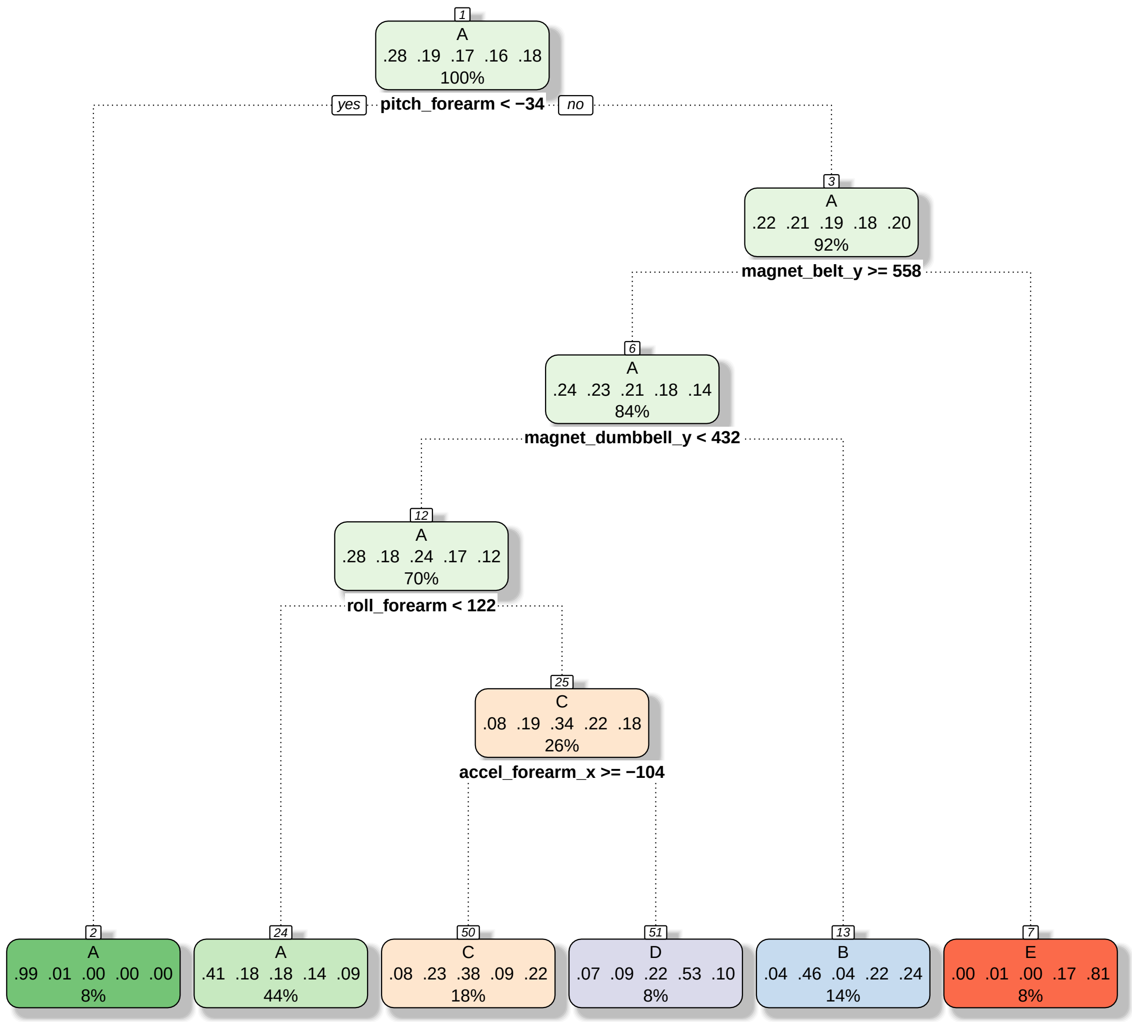
```
## Loading required package: rpart
```

```r
print(classification.tree.mod.fit)
```

```
## CART
##
## 13737 samples
##    48 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa       Accuracy SD  Kappa SD
```

```
##   0.03183806  0.5198307  0.38416010  0.03559014   0.05402609
##   0.03295697  0.5143432  0.37679771  0.03901983   0.05933401
##   0.06688028  0.3279544  0.07514525  0.08469334   0.13649378
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03183806.
```

```r
fancyRpartPlot(classification.tree.mod.fit$finalModel)
```



Rattle 2015−wrz−28 02:08:24 Aleksander.Zawada

```r
classification.tree.prediction <-
    predict(classification.tree.mod.fit, newdata = pml.training.dt.cv)
```

```r
print(
    confusionMatrix(classification.tree.prediction, pml.training.dt.cv$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1512  463  464  373  242
##          B   35  402   34  164  221
##          C  107  223  401   97  239
##          D   18   43  123  243   39
##          E    2    8    4   87  341
##
## Overall Statistics
##
##                Accuracy : 0.4926
##                  95% CI : (0.4798, 0.5055)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3375
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9032  0.35294  0.39084  0.25207  0.31516
## Specificity            0.6338  0.90434  0.86293  0.95468  0.97897
## Pos Pred Value         0.4951  0.46963  0.37582  0.52146  0.77149
## Neg Pred Value         0.9428  0.85345  0.87028  0.86695  0.86386
## Prevalence             0.2845  0.19354  0.17434  0.16381  0.18386
## Detection Rate         0.2569  0.06831  0.06814  0.04129  0.05794
## Detection Prevalence   0.5189  0.14545  0.18131  0.07918  0.07511
## Balanced Accuracy      0.7685  0.62864  0.62689  0.60338  0.64706
```

```r
classification.tree.prediction.matrix.in <-
    table(
        predict(classification.tree.mod.fit),
        pml.training.dt.training$classe)
classification.tree.in.sample.error <-
    1 - sum(diag(classification.tree.prediction.matrix.in))/
    sum(as.vector(classification.tree.prediction.matrix.in))

classification.tree.prediction.matrix.out <-
    table(classification.tree.prediction, pml.training.dt.cv$classe)
classification.tree.out.sample.error <-
    1 - sum(diag(classification.tree.prediction.matrix.out))/
    sum(as.vector(classification.tree.prediction.matrix.out))
```

The Classification Tree in-sample error is: **0.5006**.
The Classification Tree out-sample error is: **0.5074**.

Random Forests

```r
# Random Forests
random.forest.mod.fit <-
    train(classe ~ ., method = "rf", trControl = trainControl(method = "cv"),
          data = pml.training.dt.training, ntree = 150)

random.forest.prediction <-
    predict(random.forest.mod.fit, newdata = pml.training.dt.cv)

print(
    confusionMatrix(random.forest.prediction, pml.training.dt.cv$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    8    0    0    0
##          B    2 1124    5    0    0
##          C    1    7 1015   13    2
##          D    0    0    6  951    2
##          E    0    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.9922
##                  95% CI : (0.9896, 0.9943)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9901
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9868   0.9893   0.9865   0.9963
## Specificity            0.9981   0.9985   0.9953   0.9984   1.0000
## Pos Pred Value         0.9952   0.9938   0.9778   0.9917   1.0000
## Neg Pred Value         0.9993   0.9968   0.9977   0.9974   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1910   0.1725   0.1616   0.1832
## Detection Prevalence   0.2853   0.1922   0.1764   0.1630   0.1832
## Balanced Accuracy      0.9982   0.9927   0.9923   0.9924   0.9982
```

```r
varImp(random.forest.mod.fit)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 48)
##
##                      Overall
## yaw_belt              100.00
```

```
## pitch_forearm          78.00
## pitch_belt             62.61
## magnet_dumbbell_z      59.75
## roll_forearm           48.63
## magnet_dumbbell_y      47.63
## magnet_belt_y          40.67
## gyros_belt_z           28.54
## magnet_belt_z          28.08
## magnet_dumbbell_x      26.05
## roll_dumbbell          23.84
## accel_dumbbell_y       21.60
## accel_forearm_x        20.41
## total_accel_dumbbell   18.36
## accel_forearm_z        18.26
## total_accel_belt       17.64
## accel_dumbbell_z       17.26
## magnet_belt_x          17.04
## magnet_forearm_z       14.22
## yaw_arm                13.78
```

```
random.forest.prediction.matrix.in <-
    table(
        predict(random.forest.mod.fit),
        pml.training.dt.training$classe)
random.forest.in.sample.error <-
    1 - sum(diag(random.forest.prediction.matrix.in))/
    sum(as.vector(random.forest.prediction.matrix.in))

random.forest.prediction.matrix.out <-
    table(random.forest.prediction, pml.training.dt.cv$classe)
random.forest.out.sample.error <-
    1 - sum(diag(random.forest.prediction.matrix.out))/
    sum(as.vector(random.forest.prediction.matrix.out))
```

The Random Forests in-sample error is: **0.0000**.
The Random Forests out-sample error is: **0.0078**.

IV. Final prediction and conclusion

The Random Forest algorithm was chosen as a final algorithm for prediction the *classe* variable as it has very high accuracy and very low out-sample error.

```
final.prediction <-
    predict(random.forest.mod.fit, newdata = pml.testing.dt)

print(final.prediction)
```

```
##  [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
```

IV. Generate output

PML write files function.

```r
pml_write_files = function(x){
    n = length(x)
    for(i in 1:n){
        filename = paste0(output.files.dirname, "/", "problem_id_", i, ".txt")
        write.table(x[i],file=filename,quote=FALSE,
                    row.names=FALSE,col.names=FALSE)
    }
}
```

Generate output.

```r
pml_write_files(final.prediction)
```