ML - Weight Lifting Analysis

Myriam Ragni - 31 Jan. 2020

Executive Summary

The goal of this assignment is to predict the manner in which participants to a Human Activity Recognition study did weight lifting exercises based on data collected by arm/belt/forearm and dumbbell sensors. Barbell lifts were performed in five different fashions ('classe' variable in the training dataset):
- Class A: According to the specification (the only correct way of performing barbell lifts)
- Class B: Throwing the elbows to the front
- Class C: Lifting the dumbbell only halfway
- Class D: Lowering the dumbbell only halfway
- Class E: Throwing the hips to the front

Thanks to the autors (Ugulino,W.; Cardador,D.; Vega,K.; Velloso,E.; Milidiu, R.; Fuks, H.) of the publication WearableComputing: Accelerometers' Data Classification of Body Posturesand Movements for generously allowing the data collected to be used for this project.

We are provided with:
- a 'training' dataset used for the selection of a machine learning algorithm and for validation
- a 'test' dataset with 20 test cases on which we will apply the chosen model to predict the way the exercice was done.

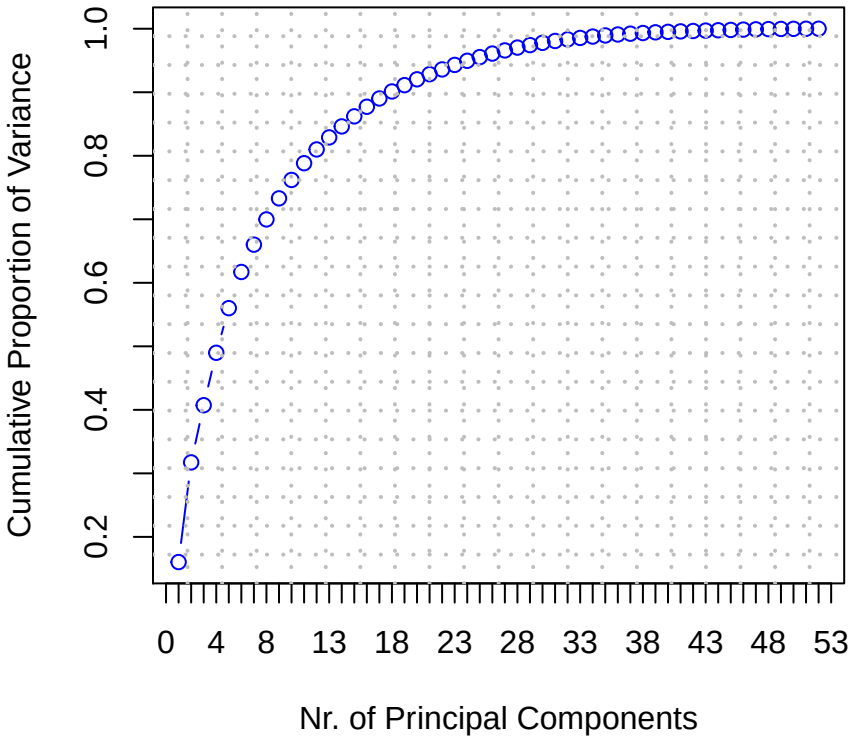
I've used the following approach:
- Based on the reults of the basic exploratory data analysis, I decided to remove data not relevant for the analysis (user information and time stamps, variables with more than 95% of NA or blank values) but to keep near zero variance predictors.
- For cross-validation purposes, I split further the training data (following the 75/25 pattern) into an explicit training dataset used to train the prediction model and a validation set to evaluate the performance of the selected model.
- I used Principal Component Analysis to further reduce the dimensionality of data.
- I chose 3 ML algorithms (Classification Tree, Random Forest and Gradient Boosting Tree) and repeated cross validation with 10 folds and 3 repeats to build the models with the training data and compared the competing models using acuracy as evaluation metric.
- I picked the model with the best accuracy (Random Forest) and assessed the performance of the RF model with the Validation dataset. The expected accuracy rate (resp. out of sample error) is high enough to validate the model.
- Finally, the RF model is used to predict the class for the 20 test cases.

Basic Exploratory Data Analysis and Data Cleanup

The 'training' data set consists of 19622 observations and 160 variables. The first seven variables provide user and time information, which is irrelevant for our problematic and are therefore excluded from the analysis. The analysis also showed an important number of variables with almost only NAs or blank values. Hence, variables with more than 95% of NA or blank values are taken out of the scope. This brings the number of variables in scope to 53.

Data Partitioning & Preprocessing with PCA

The cleaned training set was split into a pure training dataset (75% of the data) and a validation data set (25%) which will be used to assess the accuracy of the prediction model.
I also used PCA to further reduce the dimensionality of data, while keeping as much variation as possible. The below plot shows that about 43 components results in variance close to 100%.



Therefore, I selected the components PC1 to PC43 to proceed with the next steps. The same transformation is applied to the Training, Validate and Test datasets. The dimensions of these datasets are respectively:

```
## [1] "Dimension Training Dataset = 14718 44"
```

```
## [1] "Dimension Validate Dataset = 4904 44"
```

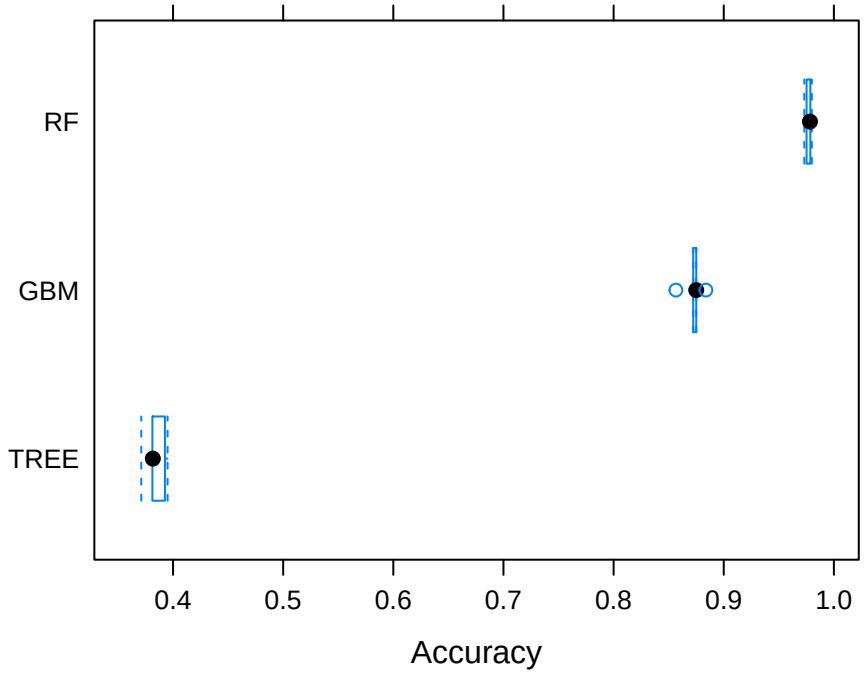
```
## [1] "Dimension Test Dataset = 20 43"
```

Model Selection

From the 3 competing ML algorithms (Classification Tree, Random Forest and Gradient Boosting Tree), the Random Forest model provides the highest accuracy as shown on the below summary and boxplot.

```
##
## Call:
## summary.resamples(object = results)
##
## Models: TREE, RF, GBM
## Number of resamples: 5
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## TREE 0.3711761 0.3812436 0.3816638 0.3843575 0.3926630 0.3950408    0
## RF   0.9731475 0.9751954 0.9782609 0.9770341 0.9786005 0.9799660    0
## GBM  0.8565602 0.8722826 0.8750000 0.8725355 0.8750424 0.8837920    0
##
## Kappa
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## TREE 0.1548114 0.1676348 0.1705718 0.1786380 0.1858852 0.2142870    0
## RF   0.9660150 0.9686132 0.9724861 0.9709320 0.9729039 0.9746419    0
```

```
## GBM  0.8181965 0.8382134 0.8414535 0.8384292 0.8415733 0.8527096    0
```



Model Assessment (Out of sample error)

The model providing the best accuracy (Random Forest) (with an acceptable level) is now used to verify the performance of the algorith with the Validation dataset, using the confusion matrix.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1389    4    0    1    1
##          B   17  925    4    0    3
##          C    0   11  837    7    0
##          D    0    0   31  769    4
##          E    0    0    2    1  898
##
## Overall Statistics
##
##                Accuracy : 0.9825
##                  95% CI : (0.9784, 0.9859)
##     No Information Rate : 0.2867
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9778
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9879   0.9840   0.9577   0.9884   0.9912
## Specificity            0.9983   0.9939   0.9955   0.9915   0.9992
## Pos Pred Value         0.9957   0.9747   0.9789   0.9565   0.9967
## Neg Pred Value         0.9952   0.9962   0.9909   0.9978   0.9980
## Prevalence             0.2867   0.1917   0.1782   0.1586   0.1847
## Detection Rate         0.2832   0.1886   0.1707   0.1568   0.1831
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9931   0.9890   0.9766   0.9900   0.9952
```

The calculated expected accuracy rate is **98.25%**, which gives an out of sample error rate of **1.75%**.

Prediction

Finally, as the out of sample error rate is acceptable, the selected model is used to predict a class for each of the 20 observations contained in the provided Testing dataset. See results below:

```
##  [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
```

```
## A B C D E
## 7 8 1 1 3
```

Appendix

Preparing the environment

```
rm(list = ls())
Sys.setlocale("LC_TIME", "English")
suppressWarnings(library(caret))
suppressWarnings(library(doParallel))
```

```
SrcFileURL1 <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
SrcFileURL2 <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
pml_training <- "./pml-training.csv"
pml_testing <- "./pml-testing.csv"
#### Check if data files were already downloaded, if not,
#### download the files
if (!file.exists(pml_training)) {
    download.file(SrcFileURL1, destfile = pml_training)
}
if (!file.exists(pml_testing)) {
    download.file(SrcFileURL2, destfile = pml_testing)
}
DF_Train <- read.csv(pml_training, header = TRUE, sep = ",")  #### raw data training data
DF_Test <- read.csv(pml_testing, header = TRUE, sep = ",")  #### raw data testing data (for prediction)
```

Basic Exploratory Data Analysis and Data Cleanup

```
dim(DF_Train)  #### 19622 obs. 160 variables
```

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

```
dim(DF_Test)  #### 20 obs. 160 variables
```

```
## [1]  20 160
```

```
table(DF_Train$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
#### Removing variables containing obvious user information and
#### timestamps
DF_TrainTidy <- DF_Train[, -c(1:7)]
DF_TestTidy <- DF_Test[, -c(1:7)]
```

```
#### Removing predictors with 95% NA or blank values
NACounts <- apply(is.na(DF_Train), MARGIN = 2, sum)
NACounts_Not0 <- NACounts[NACounts != 0]  #### All columns with at least one NA --> 67 variables wih 19
NAPerc <- sapply(DF_TrainTidy, function(x) mean(is.na(x)) | sum(x ==
    "")) > 0.95  #### Returns TRUE or FALSE
DF_TrainTidy <- DF_TrainTidy[, NAPerc == FALSE]  #### 53 predictors left
DF_TestTidy <- DF_TestTidy[, NAPerc == FALSE]
```

Data Partitioning & Preprocessing with PCA

```
set.seed(8484)
TrainIndex <- createDataPartition(DF_TrainTidy$classe, p = 0.75,
    list = FALSE)
Train <- DF_TrainTidy[TrainIndex, ]
Validate <- DF_TrainTidy[-TrainIndex, ]
print(paste0("Dimension Training Dataset = ", dim(Train)[1],
    " ", dim(Train)[2]))  #### 14718 obs. 53 variables
```

```
## [1] "Dimension Training Dataset = 14718 53"
```

```
print(paste0("Dimension Validation Dataset = ", dim(Validate)[1],
    " ", dim(Validate)[2]))  #### 4904 obs. 53 variables
```

```
## [1] "Dimension Validation Dataset = 4904 53"
```

```
#### Performs PCA, centers the variable to have mean equals to
#### zero, normalizes the variables to have standard deviation
#### equal to 1
prin_comp <- prcomp(Train[, -53], scale. = T)  #### exluding the 'classe' variable
names(prin_comp)
std_dev <- prin_comp$sdev
pr_var <- std_dev^2
prop_varex <- pr_var/sum(pr_var)
par(mar = c(4, 5, 1, 2))
plot(cumsum(prop_varex), xlab = "Nr. of Principal Components",
    ylab = "Cumulative Proportion of Variance", type = "b", col = "blue",
    xaxp = c(0, 53, 53), yaxp = c(0, 1, 10))
grid(nx = 20, ny = 20, lwd = 2, col = "grey")
```

```
#### Set up x and y to avoid slowness of caret() with model
#### syntax
y_Train <- Train[, 53]
x_Train <- prin_comp$x[, 1:43]
Train <- as.data.frame(cbind(x_Train, y_Train))
#### Transform validate and test datasets into PCA (same
#### transformation to the test set as we did to training set,
#### including the center and scaling feature)
y_Validate <- Validate[, 53]  #### variable 'classe'
x_Validate <- as.data.frame(predict(prin_comp, newdata = Validate))
x_Validate <- x_Validate[1:43]
Validate <- as.data.frame(cbind(x_Validate, y_Validate))
names(Validate)[44] <- "classe"
Test <- as.data.frame(predict(prin_comp, newdata = DF_TestTidy))
Test <- Test[1:43]
print(paste0("Dimension Training Dataset = ", dim(Train)[1],
    " ", dim(Train)[2]))  #### 14718 obs. 44 variables
print(paste0("Dimension Validate Dataset = ", dim(Validate)[1],
    " ", dim(Validate)[2]))  #### 4904 obs. 44 variables
print(paste0("Dimension Test Dataset = ", dim(Test)[1], " ",
    dim(Test)[2]))  #### 20 obs. 43 variables
```

Model Selection

```
registerDoParallel()
#### Prepare training scheme
```

```
Control <- trainControl(method = "cv", number = 5, allowParallel = TRUE)
#### Fit the models
set.seed(8484)
mdl1 <- train(x_Train, y_Train, method = "rpart", trControl = Control)
set.seed(8484)
mdl2 <- train(x_Train, y_Train, method = "rf", trControl = Control)
set.seed(8484)
mdl3 <- train(x_Train, y_Train, method = "gbm", trControl = Control,
    verbose = FALSE)
```

```
#### Collect the resampling results
results <- resamples(list(TREE = mdl1, RF = mdl2, GBM = mdl3))
summary(results)
```

```
#### Plot the Accuracy metric by model
par(mar = c(2, 2, 1, 2))
bwplot(results, metric = "Accuracy")
```

Model Assessment

```
trainpred <- predict(mdl2, newdata = Validate)
confMat <- confusionMatrix(y_Validate, trainpred)
confMat
#### true accuracy of the predicted model
accuracy <- sum(trainpred == Validate$classe)/length(trainpred)
#### out of sample error (in %)
outOfSampleError <- round((1 - accuracy) * 100, 2)
```

Prediction

```
testpred <- predict(mdl2, newdata = Test)
testpred
summary(testpred)
```