Practical machine learning

Javier Monedero

Backgrownd

The data for this assignment can be found in http://groupware.les.inf.puc-rio.br/har. (see the section on the Weight Lifting Exercise Dataset), and explicitly from Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

It is necessary to determine how well barbell are performed by subjects equiped with accelerometers on the belt, arm, forearm, and dumbells. For this purpose, I build a predictive model in which the exercise of barbell lifting is performed correctly or not.

Packages

First of all, specific packages should be loaded in order to use them.

```
library(caret)
```

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

```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(Hmisc)
```

```
## Loading required package: grid
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: Formula
##
## Attaching package: 'Hmisc'
##
## The following object is masked from 'package:randomForest':
##
##     combine
##
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

```
library(corrplot)
```

Data

Then, data should be readed to overview potential problems. In this case, the training and testing datasets present missing values named as NA or directly as blank.

```
train <- read.csv("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", header = TRUE,
test <- read.csv("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", header = TRUE, na
```

After typing dim(train), I saw that the train dataset is made from 19622 observations of 160 variables. As mentioned, removing NAs is a must to avoid problems.

```
Train<- train[,colSums(is.na(train))==0]
Test<- test[,colSums(is.na(test))==0]
dim(Train)
```

```
## [1] 19622    60
```

```
head(Train)
```

```
##   X user_name raw_timestamp_part_1 raw_timestamp_part_2   cvtd_timestamp
## 1 1  carlitos           1323084231               788290 05/12/2011 11:23
## 2 2  carlitos           1323084231               808298 05/12/2011 11:23
## 3 3  carlitos           1323084231               820366 05/12/2011 11:23
## 4 4  carlitos           1323084232               120339 05/12/2011 11:23
## 5 5  carlitos           1323084232               196328 05/12/2011 11:23
## 6 6  carlitos           1323084232               304277 05/12/2011 11:23
##   new_window num_window roll_belt pitch_belt yaw_belt total_accel_belt
## 1         no         11      1.41       8.07    -94.4                3
## 2         no         11      1.41       8.07    -94.4                3
## 3         no         11      1.42       8.07    -94.4                3
## 4         no         12      1.48       8.05    -94.4                3
## 5         no         12      1.48       8.07    -94.4                3
## 6         no         12      1.45       8.06    -94.4                3
##   gyros_belt_x gyros_belt_y gyros_belt_z accel_belt_x accel_belt_y
## 1         0.00         0.00        -0.02          -21            4
## 2         0.02         0.00        -0.02          -22            4
## 3         0.00         0.00        -0.02          -20            5
## 4         0.02         0.00        -0.03          -22            3
## 5         0.02         0.02        -0.02          -21            2
## 6         0.02         0.00        -0.02          -21            4
##   accel_belt_z magnet_belt_x magnet_belt_y magnet_belt_z roll_arm
## 1           22            -3           599          -313     -128
## 2           22            -7           608          -311     -128
## 3           23            -2           600          -305     -128
## 4           21            -6           604          -310     -128
## 5           24            -6           600          -302     -128
## 6           21             0           603          -312     -128
##   pitch_arm yaw_arm total_accel_arm gyros_arm_x gyros_arm_y gyros_arm_z
## 1      22.5    -161              34        0.00        0.00       -0.02
```

```
## 2      22.5    -161              34        0.02       -0.02       -0.02
## 3      22.5    -161              34        0.02       -0.02       -0.02
## 4      22.1    -161              34        0.02       -0.03        0.02
## 5      22.1    -161              34        0.00       -0.03        0.00
## 6      22.0    -161              34        0.02       -0.03        0.00
##   accel_arm_x accel_arm_y accel_arm_z magnet_arm_x magnet_arm_y
## 1        -288         109        -123         -368          337
## 2        -290         110        -125         -369          337
## 3        -289         110        -126         -368          344
## 4        -289         111        -123         -372          344
## 5        -289         111        -123         -374          337
## 6        -289         111        -122         -369          342
##   magnet_arm_z roll_dumbbell pitch_dumbbell yaw_dumbbell
## 1          516      13.05217      -70.49400    -84.87394
## 2          513      13.13074      -70.63751    -84.71065
## 3          513      12.85075      -70.27812    -85.14078
## 4          512      13.43120      -70.39379    -84.87363
## 5          506      13.37872      -70.42856    -84.85306
## 6          513      13.38246      -70.81759    -84.46500
##   total_accel_dumbbell gyros_dumbbell_x gyros_dumbbell_y gyros_dumbbell_z
## 1                   37                0            -0.02             0.00
## 2                   37                0            -0.02             0.00
## 3                   37                0            -0.02             0.00
## 4                   37                0            -0.02            -0.02
## 5                   37                0            -0.02             0.00
## 6                   37                0            -0.02             0.00
##   accel_dumbbell_x accel_dumbbell_y accel_dumbbell_z magnet_dumbbell_x
## 1             -234               47             -271              -559
## 2             -233               47             -269              -555
## 3             -232               46             -270              -561
## 4             -232               48             -269              -552
## 5             -233               48             -270              -554
## 6             -234               48             -269              -558
##   magnet_dumbbell_y magnet_dumbbell_z roll_forearm pitch_forearm
## 1               293               -65         28.4         -63.9
## 2               296               -64         28.3         -63.9
## 3               298               -63         28.3         -63.9
## 4               303               -60         28.1         -63.9
## 5               292               -68         28.0         -63.9
## 6               294               -66         27.9         -63.9
##   yaw_forearm total_accel_forearm gyros_forearm_x gyros_forearm_y
## 1        -153                 36            0.03            0.00
## 2        -153                 36            0.02            0.00
## 3        -152                 36            0.03           -0.02
## 4        -152                 36            0.02           -0.02
## 5        -152                 36            0.02            0.00
## 6        -152                 36            0.02           -0.02
##   gyros_forearm_z accel_forearm_x accel_forearm_y accel_forearm_z
## 1           -0.02             192             203            -215
## 2           -0.02             192             203            -216
## 3            0.00             196             204            -213
## 4            0.00             189             206            -214
## 5           -0.02             189             206            -214
## 6           -0.03             193             203            -215
```

```
##   magnet_forearm_x magnet_forearm_y magnet_forearm_z classe
## 1              -17              654              476      A
## 2              -18              661              473      A
## 3              -18              658              469      A
## 4              -16              658              469      A
## 5              -17              655              473      A
## 6               -9              660              478      A
```

Now, more than half of the original columns have been removed to facilitate calculations. But I still made more data manipulations since some variables did not represent data from accelerometer measures or participant's data. These variables include: X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window and num_window.

```
deleteColumns <- grepl("X|user_name|new_window|num_window|raw_timestamp_part_1|raw_timestamp_part_2|cvt
OKtrain <- Train[, !deleteColumns]
OKtest <- Test[, !deleteColumns]
```

Finally, I had 53 columns in both datasets.

Spliting the data

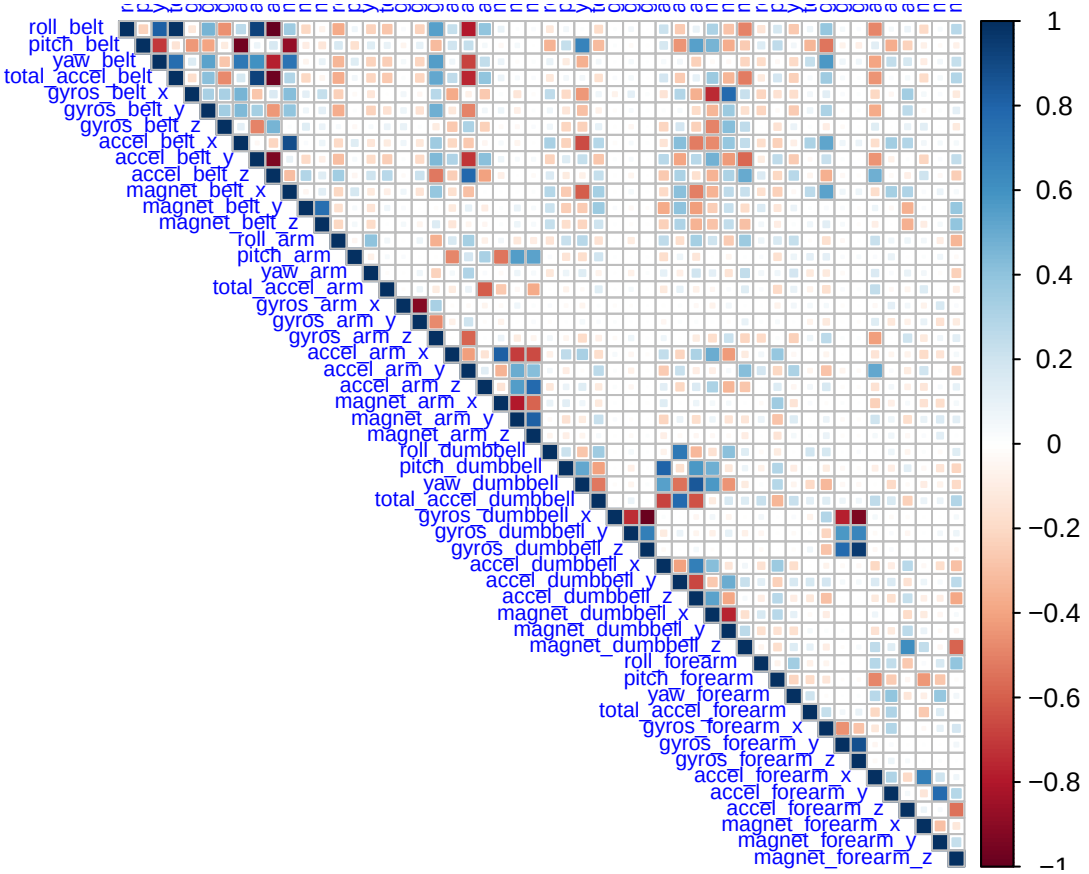
After done these modifications, I split the OKtraining set into a training set with the 70% of the observations, and a validation set with the 30% of the observations. This way, I was able to develop cross validation in this model.

```
splitTrain = createDataPartition(y = OKtrain$classe, p = 0.7, list = FALSE)
trainTrain = OKtrain[splitTrain, ]
trainValidation = OKtrain[-splitTrain, ]
```

Correlations between variables

Looking at the correlations between the variables in the datasets, it is possible to remove highly correlated predictors from this analysis and replace them with weighted combinations of predictors. As the target of the project is to predict the way participants did the exercises, I saw that this information is stored in the variable classe in the training data set. In addition, I plotted the correlation between pairs of predictors.

```
correlMatrix <- cor(trainTrain[, -53])
corrplot(correlMatrix, order = "original", method = "square", type = "upper", tl.cex= 0.7, tl.col = rgb
```



In a few words, the darker blue and red squares represent high positive and high negative correlations, respectively. Addionally, some pairs of variables are highly correlated. I achieved this with the following instructions:

```
which(correlMatrix > 0.98 & correlMatrix != 1)
```

```
## [1]   4 157
```

```
correlMatrix[which(correlMatrix > 0.98 & correlMatrix != 1)]
```

```
## [1] 0.9806696 0.9806696
```

```
correlMatrix[which(correlMatrix < -0.98 )]
```

```
## [1] -0.9920293 -0.9920293 -0.9849312 -0.9849312
```

Predictive model: Approach 1

For this project, I carried out two approaches. But first it is necessary to pre-process the data with a principal component analysis. I focused on the key column "classe" which I wanted to predict. Next, using the predict function on the pre-processed training and validation data, allowed preparing the data to the optimal conditions.

```
preProcTrain <- preProcess(trainTrain[, -53], method = "pca", thresh = 0.99)
trainPredict <- predict(preProcTrain, trainTrain[, -53])
validationTestPC <- predict(preProcTrain, trainValidation[, -53])
```

I used as first approach the randomForest model function on the whole dataset prior splitting it (OKtrain dataset). After trying several number of trees and taking into account the dimensions of the data OKtrain, I determined that a good ntree parameter is 1050.

```
approach1 <- randomForest(classe ~ ., data = OKtrain, ntree = 1050)
approach1
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = OKtrain, ntree = 1050)
##                Type of random forest: classification
##                      Number of trees: 1050
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.29%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5577    2    0    0    1 0.0005376344
## B    9 3785    3    0    0 0.0031603898
## C    0   10 3410    2    0 0.0035067212
## D    0    0   21 3193    2 0.0071517413
## E    0    0    2    4 3601 0.0016634322
```

It gave an out-of-bag (OOB) estimate of error rate: 0.29%, which is pretty acceptable and an accuracy close to 100%. Besides, due to the fact that the confusion matrix looks good, the approach fitted the training dataset well.

Next, I measured the the variable importance order estimate obtained from the classifier training algorithm.

```
varImp<-varImp(approach1)
varImp$variables<-row.names(varImp)
varImp[order(varImp$Overall,decreasing=T),]
```

```
##                        Overall           variables
## roll_belt            1296.07852           roll_belt
## yaw_belt              915.85766            yaw_belt
## pitch_forearm         786.11081       pitch_forearm
## magnet_dumbbell_z     755.10843   magnet_dumbbell_z
## pitch_belt            701.52434          pitch_belt
## magnet_dumbbell_y     692.72724   magnet_dumbbell_y
## roll_forearm          623.08377        roll_forearm
## magnet_dumbbell_x     474.63376   magnet_dumbbell_x
## roll_dumbbell         425.72741       roll_dumbbell
## accel_dumbbell_y      412.84789    accel_dumbbell_y
## magnet_belt_z         387.47909       magnet_belt_z
## accel_belt_z          387.41517        accel_belt_z
## magnet_belt_y         381.33960       magnet_belt_y
## accel_dumbbell_z      336.98561    accel_dumbbell_z
```

```
## roll_arm              327.82750            roll_arm
## accel_forearm_x       321.48128     accel_forearm_x
## gyros_belt_z          296.61773        gyros_belt_z
## magnet_forearm_z      289.26730    magnet_forearm_z
## total_accel_dumbbell  261.43265 total_accel_dumbbell
## yaw_dumbbell          252.98313        yaw_dumbbell
## yaw_arm               251.58978             yaw_arm
## accel_forearm_z       251.37340     accel_forearm_z
## gyros_dumbbell_y      250.82853    gyros_dumbbell_y
## magnet_belt_x         248.44578       magnet_belt_x
## magnet_arm_x          242.53216        magnet_arm_x
## accel_dumbbell_x      241.97154    accel_dumbbell_x
## accel_arm_x           238.83433         accel_arm_x
## magnet_arm_y          227.40280        magnet_arm_y
## total_accel_belt      223.16925    total_accel_belt
## magnet_forearm_y      221.27667    magnet_forearm_y
## magnet_forearm_x      215.52291    magnet_forearm_x
## magnet_arm_z          181.36160        magnet_arm_z
## yaw_forearm           173.21193         yaw_forearm
## pitch_arm             172.50749           pitch_arm
## pitch_dumbbell        171.66375      pitch_dumbbell
## accel_arm_y           142.93316         accel_arm_y
## accel_forearm_y       138.82748     accel_forearm_y
## gyros_arm_y           137.37220         gyros_arm_y
## accel_arm_z           133.06047         accel_arm_z
## gyros_arm_x           128.54939         gyros_arm_x
## gyros_dumbbell_x      126.48729    gyros_dumbbell_x
## accel_belt_y          125.32093        accel_belt_y
## gyros_forearm_y       123.83881     gyros_forearm_y
## gyros_belt_y          112.30038        gyros_belt_y
## accel_belt_x          111.96641        accel_belt_x
## total_accel_forearm   109.45213  total_accel_forearm
## total_accel_arm        99.81901     total_accel_arm
## gyros_belt_x           95.07248        gyros_belt_x
## gyros_forearm_z        81.41754     gyros_forearm_z
## gyros_dumbbell_z       80.18165    gyros_dumbbell_z
## gyros_forearm_x        73.00594     gyros_forearm_x
## gyros_arm_z            54.59102         gyros_arm_z
```

As can be seen, roll_belt and yaw_belt variables presented the most overall importance, indicating the algorithm employed made good use of the provided predictors.

Predictive model: Approach 2

In the second approach to the problem, I changed the method to be used on the trainTrain data set, just for trying to get even less OBB and better accuracy. In this case, I selected the train function with the rf cross validation method from the extense list of them. I selected the random forest method due to the high number of observations in comparison with the number of predictors. Note that this approach took several minutes.

```
approach2 <- train(trainTrain$classe ~ ., method = "rf", data = trainPredict, trControl = trainControl(
approach2$finalModel
```

```
##
```

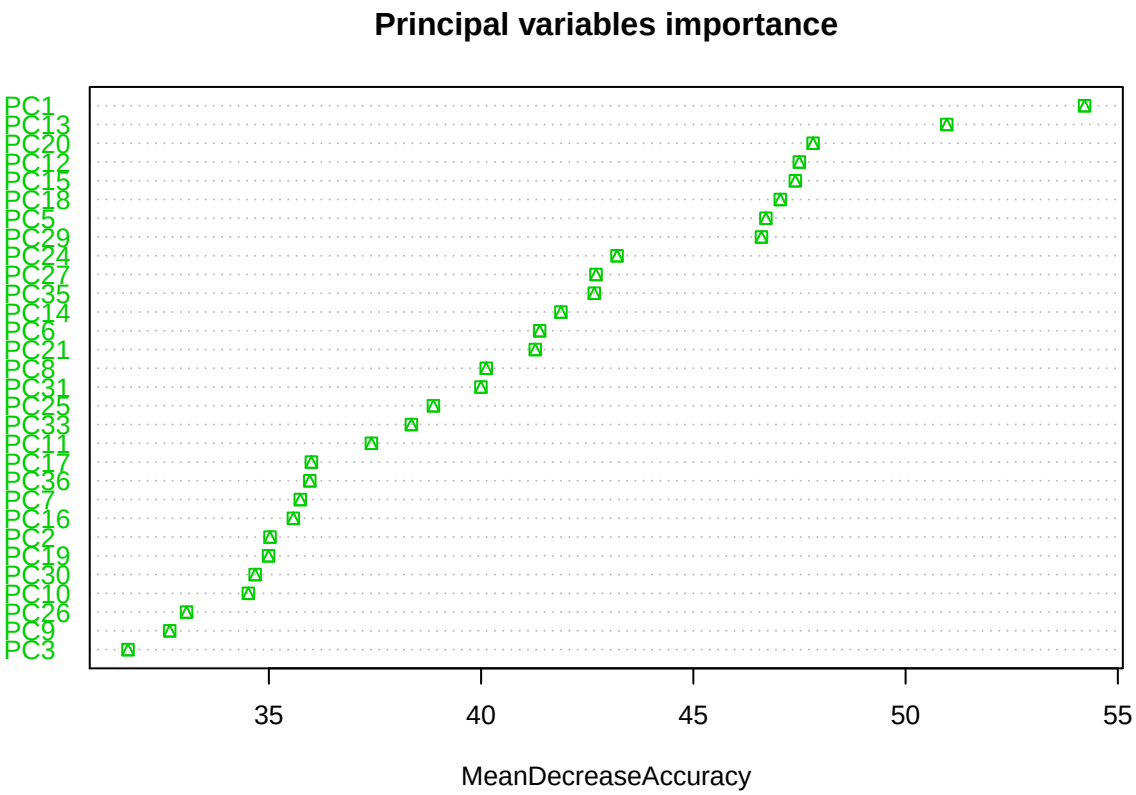
```
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 2.07%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3894    5    4    2    1 0.003072197
## B   54 2576   26    0    2 0.030850263
## C    4   33 2342   13    4 0.022537563
## D    2    0   91 2151    8 0.044849023
## E    0    7   20    9 2489 0.014257426
```

This method showed an 00B error rate of 1.98%.

Continuing with this approach, I measured the relative importance in the resulting principal components of the trained model approach2.

Now can review the relative importance of the resulting principal components of the trained model, approach2.

```
varImpPlot(approach2$finalModel, sort = TRUE, type = 1, pch = 14, col = 3, cex = .8, main = "Principal
```



Here, the degree of importance was shown on the x-axis increasing from left to right, whereas the principal variables were represented in the y-axis.

Cross validation testing and out-of-sample error estimation

Now the predict funcion was used on the trained model to be applied to the cross validation test dataset.

```
predictionTrain <- predict(approach2, validationTestPC)
confusionMatrix(trainValidation$classe, predictionTrain)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669    3    1    1    0
##          B   16 1110   13    0    0
##          C    1   10 1008    7    0
##          D    1    1   33  926    3
##          E    0    2    4    2 1074
##
## Overall Statistics
##
##                Accuracy : 0.9833
##                  95% CI : (0.9797, 0.9865)
##     No Information Rate : 0.2867
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9789
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9893   0.9858   0.9518   0.9893   0.9972
## Specificity            0.9988   0.9939   0.9963   0.9923   0.9983
## Pos Pred Value         0.9970   0.9745   0.9825   0.9606   0.9926
## Neg Pred Value         0.9957   0.9966   0.9895   0.9980   0.9994
## Prevalence             0.2867   0.1913   0.1799   0.1590   0.1830
## Detection Rate         0.2836   0.1886   0.1713   0.1573   0.1825
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9941   0.9898   0.9741   0.9908   0.9978
```

R showed an accuracy of 98.13% indicating this is a good method. The estimated out-of-sample error was 1.87%.

Predicted Results

```
testPC<-predict(preProcTrain, OKtest[,-53])
finalPrediction<-predict(approach2, testPC)
finalPrediction
```

```
##  [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
```

Conclusion

This second approach reached a 90% accuracy on the testing set provided. Although the accuracy is high, I used the first method for the 20 test cases since the accuracy in this case is higher, then obtaining better

results.