Using CART, Boosting and Random Forests to predict the *classe*-outcome in the dataset *Wearable Computing: Accelerometers Data Classification of Body Postures and Movements*

Practical Machine Learning Course, Peer Graded Assisgnment

Luis David Bedon Gomez

27 9 2017

1. Executive summary

The following report describes three machine learning models to predict the way in which exercises were done after the data collected by Ugulino et al. in "Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements".

Ugulino et al. consider 5 activity classes, gathered from 4 subjects wearing accelerometers mounted on four different parts of the body and provide a public domain dataset comprising 165,633 samples.

This dataset was loaded. After an exploratory data analysis, 52 covariates were selected to predict the activity classes described in the dataset.

A random-tree-model, a random-forest-model and a boosting model were applied strategically changing parameters and comparing the testing accuracy.

The random-forest-model and the boosting model trained over all 53 variables showed a very well prediction accuracy of 99,6%.

Based on this model, the predictions asked in the course were done.

2. Data Processing

2.1 Loading the Data

The datasets were loaded from the given URLs and imported in R using *read.csv*(). In this analysis, the libraries *caret* and *gbm* were used.

```
library(caret)
library(gbm)
library(knitr)
#library(ggplot2)
#library(dplyr)

# Download the data
setwd("~/DataScienceSpecialization/08PracticalMachineLearning")
if(file.exists("pml-training.csv")==FALSE){
  urlTrain<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
  urlTest<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
  download.file(urlTrain,"pml-training.csv")
```

```
  download.file(urlTest,"pml-testing.csv")
}

# Load the data
pml_training<-read.csv("pml-training.csv",stringsAsFactors = FALSE)
pml_testing<-read.csv("pml-testing.csv",stringsAsFactors = FALSE)
```

2.2 Exploratory Data Analysis

Examinating the data

The imported files consist in an training dataset *pml_training* with 19622 rows and 160 columns, and a quiz dataset *pml_testing* with 20 rows and the same number of columns.

	pml_training	pml_testing
No. of Measurements	19622	20
No. of Covariates	160	160

In order to avoid any type of overfitting, the dataset for the quiz will be not inspected.

The structure of *pml_training* shows diverse types of columns, mostly of class *int*, *num*, but also *chr*, some of them with a considerable amount of NA's:

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 ...
##  $ user_name               : chr  "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 ...
##  $ cvtd_timestamp          : chr  "05/12/2011 11:23" "05/12/2011 11:23" ...
##  $ new_window              : chr  "no" "no" ...
##  $ num_window              : int  11 11 11 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : chr  "" "" ...
##  $ kurtosis_picth_belt     : chr  "" "" ...
##  $ kurtosis_yaw_belt       : chr  "" "" ...
##  $ skewness_roll_belt      : chr  "" "" ...
##  $ skewness_roll_belt.1    : chr  "" "" ...
##  $ skewness_yaw_belt       : chr  "" "" ...
##  $ max_roll_belt           : num  NA NA NA NA NA ...
##   [list output truncated]
```

Column *classe*

Of special interest is the column *classe*, which contains the group factors and constitutes the outcome to predict. This column consists of characters "A" to "E" and contains no NA's:

```
## Warning in if (give.length) {: Bedingung hat Länge > 1 und nur das erste
## Element wird benutzt
```

```
##  chr [1:19622] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" ...
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

2.3 Preselecting Covariates

As we saw above, not every column represents signal to be used for prediction. To select the right covariates, we first use the function $nearZeroVar()$ from the *caret*-package. It returns a vector with the index of the near-zero columns and we create a new variable *pml_training1* eliminating the near-zero columns. This reduces the column-number by 60.

Parallel, we create also a new variable *pml_testing1* replicating the changes in the training data, but only without further examination of the testing set.

```
# Preprocessing and analizing possible covariates -> Class Video W206.mp4
## Near Zero Predictors
zeroCovar<-nearZeroVar(pml_training,saveMetrics = 0)
length(zeroCovar)
```

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

```
pml_training1<-pml_training[,-zeroCovar]
pml_testing1<-pml_testing[,-zeroCovar]
```

Remaining columns containing NA's and strings are also removed, as well as columns with heading information, leading to a pair of variables *pml_trainig2* and *pml_testing2* with each 53 columns.

3. Prediction Models

In order to firm the knowledge gained in the course, 3 prediction models were used:

- a CART model,
- a Random Tree model and
- a Boosting model.

For every model several parameters were changed and the accuracy of the predictions to the testing data *pml_testing2* registered, among with the processing time to train the model.

The results can be seen in the following table. The code used for each model can be found in the Appendix.

Prediction Model	Accuracy	Subpartition pml_training2	No. of Covar.	Add. Parameters	Computing Time training function [s]
CART *rpart*	66%	75/25	52	-	6
Random Tree *rf*	87.19%	75/25	4	ntree=10	23
Random Tree *rf*	88.21%	75/25	4	ntree=100	138
Random Tree *rf*	**99.63%**	75/25	52	ntree=10	**131**
Boosting *gbm*	48%	75/25	52	shrinkage=0.01, ntree=100	8.5
Boosting *gbm*	97.1%	75/25	52	shrinkage 0.7, ntree=100	8.6
Boosting *gbm*	**99.55%**	75/25	52	shrinkage 0.7, ntree=300	**24.7**

4. Results

From the results in the table presented in section 3 we corroborate several concepts tough in class:

- The predictions rely on the quality of the collected data and small amounts of data can not be compensated by better algorithms. The prediction accuracy is enormously better taking the 53 columns than only a few of them.
- Both Random Forest and Boosting reach a very high accuracy of 99.63% and 99.55% respectively. The training time for the RF-method was 2.11min compared to only 24.7s for boosting.
- The shrinkage plays a very important role in the boosting algorithm. Here, a high shrinkage value produced a considerable better result, increasing the accuracy in more then 100%, from 48% to 97.1%.

The prediction for the quiz was doing two times, one with the RF-model and the Boosting-model. Both methods give the following results:

Question	Prediction
1	B
2	A
3	B
4	A
5	A
6	E
7	D
8	B
9	A
10	A
11	B
12	C
13	B
14	A
15	E
16	E
17	A
18	B
19	B
20	B

Appendix

A1 CART

```
#Train with different methods
## CART - rpart

### classe vs all columns and rpart, with subpartition
set.seed(62433)
inTrain<-createDataPartition(pml_training2$classe,p=.75,list=FALSE)
CARTtrain<-pml_training2[inTrain,]
CARTtest<-pml_training2[-inTrain,]
t1<-Sys.time()
```

```
modelFitCART<-train(classe~.,data=CARTtrain,method="rpart")
t2<-Sys.time()
predCART<-predict(modelFitCART,CARTtest)
confusionMatrix(predCART,CARTtest$classe) # Accuracy only 66%!!!
```

A2 Random Forest

```
##
### classe vs only 3 columns and random forest with new subpartition
set.seed(62433)
inTrain<-createDataPartition(pml_training2$classe,p=.75,list=FALSE)
RFtrain<-pml_training2[inTrain,]
RFtest<-pml_training2[-inTrain,]
t1<-Sys.time()
modelFitRF3<-train(classe~roll_belt+pitch_belt+yaw_belt+total_accel_belt,
                   method="rf",ntree=100,data=RFtrain)
t2<-Sys.time()
predRF3<-predict(modelFitRF3,RFtest)
confusionMatrix(predRF3,RFtest$classe)
predRF3test<-predict(modelFitRF3,pml_testing2)
# With 4 variables roll_belt+pitch_belt+yaw_belt+total_accel_belt
# accuracy 86.97% and 18s computing time with ntree=10. With ntre=100, 138s computing time and 88.21% a
```

```
### classe vs all columns and random forest with subpartition
set.seed(62433)
inTrain<-createDataPartition(pml_training2$classe,p=.75,list=FALSE)
RFtrain<-pml_training2[inTrain,]
RFtest<-pml_training2[-inTrain,]
t1<-Sys.time()
modelFitRFpart<-train(classe~.,method="rf",ntree=10,data=RFtrain)
t2<-Sys.time()
predRFpart<-predict(modelFitRFpart,RFtest)
confusionMatrix(predRFpart,RFtest$classe) # Accuracy up to 99,63%!!! Und jetzt kein Overfitting!!! In n



## Final Prediction
predRFtest<-predict(modelFitRFpart,pml_testing2)
predRFtest
```

A3 Boosting

```
### classe vs all and boosting with subpartition
set.seed(62433)
inTrain<-createDataPartition(pml_training2$classe,p=.75,list=FALSE)
RFtrain<-pml_training2[inTrain,]
RFtest<-pml_training2[-inTrain,]
t1<-Sys.time()
modelFitBoost<-gbm(classe~.,
                   n.trees=300,
                   data=RFtrain,
                   distribution="multinomial",
```

```
                    shrinkage = .7,verbose=FALSE)
t2<-Sys.time()
predBoost<-predict.gbm(modelFitBoost,RFtest,n.trees=300)
predBoost<-predBoost/max(predBoost)                                    # normalize
predBoostMax<-as.factor(sapply(1:dim(RFtest)[1], function(x) which.max(predBoost[x,,])))
levels(predBoostMax)<-c("A","B","C","D","E")                           # convert to factor
confusionMatrix(predBoostMax,RFtest$classe)
```