PLM - Prediction Assignment Writeup

S Nenning

9 October 2018

1. Synopsis

The objective of this project is to create a prediction model on the manner how **Unilateral Dumbbell Biceps Curl** exercise is performed using training data collected from accelerometers on the belt, forearm, arm, and dumbell of 6 participants performing the exercise. The participants were asked to perform the exercise correctly and incorrectly in 5 different ways and its execution was then classified by an experienced weight lifter into class A to class E.

The machine learning algorithm (prediction) model is tested on the 20 test cases available in the test data provided for this project.

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Summary

3 machine learning algorithm methods have been validated. **Random Forest** with an accuracy of **0.99** has been selected for predicting the outcome for the 20 test cases, which is:

[1] B A C A A E D B A A B C B A E E A B B B

2. Data Analysis

Loading and preprocessing the data

As the first step, data files are downloaded from website, when not downlaoded previously to R working directory, and read it into R as ‘raw’ pml training (pml_training_raw) and ‘raw’ pml testing (pml_testing_raw) dataframe.

```
# download files only  if not yet existing in working directory
if (!file.exists("pml-training.csv")) {
  fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
  download.file(fileUrl, destfile = "pml-training.csv", method = "curl")
}
pml_training_raw <- read.csv(file = "pml-training.csv", header = TRUE, na.strings =
"NA")
if (!file.exists("pml-testing.csv")) {
  fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
  download.file(fileUrl, destfile = "pml-testing.csv", method = "curl")
}
pml_testing_raw <-  read.csv(file = "pml-testing.csv", header = TRUE, na.strings =
"NA")
```

A brief exploratory data analysis is done to understand the data content and number of available records. The training data (*pml_training_raw*) contains 160 columns with 19622 records. The structure of the 'pml_training_raw' data contains variables with NA and #DIV/0! values (please refer to appendix to see the data structure) which cannot be used for prediction. Hence, I remove them from data set, so do I with user and data record descriptive variables in the data set (columns 1 to 7) like user name and time stamp.
The outcome variable **classe** is in the last column of cleansed training data set (**pml_training**).
I reduce the testing data (**pml_testing**) to the same predictor variables; outcome variable 'classe' is not included in testing data since it will be predicted.

```
# data cleansing NA
pml_training <- pml_training_raw[,colSums(is.na(pml_training_raw))==0]
# data cleansing columns with #DIV/0!
pml_training <- pml_training[,colSums(sapply(pml_training, function(x) grepl("#DIV/
0!",x)))==0]
# remove user and data record descriptive variables
pml_training <- pml_training[,-c(1:7)]

#testing data is reduced to the same columns as training data (excluding outcome var
iable 'classe' which is in the last column of training data)
col_test <- names(pml_training[,-ncol(pml_training)])
testing <-  pml_testing_raw[,col_test]
```

The cleansed training data (*pml_training*) contains 53 columns. Please refer to appendix for structure of cleansed training data.
Outcome variable *classe* is a factor calling for a **classification machine learning algorithm** model. 2 models are looked at, next.

Machine Learning Algorithm models

Preparing data

The training data is split into 'training' and 'validation' data using a split ratio of 0.7. The training data is used to 'train' the models; the 'validation' data for validating the model afterwards. Data 'testing' is used to the predict the classification for the 20 test cases included in the test data.

```
# setting seed to get same sample set when creating data partition.
set.seed(12321)
intrain <- createDataPartition(pml_training$classe, p=0.7, list = FALSE)

training <- pml_training[intrain,]
validation <-  pml_training[-intrain,]
```

The 'training' data contains 13737 records, 'validation' data 5885 records, and 'testing' 20 records.

Fitting the model

I have chosen to use 2 different classification machine learning algorthm for fitting the model. The 'Final Model' values are printing for each model.

a.) Fitting Model using **Random Forest** algorithm

Random forests creates decision trees on randomly selected data samples, gets prediction from

each tree and selects the best solution. I'm using *randomForest* funcion from package *randomforest*

```
# fit a model with random forest
modfit_rf<-randomForest(classe ~., data = training, ntree = 50)
print(modfit_rf)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, ntree = 50)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.88%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3900    3    0    1    2 0.001536098
## B   27 2617   12    2    0 0.015425132
## C    0   15 2374    7    0 0.009181970
## D    0    1   32 2215    4 0.016429840
## E    0    1    3   11 2510 0.005940594
```

b.) Fitting Model using **Gradient boosting modelling**

Boosting is another approach to improve the predictions resulting from a decision tree. Trees are 'grown' sequentially: each tree is grown using information from previously grown trees. I'm using method **gbm** in *train* funcion from package *caret*.

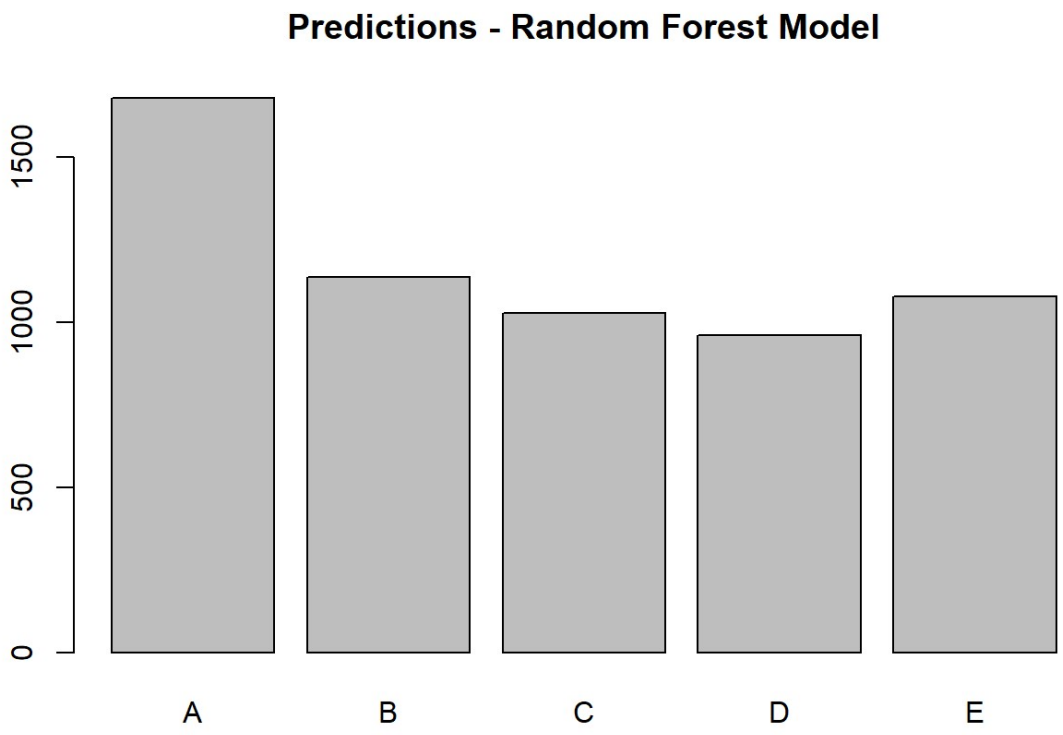
```
# fit a model with boosting
modfit_gbm <- train(classe ~., data = training, method = "gbm", verbose = FALSE, trC
ontrol = trainControl(method = 'cv', number=5))
print(modfit_gbm)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10991, 10988, 10989, 10990, 10990
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7540215  0.6882542
##   1                  100      0.8204105  0.7725929
##   1                  150      0.8506950  0.8109886
##   2                   50      0.8570267  0.8187995
##   2                  100      0.9048537  0.8795846
##   2                  150      0.9315705  0.9134063
##   3                   50      0.8944453  0.8663519
##   3                  100      0.9405982  0.9248326
##   3                  150      0.9607612  0.9503507
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

Model validation

The Machine Learning Algorithm models are validated against the ‘validation’ data.
Steps taken for each model is to:
- Predict the outcome value (variable *classe*) with the model using the validation data.
- Plot the predicted values (see histograms Fig1 to Fig2 below)
- Compute and print confusion matrix to get model parameters like accuracy of prediction model.

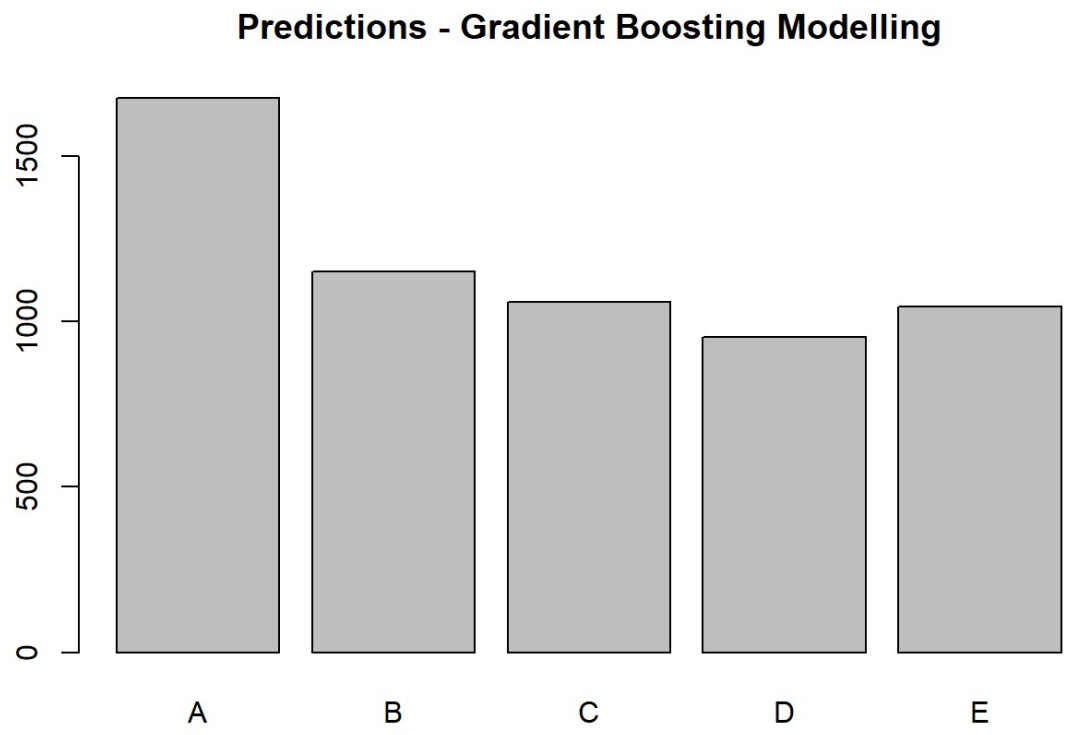
```
# Prediction with random forest model
pred_rf <- predict(modfit_rf, newdata  = validation)
plot(pred_rf, main = "Predictions - Random Forest Model")
```




```
cfm_rf <- confusionMatrix(validation$classe, pred_rf)
print(cfm_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    6 1128    5    0    0
##          C    0    8 1016    2    0
##          D    0    0    8  956    0
##          E    0    0    0    4 1078
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9921, 0.9961)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9929
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9930   0.9874   0.9938   1.0000
## Specificity            1.0000   0.9977   0.9979   0.9984   0.9992
## Pos Pred Value         1.0000   0.9903   0.9903   0.9917   0.9963
## Neg Pred Value         0.9986   0.9983   0.9973   0.9988   1.0000
## Prevalence             0.2855   0.1930   0.1749   0.1635   0.1832
## Detection Rate         0.2845   0.1917   0.1726   0.1624   0.1832
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9982   0.9953   0.9927   0.9961   0.9996
```

```
# Prediction with boosting model
pred_gbm <- predict(modfit_gbm, newdata  = validation)
plot(pred_gbm, main = "Predictions - Gradient Boosting Modelling")
```




```
cfm_gbm <- confusionMatrix(validation$classe, pred_gbm)
print(cfm_gbm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1644   20    6    3    1
##          B   30 1081   28    0    0
##          C    0   34  979   13    0
##          D    1    1   32  923    7
##          E    1   16   15   13 1037
##
## Overall Statistics
##
##                Accuracy : 0.9624
##                  95% CI : (0.9573, 0.9672)
##     No Information Rate : 0.2848
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9525
##  Mcnemar's Test P-Value : 1.472e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9809   0.9384   0.9236   0.9695   0.9923
## Specificity            0.9929   0.9877   0.9903   0.9917   0.9907
## Pos Pred Value         0.9821   0.9491   0.9542   0.9575   0.9584
## Neg Pred Value         0.9924   0.9850   0.9833   0.9941   0.9983
## Prevalence             0.2848   0.1958   0.1801   0.1618   0.1776
## Detection Rate         0.2794   0.1837   0.1664   0.1568   0.1762
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9869   0.9631   0.9569   0.9806   0.9915
```

The 2 models have following **accuracy**; values are retrieved from confusion matrix of model:
- Random Forest: 0.99
- Gradient Boosting: 0.96

Results

Based on the accuracy of the models, I select the model using the **Random Forest** Machine Learning Algorithm with an accuracy of 0.99 against the validation data.

The predicted outcome for the 20 test cases from the testing data is printed below.

```
# Prediction with random forest model
pred_rf_test <- predict(modfit_rf, newdata  = testing)
pred_rf_test
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

Appendix

Additional information

Data Structure of Training Data (pml_training_raw, before manipulation)

```
# Structure of training data (raw)
str(pml_training_raw)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2
2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 132
3084232 1323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 36829
6 440390 484323 484434 ...
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9
9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1
1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.
45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.
17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.
4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1
1 1 1 1 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1
1 1 1 1 ...
##  $ kurtosis_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1
1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1
1 1 1 1 ...
##  $ skewness_roll_belt.1    : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1
1 1 1 1 ...
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1
1 ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1
1 1 1 1 1 ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1
1 1 1 1 1 ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1
1 1 1 1 1 1 ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.0
2 -0.02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -3
08 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -1
28 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.
6 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -1
61 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.
03 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -2
88 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -1
24 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -3
76 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ kurtosis_roll_arm       : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1
1 1 1 1 ...
##  $ kurtosis_picth_arm      : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1
1 1 1 1 ...
##  $ kurtosis_yaw_arm        : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1
1 1 1 1 ...
##  $ skewness_roll_arm       : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1
1 1 1 1 ...
##  $ skewness_pitch_arm      : Factor w/ 328 levels "","-0.00184",..: 1 1 1 1 1 1
1 1 1 1 ...
```

```
##  $ skewness_yaw_arm        : Factor w/ 395 levels "","-0.00311",..: 1 1 1 1 1 1
1 1 1 1 ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell            : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ kurtosis_roll_dumbbell  : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1
1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 1
1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1
1 ...
##  $ skewness_roll_dumbbell  : Factor w/ 401 levels "","-0.0082","-0.0096",..: 1 1
1 1 1 1 1 1 1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 1
1 1 1 1 1 1 1 ...
##  $ skewness_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1
1 ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1
1 1 1 1 1 ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1
1 1 1 1 1 ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

Data Structure of cleansed Training Data (pml_training, after manipulation)

```
# Structure of training data
str(pml_training)
```

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.4
5 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.1
7 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -9
4.4 -94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y        : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.
02 0 ...
##  $ accel_belt_x        : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y        : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z        : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x       : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y       : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z       : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -30
8 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -12
8 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -16
1 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03
-0.03 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x         : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -28
8 ...
##  $ accel_arm_y         : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z         : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -12
4 ...
##  $ magnet_arm_x        : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -37
6 ...
##  $ magnet_arm_y        : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z        : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.
02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -23
5 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -27
0 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -55
```

```
8 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -6
3.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -15
2 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.0
2 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -21
5 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1
1 1 ...
```