Practical Machine Learning wk 4 course work

Matti Niemist?

January 28, 2018

Instructions for the course project

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.

Let’s get started, load and look at the data

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3
```

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

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

```
library(doParallel)
```

```
## Warning: package 'doParallel' was built under R version 3.4.3
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 3.4.3
```

```
## Loading required package: iterators
```

```
## Warning: package 'iterators' was built under R version 3.4.3
```

```
## Loading required package: parallel
```

```
registerDoParallel(cores=2)
if(!file.exists("pml-training.csv"))
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", "pml-tra
ining.csv")
if(!file.exists("pml-testing.csv"))
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", "pml-test
ing.csv")
training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")
dim(training)
```

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

```
dim(testing)
```

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

So the total number of variables is quite big, 160. There are 19622 observations in the training set and 20 observations in the test set. For sake of saving some screen length summaries and/or str of the data is not presented here, see end of document for listings.

Steps in the exercise

1. build model(s)
2. cross validate
3. estimate out of sample error rate
4. explain made choices

Some data cleaning

OK there seems to be a lot of missing values in some of the variables. Let's check the situation

```
nas <- is.na(training)
sums <- apply(nas, 2, sum)
```

Looks like some variables have 19216 missing datapoints out of the 19622 observations. Let's make a guestimate that these can be omitted. Most likely this move will make me fail making 20/20 in the prediction quiz, but newertheless this is the chosen path

```
train2 <- training[,sums <= 10000]
sum(is.na(train2))
```

```
## [1] 0
```

OK. we are left with 93 variables and none of them have any missing values left.

Building models

Let’s first see what are the possible outcomes that we are trying to predict. Also, let’s build the data frames for building the model and cross validation

```
levels(training$classe)
```

```
## [1] "A" "B" "C" "D" "E"
```

```
inbuild <- createDataPartition(y = training$classe, p = 0.7, list = FALSE)
validation <- train2[-inbuild,]
buildData <- train2[inbuild,]
```

So the decision was to use 70% of the data for building the models and 30% of the data for validation. This should do it. After some googling it seems that Naive Bayes, Neural Networks and SVM are good choices for multi-class classification. In this exercise I’m going to start with SVM adn later try Naive Bayes as Neural nets are well covered in Deep Learning Specialization (which I highly recommend as well :))

SVM

Next, let’s fit a svm model with all possible remaining variables. By setting the method repeatedcv we do cross validation while building model. The data is split to 10 folds and the whole process is repeated 3 times.

```
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(3233)
```

```
mod1 <- train(classe ~., data = buildData, method = "svmLinear",
                 trControl=trctrl,
                 preProcess = c("center", "scale"),
                 tuneLength = 10)
## Save model for future use
saveRDS(mod1, "./mod1.rds")
```

Appararently the was a lot of data, training the model took almost 2h. Let’s predict on the validation set using our new model and see what the confusion matrix looks like.

```
## load pre-saved model to avoid 2h re-train in every iteration
mod1 <- readRDS("./mod1.rds")
mod1_pred <- predict(mod1, newdata = validation)
mod1$finalModel
```

```
## Support Vector Machine object of class "ksvm" 
## 
## SV type: C-svc  (classification) 
##  parameter : cost C = 1 
## 
## Linear (vanilla) kernel function. 
## 
## Number of Support Vectors : 537 
## 
## Objective Function Value : -7.1053 -1.3077 -0.4779 -0.1853 -6.3692 -1.3518 -0.4074 -7.3808 -
1.3132 -6.432 
## Training error : 0
```

```
confusionMatrix(mod1_pred, validation$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    0    0    0
##          C    0    0 1026    0    0
##          D    0    0    0  964    0
##          E    0    0    0    0 1082
## 
## Overall Statistics
##                                      
##                Accuracy : 1          
##                  95% CI : (0.9994, 1)
##     No Information Rate : 0.2845     
##     P-Value [Acc > NIR] : < 2.2e-16  
##                                      
##                   Kappa : 1          
##  Mcnemar's Test P-Value : NA         
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

But was it worth the wait? finalModel tells us that we got C-svc type svm with training error of 0. This sounds great but raises a concern of overfitted model. confusionMatrix also tells that Accuracy is 1 so it will be very interesting to see whether or not the model will give poor results when predicting on the test set. Also the 95% confidence interval is 0.9994, 1, so out of sample error rate should basically be 0. This can be also verified from the Reference-Prediction matrix produced by the confusionMatrix call. Looks perfect, which obviously cannot be the case and remains to be seen how well the model generalizes on the test set.

EDIT: Like expected the model was badly overfitted and ended up classifying plain A’s in the test set.

Naive-Bayes

Still trying the Naive-Bayes model to see how it compares to the (apparently) so well working SVM model. (Note to run this code, remove eval=F)

```
mod2 <- train(classe ~., data = buildData, method = "naive_bayes",
                 trControl=trctrl,
                 preProcess = c("center", "scale"),
                 tuneLength = 10)
## Save model for future use
saveRDS(mod2, "./mod2.rds")
```

```
## load pre-saved model to avoid 2h re-train in every iteration
mod2 <- readRDS("./mod2.rds")
mod2_pred <- predict(mod2, newdata = validation)
mod2$finalModel
confusionMatrix(mod2_pred, validation$classe)
```

This model was really bad. Accuracy is only 0.18 and all cross validation classification items ended up being class E. I’ll stop here without further debugging what went wrong in the model.

SVM again, but with limited variables

Re-reading the instructions the goal was to predict classe with accelometer data from arm, forearm, belt and dumbbell. Let’s re-build the data sets and the SVM model.

```
treeni3 <- train2[,c("classe", "accel_forearm_x","accel_forearm_y", "accel_forearm_z", "total_ac
cel_forearm", "accel_dumbbell_x", "accel_dumbbell_y", "accel_dumbbell_z", "total_accel_dumbbell"
, "accel_arm_x","accel_arm_y", "accel_arm_z", "total_accel_arm", "accel_belt_x","accel_belt_y",
"accel_belt_z", "total_accel_belt")]

testi3 <- testing[,c("accel_forearm_x","accel_forearm_y", "accel_forearm_z", "total_accel_forear
m", "accel_dumbbell_x", "accel_dumbbell_y", "accel_dumbbell_z", "total_accel_dumbbell", "accel_a
rm_x","accel_arm_y", "accel_arm_z", "total_accel_arm", "accel_belt_x","accel_belt_y", "accel_bel
t_z", "total_accel_belt")]

build3 <- createDataPartition(y = treeni3$classe, p = 0.7, list = FALSE)
valid3 <- treeni3[-build3,]
bData3 <- treeni3[build3,]
```

```
mod3 <- train(classe ~., data = bData, method = "svmLinear",
              trControl=trctrl,
              preProcess = c("center", "scale"),
              tuneLength = 10)
saveRDS(mod3, "./mod3.rds")
```

Look at the results

```
mod3 <- readRDS("./mod3.rds")
mod3_pred <- predict(mod3, newdata = valid3)
mod3$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 11524
##
## Objective Function Value : -3739.986 -4250.393 -3082.282 -2919.028 -2851.323 -2272.62 -3253.8
16 -1941.383 -2292.46 -2360.322
## Training error : 0.449807
```

```
confusionMatrix(mod3_pred, valid3$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1137  312  438  202  142
##          B   83  561  104   46  183
##          C  180  136  428   35   94
##          D  262   77   46  606  133
##          E   12   53   10   75  530
## 
## Overall Statistics
##                                           
##                Accuracy : 0.5543          
##                  95% CI : (0.5415, 0.567) 
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.4298          
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6792  0.49254  0.41715   0.6286  0.48983
## Specificity            0.7402  0.91235  0.90842   0.8947  0.96877
## Pos Pred Value         0.5096  0.57421  0.49026   0.5391  0.77941
## Neg Pred Value         0.8530  0.88223  0.88069   0.9248  0.89395
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.1932  0.09533  0.07273   0.1030  0.09006
## Detection Prevalence   0.3791  0.16602  0.14834   0.1910  0.11555
## Balanced Accuracy      0.7097  0.70244  0.66279   0.7617  0.72930
```

The model looks bad, accuracy is only around 55%. So just using the accelerometer values in not enough to build a valid classification model. Let’s include more variables

```
treeni4 <- train2[,c("classe", "roll_belt", "roll_arm", "roll_dumbbell", "roll_forearm", "pitch_
belt", "pitch_arm", "pitch_dumbbell", "pitch_forearm", "yaw_belt", "yaw_arm", "yaw_dumbbell", "y
aw_forearm", "accel_forearm_x","accel_forearm_y", "accel_forearm_z", "total_accel_forearm", "acc
el_dumbbell_x", "accel_dumbbell_y", "accel_dumbbell_z", "total_accel_dumbbell", "accel_arm_x","a
ccel_arm_y", "accel_arm_z", "total_accel_arm", "accel_belt_x","accel_belt_y", "accel_belt_z", "t
otal_accel_belt", "gyros_belt_x", "gyros_belt_y", "gyros_belt_z", "magnet_belt_x", "magnet_belt_
y", "magnet_belt_z", "gyros_arm_x", "gyros_arm_y", "gyros_arm_z", "gyros_dumbbell_x", "gyros_dum
bbell_y", "gyros_dumbbell_z", "gyros_forearm_x", "gyros_forearm_y", "gyros_forearm_x", "magnet_a
rm_x", "magnet_arm_y", "magnet_arm_z", "magnet_dumbbell_x", "magnet_dumbbell_y", "magnet_dumbbel
l_z", "magnet_forearm_x", "magnet_forearm_y", "magnet_forearm_z")]

testi4 <- testing[,c("roll_belt", "roll_arm", "roll_dumbbell", "roll_forearm", "pitch_belt", "pi
tch_arm", "pitch_dumbbell", "pitch_forearm", "yaw_belt", "yaw_arm", "yaw_dumbbell", "yaw_forear
m", "accel_forearm_x","accel_forearm_y", "accel_forearm_z", "total_accel_forearm", "accel_dumbbe
ll_x", "accel_dumbbell_y", "accel_dumbbell_z", "total_accel_dumbbell", "accel_arm_x","accel_arm_
y", "accel_arm_z", "total_accel_arm", "accel_belt_x","accel_belt_y", "accel_belt_z", "total_acce
l_belt", "gyros_belt_x", "gyros_belt_y", "gyros_belt_z", "magnet_belt_x", "magnet_belt_y", "magn
et_belt_z", "gyros_arm_x", "gyros_arm_y", "gyros_arm_z", "gyros_dumbbell_x", "gyros_dumbbell_y",
 "gyros_dumbbell_z", "gyros_forearm_x", "gyros_forearm_y", "gyros_forearm_x", "magnet_arm_x", "m
agnet_arm_y", "magnet_arm_z", "magnet_dumbbell_x", "magnet_dumbbell_y", "magnet_dumbbell_z", "ma
gnet_forearm_x", "magnet_forearm_y", "magnet_forearm_z")]

build4 <- createDataPartition(y = treeni4$classe, p = 0.7, list = FALSE)
valid4 <- treeni4[-build4,]
bData4 <- treeni4[build4,]
```

```
mod4 <- train(classe ~., data = bData4, method = "svmLinear",
              trControl=trctrl,
              preProcess = c("center", "scale"),
              tuneLength = 10)
saveRDS(mod4, "./mod4.rds")
```

```
mod4 <- readRDS("./mod4.rds")
mod4_pred <- predict(mod4, newdata = valid4)
mod4$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 7212
##
## Objective Function Value : -1443.719 -1264.544 -1039.324 -658.0457 -1322.28 -883.8861 -1765.0
64 -1233.191 -1063.673 -1210.19
## Training error : 0.211545
```

```
confusionMatrix(mod4_pred, valid4$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1549  149   89   68   70
##          B   34  817   75   39  129
##          C   40   77  817  112   65
##          D   42   22   24  704   62
##          E    9   74   21   41  756
## 
## Overall Statistics
##                                           
##                Accuracy : 0.789           
##                  95% CI : (0.7783, 0.7993)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.7315          
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9253   0.7173   0.7963   0.7303   0.6987
## Specificity            0.9107   0.9416   0.9395   0.9695   0.9698
## Pos Pred Value         0.8047   0.7468   0.7354   0.8244   0.8391
## Neg Pred Value         0.9684   0.9328   0.9562   0.9483   0.9346
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2632   0.1388   0.1388   0.1196   0.1285
## Detection Prevalence   0.3271   0.1859   0.1888   0.1451   0.1531
## Balanced Accuracy      0.9180   0.8295   0.8679   0.8499   0.8343
```

Better, accuracy is now ~79% but still not sufficient. After many more frustrating trials and reading I ended up building following model.

```
treeni5 <- train2[,c("classe", "user_name", "roll_belt", "roll_arm", "roll_dumbbell", "roll_fore
arm", "pitch_belt", "pitch_arm", "pitch_dumbbell", "pitch_forearm", "yaw_belt", "yaw_arm", "yaw_
dumbbell", "yaw_forearm", "accel_forearm_x","accel_forearm_y", "accel_forearm_z", "total_accel_f
orearm", "accel_dumbbell_x", "accel_dumbbell_y", "accel_dumbbell_z", "total_accel_dumbbell", "ac
cel_arm_x","accel_arm_y", "accel_arm_z", "total_accel_arm", "accel_belt_x","accel_belt_y", "acce
l_belt_z", "total_accel_belt", "gyros_belt_x", "gyros_belt_y", "gyros_belt_z", "magnet_belt_x",
"magnet_belt_y", "magnet_belt_z", "gyros_arm_x", "gyros_arm_y", "gyros_arm_z", "gyros_dumbbell_
x", "gyros_dumbbell_y", "gyros_dumbbell_z", "gyros_forearm_x", "gyros_forearm_y", "gyros_forearm
_x", "magnet_arm_x", "magnet_arm_y", "magnet_arm_z", "magnet_dumbbell_x", "magnet_dumbbell_y",
"magnet_dumbbell_z", "magnet_forearm_x", "magnet_forearm_y", "magnet_forearm_z")]

testi5 <- testing[,c("user_name", "roll_belt", "roll_arm", "roll_dumbbell", "roll_forearm", "pit
ch_belt", "pitch_arm", "pitch_dumbbell", "pitch_forearm", "yaw_belt", "yaw_arm", "yaw_dumbbell",
 "yaw_forearm", "accel_forearm_x","accel_forearm_y", "accel_forearm_z", "total_accel_forearm",
"accel_dumbbell_x", "accel_dumbbell_y", "accel_dumbbell_z", "total_accel_dumbbell", "accel_arm_
x","accel_arm_y", "accel_arm_z", "total_accel_arm", "accel_belt_x","accel_belt_y", "accel_belt_
z", "total_accel_belt", "gyros_belt_x", "gyros_belt_y", "gyros_belt_z", "magnet_belt_x", "magnet
_belt_y", "magnet_belt_z", "gyros_arm_x", "gyros_arm_y", "gyros_arm_z", "gyros_dumbbell_x", "gyr
os_dumbbell_y", "gyros_dumbbell_z", "gyros_forearm_x", "gyros_forearm_y", "gyros_forearm_x", "ma
gnet_arm_x", "magnet_arm_y", "magnet_arm_z", "magnet_dumbbell_x", "magnet_dumbbell_y", "magnet_d
umbbell_z", "magnet_forearm_x", "magnet_forearm_y", "magnet_forearm_z")]

build5 <- createDataPartition(y = treeni5$classe, p = 0.7, list = FALSE)
valid5 <- treeni5[-build5,]
bData5 <- treeni5[build5,]
```

```
mod5 <- train(classe ~., data = bData5, method = "svmRadial",
              trControl=trctrl,
              preProcess = c("center", "scale"),
              tuneLength = 10)
saveRDS(mod5, "./mod5.rds")
```

```
mod5 <- readRDS("./mod5.rds")
mod5_pred <- predict(mod5, newdata = valid5)
mod5$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 128
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0121734771450409
##
## Number of Support Vectors : 3161
##
## Objective Function Value : -9420.6 -2017.554 -1632.161 -626.9259 -4406.045 -919.5816 -1369.66
2 -18812.67 -2697.088 -3030.466
## Training error : 0.001165
```

```
confusionMatrix(mod5_pred, valid5$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    3    0    0    0
##          B    0 1134    2    0    0
##          C    0    2 1021    8    1
##          D    0    0    3  953    0
##          E    2    0    0    3 1081
##
## Overall Statistics
##
##                Accuracy : 0.9959
##                  95% CI : (0.9939, 0.9974)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9948
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9956   0.9951   0.9886   0.9991
## Specificity            0.9993   0.9996   0.9977   0.9994   0.9990
## Pos Pred Value         0.9982   0.9982   0.9893   0.9969   0.9954
## Neg Pred Value         0.9995   0.9989   0.9990   0.9978   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1927   0.1735   0.1619   0.1837
## Detection Prevalence   0.2846   0.1930   0.1754   0.1624   0.1845
## Balanced Accuracy      0.9990   0.9976   0.9964   0.9940   0.9990
```

Finally a model that seems to perform very well on the cross validation data but does not seem to be (hopefully) overfitted. It will be interesting to see how the model will perform on the final course quiz.

Conclusions and reasoning

So the winner in this case was model mod6 with accuracy close 1 of and out of sample error close to 0 on the validation set, which was 30% of the training set. Cross validation was already done during model built time by using options repeatedcv, 10 folds and 3 repeats. As sanity check remaining 30% was used to cross validate the model before selecting it as the final model. It will be interesting see how it works with the test set.

Key steps to achieve these results was to eliminate unnecessary variables from the data set basically by trial and error. In retrospect some other mechanism could have made sense as well, like trying PCA on the preprosessing phase. Newertheless, this was a good learning experiment. SVM and Naive-Bayes were chosen as model types as they tend to work well on (multiclass) classification problems. Neural nets were omitted for personal reasons as I study them currently on a separate course.

Appendix

more information about the traingin and test data sets.

```
str(training)
```

```
## '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 1323084232 1323
084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484
323 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 -0.02 -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 -308 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ 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 -161 ...
##  $ 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 -288 ...
##  $ 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 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ 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 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 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 1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 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]
```

```
str(testing)
```

```
## 'data.frame':    20 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 6 5 5 1 4 5 5 5 2 3
...
##  $ raw_timestamp_part_1    : int  1323095002 1322673067 1322673075 1322832789 1322489635 1322
673149 1322673128 1322673076 1323084240 1322837822 ...
##  $ raw_timestamp_part_2    : int  868349 778725 342967 560311 814776 510661 766645 54671 9163
13 384285 ...
##  $ cvtd_timestamp          : Factor w/ 11 levels "02/12/2011 13:33",..: 5 10 10 1 6 11 11 10
3 2 ...
##  $ new_window              : Factor w/ 1 level "no": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  74 431 439 194 235 504 485 440 323 664 ...
##  $ roll_belt               : num  123 1.02 0.87 125 1.35 -5.92 1.2 0.43 0.93 114 ...
##  $ pitch_belt              : num  27 4.87 1.82 -41.6 3.33 1.59 4.44 4.15 6.72 22.4 ...
##  $ yaw_belt                : num  -4.75 -88.9 -88.5 162 -88.6 -87.7 -87.3 -88.5 -93.7 -13.1
...
##  $ total_accel_belt        : int  20 4 5 17 3 4 4 4 4 18 ...
##  $ kurtosis_roll_belt      : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1    : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ max_picth_belt          : logi  NA NA NA NA NA NA ...
##  $ max_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ min_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ min_pitch_belt          : logi  NA NA NA NA NA NA ...
##  $ min_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_belt     : logi  NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : logi  NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : logi  NA NA NA NA NA NA ...
##  $ var_total_accel_belt    : logi  NA NA NA NA NA NA ...
##  $ avg_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : logi  NA NA NA NA NA NA ...
##  $ var_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : logi  NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : logi  NA NA NA NA NA NA ...
##  $ var_pitch_belt          : logi  NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : logi  NA NA NA NA NA NA ...
##  $ var_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  -0.5 -0.06 0.05 0.11 0.03 0.1 -0.06 -0.18 0.1 0.14 ...
##  $ gyros_belt_y            : num  -0.02 -0.02 0.02 0.11 0.02 0.05 0 -0.02 0 0.11 ...
##  $ gyros_belt_z            : num  -0.46 -0.07 0.03 -0.16 0 -0.13 0 -0.03 -0.02 -0.16 ...
##  $ accel_belt_x            : int  -38 -13 1 46 -8 -11 -14 -10 -15 -25 ...
##  $ accel_belt_y            : int  69 11 -1 45 4 -16 2 -2 1 63 ...
##  $ accel_belt_z            : int  -179 39 49 -156 27 38 35 42 32 -158 ...
##  $ magnet_belt_x           : int  -13 43 29 169 33 31 50 39 -6 10 ...
##  $ magnet_belt_y           : int  581 636 631 608 566 638 622 635 600 601 ...
##  $ magnet_belt_z           : int  -382 -309 -312 -304 -418 -291 -315 -305 -302 -330 ...
##  $ roll_arm                : num  40.7 0 0 -109 76.1 0 0 0 -137 -82.4 ...
##  $ pitch_arm               : num  -27.8 0 0 55 2.76 0 0 0 11.2 -63.8 ...
##  $ yaw_arm                : num  178 0 0 -142 102 0 0 0 -167 -75.3 ...
##  $ total_accel_arm        : int  10 38 44 25 29 14 15 22 34 32 ...
##  $ var_accel_arm          : logi  NA NA NA NA NA NA ...
##  $ avg_roll_arm           : logi  NA NA NA NA NA NA ...
##  $ stddev_roll_arm        : logi  NA NA NA NA NA NA ...
##  $ var_roll_arm           : logi  NA NA NA NA NA NA ...
##  $ avg_pitch_arm          : logi  NA NA NA NA NA NA ...
##  $ stddev_pitch_arm       : logi  NA NA NA NA NA NA ...
##  $ var_pitch_arm          : logi  NA NA NA NA NA NA ...
##  $ avg_yaw_arm            : logi  NA NA NA NA NA NA ...
##  $ stddev_yaw_arm         : logi  NA NA NA NA NA NA ...
##  $ var_yaw_arm            : logi  NA NA NA NA NA NA ...
##  $ gyros_arm_x            : num  -1.65 -1.17 2.1 0.22 -1.96 0.02 2.36 -3.71 0.03 0.26 ...
##  $ gyros_arm_y            : num  0.48 0.85 -1.36 -0.51 0.79 0.05 -1.01 1.85 -0.02 -0.5 ...
##  $ gyros_arm_z            : num  -0.18 -0.43 1.13 0.92 -0.54 -0.07 0.89 -0.69 -0.02 0.79 ...
##  $ accel_arm_x            : int  16 -290 -341 -238 -197 -26 99 -98 -287 -301 ...
##  $ accel_arm_y            : int  38 215 245 -57 200 130 79 175 111 -42 ...
##  $ accel_arm_z            : int  93 -90 -87 6 -30 -19 -67 -78 -122 -80 ...
##  $ magnet_arm_x           : int  -326 -325 -264 -173 -170 396 702 535 -367 -420 ...
##  $ magnet_arm_y           : int  385 447 474 257 275 176 15 215 335 294 ...
##  $ magnet_arm_z           : int  481 434 413 633 617 516 217 385 520 493 ...
##  $ kurtosis_roll_arm      : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm     : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_arm      : logi  NA NA NA NA NA NA ...
##  $ skewness_pitch_arm     : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_arm       : logi  NA NA NA NA NA NA ...
##  $ max_roll_arm           : logi  NA NA NA NA NA NA ...
##  $ max_picth_arm          : logi  NA NA NA NA NA NA ...
##  $ max_yaw_arm            : logi  NA NA NA NA NA NA ...
##  $ min_roll_arm           : logi  NA NA NA NA NA NA ...
##  $ min_pitch_arm          : logi  NA NA NA NA NA NA ...
##  $ min_yaw_arm            : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_arm     : logi  NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm    : logi  NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm      : logi  NA NA NA NA NA NA ...
##  $ roll_dumbbell          : num  -17.7 54.5 57.1 43.1 -101.4 ...
##  $ pitch_dumbbell         : num  25 -53.7 -51.4 -30 -53.4 ...
##  $ yaw_dumbbell           : num  126.2 -75.5 -75.2 -103.3 -14.2 ...
##  $ kurtosis_roll_dumbbell : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell: logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell : logi  NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell: logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ max_roll_dumbbell      : logi  NA NA NA NA NA NA ...
##  $ max_picth_dumbbell     : logi  NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell       : logi  NA NA NA NA NA NA ...
##  $ min_roll_dumbbell      : logi  NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell     : logi  NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell       : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell: logi  NA NA NA NA NA NA ...
##   [list output truncated]
```