Practical Machine Learning Course Project

by Haim Kotler, Sep 2017

Basic Data Exploration and Cleaning

First I will load the training dataset, and explore its columns:

```
trainingRaw <- read.csv("pml-training.csv")
str (trainingRaw)
```

```
## '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 132
3084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2     : int  788290 808298 820366 120339 196328 304277 368296 440390 48
4323 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]
```

As can be seen above, many of the columns contain large number of empty or NA values, and therefore are presumed not suiteable to be used as covariates.

I am constructing a new dataset, which contains only columns which are potential covariates, leaving out those that include lot of empty data (more than 10%), and those with ‘chronoligal’ data, such as time stamps, which probeably do not have any impact on prediction.

I decided to leave the name of the user in, as this may be significant for prediction.

For cross validation I decided to split the trainig data into training (75%) and valiudation data (25%). This way I am building the model on the training data and using the validation data to test my models before implementing them on the actual given test data.

```
    library(caret)

    ## select possible relevant covariates
    actCols <- 2
    for (i in 8:ncol(trainingRaw)){
        nas = (length (trainingRaw[is.na(trainingRaw[i]) ,1]) + length (trainingRaw[trainingRaw
[i]=="",1]))
        if (nas < 0.1 * nrow(trainingRaw)) actCols <- c(actCols,i)
    }
```

```r
## split to training and validation data sets
set.seed(1812)
inTrain <- createDataPartition(y=trainingRaw$classe,p=0.75, list=FALSE)
training <- trainingRaw[inTrain,actCols]
validating <- trainingRaw[-inTrain,actCols]

## Columns of final training dataset
print(colnames(training))
```

```
##  [1] "user_name"            "roll_belt"            "pitch_belt"
##  [4] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
##  [7] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [10] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [13] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [16] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [19] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [22] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [25] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [28] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [31] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [34] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [37] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [40] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [43] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [46] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [49] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [52] "magnet_forearm_y"     "magnet_forearm_z"     "classe"
```

Exploring the covariates

After cleaning the data we are left with 54 covariates (including the classe column).

This is still a verly large number of covariates which are difficult to handle in the more computationly intensive prediction algorithems.

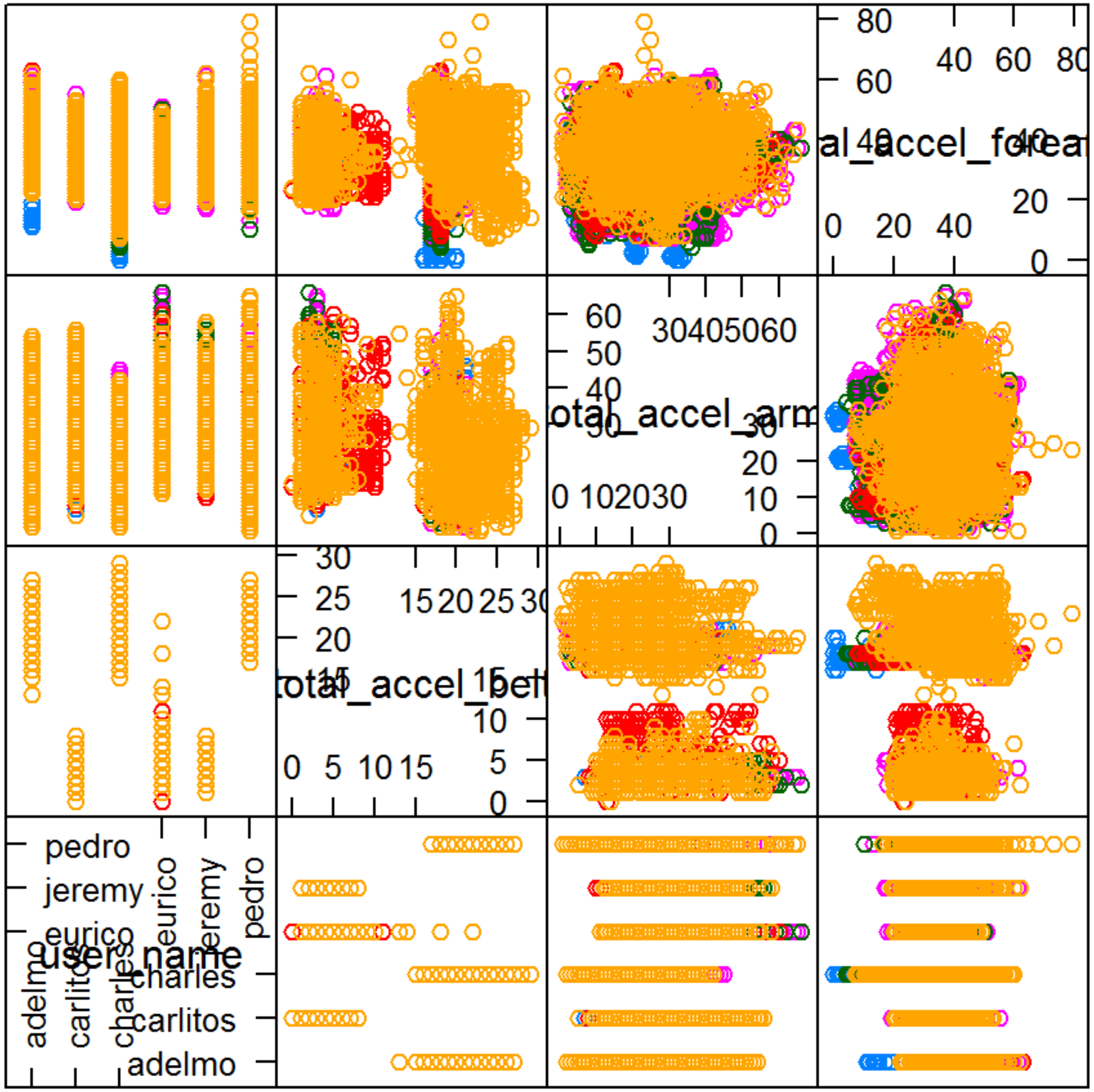
At this point our goal is to see if some covariates stand out (show a strong corelation to the classe), so a smaller sub group of covariates may be used for prediction.

I am prsenting several faeture plots that demonstrate correlation between pairs on parametrs, and with the 'classe' parameter (displayed on these charts using differnt colours).

I decided to focus first on the dumbbels readings, asuuming that, being the final objective of the excersize, it may contain the most data about the execution classe.

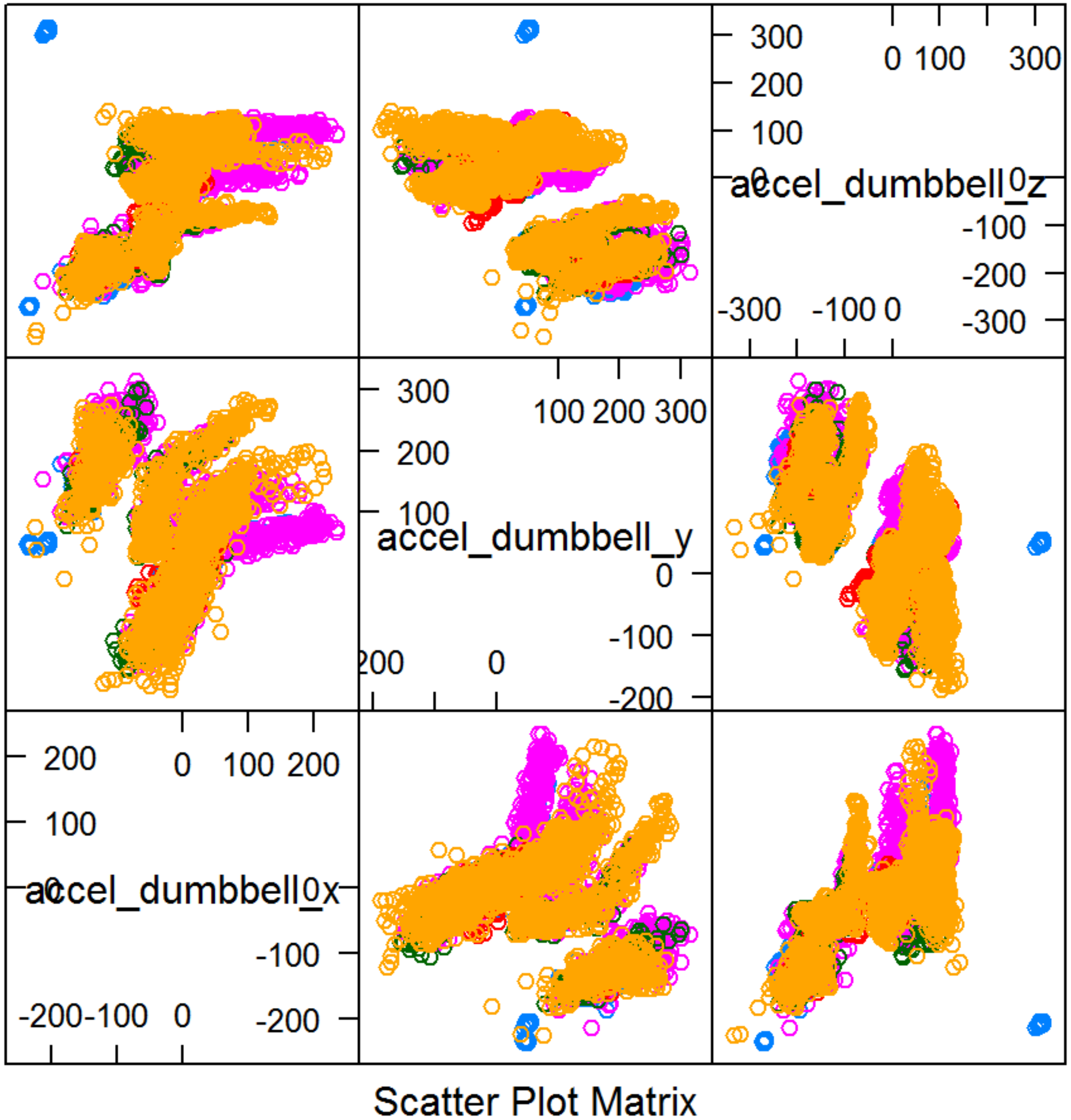
```
    library(ggplot2)
    library(caret)

featurePlot(x=training[, c("user_name","total_accel_belt","total_accel_arm","total_accel_forearm")], y = training$classe, plot="pairs")
```

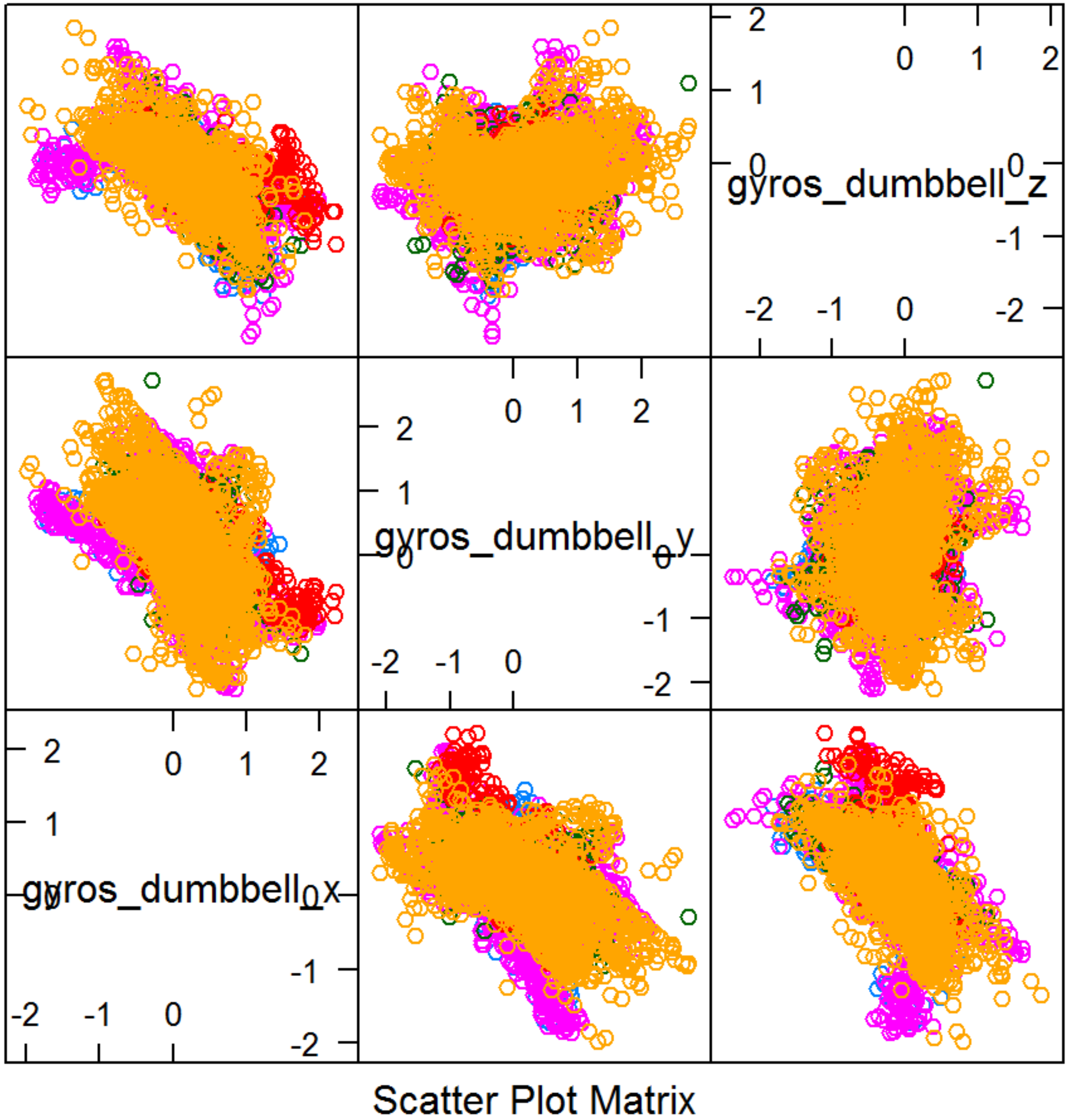


Scatter Plot Matrix

```
featurePlot(x=training[, c("accel_dumbbell_x","accel_dumbbell_y","accel_dumbbell_z")], y = trai
ning$classe, plot="pairs")
```



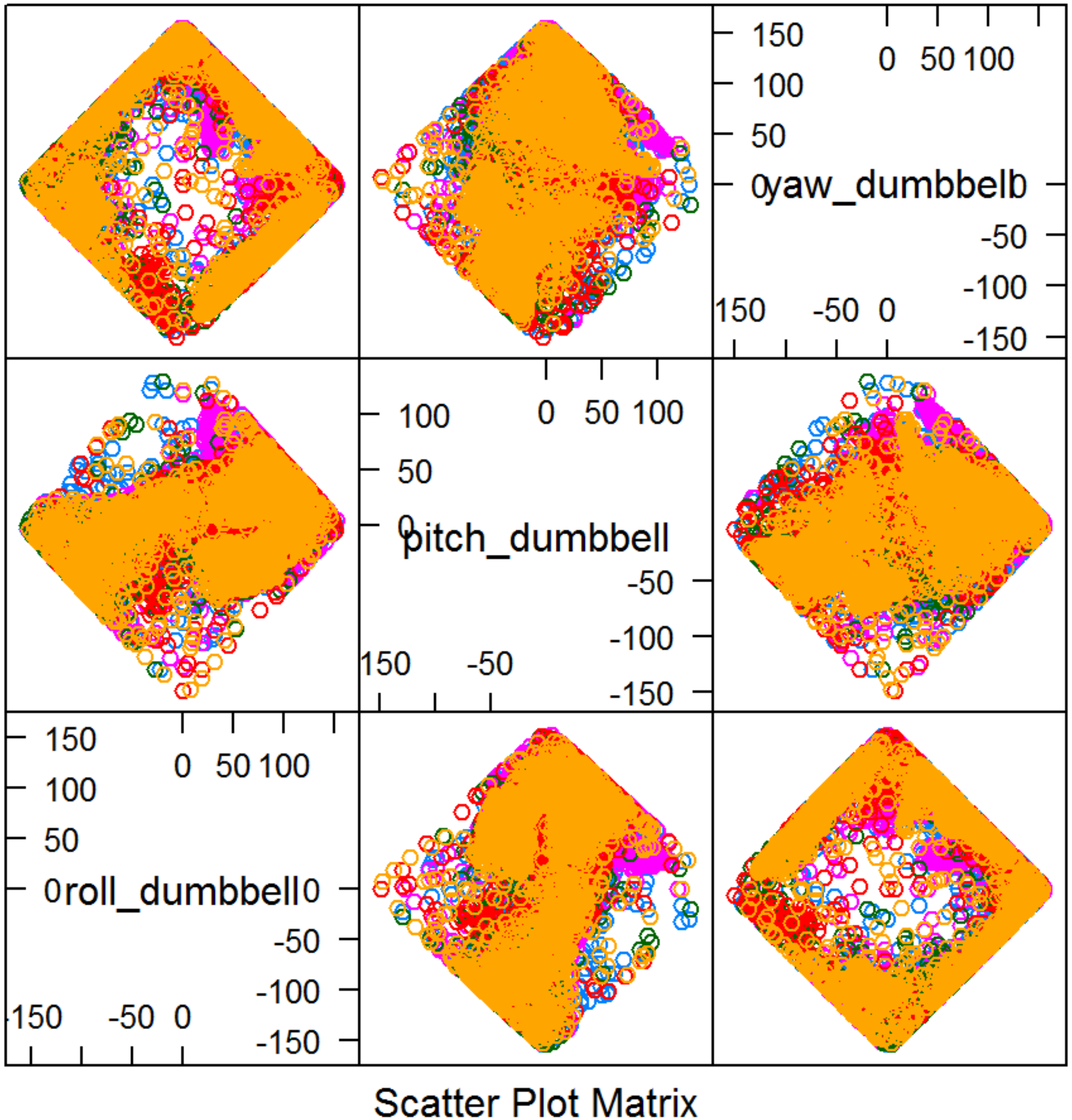

```
featurePlot(x=training[, c("gyros_dumbbell_x","gyros_dumbbell_y","gyros_dumbbell_z")], y = trai
ning$classe, plot="pairs")
```




```
featurePlot(x=training[, c("roll_dumbbell","pitch_dumbbell","yaw_dumbbell")], y = training$clas
se, plot="pairs")

featurePlot(x=training[, c("roll_dumbbell","pitch_dumbbell","yaw_dumbbell")], y = training$clas
```

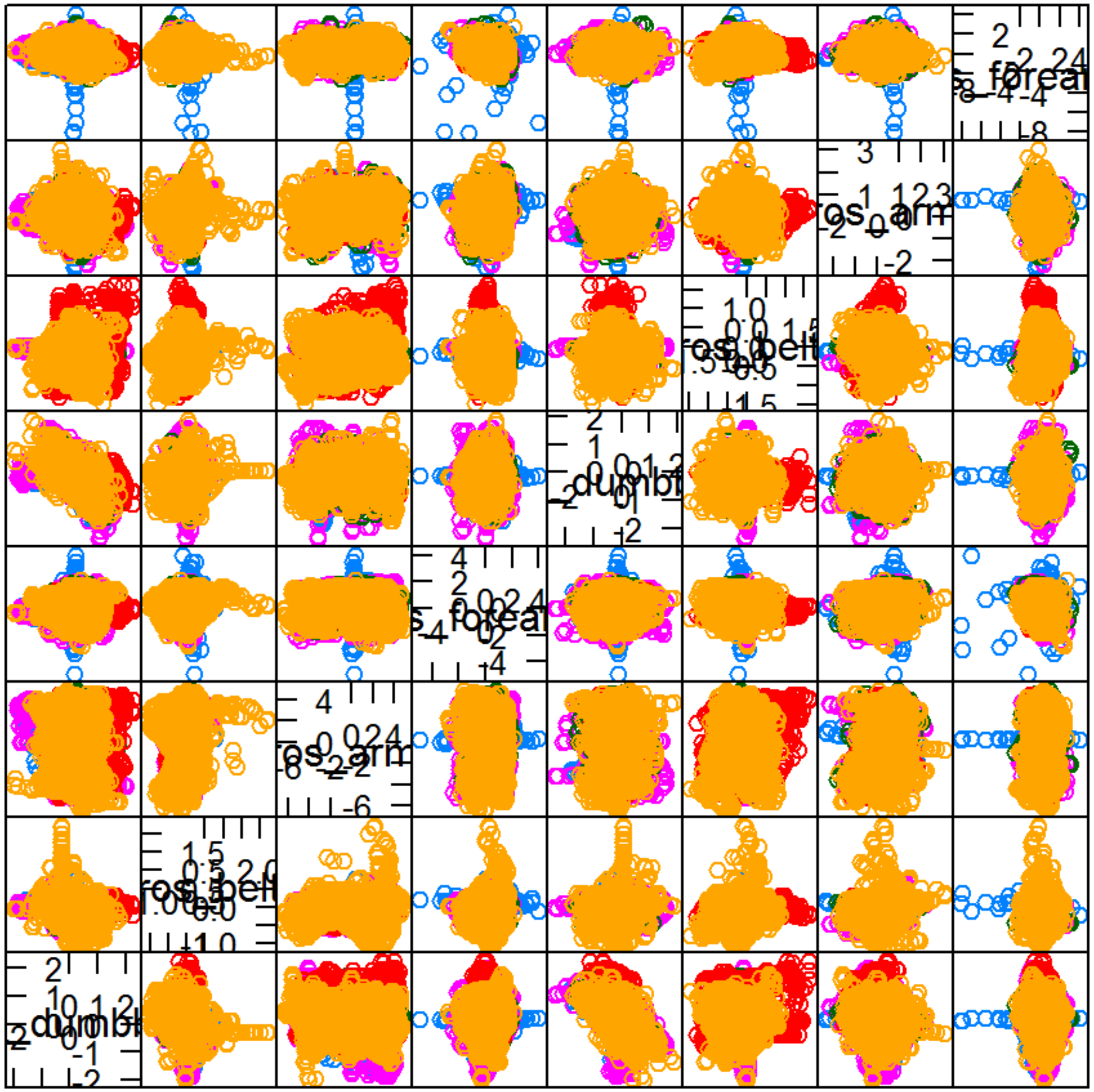
```
se, plot="pairs")
```



As can be seen on the plots the gyros and acceleration data in the X an Z direction shows some good separation between the calsses, while the Y direction is weaker. The Yaw/Pitch/Roll data does not seem to have any predictive power.

Based on this I decided to draw a new plot that focuses on the X and Z gyros readings, to explore more.

```
    library(ggplot2)
    library(caret)
featurePlot(x=training[, c("gyros_dumbbell_x","gyros_belt_x","gyros_arm_x", "gyros_forearm_x",
"gyros_dumbbell_z","gyros_belt_z","gyros_arm_z", "gyros_forearm_z")], y = training$classe, plot
="pairs")
```



Scatter Plot Matrix

From this Plot it seems that the combination of “gyros_dumbbell_x”,“gyros_arm_x”, “gyros_dumbbell_z”,“gyros_belt_z”,“gyros_arm_z” and “gyros_forearm_z” may have a good predictive quality since these readings show relatively good separation of classes.

Performing prediction using different models

I am doing the prediction on the 'training' dataset according to several prdeiction models, and test it on the 'Validation' dataset.

The success rate of each method is measured as number of succesfull predicted classes divided by the total number of Rows in the validatuion data set.

Linear regression model - trying to use this model failed and returned errors for any selection of covariates (results not shown). This is expected since this problem does not have a linear characted, but rather a classification character.

prediction with trees - I tried prediction with trees, on all covariates and a partial set of covariates (based on the covariates investigation above). In both casses the prediction success rate was weak, less than 50%, See results in a table below.

Boosting - I tried gbm boosting, however couldnt make the model produce a model for any subset of covariates. This may be a computational limiation of my machine. results not shown.

Bagging - finally I tried bagging using "treebag" on all covariates. this produced a very high success rate of more than 98.6% which seems satisfatory for this problem.

1. Prediction with trees - all covariates

```
modRpart <- train(classe ~ . ,data =training, method="rpart")
predRpart <- predict(modRpart,validating)

## calculate success rate
res <- data.frame(p <- predRpart, v <- validating$classe)
res$comp <- (res$p == res$v)

rpartPercent <- nrow(res[res$comp,])/nrow(res)
```

2. Prediction with trees - subset of covariates

```
modRpart1 <- train(classe ~ gyros_dumbbell_x +
                       gyros_arm_x + gyros_dumbbell_z + gyros_belt_z +
                       gyros_arm_z + gyros_forearm_z ,data =training, method="rpart")
predRpart1 <- predict(modRpart1,validating)

res <- data.frame(p <- predRpart1, v <- validating$classe)
res$comp <- (res$p == res$v)

rpartPercent1 <- nrow(res[res$comp,])/nrow(res)
```

3. Boosting - reamrked, does not converge to a solution

```
 # modGbm <- train(classe ~ gyros_dumbbell_x + gyros_arm_x +
 #   gyros_dumbbell_z + gyros_belt_z + gyros_arm_z + gyros_forearm_z ,
 #    data=training[sample.int(nrow(training),10000),], method="gbm", verbose=FALSE )
 # predGbm <- predict(modGbm,validating)
 #
 # res <- data.frame(p <- predGbm, v <- validating$classe)
 # res$comp <- (res$p == res$v)
 #
 # gbmPercent <- nrow(res[res$comp,])/nrow(res)
 # print (gbmPercent)
```

4. Bagging - all covariates

```
modBag <- train(classe ~ . ,data =training, method="treebag")
predBag <- predict(modBag,validating)

res <- data.frame(p <- predBag, v <- validating$classe)
res$comp <- (res$p == res$v)
```

```
bagPercent <- nrow(res[res$comp,])/nrow(res)
```

Display cuccess rates of all methods

```
successM<- data.frame(c("trees - all", "trees - partial",  "bagging"),
               c(rpartPercent, rpartPercent1, bagPercent))
colnames(successM) <- c("Model", "Success rate")
print(successM)
```

```
##             Model Success rate
## 1     trees - all    0.4938825
## 2 trees - partial    0.3615416
## 3         bagging    0.9867455
```

predicting the test dataset

As explained before, I am using the bagging model, since it seems to have the highest success rate on the validation data.

Following are the results of calsse prediction on the test dataset:

```
testing <- read.csv("pml-testing.csv")
predtest <- predict(modBag,testing)

data.frame(row_num <- testing$X, predicted_classe <- predtest)
```

```
##    row_num....testing.X predicted_classe....predtest
## 1                     1                            B
## 2                     2                            A
```

```
## 3                      3                            B
## 4                      4                            A
## 5                      5                            A
## 6                      6                            E
## 7                      7                            D
## 8                      8                            B
## 9                      9                            A
## 10                    10                            A
## 11                    11                            B
## 12                    12                            C
## 13                    13                            B
## 14                    14                            A
## 15                    15                            E
## 16                    16                            E
## 17                    17                            A
## 18                    18                            B
## 19                    19                            B
## 20                    20                            B
```

According to the ‘Course Project Prediction Quiz’ this prediction is 100% correct!