Practical Machine Learning

Task

To build the algorithm that could tell wether the exercise was performeed correctly or not using the information from body sensors.

Loading the data:

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.2
```

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

```
library(ggplot2)
train<-read.csv("pml-training.csv")
test<-read.csv("pml-testing.csv") ## since these data do
not have any information on the real class of the performer
we can not use them as a testing set that we need to get
from the train data.frame.
```

Splitting it in the usual fashion:

```
intrain<-createDataPartition(y=train$classe, p=0.75,
list=FALSE)
training<-train[intrain,]
testing<-train[-intrain,]
```

We have made a number of pictures in order to visually represent the data. Some of the figures can be seen in Appendix. We manually pick several featueres that seem to have the most prominent influence on the outcome. An formed a data-sets specifically for them (we mirror this trimming for the test set too).

```
pertrain<-data.frame(training$pitch_belt,
training$min_yaw_belt, training$gyros_belt_y,
training$gyros_belt_x, training$magnet_belt_z,
training$accel_belt_y, training$total_accel_arm,
training$pitch_forearm, training$accel_forearm_y,
training$magnet_forearm_z, training$classe)
colnames(pertrain) <- c('pitch_belt', 'min_yaw_belt',
'gyros_belt_y', 'gyros_belt_x', 'magnet_belt_z',
'accel_belt_y', 'total_accel_arm', 'pitch_forearm',
'accel_forearm_y', 'magnet_forearm_z', 'classe')

pertest<-data.frame(testing$pitch_belt,
testing$min_yaw_belt, testing$gyros_belt_y,
testing$gyros_belt_x, testing$magnet_belt_z,
testing$accel_belt_y, testing$total_accel_arm,
testing$pitch_forearm, testing$accel_forearm_y,
testing$magnet_forearm_z,testing$classe)
pertest <- pertest[complete.cases(pertest),]
colnames(pertest) <- c('pitch_belt', 'min_yaw_belt',
'gyros_belt_y', 'gyros_belt_x', 'magnet_belt_z',
'accel_belt_y', 'total_accel_arm', 'pitch_forearm',
'accel_forearm_y', 'magnet_forearm_z', 'classe')
```

We build a model using GBM method. It is rather time-consuming but it does a bootstrap with 25 repetitions.

```
model1<-train(as.factor(classe)~.,data=pertrain,method =
"gbm",
              ## This last option is actually one
              ## for gbm() that passes through
              verbose = FALSE)
```

```
## Loading required package: gbm
## Loading required package: survival
## Loading required package: splines
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: parallel
## Loaded gbm 2.1
## Loading required package: plyr
```

```
## Warning: variable 33: min_yaw_belt0.9 has no variation.
## Warning: variable 35: min_yaw_belt1.1 has no variation.
## Warning: variable 40: min_yaw_belt1.7 has no variation.
## Warning: variable 42: min_yaw_belt1.9 has no variation.
## Warning: variable 46: min_yaw_belt2.1 has no variation.
## Warning: variable 48: min_yaw_belt2.3 has no variation.
## Warning: variable 50: min_yaw_belt2.5 has no variation.
## Warning: variable 51: min_yaw_belt2.6 has no variation.
```

```
## Stochastic Gradient Boosting
##
## 14718 samples
##    10 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 14718, 14718, 14718, 14718,
14718, 14718, ...
##
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy  Kappa   Accuracy
SD  Kappa SD
##   1                   50      0.5552    0.4388  0.008495
0.010648
##   1                  100      0.6056    0.5027  0.008295
0.010502
##   1                  150      0.6353    0.5402  0.006562
0.008242
##   2                   50      0.6844    0.6022  0.008691
0.010982
##   2                  100      0.7323    0.6627  0.006404
0.008047
##   2                  150      0.7609    0.6986  0.006108
0.007727
##   3                   50      0.7316    0.6619  0.005570
0.007070
##   3                  100      0.7777    0.7196  0.005882
0.007560
##   3                  150      0.8028    0.7512  0.004440
0.005690
##
## Tuning parameter 'shrinkage' was held constant at a
value of 0.1
## 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 and shrinkage = 0.1.
```

```
prediction<-predict(model1,newdata=pertest)
confusionMatrix(prediction,as.factor(pertest$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1188   96   36   17    9
##          B   97  673   63   22   22
##          C   50  111  665   82   30
##          D   38   47   71  659   50
##          E   22   22   20   24  790
##
## Overall Statistics
##
##                Accuracy : 0.811
##                  95% CI : (0.799, 0.821)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.761
##  Mcnemar's Test P-Value : 2.69e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D
Class: E
## Sensitivity             0.852    0.709    0.778    0.820
0.877
## Specificity             0.955    0.948    0.933    0.950
0.978
## Pos Pred Value          0.883    0.767    0.709    0.762
0.900
## Neg Pred Value          0.942    0.931    0.952    0.964
0.972
## Prevalence              0.284    0.194    0.174    0.164
0.184
## Detection Rate          0.242    0.137    0.136    0.134
0.161
## Detection Prevalence    0.274    0.179    0.191    0.176
0.179
## Balanced Accuracy       0.903    0.829    0.855    0.885
0.927
```

The model have overall accuracy of 80% and a good P-value. I am sure it can be improved, but now I do not have much time, unfortunately.

Appendix

Here you can see some plots that we have produced to pic features. We have plotted all the features (and thier influence on classe-variable), but here we give just sevetral examplese (since it is a standard procedure).

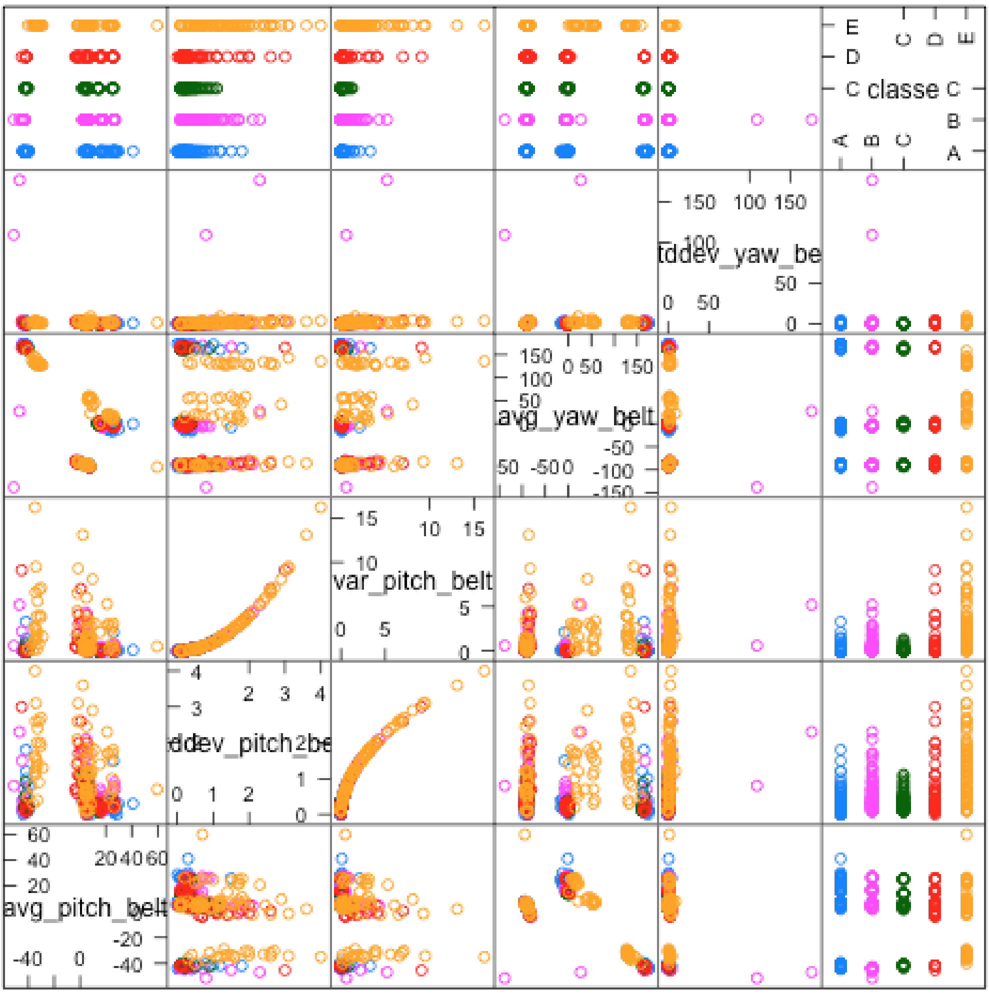
```
plot(features5)
```

```
library(ggplot2)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.2
```

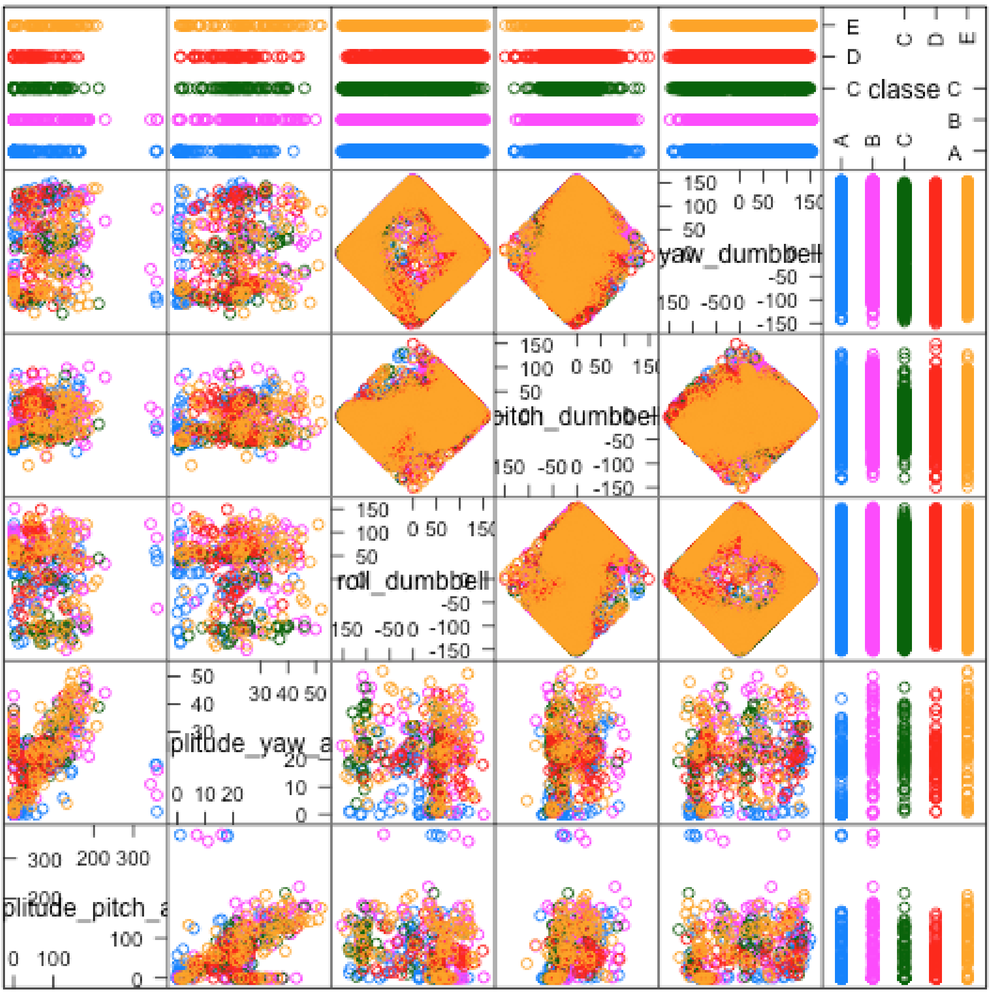
```
## Loading required package: lattice
```

```
train<-read.csv("pml-training.csv")
featurePlot(x=train[,c('avg_pitch_belt',
'stddev_pitch_belt', 'var_pitch_belt', 'avg_yaw_belt',
'stddev_yaw_belt','classe')], y=train$classe,plot="pairs")
```



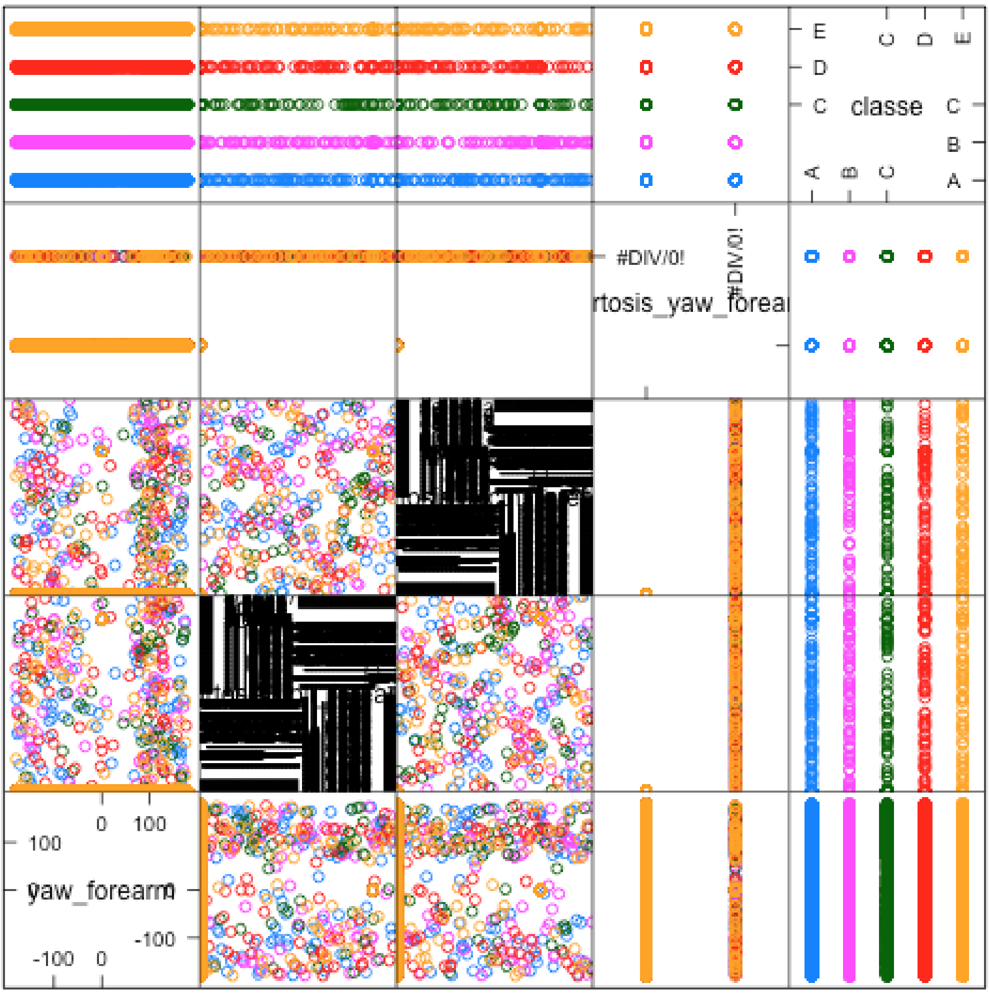
Scatter Plot Matrix

```
featurePlot(x=train[,c('amplitude_pitch_arm',
'amplitude_yaw_arm', 'roll_dumbbell', 'pitch_dumbbell',
'yaw_dumbbell','classe')], y=train$classe,plot="pairs")
```



Scatter Plot Matrix

```
featurePlot(x=train[,c('yaw_forearm',
'kurtosis_roll_forearm', 'kurtosis_picth_forearm',
'kurtosis_yaw_forearm','classe')],
y=train$classe,plot="pairs")
```



Scatter Plot Matrix