ML course project

Jan Vis

26 October 2014

Introduction

New devices made it possible to collect a large amount of data about ones personal activity relatively inexpensively. In this project we are dealing with weight lifting data. The goal is to use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants and predict the quality of the exercises. For additional details see [1].

Exploratory analysis 1: A two in one data set

At a first glance it seems that the data contains a lot of 'NA' values. Looking to the data in a spreadsheet application reveals a large number of columns that only contain data in the row for which the column "new_window" contains the value "yes". It seems these columns contain summarizing data as indicated in [1] section 5.1 and by the names of these columns (starting with e.g. max, avg, var etc.).

As, for some row with 'new_window' equals 'yes', the data in these columns is not pertinent to the sample data in that row (but a summary of data in other rows), it should not be used for a building a prediction model. So we remove these columns.

```
noRows  <- which(train_test$new_window == 'no')    # rows with new_window == 'no'
yesCols <- which (is.na (train_test[noRows[1],]) ) # columns with NA if new_window == 'n
o'
train_testRaw <- train_test[,-yesCols]             # Raw, i.e. excl. the summary data
length(yesCols)
```

```
## [1] 100
```

Exploratory analysis 2: Other redundant columns

Looking to columns, the first 7 are named:

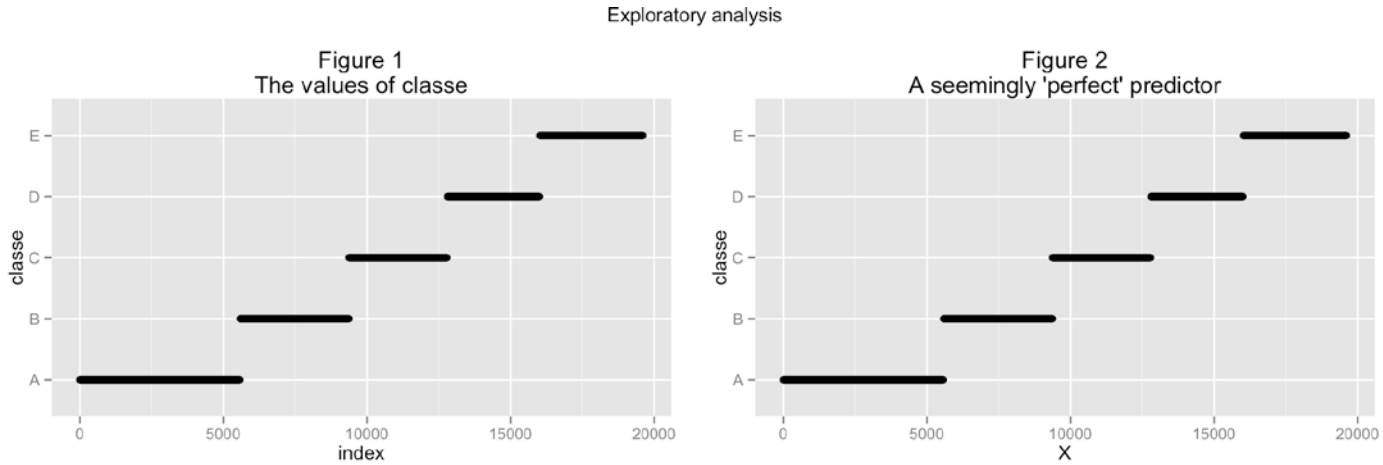
```
colnames(train_testRaw)[1:7]
```

```
## [1] "X"                    "user_name"            "raw_timestamp_part_1"
## [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
## [7] "num_window"
```

As we want to predict the quality of weight lifting based on sensor information, we do not want to make our model dependent on the administrative data like a row number (X), the name of the performer (user_name), time stamps (raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp), new_window (now always 'no') and a window numbering (num_window).

As the most striking example take column ‘X’, which contains a row number. However, as the ‘classe’ column is also ordered, the value of X would be a perfect predictor in a training set. (see figure 1 and 2). However X would perform less then perfect for a validation set which has its own row numbered.

```
cl <- train_testRaw$classe
df <- data.frame (X = train_testRaw$X, classe=cl, index=seq (1,length (cl)))
plot1 <- qplot (index, classe, data=df, main="Figure 1\nThe values of classe")
plot2 <- qplot (X, classe, , data=df, main = "Figure 2\nA seemingly 'perfect' predictor"
)
grid.arrange(plot1, plot2, ncol = 2, main = "Exploratory analysis")
```



So we remove these first 7 columns.

```
train_testRaw <- train_testRaw[,-c(1:7)]
```

Figures 1 shows also that each value of classe is supported with roughly the same number of samples. So, in this respect, we may expect similar performance characteristics for each classe value.

Splitting Data into Training and testing for validation

Before starting to train we need to spit the train_testRaw data into a training set and a test set in order to facilitate validation. I put 75% of the samples int the training set.

```
set.seed (12345678)
index_train <- createDataPartition(y=train_testRaw$classe, p=0.75, list=FALSE)
training <- train_testRaw[index_train,]
testing  <- train_testRaw[-index_train,]
dim(training); dim(testing)
```

```
## [1] 14718    53
```

```
## [1] 4904   53
```

Train a candidate model

Now we are ready to train on our training set. The method chosen is ‘random forrest’. As there are no reasons to assume linearity, a tree model is a good candidate and, within the tree models, random forest is known for its accuracy.

```
    fm <- randomForest (classe~., data=training)
    vi <- importance (fm)[,1]
    vis <- sort(vi, decreasing = TRUE, index.return=TRUE)
    cat ("The twelve most important columns are:",vis$ix[1:12],"\n")
```

```
## The twelve most important columns are: 1 3 41 39 2 38 40 37 27 13 35 12
```

Test the model against the testing data

The prove of the pudding. How well does the model perform when validated against the test data in testing?

```
predictionTesting <- predict(fm,testing)
cmTesting     <- confusionMatrix(predictionTesting, testing$classe)
```

As shown in figure 3 below, the prediction is quite accurate. Only 17 out of sample errors on 4904 samples, i.e. **the out of sample error rate for this test set equals .347 % and the accuracy is 99.653 %.**

```
# some (primitive?) code to plot the error as reported in cmTesting
# The data is reaggenged to comply with a qplot statement I know
err <- cmTesting$table
d <- dim(err)[1]
for (i in 1:d) err[i,i]<-0
perError <- matrix (nrow = sum(err), ncol=2)
colnames (perError) <- c('Predicted classe', 'Actual classe')
classeNames <- c('A', 'B', 'C', 'D', 'E')
n <- 1
for (i in 1:d) for (j in 1:d) if (err[i,j]>0) for (k in 1:err[i,j]) {
    perError [n,1] <- i ; perError[n,2] <- classeNames[j]; n <- n+1
}
df <- data.frame (PredictedClasse=perError[,1], ActualClasse=perError[,2])
qplot(ActualClasse, data=df, fill=PredictedClasse, geom="bar",
      main = 'Figure 3\nNumber of errors for 4903 test samples',
      ylab = 'Number of predicted classe errors',
      xlab = 'Actual classe')
```

For more details the confusion matrix data is provided below.

```
cmTesting
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392    7    0    0    0
##          B    2  942    2    0    0
##          C    0    0  853    3    0
##          D    0    0    0  800    1
##          E    1    0    0    1  900
##
## Overall Statistics
##
##                Accuracy : 0.9965
##                  95% CI : (0.9945, 0.998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9956
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9926   0.9977   0.9950   0.9989
## Specificity            0.9980   0.9990   0.9993   0.9998   0.9995
## Pos Pred Value         0.9950   0.9958   0.9965   0.9988   0.9978
## Neg Pred Value         0.9991   0.9982   0.9995   0.9990   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2838   0.1921   0.1739   0.1631   0.1835
## Detection Prevalence   0.2853   0.1929   0.1746   0.1633   0.1839
## Balanced Accuracy      0.9979   0.9958   0.9985   0.9974   0.9992
```

Literature

[1] Eduardo Velloso e.a. Qualitative Activity Recognition of Weight Lifting Exercises AH'13 4th Augmented Human International Conference, Stuttgart, Germany — March 07-08, 2013 Available at http://groupware.les.inf.puc-rio.br/work.jsf?p1=11201 (http://groupware.les.inf.puc-rio.br/work.jsf?p1=11201)