ML_project

Ondřej Svoboda

19th April 2016

Introduction

This is the final project for Coursera course Practical Machine Learning provided by John Hopkins University. It is part of the Data Science specialization.

Exploratory data analysis

I probably have to reduce the number of descriptors. It looks like lots of them contain mostly NA. these should be obvious candidates for removing.

Another question is how to deal with time variables. I have to look whether there is some pattern. If there wasn't, all would be fine and I could get rid of time. If on the other hand, there is some time pattern, the situation gets significantly more complicated and I would have to take time into account.

I should look for variables which change the most with respect to classe predictor.

Also some form of multiple cross validation will surely be needed

```
training<-read.csv("training.csv",header = TRUE,na.strings = c("NA","","#DIV/0!"))
```

I will get rid of columns which have more than 1/3 of NAs. Whether the choice is 1/3 as is in my case is of course up to discussion, but generally, I want to keep columns which have great variance. Since otherwise I only could replace NAs by some averages and there is only a few real values, I would be replacing them basically for constant which would not contribute to the model anyway.

```
allcols<-names(training)

# vector to be filled with column names which contain some actual values
mycols<-c()

for (colname in allcols){
        #print(colname)
        col=subset(training,select = colname)
        collength=sum(is.na(col))
        collength
        #print(collength)
        if (!(collength > (nrow(training))/3)){
                mycols<-c(mycols,colname)}
}

training2<-subset(training,select=mycols)

#this is just a row number, therefore no use...
training2$X=NULL
```

The dataset training2 now contains much less variables. Doing this, I have got rid of most NAs. Now, I will control whether there some some additional sparse NAs.

```
sum(complete.cases(training2))
```

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

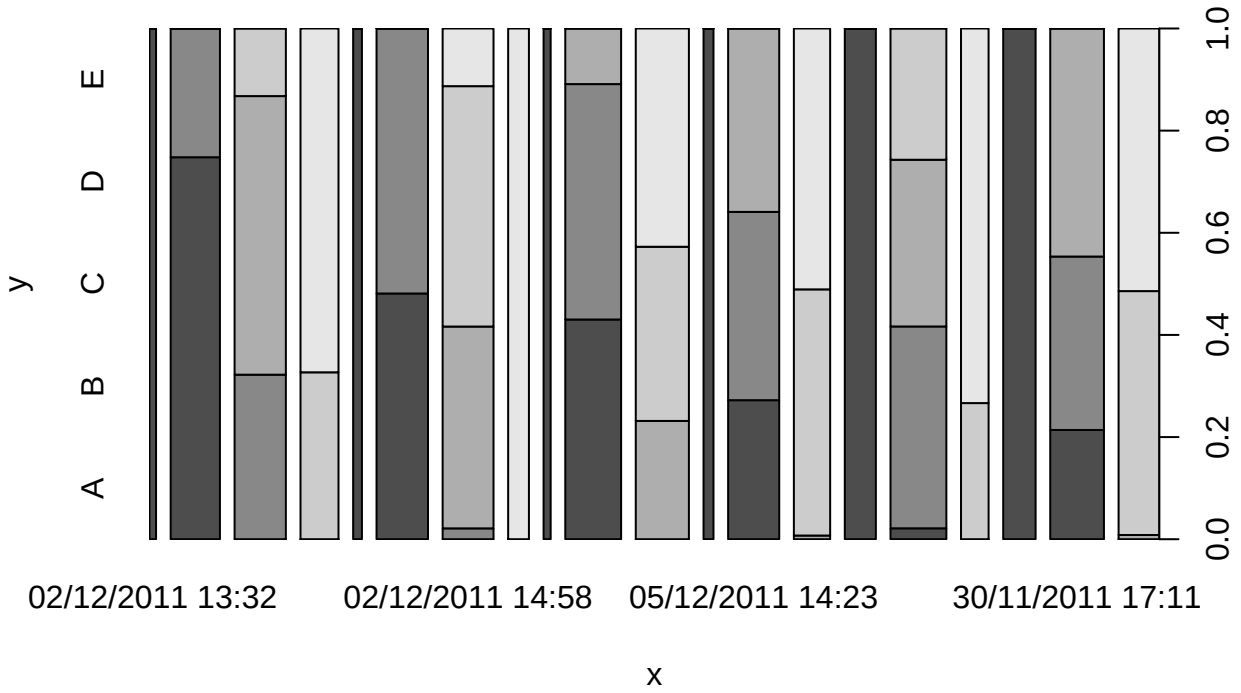
```
dim(training2)
```

```
## [1] 19622    59
```

Since both number give the same result, I do not have to bother with NAs anymore.

Let‘s look at time evolution now.

```
plot(training2$cvtd_timestamp,training2$classe)
```



This qualitative plot shows that for each time frame, the distribution of classes is different. This is unfortunate because it means that I will have to model it as a time series. At the same time it is not unreasonable. The other thing is that in this case, I do not try to do any forecast whatsover, so just maybe it can be possible to use time series information as one of the predictors. Another thing are the raw time data. I can probably safely dismiss these. If I wanted to further improve the model later one, I should give a thought or two right here.

```
training2$raw_timestamp_part_1=NULL
training2$raw_timestamp_part_2=NULL
```

Now, I can try to do prediction on the training set to see whether I am at the very least able to reproduce it correctly on the training set...

Predictive modeling

```
library(caret)
```

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

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

now, let's apply the model. In my case, I will apply the rpart model (regression trees)

```
library(rpart)
modelRpart<-rpart(classe~.,data=training2,method="class")
confusionMatrix(predict(modelRpart,type="class"),reference=training2$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5369  596    0    0    0
##          B   75 2721  294  159    0
##          C  136  457 3070  498  140
##          D    0   23   32 2047  198
##          E    0    0   26  512 3269
##
## Overall Statistics
##
##                Accuracy : 0.8397
##                  95% CI : (0.8345, 0.8448)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7966
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9622   0.7166   0.8971   0.6365   0.9063
## Specificity            0.9576   0.9666   0.9240   0.9846   0.9664
## Pos Pred Value         0.9001   0.8375   0.7138   0.8900   0.8587
## Neg Pred Value         0.9846   0.9343   0.9770   0.9325   0.9786
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2736   0.1387   0.1565   0.1043   0.1666
## Detection Prevalence   0.3040   0.1656   0.2192   0.1172   0.1940
## Balanced Accuracy      0.9599   0.8416   0.9106   0.8105   0.9363
```

Looking at the training set, I see that it can predict roughly 85 % of the cases correctly. It is by no means not the best model in the world, but it will suffice for the purposes of my presentation.

Now, I would like to process the testing dataset in exactly the same way...

```
testing<-read.csv("testing.csv",header = TRUE,na.strings = c("NA","","#DIV/0!"))
variables<-names(training2)
testing2<-subset(testing,select=subset(variables,subset = variables!="classe"))
predict(modelRpart,newdata = testing2,type="class")
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  C  A  A  E  D  C  A  A  B  C  B  A  E  E  A  A  B  B
## Levels: A B C D E
```

Conclusion

Using the tree based model, I was able to correctly identify 17 out of 20 outcomes. This suggests the model was by no means overfitted; it performed equally well for both the training set and the test set.