Random Forests and Ensemble methods **Big Data Management and Analytics**

Exercise: Bank Marketing Random Forests and ensemble methods

Data cooking

Bank marketing dataset is being used to generate RF models, data is loaded from file *bank-full.csv*, then cooked as in previous practice, with just *log10* most of times, scale and center.

Model

First we try a standard classification tree (CART). r*part* library is used to build and plot the tree:

```
Classification tree:
rpart(formula = subscribed ~ ., data = learn.data, control = rpart.control(cp = 1e-04))

Variables actually used in tree construction:
 [1] age       balance   campaign  contact   day       education housing   job       marital
month
[11] pdays     poutcome  previous
```

A model is built and pruned using best *cp*, obtaining the following:

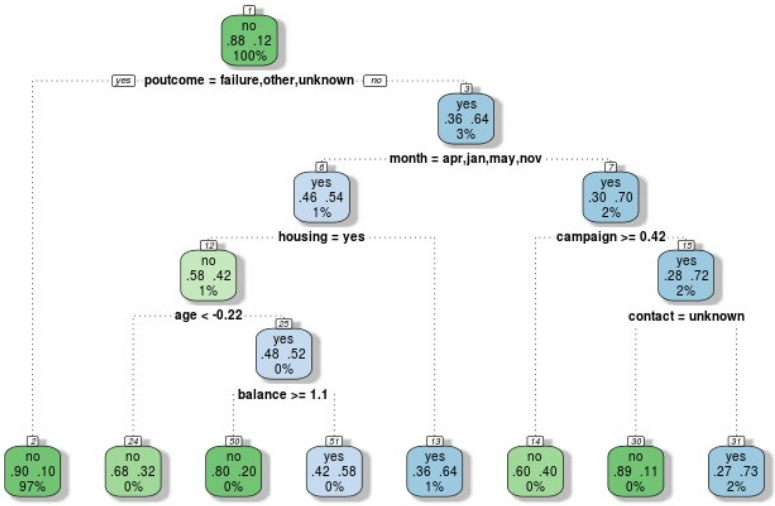

Figure 1 - standard classification tree

Getting the following confusion matrix:

	Pred:no	Pred:yes
Actual:no	26330	271
Actual:yes	2922	618

Leading to: `total correct: 89.25017 %`
→ Error: **10.74983 %**

With RF we get to improve that a little bit (with *ntree=300*):

number of trees	*error rate*
100	*10.84%*
200	*10.76%*
300	***10.63%***
400	*10.71%*
500	*10.70%*
600	*10.72%*

```
randomForest(formula = subscribed ~ ., data = learn.data, ntree = 300, proximity = FALSE)
               Type of random forest: classification
                     Number of trees: 300
No. of variables tried at each split: 4

        OOB estimate of  error rate: 10.63%
Confusion matrix:
       no yes class.error
no  26001 592   0.0222615
yes  2613 935   0.7364713
```

We can obtain even better results using *tuneRF* to tune *mtry* parameter:

```
bestmtry <- tuneRF(learn.data[-17], learn.data$subscribed, ntreeTry=100,
stepFactor=1.5,improve=0.01, trace=TRUE, plot=TRUE, dobest=FALSE)
```

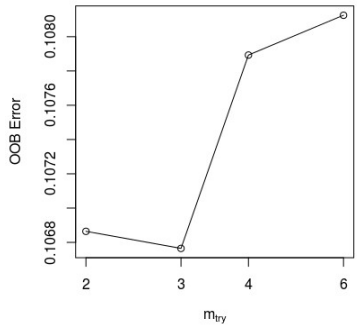

Figure 2 - m_{try} tuning

Getting *mtry=3*, used to generate a new tree:

```
Call:
 randomForest(formula = subscribed ~ ., data = learn.data,
ntreeTry = 1000,      mtry = 3, proximity = FALSE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 10.61%
Confusion matrix:
       no         yes    class.error
no  26159 (TN)  453 (FP)   0.0170224
yes  2744 (FN)  785 (TP)   0.7775574
```

Error rate is improved, but classification error is still highly unbalanced.

However, if we try to balance classes classification error, we would obtain worse total error rate, for example *randomForest* with *classwt = c(1, 100)*:

```
    OOB estimate of  error rate: 45.1%
Confusion matrix:
      no     yes     class.error
no  14320  12292   0.4618969
yes  1303   2226   0.3692264
```

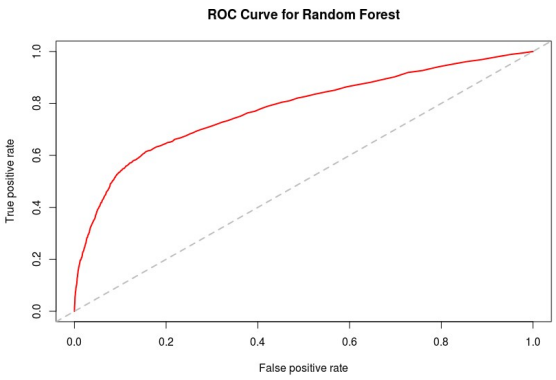

Figure 3 - ROC curve for model with error 10.61%

Importance of variables in Random Forest model can be observed.

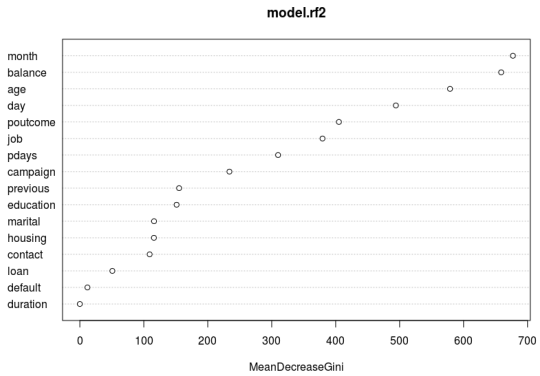

Figure 4 - variable importance as measured by a Random Forest

Comparison with other algorithms

Algorithm	*Accuracy*
LDA	*13.66%*
LDA_CV	*13.32%*
RDA	*14.10%*
RDA_CV	*14.25%*
QDA	*14.11%*
Logistic regression	***9.78%***
SVM *linear* (cost=1, gamma=1, epsilon=0.1)	*10.72%*
RF (ntree=300, m_{try}=3)	*10.61%*

Logistic regression is more accurate, however probably *RF* is faster at predictions, then, this compromise should be solved as required for specific cases.