Закиров Д.

КОКУС ОРМОНДОР (RANDOM FORESTS) АЛГОРИТМИ НЕГИЗИНДЕ КРЕДИТТИК СКОРИНГ: НАТЫЙЖАЛУУ ЭМПИРИКАЛЫК МИСАЛ

Закиров Д.

КРЕДИТНЫЙ СКОРИНГ ОСНОВАННЫЙ НА АЛГОРИТМЕ СЛУЧАЙНОГО ЛЕСА (RANDOM FORESTS): ЭФФЕКТИВНЫЙ ЭМПИРИЧЕСКИЙ ПРИМЕР

D. Zakirov

CREDIT SCORING BASED ON RANDOM FORESTS ALGORITHM: AN EFFECTIVE EMPIRICAL EXAMPLE

УДК: 005.95:339.137.2

Бул изилдөө Кокус Ормондор алгоритминде негизделген кредиттик скоринг үлгүсүн тактыгын баалоого багытталып, код жана графикалык визуализация колдоосу менен толук сүрөттөлгөн. Кредиттик скоринг банк секторунун жана каржы институттарынын эн маанилүү процесстеринин бири болуп эсептелет. Ал кардарлардын насыяны төлөөгө жөндөмсүздүгүн алдын алуу жана банкроттук коркунучун азайтуу үчүн багытталган. Ошентип, бул документ жогоруда тизмеленген ыкманы пайдалануу менен кредиттик скоринг үчүн жакшы чечим сунуштайт.

***Негизги сөздөр:** кредиттик скоринг, Кокус Ормондор, маалымат талдоо, R программалоо тили.*

Данное исследование направлено на оценку точности модели кредитного скоринга, основанной на алгоритме Случайного Леса с подробным описанием в коде и графических визуализаций. Кредитный Скоринг является одним из наиболее важным процессом в банковском секторе и финансовых институтов. Он направлен на предотвращение неплатежеспособности клиента при оплате кредитов и свести к минимуму риск банкротства. Таким образом, этот документ предлагает хорошее решение кредитного скоринга с использованием выше описанного метода.

***Ключевые слова:** кредитный скоринг, Случайные Леса, анализ данных, язык программирования R.*

This research paper aims to evaluate the accuracy of credit scoring model based on Random Forest algorithm with a detailed description in code and graphical visualizations. Credit Scoring is one of the most significant process in the banking sector and financial institutions. It aims to prevent customer failure when paying loans and minimize risk of defaulting. Therefore, this paper proposes a good solution to the credit scoring using the above method.

***Key words:** credit scoring, random forest, data mining, R programming.*

Introduction

This paper is third in a series of articles related with data mining and credit scoring. First article was about application of data mining in the banking sector where we discussed about data mining principles and techniques and in addition about main areas of data mining used in processes of banking (card frauds, marketing, risk management, credit scoring etc.).

The second article was the literature review of data mining techniques but with broader definitions and usage of them in retail credit scoring. We use dataset in order to understand how each reviewed data mining technique shows the result.

The current paper is mostly an empirical example based article as we will review real life dataset and its output after transformation by using Random Forests data mining method by applying R programming features. As we have already discussed about data mining and its methods and credit scoring in our previous two articles we will not go into detail about what is data mining and what are techniques used in it.

The paper will briefly describe Random Forest algorithm at the beginning and then we pass on practical side of the research. We will examine dataset and development environment, which is R programming in our case. Later we will look in detail chunks of code parts where we use random forest techniques to analyze and calculate credit scoring results of given dataset.

Random Forests

Random Forest is a many-sided and flexible machine learning method capable of performing both regression and classification tasks. It also undertakes dimensional reduction methods, treats missing values, outlier values and other essential steps of data exploration, and does a fairly good job. Random Forest is a kind of ensemble learning method, where a group of weak models combine to form a powerful model [1].

The history began way back in 1980s. This algorithm was a result of incessant team work by Leo Breiman, Adele Cutler, Ho Tin Kam, Dietterich, Amit and Geman. Each of them played a significant role in early development of random forest. The algorithm for actuating a random forest was developed by Leo Breiman and Adele Cutler. This algorithm is register in their (Leo and Adele) trademark. Amit, Gemen, Ho Tim Kam, independently introduced the idea of random selection of features and using Breiman's 'Bagging' idea constructed this collection of decision trees with controlled variance. Later, Deitterich introduced the idea of random node optimization [2].

In Random Forest, we grow multiple trees as opposed to a single tree in CART (Classification And Regression Trees) model. To classify a new object based on attributes, each tree gives a classification and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest) and in case of regression, it takes the average of outputs by different trees.

The strengths of Random Forests are that their runtimes are quite fast, and they are able to deal with unbalanced and missing data. Random Forest weaknesses are that when used for regression they cannot predict beyond the range in the training data, and that they may over-fit data sets that are particularly noisy. Of course, the best test of any algorithm is how well it works upon your own data set [3].

Implementation of Random Forests Model in R

As the first step we read the data from csv file and do some basic data cleaning. This is important as this dataset will be our training dataset and we need to have cleanest possible data.

```
dmr_data <-
read.csv("C:/Users/dilmuratz/GoogleDrive/PHD/thesis/my_topic/my_project_randomf/dmr_new.csv",header=T)
head(dmr_data)
```

We will use ready package randomForest in R [4]

```
library(randomForest)
```

As the main plotting package we will include ggplot2 [5]

```
library(ggplot2)
```

As it is already known machine learning and data mining techniques use two types of dataset when modeling: training and test dataset. Below we divide data into training and test dataset.

```
set.seed(12345)
#set.seed(43353)
#set.seed(4645767685)
smp_size1 <- floor(0.75 * nrow(dmr_data2))
train_index1 <- sample(seq_len(nrow(dmr_data2)), size = smp_size1)
trainX <-X[train_index1, ]
trainY<-Y[train_index1]
testX<-X[-train_index1,]
testY <- Y[-train_index1 ]
nmin <- sum(trainY == "Rejected")
```

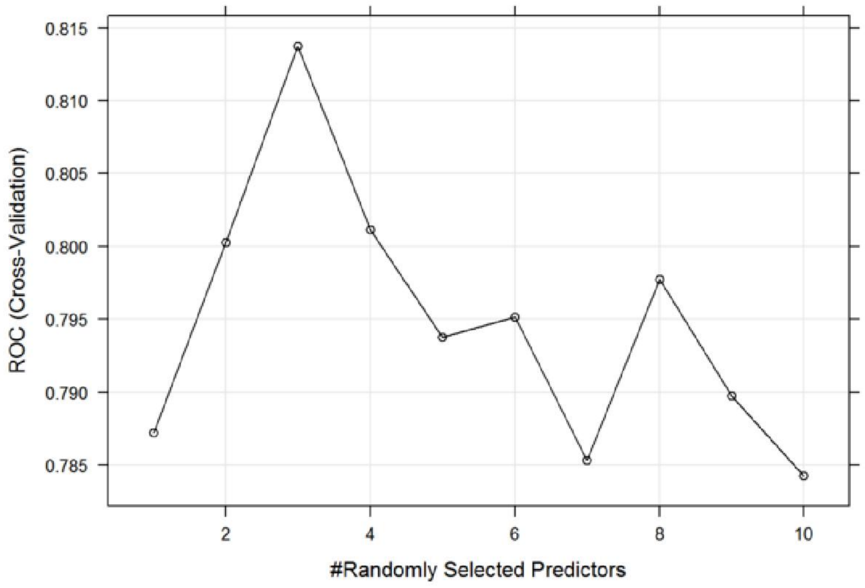
Now below we train two types of random forest. rf_model is the model without down sampling and rf_model_US is model with down-sampling.

```
rf_model_US<-
train(trainX,trainY,method="rf",trControl=trainControl(method="cv",number=10,classProbs=TRUE,summaryFunction
=twoClassSummary),metric="ROC",allowParallel=TRUE,tuneGrid=expand.grid(mtry=c(1,2,3,4,5,6,7,8,9,10)),ntree =
1500,strata=trainY,sampsize=rep(nmin,2))

rf_model<-
train(trainX,trainY,method="rf",trControl=trainControl(method="cv",number=10,classProbs=TRUE,summaryFunction
=twoClassSummary),metric="ROC",allowParallel=TRUE,tuneGrid=expand.grid(mtry=c(1,2,3,4,5,6,7,8,9,10)),ntree =
1500)
```

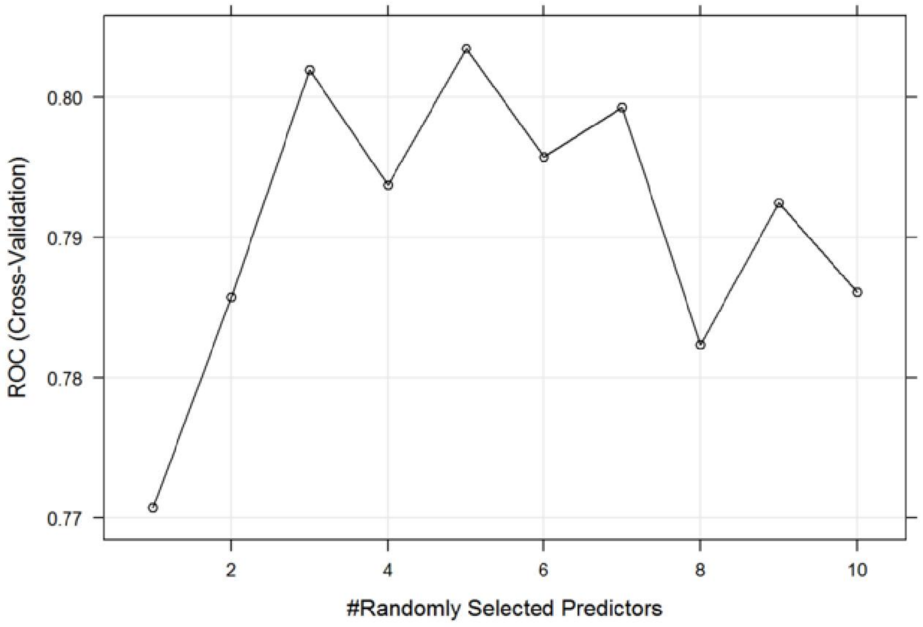
Printing details of rf_model_US (down sampled randomForest) and best rf_model_US. Both the randomForest models are built to try 1 to 10 variables given by mtry parameter.

```
trellis.par.set(caretTheme())
plot(rf_model, metric = "ROC")
```



Predicting on the test set using rf_model.

```
trellis.par.set(caretTheme())
plot(rf_model_US, metric = "ROC")
```



```
rf_test<-predict(rf_model, testX)
table(rf_test,testY)
```

```
##          testY
## rf_test    Approved Rejected
##   Approved       70        2
##   Rejected        0        6
```

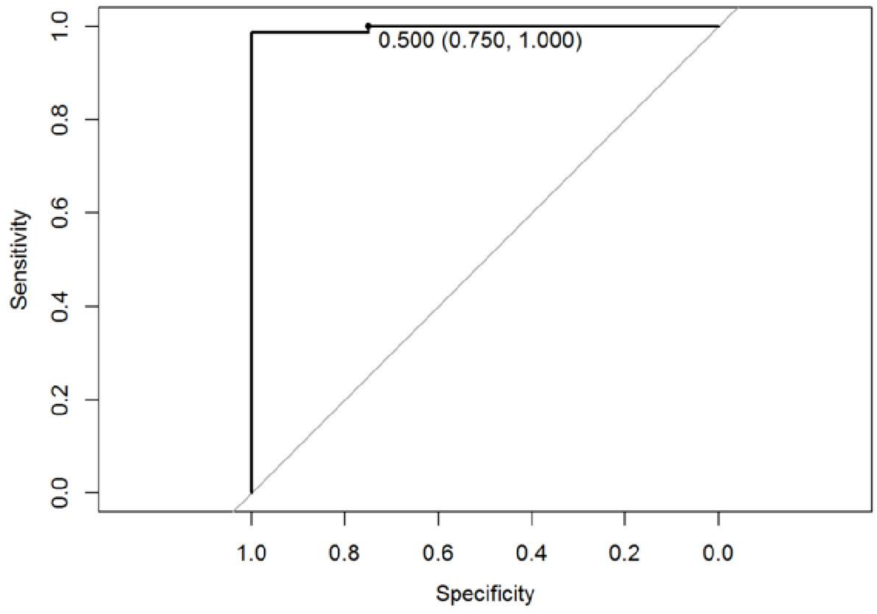
Predicting on the test set using rf_model_US.

```
rf_test_US<-predict(rf_model_US, testX)
table(rf_test_US,testY)
```

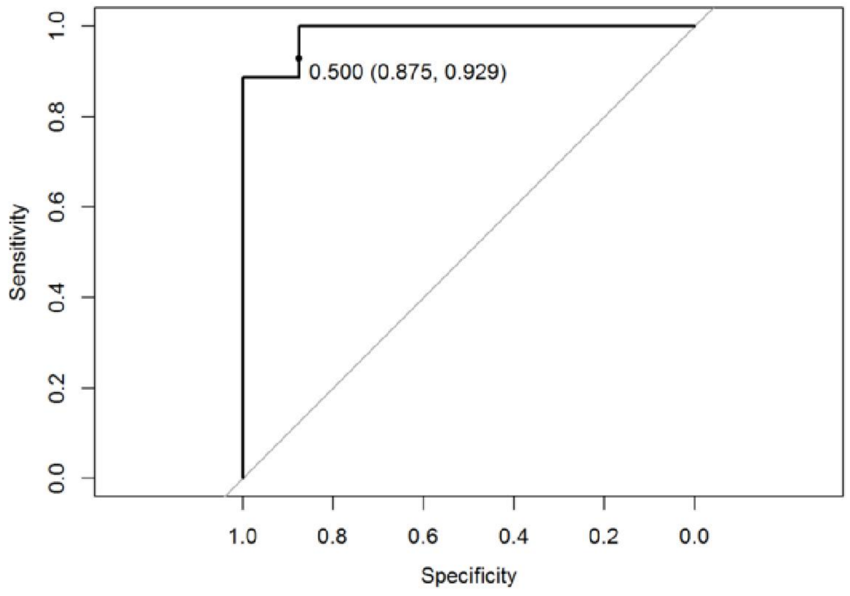
```
##          testY
## rf_test_US Approved Rejected
##   Approved       65        1
##   Rejected        5        7
```

Plotting ROC curves for both the models.

```
rf_roc <- roc(testY,
          predict(rf_model, testX, type = "prob")[,1],
          levels = rev(levels(testY)))
plot(rf_roc,print.thres=c(.5))
```



```
rf_roc_US<- roc(testY,
          predict(rf_model_US, testX, type = "prob")[,1],
          levels = rev(levels(testY)))
plot(rf_roc_US,print.thres=c(.5))
```



```
##
## Call:
## roc.default(response = testY, predictor = predict(rf_model_US,     testX, type = "prob")[, 1], levels = rev(levels(testY)))
##
## Data: predict(rf_model_US, testX, type = "prob")[, 1] in 8 controls (testY Rejected) < 70 cases (testY Approved).
## Area under the curve: 0.9857
```

So from the above conclusions we can make decision that we will use rf_model_US classifier as our modeling algorithm.

Conclusion

In this article, we looked at one of most common machine learning algorithm Random Forest followed by its pros & cons, method to tune parameters and explained & compared it using a practice problem. I would suggest to use random forest and analyse the power of this model by tuning the parameters. Also it is highly recommended to note that dataset quality is very important while modeling data thus data preparation phase should not be omitted when analyzing.

References:

1. Breiman L., Random Forests. Machine Learning, October 2001, Volume 45, Issue 1, pp 5-32.
2. Biau G., Scornet E., A Random Forest Guided Tour, Sorbonne Universit´es, UPMC Univ Paris, November 2015
3. [ONLINE]http://www.analyticsvidhya.com/blog/2015/09/random-forest-algorithm-multiple-challenges/
4. randomForest R package https://cran.r-project.org/web/packages/randomForest/randomForest.pdf
5. ggplot2 R package https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf

Рецензент: к.э.н. Мамадиев Б.Ы.