Rishabh Taneja

Northeastern University

ALY 6020: Predictive Analytics

FALL ‘18

Instructor Name: Justin Rodgers

**Part A: Decision Trees**:

1. Please work on the same Example " identifying risky bank loans using C5.0 decision trees" from Chapter 5 of the textbook “Machine Learning with R - Second Edition”, located in pages136-149. Textbook is attached to this assignment folder. Use the same dataset (credit.csv), which has also been acquired for you and attached to this assignment folder. Further information on this data can be found at the UCI Machine Learning Data Repository (http://archive.ics.uci.edu/ml) by Hans Hofmann of the University of Hamburg. The dataset contains information on loans obtained from a credit agency in Germany.
2. Follow the same five steps to get the same results as in the text.

**INTRODUCTION**:

The below example is an illustration of C5.0 decision trees algorithm which is used to identify risky bank loans. C5.0 is used because of the benefits it provides which is, that it is quite opinionated towards pruning. In other words, it automatically takes care of the decisions using reasonable defaults. The main moto of this algorithm is to post-prune the tree. Once the tree is grown quite large, the nodes and branches which have very minimal effect on errors are removed. In some cases, branches are replaced by simple decisions, termed as subtree replacement.

**METHOD**:

We use C5.0 decision tree algorithm which uses pruning mechanism to get rid of branches which are useless and replace it with the meaningful ones. This prevents from overfitting. The main reason to use this algorithm is that it is quite easy to implement and quite feasible to adjust the training options.

* 1. **Collecting data**

The credit dataset includes 1,000 examples on loans, plus a set of numeric and nominal features indicating the characteristics of the loan and the loan applicant. A class variable indicates whether the loan went into default.

* 1. **Exploring the data**

credit <- read.csv(file.choose(), header = TRUE)

str(credit)

A close up of a keyboard

Description automatically generated

From the above results, we can see that there are 1000 observations with 17 variables out of which 10 are factors and 7 and integers.

table(credit$checking\_balance)

A screenshot of a cell phone

Description automatically generated

table(credit$savings\_balance)



summary(credit$months\_loan\_duration)

A close up of a screen

Description automatically generated

summary(credit$amount)

A close up of a screen

Description automatically generated

The loan amount ranged from 250 to 18424 DM across 4 to 72 months.

summary(credit$default)



We are using this algorithm to focus on number of defaults and analyze who are at high risk and thus prevent from giving them the credit requests.

**Data preparation**

set.seed(123)

Seed set at 123 for consistent output.

train\_sample <- sample(1000, 900)

We are using the 90:10 ratio for train and test respectively.

str(train\_sample)

credit\_train <- credit[train\_sample, ]

credit\_test <- credit[-train\_sample, ]

prop.table(table(credit\_train$default))

A close up of a logo

Description automatically generated

prop.table(table(credit\_test$default))

A picture containing object

Description automatically generated

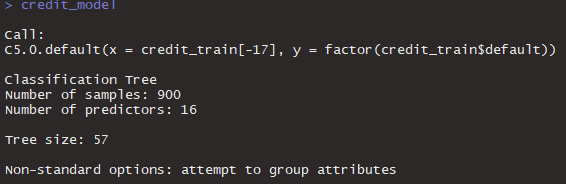
From the above results, it seems that the data is evenly split so we can move forward with the analysis.

* 1. **Training a model on data**

library(C50)

credit\_model <- C5.0(credit\_train[-17], factor(credit\_train$default))

credit\_model



From the above results, we can note the number of samples are 900 and the number of predictors 16. The factor() function is used since the C5.0 explicitly work on factor variables. The tree size generated is 57 which shows the depth of the tree.

summary(credit\_model)

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post with text and a black background

Description automatically generated

The error percentage output tells us that there were 133 wrongly predicted values out of possible 900 which amounts to about 14.8% of the total.

This breaks down to 35 No values were predicted as Yes and 98 Yes values were predicted as No.

* 1. **Evaluating model performance**

credit\_pred <- predict(credit\_model, credit\_test)

library(gmodels)

CrossTable(credit\_test$default, credit\_pred,

prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,

dnn = c('actual default', 'predicted default'))

A screenshot of a computer

Description automatically generated

The cross-table is used here to evaluate the accuracy of the model.

The test loan application records were used in this case where out of total 100, 14 did default and 59 did not, which were correctly predicted by the model.

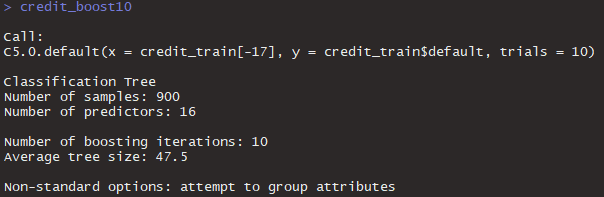
Thus, the accuracy is about 59+14/100 = 73/100 = 0.73 or 73% which is quite a good sign for the model overall. However, when we consider individual predictions, the model only predicted 14 out of 33 values as default, which is only 14/33= 0.424 or 42.4%. This is not a good sign and can be quite costly for the company.

* 1. **Improving model performance**

credit\_boost10 <- C5.0(credit\_train[-17], credit\_train$default,

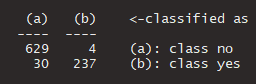
trials = 10)

credit\_boost10



credit\_boost\_pred10 <- predict(credit\_boost10, credit\_test)

Summary(credit\_boost10)



From the results, we can see that the classifier correctly predicted 866/900 = 96.2% values. However, it wrongly predicted 34/900 = 3.8% values which is a good sign. It remains to be seen how it will perform on the test dataset.

CrossTable(credit\_test$default, credit\_boost\_pred10,

prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,

dnn = c('actual default', 'predicted default'))

A screenshot of a cell phone

Description automatically generated

The cross-table is used here to evaluate the accuracy of the model.

The test loan application records were used in this case where out of total 100, 20 did default and 62 did not, which were correctly predicted by the model.

Thus, the accuracy is about 20+62/100 = 82/100 = 0.82 or 82% which is quite a good sign for the model overall. However, when we consider individual predictions, the model only predicted 20 out of 33 values as default, which is only 20/33= 0.606 or 60.6%. There is some improvement, but this is not a good sign and can be quite costly for the company. This may be because of quite small training dataset.

matrix\_dimensions <- list(c("no", "yes"), c("no", "yes"))

names(matrix\_dimensions) <- c("predicted", "actual")

matrix\_dimensions

A screenshot of a cell phone

Description automatically generated

error\_cost <- matrix(c(0, 1, 4, 0), nrow = 2,

dimnames = matrix\_dimensions)

error\_cost

A screenshot of a cell phone

Description automatically generated

As we see, there is no cost defined if the classifier predicts the values correctly but if it does not, it has cost to it which is 4 if actual is yes but prediction is no and it is 1, if the actual is no and predicted is yes.

credit\_cost <- C5.0(credit\_train[-17], credit\_train$default,

costs = error\_cost)

credit\_cost\_pred <- predict(credit\_cost, credit\_test)

CrossTable(credit\_test$default, credit\_cost\_pred,

prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,

dnn = c('actual default', 'predicted default'))

A screenshot of a computer screen

Description automatically generated

After the boosting and testing it on test dataset, we can notice that the model has become more worse. With correctly predicting 59 not defaults earlier, it has come down to 37. The total correct predictions are 37+26/100 = 0.63 or 63% which has also come down overall. However, the only improvement is in the correctly predicted defaults which is 26/33 = 78.7%.

1. Give a good summary and conclusion of your findings with insight

**SUMMARY and CONCLUSION:** From the above results, we can note a few things which are that the loan amount ranged from 250 to 18424 DM across 4 to 72 months. We are using this algorithm to focus on number of defaults and analyze who are at high risk and thus prevent from giving them the credit requests. the number of samples are 900 and the number of predictors 16. The factor() function is used since the C5.0 explicitly work on factor variables. The tree size generated before the iterations is 57 which shows the depth of the tree. After the boosting and iteration process, the size of the tree reduced to 47.

From the summary of the credit\_model, we can note that the classifier will denote it as “not likely to default” in case the checking account balance is unknown or greater than 200 DM(currency),else the balance would be less than 0 or between 1 and 200. If the credit history is good, classify as “likely default”. After the evaluation of training dataset, the error percentage output tells us that there were 133 wrongly predicted values out of possible 900 which amounts to about 14.8% of the total. This breaks down to 35 No values were predicted as Yes and 98 Yes values were predicted as No.

**DISCUSSION**: Before boosting –

The test loan application records were used in this case where out of total 100, 14 did default and 59 did not, which were correctly predicted by the model. Thus, the accuracy is about 59+14/100 = 73/100 = 0.73 or 73% which is quite a good sign for the model overall. However, when we consider individual predictions, the model only predicted 14 out of 33 values as default, which is only 14/33= 0.424 or 42.4%. This is not a good sign and can be quite costly for the company.

After boosting –

The test loan application records were used in this case where out of total 100, 20 did default and 62 did not, which were correctly predicted by the model.

Thus, the accuracy is about 20+62/100 = 82/100 = 0.82 or 82% which is quite a good sign for the model overall. However, when we consider individual predictions, the model only predicted 20 out of 33 values as default, which is only 20/33= 0.606 or 60.6%. There is some improvement, but this is not a good sign and can be quite costly for the company. This may be because of quite small training dataset.

After the boosting and testing it on a test dataset, we can notice that the model has become more worse. With correctly predicting 59 not defaults earlier, it has come down to 37. The total correct predictions are 37+26/100 = 0.63 or 63% which has also come down overall. However, the only improvement is in the correctly predicted defaults which is 26/33 = 78.7%.

**REFERENCES:**

* 1. Lantz, B. (n.d.). Machine Learning with R - Second Edition. Retrieved from <https://www.oreilly.com/library/view/machine-learning-with/9781784393908/>
  2. (n.d.). Retrieved from <https://www.ibm.com/support/knowledgecenter/en/SS3RA7_15.0.0/com.ibm.spss.modeler.help/c50node_general.htm>
  3. Donges, N., & Donges, N. (2018, February 22). The Random Forest Algorithm. Retrieved from <https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd>