The University of Akron

College of Business, Department of Management

Advanced Data Analytics Topics (ISM:663)

Project 5

Identify Risky Bank Loans using C5.0 Algorithm

By:

Parwan Sharma

Student id – 4808380

MSM Business Analytics

Abstract

This Project is based on the use of C5.0 algorithm which is one of the most well known and widely used decision tree algorithm. It is one of the most widely implemented supervised machine learning algorithms to make industry standard decision trees and is preferred over other machine learning models like neural network etc. The data used in this project is called “credit.csv” a dataset based on the loan information of a German credit agency. It includes a total of 1000 cases with a number of variables.

In this project we will use this data and create a decision tree using the C5.0 algorithm and analyze which variables can be good predictors whether a loan goes into default or not. We will prepare and handle data, train the model and then evaluate and improve the performance by boosting the accuracy.

Introduction

**Decision Tree**

It is a type of machine learning and data mining algorithm which is a represents the relationship between major features and potential outcomes of a model. It is called a tree as it looks like an inverted tree with a wide trunk which gradually narrows down to a branches. What it represents is a wide data which gradually splits into a degree of subsets based on the purity of the set.

The tree begins with a root node which is topmost or the starting point of a decision tree. It is followed by Decision nodes which is are the choice made by the algorithm. This choice leads to the creation of branches which represents outcomes. These branches are further divided by the algorithm to achieve the maximum purity. In cases where the algorithm achieves outcomes with high purity, the branch gets terminated. These end points are called leaf or terminal nodes.

This algorithm leads to model which has a flow chart like structure making it easy to read and understand, making it more transparent from a legal point of view which makes it a primary choice for credit scoring models, market studies for customer behaviour and diagnosis of medical conditions.

**C5.0 Algorithm**

There are a number of implementations of decision tree and out of all of those the most widely known and used is C5.0 algorithm. It is the most used decision tree algorithm due to its major advantages over other data based machine learning models such as neural network and Support vector machines, which are both black box methods(highly sophisticated mathematics which is hard to comprehend). As this models performs nearly well compared to other option and at the same time is much easier to understand making it an industry standard to produce decision tree.

Some of the major advantage is that it is all purpose classifier with a highly automatic learning process that can handle both numeric or nominal features. The result generated by this model is easy to interpret for both small and large data set.

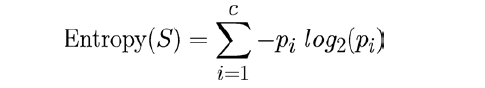
Some of the major disadvantages of this model is that it is highly biased toward features which have large magnitude. The model is highly prone to over fitting or under fitting. The model sometimes produce large trees which can be difficult to interpret. Often slight changes in the training data set leads to substantial change in the decision logic.

**Feature selection for tree building using Entropy**

The efficiency of a decision tree is based upon its ability to identify which feature it should split upon. This is done by the algorithm based on the purity of the class. Purity can be defined as the number of classes are present in a subset. that is if a subset contains only one class it would be called pure.

The C 5.0 uses a concept called entropy which quantifies disorder or randomness within a class. Therefore the higher the entropy is the more diverse it is in nature. Therefore set with high entropy will provide little information. The algorithm will try to find the best split to reduce entropy to increase similarity within the groups.

The entropy of a dataset with n classes varies between 0 and log2(n). When the entropy is at its minimum, it means that the sample is entirely homogeneous, while the maximum value of entropy indicates that the data are highly diverse, and no group has even a slight majority.



The total entropy is the sum of each of the entropy of the n partitions weighted by proportions. If the information gained from a set is zero, then there is no point in making the split as it will not cause a reduction in entropy.

**Pruning the resulting tree**

In cases where the number of features or variables affecting the predictions are high in number, the decision tree may grow indefinitely by choosing splits in order to reduce the entropy. A highly split tree with high number of branches will become overly specific and will be overfitted. Also, a large tree would be difficult to interpret making it counterintuitive. To avoid this situation we use a process called pruning, which reduce the size of a tree. There are two ways of pruning a decision tree:

* Pre-Pruning

When we stop a tree from growing once it reaches a set number of decision nodes which contain only few examples. One major downside of this step is that it would not be clear if the tree misses small but important patterns.

* Post-Pruning

In this case the tree is allowed to grow too large in size, so it achieves purity in each leaf node. Then the leaf nodes are removed based on their importance or it is pruned back to a point where the cross validation error is minimum.

Problem Description

For banks, loaning out money is one of the primary sources of income, for the same reason banks have been keen on lending out money in form of loans. But during the financial crisis of the year 2007-2008, banks were left with low funds and had to be accurate with their decisions which made them turn to machine learning to better predict risky loans. This process is also regulated by government and the banks are expected to provide the grounds on which they lend the money to their customers. Therefore they turned to decision tree due to its transparency, easy to understand flowchart like structure making it easier to be explained when compared to other machine learning algorithms.

In this report the data used is a real world data which is obtained from a German credit agency. The data contains the details and features of 1000 loan examples. We will use this data to make a C5.0 decision tree algorithm to analyze which variables can be a good predictor of a loan default. We will also focus on reducing type 1 and type 2 errors and will improve the model accuracy by adaptive boosting.

Objectives:

The primary goal of this report is to:

* To comprehensively introduce C5.0 Decision tree algorithm, explaining their mathematical foundations and applications in indemnifying risky loan situations.
* Outline the method involved in building and training the model.
* Balancing and penalizing type 1 and type 2 errors to improve the model.
* Using boost the model using adaptive boosting.
* Propose recommendations for future research and development in this field.

Method

This project will be using the dataset “credit.csv.” The primary source of literature used is “Machine Learning with R, by Brett Lantz, 2nd Ed., Packet Publishing, 2015 (ISBN: 978-1-78439-390-8)”.

Listed below are the steps taken in the report:

* Step 1 – Collecting data.
* Step 2 – Exploring and preparing the data.
* Step 3 – Creating random training and test datasets.
* Step 4 – Training a model on the data.
* Step 5 – Evaluating model performance.
* Step 6 – Improving model performance.
* Step 7 – Type I vs type II errors.

Steps taken:

**Step 1 – Collecting data.**

The data used in this model is called the “credit.csv” which contains 1000 examples of previous customers of a German credit agency. We will import the data using the read.csv() function. We will not be using stringAsFactor as the majority of data is nominal in nature.

Text, letter

Description automatically generated

**Step 2 – Exploring and preparing the data.**

The data shows a total of 1000 observations and 17 features or variables. We will use the table() function to check the couple of the variables such as the checking and savings balance. These two variables are important predictors of loan default as amount held in a bank account is often a qualifier for a set loan amount.

Text, letter

Description automatically generated

The other important numeric features are duration and the amount of credit. The average loan amount is 3271 with a median of 2320 and ranges from 250 to 18424 DM(Deutsche Marks). The average time is 20.9 and median is 18, and it ranges from 4 to 72 months.

Text

Description automatically generated with low confidence

The default table shows us the loans which went into default. Which in this case is a total of 30 % which is a certainly high.



**Step 3 – Creating random training and test datasets.**

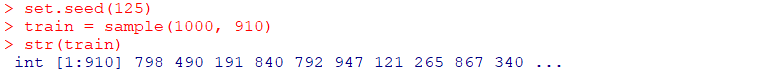
We will split the data into two sets, that is the training set and a test set. With the training set we will feed the data to the model and build the decision tree which will contain 91% or 910 examples. We will use rest of the 9% or 90 applicants as the test set to test the new applicants.

To test the effectiveness of a model, t is crucial the that the sample for training is derived randomly to avoid the model to work on set patterns which can affect the model performance.

For Example - if the data is arranged as per the dates, the training data will have 91% of the old data and the test set will have 9% of the most recent data. If there is a difference in the general pattern of loan defaults between the old and the recent data, then the model will not be accurate.

In this case the data is not random. To solve this issue, we will use the set.seed() function. This will select a subset of the record at random. We will set the seed value in this case at 125, which will cause the randomization to follow a particular sequence.

We will use sample()function to select random 910 values at random.



We will create the training and test data set by splitting the it into 91 and 9% sets by using vector to select rows.



We get roughly 30% of loan defaults in both the training and test datasets, which indicates that the data is fairly split.

Graphical user interface

Description automatically generated with medium confidence

**Step 4 – Training a model on the data.**

We will install the C50 package to use the C5.0 algorithm which will be used to train the decision tree model. The default C5.0 configuration will be employed, as demonstrated in the code below. In order to train the data frame, we must omit the 17th column, which is the default class variable, while providing it as the target factor vector for classification.

We also see that the C5.0 requires a factor outcome whereas R has the credit\_train$default read as numeric rather than factor, hence we convert in into factor using as.factor()

A picture containing text

Description automatically generated

The cre\_model now containsC5.0 decision tree

Graphical user interface, text, application, email, website

Description automatically generated

It shows that the tree size is 72, which is considerably large, with a total of 16 predictor values for a total of 910 values.

We use the summary() function to see the tree’s decisions. From the summary, we can understand that :

* If the checking balance is unknown, then it is classified as “not likely to default”.
* Otherwise yes if the checking balance is anything other than unknown.
* If the age is les than or equal to 30, then there is a high chance that the loan is likely to default

The parentheses explain the decision and number of incorrectly classified decision. For example, in case of age 10 examples reach the decision whereas 1 was incorrect.

Graphical user interface, text

Description automatically generated

The summary also displays a confusion matrix which indicates the incorrectly classified records in a training data. We see that model classifies all but 115 of the 910 training examples for an error rate of 12.6%. A total of 11 false positives(incorrectly classified as yes) and 104 false negatives(misclassified as no).

Graphical user interface, text, application

Description automatically generated

**Step 5 – Evaluating model performance.**

To avoid the model from overfitting which is a highly likely phenomenon in case of decision trees, we will evaluate the model on the basis of a test set.

We will install and load the gmodels package which enables us to use the Crosstable() function allowing us to create a vector of predicted and actual class values. If you set the prop.c and prop.r parameters to FALSE, the table's column and row percentages will be eliminated. However, the prop.t percentage will remain, indicating the proportion of records in the cell out of the total number of records.

A picture containing table

Description automatically generated

From the table above, we learn that Out of the 90 test loan application records, our model correctly predicted that 54 did not default and 8 did default. The model only correctly predicted 8 of the 26 actual loan defaults in the test data.

This type of errors can be very costly, as the bank will lose money with every default.

**Step 6 – Improving model performance.**

To improve the model performance we will use of adaptive boosting which is a process in which the algorithm decides the best class for each example. This is done by simple combining several example with complementary strength and weakness to improve the accuracy of the classifiers.

We use the C5.0 function to add the boosting and add trial parameter to set a upper limit to the addition of trees. We start by setting the value of trial at 10 which is a standard an dis known to reduce error rate by 25%

Graphical user interface, text, application

Description automatically generated

We see that the classifier makes a total of 16 mistakes out of 910 training examples. This is a significant improvement .

Text

Description automatically generated

Let us check if there is a similar improvement in the test data as well.

We see that the total error from the prior model is reduced from 28 to 25% in the boosted model. It is not a significant and is less than 25%. This lack of improvement might be the result of a relatively small training dataset.

Table

Description automatically generated

We performed adaptive boosting by setting the trial parameter as 15. But we concluded that it is giving more false predictions (27 when compared to 25 from previous paramenter) compared to when the trials was set at 10.

Increasing the trial parameter from 10 to 15 can lead to overfitting, which may result in higher false predictions on the test data. Increasing the trial parameter beyond a certain threshold may result in the model fitting too closely to the training data, causing it to memorize noise and produce overly complex decision boundaries.

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated with medium confidence

Table

Description automatically generated with medium confidence

**Step 7 – Type I vs type II errors.**

In the decision tree model, we see that there are two different types of false predictions, that is, false positives and false negatives. Even though these are both simply errors, but in this case one error will be more costly to a bank. The false negatives will result in rejecting large number of borderline applicants which will result in high loss of profits which will far outweigh the massive loss from loan defaults.

We use the C5.0 algorithm to assign a penalty on the costly mistake. We do it assigning penalties in a cost matrix specifying how costly each error is. We will describe the 2x2 matrix using list of two vectors each with two values for both actual and predicted values.

A picture containing logo

Description automatically generated

Now we assign penalties to different types of to different types of errors. In this case we are giving a false negative(predicted no but actually yes) four times as much as a missed opportunity.

A picture containing text

Description automatically generated

To see the impact of classification we will apply it to the decision tree as mentioned below which will produce the following confusion matrix.

Table

Description automatically generated with medium confidence

Compared to the boosted model(trial = 10), this model makes more mistakes, a total of 29. But we see that the type of errors made is different this time. By assigning four times the penalties on the false negative (predicted no but actually yes) we see a total reduction of 9 errors( 15 – 6 errors). Hence, this trade will result in a smaller number of false negatives at the expense of increasing false positives, which may be acceptable in this situation.

Conclusion

From this report we conclude the following:

* Decision tree is one of the most widely used and implemented machine learning algorithm due to its transparent nature and easy to understand flow-chart like description making it a top choice for banking and credit sectors.
* C5.0 is the most implemented decision tree algorithm.
* Through the analysis of data we conclude that the amount of data available for the training set is not enough to prepare a good model, as we get low improvement(less than 25%) even when the model is boosted.
* Boosting the model with a higher trial parameter such as 15, will lead to overfitting of the data and will give even worse results.
* We can assign penalties to different types of outcomes of a confusion matrix, which in this case was a false negative assuming that bank loan defaults cost more than missed loaning opportunities.

Limitations and Further improvements

Some limitations of the C5.0 algorithms are:

* Model is highly susceptible to overfitting and underfitting.
* One can face issues to model some relationships due to reliance on axis-parallel splits.
* The algorithm can often produce large decision trees which are hard to interpret and hence counterintuitive.
* There is always a bias toward the split which is high in terms of magnitude.
* Slight changes in the training set will lead to substantial changes to the logic of the decision.

Some potential improvements of the C5.0 algorithms are:

* Ability to manage missing values.
* Efficient dealing with imbalanced datasets.
* Incorporate more diverse splitting rules.
* Using hyperparameters, such as the complexity parameter and the minimum number of examples required for a split.

Coding

> cre = read.csv("credit.csv")

> str(cre)

> table(cre$checking\_balance)

> table(cre$savings\_balance)

> summary(cre$months\_loan\_duration)

> summary(cre$amount)

> table(cre$default)

> set.seed(125)

> train = sample(1000, 910)

> str(train)

> cre\_train = cre[train, ]

> cre\_test = cre[-train, ]

> prop.table(table(cre\_train$default))

> prop.table(table(cre\_test$default))

> install.packages("C50")

> library("C50")

> cre\_train$default = as.factor(cre\_train$default)

> cre\_model = C5.0(cre\_train[-17], cre\_train$default)

> cre\_model

> summary(cre\_model)

> cre\_pred = predict(cre\_model, cre\_test)

> library("gmodels")

> CrossTable(cre\_test$default, cre\_pred,

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

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

> cre\_boost10 = C5.0(cre\_train[-17], cre\_train$default, trial = 10)

> cre\_boost10

> summary(cre\_boost10)

> cre\_boost\_pred10 = predict(cre\_boost10, cre\_test)

> CrossTable(cre\_test$default, cre\_boost\_pred10,

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

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

> cre\_boost15 = C5.0(cre\_train[-17], cre\_train$default, trial = 15)

> cre\_boost15

> summary(cre\_boost15)

> cre\_boost\_pred15 = predict(cre\_boost15, cre\_test)

> CrossTable(cre\_test$default, cre\_boost\_pred15,

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

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

> matrix\_dim = list(c("no", "yes"), c("no", "yes"))

> names(matrix\_dim) = c("predicted", "actual")

> matrix\_dim

> error\_cost = matrix(c(0, 1, 4, 0), nrow = 2,dimnames = matrix\_dim)

> error\_cost

> cre\_cost = C5.0(cre\_train[-17], cre\_train$default,costs = error\_cost)

> cre\_cost\_pred = predict(cre\_cost, cre\_test)

> CrossTable(cre\_test$default, cre\_cost\_pred,

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

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

Reference

* Machine Learning with R, by Brett Lantz, 2nd Ed., Packet Publishing, 2015 (ISBN: 978-1-78439-390-8)