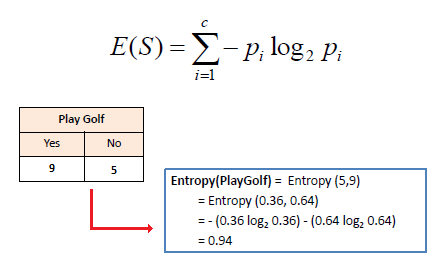
***Decision Trees***

**Aim**: To build a model which accurately predicts the risky bank loans because of the less credit available and tightened lending system. Usually, loans are accepted for customer with high credit rating and lending practices are carefully observed by governments and executives. Banks should be able to explain why loans are accepted and rejected. In present world, approvals and rejections are happening instantaneously either on phone calls or internet, so there is high chance that banks are more likely to deploy machine learning models for automatic disbursements or rejections.

**Algorithm selected**: Decision tree model is a divide and conquer strategy, used for this case as it solves this problem directly. Decision trees are built as C5.0 algorithm in R. Results from C5.0 is easier to understand and deploy.

C5.0 uses entropy which quantifies the randomness, or disorder, within a set of class values. Sets with high entropy are very diverse and provide little information about other items that may also belong in the set. The decision tree hopes to find splits that reduce entropy, ultimately increasing homogeneity within the groups.



If the feature has single class, it is called pure or leaf node which can’t be divided further. If the feature has multiple class, that will be a root node. The algorithm calculates the change in homogeneity that would result from a split on each possible feature, which is a measure known as information gain. The information gain for a feature F is calculated as the difference between the entropy in the segment before the split (S1) and the partitions resulting from the split (S2) Splitting the data is based on the information gain value. The feature with highest information gain is selected for further split.

The higher the information gain, the better a feature is at creating homogeneous groups after a split on this feature. If the information gain is zero, there is no splitting on this feature. On the other hand, the maximum information gain is equal to the entropy prior to the split.

A decision tree can continue to grow indefinitely, choosing splitting features and dividing the data into smaller and smaller partitions until each example is perfectly classified or the algorithm runs out of features to split on. However, if the tree grows overly large, many of the decisions it makes will be overly specific and the model will be overfitted to the training data. The process of pruning a decision tree involves reducing its size such that it generalizes better to unseen data. Pruning is of 2 types pre pruning and post pruning. C5.0 algorithm has post pruning strategy.

Let’s dive in into the steps now

1. **DATA COLLECTION**: Data for this project needs large instances of past bank loans and their outcome. Data set from German credit agency is used in this case “credit.csv” from UCI machine learning repository http://archive.ics.uci.edu/ml

This data set includes 1000 instances and 17 features comprising of continuous and categorial variables describing the characteristics of loan and loan requestor. The default feature shows whether the applicant is risky and went into delinquency and that will be out response variable.

1. **DATA PREPARATION**:

We will start importing the csv data in R,

* credit=read.csv("credit.csv")

#To see data structure

* str(credit)

*'data.frame': 1000 obs. of 17 variables:*

*$ checking\_balance : chr "< 0 DM" "1 - 200 DM" "unknown" "< 0 DM" ...*

*$ months\_loan\_duration: int 6 48 12 42 24 36 24 36 12 30 ...*

*$ credit\_history : chr "critical" "good" "critical" "good" ...*

*$ purpose : chr "furniture/appliances" "furniture/appliances" "education" "furniture/appliances" ...*

*$ amount : int 1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...*

*$ savings\_balance : chr "unknown" "< 100 DM" "< 100 DM" "< 100 DM" ...*

*$ employment\_duration : chr "> 7 years" "1 - 4 years" "4 - 7 years" "4 - 7 years" ...*

*$ percent\_of\_income : int 4 2 2 2 3 2 3 2 2 4 ...*

*$ years\_at\_residence : int 4 2 3 4 4 4 4 2 4 2 ...*

*$ age : int 67 22 49 45 53 35 53 35 61 28 ...*

*$ other\_credit : chr "none" "none" "none" "none" ...*

*$ housing : chr "own" "own" "own" "other" ...*

*$ existing\_loans\_count: int 2 1 1 1 2 1 1 1 1 2 ...*

*$ job : chr "skilled" "skilled" "unskilled" "skilled" ...*

*$ dependents : int 1 1 2 2 2 2 1 1 1 1 ...*

*$ phone : chr "yes" "no" "no" "no" ...*

*$ default : chr "no" "yes" "no" "no" ...*

This is the structure of the data with combination of continuous and categorial data types.

Applicants checking and savings accounts gives us clue for predicting the default variable. As both are categorial in nature, using table () which gives frequency of instances in form of table.

* table(credit$checking\_balance)

*< 0 DM > 200 DM 1 - 200 DM unknown*

*274 63 269 394*

* table(credit$savings\_balance)

*< 100 DM > 1000 DM 100 - 500 DM 500 - 1000 DM unknown*

*603 48 103 63 183*

* summary(credit$months\_loan\_duration)

*Min. 1st Qu. Median Mean 3rd Qu. Max.*

*4.0 12.0 18.0 20.9 24.0 72.0*

* summary(credit$amount)

*Min. 1st Qu. Median Mean 3rd Qu. Max.*

*250 1366 2320 3271 3972 18424*

From summary table, we can figure out that loan amount ranged from minimum of 250DM to maximum of 18424 DM with mean of 3271DM. Loan duration ranged from 4 months to 72 months with mean of 20.9 months.

* table(credit$default)

*no yes*

*700 300*

Out of 1000 instances ,700 went into default. A high rate of default is undesirable for a bank because it means that the bank is unlikely to fully recover its investment. If we are successful, our model will identify applicants that are at high risk to default, allowing the bank to refuse credit requests

**3.DATA PREPARATION:** Dividing the data into training and test data sets. 900instances out of 1000 are selected for training the decision tree and 100 for evaluating performance of decision trees.

Credit data is not randomly distributed. If we take first 900 instances and train the model will be aligned to one response variable. So, using sample () function, lets us do random sampling. Before doing random sampling, we set seed value to make sure the same random samples are picked up even when the model is used later and helps get identical results.

* set.seed(123)
* train\_sample=sample(1000, 900)
* str(train\_sample)

*int [1:900] 415 463 179 526 195 938 818 118 299 229 ...*

Train sample now has randomly selected instances. Preparing the training set with these samples and test set without these samples.

* credit\_train=credit[train\_sample, ]
* credit\_test=credit[-train\_sample, ]
* prop.table(table(credit\_train$default))

*no yes*

*0.7055556 0.2944444*

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

*no yes*

*0.65 0.35*

This seems a fair split as both the sets has almost 30% defaulted loans.

**4.TRAINING DATA:** Decision Trees in R is performed using C5.0 algorithm in C5.0 package.

* install.packages("C50")
* library(C50)

Excluding the default feature as that is a response,

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

*Call:*

*C5.0.default(x = credit\_train[-17], y = credit\_train$default)*

*Classification Tree*

*Number of samples: 900*

*Number of predictors: 16*

*Tree size: 69*

This indicates that decision tree has 69 splits (decisions made)

* summary(credit\_model)

*Text, letter

Description automatically generated* *Text, timeline

Description automatically generated*

Text

Description automatically generated

First 3 Conclusions from the model are as follows:

1.If checking balance is unknown or greater than 200DM, then the classified as not default for 412 times.

2.If checking balance is <0DM and between 1-200 DM, and credit history is perfect and very good, and housing is rent then classified as likely to default.

3. If checking balance is <0DM and between 1-200 DM, and credit history is perfect and very good, housing is other, employment duration is greater than 7 years ,1-4 years,4-7 years then classified as likely to default.

Summary of model gives the following evaluation matrix.

*Evaluation on training data (900 cases):*

*Decision Tree*

*----------------*

*Size Errors*

*69 99(11.0%) <<*

*(a) (b) <-classified as*

*---- ----*

*625 10 (a): class no*

*89 176 (b): class yes*

The model correctly classified all but 99 of the 900 training instances for an error rate of 11.0 percent. A total of 10 actual no values were incorrectly classified as yes (false positives), while 89 yes values were misclassified as no (false negatives). Decision trees usually tend to overfit the model to the training data

**5.EVALUATING THE MODEL:** We have seen the evaluation of training data, now doing it for test data,

* 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'))

Table, calendar

Description automatically generated

Model has an accuracy rate of 70 percent and error rate is 30 percent. Model correctly predicted 15 out of 35 actual loan defaults in test data, remaining 20 were wrongly classified. These types of errors are costly as the banks will be on verge of losing the money. We must try eliminating or reduce those type of errors.

**6.IMPROVING THE MODEL:** Our model has high error rates so its inefficient in real world cases. We must adjust the model to improve the model performance and reduce the costly errors.

**Improvement 1:**C5.0 algorithm provides boosting techniques which is nothing but bringing more weak models together to make it strong model. In C5.0 function, trail is the parameter which helps in boosting. Trail keeps the upper limit for the model, algorithm will stop adding trees if it recognizes that additional trials do not seem to be improving the accuracy.

It was believed that adding trail as 10 decreases the error rate by 25 percent

* credit\_boost10= C5.0(credit\_train[-17],credit\_train$default,trials = 10)
* credit\_boost10
* summary(credit\_boost10)

*57 was the tree size, in this case, now evaluating the performance*

* credit\_boost\_pred10=predict(credit\_boost10, credit\_test)
* CrossTable(credit\_test$default, credit\_boost\_pred10,prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,dnn = c('actual default', 'predicted default'))

Table

Description automatically generated

Accuracy rate is same as 70 percent before improvement, but the costly errors are reduced by 3 counts after boosting. The lack of an even greater improvement may be a function of our relatively small training dataset, or it may just be a very difficult problem to solve

**Improvement 2**: Using the cost matrix: The C5.0 algorithm allows us to assign a penalty to different types of errors, to discourage a tree from making more costly mistakes. The penalties are designated in a cost matrix, which specifies how much costlier each error is, relative to any other prediction.

To begin constructing the cost matrix, we need to start by specifying the dimensions. Since the predicted and actual values can both take two values, yes or no, we need to describe a 2 x 2 matrix, using a list of two vectors, each with two values. At the same time, we'll also name the matrix dimensions to avoid confusion later:

* matrix\_dimensions=list(c("no", "yes"), c("no", "yes"))
* names(matrix\_dimensions)=c("predicted", "actual")
* matrix\_dimensions
* error\_cost=matrix(c(0, 1, 4, 0), nrow = 2,dimnames = matrix\_dimensions)
* error\_cost
* 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'))

Table, calendar

Description automatically generated

Though accuracy didn’t improve, costly errors have been reduced drastically. This trade resulting in a reduction of false negatives at the expense of increasing false positives may be acceptable if our cost estimates were accurate

**Conclusion:** Decision Tree model is built using C5.0 algorithm, the resulting model has 70% accuracy with 20 costly errors (Actual loan is default but predicted as not). By applying boosting technique, though the accuracy didn’t improve costly errors reduced by 3 counts. Our model at this stage is not best enough to use in real world cases. Errors were further reduced to 7 counts by applying tradeoff between costly errors and accuracy using cost error matrix.

**Limitations of Decision Trees:**

* They often biased toward splits on features having many levels and easy to overfit or underfit the model
* Small changes in the training data can result in large changes to structure of decision logic which shows different result from what user actually gets
* Large trees can be difficult to interpret and the decisions they make may seem counterintuitive