DATA SCIENCE IN FINANCE USE CASE

Classification
Using Decision Trees

Decision Trees: are versatile Machine Learning algorithm that can perform both classification and regression tasks. They are very powerful algorithms, capable of fitting complex datasets. Besides, decision trees are fundamental components of random forests, which are among the most potent Machine Learning algorithms available today.

Decision trees can be widely used in the banking industry due to their high accuracy and ability to formulate a statistical model in plain language.
There are steps involved;

1. Importing data (data collected)
2. Cleaning the dataset
3. Create train/test set
4. Build model
5. Make prediction
6. Measure performance

1. **Import, exploring and preparing the data.**

```
credit.data=read.csv(choose.files(),h=T)

str(credit.data)

'data.frame':  5000 obs. of 9 variables:

 $ age     : int  41 30 40 41 57 45 36 39 43 34 ...

 $ ed      : int  3 1 1 1 1 1 1 1 1 3 ...

 $ employ  : int  17 13 15 15 7 0 1 20 12 7 ...

 $ address : int  12 8 14 14 37 13 3 9 11 12 ...

 $ income  : num  35.9 46.7 61.8 72 25.6 28.1 19.6 80.5 68.7 33.8 ...

 $ debtinc : num  11.9 17.9 10.6 29.7 15.9 ...
```

$ creddebt: num 504 1353 3439 4166 1498 ...

$ othdebt : num 3.77 7 3.14 17.2 2.56 0.28 1.3 8.06 3.26 2.2 ...

$ default : int 0 0 0 0 0 0 1 0 0 0 ...

We see the expected 5,000 observations and 9 variables, which are a combination of categorical and numerical data types.
The default variable indicates whether the loan applicant was unable to meet the agreed payment terms and went into default. A total of 25.1 percent of the loans went into default and 74.9 percent not defaulted.

table(default)

Default	
0	1
2996	1004

A high default rate is undesirable for a bank because it means that the bank is unlikely to fully get back its investment. Successful model will identify applicants that are likely to default, so that this number can be reduced and any other decision can be made. Data is also cleaned by making sure rem(NA) from the dataset.

The common practice is to split the dataset, 4000 percent of the data serves to train the model, and 1000 percent to make predictions. Therefore, need to create two separate data frames. You don't want to touch the test set until you finish building your model. You can create a function name credit_train() and credit_test that takes three arguments.
set.seed(1:4000)#used to generate random numbers in a predefined sequence.
credit.random <- credit.data[order(runif(5000)),]# random number generation function, which by default generates a sequence of random numbers between 0 and 1.
credit_train <- credit.random[1:4000,]
credit_test <- credit.random[4001:5000,]

summary(creddebt)
summary(credit.random$creddebt)
We can use the head() function to examine the first few values in each data frame
Since the summary statistics are identical while the first few values are different, this implies that our random shuffle worked correctly.

head(creddebt)

head(credit.random$creddebt)

Model building

credit.model=C5.0(credit_train[-9], credit_train$default)

This text shows some simple facts about the tree, including the function call that generated it, the number of features, and examples (that is, samples) used to grow the tree. Also it list the tree size, which And the decisions' deep. The numbers in parentheses indicate the number of examples meeting the criteria for that decision, and the number incorrectly classified by the decision. For instance, on the first line, indicates that of the numerator value examples reaching the decision and one that are incorrectly classified as no, that is, not likely to default.

From the credit.model we can now do predictions.

credit_prediction <- predict (credit.model, credit_test)

To evaluate model performance

library(gmodels)

CrossTable(credit_test$default, credit_prediction,
prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
dnn = c('actual default', 'predicted default'))

This will generate a chi-square table(contingency table)for actual default against the predicted default.