UPPSALA UNIVERSITET

Using Decision Tree to Analyze the Turnover of Employees

Master thesis

By Gao Ying

Email address: gaoying1993@outlook.com

Thesis Supervisor:

Vladislav Valkovsky

Associate Professor, PhD

Uppsala University

Department of Informatics and Media

Information Systems

May 2017

Preface

I would like to thank my supervisor Vladislav Valkovsky, who is really patient and helped me a lot with my thesis. He always encourages me during the work so that I become confident to myself. I believe without him, I can’t finish my thesis.

I also want to thank Swedish Government, who made it possible for us foreign students to study at one of the prestigious university in the world and Mr Anders Wall, who gave me scholarship so that I can come to Sweden to study. During my life in Sweden, I had many new experience that I never had before. I got many new friends as well as advanced knowledge, I think it will be a precious memory in my life.

What’s more, I want to thank Anneli Edman. She is a good and responsible teacher who helped a lot both in my study and life. She has great passion in teaching, which makes study become much fun. And when I have some problems, she will try her best to help me solve it, I feel really warm.

Finally, I also like to thank my parents. They are really great parents and always support me whatever I did. Without their support, I won’t have the courage to study abroad. Thank for their understanding and encouragement, I love them.

Abstract

Employees turnover has become a very common phenomenon in our daily life. This phenomenon will result in many problems for companies. They need to hire new employees and train them so that the vacancy positions can be filled. They also need to change work arrangement and the duty of some employees. These will cost much money and energy. What's more, if many employees turnover from a company, absolutely it will influence the reputation and stability of this company. There may be many different reasons why they choose to leave the company. So it's important for companies to find out the most important reasons and predict who has the high possibility to leave so that companies can do some changes to make a better human resource management.

In this thesis, I choose to use decision tree algorithm to analyze main reasons for employee turnover in data mining method. I plan to analyze ID3, C4.5 and CART algorithms by using excel and R package. All data is collected from kaggle.com (which is a data mining competition website) in csv file. This data set has more than ten thousand records and considers many characteristics of employees such as satisfaction level, the score of last evaluation, project number, average work hours per month, the amount of work accidents and so on. I will choose a best algorithm to build model and then evaluate the model as well as do optimization. At last I will give some suggestions for human resource department.

This is a very practical and useful area. If it has been solved well, the same method and model can also be used to study customer churn rate, customer relation management, young people unemployment rate and etc.

Key Words

Data mining; decision tree; employee turnover; human resource management; ID3; C4.5; CART

Table of Contents

1 Introduction

1.1 Background

The percentage of employees that a company need to replace during a given time period is called employee turnover rate (Beam, 2014). For any organizations, employees are very important especially those with full experience and skills. Good employees will help the organization get success and gain profits. So how to keep workers stay in the company becomes a serious problem to consider in human resource management. However, employee turnover has become a very common phenomena. More and more employees leave their original companies for different reasons. Some people may worry about the future career opportunity, some people are not satisfied with their salary, some may think the working hours are too long and working conditions are not good, some don't like the quality of manager and they think their work lack communication and etc.(Carey & Ogden, 2004).

Employee turnover will cause many problems. According to a research, if a company loses a good employee, it will cause 1.5 times money cost than recruiting and training a new employee (source: Finders & Keepers). It can also lower customer satisfaction and retention (Bisk, 2017). It will influence time, team dynamic, productive and continuity and etc. (Frost). So it is of great importance for companies to find out the reason that influence turnover most and solve this problem in an effective and efficient way.

From the eighties of the twentieth century, companies began to use computer informatization methods to do human resource management. And from nineties, more companies began to use some software or tools like Visual Basic, Visual Foxpro or Microsoft Office. With the rapid development of technology, data mining has been used in human resource management in twenty first century (Strohmeier & Piazza, 2012). It can help companies make better decisions, manage new employees and also analyze the turnover of old employees (Ranjan et al., 2008). So I am planning to use data mining to analyze employee turnover and help companies come up with solutions to make better human resource management.

1.2 Problem Statements and Definition

Employee turnover is belong to recruitment and selecting process in human resource management. It can be divided into two different kinds, voluntary turnover and involuntary turnover. Voluntary turnover is employees quit from their positions by volunteer, involuntary turnover is that employees are not happy to leave the position but employers make the decision to let them leave (Trip, n.d.).

Now there are many big companies developing some popular data mining software that can be used in human resource management and some mature tools are:

(1) Cognos: Cognos is a software in business intelligence and performance plan these two areas. It helps solve CPM like enterprises plan, scorecard and business intelligence (IBM).

(2) SPSS: SPSS combines many statistics technologies like cluster analysis, regression analysis, time series analysis with graphics as well as some data mining methods like neural network, decision tree and etc. together. SPSS first introduced data mining flow into statistic software, and users can do data cleaning, data transfer and model built in a work flow environment very easily (Yockey, 2010).

(3) KnowledgeStudio: It's also a statistic software and has several advantages: (a) short response time and fast running speed; (b) it's easy to understand the model and report that we get; (c) we can easily add new algorithms and import external model (Berson & Smith, 1999).

(4) Weka: It is a machine learning software written in Java and the full name is Waikato Environment for Knowledge Analysis. It has user-friendly interfaces which uses graph to guide people operating functions. Also it has visualization tools and algorithms to do data analysis and predict model (Witten et al., 2011).

(5) R: The R language is very popular in statistics and data analysis. There are many packages in R that we can use directly and it's public and free (Fox & Andersen, 2005).

I chose R to do analysis because that R is a public software and there are many packages inside, people can use these packages directly. There is no necessary to write code, and R can create many great graphs which may make analysis result shown better.

Now data mining has been used in human resource management mainly in these three areas: (1) turnover analysis; (2) labor planning; (3) recruitment analysis. And there are many data mining methods to solve related human resource management problems. It is not a new area to use data mining and many related researches have been done. For example, decision tree has been used to improve human resources in construction company (Chang & Guan, 2008) and also been used to improve employee selection and enhance human resources (Chien & Chen, 2008). Sexton et al. (2005) used neural network to do research in employee turnover. Cho and Ngai (2003) used regression tree C4.5 algorithm and network (feed-forward) to select insurance sales agents. Naive Bayesian classifier has been used in job performance prediction by Valle, Varast and Ruz (2012). Neural network and self-organizing map has been used to predict turnover rate for technology professionals by Fan et al. in 2012. Tamizharasi and UmaRani (2014) have used decision trees, logistic regression and neural network to analyze employee turnover. And all data models were built using SEMMA (Sample, Explore, Modify, Model, and Assess) methodology.

1.3 Motivation

Although we have used data mining methods in many human resource management areas, there are still some problems. We need to know how to certificate that the management system is proper and how to do quantitative analysis for qualitative data. Most data mining methods are used in selecting and enhance, but satisfaction and turnover are not very much. Here are some common problems that exist in human resource management:

(1) How to quantitative analysis to help human resource department get real results.

Some companies analyze the turnover reasons by having a talk with these employees. However, different people have different thoughts and they may not express their real thoughts very well. Also the talking skills and methods with people are different . So it’s hard to know the exact reasons why employees turnover and if we use these interview data to analyze, it will have big error with real situation.

(2) Reduce the influence of subjective factors on decision making.

For lack of quantitative analysis, there are many subjective factors in human resource management, which are not very fair for employees.

(3) Different situations may have different solutions, different algorithms will influence the efficiency to solve the problems.

Some algorithms have been used in analyzing employee turnover, but they perform not very well because different data set may have different features. We need to choose proper algorithm for our own data set and we can do some optimization in certain situation.

So I will use data mining methods solving employee turnover problems. My research goal is to use decision tree algorithm to analyze the most important reasons that contribute to high employee turnover and create a priority table for human resource management department. First I'm planning to analyze ID3, C4.5 and CART algorithms in decision tree. By using them in real data set, compare difference of these three algorithms and choose the best one suits the real data set to build model to get the main reasons for employee turnover. After building model, I will also evaluate the model by some metrics like confusion matrix, ROC and AUC graphs, MAE, MSE and NMSE. These metrics will help us better understand whether our model is good enough. If these metrics show the model is good, I can use it directly. If not, then I need to modify this model. I will try to optimize this model and compare it with before to check whether it becomes better. I will mainly use some exist packages in R, if no such packages exist I will build in RStudio to write algorithm. At last, I will use the model to predict the turnover happening possibility of employees and create a priority table to show those people who should be retained by human resource department first. If the prediction can be done in an effective and efficient algorithm, it will reduce a great amount of loss in companies.

1.4 Structure

This thesis is introducing some data mining algorithms and selecting decision tree algorithm to do analysis. I use employee data to find the most important reasons that influence turnover and present prior employees that needed to talk first for human resource department and help decision making. Here is the structure of the thesis:

Chapter 1 is about the background of employee turnover situation and how data mining methods and tools used in human resource management as well as why I want to use data mining to solve employee turnover problem.
Chapter 2 is about the definition of data mining and some common algorithms and advantages and disadvantages of these algorithms. Also I will explain why I choose to use decision tree algorithm in employee turnover problem.
Chapter 3 mainly focuses on the definition of decision tree and different algorithms in decision tree.
Chapter 4 is to use visualization ways to show the analysis results. I will do data cleaning, data transfer, build model in R, evaluate model.
Chapter 5 is how to optimize the algorithm and do prune if overfitting situation happens in employees data set.
Chapter 6 is the suggestion of result and the meaning of results that can help human resource department do decision making.
Chapter 7 is the conclusion of this thesis and some problems that haven't been solved so far and will continue research in the future.

2 Data Mining and Basic Algorithm

Data mining has become very popular because nowadays people have more desire with the implicit knowledge in data and large-scale database systems have been used extensively. With the rapid development of society, data mining has turned to data analysis from simple inquiry in the past. Data can be seen as information and knowledge and database is an effective way to sort data. However, with the huge volume of data increase, simple database inquiry technology is not enough for the requirement of data. So, scientists begin to consider how to get useful information and knowledge from huge amount data, that's how data mining technology produced (Han et al., 2001).

2.1 Data Mining

2.1.1 Definition and Feature of Data Mining

Data mining is the process to select implicit and useful information and knowledge that people don't know before from huge volume, incomplete, random, noisy and blurred data. What we want is to discover information that we don't know before, but this information can be understood and also implemented. So in general, data mining process can also be described as Knowledge Discovery in Database (KDD). But in fact, data mining and KDD are not exactly same. KDD can be seen as a special example in data mining and data mining can be seen as a process in KDD (Han et al., 2011).
Data mining has very broad applications, it includes data visualization, database systems, artificial intelligence, statistics knowledge, parallel computing and etc. It can be defined as broad data mining and narrow data mining. Broad data mining is using computer as tools to do data analysis, it contains traditional statistical methods. However, narrow data mining emphasizes discovering knowledge from data set in automatic and heuristic ways (Han et al., 2011). We usually define data mining in narrow way.

Compared with statistics technology, data mining can be combined with database technology very well. And neural network, genetic algorithm, neural blurred and etc can also be used in data mining technology besides statistics technology. In general, there are seven steps to do data mining, which are as follow.

Data cleaning: deal with the noisy data and empty data and some unrealistic data that are different from most data.

Data integration: combine different data sets which have common attributes according to different data mining aims.

Data selection: select useful data according to different data mining aim.

Data switch: change data in data set to proper forms that can be used to build models.

Data mining: use proper models to extract data.

Data evaluation: evaluate results according to different data mining targets.

Knowledge presentation: present results in the way that users can understand very clearly.

2.1.2 Function of Data Mining

Data mining is a very useful tool, it can help predict some trends or behaviors in the future from some features of data. This can be used to discover customers behavior, employee management, create profits and reduce cost, seize business opportunities and get better competition advantages. The aim of data mining can be divided into two parts, description task and prediction task (Han et al., 2011). Description task is to find human interpretable patterns that describe the general features of data set. And prediction task is to predict the value of a particular attribute based on the values of other attributes. Data mining involves six common classes of tasks (Fayyad el at., 1996), which are as follow:

Anomaly/Outlier detection: Data that is different from most data in the data set is called noise. But sometimes noise can also be meaningful, then we need to do analysis of these minority data. It is usually used in pretreatment of data set to delete abnormal data, which can improve accuracy in supervised learning (Hodge & Austin, 2004).

Association rule learning: Search and discover the relationship between different variables. The rules present the condition that frequent attributes occur at same time in data set (Piatetsky-Shapiro,1991).

Clustering: According to the data distribution, discover distribution features and divide data into groups. Data in one group is called a cluster, they are more similar than those in other groups.
Classification: It is a general process related to categorization. Classification can be used to describe and predict.
Regression: It is a statistical way to estimate the relationships between variables. Regression models are mainly used for prediction analysis. There are two kinds of regression, linear and non-linear. Linear regression usually analyzes the relationship between response (dependent) variables and predictor (independent) variables.
Summarization: It is a process that reduces some useless parts in text document by computer program and then do a summary to keep the most important points of original document.

2.2 Algorithms

There are some basic algorithms that are popular used in data mining. The IEEE International Conference on Data Mining(ICDM) which was held in Hong Kong in December, 2006 ranked top ten algorithms that are common used effectively and efficiently. They are SVM, kNN, Apriori, Naive Bayes, C4.5, k-Means, EM, PageRank, AdaBoost and CART. I will introduce some of them as follow.

2.2.1 SVM

The full name is called support vertex machine. It’s a supervised learning method and most used in non-linear and high dimensions data mining. SVM is to find a hyperplane that divides similar samples in data set (Cortes, 1995). However, it has some disadvantages. First, when training set is very large, SVM won’t perform well because matrix needs to cost much time to calculate. The second is that SVM is a two classifications method, but in our situation we need to do multiple classifications. That’s why I didn’t choose this algorithm.

2.2.2 kNN

It is called k nearest neighbor, which uses k “closest” points (nearest neighbor) for performing classification (k is a positive integer, typically small) (Altman, 1992). It’s a supervised learning method and a good way to do classification and regression. However, if we use kNN in the small size samples, it won’t perform well. It is very sensitive to the local structure of the data. And it has very large calculation because we need to calculate k adjacent points for the sample that needs to be classified.

2.2.3 Apriori

It’s based on association rules to find the relationships between different items. First we need to find frequent item sets in data set and do analysis of these item sets. We make association rules and then evaluate decision data according to these rules, finally choose rules that have larger confidence and support than required smallest one (Piatetsky-Shapiro, 1991). It is usually used in decision support area. Actually in my data set, we can also use association rules. But, decision tree have a better structure that we can get the rules more clearly.

2.2.4 Artificial Neural Network

We can divide a neural network into three parts, they are input layer, output layer and hidden layer. If the amount of hidden layer and nodes of each layer are more, the neural network is more complicated. The nodes in one layer are connected only to the nodes in the next layer. It usually uses Backpropagation algorithm. Starting with the output layer, to propagate error back to the previous layer in order to update the weights between the two layers, until the earliest hidden layer is reached (McCulloch, 1943).

2.2.5 Naive Bayes

It has good classification efficiency and stable classification effect. The main thought of Naive Bayes is when a classification given, calculate the happening possibility of each condition (Russell, 1995).

2.2.6 k-means

It’s a cluster algorithm and unsupervised learning method. We divide data into k clusters and samples in each cluster have high similarity. Then we calculate average value in each cluster and put other data outside this cluster in a closest one. Do it again and again until the average value is unchanged (MacQueen, 1967). It can do clusters, but our aim is to do classification and I want to get rules from features. So I don’t choose it.

In a summary, I have drawn a picture as figure 2.1 to show when to use these algorithms. We can see from the graph that a data mining task can be divided into description and prediction these two parts. Our aim is to do prediction and do classification and get the relationships between different employees features and the result whether they leave the company.

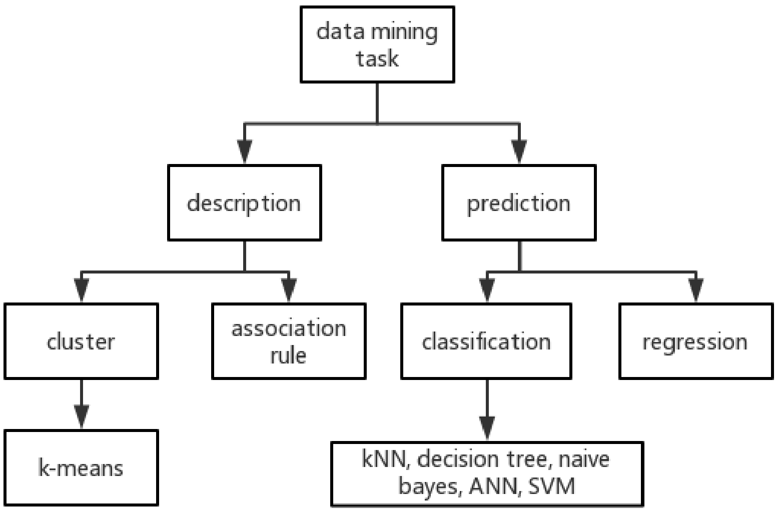


Figure 2.1 data mining task

2.3 Reasons to Choose Decision Tree

This thesis aims on main reasons of employee turnover and prediction of turnover happening possibility of employees from their features by data mining. I want to use decision tree because it is a very natural thoughts in making decision. And as I mentioned in chapter 2.2 algorithms, we can see other algorithms are not so good in this situation. What’s more, our data set is very huge and need to do multiple classifications. Here are some advantages of decision tree:

(1) Decision tree costs less time and has fast speed to do classification, which can avoid decision error and all kinds of deviation. Through a case study (Sikaroudi et al., 2015),

decision tree has the best performance in general problems.

(2) Decision tree is easy to describe and the tree structure of rules is easy to understand. People in human resource department don’t have too much data mining knowledge, but they can understand rules made by decision tree in the way IF-THEN.

(3) Decision tree algorithm has good prediction of independent features, which may perform well in different human resource data.

(4) Decision tree is to use Entropy as the metrics to judge which feature will be the tree root and then split according to information gain, gini gain or gini ratio. In this way, it’s easy to find key features that influence classification result. And the importance of features is reflected by the layer of decision tree. If the layer of tree is higher, this feature is more important, it’s very obvious.

3 Decision Tree

Decision tree is a tree structure which looks similar as flow chart. Every node in the tree represents the test of an attribute, every branch represents the output of the test, and every leaf means a class or distribution of classes. The top node is root node, from root node to one leaf consists a classification rule. So decision tree is easy to be transferred to classification rules. It has many algorithms, but the main idea is using from top to bottom induction method. And the most important part is choosing which attribute to be the node as well as the evaluation of whether the tree is correct.

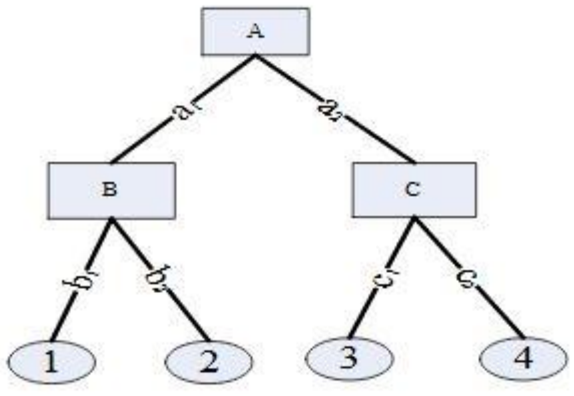


Figure 3.1 decision tree structure

As figure 3.1 shows, it’s a very simple decision tree. A, B, C represents different attributes separately in one data set. And each branch like a1,a2,b1,b2,c1,c2 represents the value of split attribute. Leaf node 1,2,3,4 represents the class of decision attribute in each sample set.

There are mainly two steps in decision tree, build a decision tree and do pruning. The thought of building decision tree is called CLS. We have a data set S, the attributes set is A, decision attributes set is D, the whole process is as follow:

(1) Make S be the root node, if all data in S belongs to the same class, turn node to leaf node.

(2) Otherwise choose one attribute $a \in A$ and divide nodes according to different values of attribute a. S has the number of m lower layer nodes, branches represent the situation of different values of a.

(3) Induct step 1 and step 2 for m branch nodes.

(4) If attributes in one node belong to same class or there is no node to divide, then stop.

The most two important things in decision tree are: 1 how to decide best split node? 2 when to stop splitting? Because real data can't be pure. There must be data attributes miss, data inaccurate, noise these situations, which will result in overfit. Overfit may lower the accuracy of the classification and prediction of decision tree and increase the complexity of tree structure. So after building a tree, we also need to pruning.

3.1 ID3

ID3 algorithm was come up by Quinlan in 1986, which is based on Entropy. We need to calculate Entropy for each attribute in the data set and test attribute from the value of Entropy, then choose attribute with bigger Entropy gain to split decision tree step by step. From information theory, the size of Entropy gain reflects the uncertainty of attribute selection. The bigger the Entropy gain is, the smaller the uncertainty is, vice versa. So we use attribute with biggest Entropy gain to be test attribute.

If a set S has s samples inside, it will decide classification attributes with n values Ci, i=(1,2,...,n), si is the number of samples in Ci. For a sample data set, the total Entropy is:

$$I\left(s_1, s_2, \ldots s_n\right) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

pi is the possibility of any sample belongs to Ci, and can be calculated by $\frac{s_i}{s}$.

For example, sample S has an attribute A, and A has w different value{a1, a2, ...,aw}. Sample S is split by attribute A and becomes w subsets {S1, S2, ...,SW}. S has some samples in Sj, and they have value aj in attribute A, and these subsets are new branches split by test attribute A. If sij is the amount of samples in Sj whose class is ci, the Entropy of A is:

$$E\left(A\right) = \sum_{j=1}^{w} \frac{s_{1j} + s_{2j} + \ldots + s_{nj}}{s} I(s_{1j}, s_{2j}, \ldots, s_{nj})$$

$I(s_{1j}, s_{2j}, ..., s_{nj}) = -\sum_{i=1}^{n} p_{ij} \log_2(p_{ij})$, $p_{ij} = \frac{s_{ij}}{|S_j|}$ is the possibility of samples belong to class ci in Sj. So when use attribute A to split S, the Entropy gain is:

$$Gain(A) = I(s_1, s_2, ..., s_n) - E(A)$$

Here is a typical example of real data set as table 3.1 shows, we need to decide whether to play golf according to the weather condition (Fürnkranz, n.d.).

Table 3.1 weather condition 1

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

There are five attributes in the data set and the last one is decision attribute. First we calculate the entropy of classification attribute, in these 14 records, will go to play has 9 records and won't play has 5 records.

$$I(s_1, s_2) = I(9,5) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$$

Then we need to calculate the entropy gain of each attribute and choose the biggest one as first split node. We calculate outlook first, it can be divided into three data sets:

```
sunny,hot,high,FALSE,no
sunny,hot,high,TRUE,no
sunny,mild,high,FALSE,no
sunny,cool,normal,FALSE,yes
sunny,mild,normal,TRUE,yes

overcast,hot,high,FALSE,yes
overcast,cool,normal,TRUE,yes
overcast,mild,high,TRUE,yes
overcast,hot,normal,FALSE,yes

rainy,mild,high,FALSE,yes
rainy,cool,normal,FALSE,yes
rainy,cool,normal,TRUE,no
rainy,mild,normal,FALSE,yes
rainy,mild,high,TRUE,no
```

Then we can calculate the entropy for sunny, overcast and rainy. Which are as follow:

$$E(sunny) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$$

$$E(overcast) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0$$

$$E(rainy) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$$

After calculating the entropy for these three outlooks, we can get the total entropy of outlook.

$$E(outlook) = \frac{5}{14}E(sunny) + \frac{0}{14}E(overcast) + \frac{5}{14}E(rainy) = 0.694$$

The entropy gain is the entropy of classification attribute minus the total entropy of outlook.

$$Gain(outlook) = I(s_1, s_2) - E(outlook) = 0.246$$

In the same way, we can get the entropy of temperature, the entropy of humidity and also the entropy of windy:

$E(temperature) = 0.912$

$E(humidity) = 0.786$

$E(windy) = 0.893$

Then we can get the entropy gain of temperature, humidity and windy.

$$Gain(temperature) = I(s_1, s_2) - E(temperature) = 0.028$$

$$Gain(humidity) = I(s_1, s_2) - E(humidity) = 0.154$$

$$Gain(windy) = I(s_1, s_2) - E(windy) = 0.047$$

We can see that the biggest entropy gain is outlook, so we choose outlook to be the first split node.

And then I need to do iteration and calculate the biggest entropy gain for next split node. We know if it’s overcast, $E(overcast) = 0$, it stops splitting and the decision will be yes. We need also consider sunny branch and rainy branch.

If it’s sunny, we can get the entropy of classification attribute is:

$$I(s_1, s_2) = I\left(3, 2\right) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$$

As the same showed above, we can calculate the entropy of temperature, humidity and windy

$E(temperature) = 0.4$, $E(humidity) = 0$, $E(windy) = 0.951$.

And the entropy gain of temperature, humidity and windy is:

$$Gain(temperature) = I(s_1, s_2) - E(temperature) = 0.571$$

$$Gain(humidity) = I(s_1, s_2) - E(humidity) = 0.971$$

If humidity is high, the decision is no; if it’s normal, the decision is yes. This branch stops split at humidity.

$$Gain(windy) = I(s_1, s_2) - E(windy) = 0.02$$

The biggest gain is humidity, so next split node is humidity. $E(humidity) = 0$, so we stop splitting in humidity branch.

If it’s rainy, the calculation is almost the same as sunny. We can get the biggest gain is windy, so the next split node is windy. $E(windy) = 0$, so we stop splitting in windy branch. The decision tree is as follow:

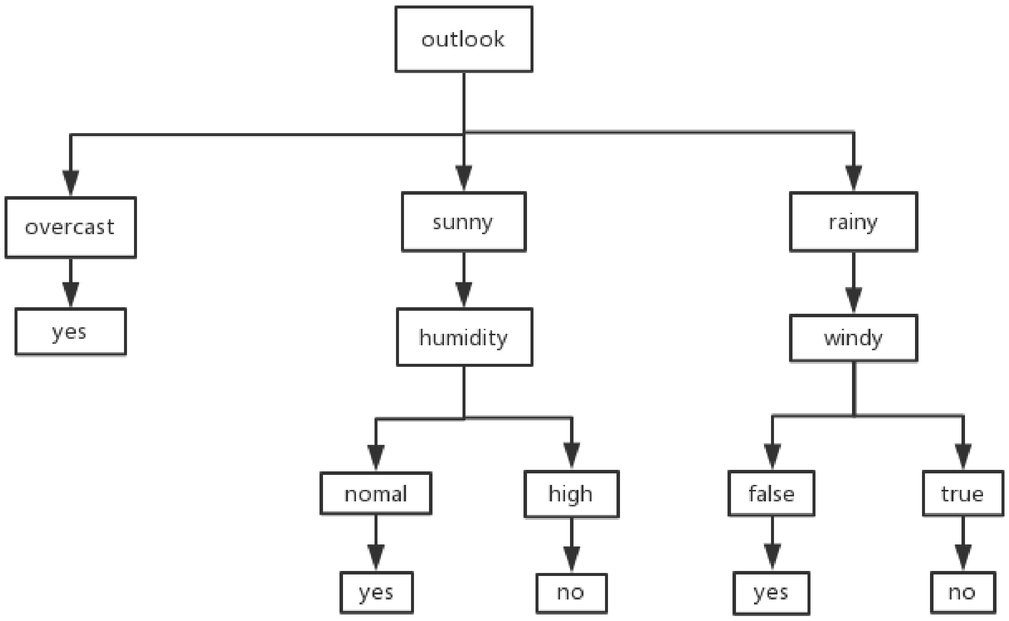


Figure 3.2 ID3 decision tree example

Here is the description of ID3 algorithm:

1 Begin

2 If (T is empty) then return(null);

3 N=a new node;

4 If (there are no predictive attributes in T) then

5 Label N with most common value of C in T(deterministic tree) or with frequencies of C in T(probabilistic tree);

6 Else if(all instances in T have the same value V of C) then

7 Label N, "X.C=V with probability 1";

8 Else begin

9 For each attribute A in T compute AVG ENTROPY(A,C,T);

10 AS=the attribute for which AVG ENTROPY(A,C,T) is minimal;

11 If(AVG ENTROPY(AS,C,T) is not substantially smaller than ENTROPY(C,T)) then

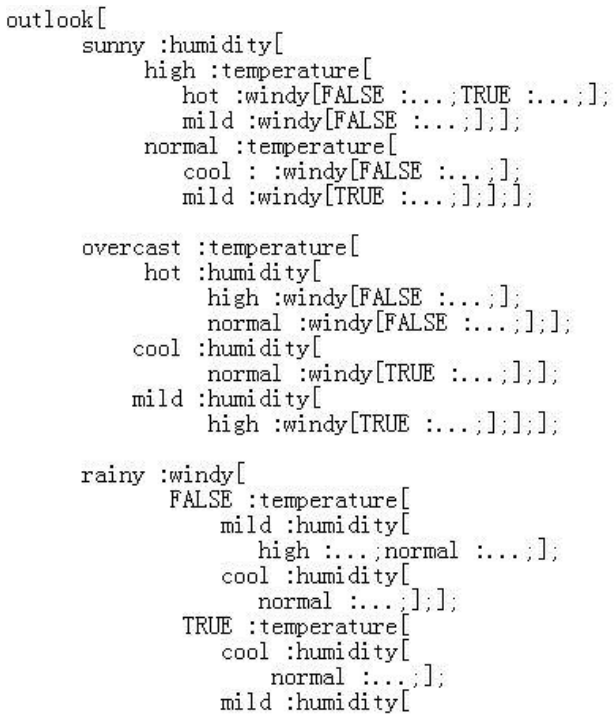
12 Label N with most common value of C in T(deterministic tree) or with frequencies of C in T(probabilistic tree);

```
13 Else begin
14 Label N with AS;
15 For each value V of AS DO begin ;
16 N1-ID3(subtable(T,A,V),C);
17 If(N!=null) then make an arc from N to N1 labeled V;
18 End
19 End
20 End
21 Return N;
22 End
```

```
Code in R:
calculateEntropy <- function(data){
t <- table(data)     #calculate how many times each result will appear
sum <- sum(t)        #total times
t <- t[t!=0]         #delete variable that is not 0
entropy <- -sum(log2(t/sum)*(t/sum))
return(entropy)
}                     #calculate entropy of two column
calculateEntropy2 <- function(data){
var <- table(data[1])
p <- var/sum(var)
varnames <- names(var)
array <- c()
for(name in varnames){
array <- append(array,calculateEntropy(subset(data,data[1]==name,sele
ct=2)))
        }
return(sum(array*p))
}
```

```
buildTree <- function(data){

#if entropy is 0, stop

if(length(unique(data$result)) == 1){

cat(data$result[1])

return()}

#prune

if(length(names(data)) == 1){

cat("...")

return()}

entropy <- calculateEntropy(data$result)   //start calculate

labels <- names(data)

label <- ""

temp <- Inf

subentropy <- c()

for(i in 1:(length(data)-1)){

temp2 <- calculateEntropy2(data[c(i,length(labels))])

if(temp2 < temp){

temp <- temp2            #record minimum entropy

label <- labels[i]       #the class that has minimum entropy

  }

subentropy <- append(subentropy,temp2)  #entropy of each subset

}

cat(label)

cat("[")

nextLabels <- labels[labels != label]

for(value in unlist(unique(data[label]))){

cat(value,":")

buildTree(subset(data,data[label]==value,select=nextLabels))

cat(";")

  }
cat("]")

}
```

Here is the result showing in the R:



```
outlook[
      sunny :humidity[
           high :temperature[
              hot :windy[FALSE :...;TRUE :...;];
              mild :windy[FALSE :...;];];
           normal :temperature[
              cool : :windy[FALSE :...;];
              mild :windy[TRUE :...;];];];

      overcast :temperature[
           hot :humidity[
                high :windy[FALSE :...;];
                normal :windy[FALSE :...;];];
          cool :humidity[
                normal :windy[TRUE :...;];];
          mild :humidity[
                high :windy[TRUE :...;];];];

      rainy :windy[
             FALSE :temperature[
                 mild :humidity[
                    high :...;normal :...;];
                 cool :humidity[
                    normal :...;];];
              TRUE :temperature[
                 cool :humidity[
                     normal :...;];
                 mild :humidity[
```

Figure 3.3 ID3 decision tree result in R

3.2 C4.5

C4.5 algorithm uses gain ratio instead of information gain to decide split feature, features with higher gain ratio will be the next split node. Compared with ID3 algorithm, C4.5 has some advantages. It has a better way to deal with continuous features. In data pretreatment, it use consistency method of continuous attribute values. In tree built process, it can tolerate some missing features.

Gain ratio is the value of gain divide split information. For example, if sample S has a feature A and A has w different values{a1, a2, ...aw}, according to values of A we can divide sample S into w subsets{S1, S2,...SW}

$$GainRatio(A) = \frac{Gain(A)}{Spliti(A)}, SplitI(A) = -\sum_{j=1}^{w} p_j \log_2(p_j)$$

I use a similar data table whether to play golf as table 3.2 shows to give an example of how C4.5 algorithm runs (Fürnkranz, n.d.).

Table 3.2 weather condition 2

outlook	temperature	humidity	windy	play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	78	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rainy	71	80	true	no

For discrete data, the calculation is similar with ID3 except we use GainRatio instead of Entropy Gain. From the calculation in chapter 3.1 ID3, I can get the entropy gain of outlook and total entropy of outlook. $Gain(outlook) = 0.246$, $splitI(outlook) = 0.694$.

Also, from the definition of GainRatio, we can get the GainRatio of outlook is:

$$GainRatio(outlook) = \frac{Gain(outlook)}{splitI(outlook)} = \frac{0.246}{0.694} = 0.354.$$

In the same calculation way of outlook, I can also get the GainRatio of windy,

$GainRatio(windy) = 0.053$.

For continuous data, we can't calculate the GainRatio directly. We need to find split threshold first and then calculate GainRatio. ID3 algorithm can't deal with continuous data, so in chapter 3.1 ID3 we don't consider continuous situation. First we sort temperature from low to high as table 3.3 shows.

Table 3.3 temperature condition 1

temperature	64	65	68	69	70	71	72	72	75	75	80	81	83	85
play	yes	no	yes	yes	yes	no	no	yes	yes	yes	no	yes	yes	no

We make the continuous data discretization, <=vj will be divided to left tree, >vj will be divided to right tree. If the value of decision feature changes, then corresponding temperature is a vj. In this data set, vj will be 64, 65, 68, 71, 72, 80, 81. Because when temperature is 64, decision whether to play is yes; when temperature is 65, decision whether to play is no, the decision feature changes. Although 85 also meets the requirement, there is no value larger than 85. So I didn't choose vj as 85. We use biggest entropy gain to determine which threshold we should use. So I calculate 7 times to choose one with biggest entropy gain.

We can get that the entropy of classification attribute is:

$$I(s_1,s_2)=I\left(9,5\right)=-\frac{9}{14}\log_2\frac{9}{14}-\frac{5}{14}\log_2\frac{5}{14}=0.94$$

We divide temperature by >64 and <=64. When temperature>64, there are 8 yes and 5 no; when temperature<=64, there are 1 yes and 0 no.

The entropy of temperature is:

$$E(temperature)=\frac{13}{14}\left(-\frac{8}{13}\log_2\frac{8}{13}-\frac{5}{13}\log_2\frac{5}{13}\right)+\frac{1}{14}\left(-\frac{1}{1}\log_2\frac{1}{1}-\frac{0}{1}\log_2\frac{0}{1}\right)=0.893$$

The entropy gain of temperature is the entropy of classification attribute minus the entropy of temperature: $Gain(temperature)=0.047$

So in the same way, we can calculate the entropy gain when the threshold is 65, 68, 71, 72, 80 81 respectively.

When temperature>65, there are 8 yes and 4 no; when temperature<=65, there are 1 yes and 1 no. We can get $Gain(temperature)=0.01$.

When temperature>68, there are 7 yes and 4 no; when temperature<=68, there are 2 yes and 1 no. We can get $Gain(temperature)=0$.

When temperature>71, there are 5 yes and 3 no; when temperature<=71, there are 4 yes and 2 no. We can get $Gain(temperature) = 0.014$.

When temperature>72, there are 4 yes and 2 no; when temperature<=72, there are 5 yes and 3 no. We can get $Gain(temperature) = 0.014$.

When temperature>80, there are 2 yes and 1 no; when temperature<=80, there are 7 yes and 4 no. We can get $Gain(temperature) = 0$.

When temperature>81, there are 1 yes and 1 no; when temperature<=80, there are 8 yes and 4 no. We can get $Gain(temperature) = 0.01$.

The biggest information gain of temperature is when divide temperature by >64 and <=64. So we choose 64 as the threshold in temperature.

And then we can get the gain ratio of temperature is:

$$GainRatio(temperature) = \frac{Gain(temperature)}{splitI(temperature)} = \frac{0.047}{0.893} = 0.053$$

We use same way to calculate humidity, first we also need to ascend the humidity.

Table 3.4 humidity condition 1

humidity	65	70	70	70	75	78	80	80	80	85	90	90	95	96
play	yes	no	yes	yes	yes	yes	yes	no	yes	no	yes	no	no	yes

From the same method to deal with continuous data in temperature, vj will be 65, 70, 80, 85, 90. So we need to calculate 5 times to choose one with biggest entropy gain.

As the same way in temperature, we can get when threshold is 65, $Gain(humidity) = 0.047$

When threshold is 70, $Gain(humidity) = 0.014$

When threshold is 80, $Gain(humidity) = 0.102$

When threshold is 85, $Gain(humidity) = 0.025$

When threshold is 90, $Gain(humidity) = 0.01$

The biggest information gain of humidity is when divide humidity by >80 and <=80. We choose 80 as the threshold. And we can also get humidity gain ratio is:

$$GainRatio(humidity) = \frac{Gain(humidity)}{splitI(humidity)} = 0.12$$

Compare in outlook, humidity and temperature, the biggest gain ratio is outlook. So outlook will be the first split node, and we can split data in the following structure .

```
sunny,85,85,FALSE,no

sunny,80,90,TRUE,no

sunny,72,95,FALSE,no

sunny,69,70,FALSE,yes

sunny,75,70,TRUE,yes


overcast,83,78,FALSE,yes

overcast,64,65,TRUE,yes

overcast,72,90,TRUE,yes

overcast,81,75,FALSE,yes


rainy,70,96,FALSE,yes

rainy,68,80,FALSE,yes

rainy,65,70,TRUE,no

rainy,75,80,FALSE,yes

rainy,71,80,TRUE,no
```

The classification values are the same when the node is overcast, so stop splitting in overcast branch. And we then need to consider sunny and rainy situation.

In sunny situation, we need to consider the next split node. They can be windy, temperature or humidity. So I will calculate the gain ratio of these features and they are totally same as above steps. $I(s_1, s_2) = I\left(3, 2\right) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$, $splitI(windy) = 0.951$,

$Gain(windy) = 0.02$.

$$GainRatio(windy) = \frac{Gain(windy)}{splitI(windy)} = 0.021.$$

Table 3.5 temperature condition 2

temperature	69	72	75	80	85
play	yes	no	yes	no	no

vj will be 69, 72, 75, 80.

When temperature>69, $Gain(temperature) = 0.571$.

When temperature>72, $Gain(temperature) = 0.021$.

When temperature>75, $Gain(temperature) = 0.421$.

When temperature>80, $Gain(temperature) = 0.171$.

We can see when it is divided by 69, temperature has biggest gain. So we choose 69 as the threshold of temperature.

$$GainRatio(temperature) = \frac{Gain(temperature)}{splitI(temperature)} = 1.4275$$

Table 3.6 humidity condition 2

humidity	70	70	85	90	95
play	yes	yes	no	no	no

vj will be 70, 85.

When humidity>70, $Gain(humidity) = 0.971$.

When humidity>85, $Gain(humidity) = 0.421$.

We can see when it is divided by 70, humidity has biggest gain. So we choose 70 as the threshold of humidity.

$$GainRatio(humidity) = \frac{Gain(humidity)}{splitI(humidity)} = \infty$$

Because humidity has biggest gain ratio, the next split node is humidity. $E(humidity) = 0$, so we stop splitting.

And in rainy situation, $I(s_1, s_2) = I(3,2) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$

We also need to decide which feature will be the next split node by calculating their gain ratio.

$splitI(windy) = 0$, $Gain(windy) = 0.971$. Gain ratio of windy will be ∞, gain ratio of other features can't be larger than ∞. So next split node is windy and then stop splitting, the decision tree is as follow:

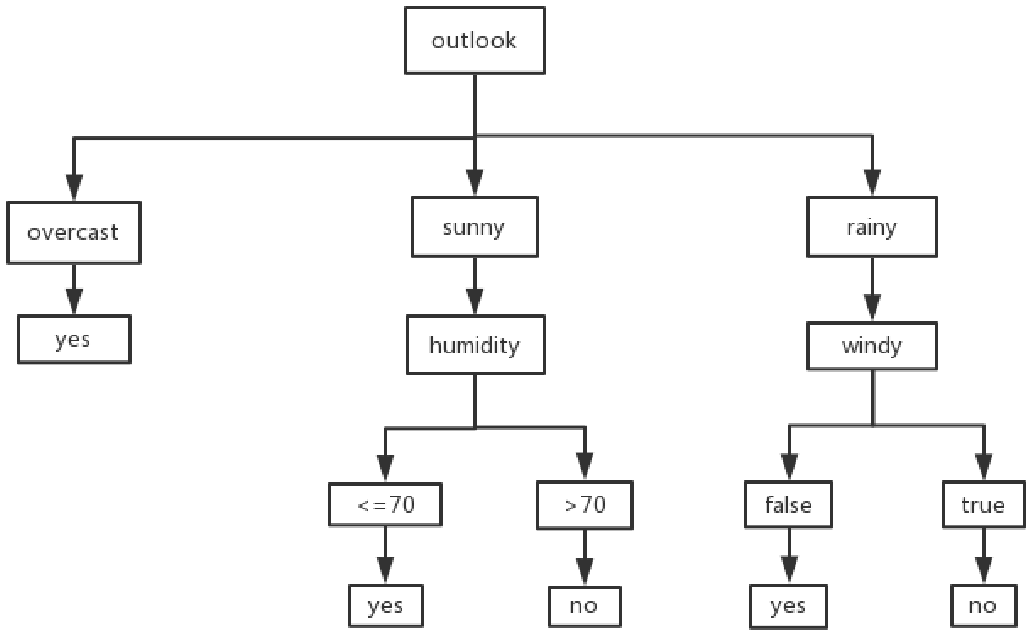


Figure 3.3 C4.5 decision tree example

Here is the steps:

Input: an attribute-valued dataset D

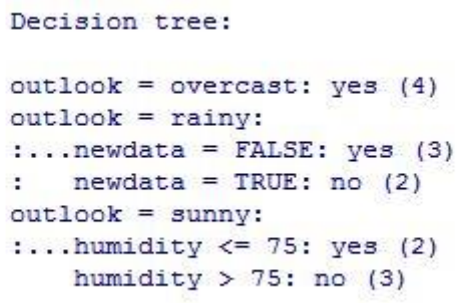
```
1 Tree={}
2 If D is"pure" or other stopping criteria met then
3 Terminate
4 End if
5 For all attribute $a \in D$ do
6 Compute information theoretic criteria if we split on a
7 End for
8 $a_{best} =$ best attribute according to above compute criteria
```

```
9 Tree=create a decision node that tests $a_{best}$ in the root
10 $D_v$ =induced sub-datasets from D based on $a_{best}$
11 For all $D_v$ do
12 $tree_v$ =C4.5( $D_v$ )
13 Attach $tree_v$ to the corresponding branch of tree
14 End for
15 Return tree
```

There is a package C50 in R, we can use it directly.

```
R code:
sample<-read.csv("sample.csv")
newdata<-as.factor(sample$windy)
newdata1<-data.frame(sample[,-4],newdata)
model<-C5.0(newdata1[,-4],newdata1$play)
summary(model)
```

I got the decision tree as follows:



```
Decision tree:

outlook = overcast: yes (4)
outlook = rainy:
:...newdata = FALSE: yes (3)
:   newdata = TRUE: no (2)
outlook = sunny:
:...humidity <= 75: yes (2)
    humidity > 75: no (3)
```

Figure 3.4 C4.5 decision tree result

3.3 CART

It’s called classification and regression tree, the algorithm can create simple classification binary tree and also regression tree. These two tree will be a little different when building and here I only consider classification tree. It use Gini coefficient to select the feature as split node and then get binary tree.

If data set S has n classes, $Gini(S) = 1 - \sum_{i=1}^{n} p_i^2$

pi is the possibility of ith data, the less gini coefficient is, the better the split node is. And if S split to S1 and S2, $splitGini(S) = \frac{|S_1|}{|S|} Gini(S_1) + \frac{|S_2|}{|S|} Gini(S_2)$

I will use the same data set in C4.5 algorithm (Fürnkranz, n.d.). Because this algorithm will create a binary tree, I need to divide some features into two groups. For example, I may divide outlook feature into sunny and not sunny. As continuous features, we use the same way as C4.5 algorithm to find threshold.

Actually there are many different way to divide features, we can divide outlook feature into sunny and not sunny. Then I calculate the split gini of outlook.

$$Gini(sunny) = 1 - \left[\left(\frac{2}{5} \right)^2 + \left(\frac{3}{5} \right)^2 \right] = \frac{12}{25}$$

$$Gini(not\_sunny) = 1 - \left[\left(\frac{2}{9} \right)^2 + \left(\frac{7}{9} \right)^2 \right] = \frac{28}{81}$$

out $splitGini(outlook) = \frac{5}{14} \times \frac{12}{25} + \frac{9}{14} \times \frac{28}{81} = 0.394$

Or in the same way showed above, it can be divided by rainy and not rainy,

$splitGini(outlook) = 0.457$.

Or it can be divided by overcast and not overcast, $splitGini(outlook) = 0.357$.

The smallest split gini of outlook is 0.357, so I divided outlook feature into overcast and not overcast. In the same way, I divided windy into true and not true and get the split gini of windy is $splitGini(windy) = 0.429$.

From the calculation in chapter 3.2 C4.5, we know that when it is divided by 64, temperature has biggest gain. So I divided temperature into larger than 64 and not larger than 64, and get the split gini of temperature is $splitGini(temperature) = 0.44$.

As the same calculation method, we can get the split gini of humidity is

$splitGini(humidity) = 0.394$.

The smallest split gini is outlook, so the first node is outlook. Two branches are overcast and not overcast. We can divide data into data sets as follow:

```
overcast,83,78,FALSE,yes

overcast,64,65,TRUE,yes

overcast,72,90,TRUE,yes

overcast,81,75,FALSE,yes


rainy,70,96,FALSE,yes

rainy,68,80,FALSE,yes

rainy,65,70,TRUE,no

rainy,75,80,FALSE,yes

rainy,71,80,TRUE,no

sunny,85,85,FALSE,no

sunny,80,90,TRUE,no

sunny,72,95,FALSE,no

sunny,69,70,FALSE,yes

sunny,75,70,TRUE,yes
```

So there are overcast and not overcast these two situations. We then need to decide next split node in these two situations. For overcast, all classification are yes, it stops splitting. So we just need to consider not overcast situation. I will calculate the classification entropy and split gini of each feature.

$$I(s_1, s_2) = I(5,5) = -\frac{5}{10}\log_2\frac{5}{10} - \frac{5}{10}\log_2\frac{5}{10} = 1$$

For windy feature, $splitGini(wind) = 0.417$.

For temperature feature, it's continuous and same way in chapter 3.2 C4.5.

Table 3.7 temperature condition 3

temperature	65	68	69	70	71	72	75	75	80	85
play	no	yes	yes	yes	no	no	yes	yes	no	no

vi can be 65, 68,71,75, 80.

When temperature is divided to >65 and <=65, $Gain(temperature) = 0.108$.

When temperature is divided to >68 and <=68, $Gain(temperature) = 0$.

When temperature is divided to >71 and <=71, $Gain(temperature) = 0.029$.

When temperature is divided to >75 and <=75, $Gain(temperature) = 0.236$.

When temperature is divided to >80 and <=80, $Gain(temperature) = 0.108$.

The biggest gain is when divided by 75, so I choose 75 as the threshold of temperature and split gini of temperature is $splitGini(temperature) = 0.375$.

For humidity feature, it's also continuous and same method.

Table 3.8 humidity condition 3

humidity	70	70	70	80	80	80	85	90	95	96
play	no	yes	yes	yes	no	yes	no	no	no	yes

vi can be 70, 80,85.

When humidity is divided to >70 and <=70, $Gain(humidity) = 0.035$.

When humidity is divided to >80 and <=80, $Gain(humidity) = 0.125$.

When humidity is divided to >85 and <=85, $Gain(humidity) = 0.035$.

The biggest gain is when divided by 80, so I choose 80 as the threshold of humidity and split gini of humidity is $splitGini(humidity) = 0.417$.

The smallest split gini is temperature, so next split node is temperature and two branches are >75 and <=75. We can divide data into data sets as follow:

```
65,70,TRUE,no

68,80,FALSE,yes

69,70,FALSE,yes

70,96,FALSE,yes

71,80,TRUE,no

72,95,FALSE,no
```

```
75,80,FALSE,yes

75,70,TRUE,yes


80,90,TRUE,no

85,85,FALSE,no
```

We get two situations, when temperature >75 all classification are no, so we don't need to split. For temperature<=75, we need to consider next split node. I calculate the splitgini of windy and humidity.

The classification entropy is $I(s_1,s_2)=I(3,5)=-\frac{3}{8}\log_2\frac{3}{8}-\frac{5}{8}\log_2\frac{5}{8}=0.954$.

$$splitGini(windy)=\frac{3}{8}\times\left[1-(\frac{1}{3})^2-(\frac{2}{3})^2\right]+\frac{5}{8}\times\left[1-(\frac{4}{5})^2-(\frac{1}{5})^2\right]=0.367$$

Table 3.9 humidity condition 4

humidity	70	70	70	80	80	80	95	96
play	no	yes	yes	yes	no	yes	no	yes

vi can be 70, 80, 95.

When humidity is divided to >70 and <=70, $Gain(humidity)=0.692$.

When humidity is divided to >80 and <=80, $Gain(humidity)=0.015$.

When humidity is divided to >95 and <=95, $Gain(humidity)=0.092$.

The biggest gain of humidity is divided by 70, so I choose 70 as the threshold of humidity.

$$splitGini(humidity)=\frac{3}{8}\times\left[1-(\frac{1}{3})^2-(\frac{2}{3})^2\right]+\frac{5}{8}\times\left[1-(\frac{3}{5})^2-(\frac{2}{5})^2\right]=0.467$$

The smaller split gini is windy, so the next split node is windy. And we can divide data into two data sets as follow:

```
70,TRUE,no

70,TRUE,yes

80,TRUE,no
```

```
80,FALSE,yes

70,FALSE,yes

96,FALSE,yes

95,FALSE,no

80,FALSE,yes
```

Then we need to consider these two situations, windy is true and windy is false. And then find next split node for these two situations.

For windy is true, the classification entropy is $I(s_1,s_2)=I(1,2)=0.918$, divided by humidity>70 and <=70.

For windy is false, the classification entropy is $I(s_1,s_2)=I(1,4)=0.722$.

Table 3.10 humidity condition 5

humidity	70	80	80	95	96
play	yes	yes	yes	no	yes

vi can be 70, 95.

When humidity is divided to >70 and <=70, $Gain(humidity)=0.073$.

When humidity is divided to >80 and <=80, $Gain(humidity)=0.322$.

When humidity is divided to >95 and <=95, $Gain(humidity)=0.073$.

The biggest gain is divided by 80, so I choose 80 as the threshold of humidity. And the decision tree is as follow:

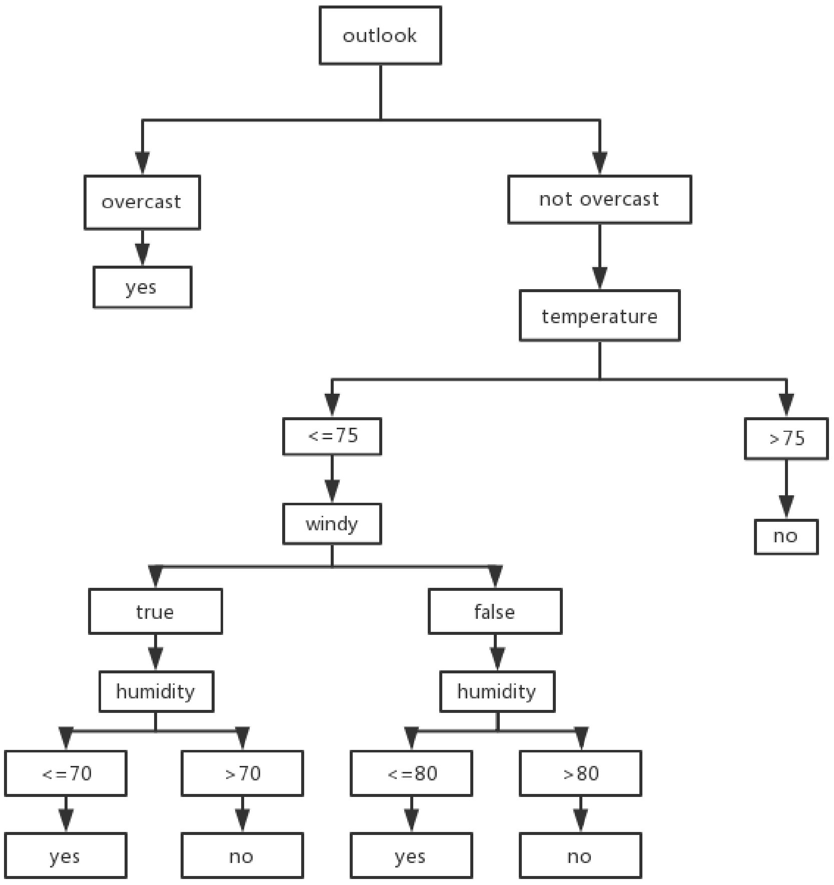
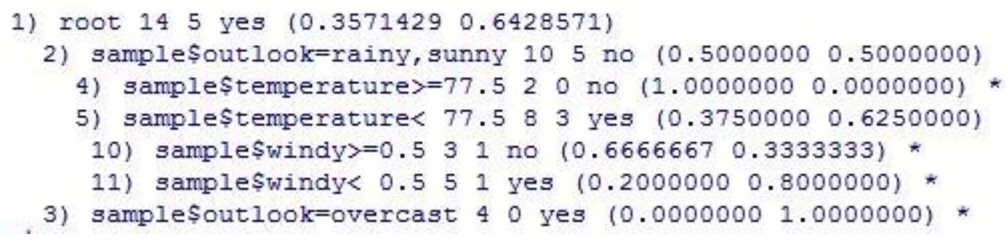


Figure 3.5 CART decision tree example

There is a package rpart in R, we can use it directly.

```
R code:

model<-rpart(sample$play~sample$outlook+sample$temperature+sample$hum
idity+sample$windy,sample,method="class",control=rpart.control(minbuc
ket=2))
```

If change the value of minbucket, the decision tree will be changed. Minbuket means the minimum amount of values a node has. We can get the decision tree as follows:



```
1) root 14 5 yes (0.3571429 0.6428571)
  2) sample$outlook=rainy,sunny 10 5 no (0.5000000 0.5000000)
    4) sample$temperature>=77.5 2 0 no (1.0000000 0.0000000) *
    5) sample$temperature< 77.5 8 3 yes (0.3750000 0.6250000)
     10) sample$windy>=0.5 3 1 no (0.6666667 0.3333333) *
     11) sample$windy< 0.5 5 1 yes (0.2000000 0.8000000) *
  3) sample$outlook=overcast 4 0 yes (0.0000000 1.0000000) *
```

Figure 3.6 CART decision tree result

4 Result in Data Set and Evaluation

Here is the comparison of these three algorithm:

Table 4.1 algorithms comparison

algorithm	feature chosen	continuous features	the volume of data set	tree structure
ID3	entropy gain	can’t deal with	normal	multiple tree
C4.5	gain ratio	can deal with	large	multiple tree
CART	gini coefficient	can deal with	large	binary tree

These three algorithms use different ways to choose best split feature and there are also several ways to decide when to stop splitting.

1 if entropy gain/ gain ratio/ gini coefficient is less than a threshold, stop splitting (when entropy is 0, it’s a special situation, stop splitting).

2 if all classification values are the same at one feature, stop splitting.

I choose 10 records in original data set (data came from kaggle.com) randomly and put them in the table below to show what kind of data do we have:

Table 4.2 data record

satisfy	evaluate	project	hour	time	accident	promotion	sale	salary	left
0.43	0.48	2	136	3	0	0	RandD	high	1
0.4	0.49	2	160	3	0	0	marketing	low	1
0.11	0.84	7	310	4	0	0	sales	medium	1
0.84	0.82	5	240	5	0	0	accounting	medium	1
0.84	0.84	5	238	5	0	0	support	medium	1
0.51	0.6	7	243	5	0	0	technical	medium	1
0.66	0.91	5	248	4	0	0	management	low	1
0.42	0.56	2	137	3	0	0	marketing	low	1
0.74	0.64	4	268	3	0	0	sales	low	0
0.56	0.58	4	258	3	0	0	sales	medium	0

This table contains 10 attributes, the last one is classification attribute to decide whether the employee leaves. And here are some description of these attributes:

satisfy: Level of satisfaction (from 0 score to 1 score, 1 is full mark), numeric

evaluate: Last evaluation of employee performance (from 0 score to 1 score, 1 is full mark), numeric

project: Number of projects this employee completed while he/she is at work, int

hour: Average monthly hours the employee has, int

time: Number of years this employee spent in the company, int

accident:Whether the employee had a workplace accident (1 means had or 0 means not), int

promotion: Whether the employee was promoted in the last five years (1 means yes or 0 means no), int

sale: Department in which they work for, factor

salary: Relative level of salary (low medium high), factor

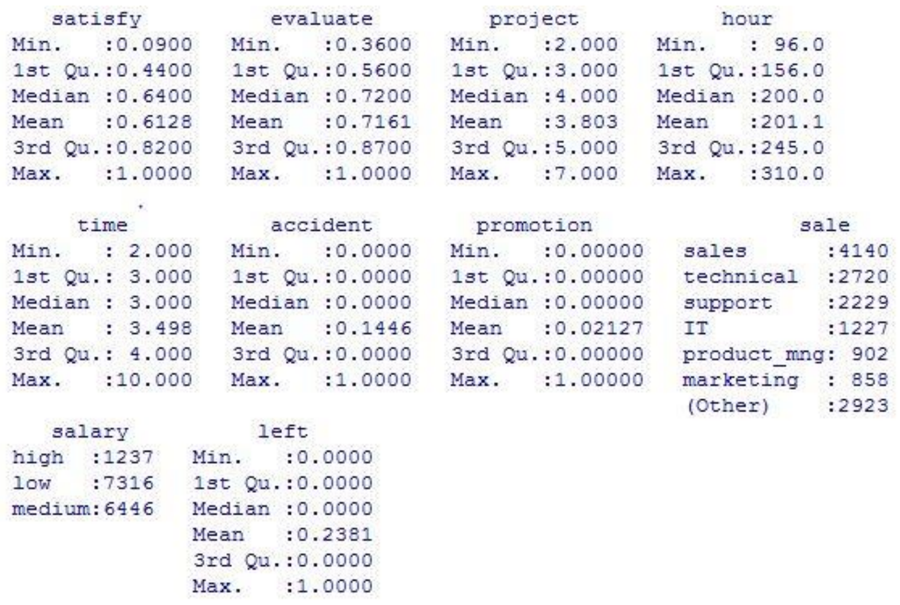
left: Whether the employee left the workplace or not (1 means yes or 0 means no), int

As we can see, in our employee data set, there are both continuous data and dispersed data. So ID3 algorithm is not very good. As the features and records are too much in data set, I think binary tree will show the result better compared with multiple tree. So I will use CART algorithm in R to deal with these data.

4.1 Pretreatment

We read employee data into R and to see some basic information in the data set, which is as follow:

```
data<-read.csv("employee.csv")

summary(data)
```

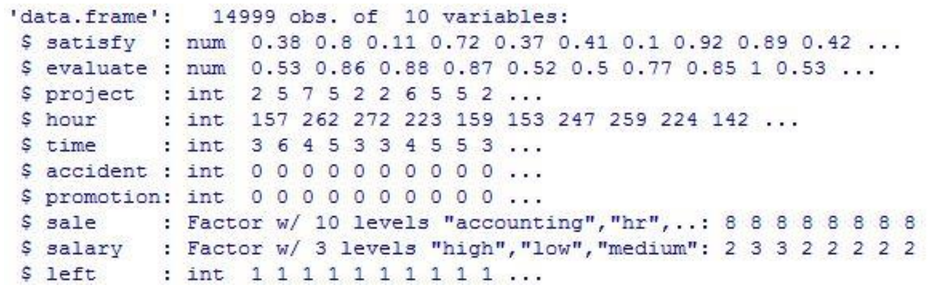


```
    satisfy          evaluate          project            hour
 Min.   :0.0900   Min.   :0.3600   Min.   :2.000   Min.   : 96.0
 1st Qu.:0.4400   1st Qu.:0.5600   1st Qu.:3.000   1st Qu.:156.0
 Median :0.6400   Median :0.7200   Median :4.000   Median :200.0
 Mean   :0.6128   Mean   :0.7161   Mean   :3.803   Mean   :201.1
 3rd Qu.:0.8200   3rd Qu.:0.8700   3rd Qu.:5.000   3rd Qu.:245.0
 Max.   :1.0000   Max.   :1.0000   Max.   :7.000   Max.   :310.0
      time            accident          promotion              sale
 Min.   : 2.000   Min.   :0.0000   Min.   :0.00000   sales      :4140
 1st Qu.: 3.000   1st Qu.:0.0000   1st Qu.:0.00000   technical  :2720
 Median : 3.000   Median :0.0000   Median :0.00000   support    :2229
 Mean   : 3.498   Mean   :0.1446   Mean   :0.02127   IT         :1227
 3rd Qu.: 4.000   3rd Qu.:0.0000   3rd Qu.:0.00000   product_mng: 902
 Max.   :10.000   Max.   :1.0000   Max.   :1.00000   marketing  : 858
                                                     (Other)    :2923
    salary          left
 high  :1237   Min.   :0.0000
 low   :7316   1st Qu.:0.0000
 medium:6446   Median :0.0000
               Mean   :0.2381
               3rd Qu.:0.0000
               Max.   :1.0000
```

Figure 4.1 summary result

We can see from figure 4.1 summary result that the mean score of satisfaction level is around 0.61 and last evaluation average mark is around 0.72. And almost each employee works on 3 to 4 projects a year and about 201 hours per month. What's more, the average work years are around 3.5 years. The mean turnover rate is equal to 23.81%. There is no data missing in this data set, so we don't need to fill missing data. If in some data sets there is data missing, we need to delete empty place or fill empty by experience or similar data.

```
R code:

str(data)
```



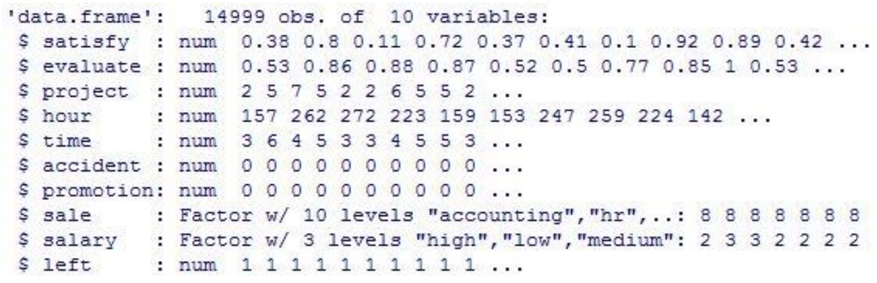
```
'data.frame':   14999 obs. of  10 variables:
 $ satisfy  : num  0.38 0.8 0.11 0.72 0.37 0.41 0.1 0.92 0.89 0.42 ...
 $ evaluate : num  0.53 0.86 0.88 0.87 0.52 0.5 0.77 0.85 1 0.53 ...
 $ project  : int  2 5 7 5 2 2 6 5 5 2 ...
 $ hour     : int  157 262 272 223 159 153 247 259 224 142 ...
 $ time     : int  3 6 4 5 3 3 4 5 5 3 ...
 $ accident : int  0 0 0 0 0 0 0 0 0 0 ...
 $ promotion: int  0 0 0 0 0 0 0 0 0 0 ...
 $ sale     : Factor w/ 10 levels "accounting","hr",..: 8 8 8 8 8 8 8 8
 $ salary   : Factor w/ 3 levels "high","low","medium": 2 3 3 2 2 2 2 2
 $ left     : int  1 1 1 1 1 1 1 1 1 1 ...
```

Figure 4.2 data structure result 1

From figure 4.2, we can see data types of satisfy and evaluate are number, data types of sale and salary are factor, rests are int. Latter we will analyze the correlation, because correlation can only be used in numeric data, I need to turn other data into numeric data.

```
data$project<-as.numeric(data$project)
```

```
data$hour<-as.numeric(data$hour)

data$time<-as.numeric(data$time)

data$accident<-as.numeric(data$accident)

data$promotion<-as.numeric(data$promotion)

data$left<-as.numeric(data$left)

str(data)
```



```
'data.frame':   14999 obs. of  10 variables:
 $ satisfy  : num  0.38 0.8 0.11 0.72 0.37 0.41 0.1 0.92 0.89 0.42 ...
 $ evaluate : num  0.53 0.86 0.88 0.87 0.52 0.5 0.77 0.85 1 0.53 ...
 $ project  : num  2 5 7 5 2 2 6 5 5 2 ...
 $ hour     : num  157 262 272 223 159 153 247 259 224 142 ...
 $ time     : num  3 6 4 5 3 3 4 5 5 3 ...
 $ accident : num  0 0 0 0 0 0 0 0 0 0 ...
 $ promotion: num  0 0 0 0 0 0 0 0 0 0 ...
 $ sale     : Factor w/ 10 levels "accounting","hr",..: 8 8 8 8 8 8 8
 $ salary   : Factor w/ 3 levels "high","low","medium": 2 3 3 2 2 2 2
 $ left     : num  1 1 1 1 1 1 1 1 1 1 ...
```

Figure 4.3 data structure result 2

Figure 4.3 shows the result after we changing the data type. Then we need to find the correlations between each variable, as figure 4.4 shows the size of bubbles represents the significance of the correlation. The blue color means variables with positive relationship and the red color means variables with negative relationship.

```
c<-data.frame(data)

correlation<-c[,-c(8:9)]

m<-cor(correlation)

library("corrplot")

corrplot(m)
```

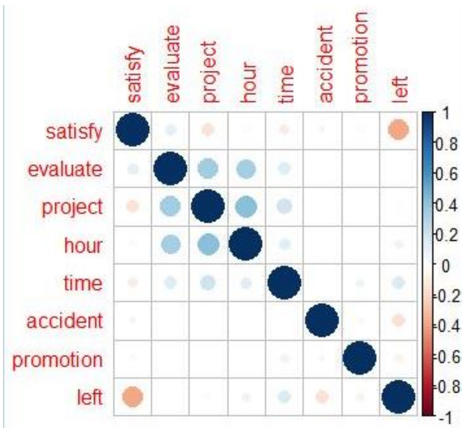


Figure 4.4 correlation significance

We can see from figure 4.4 that satisfy, accident and promotion have negative correlation with left. Hour and time have positive correlation with left. And satisfy, time and accident have more significance. So we can conclude people who have low satisfaction level, low accident and work more but didn't get promoted within the past five years will have more possibility to leave. Then we do some box plots as figure 4.5, figure 4.6 and figure 4.7 to see the relationship between two different features and use hist graph as figure 4.8 and figure 4.9 to see some information in left employees.

```
library("ggplot2")
ggplot(data,aes(x=salary,y=satisfy,fill=factor(left),colour=factor(le
ft)))+geom_boxplot(outlier.colour="black")+xlab("salary")+ylab("satis
fy")
ggplot(data,aes(x=salary,y=time,fill=factor(left),colour=factor(left)
))+geom_boxplot(outlier.colour="black")+xlab("salary")+ylab("time")
ggplot(data,aes(x=salary,y=hour,fill=factor(left),colour=factor(left)
))+geom_boxplot(outlier.colour="black")+xlab("salary")+ylab("hour")
```

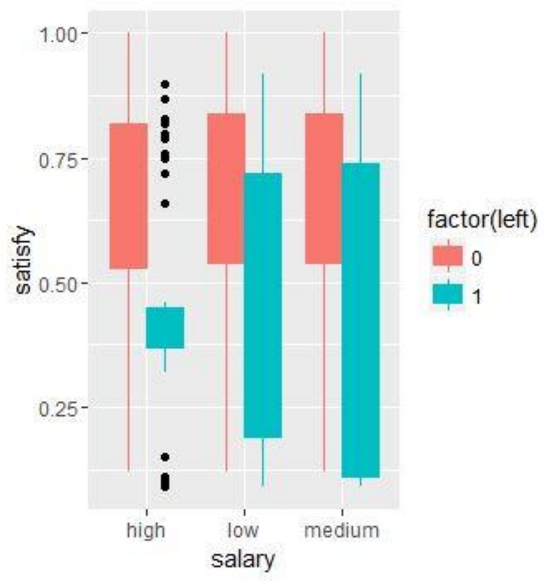


Figure 4.5 salary and satisfy

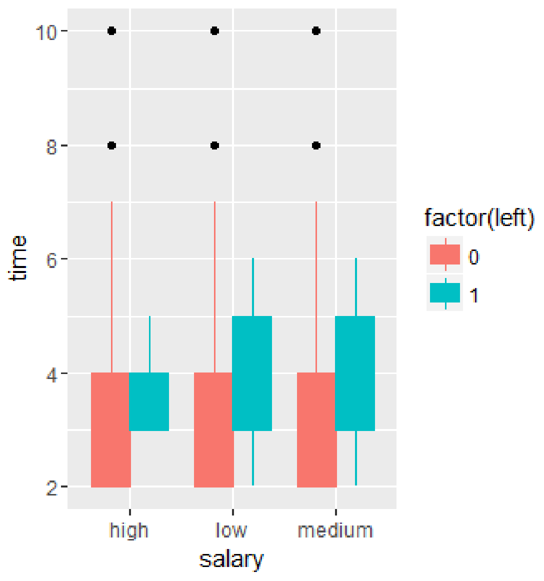


Figure 4.6 salary and time

We can see from the box plot of figure 4.5, between salary and satisfaction level, people with lower and medium salary prefer to leaving more. And when they have similar salary, people who left have lower satisfaction level than those who remain. From box plot of figure 4.6, between salary and time, people leave more when they worked many years with experience in companies but only have low and medium salary.

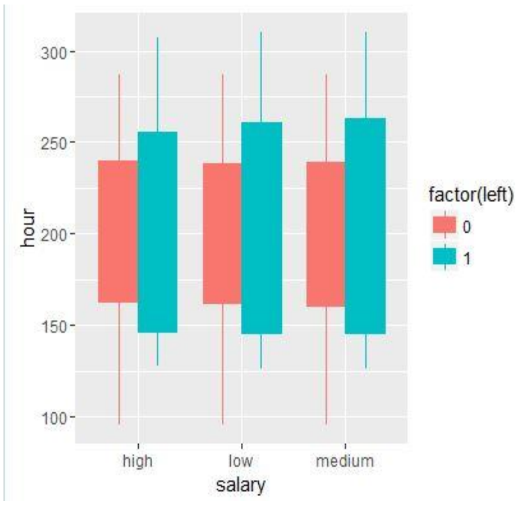
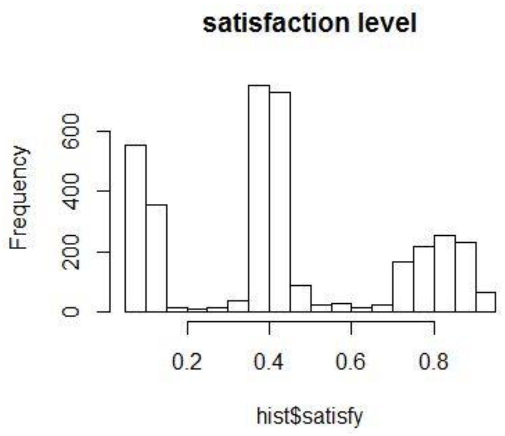


Figure 4.7 salary and hour

From box plot of figure 4.7, between salary and hour, people with low and medium salary but work more prefer to leaving.

```
hist<-subset(c,left==1)

hist(hist$satisfy,main="satisfaction level")

hist(hist$evaluate,main="last evaluation")
```



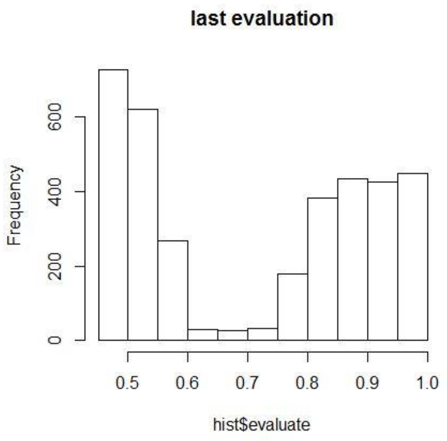


Figure 4.8 satisfaction level

Figure 4.9 last evaluation

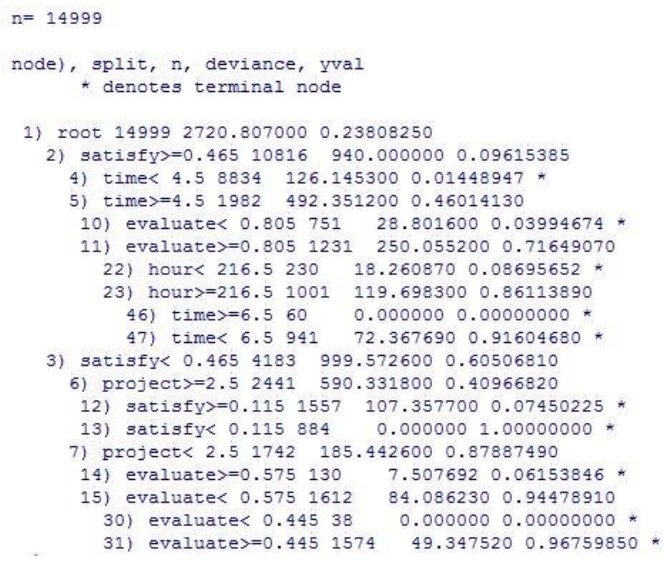
From hist graph of figure 4.8 and figure 4.9, we can see people with good evaluation and satisfaction level may also leave the company.

So I guess the possible reasons for people leaving are: in general, people work much but didn’t get enough payment and promotion. For some experienced people, they may think their work doesn’t have any challenges, they want more changes.

4.2 Build Model

I will use rpart package in R to build model, R code:

```
library("rpart")

library("rpart.plot")

model<-rpart(left~.,data=data)

model
```



```
n= 14999

node), split, n, deviance, yval
      * denotes terminal node

 1) root 14999 2720.807000 0.23808250
   2) satisfy>=0.465 10816  940.000000 0.09615385
     4) time< 4.5 8834  126.145300 0.01448947 *
     5) time>=4.5 1982  492.351200 0.46014130
      10) evaluate< 0.805 751   28.801600 0.03994674 *
      11) evaluate>=0.805 1231  250.055200 0.71649070
        22) hour< 216.5 230   18.260870 0.08695652 *
        23) hour>=216.5 1001  119.698300 0.86113890
          46) time>=6.5 60    0.000000 0.00000000 *
          47) time< 6.5 941   72.367690 0.91604680 *
   3) satisfy< 0.465 4183  999.572600 0.60506810
     6) project>=2.5 2441  590.331800 0.40966820
      12) satisfy>=0.115 1557  107.357700 0.07450225 *
      13) satisfy< 0.115 884    0.000000 1.00000000 *
     7) project< 2.5 1742  185.442600 0.87887490
      14) evaluate>=0.575 130    7.507692 0.06153846 *
      15) evaluate< 0.575 1612   84.086230 0.94478910
        30) evaluate< 0.445 38    0.000000 0.00000000 *
        31) evaluate>=0.445 1574   49.347520 0.96759850 *
```

Figure 4.10 decision tree rules 1

```
rpart.plot(model)
```

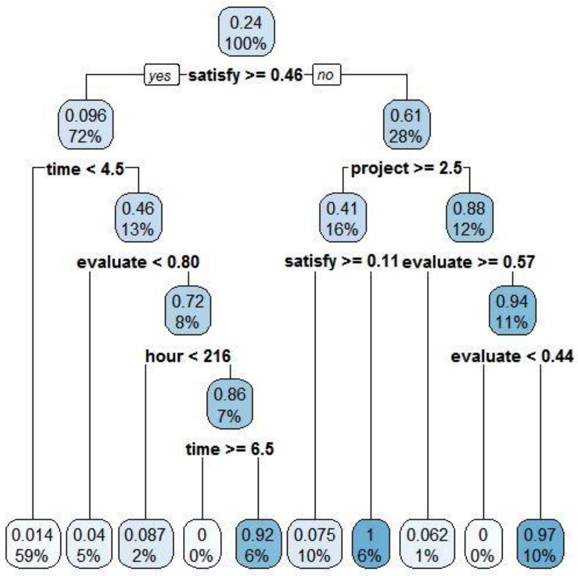
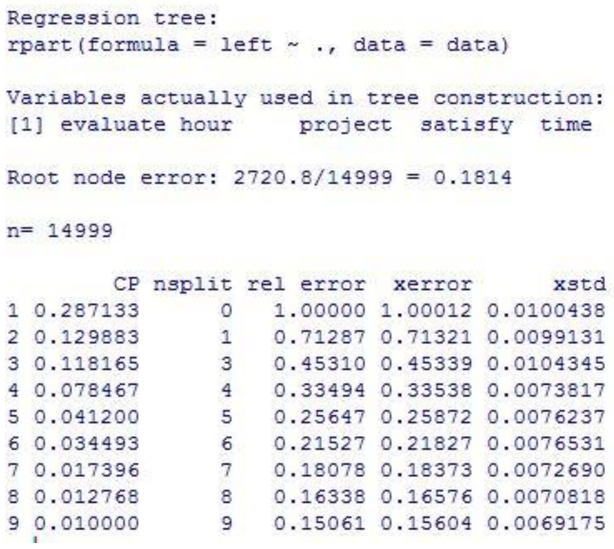


Figure 4.11 decision tree result 1

```
printcp(model)
```



```
Regression tree:
rpart(formula = left ~ ., data = data)

Variables actually used in tree construction:
[1] evaluate hour     project  satisfy  time

Root node error: 2720.8/14999 = 0.1814

n= 14999

        CP nsplit rel error  xerror      xstd
1 0.287133      0   1.00000 1.00012 0.0100438
2 0.129883      1   0.71287 0.71321 0.0099131
3 0.118165      3   0.45310 0.45339 0.0104345
4 0.078467      4   0.33494 0.33538 0.0073817
5 0.041200      5   0.25647 0.25872 0.0076237
6 0.034493      6   0.21527 0.21827 0.0076531
7 0.017396      7   0.18078 0.18373 0.0072690
8 0.012768      8   0.16338 0.16576 0.0070818
9 0.010000      9   0.15061 0.15604 0.0069175
```

Figure 4.12 cp value

We can see from figure 4.11, in general situation, satisfy, project, evaluate, time, hour these five features will construct decision tree. Our guess is not very right, the reasons don’t have too much relationship with salary. And the rules we can conclude from decision tree as figure 4.10 shows, they are:

Rule1: if satisfy<0.465, number>=2.5, satisfy>=0.115, won’t leave.

Rule2: if satisfy<0.465, number>=2.5, satisfy<0.115, will leave.

Rule3: if satisfy<0.465, number<2.5, evaluate>=0.575, won’t leave.

Rule4: if satisfy<0.465, number<2.5, evaluate<0.575, evaluate>=0.445, won’t leave.

Rule5: if satisfy<0.465, number<2.5, evaluate<0.575, evaluate<0.445, will leave.

Rule6: if satisfy>=0.465, time<4.5, won’t leave.

Rule7: if satisfy>=0.465, time>=4.5, evaluate<0.805, won’t leave.

Rule8: if satisfy>=0.465, time>=4.5, evaluate>=0.805, hour<216.5, won’t leave.

Rule9: if satisfy>=0.465, time>=4.5, evaluate>=0.805, hour>=216.5, time<6.5, won’t leave.

Rule10: if satisfy>=0.465, time>=4.5, evaluate>=0.805, hour>=216.5, time>=6.5, will leave.

Satisfaction level appears to be the most important reason. So in brief,

1 If the satisfaction level is very low, employees will leave; If satisfaction level is above 0.46, you are much more likely to stay.

2 If employees have a low satisfaction and evaluation and don’t have many projects, they will

leave.

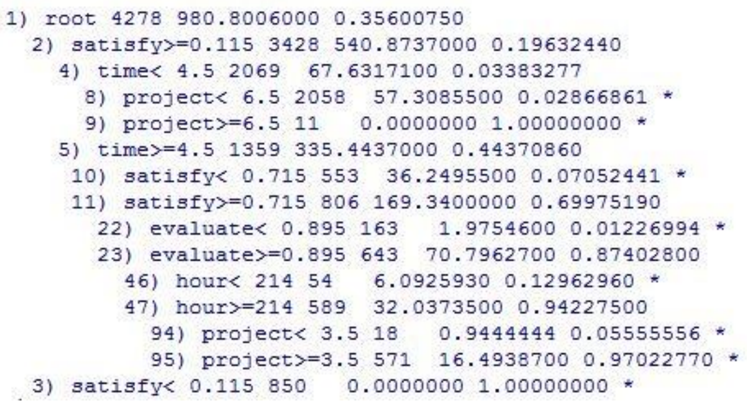
3 If employees stay in the company for more than 6.5 years and have good satisfaction and evaluation, but they work too much every month they will leave.

```
nrow(hist)
good_employee_leaving<-subset(hist,evaluate>=0.9|time>=8|project>=6)
nrow(good_employee_leaving)
```

We can get the result that there are 3571 people leave, and there are 1523 good employees leave. There are almost half of employees who are good employees leave the company. Good means they have high evaluation or working experience. I set people who have evaluation larger than 0.9 or working time is larger than 8 years or have larger than 6 projects as good employees. I want to find do good employees leave the company for same reasons as employees in general situation.

Then I built a decision tree model for good employees. Actually it’s hard to define what is good employee. Different standard of good employees will result in different decision tree rules.

```
good_employee<-subset(data,evaluate>=0.9|time>=8|project>=6)
model1<-rpart(left~.,data=good_employee)
model1
```



```
1) root 4278 980.8006000 0.35600750
  2) satisfy>=0.115 3428 540.8737000 0.19632440
    4) time< 4.5 2069  67.6317100 0.03383277
      8) project< 6.5 2058  57.3085500 0.02866861 *
      9) project>=6.5 11   0.0000000 1.00000000 *
    5) time>=4.5 1359 335.4437000 0.44370860
     10) satisfy< 0.715 553  36.2495500 0.07052441 *
     11) satisfy>=0.715 806 169.3400000 0.69975190
       22) evaluate< 0.895 163   1.9754600 0.01226994 *
       23) evaluate>=0.895 643  70.7962700 0.87402800
         46) hour< 214 54   6.0925930 0.12962960 *
         47) hour>=214 589  32.0373500 0.94227500
           94) project< 3.5 18   0.9444444 0.05555556 *
           95) project>=3.5 571  16.4938700 0.97022770 *
  3) satisfy< 0.115 850   0.0000000 1.00000000 *
```

Figure 13 decision tree rules 2

```
rpart.plot(model1,tweak=1.2)
```

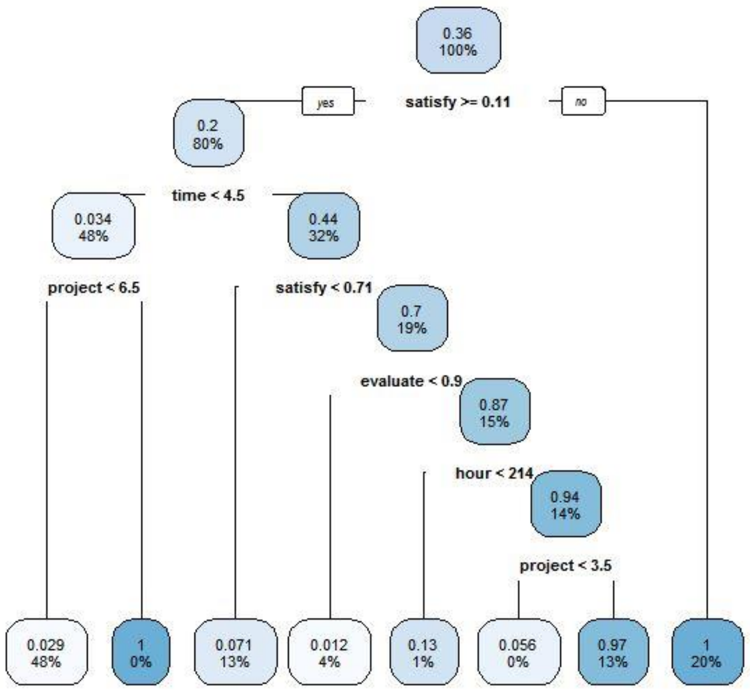


Figure 4.14 decision tree result 2

We can see from figure 4.14, for good employees (valuate>=0.9|time>=8|project>=6), satisfy, time, evaluate, hour, project these five features will construct decision tree. Satisfaction is also the most important influence for good employees. And the rules we can conclude from decision tree as figure 4.13 shows:

Rule1: if satisfy<0.11, will leave.

Rule2: if satisfy>=0.11, time<4.5, project<6.5, won’t leave.

Rule3: if satisfy>=0.11, time<4.5, project>=6.5, will leave.

Rule4: if satisfy>=0.11, time>=4.5, satisfy<0.71, won’t leave.

Rule5: if satisfy>=0.11, time>=4.5, satisfy>=0.71, evaluate<0.9, won’t leave.

Rule6: if satisfy>=0.11, time>=4.5, satisfy>=0.71, evaluate>=0.9, hour<214, won’t leave.

Rule7: if satisfy>=0.11, time>=4.5, satisfy>=0.71, evaluate>=0.9, hour>=214, project<3.5, won’t leave.

Rule8: if satisfy>=0.11, time>=4.5, satisfy>=0.71, evaluate>=0.9, hour>=214, project>=3.5, will leave.

So in brief, we can conclude:

1 If the employees are not happy with their work, they will leave.

2 If employees stay in company for a short time but many projects experience, they will leave.
3 If employees are happy with their work and they have some experience, but they work too much they will leave.

Combine general situation and analysis for good employees, satisfaction level is most important for these two situations. And the conclusions are very similar, we can summary that if you are good and overworked you leave. If you are unhappy you tend to leave, especially if you are not getting enough projects experience. But there is a very interesting thing, in general situation if employees stay in the company for more than 6.5 years and have good satisfaction and evaluation, but they work too much every month, they will leave. However, for good employees, if they stay in the company for more than 4.5 years and have good satisfaction and evaluation, but they work too much every month, they will leave. I think the reason may be if good employees stay in a company for a long time but they can't get work and life balance, they will choose to find other jobs. For not so good employees, they may want to work hard to be better or get more training in the company.

4.3 Training Set and Test Set

We have 15000 records in the data set, so we need to divide them into training set and test set. There are several ways to divide training set and test set.
(1) Usually we will divide data into four parts, 1/4 of which is test set and 3/4 of which is training set. We will do this process k times randomly to build models and get average value of these k models.
(2) Cross validation, which means dividing data set into test set and training set and each record can be both training set and test set.
① Leave one out cross validation: choose one data as test set and remaining N-1 data records as training set, do N times to get average value.
② K-fold cross validation: divide data set into k subsets, choose one subset as test set, remaining k-1 as training set and do k times to get average value. Usually we make k equals to 10.

I will choose 10-fold cross validation to get training set and test set.

```
R code:

library("lattice")

library("ggplot2")

library("caret")

train_control<-trainControl(method="cv",number=10)
```

I named training set as train_control, and use trainControl( ) function in R to get training set. I then use train_control as training set to build model, rest data to be test set to evaluate the model.

4.4 Predict and Evaluate

After building models, we also need to evaluate whether this model is good. There are several metrics to evaluate models in data mining, like TPR, FPR, TNR, accuracy, precision, recall and F-measure and so on.

4.4.1 Confusion Matrix

TPR, FPR and TNR are metrics in confusion matrix. If we divide a sample into two part, there will be positive(1) and negative(0). And there will be four situation, if the sample is positive and can be predicted to be positive, it’s true positive (TP); if it is predicted to be negative, it’s false negative (FN). If the sample is negative and can be predicted to be negative, it’s true negative (TN); if it’s predicted to be positive, it’s false positive (FP).

Table 4.3 confusion matrix

	Predict 1	Predict 0
Actual 1	TP	FN
Actual 0	FP	TN

Accuracy is the bias that we measure the data from its real value (JCGM 200, 2008),

Accuracy=(TP+TN)/(TP+TN+FP+FN)=1-error rate.

Precision is correct information selected divide all information selected (JCGM 200, 2008),

precision=TP/(TP+FP).

Recall is correct information selected divide all information in sample,

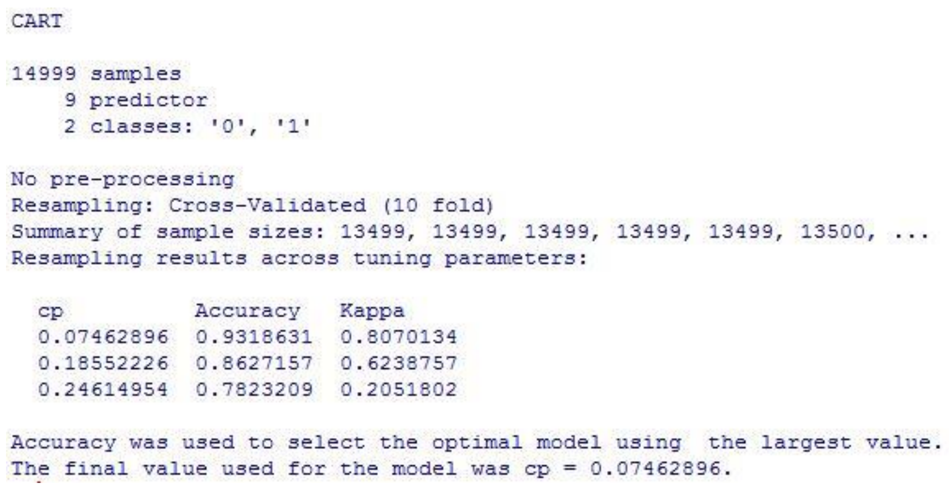
recall=TP/(TP+FN)=TPR.

Sometimes precision and recall will be contradictory, so we need to consider these two integrated.

F1=2*precision*recall/(precision+recall)=2TP/(TP+FP)(TP+FN)

When we use confusion matrix to evaluate the model, there is a package confusionMatrix( )

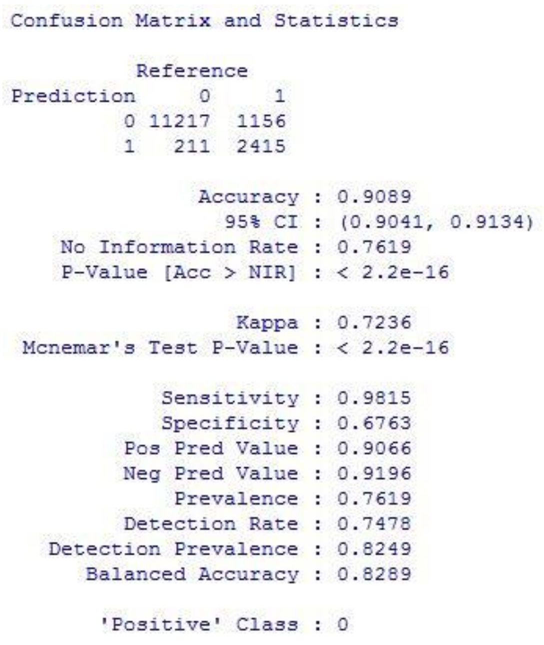
```
data$left<-as.factor(data$left)

rpartmodel<- train(left~., data=data, trControl=train_control,

method="rpart")

print(rpartmodel)
```



```
CART

14999 samples
    9 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 13499, 13499, 13499, 13499, 13499, 13500, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.07462896  0.9318631  0.8070134
  0.18552226  0.8627157  0.6238757
  0.24614954  0.7823209  0.2051802

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was cp = 0.07462896.
```

Figure 4.17 summary of the model

```
predictions<- predict(rpartmodel,data)

tree<-cbind(data,predictions)

confusionMatrix(tree$predictions,data$left)
```



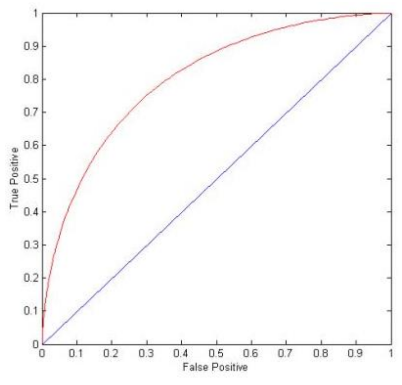
```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 11217  1156
         1   211  2415

               Accuracy : 0.9089
                 95% CI : (0.9041, 0.9134)
    No Information Rate : 0.7619
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7236
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9815
            Specificity : 0.6763
         Pos Pred Value : 0.9066
         Neg Pred Value : 0.9196
             Prevalence : 0.7619
         Detection Rate : 0.7478
   Detection Prevalence : 0.8249
      Balanced Accuracy : 0.8289

       'Positive' Class : 0
```

Figure 4.18 confusion matrix result

4.4.2 ROC and AUC

ROC (receiver operating characteristic) is a graphic method to show the relationship between true positive rate and false positive rate of a classifier.

TPR is true positive rate, TPR=TP/(TP+FN)=TP/#POS=recall, it can also be called specificity. FPR is false positive rate, FPR=FP/(FP+TN)=FP/#NEG=1-TPR. Definition of TP, FP, TN, FN can be found in table 4.3.



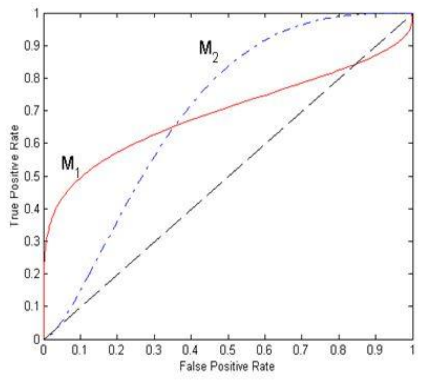


Figure 4.15 ROC Curve Figure 4.16 AUC

ROC Curve is as figure 4.15 shows, different coordinate means different classification. For example, the node (0,1) means FPR=0, TPR=1. According to the formulas of TPR and FPR, we can get FP=0 and FN=0. So this is the best situation because it does all classifications correctly. For node (1,0), which means FPR=1 and TPR=0. So we can get TP=0, TN=0. This is actually the worst situation because it avoids all correct classification. For node (0,0), we can get FPR=TPR=0, so FP=TP=0. So it declares everything to be negative class. Similar, for node (1,1), we can get FPR=TPR=1 and FP=TP=1. It declares everything to be positive class. So we can get the conclusion, the closer the ROC curve is to the upper left corner, the better the performance of the classifier.

AUC is the area under ROC Curve, we can see from figure 4.16 that ideal area=1, random guess=0.5. M1 is better for small FPR, M2 is better for large FPR.

```
library("gplots")
library("ROCR")
tree$predictions<-as.numeric(paste(tree$predictions))
perf.obj<-prediction(predictions=tree$predictions,labels=tree$left)
roc.obj<-performance(perf.obj,measure="tpr",x.measure="fpr")
auc<-slot(performance(perf.obj,"auc"),"y.values")[[1]]
plot(roc.obj)
abline(h=seq(0,1,0.05), v=seq(0,1,0.05), col = "lightgray", lty =
"dotted")
lines(c(0,1),c(0,1), col = "gray", lwd =2)
text(0.6,0.2,paste("AUC=", round(auc,4), sep=""), cex=1.4)
title("ROC Curve")
```

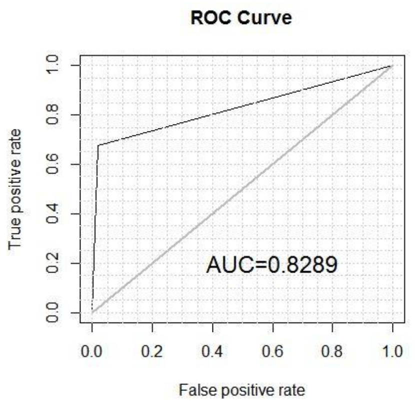


Figure 4.19 ROC curves

4.4.3 MAE, MSE, NMSE

Because we have training set and test set, in training set we have the classification result which called observations. While in test set we don’t have the classification result. We need to use model to predict the classification result, which called predictions. MAE is the mean absolute error, MSE is the mean square error, NMSE is normalised mean square error which estimates overall deviations between predictions and observations. Here are formulas of them:

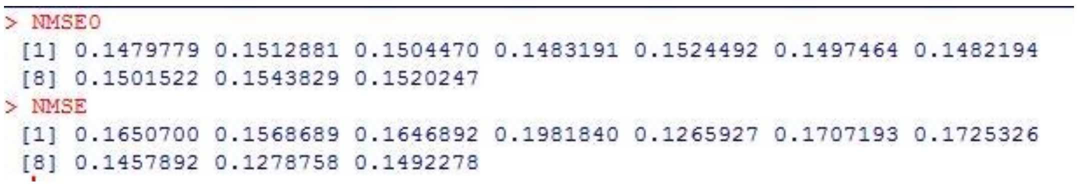
MAE=mean(abs(predictions-observations)),

MSE=mean(predictions-observations)^2,

NMSE=mean((predictions-observations)^2)/mean((mean(observations)-observations)^2).

The value of NMSE is usually between 0 and 1, and if the value is smaller, the model is better. Usually when we do cross validation, we use NMSE to value whether the cross validation is good.

```
n<-length(data$left)
index1<-1:n
index2<-rep(1:10,ceiling(n/10))[1:n]
index2<-sample(index2,n)
NMSE<-rep(0,10)
NMSE0<-NMSE
for(i in 1:10){
x<-index1[index2==i]
reg<-rpart(left~.,data[-x,])
y0<-predict(reg,data[-x,])
y1<-predict(reg,data[x,])
NMSE0[i]<-mean((data$left[-x]-y0)^2)/mean((data$left[-x]-mean(data$le
ft[-x]))^2)
NMSE[i]<-mean((data$left[x]-y1)^2)/mean((data$left[x]-mean(data$left[
x]))^2)}
```



```
> NMSE0
 [1] 0.1479779 0.1512881 0.1504470 0.1483191 0.1524492 0.1497464 0.1482194
 [8] 0.1501522 0.1543829 0.1520247
> NMSE
 [1] 0.1650700 0.1568689 0.1646892 0.1981840 0.1265927 0.1707193 0.1725326
 [8] 0.1457892 0.1278758 0.1492278
```

Figure 4.20 NMSE result

We can see from figure 4.18, the accuracy of confusion matrix is 90.89%. From figure 4.19, AUC is 0.829. And from figure 4.20, NMSE0 and NMSE don’t have too much error with each other, so the data doesn’t overfit.

5 Decision Tree Optimization

5.1 Overfitting

In classification problem, there will be overfit phenomenon. Overfitting is a model can do better than other models in training set, but when put it outside training set, it won't fit the data very well (Mansour, 1997). Overfitting results in decision trees that are more complex than necessary. There are several reasons why this phenomenon happened. The main reason is training data set is too small or there are noises in training data set. And the more complicated the model is, there is higher possibility the overfit will happen. So we always prefer simple model, which is called Occam's Razor. In order to solve overfit problem, we need to lower the complexity of the model. So we need to do prune of a tree.

5.2 Prune

Pruning is using statistic ways to delete the least reliable branches of a tree to improve the speed and ability of future data classification. For example, if we have a space H, and there is a hypothesis h and h belongs to H. If there is another h' belongs to H which makes the error rate of h is smaller than h' in training set, but in other data set the error rate of h' is smaller than h. Then we can say hypothesis h overfits training set data. There are two ways to do prune, one is post-pruning and another is pre-pruning. We need to compare training errors and generalization errors. Error at a given node t: $IM(t) = Error(t) = 1 - \max_j P(j \mid t)$. When records are equally distributed among all classes, implying least interesting information. We get the maximum value, which is $1 - \frac{1}{c}$. When all records belong to one class, implying most interesting information. We get the minimum value, which is 0.

C1	1
C2	5

$$P(C1) = \frac{1}{6}, \quad P(C2) = \frac{5}{6}, \quad Error = 1 - \max(\frac{1}{6}, \frac{5}{6}) = \frac{1}{6}$$

5.2.1 Optimistic Estimate and Pessimistic Estimate

Training error is also called resubstitution error or apparent error, it's the error in training data set and we use e(T) to represent. Generalization error is the model performs in test data set, we use e'(T) to represent.

Assume that the training set can represent the whole data very well. We can get in optimistic estimation, the generalization error equals to the training error, which is e'(T)=e(T). A decision tree induction algorithm selects the model that has lowest training error rate. Here is an example of optimistic estimate as figure 5.1 shows:

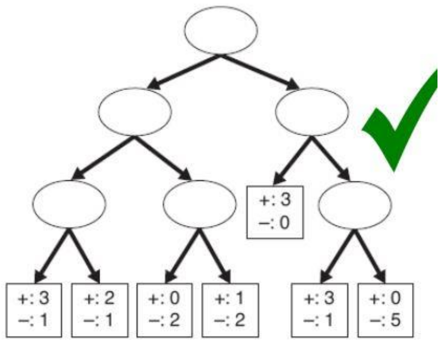


Figure 5.1 optimistic estimate

We can get in optimistic estimation, training errors=4, training error rate=4/24=16.7%, generalization errors=4, generalization error rate=4/24=16.7%.

We can get in pessimistic estimation, the generalization error is training error pluses the complexity of the model, which is $e'(T)=e(T)+\Omega(T)$. For each leaf node, $e'(t)=e(t)+\Omega(T)=e(t)+k, k>0$.

Total errors: $e'(T)=e(T)+N\times 0.5$, N is the number of leaf nodes.

For example, a tree with 40 leaf nodes and 15 errors on training (out of 1000 instances), we can get in pessimistic estimation, training errors=15, training error rate=15/1000=1%, generalization errors=15+40*0.5=35, generalization error rate=35/1000=3.5%.

5.2.2 Pre-pruning

Pre-pruning is to stop the algorithm before it becomes a fully grown tree, which means we need to determine whether it's necessary to continue dividing the training set samples when we are at current node (Sayad, 2017). This method requires more restrictive constraints, such that we stop the growth of branches when impurity is lower than a certain threshold. The advantage is this way can avoid overfiting complex sub-trees, however the problem is it's hard for us to choose right threshold. If threshold is too high, there will be underfitting; if threshold is too low, overfitting problems won't be solved totally.

5.2.3 Post-pruning

It's hard to predict when to do pre-pruning because we don't know how to set right threshold. So usually we will use post-pruning. Post-pruning is from bottom to top to prune, we need to calculate the error before and after pruning (Sayad, 2017). If error is smaller after pruning, we will do pruning. Otherwise, we won't do pruning. There are two ways to do post-pruning. One is using new leave node replaces sub-trees, another is using the most used branch replaces sub-trees.

Here is an example of post-pruning.

Case 1:

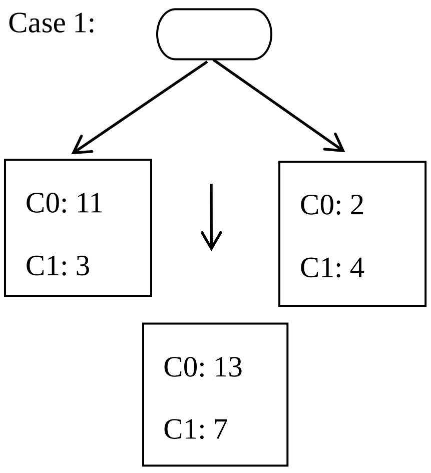


Figure 5.2 post-pruning case 1

Case 2:

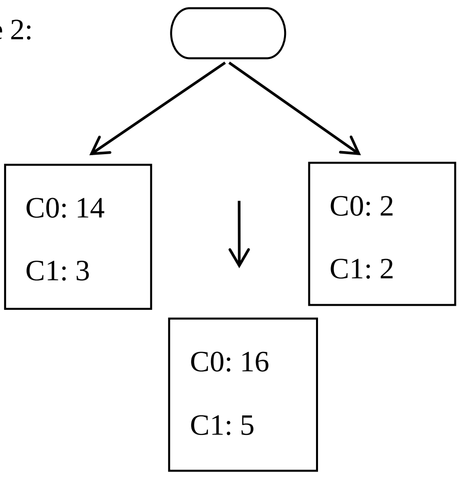


Figure 5.3 post-pruning case 2

For case 1, before prune, optimistic(error)=3+2=5, pessimistic(error)=5+2*0.5=6.

After prune, optimistic(error)=7, pessimistic(error)=7+1*0.5=7.5.

Both optimistic error and pessimistic error is larger than before, so we won't do prune.

For case 2, before prune, optimistic(error)=3+2=5, pessimistic(error)=5+2*0.5=6.

After prune, optimistic(error)=5, pessimistic(error)=5+1*0.5=5.5.

We can see optimistic error doesn't change and pessimistic error is smaller than before, we will do prune in case 2.

5.3 Optimization for Employee Data

We try to do prune for our model and evaluate the model after pruning. If the evaluation metrics are better then before, we choose the model after pruning. Otherwise, we choose the model before pruning. In R, there is a package called prune() which can do prune. We can choose proper cp value to fix the model, cp is complexity pamemeter. We need to make Xerror small and choose the smallest cp value, in R we have printcp( ) to get cp, xerror and xstd. We can see from build model part, when cp is 0.01, xerror is the smallest. So we make cp=0.01 and here is the result of after pruning.

```
R code:
model2<-prune(model,cp=0.01)
rpart.plot(model2)
```

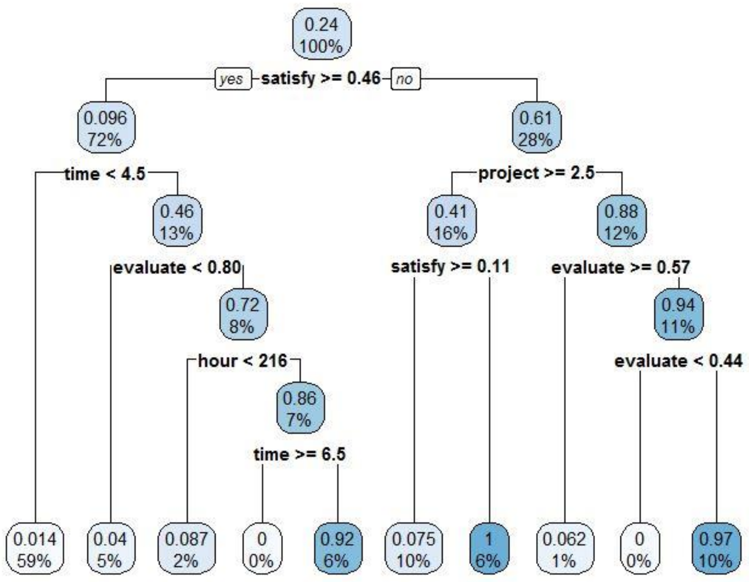
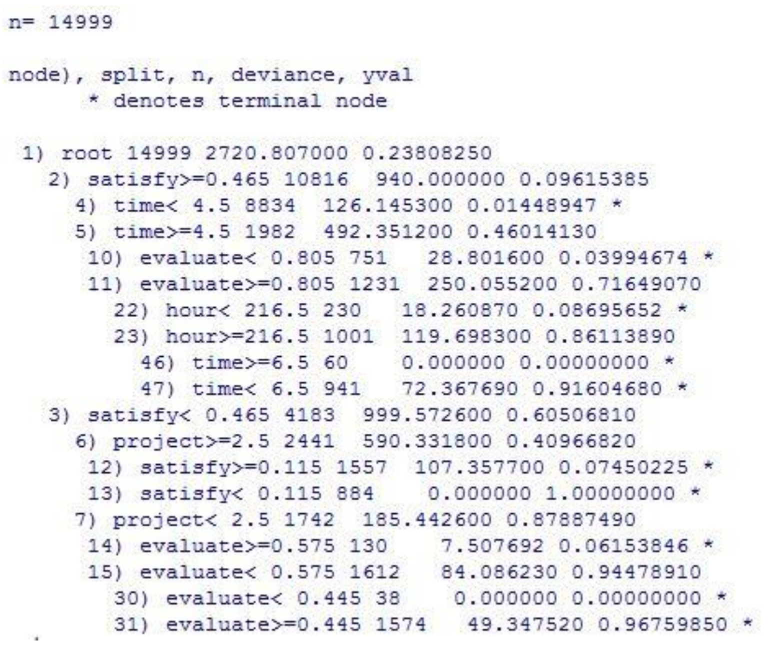


Figure 5.4 decision tree result after pruning

```
model2
```



```
n= 14999

node), split, n, deviance, yval
      * denotes terminal node

 1) root 14999 2720.807000 0.23808250
   2) satisfy>=0.465 10816  940.000000 0.09615385
     4) time< 4.5 8834  126.145300 0.01448947 *
     5) time>=4.5 1982  492.351200 0.46014130
      10) evaluate< 0.805 751   28.801600 0.03994674 *
      11) evaluate>=0.805 1231  250.055200 0.71649070
        22) hour< 216.5 230   18.260870 0.08695652 *
        23) hour>=216.5 1001  119.698300 0.86113890
          46) time>=6.5 60    0.000000 0.00000000 *
          47) time< 6.5 941   72.367690 0.91604680 *
   3) satisfy< 0.465 4183  999.572600 0.60506810
     6) project>=2.5 2441  590.331800 0.40966820
      12) satisfy>=0.115 1557  107.357700 0.07450225 *
      13) satisfy< 0.115 884    0.000000 1.00000000 *
     7) project< 2.5 1742  185.442600 0.87887490
      14) evaluate>=0.575 130    7.507692 0.06153846 *
      15) evaluate< 0.575 1612   84.086230 0.94478910
        30) evaluate< 0.445 38    0.000000 0.00000000 *
        31) evaluate>=0.445 1574   49.347520 0.96759850 *
```

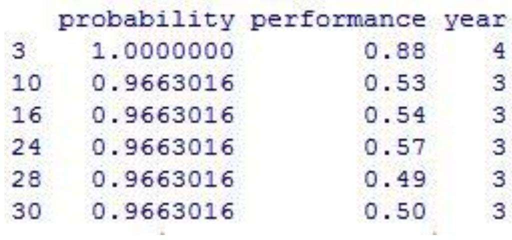
Figure 5.5 decision tree rule after pruning

We found the result hasn’t changed, so actually it doesn’t need to do pruning.

6 Suggestion

We can see from the confusion matrix and the accuracy figures that the model has 90.89% accuracy and for a Kappa is 72.36%, which shows the model is good. It's not sensible to focus on every employee who wants to leave because it costs time and energy for human resources management department. I think we need to focus on employees that have high probability to leave and also high mark in evaluation and working years. I put probability, evaluation and working years in a data frame called summary, here are some records.

```
training<-createDataPartition(data$left,p= .75,list=FALSE)
train<-data[training,]
test<-data[-training,]
model3<-rpart(left~.,data=train)
probaToLeave=predict(model3,newdata=test)
summary=data.frame(probaToLeave)
summary$performance=test$evaluate
summary$year=test$time
summary<-summary[,-1]
colnames(summary)[1]<-'probability'
head(summary)
```

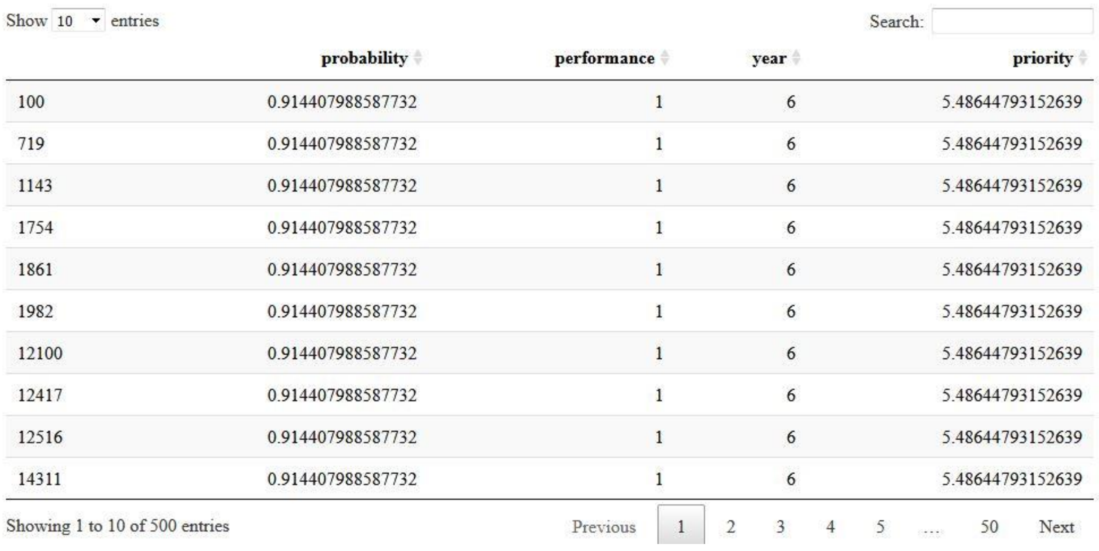


```
   probability performance year
3    1.0000000        0.88    4
10   0.9663016        0.53    3
16   0.9663016        0.54    3
24   0.9663016        0.57    3
28   0.9663016        0.49    3
30   0.9663016        0.50    3
```

Figure 6.1 prediction

Then I build a data table and rank the probability to leave and the performance as well as working years. The priority is the multiply of probability, performance and working years. Because I think people who work in a company for many years may know the company very well and can work very effectively and efficiently. People who got high performance means they can manage their work very well and can contribute to the company. So these people are most valuable for companies, we want to remain them. Therefore, we need to talk with people who have high possibility to leave and high performance mark as well as long working years.

```
library("DT")
summary$priority=summary$probability*summary$performance*summary$year
order=summary[order(summary$priority,decreasing = TRUE),]
order<-head(order,n=500)
datatable(order)
```



	probability	performance	year	priority
100	0.914407988587732	1	6	5.48644793152639
719	0.914407988587732	1	6	5.48644793152639
1143	0.914407988587732	1	6	5.48644793152639
1754	0.914407988587732	1	6	5.48644793152639
1861	0.914407988587732	1	6	5.48644793152639
1982	0.914407988587732	1	6	5.48644793152639
12100	0.914407988587732	1	6	5.48644793152639
12417	0.914407988587732	1	6	5.48644793152639
12516	0.914407988587732	1	6	5.48644793152639
14311	0.914407988587732	1	6	5.48644793152639

Showing 1 to 10 of 500 entries

Figure 6.2 prior table

I choose to present first 500 employees that the company should retain. We can see their probability to leave, performance and working years. The priority of employees that should talk with considers working years, performance and probability to leave. After grouping them in each department, we could email different managers to tell them which valuable employees might leave soon. And managers need to take some actions like talk with these valuable employees, promote these employees, reduce working hours or give higher salary to these people.

7 Conclusion

This thesis aims at the situation that human resource department faces high employees turnover in the company especially some experienced employees will leave. I use R to do data clean and transfer to get useful huge data of employees information. Then use decision tree algorithm and do optimization of decision tree. Here are some results I got through the analysis of employee data:

(1) I analyze different classify algorithms and compare the advantages and disadvantages of each algorithm. Finally I decide to use decision tree algorithm for employee data and introduce the algorithm into the research.

(2) I introduce three common decision tree algorithms, ID3, C4.5 and CART. And compare difference of these three decision tree algorithms and decide to use CART for real data set analysis.

(3) I use R to build model and find the reasons that influence employee turnover most. I found in general, satisfaction level is the major parameter to stay with the company. And number of projects also has an impact, employees with 3 to 4 projects tend to stay (maybe for life and work balance). There are only few employees with high salary, salary may not be a concern. Then I evaluate model and do pruning for the model. Last I made a table to represent the prior of employees that human resource department should have a talk with and remain them. Because not all employees we need to retain, we should consider human resource and select valuable employees to remain. This information may help decision makers get better and sensible decision and help the company run effectively and efficiently.

Of course the thesis also has some insufficient needed to be improved, and these future research could be:

(1) How to define good employees? (People with high evaluation? Long years in a company? Or many projects experience?)

(2) Is using probability to leave, performance and working years as priority a sensible way? Should we consider other influence?

(3) What polities we should make to remain employees and motivate them through the results

of employee turnover analysis so that we can reduce good employees turnover. And for different employees should we take different actions? (For experienced people with high evaluation and satisfaction, maybe we can reduce their working hours and improve salary. For people who have low satisfaction but work for a long time, maybe change the working environment or organize some events to help them get along well with other staff...)
(4) There could be some other optimization except doing pruning of CART algorithm to make it more effectively and efficiently, like do algorithm parallelization.

Due to my limited ability, the insufficient of the thesis need to be improved in the future research. And I need to learn more about different and advanced algorithms of decision tree and try to combine theory with practical problems.

Reference

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician, Volume 46, Issue 3, Pages 175–185.

Beam, J. (2014). What is Employee Turnover? Retrieved from WiseGeek: http://www.wisegeek.org/what-is-employee-turnover.html

Berson, A., and Smith, S. J. (2002). Building Data Mining Applications for CRM. New York: McGraw-Hill, Page: 245.

Bisk. (2017). Impact of Employee Turnover and Solutions to Help High employee turnover also can lower customer satisfaction and retention. University of Florida Executive Education. Retrieved from: https://essentialsofbusiness.ufexec.ufl.edu/resources/leadership/impact-of-employee-turnover-and-solutions-to-help/#.WJWzHfKAT2k

Carey, D.C. , & Ogden, D. (2004). The human side of M & A: how CEOs leverage the most important asset in deal making. Oxford University Press.

Chang, Y., & Guan, M. (2008, December). Data mining to improve human resources in construction company. International Seminar on Business and Information Management.

Chien, C.F., & Chen, L.F. (2008, January). Data mining to improve personnel selection and enhance human capital: A case study in high-technology industry. Expert Systems with applications, Volume 34, Issue 1, Pages 280–290.

Cho, V., & Ngai, E.W.T. (2003, July). Data mining for selection of insurance sales agents. Expert systems, Volume 20, Issue 3, Pages 123–132.

Cortes, C., Vapnik, V. (1995). Support-vector networks. Machine Learning, Volume 20, Issue 3, Pages 273–297.

Fan, C.Y., Fan, P.S., Chan, T.Y., & Chang, S.H. (2012, August). Using hybrid data mining and machine learning clustering analysis to predict the turnover rate for technology professionals. Expert Systems with Applications, Volume 39, Issue 10, Pages 8844–8851.

Frost, S. Problems of High Turnover Rates. Retrieved from: http://smallbusiness.chron.com/problems-high-turnover-rates-11659.html

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases, AI magazine. Retrieved from: http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf

Fürnkranz,J. (n.d.). Decision-Tree Learning. Retrieved from: http://www.ke.tu-darmstadt.de/lehre/archiv/ws0809/mldm/dt.pdf

Fox, John. & Andersen, Robert. (2005). Using the R Statistical Computing Environment to Teach Social Statistics Courses. Department of Sociology, McMaster University.

JCGM 200. (2008). Basic and general concepts and associated terms (VIM). International vocabulary of metrology.

Hodge, V.J., & Austin, J. (2004, October). A Survey of Outlier Detection Methodologies. Artificial Intelligence Review, Volume 22, Issue 2, Pages 85–126. Retrieved from: http://eprints.whiterose.ac.uk/767/1/hodgevj4.pdf

Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Morgan Kaufmann, 3 rd ed., Page 5.

IBM. (2011). IBM Cognos 8 Mainly this document for Documentation, Page 10.

McCulloch, Warren. S., Pitts, Walter. (1943). A Logical Calculus of Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, Volume 5, Issue 4, Pages 115–133.

MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press, Pages 281–297.

Mansour, Y. (1997). Pessimistic decision tree pruning based on tree size. 14th International Conference on Machine Learning: 195–201.

Piatetsky-Shapiro, Gregory. (1991). Discovery, analysis, and presentation of strong rules. Knowledge Discovery in Databases, AAAI/MIT Press, Cambridge, MA Pages 229-238.

Ranjan, J., Goyal, D.P., & Ahson, SI. (2008). Data mining techniques for better decisions in human resource management systems. International Journal of Business Information Systems (IJBIS), Volume 3, Issue 5.

Russell, Stuart. J., & Norvig, Peter. (1995). Artificial Intelligence: A Modern Approach (2nd ed.). Prentice Hall.

Sexton, R.S., McMurtrey, S., Michalopoulos, J.O., & Smith, A.M. (2005, October). Employee turnover: a neural network solution. Computers & Operations Research, Volume 32, Issue 10, Pages 2635–2651.

Strohmeier, S., &Piazza, F. (2013, June). Domain driven data mining in human resource management: A review of current research. Expert Systems with Applications,Volume 40, Issue 7, Pages 2410–2420.

Sikaroudi, E., Mohammad, A., & Ghousi, R. (2015, Autumn). A data mining approach to employee turnover prediction (case study: Arak automotive parts manufacturing). Journal of Industrial and Systems Engineering. Article 7, Volume 8, Issue 4, Pages 106-121.

Sayad. S. (2017). Decision tree-overfitting. Retrived from: http://www.saedsayad.com/decision_tree_overfitting.htm

Trip, R. (n.d.). Turnover-State of Oklahoma Website. Retrieved from www.ok.gov: http://www.ok.gov/opm/documents/Employee%20Turnover%20Presentation.ppt

Tamizharasi, K., UmaRani, Dr. (2014). Employee Turnover Analysis with Application of Data Mining Methods. International Journal of Computer Science and Information Technologies, Volume. 5, No. 1, Pages 562-566.

Valle, M.A., Varas, S., & Ruz, G.A. (2012, September). Job performance prediction in a call center using a naive Bayes classifier. Expert Systems with Applications, Volume 39, Issue 11, Pages 9939–9945.

What's the Real Cost of Turnover? Source: Finders & Keepers, published by Alberta Human Resources and Employment. Retrieved from: https://www.go2hr.ca/articles/what%e2%80%99s-real-cost-turnover

Witten, I. H., Frank, E., Hall, M. A., Pal, C. J. (2011). Data Mining: Practical machine learning tools and techniques(3rd ed.). Morgan Kaufmann, San Francisco.

Yockey, R. D. (2010). SPSS demystified: A step by step approach. Prentice Hall, Pages 10-20.

Appendix

```
data<-read.csv("employee.csv")

summary(data)

str(data)

data$project<-as.numeric(data$project)

data$hour<-as.numeric(data$hour)

data$time<-as.numeric(data$time)

data$accident<-as.numeric(data$accident)

data$promotion<-as.numeric(data$promotion)

data$left<-as.numeric(data$left)

str(data)

c<-data.frame(data)

correlation<-c[,-c(8:9)]

m<-cor(correlation)

library("corrplot")

corrplot(m)


library("ggplot2")

ggplot(data,aes(x=salary,y=satisfy,fill=factor(left),colour=factor(le
ft)))+geom_boxplot(outlier.colour="black")+xlab("salary")+ylab("satis
fy")

ggplot(data,aes(x=salary,y=time,fill=factor(left),colour=factor(left)
))+geom_boxplot(outlier.colour="black")+xlab("salary")+ylab("time")

ggplot(data,aes(x=salary,y=hour,fill=factor(left),colour=factor(left)
))+geom_boxplot(outlier.colour="black")+xlab("salary")+ylab("hour")

hist<-subset(c,left==1)

hist(hist$satisfy,main="satisfaction level")

hist(hist$evaluate,main="last evaluation")

plot(hist$salary,main="salary")
```

```
library("rpart")

library("rpart.plot")

model<-rpart(left~.,data=data)

model

rpart.plot(model, tweak=1.2)

printcp(model)


nrow(hist)

good_employee_leaving<-subset(hist,evaluate>=0.9|time>=8|project>=6)

nrow(good_employee_leaving)

good_employee<-subset(data,evaluate>=0.9|time>=8|project>=6)

model1<--rpart(left~.,data=good_employee)

model1

rpart.plot(model1, tweak=1.2)


library("lattice")

library("ggplot2")

library("caret")

train_control<-trainControl(method="cv",number=10)

data$left<-as.factor(data$left)

rpartmodel<- train(left~., data=data, trControl=train_control,

method="rpart")

print(rpartmodel)

predictions<- predict(rpartmodel,data)

tree<-cbind(data,predictions)

confusionMatrix(tree$predictions,data$left)


library("gplots")

library("ROCR")

tree$predictions<-as.numeric(paste(tree$predictions))
```

```
perf.obj<-prediction(predictions=tree$predictions,labels=tree$left)
roc.obj<-performance(perf.obj,measure="tpr",x.measure="fpr")
auc<-slot(performance(perf.obj,"auc"),"y.values")[[1]]
plot(roc.obj)
abline(h=seq(0,1,0.05), v=seq(0,1,0.05), col = "lightgray", lty =
"dotted")
lines(c(0,1),c(0,1), col = "gray", lwd =2)
text(0.6,0.2,paste("AUC=", round(auc,4), sep=""), cex=1.4)
title("ROC Curve")

n<-length(data$left)
index1<-1:n
index2<-rep(1:10,ceiling(n/10))[1:n]
index2<-sample(index2,n)
NMSE<-rep(0,10)
NMSE0<-NMSE
for(i in 1:10){
x<-index1[index2==i]
reg<-rpart(left~.,data[-x,])
y0<-predict(reg,data[-x,])
y1<-predict(reg,data[x,])
NMSE0[i]<-mean((data$left[-x]-y0)^2)/mean((data$left[-x]-mean(data$le
ft[-x]))^2)
NMSE[i]<-mean((data$left[x]-y1)^2)/mean((data$left[x]-mean(data$left[
x]))^2)}
NMSE0
NMSE

model2<-prune(model,cp=0.01)
rpart.plot(model2)
```

```
model2

training<-createDataPartition(data$left,p= .75,list=FALSE)
train<-data[training,]
test<-data[-training,]
model3<-rpart(left~.,data=train)
probaToLeave=predict(model3,newdata=test)
summary=data.frame(probaToLeave)
summary$performance=test$evaluate
summary$year=test$time
summary<-summary[,-1]
colnames(summary)[1]<-'probability'
head(summary)
library("DT")
summary$priority=summary$probability*summary$performance*summary$year
order=summary[order(summary$priority,decreasing = TRUE),]
order<-head(order,n=500)
datatable(order)
```