What is Classification in Data Mining?

Classification in data mining is a common technique that separates data points into different classes. It allows you to organize data sets of all sorts, including complex and large datasets as well as small and simple ones.

It primarily involves using algorithms that you can easily modify to improve the data quality.The primary goal of classification is to connect a variable of interest with the required variables. The variable of interest should be of qualitative type.

The algorithm establishes the link between the variables for prediction. The algorithm you use for classification in data mining is called the classifier, and observations you make through the same are called the instances. You use classification techniques in data mining when you have to work with qualitative variables.

There are multiple types of classification algorithms, each with its unique functionality and application. All of those algorithms are used to extract data from a dataset. Which application you use for a particular task depends on the goal of the task and the kind of data you need to extract.

Types of Classification Techniques in Data Mining

Before we discuss the various classification algorithms in data mining, let’s first look at the type of classification techniques available. Primarily, we can divide the classification algorithms into two categories:

1. Generative
2. Discriminative

Here’s a brief explanation of these two categories:

Generative

A generative classification algorithm models the distribution of individual classes. It tries to learn the model which creates the data through estimation of distributions and assumptions of the model. You can use generative algorithms to predict unseen data.

A prominent generative algorithm is the Naive Bayes Classifier.

Discriminative

It’s a rudimentary classification algorithm that determines a class for a row of data. It models by using the observed data and depends on the data quality instead of its distributions.

Logistic regression is an excellent type of discriminative classifiers.

Steps for Building a Classifier

The building of a classification model consists of two steps.

1. **Learning Step (Training Phase):**
   The learning step is where you train the algorithms to produce the desired results. The input parameters are available during the learning phase. We have the model's settings and the input data. We are teaching the algorithm something new. The algorithm changes the training parameters during training. It also alters the input data before producing an output.
2. **Classification Step (Testing Phase)**
   In this step, the model predicts class labels and evaluates the built model on test data to estimate the classification rules' accuracy.

Classifiers in Machine Learning

Classification is a highly popular aspect of data mining. As a result, machine learning has many classifiers:

1. Logistic regression
2. Linear regression
3. Decision trees
4. Random forest
5. Naive Bayes
6. Support Vector Machines

7. K-nearest neighbours

1. Logistic Regression

Logistic regression allows you to model the probability of a particular event or class. It uses a logistic to model a binary dependent variable. It gives you the probabilities of a single trial. Because logistic regression was built for classification and helps you understand the impact of multiple independent variables on a single outcome variable.

The issue with logistic regression is that it only works when your predicted variable is binary, and all the predictors are independent. Also, it assumes that the data doesn't have any missing values, which can be quite an issue.

2. Linear Regression

Linear regression is based on supervised learning and performs regression. It models a prediction value according to independent variables. Primarily, we use it to find out the relationship between the forecasting and the variables.

It predicts a dependent variable value according to a specific independent variable. Particularly, it finds the linear relationship between the independent variable and the dependent variable. It's excellent for data you can separate linear and is highly efficient. However, it is prone to overfitting and nose. Moreover, it relies on the assumption that the independent and dependent variables are related linearly.

3. Decision Trees

The decision tree is the most robust classification technique in data mining. It is a flowchart similar to a tree structure. Here, every internal node refers to a test on a condition, and each branch stands for an outcome of the test (whether it's true or false). Every leaf node in a decision tree holds a class label.

You can split the data into different classes according to the decision tree. It would predict which classes a new data point would belong to according to the created decision tree. Its prediction boundaries are vertical and horizontal lines.

4. Random forest

The random forest classifier fits multiple decision trees on different dataset sub-samples. It uses the average to enhance its predictive accuracy and manage overfitting. The sub-sample size is always equal to the input sample size; however, the samples are drawn with replacement.

A peculiar advantage of the random forest classifier is it reduces overfitting. Moreover, this classifier has significantly more accuracy than decision trees. However, it is a lot slower algorithm for real-time prediction and is a highly complicated algorithm, hence, very challenging to implement effectively.

5. Naive Bayes

The Naive Bayes algorithm assumes that every feature is independent of each other and that all the features contribute equally to the outcome.

Another assumption this algorithm relies upon is that all features have equal importance. It has many applications in today’s world, such as spam filtering and classifying documents. Naive Bayes only requires a small quantity of training data for the estimation of the required parameters. Moreover, a Naive Bayes classifier is significantly faster than other sophisticated and advanced classifiers.

However, the Naive Bayes classifier is notorious for being poor at estimation because it assumes all features are of equal importance, which is not true in most real-world scenarios.

6. Support Vector Machine

The Support vector machine algorithm, also known as SVM, represents the training data in space differentiated into categories by large gaps. New data points are then mapped into the same space, and their categories are predicted according to the side of the gap they fall into. This algorithm is especially useful in high dimensional spaces and is quite memory efficient because it only employs a subset of training points in its decision function.

This algorithm lags in providing probability estimations. You'd need to calculate them through five-fold cross-validation, which is highly expensive.

7. K-Nearest Neighbours

The k-nearest neighbor algorithm has non-linear prediction boundaries as it's a non-linear classifier. It predicts the class of a new test data point by finding its k nearest neighbours' class. You'd select the k nearest neighbours of a test data point by using the Euclidean distance. In the k nearest neighbours, you'd have to count the number of data points present in different categories, and you'd assign the new data point to the category with the most neighbors.

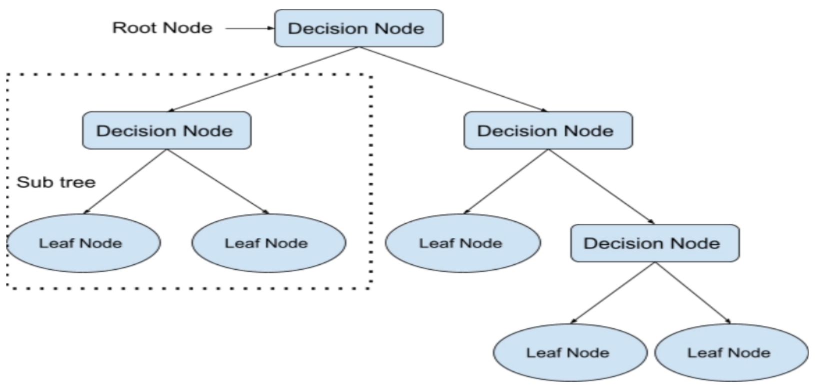
It's quite an expensive algorithm as finding the value of k takes a lot of resources. Moreover, it also has to calculate the distance of every instance to every training sample, which further enhances its computing cost.

Decision Trees

A decision tree is used to solve both classification and regression problems. But it is most commonly employed to solve classification issues.

- Internal nodes represent dataset attributes, branches represent decision rules, and each leaf node provides the conclusion in this tree-structured classifier.
- A decision tree is based on the idea that it asks a question and divides the tree into subtrees based on the answer (Yes/No).

  A decision tree's general structure is shown in the diagram below:



How Does A Decision Tree Work?

A decision tree is a supervised learning algorithm that works for both discrete and continuous variables. It splits the dataset into subsets on the basis of the most significant attribute in the dataset. How the decision tree identifies this attribute and how this splitting is done is decided by the algorithms.

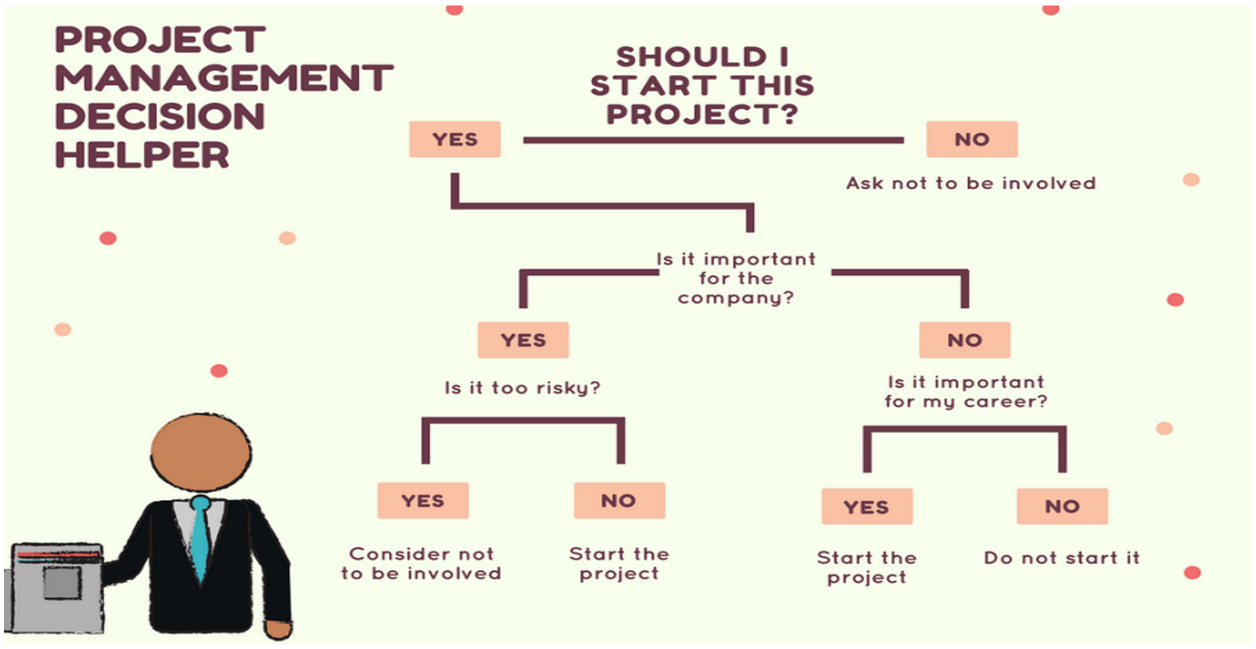
The most significant predictor is designated as the root node, splitting is done to form sub-nodes called decision nodes, and the nodes which do not split further are terminal or leaf nodes.

In the decision tree, the dataset is divided into homogeneous and non-overlapping regions. It follows a top-down approach as the top region presents all the observations at a single place which splits into two or more branches that further split. This approach is also called a *greedy approach* as it only considers the current node between the worked on without focusing on the future nodes.
The decision tree algorithms will continue running until a stop criteria such as the minimum number of observations etc. is reached.

Once a decision tree is built, many nodes may represent outliers or noisy data. Tree pruning method is applied to remove unwanted data. This, in turn, improves the accuracy of the classification model.

To find the accuracy of the model, a test set consisting of test tuples and class labels is used. The percentages of the test set tuples are correctly classified by the model to identify the accuracy of the model. If the model is found to be accurate then it is used to classify the data tuples for which the class labels are not known.

Some of the decision tree algorithms include Hunt’s Algorithm, ID3, CD4.5, and CART.


PROJECT MANAGEMENT DECISION HELPER
SHOULD I START THIS PROJECT?
YES
NO
Ask not to be involved
Is it important for the company?
YES
NO
Is it too risky?
Is it important for my career?
YES
NO
YES
NO
Consider not to be involved
Start the project
Start the project
Do not start it

List of Applications

Decision trees are mostly used by information experts to carry on an analytical investigation. They might be used extensively for business purposes to analyze or predict difficulties. The flexibility of the decision tree allows them to be used in a different area:

1. Healthcare

Decision trees allow the prediction of whether a patient is suffering from a particular disease with conditions of age, weight, sex, etc. Other predictions include deciding the effect of medicine considering factors like composition, period of manufacture, etc.

2. Banking sectors

Decision trees help in predicting whether a person is eligible for a loan considering his financial status, salary, family members, etc. It can also identify credit card frauds, loan defaults, etc.

3. Educational Sectors

Shortlisting of a student based on his merit score, attendance, etc. can be decided with the help of decision trees.

Types of Decision Trees

Hunt’s algorithm, which was developed in the 1960s to model human learning in Psychology, forms the foundation of many popular decision tree algorithms, such as the following:

- ID3: Ross Quinlan is credited within the development of ID3, which is shorthand for “Iterative Dichotomiser 3.” This algorithm leverages entropy and information gain as metrics to evaluate candidate splits.
- C4.5: This algorithm is considered a later iteration of ID3, which was also developed by Quinlan. It can use information gain or gain ratios to evaluate split points within the decision trees.
- CART: The term, CART, is an abbreviation for “classification and regression trees” and was introduced by Leo Breiman. This algorithm typically utilizes Gini impurity to identify the ideal attribute to split on. Gini impurity measures how often a randomly chosen attribute is misclassified. When evaluating using Gini impurity, a lower value is more ideal.

How To Select Attributes For Creating A Tree?

Attribute selection measures are also called splitting rules to decide how the tuples are going to split. The splitting criteria are used to best partition the dataset. These measures provide a ranking to the attributes for partitioning the training tuples.

The most popular methods of selecting the attribute are information gain, Gini index.

#1) Information Gain

This method is the main method that is used to build decision trees. It reduces the information that is required to classify the tuples. It reduces the number of tests that are needed to classify the given tuple. The attribute with the highest information gain is selected.

The original information needed for classification of a tuple in dataset D is given by:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Where p is the probability that the tuple belongs to class C. The information is encoded in bits, therefore, log to the base 2 is used. E(s) represents the average amount of

information required to find out the class label of dataset D. This information gain is also called **Entropy**.
The information required for exact classification after portioning is given by the formula:

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

Where P (c) is the weight of partition. This information represents the information needed to classify the dataset D on portioning by X.

Information gain is the difference between the original and expected information that is required to classify the tuples of dataset D.

$$Gain(T,X) = Entropy(T) - Entropy(T,X)$$

Gain is the reduction of information that is required by knowing the value of X. The attribute with the highest information gain is chosen as “best”.

#2) Gain Ratio

Information gain might sometimes result in portioning useless for classification. However, the Gain ratio splits the training data set into partitions and considers the number of tuples of the outcome with respect to the total tuples. The attribute with the max gain ratio is used as a splitting attribute.

$$\text{Gain Ratio (A)} = \frac{\text{Gain (A)}}{\text{SplitInfo (D)}}$$

Entropy and Information Gain

It’s difficult to explain information gain without first discussing entropy. Entropy is a concept that stems from information theory, which measures the impurity of the sample values. It is defined with by the following formula, where:

$$\text{Entropy}(S) = -\sum_{c \in C} p(c)\log_2 p(c)$$

- S represents the data set that entropy is calculated
- c represents the classes in set, S
- p(c) represents the proportion of data points that belong to class c to the number of total data points in set, S

Entropy values can fall between 0 and 1. If all samples in data set, S, belong to one class, then entropy will equal zero. If half of the samples are classified as one class and the other half are in another class, entropy will be at its highest at 1. In order to select the best feature to split on and find the optimal decision tree, the attribute with the smallest amount of entropy should be used. Information gain represents the difference in entropy before and after a split on a given attribute. The attribute with the highest information gain will produce the best split as it's doing the best job at classifying the training data according to its target classification. Information gain is usually represented with the following formula, where:

- *a* represents a specific attribute or class label
- *Entropy(S)* is the entropy of dataset, S
- |Sv|/ |S| represents the proportion of the values in S_v to the number of values in dataset, S
- *Entropy*(S_v) is the entropy of dataset, S_v

Let's walk through an example to solidify these concepts. Imagine that we have the following arbitrary dataset:

Example of Decision Tree Algorithm

Constructing a Decision Tree

Let us take an example of the last 10 days weather dataset with attributes outlook, temperature, wind, and humidity. The outcome variable will be playing cricket or not. We will use the ID3 algorithm to build the decision tree.

Day	Outlook	Temperature	Humidity	Wind	Play cricket
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes

Day	Outlook	Temperature	Humidity	Wind	Play cricket
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Step1: The first step will be to create a root node.
Step2: If all results are yes, then the leaf node "yes" will be returned else the leaf node "no" will be returned.
Step3: Find out the Entropy of all observations and entropy with attribute "x" that is E(S) and E(S, x).
Step4: Find out the information gain and select the attribute with high information gain.
Step5: Repeat the above steps until all attributes are covered.

Calculation of Entropy:

Yes	No
9	5

$$Entropy(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

$$Entropy(S) = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right)$$

$$= 0.940$$

If entropy is zero, it means that all members belong to the same class and if entropy is one then it means that half of the tuples belong to one class and one of them belong to other class. 0.94 means fair distribution.

Find the information gain attribute which gives maximum information gain.

For Example “Wind”, it takes two values: Strong and Weak, therefore, x = {Strong, Weak}.

$$IG(S, Wind) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

Find out H(x), P(x) for x =weak and x= strong. H(S) is already calculated above.

Weak= 8

Strong= 8

$$P(S_{weak}) = \frac{Number\ of\ Weak}{Total} = \frac{8}{14}$$

$$P(S_{strong}) = \frac{Number\ of\ Strong}{Total} = \frac{6}{14}$$

For “weak” wind, 6 of them say “Yes” to play cricket and 2 of them say “No”. So entropy will be:

$$Entropy(S_{weak}) = -\left(\frac{6}{8}\right)\log_2\left(\frac{6}{8}\right) - \left(\frac{2}{8}\right)\log_2\left(\frac{2}{8}\right) = 0.811$$

For “strong” wind, 3 said “No” to play cricket and 3 said “Yes”.

$$Entropy(S_{strong}) = -\left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right) - \left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right) = 1.000$$

This shows perfect randomness as half items belong to one class and the remaining half belong to others.

Calculate the information gain,

$$IG(S, Wind) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

$$IG(S, Wind) = H(S) - P(S_{weak}) * H(S_{weak}) - P(S_{strong}) * H(S_{strong})$$

$$= 0.940 - \left(\frac{8}{14}\right)(0.811) - \left(\frac{6}{14}\right)(1.00)$$

$$= 0.048$$

Similarly the information gain for other attributes is:

$$IG(S, Outlook) = 0.246$$

$$IG(S, Temperature) = 0.029$$

$$IG(S, Humidity) = 0.151$$

The attribute outlook has the **highest information gain** of 0.246, thus it is chosen as root.
Overcast has 3 values: Sunny, Overcast and Rain. Overcast with play cricket is always “Yes”. So it ends up with a leaf node, “yes”. For the other values “Sunny” and “Rain”.

Table for Outlook as “Sunny” will be:

Temperature	**Humidity**	**Wind**
Hot	High	Weak
Hot	High	Strong
Mild	High	Weak
Cool	Normal	Weak
Mild	Normal	Strong

Entropy for “Outlook” “Sunny” is:

$$H(S_{sunny}) = \left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) - \left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) = 0.96$$

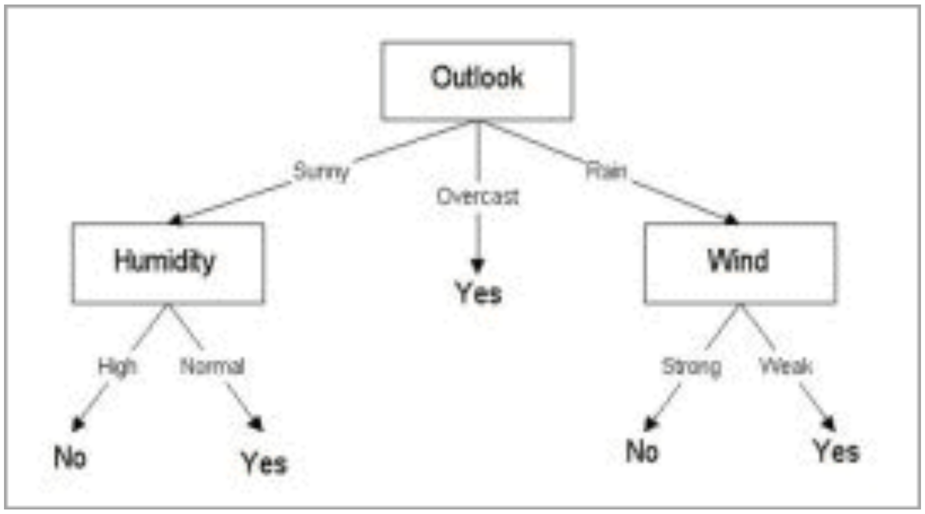
Information gain for attributes with respect to Sunny is:

$$IG(S_{sunny}, Humidity) = 0.96$$

$$IG(S_{sunny}, Temperature) = 0.57$$

$$IG(S_{sunny}, Wind) = 0.019$$

The information gain for humidity is highest, therefore it is chosen as the next node.
Similarly, Entropy is calculated for Rain. **Wind gives the highest information gain**.
The decision tree would look like below:



Generating a decision tree form training tuples of data partition D
Algorithm : Generate_decision_tree

Input:
Data partition, D, which is a set of training tuples
and their associated class labels.
attribute_list, the set of candidate attributes.
Attribute selection method, a procedure to determine the
splitting criterion that best partitions that the data
tuples into individual classes. This criterion includes a
splitting_attribute and either a splitting point or splitting subset.

Output:
A Decision Tree

Method

```
create a node N;

if tuples in D are all of the same class, C then
   return N as leaf node labeled with class C;

if attribute_list is empty then
   return N as leaf node with labeled
   with majority class in D;|| majority voting

apply attribute_selection_method(D, attribute_list)
to find the best splitting_criterion;
label node N with splitting_criterion;

if splitting_attribute is discrete-valued and
   multiway splits allowed then  // no restricted to binary trees
```

```
attribute_list = splitting attribute; // remove splitting attribute
for each outcome j of splitting criterion

  // partition the tuples and grow subtrees for each partition
  let Dj be the set of data tuples in D satisfying outcome j; // a partition

  if Dj is empty then
    attach a leaf labeled with the majority
    class in D to node N;
  else
    attach the node returned by Generate
    decision tree(Dj, attribute list) to node N;
  end for
return N;
```

CART Algorithm

In the decision tree, the nodes are split into subnodes on the basis of a threshold value of an attribute. The CART algorithm does that by searching for the best homogeneity for the subnodes, with the help of the Gini Index criterion.

The root node is taken as the training set and is split into two by considering the best attribute and threshold value. Further, the subsets are also split using the same logic. This continues till the last pure sub-set is found in the tree or the maximum number of leaves possible in that growing tree. This is also known as Tree Pruning.

Gini Impurity or Gini Index

Gini impurity is the probability of incorrectly classifying random data point in the dataset if it were labeled based on the class distribution of the dataset. Similar to entropy, if set, S, is pure—i.e. belonging to one class) then, its impurity is zero. This is denoted by the following formula:

$$\text{Gini Impurity} = 1 - \sum_i (p_i)^2$$

Advantages

- Easy to interpret: The Boolean logic and visual representations of decision trees make them easier to understand and consume. The hierarchical nature of a decision tree also makes it easy to see which attributes are most important, which isn't always clear with other algorithms, like neural networks.
- Little to no data preparation required: Decision trees have a number of characteristics, which make it more flexible than other classifiers. It can handle various data types—i.e. discrete or continuous values, and continuous values can be converted into categorical values through the use of thresholds. Additionally, it can also handle values with missing values, which can be problematic for other classifiers, like Naïve Bayes.
- More flexible: Decision trees can be leveraged for both classification and regression tasks, making it more flexible than some other algorithms. It's also insensitive to underlying relationships between attributes; this means that if two variables are highly correlated, the algorithm will only choose one of the features to split on.
Disadvantages

- Prone to overfitting: Complex decision trees tend to overfit and do not generalize well to new data. This scenario can be avoided through the processes of pre-pruning or post-pruning. Pre-pruning halts tree growth when there is insufficient data while post-pruning removes subtrees with inadequate data after tree construction.
- High variance estimators: Small variations within data can produce a very different decision tree. Bagging, or the averaging of estimates, can be a method of reducing variance of decision trees. However, this approach is limited as it can lead to highly correlated predictors.
- More costly: Given that decision trees take a greedy search approach during construction, they can be more expensive to train compared to other algorithms.
- Not fully supported in scikit-learn: Scikit-learn is a popular machine learning library based in Python. While this library does have a Decision Tree module (DecisionTreeClassifier, link resides outside of ibm.com), the current implementation does not support categorical variables.

Data set

There are 14 instances of golf playing decisions based on outlook, temperature, humidity and wind factors.

Day	**Outlook**	**Temp.**	**Humidity**	**Wind**	**Decision**
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes

11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Gini index

Gini index is a metric for classification tasks in CART. It stores sum of squared probabilities of each class. We can formulate it as illustrated below.

$$\text{Gini} = 1 - \Sigma (Pi)^2 \text{ for i=1 to number of classes}$$

Outlook

Outlook is a nominal feature. It can be sunny, overcast or rain. I will summarize the final decisions for outlook feature.

Outlook	Yes	No	Number of instances
Sunny	2	3	5

Overcast	4	0	4
Rain	3	2	5

Gini(Outlook=Sunny) = 1 – $(2/5)^2$ – $(3/5)^2$ = 1 – 0.16 – 0.36 = 0.48

Gini(Outlook=Overcast) = 1 – $(4/4)^2$ – $(0/4)^2$ = 0

Gini(Outlook=Rain) = 1 – $(3/5)^2$ – $(2/5)^2$ = 1 – 0.36 – 0.16 = 0.48

Then, we will calculate weighted sum of gini indexes for outlook feature.

Gini(Outlook) = (5/14) x 0.48 + (4/14) x 0 + (5/14) x 0.48 = 0.171 + 0 + 0.171 = 0.342

Temperature

Similarly, temperature is a nominal feature and it could have 3 different values: Cool, Hot and Mild. Let's summarize decisions for temperature feature.

Temperature	Yes	No	Number of instances
Hot	2	2	4
Cool	3	1	4
Mild	4	2	6

Gini(Temp=Hot) = 1 – $(2/4)^2$ – $(2/4)^2$ = 0.5

Gini(Temp=Cool) = 1 – $(3/4)^2$ – $(1/4)^2$ = 1 – 0.5625 – 0.0625 = 0.375

Gini(Temp=Mild) = 1 – $(4/6)^2$ – $(2/6)^2$ = 1 – 0.444 – 0.111 = 0.445

We'll calculate weighted sum of gini index for temperature feature

Gini(Temp) = (4/14) x 0.5 + (4/14) x 0.375 + (6/14) x 0.445 = 0.142 + 0.107 + 0.190 = 0.439

Humidity

Humidity is a binary class feature. It can be high or normal.

Humidity	Yes	No	Number of instances
High	3	4	7
Normal	6	1	7

Gini(Humidity=High) = 1 – $(3/7)^2$ – $(4/7)^2$ = 1 – 0.183 – 0.326 = 0.489

Gini(Humidity=Normal) = 1 – $(6/7)^2$ – $(1/7)^2$ = 1 – 0.734 – 0.02 = 0.244

Weighted sum for humidity feature will be calculated next

Gini(Humidity) = (7/14) x 0.489 + (7/14) x 0.244 = 0.367

Wind

Wind is a binary class similar to humidity. It can be weak and strong.

Wind	Yes	No	Number of instances
Weak	6	2	8
Strong	3	3	6

Gini(Wind=Weak) = 1 – $(6/8)^2$ – $(2/8)^2$ = 1 – 0.5625 – 0.062 = 0.375

Gini(Wind=Strong) = 1 – $(3/6)^2$ – $(3/6)^2$ = 1 – 0.25 – 0.25 = 0.5

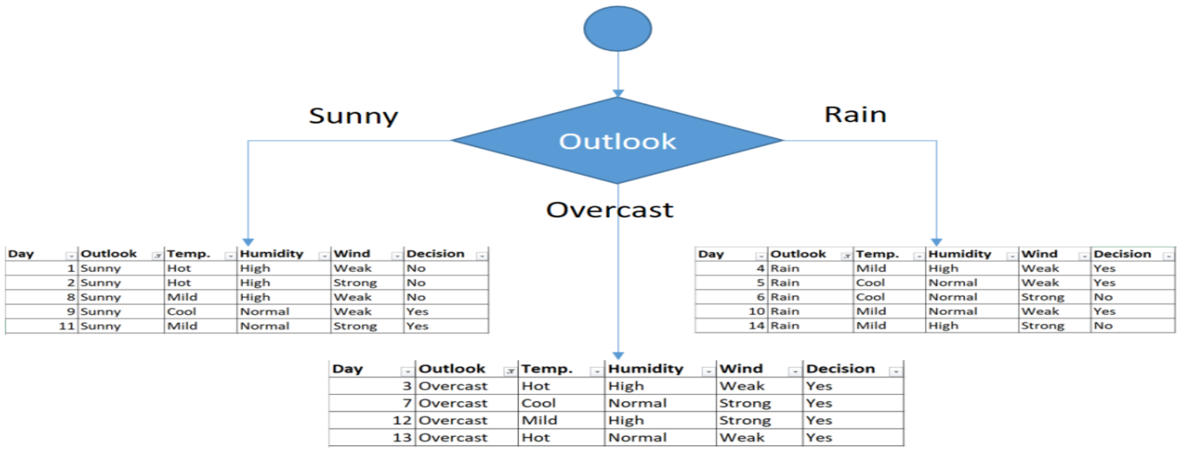
Gini(Wind) = (8/14) x 0.375 + (6/14) x 0.5 = 0.428

Time to decide

We’ve calculated gini index values for each feature. The winner will be outlook feature because its cost is the lowest.

Feature	Gini index
Outlook	0.342
Temperature	0.439
Humidity	0.367
Wind	0.428

We’ll put outlook decision at the top of the tree.



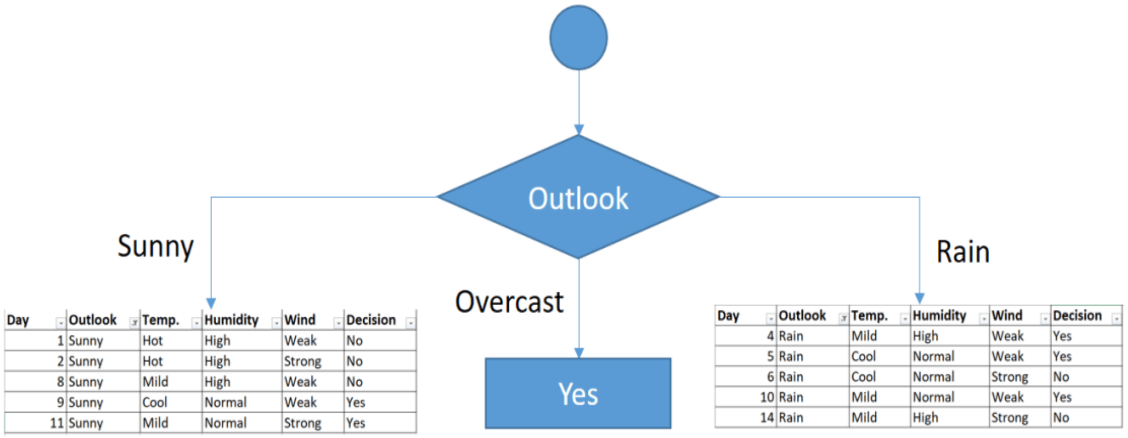
Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Day	Outlook	Temp.	Humidity	Wind	Decision
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

First decision would be outlook feature

You might realize that sub dataset in the overcast leaf has only yes decisions. This means that overcast leaf is over.



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Tree is over for overcast outlook leaf

We will apply same principles to those sub datasets in the following steps.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No

2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Focus on the sub dataset for sunny outlook. We need to find the gini index scores for temperature, humidity and wind features respectively.

Temperature	Yes	No	Number of instances
Hot	0	2	2
Cool	1	0	1
Mild	1	1	2

Gini of temperature for sunny outlook

Gini(Outlook=Sunny and Temp.=Hot) = $1 – (0/2)^2 – (2/2)^2 = 0$

Gini(Outlook=Sunny and Temp.=Cool) = $1 – (1/1)^2 – (0/1)^2 = 0$

Gini(Outlook=Sunny and Temp.=Mild) = $1 – (1/2)^2 – (1/2)^2 = 1 – 0.25 – 0.25 = 0.5$

Gini(Outlook=Sunny and Temp.) = (2/5)x0 + (1/5)x0 + (2/5)x0.5 = 0.2

Gini of humidity for sunny outlook

Humidity	Yes	No	Number of instances
High	0	3	3
Normal	2	0	2

Gini(Outlook=Sunny and Humidity=High) = 1 – $(0/3)^2$ – $(3/3)^2$ = 0

Gini(Outlook=Sunny and Humidity=Normal) = 1 – $(2/2)^2$ – $(0/2)^2$ = 0

Gini(Outlook=Sunny and Humidity) = (3/5)x0 + (2/5)x0 = 0

Gini of wind for sunny outlook

Wind	Yes	No	Number of instances
Weak	1	2	3
Strong	1	1	2

Gini(Outlook=Sunny and Wind=Weak) = 1 – $(1/3)^2$ – $(2/3)^2$ = 0.266

Gini(Outlook=Sunny and Wind=Strong) = 1- $(1/2)^2$ – $(1/2)^2$ = 0.2

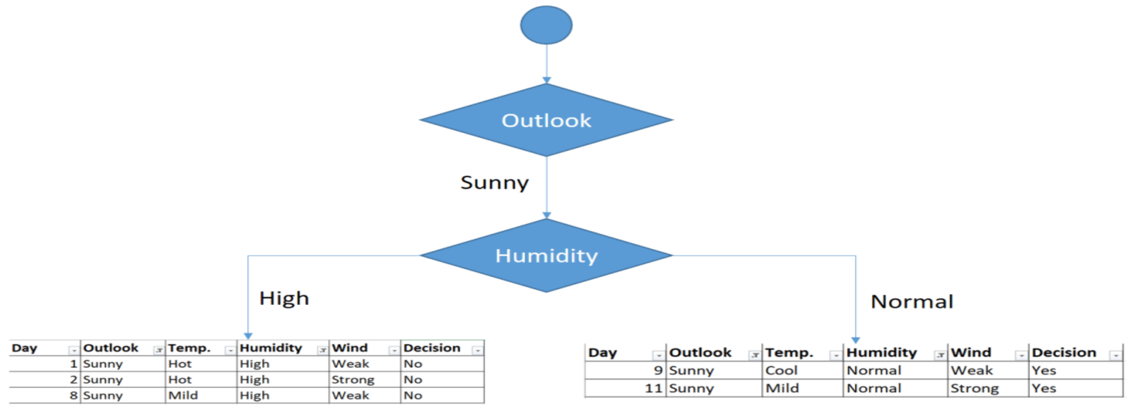
Gini(Outlook=Sunny and Wind) = (3/5)x0.266 + (2/5)x0.2 = 0.466

Decision for sunny outlook

We’ve calculated gini index scores for feature when outlook is sunny. The winner is humidity because it has the lowest value.

Feature	Gini index
Temperature	0.2
Humidity	0
Wind	0.466

We’ll put humidity check at the extension of sunny outlook.

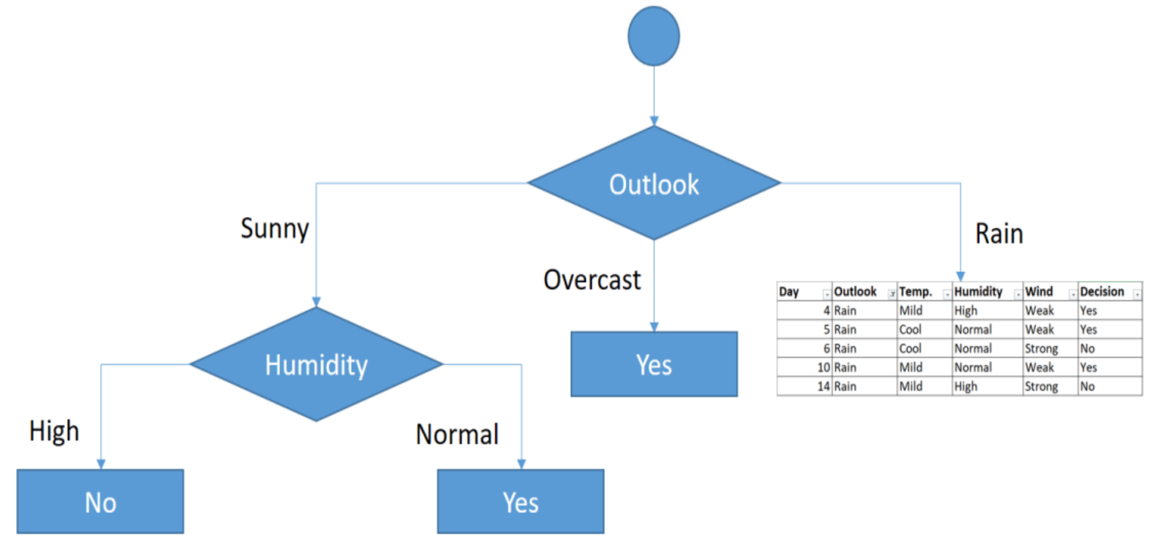


Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No

Day	Outlook	Temp.	Humidity	Wind	Decision
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Sub datasets for high and normal humidity

As seen, decision is always no for high humidity and sunny outlook. On the other hand, decision will always be yes for normal humidity and sunny outlook. This branch is over.



Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Decisions for high and normal humidity

Now, we need to focus on rain outlook.

Rain outlook

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

We'll calculate gini index scores for temperature, humidity and wind features when outlook is rain.

Gini of temprature for rain outlook

Temperature	Yes	No	Number of instances
Cool	1	1	2
Mild	2	1	3

Gini(Outlook=Rain and Temp.=Cool) = 1 – $(1/2)^2$ – $(1/2)^2$ = 0.5

Gini(Outlook=Rain and Temp.=Mild) = 1 – $(2/3)^2$ – $(1/3)^2$ = 0.444

Gini(Outlook=Rain and Temp.) = (2/5)x0.5 + (3/5)x0.444 = 0.466

Gini of humidity for rain outlook

Humidity	Yes	No	Number of instances
High	1	1	2
Normal	2	1	3

Gini(Outlook=Rain and Humidity=High) = 1 – $(1/2)^2$ – $(1/2)^2$ = 0.5

Gini(Outlook=Rain and Humidity=Normal) = 1 – $(2/3)^2$ – $(1/3)^2$ = 0.444

Gini(Outlook=Rain and Humidity) = (2/5)x0.5 + (3/5)x0.444 = 0.466

Gini of wind for rain outlook

Wind	Yes	No	Number of instances
Weak	3	0	3

Strong	0	2	2

Gini(Outlook=Rain and Wind=Weak) = 1 – $(3/3)^2$ – $(0/3)^2$ = 0

Gini(Outlook=Rain and Wind=Strong) = 1 – $(0/2)^2$ – $(2/2)^2$ = 0

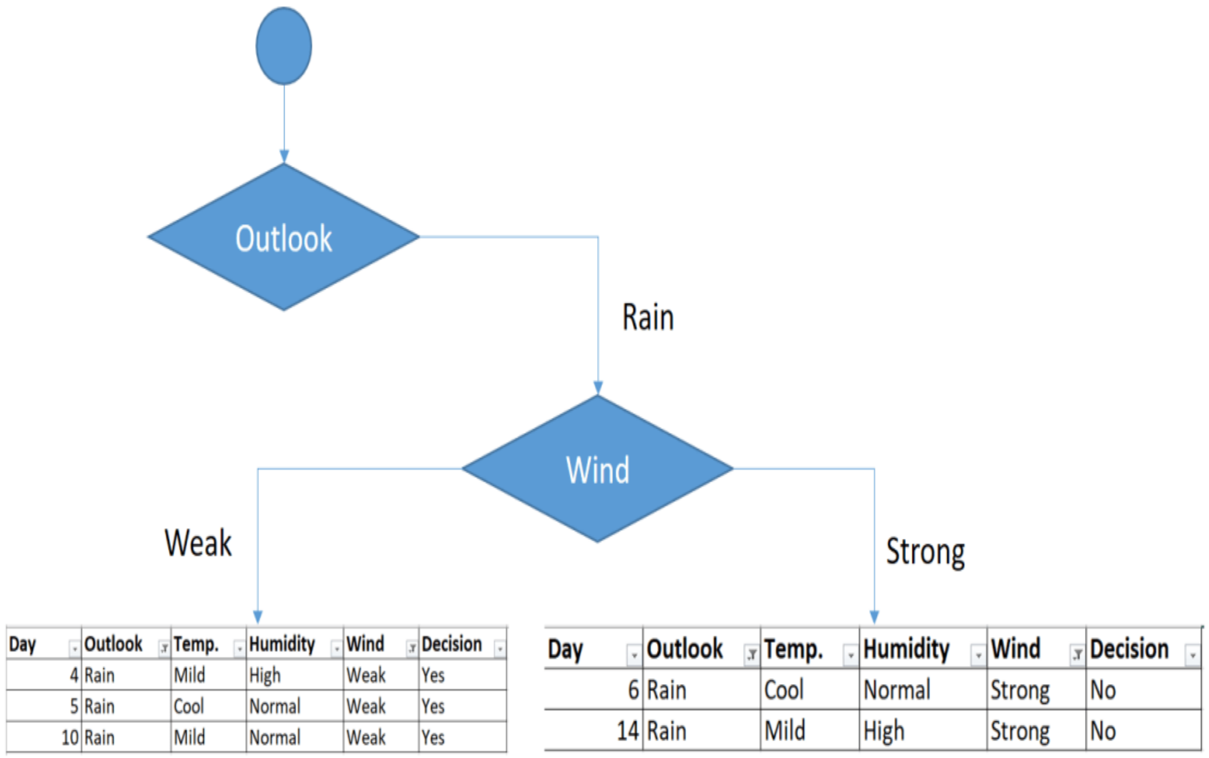
Gini(Outlook=Rain and Wind) = (3/5)x0 + (2/5)x0 = 0

Decision for rain outlook

The winner is wind feature for rain outlook because it has the minimum gini index score in features.

Feature	Gini index
Temperature	0.466
Humidity	0.466
Wind	0

Put the wind feature for rain outlook branch and monitor the new sub data sets.

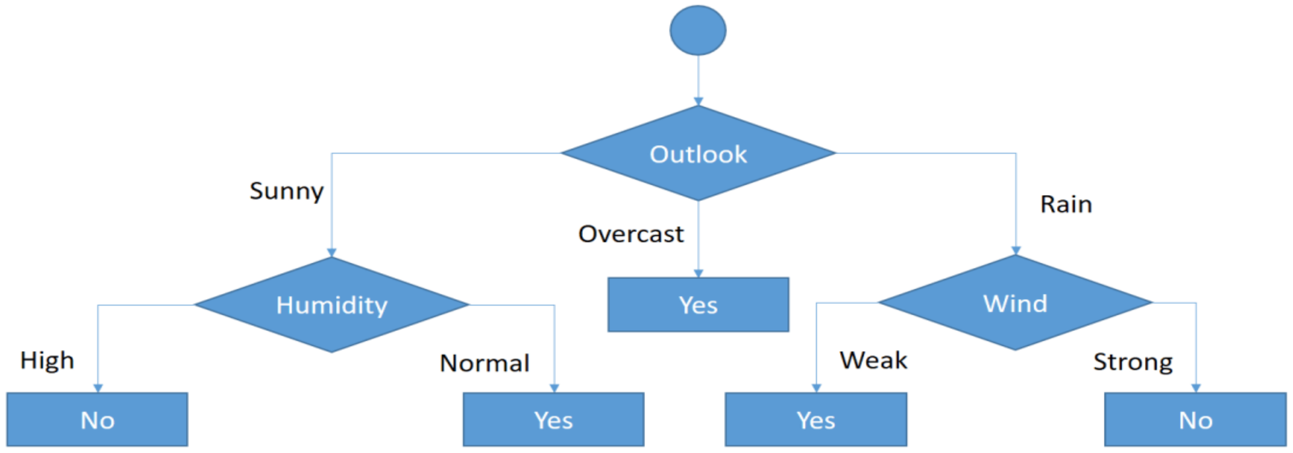


Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No

Sub data sets for weak and strong wind and rain outlook

As seen, decision is always yes when wind is weak. On the other hand, decision is always no if wind is strong. This means that this branch is over.



Final form of the decision tree built by CART algorithm

Advantages of CART algorithm

1. The CART algorithm is nonparametric, thus it does not depend on information from a certain sort of distribution.
2. The CART algorithm combines both testings with a test data set and cross-validation to more precisely measure the goodness of fit.
3. CART allows one to utilize the same variables many times in various regions of the tree. This skill is capable of revealing intricate interdependencies between groups of variables.
4. Outliers in the input variables have no meaningful effect on CART.
5. One can loosen halting restrictions to allow decision trees to overgrow and then trim the tree down to its ideal size. This method reduces the likelihood of missing essential structure in the data set by terminating too soon.
6. To choose the input set of variables, CART can be used in combination with other prediction algorithms.

Pseudo-code of the CART algorithm

```
d = 0, endtree = 0
Note(0) = 1, Node(1) = 0, Node(2) = 0
while endtree < 1
    if Node($2^d$-1) + Node($2^d$) + .... + Node($2^{d+1}$-2) = 2 - $2^{d+1}$
        endtree = 1
    else
        do i = $2^d$-1, $2^d$, .... , $2^{d+1}$-2
            if Node(i) > -1
                Split tree
            else
                Node(2i+1) = -1
                Node(2i+2) = -1
            end if
        end do
    end if
d = d + 1
```

```
        end  while
```

Overfitting In Decision Trees

Overfitting happens when a decision tree tries to be as perfect as possible by increasing the depth of tests and thereby reduces the error. This results in very complex trees and leads to overfitting.

Overfitting reduces the predictive nature of the decision tree. The approaches to avoid overfitting of the trees include pre pruning and post pruning.

What Is Tree Pruning?

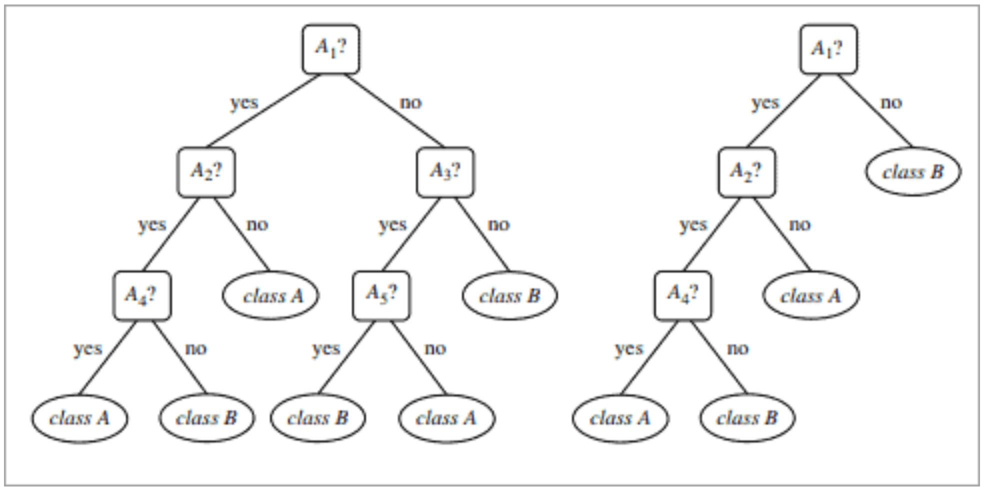
Pruning is the method of removing the unused branches from the decision tree. Some branches of the decision tree might represent outliers or noisy data.

Tree pruning is the method to reduce the unwanted branches of the tree. This will reduce the complexity of the tree and help in effective predictive analysis. It reduces the overfitting as it removes the unimportant branches from the trees.

There are two ways of pruning the tree:
#1) Prepruning: In this approach, the construction of the decision tree is stopped early. It means it is decided not to further partition the branches. The last node constructed becomes the leaf node and this leaf node may hold the most frequent class among the tuples.
The attribute selection measures are used to find out the weightage of the split. Threshold values are prescribed to decide which splits are regarded as useful. If the portioning of the node results in splitting by falling below threshold then the process is halted.

#2) Postpruning: This method removes the outlier branches from a fully grown tree. The unwanted branches are removed and replaced by a leaf node denoting the most frequent class label. This technique requires more computation than prepruning, however, it is more reliable.
The pruned trees are more precise and compact when compared to unpruned trees but they carry a disadvantage of replication and repetition.

Repetition occurs when the same attribute is tested again and again along a branch of a tree. *Replication* occurs when the duplicate subtrees are present within the tree. These issues can be solved by multivariate splits.
The Below image shows an unpruned and pruned tree.



Advantages Of Decision Tree Classification

Enlisted below are the various merits of Decision Tree Classification:

1. Decision tree classification does not require any domain knowledge, hence, it is appropriate for the knowledge discovery process.
2. The representation of data in the form of the tree is easily understood by humans and it is intuitive.
3. It can handle multidimensional data.
4. It is a quick process with great accuracy.

Disadvantages Of Decision Tree Classification

Given below are the various demerits of Decision Tree Classification:

1. Sometimes decision trees become very complex and these are called overfitted trees.
2. The decision tree algorithm may not be an optimal solution.
3. The decision trees may return a biased solution if some class label dominates it.

Conclusion

Decision Trees are data mining techniques for classification and regression analysis.

This technique is now spanning over many areas like medical diagnosis, target marketing, etc. These trees are constructed by following an algorithm such as ID3, CART. These algorithms find different ways to split the data into partitions.

It is the most widely known supervised learning technique that is used in machine learning and pattern analysis. The decision trees predict the values of the target variable by building models through learning from the training set provided to the system.