Machine Learning - Overview

DR DV Ramana
Academic Advisor
And Program Manager

Agenda - Theory

- Hierarchical Clustering
- Clustering Case Study
- Principal Component analysis
- Decision Trees
- Classification and Regression Trees
- Concept of Model Ensembling

Agenda -LAB

1. Explore scikit learn Library
2. Download “mall_customers.csv” dataset from kaggle.
   - (a) Form n no. of clusters according to your observation
   - (b) Find best K value
   - (c) Get wss value for each cluster

Hierarchical Clustering

Data science is the field of study that combines

- Domain expertise
- Programming skills, and
- Knowledge of mathematics and statistics to extract meaningful insights from data

Data scientist is a professional responsible for

- Data Collecting
- Data Analyzing, and
- Interpreting extremely large amounts of data.

Big Data is larger, more complex data sets, especially from new data sources

Clustering Case Study

Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior

Machine Learning (ML) is a sub-field of artificial intelligence (AI) that focuses on building applications that can automatically and periodically learn and improve from experience or through gathered data and help in solving practical problems.

Clustering Case Study

Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior

Machine Learning (ML) is a sub-field of artificial intelligence (AI) that focuses on building applications that can automatically and periodically learn and improve from experience or through gathered data and help in solving practical problems.

Principal Component Analysis

Principal Component Analysis is a popular unsupervised learning technique for reducing the dimensionality of data

Principal Component Analysis increases interpretability yet, at the same time, it minimizes information loss

Principal Component Analysis(PCA) is a method that is used to reduce the dimensionality of large amounts of data

Principal Component Analysis helps to find the most significant features in a dataset and makes the data easy for plotting in 2D and 3D

Principal component analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set

Principal Component Analysis(PCA) transforms many variables into a smaller set without sacrificing the information contained in the original set, thus reducing the dimensionality of the data

Principal Component Analysis

PCA should be used mainly for variables which are strongly correlated

If the relationship is weak between variables, PCA does not work well to reduce data

if most of the correlation coefficients are smaller than 0.3, PCA will not help

Principal Component Analysis (PCA) is one of the most commonly used unsupervised machine learning algorithms across a variety of applications

- Exploratory data analysis
- Dimensionality Reduction
- Information compression
- Data de-noising

Principal Component Analysis

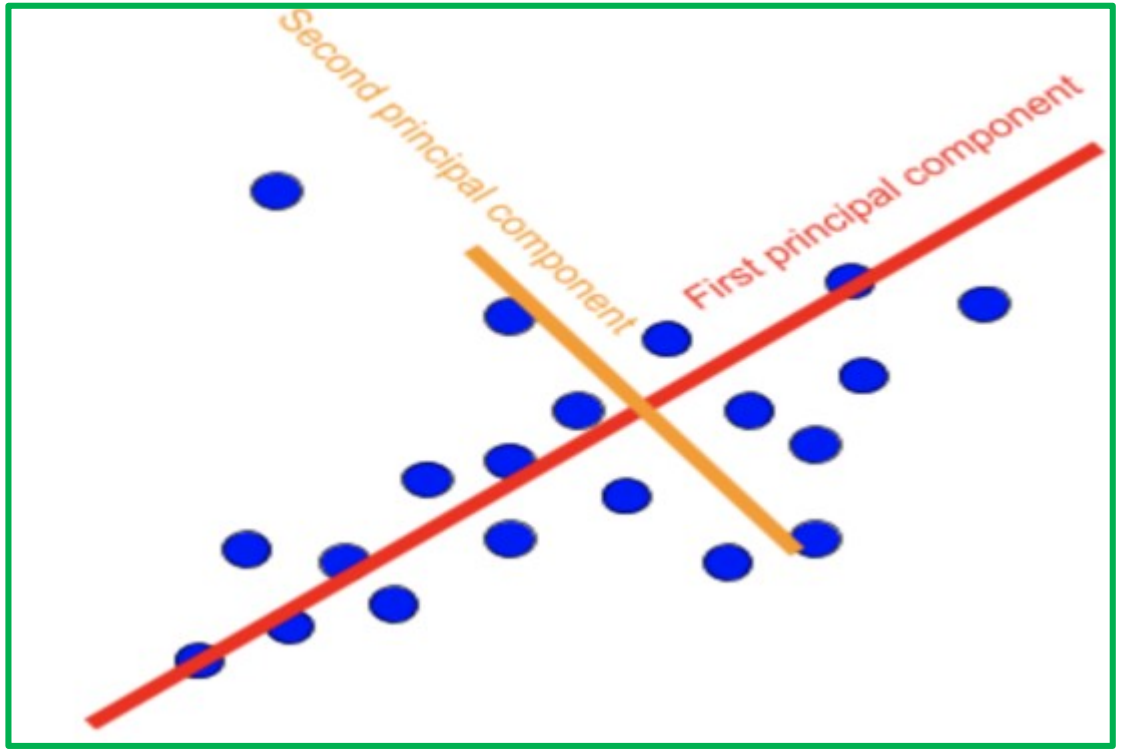
The main aim of PCA is to find such principal components, which can describe the data points with a set of well, principal components

Principal Component Analysis is basically a statistical procedure to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables

The principal components are vectors, but they are not chosen at random

First Principal Component is computed so that it explains the greatest amount of variance in the original features

Second Principal Component is orthogonal to the first, and it explains the greatest amount of variance left after the first principal component



Principal Component Analysis

The original data can be represented as feature vectors

PCA allows us to go a step further and represent the data as linear combinations of principal components

Getting principal components is equivalent to a linear transformation of data from the feature1 x feature2 axis to a PCA1 x PCA2 axis

Principal Component Analysis Advantages

Visualize multidimensional data

Data visualizations are a great tool for communicating multidimensional data as 2- or 3-dimensional plots

Compress information

Principal Component Analysis(PCA) is used to compress information to store and transmit data more efficiently

Example

Principal Component Analysis(PCA) can be used to compress images without losing too much quality, or in signal processing

Principal Component Analysis(PCA) is used to compress information to store and transmit data more efficiently

Principal Component Analysis Advantages

Simplify complex business decisions

PCA has been employed to simplify traditionally complex business decisions.

Example

Traders use over 300 financial instruments to manage portfolios

The algorithm has proven successful in the risk management of interest rate derivative portfolios, lowering the number of financial instruments from more than 300 to just 3-4 principal components

Principal Component Analysis Advantages

Clarify Convoluted Scientific Processes

PCA has been employed to simplify traditionally complex business decisions.

Example

The algorithm has been applied extensively in the understanding of convoluted and multidirectional factors, which increase the probability of neural ensembles to trigger action potentials

HOW DO YOU DO A PRINCIPAL COMPONENT ANALYSIS?

Step-1	Standardize the range of continuous initial variables
Step-2	Compute the covariance matrix to identify correlations
Step-3	Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components
Step-4	Create a feature vector to decide which principal components to keep
Step-5	Recast the data along the principal components axes

Decision Tree Learning -Introduction

A decision tree is a tree where each node represents feature(Attribute), each link(Branch) represents a decision(rule) and each leaf represents an outcome

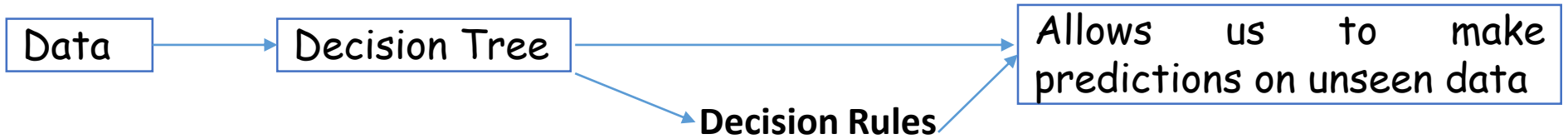
A decision tree is a simple representation for classifying examples

Decision tree learning is a method commonly used in data mining

The goal is to create a model that predicts the value of a target variable based on several input variables

A decision tree is a simple representation for classifying examples

A decision tree is constructed by looking for regularities in data.



Decision Tree Learning -Introduction

Decision tree learning is a method for approximating discrete- valued target functions, in which the learned function is represented by a decision tree.

Learned trees can also be re- represented as sets of if-then rules to improve human readability

Decision trees classify instances by sorting them done from the tree from the root to some leaf, which provides the classification of the instance.

Each node in the tree specifies a test of some attribute of the instance and each branch descending from the node corresponds to one of the possible values for this attribute.

An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute

Decision Tree Learning -Introduction

Decision tree learning is a method for approximating discrete- valued target functions, in which the learned function is represented by a decision tree

A decision tree is a tree where each node represents a feature(attribute), each link(branch) represents a decision (rule) and each leaf represents an outcome (categorical or continues value)

A decision tree or a classification tree is a tree in which each internal node is labeled with an input features. The arcs coming from a node labeled with a feature are labeled with each of the possible values of the feature.

Decision tree learning is a method for approximating discrete- valued target functions, in which the learned function is represented by a decision tree

A decision tree is a tree where each node represents a feature(attribute), each link(branch) represents a decision (rule) and each leaf represents an outcome (categorical or continues value)

A decision tree or a classification tree is a tree in which each internal node is labeled with an input features. The arcs coming from a node labeled with a feature are labeled with each of the possible values of the feature.

A decision tree has two kinds of nodes

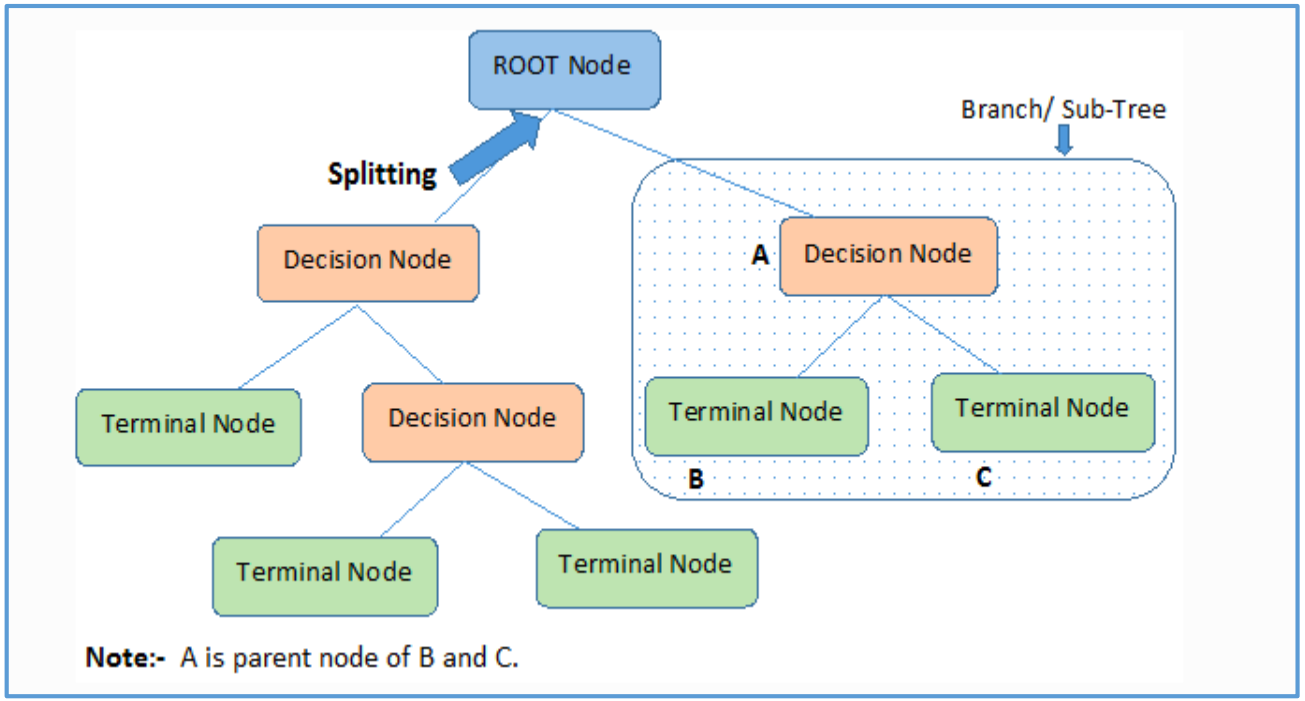
1. Each leaf node has a class label, determined by majority vote of training examples reaching that leaf

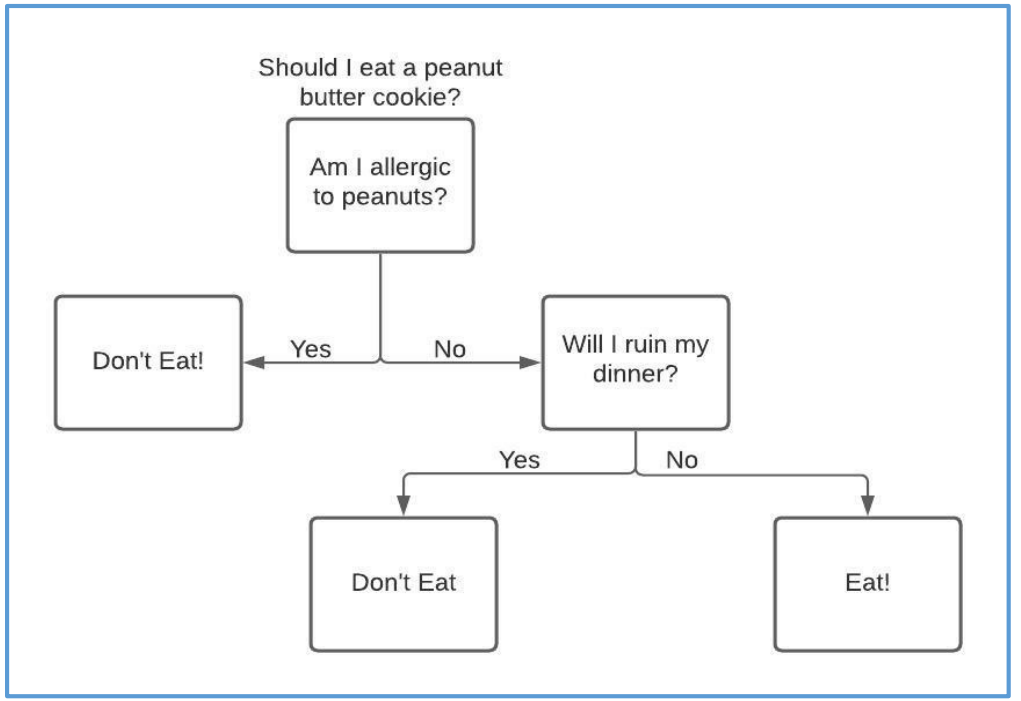
2. Each internal node is a question on features. It branch out according to the answers

Decision Tree Learning -Decision tree representation

Decision tree consists of three types of nodes:

- Decision nodes – typically represented by squares
- Chance nodes – typically represented by circles
- End nodes – typically represented by triangles

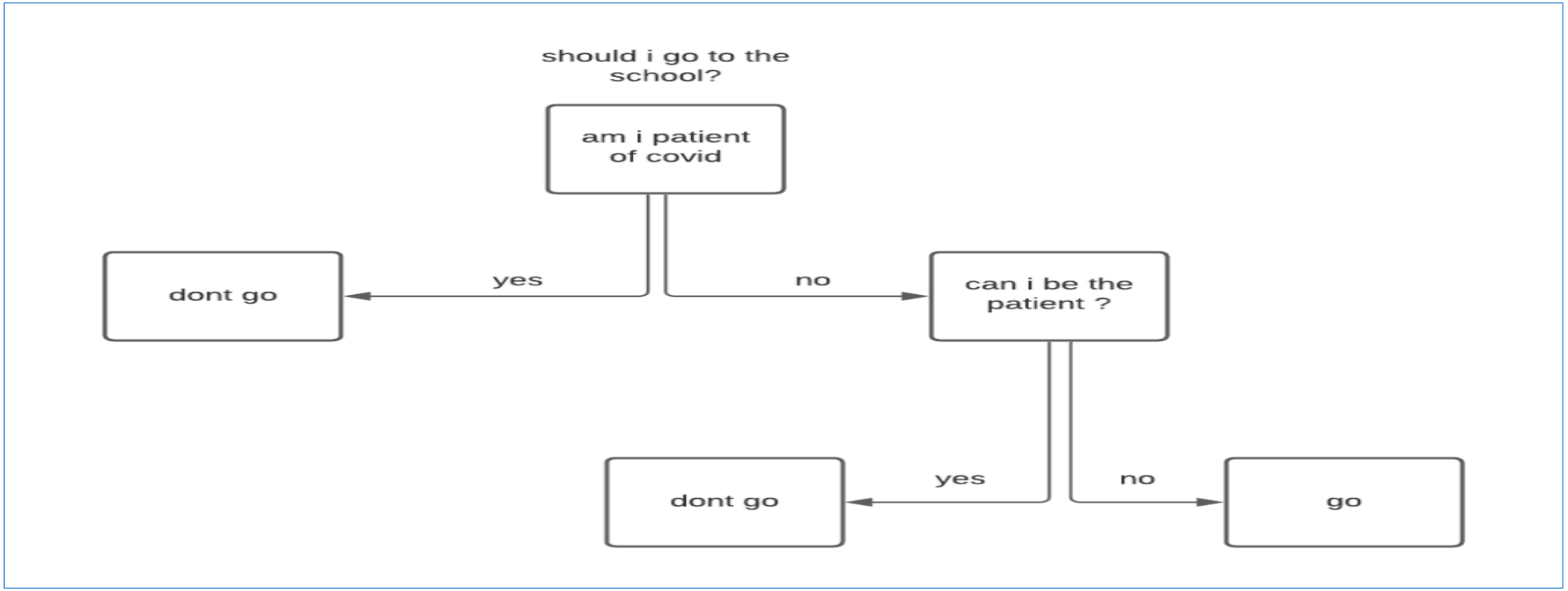




Decision Tree Learning -Decision tree representation

Decision tree classifies instances

Node	An attribute which describes an instance
Branch	Possible values of the attribute
Leaf Class	To which the instance belong



Decision Tree Learning -Decision tree representation

Decision tree representation

Each internal node tests an attribute

Each branch corresponds to attribute value

Each leaf node assigns a classification

Decision Tree Learning -Decision tree representation

Important Terminology related to Decision Trees

Root Node

Root Node represents the entire population or sample and this further gets divided into two or more homogeneous sets.

Splitting

Splitting is a process of dividing a node into two or more sub-nodes.

Decision Node

When a sub-node splits into further sub-nodes, then it is called the decision node.

Leaf / Terminal Node

Nodes do not split is called Leaf or Terminal node.

Decision Tree Learning -Decision tree representation

Important Terminology related to Decision Trees

Pruning

When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting

Branch / Sub-Tree

A subsection of the entire tree is called branch or sub-tree.

Parent and Child Node

When a sub-node splits into further sub-nodes, then it is called the decision node.

Leaf / Terminal Node

A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node

Decision Tree Learning -Appropriate problems for decision tree learning

Decision tree learning is generally best suited to problems with the following characteristics:

Instances are represented by attribute-value pairs

Instances are described by a fixed set of attributes (e.g., Temperature) and their values (e.g., Hot)

The easiest situation for decision tree learning is when each attribute takes on a small number of disjoint possible values (e.g., Hot, Mild, Cold)

Extensions to the basic algorithm allow handling real-valued attributes as well (e.g., representing Temperature numerically

The training data may contain errors

Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples

Decision Tree Learning -Decision tree representation

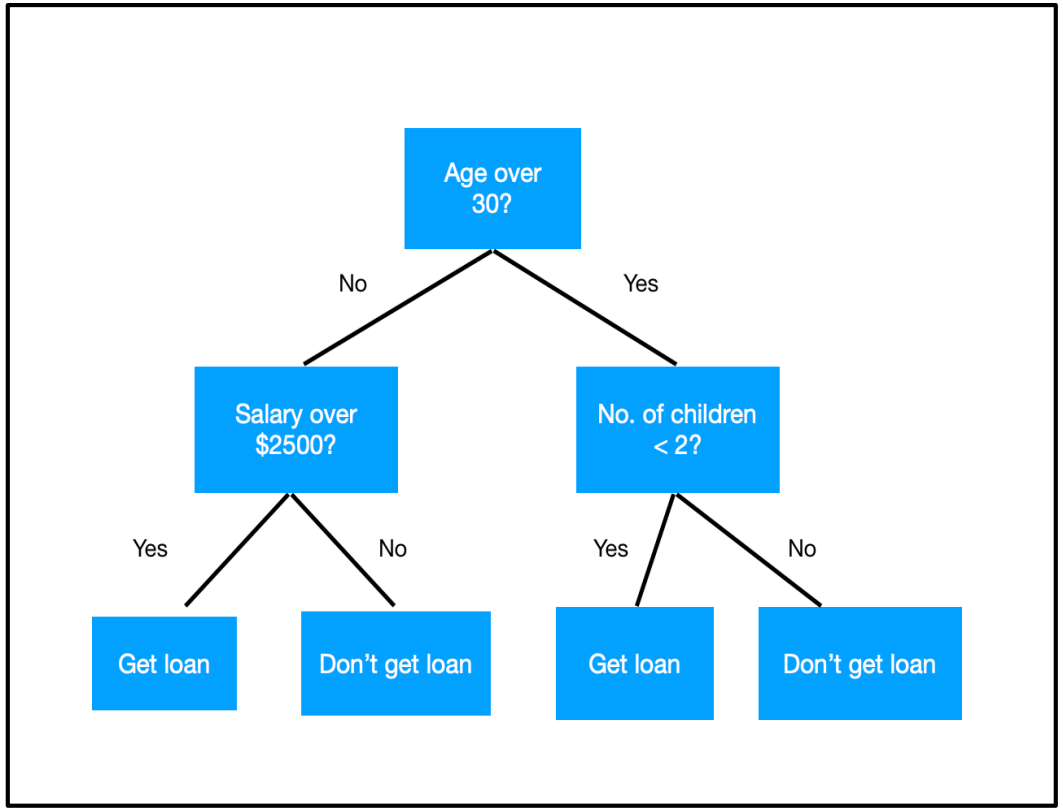
Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.

Figure represents a simple decision tree that is used to for a classification task of whether a customer gets a loan or not.

The input features are salary of the person, the number of children and the age of the person.

The decision tree uses these attributes or features and asks the right questions at the right step or node so as to classify whether the loan can be provided to the person or not.

The decision tree uses these attributes or features and asks the right questions at the right step or node so as to classify whether the loan can be provided to the person or not.



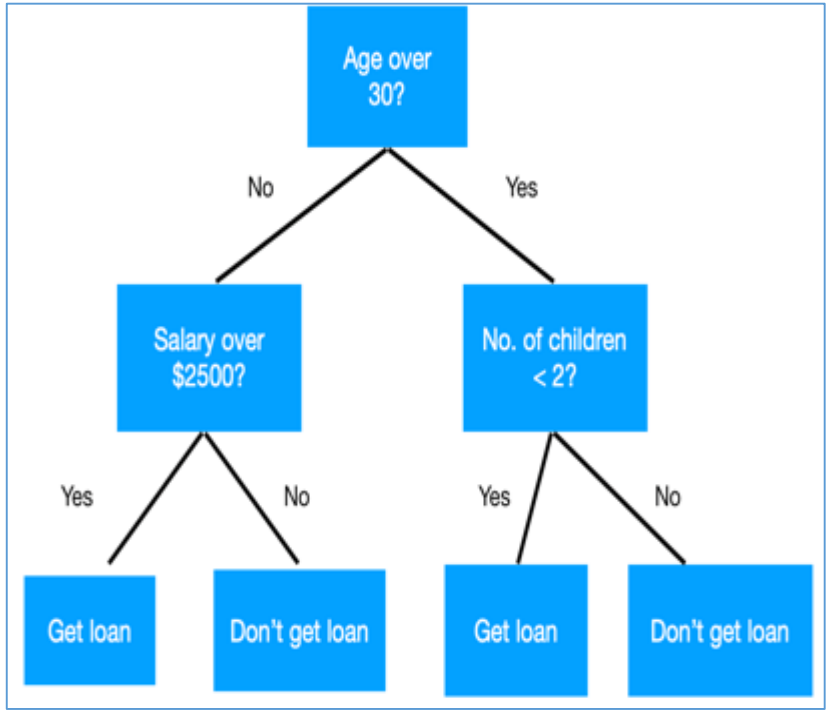
Decision Tree Learning -Decision tree representation

Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.

Figure represents a simple decision tree that is used to for a classification task of whether a customer gets a loan or not.

The input features are salary of the person, the number of children and the age of the person.

The decision tree uses these attributes or features and asks the right questions at the right step or node so as to classify whether the loan can be provided to the person or not.



Node

Blue colored rectangles that are shown above are what we call the nodes of the tree

Root Node or Root

Top most node is called as the root node -"age over 30 ?" is the root node

Leaf node

Nodes that do not have any children are called leaf nodes. (*Get Loan, Don't get Loan*). Leaf nodes hold the output labels.

Height of the above decision tree is 3
2 children for each node

The number of children for that node can also be more than 2

Decision Tree Learning -Appropriate problems for decision tree learning

Decision tree learning is generally best suited to problems with the following characteristics:

The training data may contain missing attribute values

Decision tree methods can be used even when some training examples have unknown values (e.g., if the ***Humidity*** of the day is known for only some of the training examples).

Decision Tree Learning - BASIC DECISION TREE LEARNING ALGORITHM

They are two basic Algorithm

- CART (Classification and Regression)
  - GINI Index
- ID3
  - Entropy function
  - Information Gain

THE BASIC DECISION TREE LEARNING ALGORITHM

- Most algorithms that have been developed for learning decision trees are variations on a core algorithm that employs a top-down, greedy search through the space of possible decision trees. This approach is exemplified by the ID3 algorithm and its successor C4.5

Decision Tree Learning - ID3 ALGORITHM

ID3 algorithm, stands for Iterative Dichotomiser 3, is a classification algorithm that follows a greedy approach of building a decision tree by selecting a best attribute that yields maximum Information Gain (IG) or minimum Entropy (H)

Major benefits of ID3 are:

Understandable prediction rules are created from the training data

Builds a short tree in relatively small time

ID3 only needs to test enough attributes until all data is classified

Builds a short tree in relatively small time

3 in ID3 stands for

Intelligent design

Identity and

Visionary technologies

What is the ID3 algorithm?

- ID3 stands for Iterative Dichotomiser 3
- ID3 is a precursor to the C4.5 Algorithm.
- The ID3 algorithm was invented by Ross Quinlan in 1975
- Used to generate a decision tree from a given data set by employing a top-down, greedy search, to test each attribute at every node of the tree.
- The resulting tree is used to classify future samples.

ID3 algorithm

ID3(Examples, Target_attribute, Attributes)

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a Root node for the tree
- If all Examples are positive, Return the single-node tree Root, with label = +
- If all Examples are negative, Return the single-node tree Root, with label = -
- If Attributes is empty, Return the single-node tree Root, with label = most common value of Target_attribute in Examples

ID3 algorithm

- Otherwise Begin
  - A ← the attribute from Attributes that best* classifies Examples
  - The decision attribute for Root ← A
  - For each possible value, v_i, of A,
    - Add a new tree branch below *Root*, corresponding to the test A = v_i
    - Let *Examples* $_{vi}$, be the subset of Examples that have value v_i for *A*
    - If *Examples* $_{vi}$, is empty
      - Then below this new branch add a leaf node with label = most common value of Target_attribute in Examples
      - Else below this new branch add the subtree ID3(*Examples* $_{vi}$, Targe_tattribute, Attributes – {A}))
- End
- Return Root

* The best attribute is the one with highest information gain

Decision Tree Learning - ID3 ALGORITHM

ID3 (Examples, Target_Attribute, Attributes)
Create a root node for the tree
If all examples are positive, Return the single-node tree Root, with label = +.
If all examples are negative, Return the single-node tree Root, with label = -.
If number of predicting attributes is empty, then Return the single node tree Root,
with label = most common value of the target attribute in the examples.
Otherwise Begin
A ← The Attribute that best classifies examples.
Decision Tree attribute for Root = A.
For each possible value, vi, of A,
Add a new tree branch below Root, corresponding to the test A = vi.
Let Examples(vi) be the subset of examples that have the value vi for A
If Examples(vi) is empty
Then below this new branch add a leaf node with label = most common target value in the examples
Else below this new branch add the subtree ID3 (Examples(vi), Target_Attribute, Attributes – {A})
End
Return Root

Decision Tree Algorithm:

Step 1

Begin the tree with the root node, says S, which contains the complete dataset

Step 2

Find the best attribute in the dataset using Attribute Selection Measure (ASM)

Step 3

Divide the S into subsets that contains possible values for the best attributes

Step 4

Generate the decision tree node, which contains the best attribute

Step 5

Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node

Decision Tree Learning - Issues in decision tree learning

Practical issues in learning decision trees include

Determining how deeply to grow the decision tree

Handling continuous attributes

Choosing an appropriate attribute selection measure

Handling training data with missing attribute values

Handling attributes with differing costs, and

Improving computational efficiency

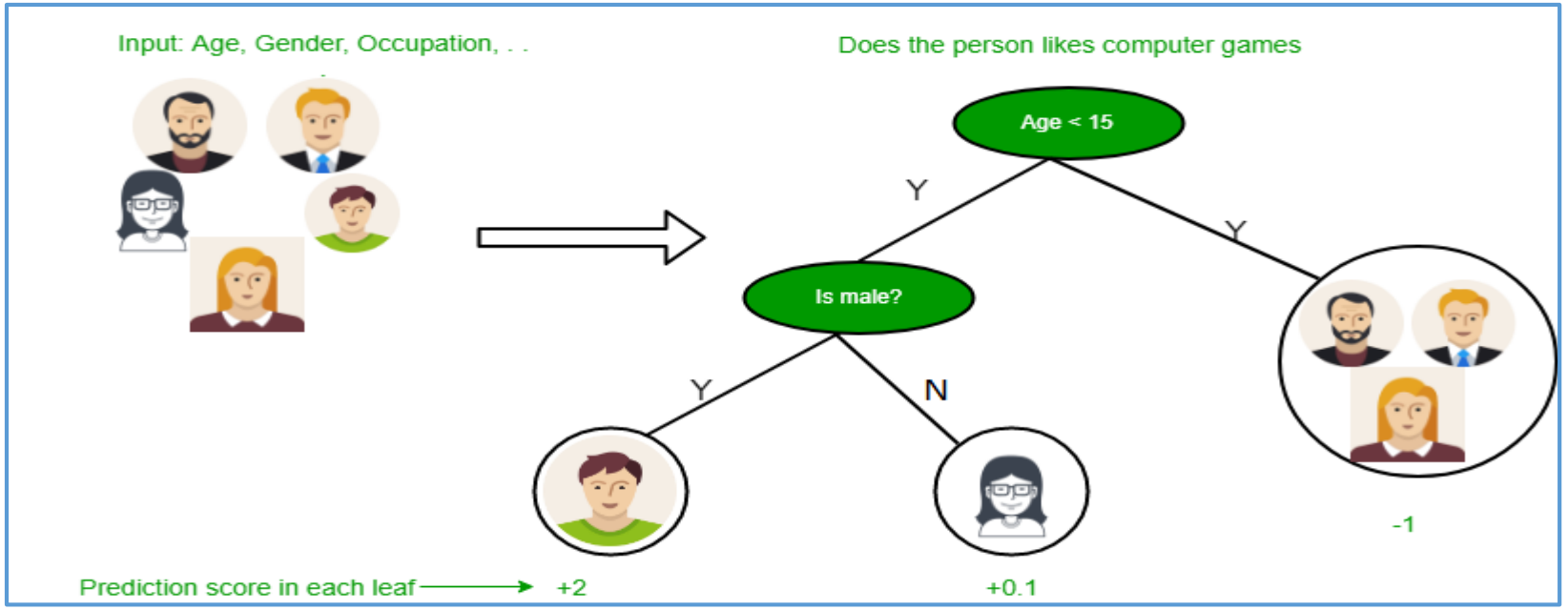
Decision Tree Introduction with example

Decision tree algorithm falls under the category of supervised learning

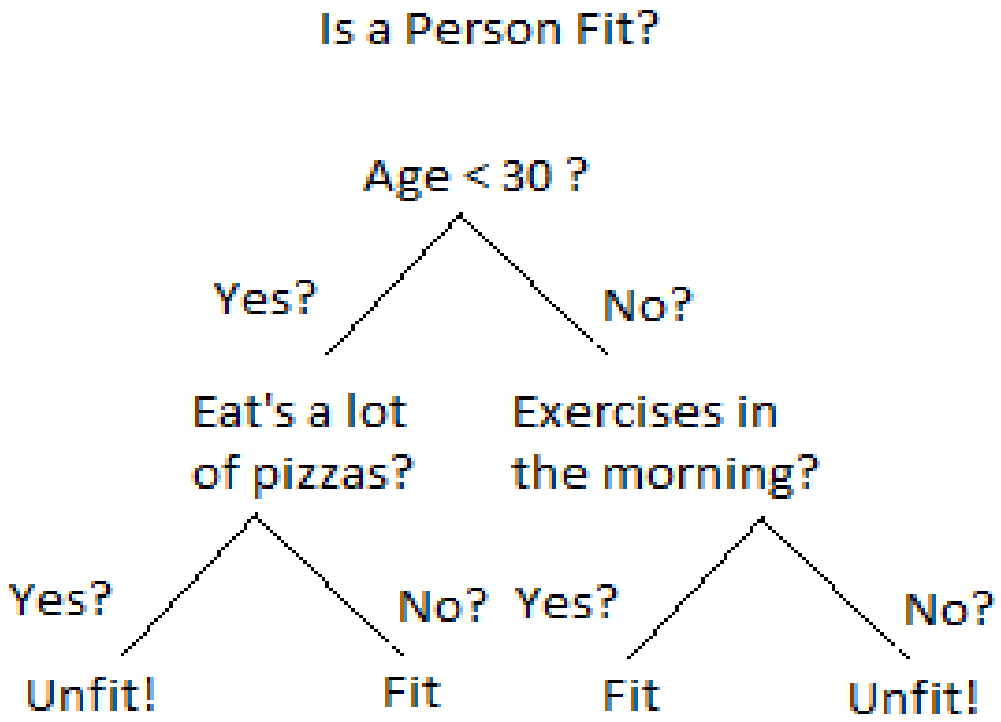
They can be used to solve both regression and classification problems

Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree

We can represent any boolean function on discrete attributes using the decision tree

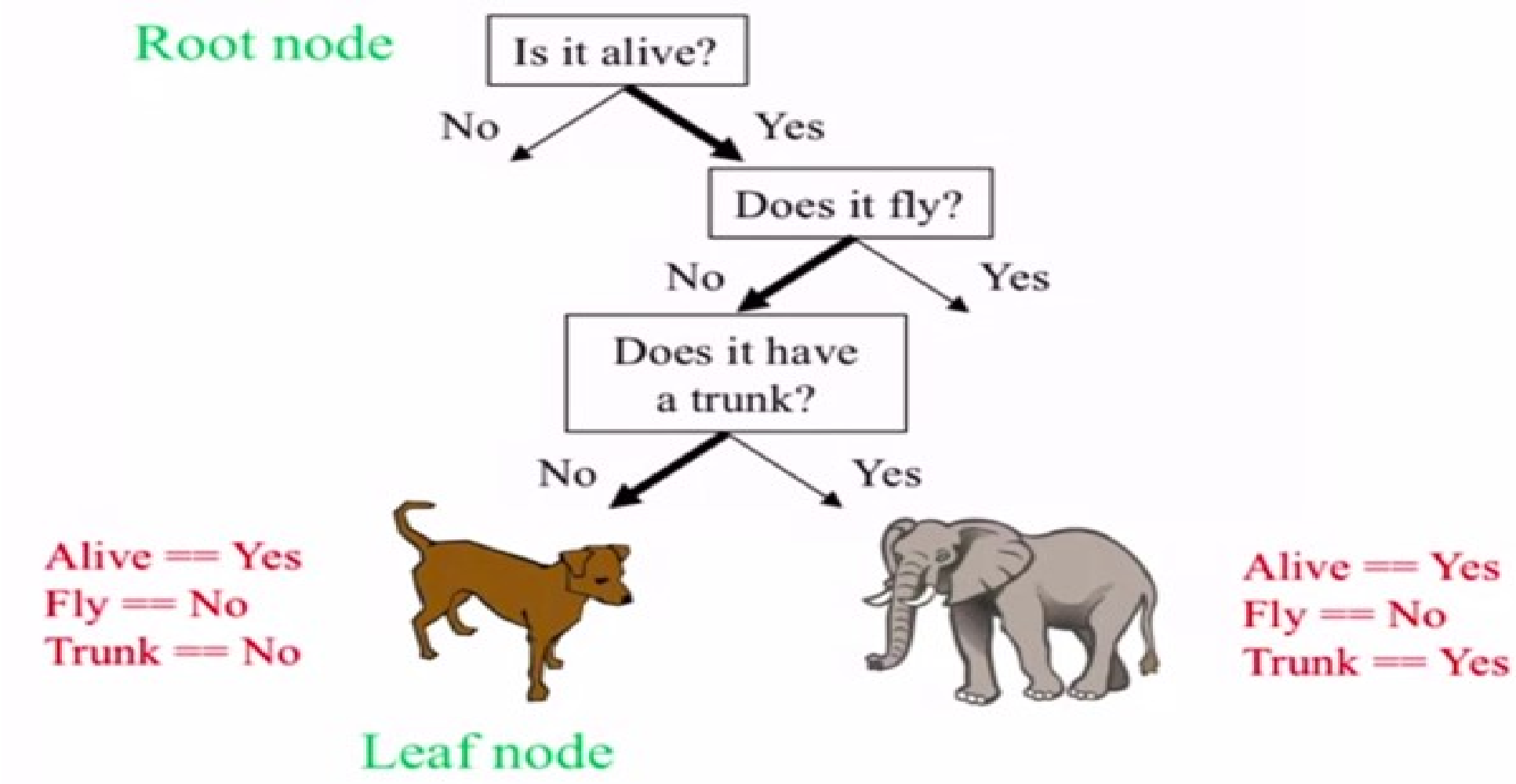


Decision Tree Introduction with example



MACHINE LEARNING

Decision Tree Introduction with example



MACHINE LEARNING

Decision Tree Introduction with example

Rain

No

Yes

Go Out

Stay In

Too Sunny

Cable Signal

No

Yes

No

Yes

Shopping

Movie

Coffee Shop

TV Shows

Root Node

Interior Nodes

Leaf Nodes

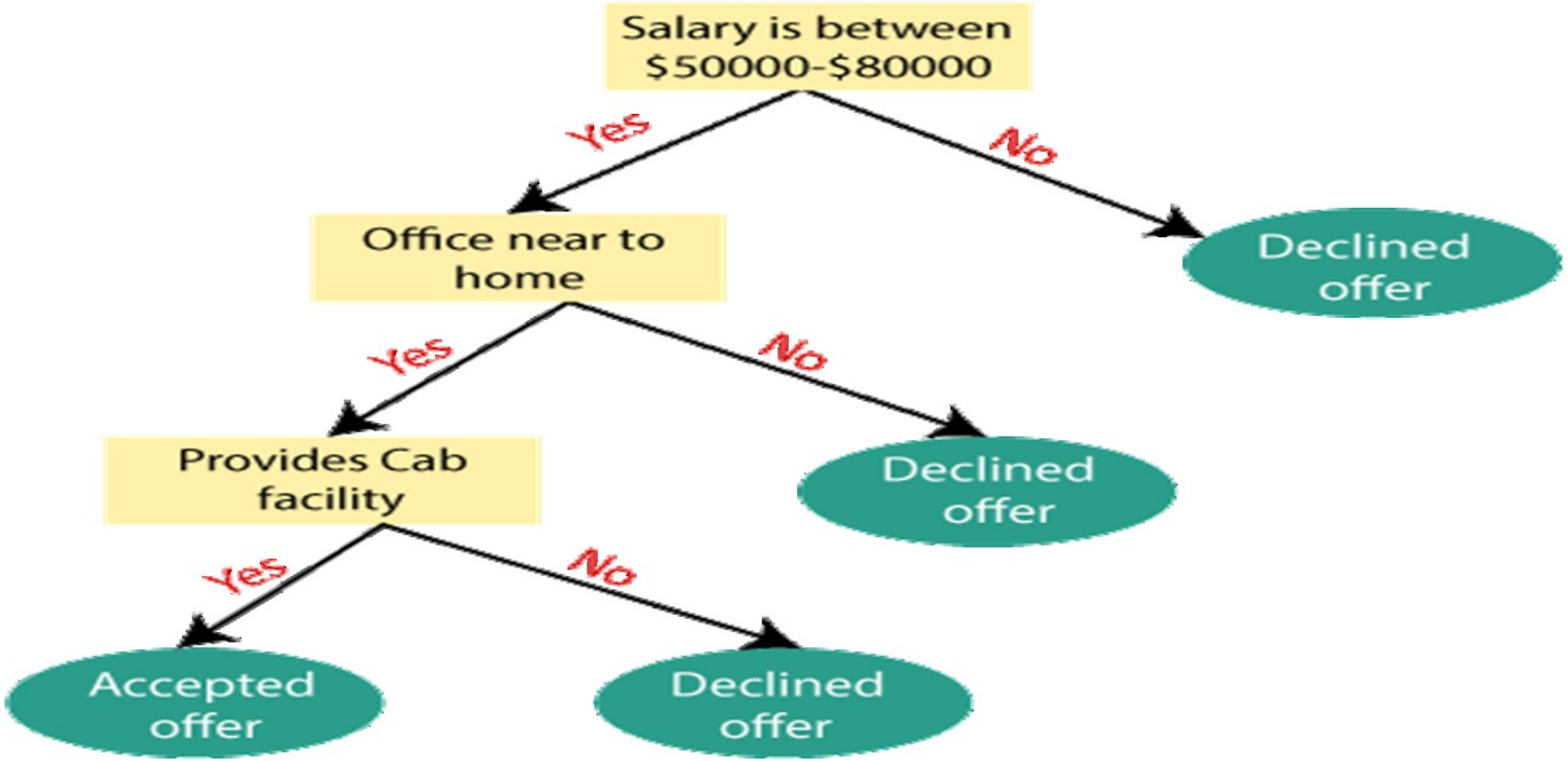
Monica's Weekend Plan

IntelliPaat

MACHINE LEARNING

Decision Tree Introduction with example

Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not.
So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM).
The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels.
The next decision node further gets split into one decision node (Cab facility) and one leaf node.
Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:



BASIC DECISION TREE LEARNING ALGORITHM

Decision Tree Learning - BASIC DECISION TREE LEARNING ALGORITHM

- ID3
- Entropy function
- Information Gain

Decision Tree Learning -Decision tree representation

Entropy

Entropy is the measurement of disorder or impurities in the information processed in machine learning

Entropy determines how a decision tree chooses to split data

Entropy is a measure of the randomness in the information being processed

The higher the entropy, the harder it is to draw any conclusions from that information

Example

Flipping a coin. When we flip a coin, then there can be two outcomes

Entropy is frequently used in one of the most common machine learning techniques-decision trees

$$E = -\sum_{i=1}^{C} p_i * log_2(p_i)$$

A measure for

Uncertainty
Purity
Information Content

C is the number of classes

$\mathbf{p_i}$ is the proportion of the $\mathbf{i}^{th}$ class in that set

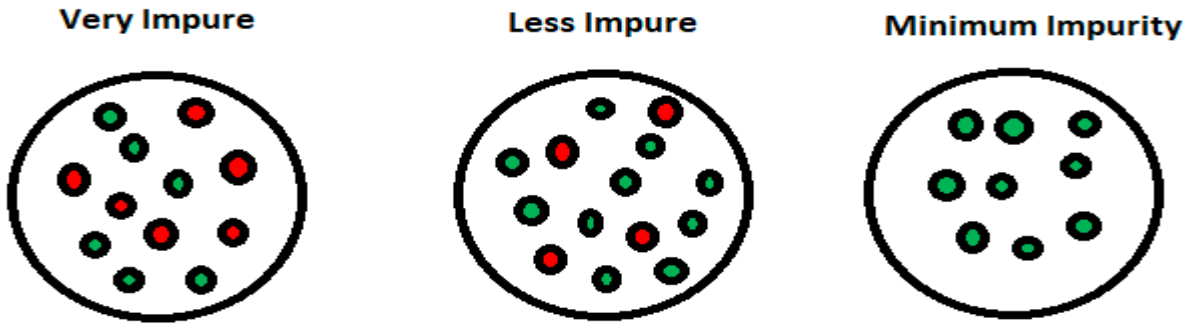
Decision Tree Learning -Entropy

Entropy

Entropy is an information theory metric that measures the impurity or uncertainty in a group of observations

Entropy determines how a decision tree chooses to split data.

The image below gives a better description of the purity of a set.



Entropy is the degree of uncertainty, impurity or disorder of a random variable, or a measure of purity

Entropy characterizes the impurity of an arbitrary class of examples

Entropy is the measurement of impurities or randomness in the data points

If all elements belong to a single class, then it is termed as "Pure", and if not then the distribution is named as "Impurity"

Decision Tree Learning -Entropy

Entropy

Entropy basically tells us how impure a collection of data is.

Impure here defines non-homogeneity.

Entropy is the measurement of homogeneity.

Example: Calculates the entropy of our data

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

Decision Tree Learning -Entropy

Entropy

Example: Calculates the entropy of our data

The dataset has

9 positive instances and

5 negative instances, therefore

$$Entropy([9+,5-]) = -(\frac{9}{14}log_2\frac{9}{14} + \frac{5}{14}log_2\frac{5}{14}) = 0.940$$ ---1

Which concludes, the data set is 94% impure or 94% non-homogeneous

What could do be the nature of Entropy for (7+,7-) and (14+,14-)

$$Entropy[7+,7-] = -(\frac{7}{14}log_2\frac{7}{14} + \frac{7}{14}log_2\frac{7}{14}) = 1$$ ---2

and

$$Entropy[14+,0-] = -(\frac{14}{14}log_2\frac{14}{14} + \frac{0}{14}log_2\frac{0}{14}) = 0$$ ---3

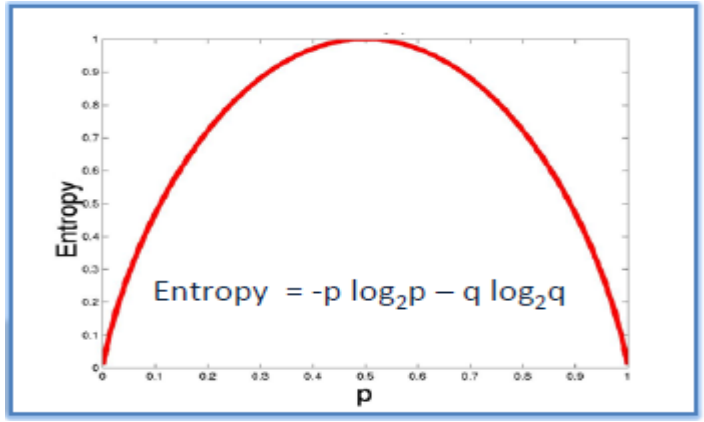
Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
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

By observing closely on equations **1**, **2** and **3**

If the data set is completely homogeneous then the impurity is 0, therefore entropy is 0 (equation **3**)

If the data set can be equally divided into two classes, then it is completely non-homogeneous & impurity is 100%, therefore entropy is 1 (equation **2.**).

If we try to plot the Entropy in a graph, it will look like Figure



It clearly shows that the Entropy is lowest when the data set is homogeneous and highest when the data set is completely non-homogeneous

Decision Tree Learning -Entropy

Entropy

Entropy --- measuring homogeneity of a learning set (Tom M. Mitchell,1997,p55)

lets assume, without loss of generality, that the resulting decision tree classifies instances into two categories, we'll call them P(positive)and N(negative)

Given a set S, containing these positive and negative target, the entropy of S related to this Boolean classification is:

Entropy(S)= - P(positive)log2P(positive) - P(negative)log2P(negative)

P(positive): proportion of positive examples in S

P(negative): proportion of negative examples in S

Example

if S is (0.5+, 0.5-) then Entropy(S) is 1, if S is (0.67+, 0.33-) then Entropy(S) is 0.92, if P is (1+, 0-) then Entropy(S) is 0

Note that the more uniform is the probability distribution, the greater is its information

Decision Tree Learning -Entropy

Entropy

Entropy is a measure of impurity of a node. By Impurity, We mean to measure the heterogeneity at a particular node.

Example:

Assume that we have 50 red balls and 50 blue balls in a Set.

In this case , proportions of the balls of both the colors are equal. Hence, the entropy would be 1. which means that the set is impure

But, If the set has 98 red balls and 2 blue balls instead of the 50-50 proportion (The same logic can be applied for a set of 98 blue balls and 2 red balls | which category does not matter What matters is that one category dominates well over the other)

Then the entropy would be low (somewhere closer to 0)

This is because now the set is mostly pure as it mostly contains balls belonging to one category

Because of this , the heterogeneity is reduced

Decision Tree Learning -Information Gain

Information Gain

The concept of entropy plays an important role in measuring the information gain

Information gain is based on the information theory

Information gain is used for determining the best features/attributes that render maximum information about a class

Information gain follows the concept of entropy while aiming at decreasing the level of entropy, beginning from the root node to the leaf nodes

Information gain computes the difference between entropy before and after split and specifies the impurity in class elements

Information Gain = Entropy before splitting - Entropy after splitting

Information gain computes the difference between entropy before and after split and specifies the impurity in class elements

Information gain (IG) measures how much "information" a feature gives us about the class

Information gain (IG) tells us how important a given attribute of the feature vectors is

Information gain (IG) is used to decide the ordering of attributes in the nodes of a decision tree

Decision Tree Learning -Information Gain

Information Gain

Information gain as a measure of how much information a feature provides about a class.

The information gain is the amount of information gained about a random variable or signal from observing another random variable

Information gain helps to determine the order of attributes in the nodes of a decision tree

Main node is referred to as the parent node, whereas sub-nodes are known as child nodes

We can use information gain to determine how good the splitting of nodes in a decision tree

The calculation of information gain should help us understand this concept better.

$$ Gain = E_{parent} - E_{children} $$

Gain represents information gain

$ E_{parent} $ is the entropy of the parent node and

E_{children} is the average entropy of the child nodes.

Decision Tree Learning -Information Gain

Information Gain

Information gain is the reduction in entropy or surprise by transforming a dataset and is often used in training decision trees

Information gain is calculated by comparing the entropy of the dataset before and after a transformation

We can use information gain to determine how good the splitting of nodes in a decision tree

Information gain is a decrease in entropy

Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values

ID3 (Iterative Dichotomiser) decision tree algorithm uses information gain

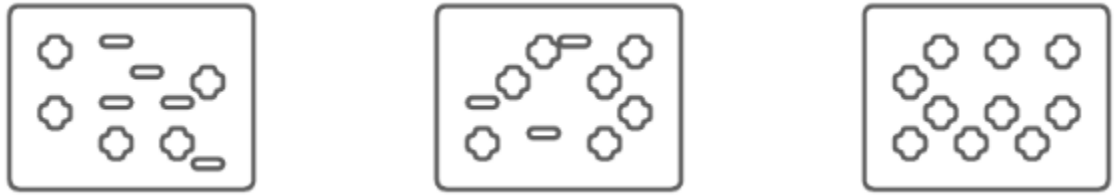
Information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches)

Decision Tree Learning -Information Gain

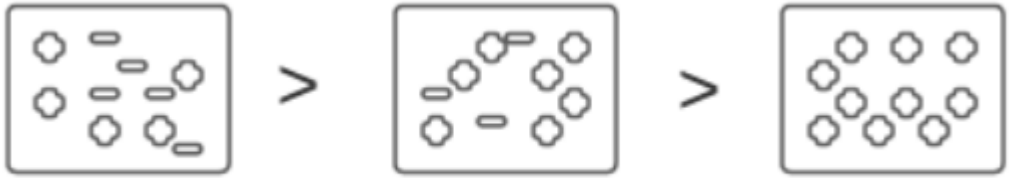
Information Gain

The information gained in the decision tree can be defined as the amount of information improved in the nodes before splitting them for making further decisions

Example:



As we can see in these three nodes we have data of two classes and here in node 3 we have data for only one class and similarly, we have less data for the second class than the first class in node 2, and node 1 is balanced



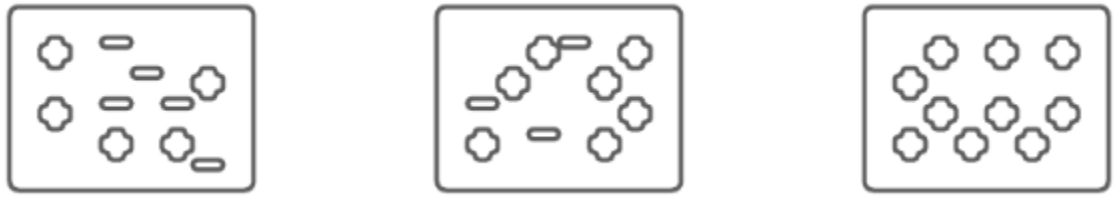
By this above, we can say that in node three we don't need to make any decision because all the instances are representing the direction of the decision in the class first side wherein in node 1 there are 50% chances to decide the direction of both classes

Decision Tree Learning -Information Gain

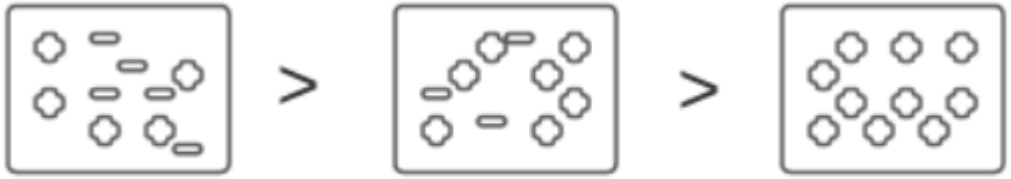
Information Gain

We can say that in node 1 we are required more information than the other nodes to describe a decision

Example:



As we can see in these three nodes we have data of two classes and here in node 3 we have data for only one class and similarly, we have less data for the second class than the first class in node 2, and node 1 is balanced

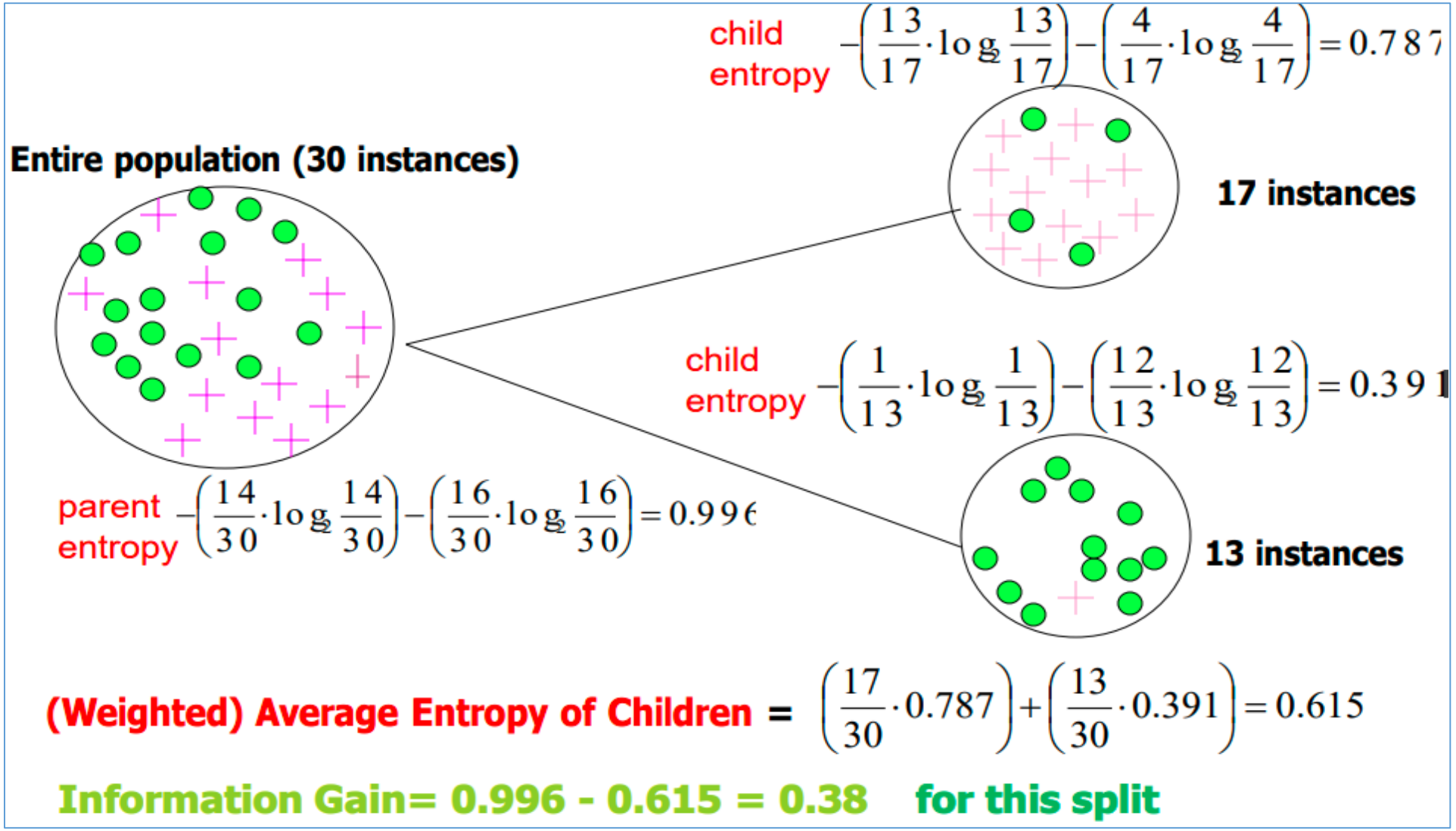


By this above, we can say that in node three we don't need to make any decision because all the instances are representing the direction of the decision in the class first side wherein in node 1 there are 50% chances to decide the direction of both classes

By the above, we can say the information gain in node 1 is higher.

Decision Tree Learning -Information Gain

Information Gain Information Gain = entropy(parent) – [average entropy(children)]



CART (Classification And Regression Tree) in Machine Learning

CART(Classification And Regression Tree) is a variation of the decision tree algorithm.

CART(Classification And Regression Tree) can handle both classification and regression tasks

Scikit-Learn uses the Classification and Regression Tree (CART) algorithm to train Decision Trees (also called "growing" trees)

Classification tree

Classification tree is an algorithm where the target variable is categorical

The algorithm is then used to identify the "Class" within which the target variable is most likely to fall

Classification trees are used when the dataset needs to be split into classes that belong to the response variable(like yes or no)

CART (Classification And Regression Tree) in Machine Learning

Regression tree

Regression tree is an algorithm where the target variable is continuous and the tree is used to predict its value

Regression trees are used when the response variable is continuous

Example

if the response variable is the temperature of the day

The algorithm is then used to identify the "Class" within which the target variable is most likely to fall

Classification trees are used when the dataset needs to be split into classes that belong to the response variable(like yes or no)

CART (Classification And Regression Tree) in Machine Learning

CART Algorithm

CART is a predictive algorithm used in Machine learning and it explains how the target variable's values can be predicted based on other matters

CART is a decision tree where each fork is split into a predictor variable and each node has a prediction for the target variable at the end

CART (Classification And Regression Tree) in Machine Learning

CART model representation

Greedy algorithm

In Greedy algorithm The input space is divided using the Greedy method which is known as a recursive binary spitting

This is a numerical method within which all of the values are aligned and several other split points are tried and assessed using a cost function

CART (Classification And Regression Tree) in Machine Learning

CART model representation

Stopping Criterion

As it works its way down the tree with the training data, the recursive binary splitting method described above must know when to stop splitting

The most frequent halting method is to utilize a minimum amount of training data allocated to every leaf node

If the count is smaller than the specified threshold, the split is rejected and also the node is considered the last leaf node

CART (Classification And Regression Tree) in Machine Learning

CART model representation

Tree pruning

Decision tree's complexity is defined as the number of splits in the tree

Trees with fewer branches are recommended as they are simple to grasp and less prone to cluster the data

Working through each leaf node in the tree and evaluating the effect of deleting it using a hold-out test set is the quickest and simplest pruning approach

CART (Classification And Regression Tree) in Machine Learning

CART model representation

Data preparation for the CART

No special data preparation is required for the CART algorithm

Concept of Model Ensembling

A model is an abstraction of phenomena in the real world

A metamodel is yet another abstraction, highlighting properties of the model itself

A metamodel is **a model that consists of statements about models**

Models we can use to obtain a meta-model are called weak learners

In this ensemble learning architecture, the inputs are passed to each weak learner while also collecting their predictions

We can use the combined prediction to build a final ensemble model

Concept of Model Ensembling

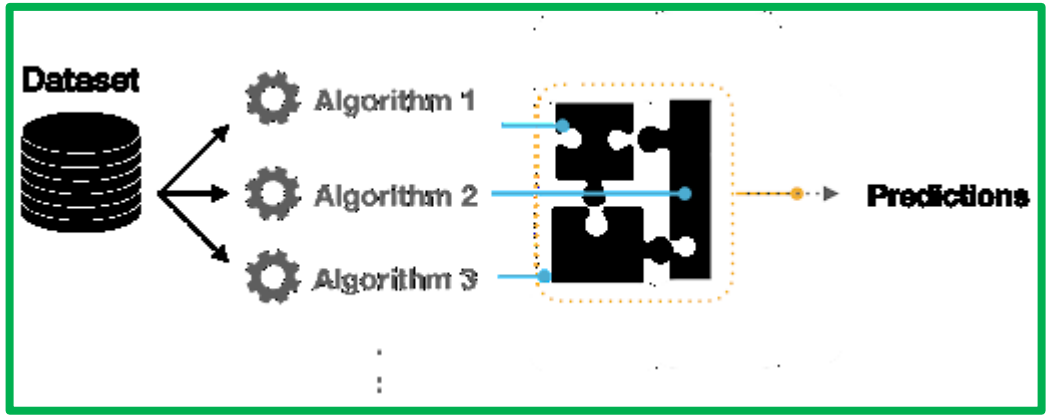
A single algorithm may not make the perfect prediction for a given data set

If we build and combine multiple models, we have the chance to boost the overall accuracy

We then implement the combination of models by aggregating the output from each model with two objectives

- Reducing the model error
- Maintaining the model's generalization

Implement such aggregation using different techniques, sometimes referred to as meta-algorithms



Concept of Model Ensembling

Ensemble modeling is a process where multiple diverse models are created to predict an outcome, either by using many different modeling algorithms or using different training data sets

Ensemble model then aggregates the prediction of each base model and results in once final prediction for the unseen data

The most popular ensemble methods are

- Boosting
- Bagging, and
- Stacking

Ensemble methods are ideal for Regression and Classification

Where they reduce bias and variance to boost the accuracy of models

Concept of Model Ensembling

Boosting

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method

AdaBoost is an ensemble learning method (also known as "meta-learning") which was initially created to increase the efficiency of binary classifiers

AdaBoost uses an iterative approach to learn from the mistakes of weak classifiers, and turn them into strong ones

Concept of Model Ensembling

Bagging

Bagging, also known as bootstrap aggregation, is the ensemble learning method that is commonly used to reduce variance within a noisy dataset

In bagging, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once

Concept of Model Ensembling

Stacking

Stacking is one of the most popular ensemble machine learning techniques used to predict multiple nodes to build a new model and improve model performance

Stacking enables us to train multiple models to solve similar problems, and based on their combined output, it builds a new model with improved performance

Thank you for Listening

Any Questions

Dr DV Ramana
Data Stratagist
Wissen Infotech

Mail Address: pythonpmg@gmail.com

To contact: +91 9959423084