1. **Recognize the differences between supervised, semi-supervised, and unsupervised learning.**

**Supervised Learning**

Labeled dataset is a set of data with predictors (input variables) and output (response/target variables). Supervised learning is where you have such a dataset, and you are searching for the best representative function that link or map predictors to relevant target. With availability of response variables, we are able to compare prediction and actual label, and hence modification applied to reduce misprediction and improve model.

Supervised learning problem can be split further into classification or regression problem according to type of output variables. A real-valued output variable, whether it is discrete or continuous, is a regression problem. A categorical output variable is classification problem. Algorithms applied should be appropriate to the corresponding categories of learning problem. Figure 1 gives examples for each category.

The ultimate goal of supervised learning is to build a model that generalizes well on future unseen data. This raises the concern about possible overfitting phenomenon. We are not going to dive further into relevant topic, brief idea about this can be found [here](https://medium.com/@hshan0103/understanding-bias-variance-trade-off-from-learning-curve-a64b4223bb02).

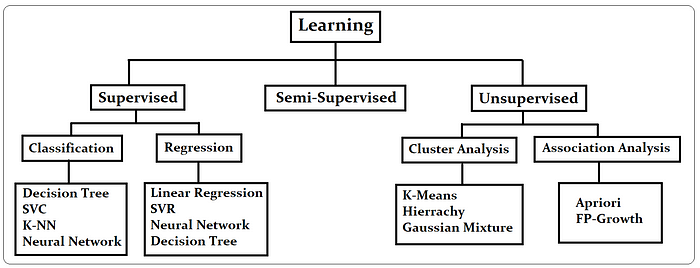


Figure 1: Different learning problems.

**Unsupervised Learning**

When we have unlabeled data, we are handling unsupervised learning problem. We are clueless whether we are clustering the data points/ tuples correctly. Therefore, challenge for unsupervised learning will be deciding point to stop learning and evaluation of model built. We are not working to do any prediction here, as ‘machine’ is not taught how to predict. Instead, we are exploring and reporting underlying insights and structure of data. Three typical areas applying unsupervised learning is clustering, dimensionality reduction and association analysis.

Clustering is the situation where we are trying to capture the common characteristics between tuples in dataset and group them according to their similarities. Association analysis is conducted to find interesting hidden rules or relationships between tuples. When we have dataset with large input dimension, one will find smaller set of input variables to simplify or reduce original input dimension, preserving as much information as possible (minimum loss of information). Here is the point where dimensionality reduction is applied as crucial feature engineering step for more time efficient training.

**Semi-Supervised Learning**

This kind of learning lies in between supervised and unsupervised. It has mixture of labeled and unlabeled data with larger proportion for latter. Data available today is mostly of this form, as it is costly, time consuming, and requires expertise to get a huge data labeled. When such data is given, unsupervised learning technique is used and followed by supervised learning. Data is divided into clusters and missing labels are predicted. It is assumed that the data points in same cluster posing same or similar label, basic way to get the labels is by voting or average value among data falling in same cluster.

1. **Describe in detail any five examples of classification problems.**

**1 - Email Spam**

The goal is to predict whether an email is a *spam*and should be delivered to the Junk folder.

There are more than one method of identifying a mail as a spam. A simple method is discussed.

The raw data comprises only the text part but ignores all images. Text is a simple sequence of words which is the input (X). The goal is to predict the binary response Y: spam or not.

The first step is to process the raw data into a vector, which can be done in several ways. The method followed here is based on the relative frequencies of most common words and punctuation marks in e-mail messages. A set of 57 such words and punctuation marks are pre-selected by researchers. This is where domain knowledge plays an important role.

Given these 57 most commonly occurring words and punctuation marks, then, in every e-mail message we would compute a relative frequency for each word, i.e., the percentage of times this word appears with respect to the total number of words in the email message.

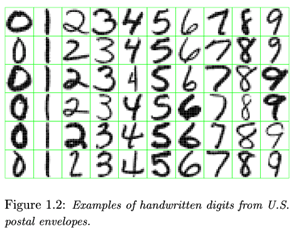
In the current example, 4601 email messages were considered in the training sample. These e-mail messages were identified as either a good e-mail or spam after reading the emails and assuming implicitly that human decision is perfect (an arguable point!). Relative frequency of the 57 most commonly used words and punctuation based on this set of emails was constructed. This is an example of supervised learning as in the training data the response Y is known.

In the future when a new email message is received, the algorithm will analyze the text sequence and compute the relative frequency for these 57 identified words. This is the new input vector to be classified into spam or not through the learning algorithm.

**2 - Handwritten Digit Recognition**

The goal is to identify images of single digits 0 - 9 correctly.

The raw data comprises images that are scaled segments from five-digit ZIP codes. In the diagram below every green box is one image. The original images are very small, containing only 16 × 16 pixels. For convenience the images below are enlarged, hence the pixelation or 'boxiness' of the numbers.



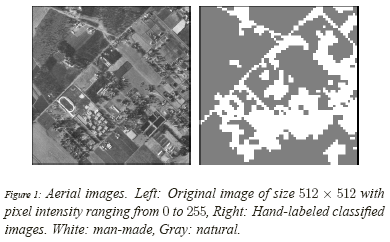
Every image is to be identified as 0 or 1 or 2 ... or 9. Since the numbers are handwritten, the task is not trivial. For instance, a '5' sometimes can very much look like a '6', and '7' is sometimes confused with '1'.

To the computer, an image is a matrix, and every pixel in the image corresponds to one entry in the matrix. Every entry is an integer ranging from a pixel intensity of 0 (black) to 255 (white). Hence the raw data can be submitted to the computer directly without any feature extraction. The image matrix was scanned row by row and then arranged into a large 256-dimensional vector. This is used as the input to train the classifier. Note that this is also a supervised learning algorithm where Y, the response, is multi-level and can take 10 values.

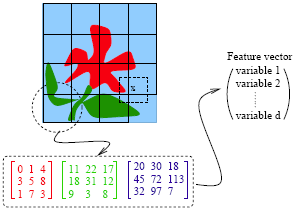
**3 - Image segmentation**

Here is a more complex example of an image processing problem. The satellite images are to be identified into man-made or natural regions. For instance, in the aerial images shown below, buildings are labeled as man-made, and the vegetation areas are labeled as natural.

These grayscale images are much larger than the previous example. These images are 512 × 512 pixels and again because these are grayscale images we can present pixel intensity with numbers 0 to 255.



In the previous example of hand-written image identification, because of the small size of the images, no feature extraction was done. However in this problem feature extraction is necessary. A standard method of feature extraction in an image processing problem is to divide images into blocks of pixels or to form a neighborhood around each pixel. As is shown in the following diagram, after dividing the images into blocks of pixels or forming a neighborhood around each pixel, each block may be described by several features. As we have seen in the previous example, grayscale images can be represented by one matrix. Every entry in a greyscale image is an integer ranging from a pixel intensity of 0 (black) to 255 (white). Color images are represented by values of RGB (red, green and blue). Color images, therefore, are represented by 3 such matrices as seen below.



For each block, a few features (or statistics) may be computed using the color vectors for the pixels in the block. This set forms a feature vector for every block.

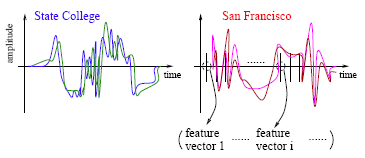
Examples of features:

* Average of R, G and B values for pixels in one block
* Variance of the brightness of the pixels (brightness is the average of RGB color values). Small variance indicates the block is visually smooth.

The feature vectors for the blocks sometimes are treated as independent samples from an unknown distribution. Ignoring f the spatial dependence among feature vectors results in performance loss. To make the learning algorithm efficient the spatial dependence needs to be exploited. Only then the accuracy in classification will improve.

**4 - Speech Recognition**

Another interesting example of data mining deals with speech recognition. For instance, if you call the University Park Airport, the system might ask you your flight number, or your origin and destination cities. The system does a very good job recognizing city names. This is a classification problem, in which each city name is a class. The number of classes is very big but finite.



The raw data involves voice amplitude sampled at discrete time points (a time sequence), which may be represented in the waveforms as shown above. In speech recognition, a very popular method is the *Hidden Markov Model*.

At every time point, one or more features, such as frequencies, are computed. The speech signal essentially becomes a sequence of frequency vectors. This sequence is assumed to be an instance of a hidden Markov model (HMM). An HMM can be estimated using multiple sample sequences under the same class (e.g., city name).

*Hidden Markov Model (HMM) Methodology:*

HMM captures the time dependence of the feature vectors. The HMM has unspecified parameters that need to be estimated. Based on the sample sequences, model estimation takes place and an HMM is obtained. This HMM is like a mathematical signature for each word. Each city name, for example, will have a different signature. In the diagram above the signatures corresponding to State College and San Francisco are compared. It is possible that several models are constructed for one word or phrase. For instance, there may be a model for a female voice as opposed to another for a male voice.

When a customer calls in for information and utters origin or destination city pairs, the system computes the likelihood of what the customer uttered under possibly thousands of models. The system finds the HMM that yields the maximum likelihood and identifies the word as the one associated with that HMM.

**5 - DNA Expression Microarray**

Our goal here is to identify disease or tissue types based on the gene expression levels.

For each sample taken from a tissue of a particular disease type, the expression levels of a very large collection of genes are measured. The input data goes through a data cleaning process. Data cleaning may include but is certainly not limited to, normalization, elimination of noise and perhaps log-scale transformations. A large volume of literature exists on the topic of cleaning microarray data.

In the example considered  96 samples were taken from 9 classes or types of tissues. It was expensive to collect the tissue samples, at least in the early days. Therefore, the sample size is often small but the dimensionality of data is very high. Every sample is measured on 4026 genes. very often microarray data analysis has its own challenges with a small number of observations and very large number of features from each observation.

**6 - DNA Sequence Classification**

Each genome is made up of DNA sequences and each DNA segment has specific biological functions. However there are DNA segments which are non-coding, i.e. they do not have any biological function (or their functionalities are not yet known). One problem in DNA sequencing is to label the sampled segments as coding or non-coding (with a biological function or without).

The raw DNA data comprises sequences of letters, e.g., A, C, G, T for each of the DNA sequences. One method of classification assumes the sequences to be realizations of random processes. Different random processes are assumed for different classes of sequences.

**3. Describe each phase of the classification process in detail.**

**Phases of classification algorithm. Phase I, is the training phase , where a classifier is learned from a set of measurements representative of the data. In Phase II, the testing phase , the learned classifier is used to discriminate between measurements that were not necessarily encountered during the training phase.**

**4. Go through the SVM model in depth using various scenarios.**

# Support Vector Machine Algorithm

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



**Example:** SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:

Backward Skip 10sPlay VideoForward Skip 10s



SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

## Types of SVM

**SVM can be of two types**

**Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

* **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

## Hyperplane and Support Vectors in the SVM algorithm:

**Hyperplane:** There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

**Support Vectors:**

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

## How does SVM works?

**Linear SVM:**

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



**Non-Linear SVM:**

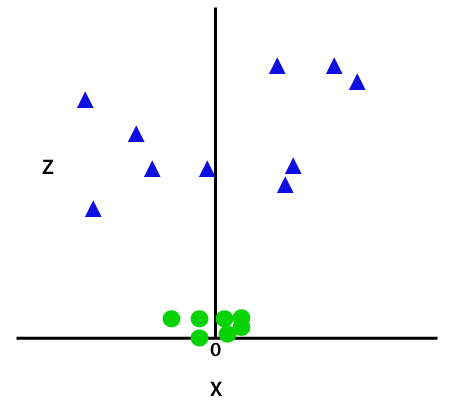
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

z=x2 +y2

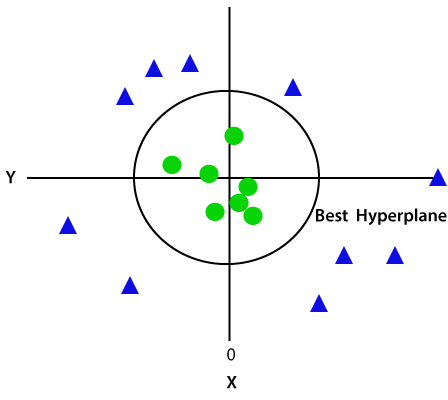
By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:



Hence we get a circumference of radius 1 in case of non-linear data.

**Python Implementation of Support Vector Machine**

Now we will implement the SVM algorithm using Python. Here we will use the same dataset **user\_data**, which we have used in Logistic regression and KNN classification.

* **Data Pre-processing step**

**5. What are some of the benefits and drawbacks of SVM?**

# Advantages of Support Vector Machine:

1. SVM works relatively well when there is a clear margin of separation between classes.
2. SVM is more effective in high dimensional spaces.
3. SVM is effective in cases where the number of dimensions is greater than the number of samples.
4. SVM is relatively memory efficient

# Disadvantages of Support Vector Machine:

1. SVM algorithm is not suitable for large data sets.
2. SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.
3. In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.
4. As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.

**6. Go over the kNN model in depth.**

KNN works in three main steps: (1) calculating the distance between the query point and each training point, (2) selecting the k-nearest neighbors to the query point, and (3) predicting the class or value of the query point based on the majority class or the mean value of the neighbors, respectively.

**7. Discuss the kNN algorithm's error rate and validation error.**

The value of k in the KNN algorithm is related to the error rate of the model. A small value of k could lead to overfitting as well as a big value of k can lead to underfitting. Overfitting imply that the model is well on the training data but has poor performance when new data is coming. Underfitting refers to a model that is not good on the training data and also cannot be generalized to predict new data.

**8. For kNN, talk about how to measure the difference between the test and training results.**

* K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
* K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
* K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
* K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
* K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
* It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
* KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
* **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

**9. Create the kNN algorithm. What is a decision tree, exactly? What are the various kinds of nodes? Explain all in depth.**

# Decision Tree

**Decision trees**are a popular and powerful tool used in various fields such as machine learning, data mining, and statistics. They provide a clear and intuitive way to make decisions based on data by modeling the relationships between different variables. This article is all about what decision trees are, how they work, their advantages and disadvantages, and their applications.

## What is a Decision Tree?

A **decision tree**is a flowchart-like structure used to make decisions or predictions. It consists of nodes representing decisions or tests on attributes, branches representing the outcome of these decisions, and leaf nodes representing final outcomes or predictions. Each internal node corresponds to a test on an attribute, each branch corresponds to the result of the test, and each leaf node corresponds to a class label or a continuous value.

## Structure of a Decision Tree

1. **Root Node**: Represents the entire dataset and the initial decision to be made.
2. **Internal Nodes**: Represent decisions or tests on attributes. Each internal node has one or more branches.
3. **Branches**: Represent the outcome of a decision or test, leading to another node.
4. **Leaf Nodes**: Represent the final decision or prediction. No further splits occur at these nodes.

## How Decision Trees Work?

The process of creating a decision tree involves:

1. **Selecting the Best Attribute**: Using a metric like Gini impurity, entropy, or information gain, the best attribute to split the data is selected.
2. **Splitting the Dataset**: The dataset is split into subsets based on the selected attribute.
3. **Repeating the Process**: The process is repeated recursively for each subset, creating a new internal node or leaf node until a stopping criterion is met (e.g., all instances in a node belong to the same class or a predefined depth is reached).

## Metrics for Splitting

* **Gini Impurity**: Measures the likelihood of an incorrect classification of a new instance if it was randomly classified according to the distribution of classes in the dataset.
  + Gini=1–∑𝑖=1(𝑝𝑖)2Gini=1–∑*i*=1*n*​(*pi*​)2, where pi​ is the probability of an instance being classified into a particular class.
* **Entropy**: Measures the amount of uncertainty or impurity in the dataset.
  + Entropy=−∑𝑖=1𝑛𝑝𝑖log⁡2(𝑝𝑖)Entropy=−∑*i*=1*n*​*pi*​log2​(*pi*​), where pi​ is the probability of an instance being classified into a particular class.
* **Information Gain**: Measures the reduction in entropy or Gini impurity after a dataset is split on an attribute.
  + InformationGain=Entropyparent–∑𝑖=1(∣𝐷𝑖∣∣𝐷∣∗Entropy(𝐷𝑖))InformationGain=Entropyparent​–∑*i*=1*n*​(∣*D*∣∣*Di*​∣​∗Entropy(*Di*​)), where Di​ is the subset of D after splitting by an attribute.

## Advantages of Decision Trees

* **Simplicity and Interpretability**: Decision trees are easy to understand and interpret. The visual representation closely mirrors human decision-making processes.
* **Versatility**: Can be used for both classification and regression tasks.
* **No Need for Feature Scaling**: Decision trees do not require normalization or scaling of the data.
* **Handles Non-linear Relationships**: Capable of capturing non-linear relationships between features and target variables

## Disadvantages of Decision Trees

* **Overfitting**: Decision trees can easily overfit the training data, especially if they are deep with many nodes.
* **Instability**: Small variations in the data can result in a completely different tree being generated.
* **Bias towards Features with More Levels**: Features with more levels can dominate the tree structure.

## Pruning

To overcome **overfitting, pruning**techniques are used. Pruning reduces the size of the tree by removing nodes that provide little power in classifying instances. There are two main types of pruning:

* **Pre-pruning (Early Stopping)**: Stops the tree from growing once it meets certain criteria (e.g., maximum depth, minimum number of samples per leaf).
* **Post-pruning**: Removes branches from a fully grown tree that do not provide significant power.

## Applications of Decision Trees

* **Business Decision Making**: Used in strategic planning and resource allocation.
* **Healthcare**: Assists in diagnosing diseases and suggesting treatment plans.
* **Finance**: Helps in credit scoring and risk assessment.
* **Marketing**: Used to segment customers and predict customer behavior.

**11. Describe the different ways to scan a decision tree.**

# Decision Tree

**Last Updated :**17 May, 2024

**Decision trees**are a popular and powerful tool used in various fields such as machine learning, data mining, and statistics. They provide a clear and intuitive way to make decisions based on data by modeling the relationships between different variables. This article is all about what decision trees are, how they work, their advantages and disadvantages, and their applications.

## What is a Decision Tree?

A **decision tree**is a flowchart-like structure used to make decisions or predictions. It consists of nodes representing decisions or tests on attributes, branches representing the outcome of these decisions, and leaf nodes representing final outcomes or predictions. Each internal node corresponds to a test on an attribute, each branch corresponds to the result of the test, and each leaf node corresponds to a class label or a continuous value.

## Structure of a Decision Tree

1. **Root Node**: Represents the entire dataset and the initial decision to be made.
2. **Internal Nodes**: Represent decisions or tests on attributes. Each internal node has one or more branches.
3. **Branches**: Represent the outcome of a decision or test, leading to another node.
4. **Leaf Nodes**: Represent the final decision or prediction. No further splits occur at these nodes.

## How Decision Trees Work?

The process of creating a decision tree involves:

1. **Selecting the Best Attribute**: Using a metric like Gini impurity, entropy, or information gain, the best attribute to split the data is selected.
2. **Splitting the Dataset**: The dataset is split into subsets based on the selected attribute.
3. **Repeating the Process**: The process is repeated recursively for each subset, creating a new internal node or leaf node until a stopping criterion is met (e.g., all instances in a node belong to the same class or a predefined depth is reached).

## Metrics for Splitting

* **Gini Impurity**: Measures the likelihood of an incorrect classification of a new instance if it was randomly classified according to the distribution of classes in the dataset.
  + Gini=1–∑𝑖=1(𝑝𝑖)2Gini=1–∑*i*=1*n*​(*pi*​)2, where pi​ is the probability of an instance being classified into a particular class.
* **Entropy**: Measures the amount of uncertainty or impurity in the dataset.
  + Entropy=−∑𝑖=1𝑛𝑝𝑖log⁡2(𝑝𝑖)Entropy=−∑*i*=1*n*​*pi*​log2​(*pi*​), where pi​ is the probability of an instance being classified into a particular class.
* **Information Gain**: Measures the reduction in entropy or Gini impurity after a dataset is split on an attribute.
  + InformationGain=Entropyparent–∑𝑖=1(∣𝐷𝑖∣∣𝐷∣∗Entropy(𝐷𝑖))InformationGain=Entropyparent​–∑*i*=1*n*​(∣*D*∣∣*Di*​∣​∗Entropy(*Di*​)), where Di​ is the subset of D after splitting by an attribute.

## Advantages of Decision Trees

* **Simplicity and Interpretability**: Decision trees are easy to understand and interpret. The visual representation closely mirrors human decision-making processes.
* **Versatility**: Can be used for both classification and regression tasks.
* **No Need for Feature Scaling**: Decision trees do not require normalization or scaling of the data.
* **Handles Non-linear Relationships**: Capable of capturing non-linear relationships between features and target variables.

## Disadvantages of Decision Trees

* **Overfitting**: Decision trees can easily overfit the training data, especially if they are deep with many nodes.
* **Instability**: Small variations in the data can result in a completely different tree being generated.
* **Bias towards Features with More Levels**: Features with more levels can dominate the tree structure

## Pruning

To overcome **overfitting, pruning**techniques are used. Pruning reduces the size of the tree by removing nodes that provide little power in classifying instances. There are two main types of pruning:

* **Pre-pruning (Early Stopping)**: Stops the tree from growing once it meets certain criteria (e.g., maximum depth, minimum number of samples per leaf).
* **Post-pruning**: Removes branches from a fully grown tree that do not provide significant power.

## Applications of Decision Trees

* **Business Decision Making**: Used in strategic planning and resource allocation.
* **Healthcare**: Assists in diagnosing diseases and suggesting treatment plans.
* **Finance**: Helps in credit scoring and risk assessment.
* **Marketing**: Used to segment customers and predict customer behavior.

**12. Describe in depth the decision tree algorithm.**

# Decision Tree Classification Algorithm

* Decision Tree is a **Supervised learning technique**that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where**internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
* In a Decision tree, there are two nodes, which are the **Decision Node** and**Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
* The decisions or the test are performed on the basis of features of the given dataset.
* **It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.**
* It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
* In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
* A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
* Below diagram explains the general structure of a decision tree:

#### Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.



## Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

* Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
* The logic behind the decision tree can be easily understood because it shows a tree-like structure.

## Decision Tree Terminologies

 **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

 **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

 **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

 **Branch/Sub Tree:** A tree formed by splitting the tree.

 **Pruning:** Pruning is the process of removing the unwanted branches from the tree.

 **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

**How does the Decision Tree algorithm Work?**

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below 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.

**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:



## Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

* **Information Gain**
* **Gini Index**

### 1. Information Gain:

* Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
* It calculates how much information a feature provides us about a class.
* According to the value of information gain, we split the node and build the decision tree.
* A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

1. Information Gain= Entropy(S)- [(Weighted Avg) \*Entropy(each feature)

**Entropy:** Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

**Where,**

* **S= Total number of samples**
* **P(yes)= probability of yes**
* **P(no)= probability of no**

### 2. Gini Index:

* Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
* An attribute with the low Gini index should be preferred as compared to the high Gini index.
* It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
* Gini index can be calculated using the below formula:

Gini Index= 1- ∑jPj2

## Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning**technology used:

* **Cost Complexity Pruning**
* **Reduced Error Pruning.**

## Advantages of the Decision Tree

* It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
* It can be very useful for solving decision-related problems.
* It helps to think about all the possible outcomes for a problem.
* There is less requirement of data cleaning compared to other algorithms.

## Disadvantages of the Decision Tree

* The decision tree contains lots of layers, which makes it complex.
* It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
* For more class labels, the computational complexity of the decision tree may increase.

## Python Implementation of Decision Tree

Now we will implement the Decision tree using Python. For this, we will use the dataset "**user\_data.csv**," which we have used in previous classification models. By using the same dataset, we can compare the Decision tree classifier with other classification models such as [KNN](https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning) [SVM,](https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm) [LogisticRegression,](https://www.javatpoint.com/logistic-regression-in-machine-learning) etc.

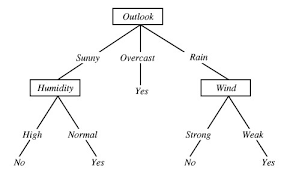
Steps will also remain the same, which are given below:

* **Data Pre-processing step**
* **Fitting a Decision-Tree algorithm to the Training set**
* **Predicting the test result**
* **Test accuracy of the result(Creation of Confusion matrix)**
* **Visualizing the test set result.**

**13. In a decision tree, what is inductive bias? What would you do to stop overfitting?**

The inductive bias of decision trees is the assumption or preference that the algorithm makes when creating the tree structure.

One common inductive bias of trees is that they tend to favor shorter trees with simpler decision rules. This helps in reducing the complexity of the model and often leads to better generalization performance on unseen data.



In the case of decision trees, **the depth of the trees** is the inductive bias. If the depth of the tree is too low, then there is too much generalization in the model.

**14.Explain advantages and disadvantages of using a decision tree?**

* **Interpretability**: Decision Trees are relatively simple to understand and interpret, making them desirable for collaborative decision-making and explaining results to non-technically oriented stakeholders.
* **Deals with Unbalanced Data**: This method is highly competent at handling diverse datasets and doesn't require balanced data to generate a robust model.
* **Variable Selection**: Decision Trees can identify the most significant variables and the relation between two or more variables, serving as a worthwhile tool for data exploration.
* **Handles Missing Values**: They have the ability to handle missing values in the dataset by looking at the probability of observing the various classes.
* **Non-parametric Nature**: They are a non-parametric method, meaning no assumptions about the space distribution and the classifier structure are made, which keeps the model simple and less prone to significant errors.

**15. Describe in depth the problems that are suitable for decision tree learning.**

* **Pruning**: This technique is commonly used to overcome overfitting. Pruning involves cutting off the branches of the tree that are contributing to overfitting, creating a simpler, more generalized model.
* **Ensemble Methods**: These can help mitigate the impact of minor variations in input data. Approaches such as bagging (Bootstrap Aggregation), boosting, and random forests involve creating multiple decision trees and combining their output.
* **Addressing Class Imbalance**: Techniques like using weighted decision trees or oversampling minority classes can help to reduce learning bias.

**16. Describe in depth the random forest model. What distinguishes a random forest?**

| **Property** | **Random Forest** | **Decision Tree** |
| --- | --- | --- |
| **Nature** | Ensemble of multiple decision trees | Single Decision Tree |
| **Interpretability** | Less interpretable due to ensemble nature. | Highly interpretable. |
| **Overfitting** | Due to ensemble averaging it is less prone to overfitting. | More prone to overfitting specially in case of deep trees. |
| **Training Time** | Since multiple trees are constructed, training time becomes more, and training speed becomes less. | A single tree needs to be built and trained, hence faster in comparison. |
| **Stability to change** | Since overall average is taken due to ensemble, it is more stable to change. | It becomes quite sensitive to variation in data. |
| **Predictive Time** | Multiple predictions, hence longer prediction time and slower prediction speed. | Faster prediction as compared to random forest, since a single prediction is made. |
| **Performance** | Generally performs well on large datasets. | It can perform well on small and large dataset as well. |
| **Handling Outliers** | Due to ensemble averaging more robust to outliers. | It is more susceptible to outliers. |
| **Feature Importance** | Do not provide feature score directly rather uses ensemble to decide feature score. | Provide feature score directly which are less reliable. |

**17. In a random forest, talk about OOB error and variable value.**

# OOB Errors for Random Forests in Scikit Learn

A random forest is an ensemble machine-learning model that is composed of multiple decision trees. A decision tree is a model that makes predictions by learning a series of simple decision rules based on the features of the data. A random forest combines the predictions of multiple decision trees to make more accurate and robust predictions.

[Random Forests](https://www.geeksforgeeks.org/random-forest-regression-in-python/)are often used for classification and regression tasks. In classification, the goal is to predict the class label (e.g., “cat” or “dog”) of each sample in the dataset. In regression, the goal is to predict a continuous target variable (e.g., the price of a house) based on the features of the data.

Random forests are popular because they are easy to train, can handle high-dimensional data, and are highly accurate. They also have the ability to handle missing values and can handle imbalanced datasets, where some classes are more prevalent than others.

To train a random forest, you need to specify the number of decision trees to use (the **n\_estimators** parameter) and the maximum depth of each tree (the **max\_depth** parameter). Other hyperparameters, such as the minimum number of samples required to split a node and the minimum number of samples required at a leaf node, can also be specified.

Once the random forest is trained, you can use it to make predictions on new data. To make a prediction, the random forest uses the predictions of the individual decision trees and combines them using a majority vote or an averaging technique.

### What is the difference between the OOB Score and the Validation score?

OOB (out-of-bag) score is a performance metric for a [machine learning](https://www.geeksforgeeks.org/machine-learning/) model, specifically for ensemble models such as random forests. It is calculated using the samples that are not used in the training of the model, which is called out-of-bag samples. These samples are used to provide an unbiased estimate of the model’s performance, which is known as the OOB score.

The validation score, on the other hand, is the performance of the model on a validation dataset. This dataset is different from the training dataset and is used to evaluate the model’s performance after it has been trained on the training dataset.

In summary, the OOB score is calculated using out-of-bag samples and is a measure of the model’s performance on unseen data. The validation score, on the other hand, is a measure of the model’s performance on a validation dataset, which is a set of samples that the model has not seen during training.

### ****OOB (out-of-bag) Errors****

OOB (out-of-bag) errors are an estimate of the performance of a random forest classifier or regressor on unseen data. In scikit-learn, the OOB error can be obtained using the oob\_score\_ attribute of the random forest classifier or regressor.

The OOB error is computed using the samples that were not included in the training of the individual trees. This is different from the error computed using the usual training and validation sets, which are used to tune the hyperparameters of the random forest.

The OOB error can be useful for evaluating the performance of the random forest on unseen data. It is not always a reliable estimate of the generalization error of the model, but it can provide a useful indication of how well the model is performing.

### Implementation of OOB Errors for Random Forests

To compute the OOB error, the samples that are not used in the training of an individual tree are known as “out-of-bag” samples. These samples are not used in the training of the tree, but they are used to compute the OOB error for that tree. The OOB error for the entire random forest is computed by averaging the OOB errors of the individual trees.