

DATA MINING (COCSC16)

UNIT 4

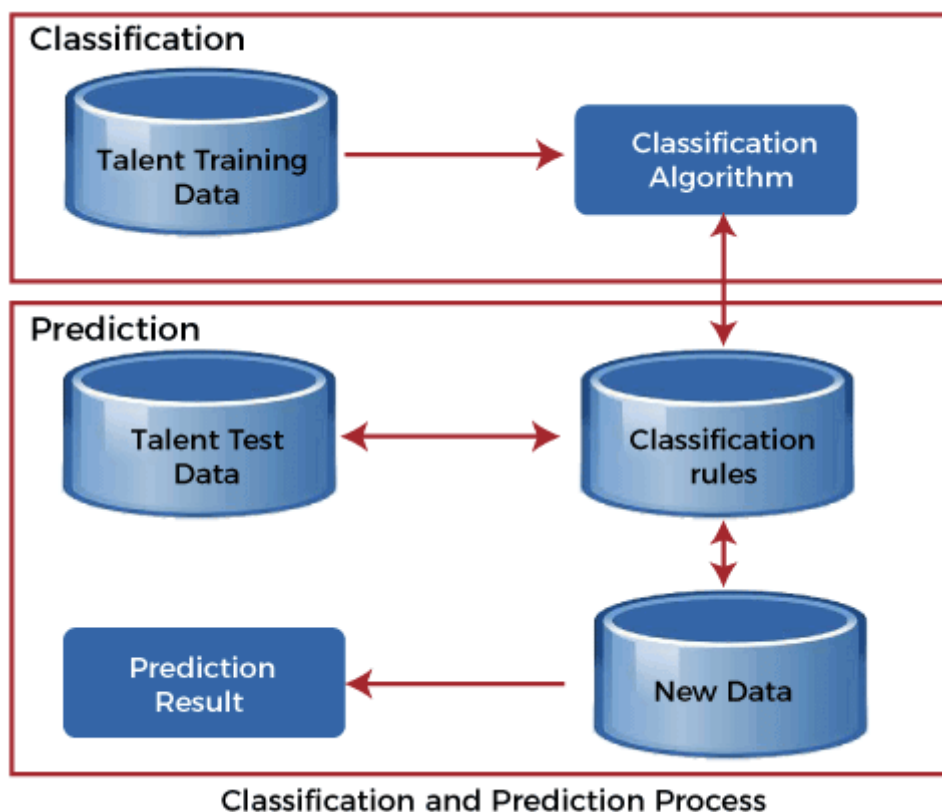
Classification and Predication in Data Mining

There are two forms of data analysis that can be used to extract models describing important classes or predict future data trends. These two forms are as follows:

1. Classification
2. Prediction

We use classification and prediction to extract a model, representing the data classes to predict future data trends. Classification predicts the categorical labels of data with the prediction models. This analysis provides us with the best understanding of the data at a large scale.

Classification models predict categorical class labels, and prediction models predict continuous-valued functions. For example, we can build a classification model to categorize bank loan applications as either safe or risky or a prediction model to predict the expenditures in dollars of potential customers on computer equipment given their income and occupation.



What is Classification?

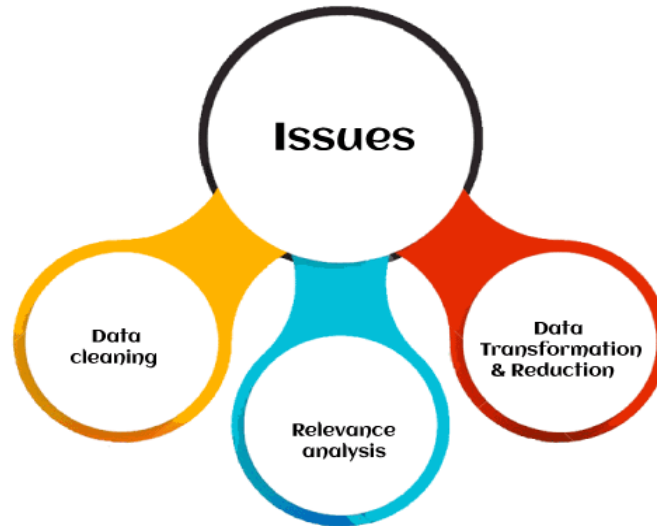
- Classification is to identify the category or the class label of a new observation. First, a set of data is used as training data. The set of input data and the corresponding outputs are given to the algorithm. So, the training data set includes the input data and their associated class labels.
- Using the training dataset, the algorithm derives a model or the classifier. The derived model can be a decision tree, mathematical formula, or a neural network. In classification, when unlabelled data is given to the model, it should find the class to which it belongs. The new data provided to the model is the test data set.
- Classification is the process of classifying a record. One simple example of classification is to check whether it is raining or not. The answer can either be yes or no. So, there is a particular number of choices. Sometimes there can be more than two classes to classify. That is called *multiclass classification*.
- The bank needs to analyze whether giving a loan to a particular customer is risky or not. **For example**, based on observable data for multiple loan borrowers, a classification model may be established that forecasts credit risk. The data could track job records, homeownership or leasing, years of residency, number, type of deposits, historical credit ranking, etc. The goal would be credit ranking, the predictors would be the other characteristics, and the data would represent a case for each consumer. In this example, a model is constructed to find the categorical label. The labels are risky or safe.

What is Prediction?

- Another process of data analysis is prediction. It is used to find a numerical output. Same as in classification, the training dataset contains the inputs and corresponding numerical output values. The algorithm derives the model or a predictor according to the training dataset. The model should find a numerical output when the new data is given.
- Unlike in classification, this method does not have a class label. The model predicts a continuous-valued function or ordered value.
- Regression is generally used for prediction. Predicting the value of a house depending on the facts such as the number of rooms, the total area, etc., is an example for prediction.
- For example, suppose the marketing manager needs to predict how much a particular customer will spend at his company during a sale. We are bothered to forecast a numerical value in this case. Therefore, an example of numeric prediction is the data processing activity. In this case, a model or a predictor will be developed that forecasts a continuous or ordered value function.

Classification and Prediction Issues

The major issue is preparing the data for Classification and Prediction. Preparing the data involves the following activities, such as:



1. **Data Cleaning:** Data cleaning involves removing the noise and treatment of missing values. The noise is removed by applying smoothing techniques, and the problem of missing values is solved by replacing a missing value with the most commonly occurring value for that attribute.
2. **Relevance Analysis:** The database may also have irrelevant attributes. Correlation analysis is used to know whether any two given attributes are related.
3. **Data Transformation and reduction:** The data can be transformed by any of the following methods.
 - **Normalization:** The data is transformed using normalization. Normalization involves scaling all values for a given attribute to make them fall within a small specified range. Normalization is used when the neural networks or the methods involving measurements are used in the learning step.
 - **Generalization:** The data can also be transformed by generalizing it to the higher concept. For this purpose, we can use the concept hierarchies.

Comparison of Classification and Prediction Methods

Here are the criteria for comparing the methods of Classification and Prediction, such as:

- **Accuracy:** The accuracy of the classifier can be referred to as the ability of the classifier to predict the class label correctly, and the accuracy of the predictor can be referred to as how well a given predictor can estimate the unknown value.
- **Speed:** The speed of the method depends on the computational cost of generating and using the classifier or predictor.

- **Robustness:** Robustness is the ability to make correct predictions or classifications. In the context of data mining, robustness is the ability of the classifier or predictor to make correct predictions from incoming unknown data.
- **Scalability:** Scalability refers to an increase or decrease in the performance of the classifier or predictor based on the given data.
- **Interpretability:** Interpretability is how readily we can understand the reasoning behind predictions or classification made by the predictor or classifier.

Difference between Classification and Prediction

- The decision tree, applied to existing data, is a classification model. We can get a class prediction by applying it to new data for which the class is unknown. The assumption is that the new data comes from a distribution similar to the data we used to construct our decision tree.
- In many instances, this is a correct assumption, so we can use the decision tree to build a predictive model.
- Classification of prediction is the process of finding a model that describes the classes or concepts of information.
- The purpose is to predict the class of objects whose class label is unknown using this model.

Below are some major differences between classification and prediction.

Classification	Prediction
Classification is the process of identifying which category a new observation belongs to based on a training data set containing observations whose category membership is known.	Prediction is the process of identifying the missing or unavailable numerical data for a new observation.
In classification, the accuracy depends on finding the class label correctly.	In prediction, the accuracy depends on how well a given predictor can guess the value of a predicted attribute for new data.
In classification, the model can be known as the classifier.	In prediction, the model can be known as the predictor.
A model or the classifier is constructed to find the categorical labels.	A model or a predictor will be constructed that predicts a continuous-valued function or ordered value.

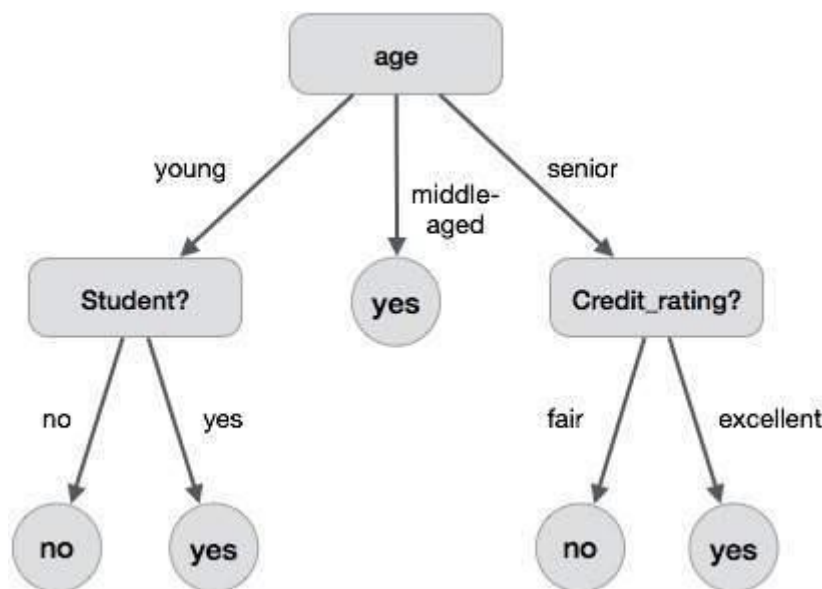
For example, the grouping of patients based on their medical records can be considered a classification.

For example, we can think of prediction as predicting the correct treatment for a particular disease for a person.

Decision Tree Induction

- A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.

The following decision tree is for the concept `buy_computer` that indicates whether a customer at a company is likely to buy a computer or not. Each internal node represents a test on an attribute. Each leaf node represents a class.



The benefits of having a decision tree are as follows –

- It does not require any domain knowledge.
- It is easy to comprehend.
- The learning and classification steps of a decision tree are simple and fast.

Decision Tree Induction Algorithm

A machine researcher named J. Ross Quinlan in 1980 developed a decision tree algorithm known as ID3 (Iterative Dichotomiser). Later, he presented C4.5, which was the successor of ID3. ID3 and C4.5 adopt a greedy approach. In this algorithm, there is no backtracking; the trees are constructed in a top-down recursive divide-and-conquer manner.

Generating a decision tree from 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 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 D_j be the set of data tuples in D satisfying outcome j; // a partition

 if D_j 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(D_j, attribute_list) to node N;

```
end for  
return N;
```

Tree Pruning

Tree pruning is performed in order to remove anomalies in the training data due to noise or outliers. The pruned trees are smaller and less complex.

Tree Pruning Approaches

There are two approaches to prune a tree –

- **Pre-pruning** – The tree is pruned by halting its construction early.
- **Post-pruning** - This approach removes a sub-tree from a fully grown tree.

Cost Complexity

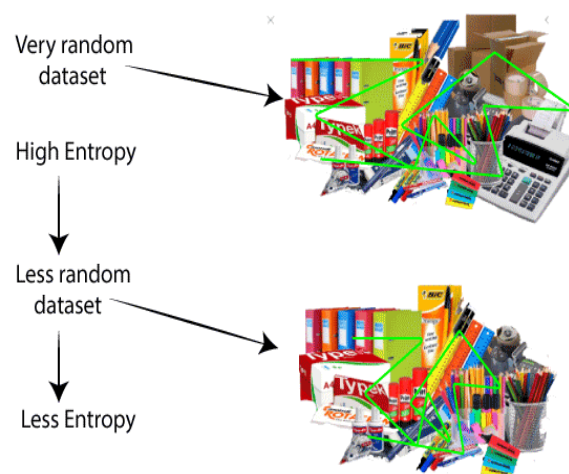
The cost complexity is measured by the following two parameters –

- Number of leaves in the tree, and
- Error rate of the tree.

Key factors:

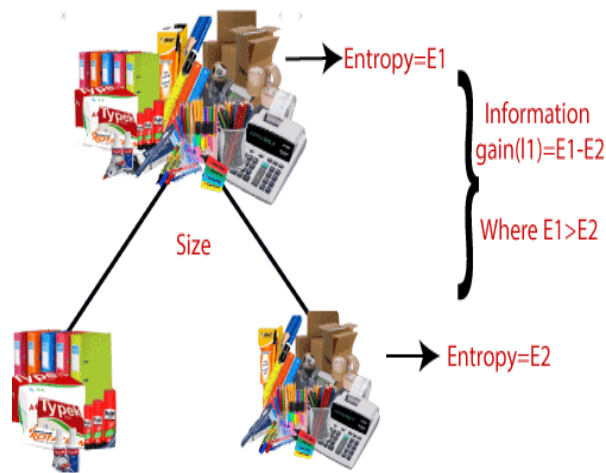
Entropy:

Entropy refers to a common way to measure impurity. In the decision tree, it measures the randomness or impurity in data sets.



Information Gain:

Information Gain refers to the decline in entropy after the dataset is split. It is also called **Entropy Reduction**. Building a decision tree is all about discovering attributes that return the highest data gain.



In short, a decision tree is just like a flow chart diagram with the terminal nodes showing decisions. Starting with the dataset, we can measure the entropy to find a way to segment the set until the data belongs to the same class.

Why are decision trees useful?

- It enables us to analyze the possible consequences of a decision thoroughly.
- It provides us a framework to measure the values of outcomes and the probability of accomplishing them.
- It helps us to make the best decisions based on existing data and best speculations.
- ✓ **In other words**, we can say that a decision tree is a hierarchical tree structure that can be used to split an extensive collection of records into smaller sets of the class by implementing a sequence of simple decision rules.
- ✓ A decision tree model comprises a set of rules for portioning a huge heterogeneous population into smaller, more homogeneous, or mutually exclusive classes.
- ✓ The attributes of the classes can be any variables from nominal, ordinal, binary, and quantitative values, in contrast, the classes must be a qualitative type, such as categorical or ordinal or binary.
- ✓ In brief, the given data of attributes together with its class, a decision tree creates a set of rules that can be used to identify the class. One rule is implemented after another, resulting in a hierarchy of segments within a segment. The hierarchy is known as the **tree**, and each segment is called a **node**.
- ✓ With each progressive division, the members from the subsequent sets become more and more similar to each other. Hence, the algorithm used to build a decision tree is

referred to as recursive partitioning. The algorithm is known as **CART** (Classification and Regression Trees)

Advantages of using decision trees:

- A decision tree does not need scaling of information.
- Missing values in data also do not influence the process of building a choice tree to any considerable extent.
- A decision tree model is automatic and simple to explain to the technical team as well as stakeholders.
- Compared to other algorithms, decision trees need less exertion for data preparation during pre-processing.
- A decision tree does not require a standardization of data.

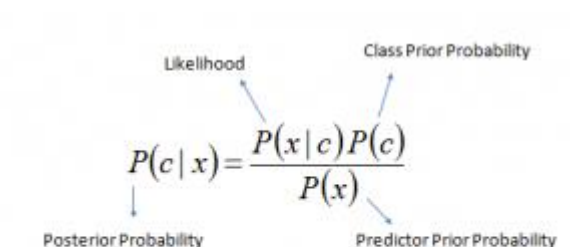
Naive Bayes Algorithm

- It is a classification technique based on Bayes' Theorem with an independence assumption among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.
- The Naïve Bayes classifier is a popular supervised machine learning algorithm used for classification tasks such as text classification. It belongs to the family of generative learning algorithms, which means that it models the distribution of inputs for a given class or category.
- This approach is based on the assumption that the features of the input data are conditionally independent given the class, allowing the algorithm to make predictions quickly and accurately.
- In statistics, naive Bayes classifiers are considered as simple probabilistic classifiers that apply Bayes' theorem. This theorem is based on the probability of a hypothesis, given the data and some prior knowledge.
- The naive Bayes classifier assumes that all features in the input data are independent of each other, which is often not true in real-world scenarios.
- However, despite this simplifying assumption, the naive Bayes classifier is widely used because of its efficiency and good performance in many real-world applications.
- Moreover, it is worth noting that naive Bayes classifiers are among the simplest Bayesian network models, yet they can achieve high accuracy levels when coupled with kernel density estimation. This technique involves using a kernel function to estimate the probability density function of the input data, allowing the classifier to improve its performance in complex scenarios where the data distribution is not well-defined.
- As a result, the naive Bayes classifier is a powerful tool in machine learning, particularly in text classification, spam filtering, and sentiment analysis, among others.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’.

An NB model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of computing posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$. Look at the equation below:


$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$

Above,

- $P(c|x)$ is the posterior probability of *class* (c , *target*) given *predictor* (x , *attributes*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of the *predictor* given *class*.
- $P(x)$ is the prior probability of the *predictor*.

How Do Naive Bayes Algorithms Work?

Below we have a training data set of weather and corresponding target variable ‘Play’ (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let’s follow the below steps to perform it.

1. Convert the data set into a frequency table

In this first step data set is converted into a frequency table

2. Create Likelihood table by finding the probabilities

Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
All	5	9
	=5/14	=9/14
	0.36	0.64

3. Use Naive Bayesian equation to calculate the posterior probability

Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of the prediction.

Problem: Players will play if the weather is sunny. Is this statement correct?

We can solve it using the above-discussed method of posterior probability.

$$P(\text{Yes} \mid \text{Sunny}) = P(\text{Sunny} \mid \text{Yes}) * P(\text{Yes}) / P(\text{Sunny})$$

Here $P(\text{Sunny} \mid \text{Yes}) * P(\text{Yes})$ is in the numerator, and $P(\text{Sunny})$ is in the denominator.

Here we have $P(\text{Sunny} \mid \text{Yes}) = 3/9 = 0.33$, $P(\text{Sunny}) = 5/14 = 0.36$, $P(\text{Yes}) = 9/14 = 0.64$

Now, $P(\text{Yes} \mid \text{Sunny}) = 0.33 * 0.64 / 0.36 = 0.60$, which has higher probability.

The Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification (nlp) and with problems having multiple classes.

What Are the Pros and Cons of Naive Bayes

Pros:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, the classifier performs better compared to other machine learning models like logistic regression or decision tree, and requires less training data.

- It performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side, Naive Bayes is also known as a bad estimator.
- Another limitation of this algorithm is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Applications of Naive Bayes Algorithms

- **Real-time Prediction:** Naive Bayesian classifier is an eager learning classifier and it is super-fast. Thus, it could be used for making predictions in real time.
- **Multi-class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayesian classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not.

Ensemble learning

- Ensemble learning gives credence to the idea of the “wisdom of crowds,” which suggests that the decision-making of a larger group of people is typically better than that of an individual expert.
- Similarly, ensemble learning refers to a group (or ensemble) of base learners, or models, which work collectively to achieve a better final prediction. A single model, also known as a base or weak learner, may not perform well individually due to high variance or high bias.
- However, when weak learners are aggregated, they can form a strong learner, as their combination reduces bias or variance, yielding better model performance.

- Ensemble methods are frequently illustrated using decision trees as this algorithm can be prone to overfitting (high variance and low bias) when it hasn't been pruned and it can also lend itself to underfitting (low variance and high bias) when it's very small, like a decision stump, which is a decision tree with one level.
- When an algorithm overfits or underfits to its training set, it cannot generalize well to new datasets, so ensemble methods are used to counteract this behavior to allow for generalization of the model to new datasets.
- While decision trees can exhibit high variance or high bias, it's worth noting that it is not the only modeling technique that leverages ensemble learning to find the "sweet spot" within the bias-variance tradeoff.

There are two techniques given below that are used to perform ensemble decision tree.

Bagging

- Bagging is used when our objective is to reduce the variance of a decision tree. Here the concept is to create a few subsets of data from the training sample, which is chosen randomly with replacement.
- Now each collection of subset data is used to prepare their decision trees thus, we end up with an ensemble of various models. The average of all the assumptions from numerous trees is used, which is more powerful than a single decision tree.
- 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.
- After several data samples are generated, these weak models are then trained independently, and depending on the type of task—regression or classification, for example—the average or majority of those predictions yield a more accurate estimate.
- As a note, the random forest algorithm is considered an extension of the bagging method, using both bagging and feature randomness to create an uncorrelated forest of decision trees.
- **Random Forest** is an expansion over bagging. It takes one additional step to predict a random subset of data. It also makes the random selection of features rather than using all features to develop trees. When we have numerous random trees, it is called the Random Forest.

These are the following steps which are taken to implement a Random Forest:

- Let us consider **X** observations **Y** features in the training data set. First, a model from the training data set is taken randomly with substitution.
- The tree is developed to the largest.
- The given steps are repeated, and prediction is given, which is based on the collection of predictions from n number of trees.

Advantages of using Random Forest technique:

- It manages a higher dimension data set very well.
- It manages missing quantities and keeps accuracy for missing data.

Disadvantages of using Random Forest technique:

Since the last prediction depends on the mean predictions from subset trees, it won't give precise value for the regression model.

How bagging works

In 1996, Leo Breiman introduced the bagging algorithm, which has three basic steps:

1. **Bootstrapping:** Bagging leverages a bootstrapping sampling technique to create diverse samples. This resampling method generates different subsets of the training dataset by selecting data points at random and with replacement. This means that each time you select a data point from the training dataset, you are able to select the same instance multiple times. As a result, a value/instance repeated twice (or more) in a sample.
2. **Parallel training:** These bootstrap samples are then trained independently and in parallel with each other using weak or base learners.
3. **Aggregation:** Finally, depending on the task (i.e., regression or classification), an average or a majority of the predictions are taken to compute a more accurate estimate. In the case of regression, an average is taken of all the outputs predicted by the individual classifiers; this is known as soft voting. For classification problems, the class with the highest majority of votes is accepted; this is known as hard voting or majority voting.

Benefits and challenges of bagging

There are a number of key advantages and challenges that the bagging method presents when used for classification or regression problems. The key benefits of bagging include:

- **Ease of implementation:** Python libraries such as scikit-learn (also known as sklearn) make it easy to combine the predictions of base learners or estimators to improve model performance.
- **Reduction of variance:** Bagging can reduce the variance within a learning algorithm. This is particularly helpful with high-dimensional data, where missing values can lead to higher variance, making it more prone to overfitting and preventing accurate generalization to new datasets.

The key challenges of bagging include:

- **Loss of interpretability:** It's difficult to draw very precise business insights through bagging because due to the averaging involved across predictions. While the output is more precise than any individual data point, a more accurate or complete dataset could also yield more precision within a single classification or regression model.
- **Computationally expensive:** Bagging slows down and grows more intensive as the number of iterations increase. Thus, it's not well-suited for real-time applications. Clustered systems or a large number of processing cores are ideal for quickly creating bagged ensembles on large test sets.
- **Less flexible:** As a technique, bagging works particularly well with algorithms that are less stable. One that are more stable or subject to high amounts of bias do not provide as much benefit as there's less variation within the dataset of the model.

Applications of Bagging

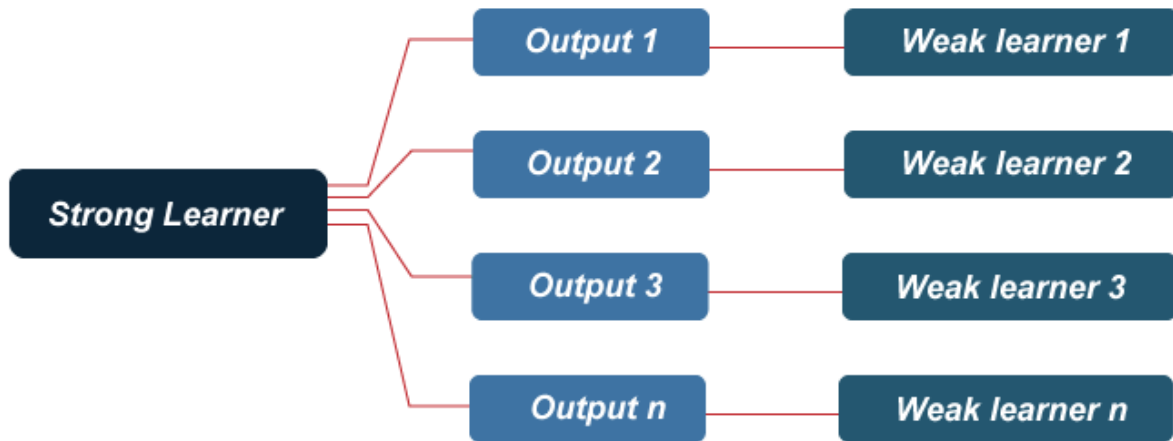
The bagging technique is used across a large number of industries, providing insights for both real-world value and interesting perspectives, Key use cases include:

- **Healthcare:** Bagging has been used to form medical data predictions. For example, research shows that ensemble methods have been used for an array of bioinformatics problems, such as gene and/or protein selection to identify a specific trait of interest.
- **IT:** Bagging can also improve the precision and accuracy in IT systems, such as one's network intrusion detection systems. Meanwhile, research looks at how bagging can improve the accuracy of network intrusion detection—and reduce the rates of false positives.
- **Environment:** Ensemble methods, such as bagging, have been applied within the field of remote sensing. More specifically, research shows how it has been used to map the types of wetlands within a coastal landscape.
- **Finance:** Bagging has also been leveraged with deep learning models in the finance industry, automating critical tasks, including fraud detection, credit risk evaluations, and option pricing problems. This research demonstrates how bagging among other machine learning techniques have been leveraged to assess loan default risk. This study highlights how bagging helps to minimize risk by to prevent credit card fraud within banking and financial institutions.

What is Boosting in Data Mining?

- Boosting is an ensemble learning method that combines a set of weak learners into strong learners to minimize training errors. In boosting, a random sample of data is selected, fitted with a model, and then trained sequentially.
- That is, each model tries to compensate for the weaknesses of its predecessor. Each classifier's weak rules are combined with each iteration to form one strict prediction rule.

- Boosting is an efficient algorithm that converts a weak learner into a strong learner. They use the concept of the weak learner and strong learner conversation through the weighted average values and higher votes values for prediction. These algorithms use decision stamp and margin maximizing classification for processing.



There are three types of Algorithms available such as AdaBoost or Adaptive boosting Algorithm, Gradient, and XG Boosting algorithm.

Let's understand this concept with the help of the following example. Let's take the example of the email. How will you recognize your email, whether it is spam or not? You can recognize it by the following conditions:

- If an email contains lots of sources, that means it is spam.
- If an email contains only one file image, then it is spam.
- If an email contains the message "You Own a lottery of \$xxxxx," it is spam.
- If an email contains some known source, then it is not spam.
- If it contains the official domain like educba.com, etc., it is not spam.

The rules mentioned above are not that powerful to recognize spam or not; hence these rules are called ***weak learners***.

To convert weak learners to the strong learner, combine the prediction of the weak learner using the following methods.

1. Using average or weighted average.
2. Consider prediction has a higher vote.

Consider the 5 rules mentioned above; there are 3 votes for spam and 2 votes for not spam. Since there is high vote spam, we consider it spam.

Why is Boosting Used?

- To solve complicated problems, we require more advanced techniques. Suppose that, given a data set of images containing images of cats and dogs, you were asked to build a model that can classify these images into two separate classes.
- Like every other person, you will start by identifying the images by using some rules given below:
 1. The image has pointy ears: Cat
 2. The image has cat-shaped eyes: Cat
 3. The image has bigger limbs: Dog
 4. The image has sharpened claws: Cat
 5. The image has a wider mouth structure: Dog

These rules help us identify whether an image is a Dog or a cat. However, the prediction would be flawed if we were to classify an image based on an individual (single) rule. These rules are called weak learners because these rules are not strong enough to classify an image as a cat or dog.

Therefore, to ensure our prediction is more accurate, we can combine the prediction from these weak learners by using the majority rule or weighted average. This makes a strong learner model.

In the above example, we have defined 5 weak learners, and the majority of these rules (i.e., 3 out of 5 learners predict the image as a cat) give us the prediction that the image is a cat. Therefore, our final output is a cat.

How does the Boosting Algorithm Work?

- The basic principle behind the working of the boosting algorithm is to generate multiple weak learners and combine their predictions to form one strict rule.
- These weak rules are generated by applying base Machine Learning algorithms on different distributions of the data set.
- These algorithms generate weak rules for each iteration. After multiple iterations, the weak learners are combined to form a strong learner that will predict a more accurate outcome.



Here's how the algorithm works:

Step 1: The base algorithm reads the data and assigns equal weight to each sample observation.

Step 2: False predictions made by the base learner are identified. In the next iteration, these false predictions are assigned to the next base learner with a higher weightage on these incorrect predictions.

Step 3: Repeat step 2 until the algorithm can correctly classify the output.

Therefore, the main aim of Boosting is to focus more on miss-classified predictions.

Types of Boosting

- Boosting methods are focused on iteratively combining weak learners to build a strong learner that can predict more accurate outcomes. As a reminder, a weak learner classifies data slightly better than random guessing.
- This approach can provide robust prediction problem results, outperform neural networks, and support vector machines for tasks.

Boosting algorithms can differ in how they create and aggregate weak learners during the sequential process. Three popular types of boosting methods include:

1. Adaptive boosting or AdaBoost:

- This method operates iteratively, identifying misclassified data points and adjusting their weights to minimize the training error. The model continues to optimize sequentially until it yields the strongest predictor.
- AdaBoost is implemented by combining several weak learners into a single strong learner. The weak learners in AdaBoost take into account a single input feature and draw out a single split decision tree called the decision stump. Each observation is weighted equally while drawing out the first decision stump.

- The results from the first decision stump are analyzed, and if any observations are wrongfully classified, they are assigned higher weights.
- A new decision stump is drawn by considering the higher-weight observations as more significant.
- Again, if any observations are misclassified, they're given a higher weight, and this process continues until all the observations fall into the right class.
- AdaBoost can be used for both classification and regression-based problems. However, it is more commonly used for classification purposes.

2. Gradient Boosting:

- Gradient Boosting is also based on sequential ensemble learning. Here the base learners are generated sequentially so that the present base learner is always more effective than the previous one, i.e., and the overall model improves sequentially with each iteration.
- The difference in this boosting type is that the weights for misclassified outcomes are not incremented. Instead, the Gradient Boosting method tries to optimize the loss function of the previous learner by adding a new model that adds weak learners to reduce the loss function.

The main idea here is to overcome the errors in the previous learner's predictions. This boosting has three main components:

- **Loss function:** The use of the loss function depends on the type of problem. The advantage of gradient boosting is that there is no need for a new boosting algorithm for each loss function.
- **Weak learner:** In gradient boosting, decision trees are used as a weak learner. A regression tree is used to give true values, which can combine to create correct predictions. Like in the AdaBoost algorithm, small trees with a single split are used, i.e., decision stump. Larger trees are used for large levels, 4-8.
- **Additive Model:** Trees are added one at a time in this model. Existing trees remain the same. During the addition of trees, gradient descent is used to minimize the loss function.

Like AdaBoost, Gradient Boosting can also be used for classification and regression problems.

3. Extreme gradient boosting or XGBoost:

- XGBoost is an advanced gradient boosting method. XGBoost, developed by Tianqi Chen, falls under the Distributed Machine Learning Community (DMLC) category.
- The main aim of this algorithm is to increase the speed and efficiency of computation. The Gradient Descent Boosting algorithm computes the output slower since they sequentially analyze the data set. Therefore, XGBoost is used to boost or extremely boost the model's performance.

XGBoost is designed to focus on computational speed and model efficiency. The main features provided by XGBoost are:

- **Parallel Processing:** XG Boost provides Parallel Processing for tree construction which uses CPU cores while training.
- **Cross-Validation:** XG Boost enables users to run cross-validation of the boosting process at each iteration, making it easy to get the exact optimum number of boosting iterations in one run.
- **Cache Optimization:** It provides Cache Optimization of the algorithms for higher execution speed.
- **Distributed Computing:** For training large models, XG Boost allows Distributed Computing.

Benefits and Challenges of Boosting

The boosting method presents many advantages and challenges for classification or regression problems. The benefits of boosting include:

- **Ease of Implementation:** Boosting can be used with several hyper-parameter tuning options to improve fitting. No data preprocessing is required, and boosting algorithms have built-in routines to handle missing data. In Python, the sci-kit-learn library of ensemble methods makes it easy to implement the popular boosting methods, including AdaBoost, XGBoost, etc.
- **Reduction of bias:** Boosting algorithms combine multiple weak learners in a sequential method, iteratively improving upon observations. This approach can help to reduce high bias, commonly seen in shallow decision trees and logistic regression models.
- **Computational Efficiency:** Since boosting algorithms have special features that increase their predictive power during training, it can help reduce dimensionality and increase computational efficiency.

And the challenges of boosting include:

- **Overfitting:** There's some dispute in the research around whether or not boosting can help reduce overfitting or make it worse. We include it under challenges because in the instances that it does occur, predictions cannot be generalized to new datasets.
- **Intense computation:** Sequential training in boosting is hard to scale up. Since each estimator is built on its predecessors, boosting models can be computationally expensive, although XGBoost seeks to address scalability issues in other boosting methods. Boosting algorithms can be slower to train when compared to bagging, as a large number of parameters can also influence the model's behavior.

- **Vulnerability to outlier data:** Boosting models are vulnerable to outliers or data values that are different from the rest of the dataset. Because each model attempts to correct the faults of its predecessor, outliers can skew results significantly.
- **Real-time implementation:** You might find it challenging to use boosting for real-time implementation because the algorithm is more complex than other processes. Boosting methods have high adaptability, so you can use various model parameters that immediately affect the model's performance.

Applications of Boosting

Boosting algorithms are well suited for artificial intelligence projects across a broad range of industries, including:

- **Healthcare:** Boosting is used to lower errors in medical data predictions, such as predicting cardiovascular risk factors and cancer patient survival rates. For example, research shows that ensemble methods significantly improve the accuracy in identifying patients who could benefit from preventive treatment of cardiovascular disease while avoiding unnecessary treatment of others. Likewise, another study found that applying boosting to multiple genomics platforms can improve the prediction of cancer survival time.
- **IT:** Gradient boosted regression trees are used in search engines for page rankings, while the Viola-Jones boosting algorithm is used for image retrieval. As noted by Cornell, boosted classifiers allow the computations to be stopped sooner when it's clear which direction a prediction is headed. A search engine can stop evaluating lower-ranked pages, while image scanners will only consider images containing the desired object.
- **Finance:** Boosting is used with deep learning models to automate critical tasks, including fraud detection, pricing analysis, and more. For example, boosting methods in credit card fraud detection and financial product pricing analysis improves the accuracy of analyzing massive data sets to minimize financial losses.

Bagging vs. boosting

Bagging and boosting are two main types of ensemble learning methods, the main difference between these learning methods is the way in which they are trained. In bagging, weak learners are trained in parallel, but in boosting, they learn sequentially. This means that a series of models are constructed and with each new model iteration, the weights of the misclassified data in the previous model are increased. This redistribution of weights helps the algorithm identify the parameters that it needs to focus on to improve its performance. AdaBoost, which stands for “adaptive boosting algorithm,” is one of the most popular boosting algorithms as it was one of the first of its kind. Other types of boosting algorithms include XGBoost, GradientBoost, and BrownBoost.

Another difference in which bagging and boosting differ are the scenarios in which they are used. For example, bagging methods are typically used on weak learners which exhibit high variance and low bias, whereas boosting methods are leveraged when low variance and high bias is observed.