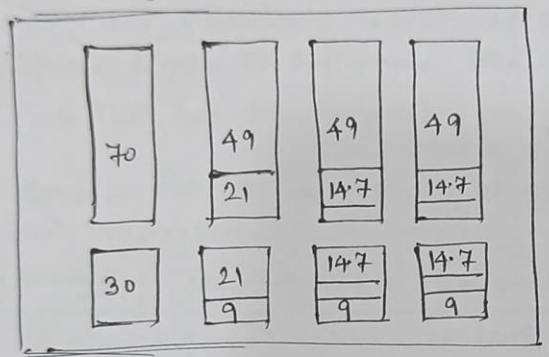
Ensemble Models

Intro: A set of Predictors with accuracy $>50\%$, can be used as an ensemble to build a predictor with higher accuracy than each individual predictor and the correlation between base predictors must be very low

Explanation: Let 3 predictors (P_1, P_2, P_3) are there for the same task. They will predict the class with respect to situation (0 or 1).
There may be one optimal predictive model P which can combine the best of the three predictors P_1, P_2, P_3.
P will predict '1' if 2 or more of the base predictors predict '1', also for '0'.



- total 100 examples, all have accuracy $= 70\%$
- P_1 predicts 70 correctly, 30 incorrectly.
- out of 70 correct Prediction by P_1, P_2 predict 70% of that $(70 \times \frac{70}{100} = 49)$
- So, these 49 will be predicted correctly by P also, regardless of P_3 does on these 49 cases.

- out of 70 correct prediction by P_1, P_2 predicts 30% of them (21) incorrectly. But of these 21, P_3 predict 70%. $(21 \times 70/100) = 14{\cdot}7$ correctly.
  ⇒ since P_1, P_3 agree on these 14·7 examples, and they are correct, P will be correct on these 14·7.
- P_1 incorrectly predicts 30 examples.
  But out of them, P_2 predict 70%. $(30 \times 70/100) = 21$ correctly.
  out of these 21, P_3 predict correctly 70%. $(21 \times 70/100)$ of it $= 14{\cdot}7$.
  ⇒ Since both P_2, P_3 agree for these 14·7 examples and they are correct, P will be correct on these 14·7

⇒ P will predict finally $(49 + 14{\cdot}7 + 14{\cdot}7) = 78{\cdot}4$ of the cases correctly which is $>78\%$ and higher than any individual predictors accuracy of 70%.

NB If the correlation between base predictors (where from ensemble is to be build) is not low, then if one predicts correctly, others also predicts correctly, if one predicts incorrectly, others also predicts incorrectly.

Example: We can build an ensemble from a logistic regression classifier, KNN classifier, a Decision Tree classifier.

Random Forest Classifier

Random Forest

- It is an ensemble of many decision trees (DT)
- Number of Decision Trees is a hyperparameter and to be tuned with Datasets.
- An ensemble of DT will have lower variance and higher accuracy than a single decision tree.
- Decision Trees should not be correlated or have low correlation.

- **Heuristics to minimize the correlation between decision trees**

⇒ Different random subsets of the features are used at different levels of the same tree as well as different trees.

→ This has the impact on de-correlating the trees.

NB Random Forest is a powerful predictors both for classification and regression.

- **How overfitting is reduced in Random Forest**

by constraining (1) minimum leaf size
(2) tree depth

HANDS ON <Bank Note classification>

Step1: Read and explore the Data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv('data_banknote_authentication.csv')
df.columns
df.columns[df.isnull().any()]
df['authentic'].value_counts()
df.sample(n=5, random_state=55).sort_values('variance')
```

// Columns: variance, skew, kurtosis, entropy, authentic (target var)

→

Step2: Build the Model using Random Forest

```
X = df.drop('authentic', axis = 1)
y = df['authentic']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
                                                    random_state = 1)
from sklearn.ensemble import RandomForestClassifier
Classifier = RandomForestClassifier(random_state = 0, min_samples_leaf=10)
Classifier.fit(X_train, y_train)
```

Step3: Evaluate the model

```
y_test_hat = Classifier.predict(X_test)
Results = pd.DataFrame({'Actual': y_test})
Column = pd.DataFrame({'Predicted': y_test_hat})
Results = Results.join(Column, set_index(Results.index))
Results.head(5)

from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_test_hat))
y_train_hat = Classifier.predict(X_train)
print(accuracy_score(y_train, y_train_hat))
```

Step4: Check the Confusion Matrix

```
from sklearn.metrics import confusion_matrix, recall_score, precision_score
cm = confusion_matrix(y_test, y_test_hat)
print(cm)

TN = cm[0][0]
FP = cm[0][1]
FN = cm[1][0]
TP = cm[1][1]
recall = TP/float(FN+TP)
print("recall:", recall)
precision = TP/float(TP+FP)
print("precision:", precision)
specificity = TN/(TN+FP)
print("specificity:", specificity)
```

→

Step5: Find out Feature Importance

→ to get the insight about the relationship between features and target variable.

```
feature_importances = pd.DataFrame(Classifier.feature_importances_,
                                   index = X_train.columns,
                                   columns = ['importance']).sort_values('importance')

feature_importance
```

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. Remember, 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 behaviour to allow for generalization of the model to new datasets.

RANDOM FOREST

A random forest is a supervised machine learning algorithm that is constructed from decision tree algorithms. This algorithm is applied in various industries such as banking and e-commerce to predict behaviour and outcomes.

This technique has very good predictive accuracy and used in BODY POSE RECOGNITION.

Definition

- A random forest is a machine learning technique that's used to solve regression and classification problems.
- **It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.**
- A random forest algorithm consists of many decision trees.
- The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.

- The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. **Increasing the number of trees increases the precision of the outcome.**
- A random forest eradicates the limitations of a decision tree algorithm. **It reduces the overfitting of datasets and increases precision.**

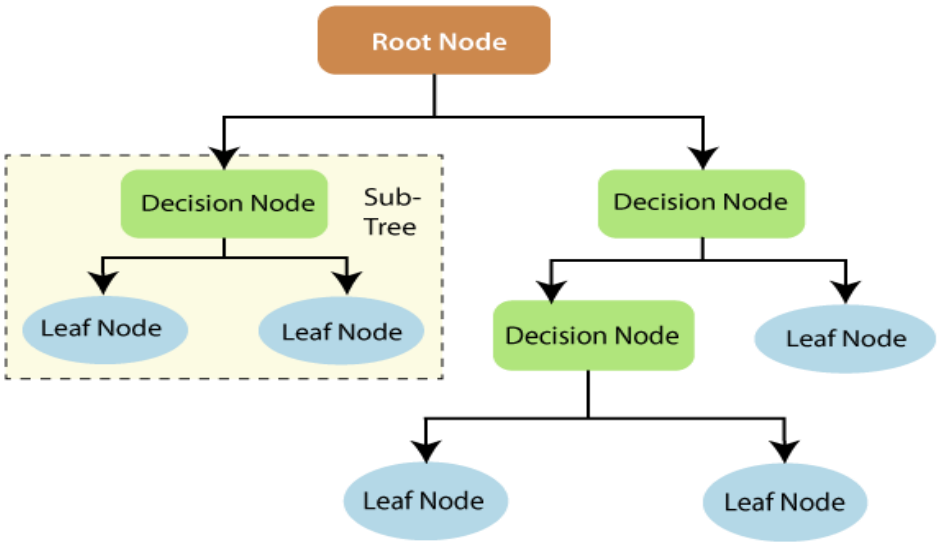
Features

- It's more accurate than the decision tree algorithm.
- It provides an effective way of handling missing data.
- It can produce a reasonable prediction without hyper-parameter tuning.
- It solves the issue of overfitting in decision trees.
- In every random forest tree, a subset of features is selected randomly at the node's splitting point.

How random forest algorithm works

Understanding decision trees

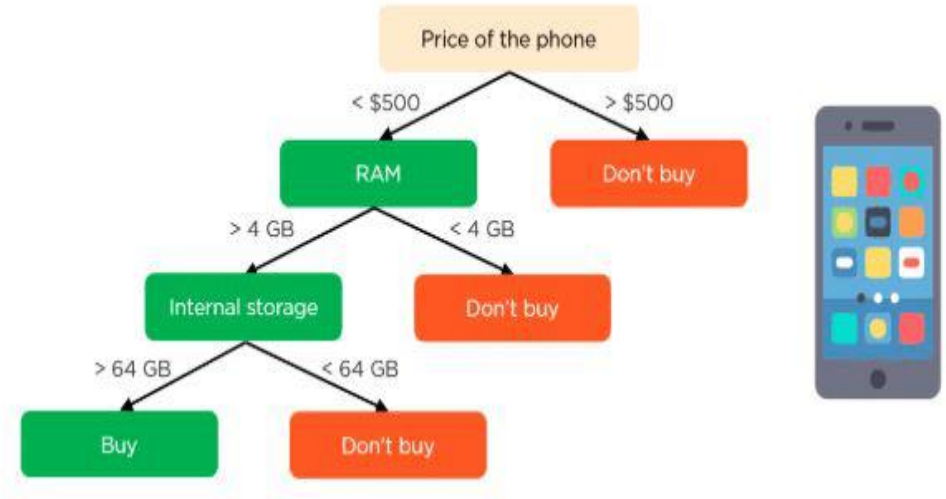
- Decision trees are the building blocks of a random forest algorithm. A decision tree is a decision support technique that forms a tree-like structure.
- A decision tree consists of three components: decision nodes, leaf nodes, and a root node. A decision tree algorithm divides a training dataset into branches, which further segregate into other branches. This sequence continues until a leaf node is attained. The leaf node cannot be segregated further.
- The nodes in the decision tree represent attributes that are used for predicting the outcome. Decision nodes provide a link to the leaves. The following diagram shows the three types of nodes in a decision tree.



- The information theory can provide more information on how decision trees work. Entropy and information gain are the building blocks of decision trees.
- Entropy is a metric for calculating uncertainty. Information gain is a measure of how uncertainty in the target variable is reduced, given a set of independent variables.
- The information gain concept involves using independent variables (features) to gain information about a target variable (class). The entropy of the target variable (Y) and the conditional entropy of Y (given X) are used to estimate the information gain. In this case, the conditional entropy is subtracted from the entropy of Y.
- Information gain is used in the training of decision trees. It helps in reducing uncertainty in these trees. A high information gain means that a high degree of uncertainty (information entropy) has been removed. Entropy and information gain are important in splitting branches, which is an important activity in the construction of decision trees.

Applying decision trees in random forest

- The main difference between the decision tree algorithm and the random forest algorithm is that establishing root nodes and segregating nodes is done randomly in the latter. The random forest employs the bagging method to generate the required prediction.
- Bagging involves using different samples of data (training data) rather than just one sample. A training dataset comprises observations and features that are used for making predictions. The decision trees produce different outputs, depending on the training data fed to the random forest algorithm. These outputs will be ranked, and the highest will be selected as the final output.
- Let's take a simple example of how a decision tree works. Suppose we want to predict if a customer will purchase a mobile phone or not. The features of the phone form the basis of his decision. This analysis can be presented in a decision tree diagram.
- The root node and decision nodes of the decision represent the features of the phone mentioned above. The leaf node represents the final output, either buying or not buying. The main features that determine the choice include the price, internal storage, and Random Access Memory (RAM). The decision tree will appear as follows.

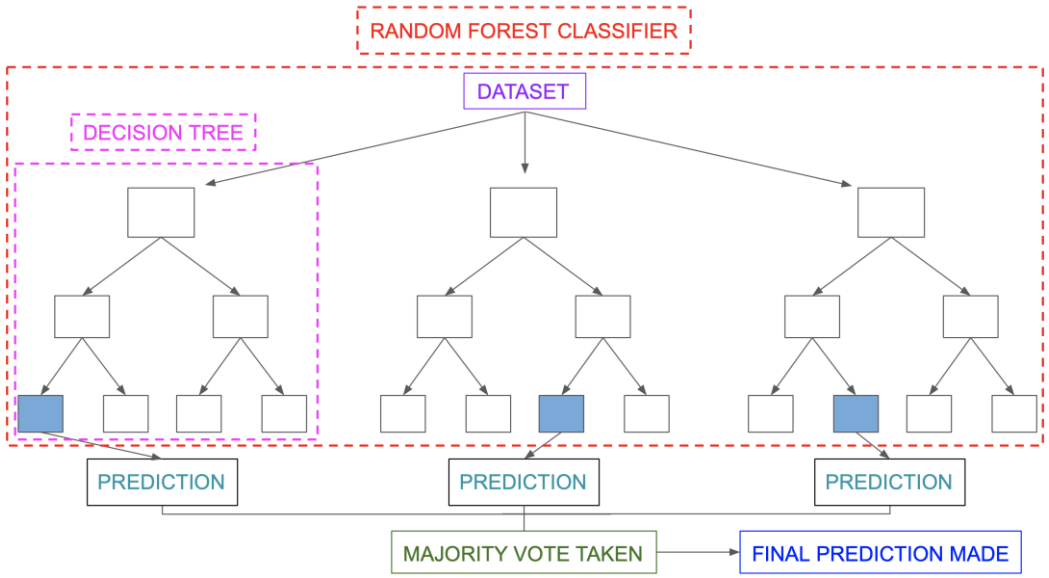


- Instead of having a single decision tree, the random forest will have many decision trees. Let's assume there are only four decision trees. In this case, the training data comprising the phone's observations and features will be divided into four root nodes.
- The root nodes could represent four features that could influence the customer's choice (price, internal storage, camera, and RAM). The random forest will split the nodes by selecting features randomly. The final prediction will be selected based on the outcome of the four trees.
- The outcome chosen by most decision trees will be the final choice. If three trees predict buying, and one tree predicts not buying, then the final prediction will be buying. In this case, it's predicted that the customer will buy the phone.

Classification in random forests

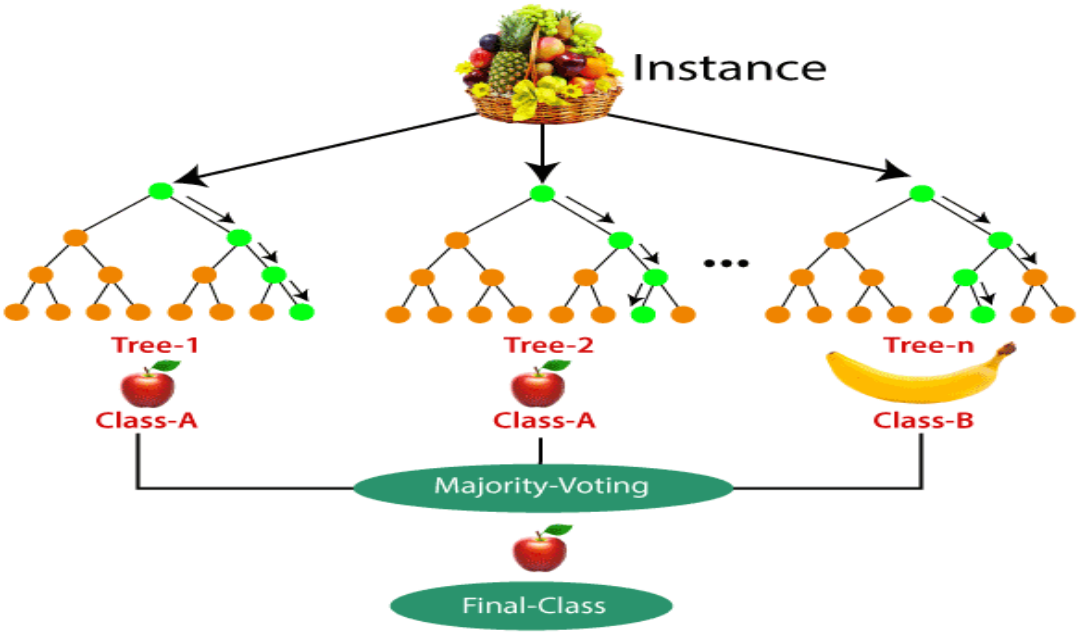
Classification in random forests employs an ensemble methodology to attain the outcome. The training data is fed to train various decision trees. This dataset consists of observations and features that will be selected randomly during the splitting of nodes.

A rain forest system relies on various decision trees. Every decision tree consists of decision nodes, leaf nodes, and a root node. The leaf node of each tree is the final output produced by that specific decision tree. **The selection of the final output follows the majority-voting system.** In this case, the output chosen by the majority of the decision trees becomes the final output of the rain forest system. The diagram below shows a simple random forest classifier.



Let's take an example of a training dataset consisting of various fruits such as bananas, apples, pineapples, and mangoes. The random forest classifier divides this dataset into subsets. These subsets are given to every decision tree in the random forest system. Each decision tree produces its specific output. For example, the prediction for trees 1 and 2 is apple.

Another decision tree (n) has predicted banana as the outcome. The random forest classifier collects the **majority voting** to provide the final prediction. The majority of the decision trees have chosen apple as their prediction. This makes the classifier choose apple as the final prediction.



Regression in random forests

Regression is the other task performed by a random forest algorithm. A random forest regression follows the concept of simple regression. Values of dependent (features) and independent variables are passed in the random forest model.

In a random forest regression, each tree produces a specific prediction. The mean prediction of the individual trees is the output of the regression. This is contrary to random forest classification, whose output is determined by the mode of the decision trees' class.

Although random forest regression and linear regression follow the same concept, they differ in terms of functions. The function of linear regression is y=bx + c, where y is the dependent variable, x is the independent variable, b is the estimation parameter, and c is a constant. The function of a complex random forest regression is like a black box.

Applications of random forest

Some of the applications of the random forest may include:

Banking

Random forest is used in banking to predict the creditworthiness of a loan applicant. This helps the lending institution make a good decision on whether to give the customer the loan or not. Banks also use the random forest algorithm to detect fraudsters.

Health care

Health professionals use random forest systems to diagnose patients. Patients are diagnosed by assessing their previous medical history. Past medical records are reviewed to establish the right dosage for the patients.

Stock market

Financial analysts use it to identify potential markets for stocks. It also enables them to identify the behaviour of stocks.

E-commerce

Through rain forest algorithms, e-commerce vendors can predict the preference of customers based on past consumption behaviour.

When to avoid using random forests

Random forest algorithms are not ideal in the following situations:

Extrapolation

Random forest regression is not ideal in the extrapolation of data. Unlike linear regression, which uses existing observations to estimate values beyond the observation range. This explains why most applications of random forest relate to classification.

Sparse data

Random forest does not produce good results when the data is very sparse (infrequent / scattered). In this case, the subset of features and the bootstrapped sample will produce an invariant space. This will lead to unproductive splits, which will affect the outcome.

Advantages of random forest

- It can perform both regression and classification tasks.
- A random forest produces good predictions that can be understood easily.
- It can handle large datasets efficiently.
- The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm.

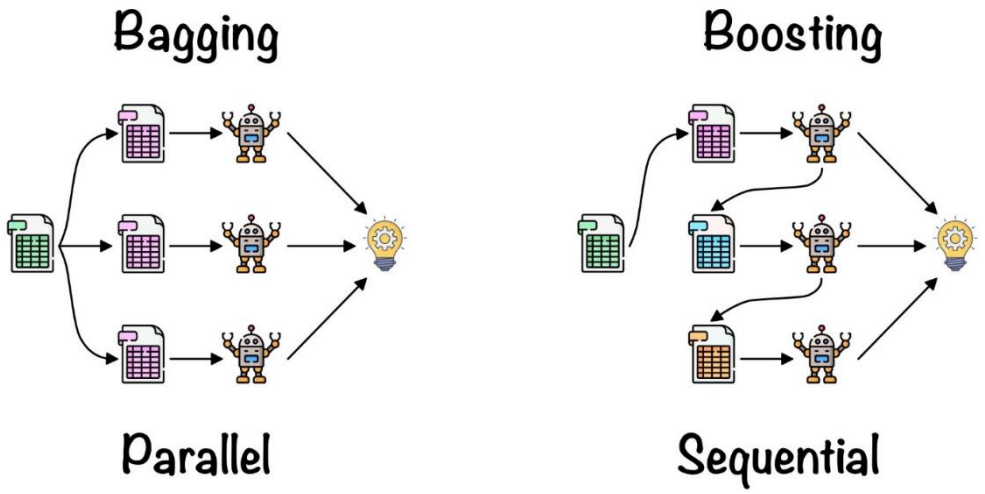
Disadvantages of random forest

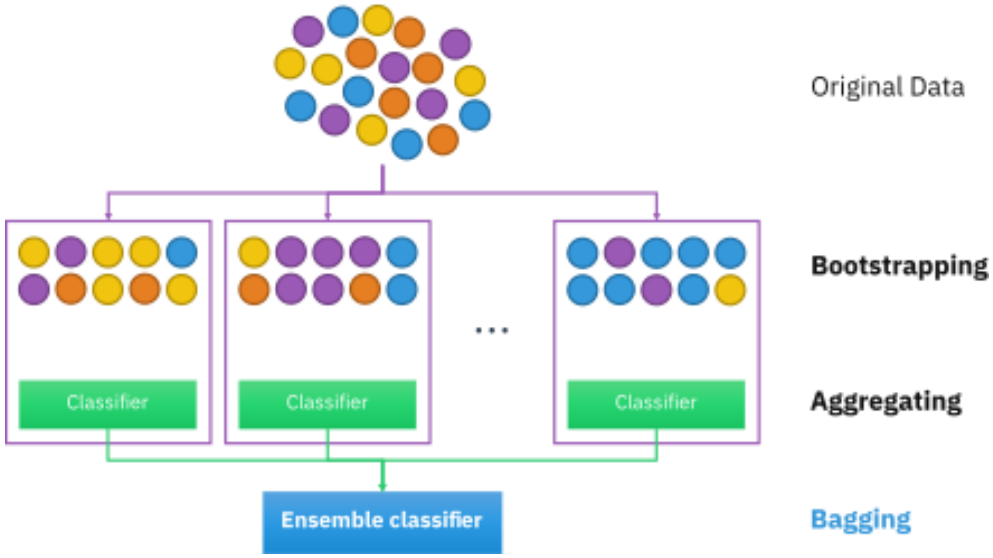
- When using a random forest, more resources are required for computation.
- It consumes more time compared to a decision tree algorithm.

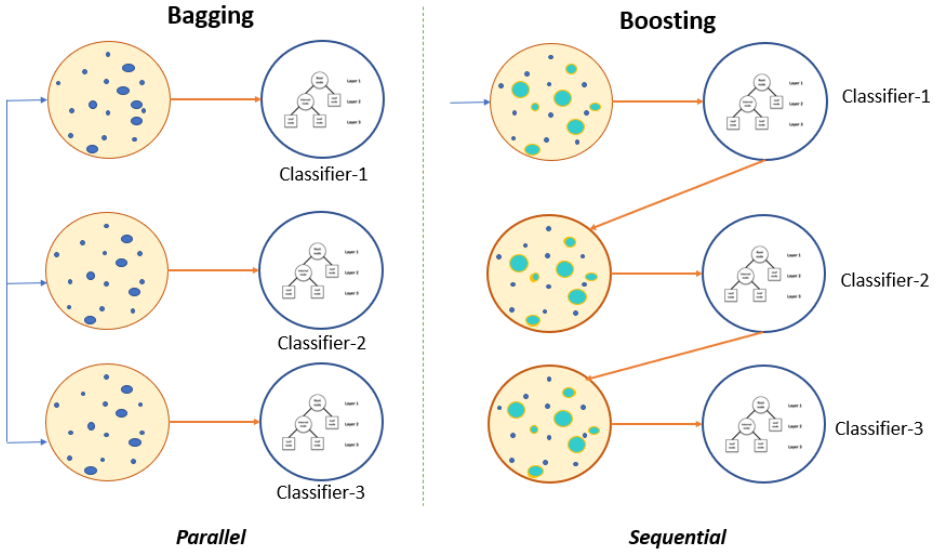
BAGGING

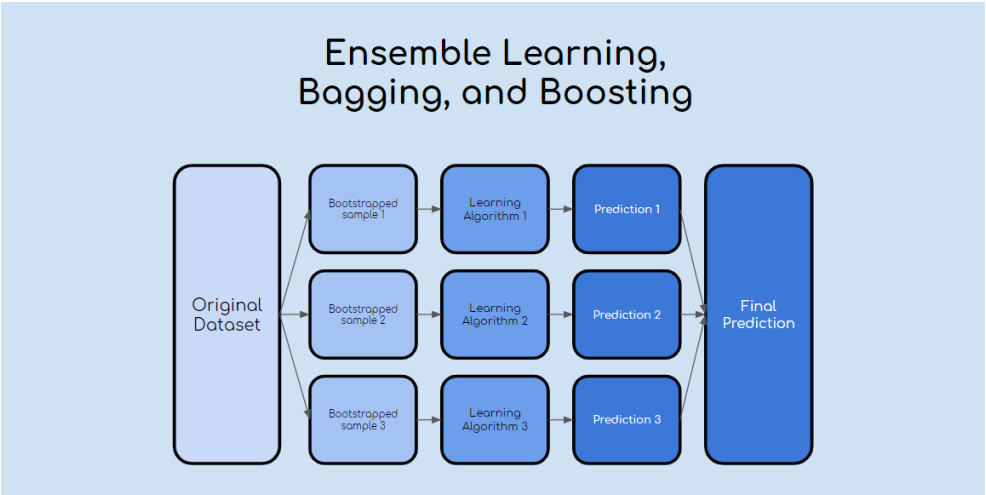
The approach under this induces a group of classifiers. When presented with an example, the classifiers are used in PARALLEL, each offering n opinion as to which class the example should be labelled with. A "**master classifier**" collects this information and then chooses the label that has received more votes.

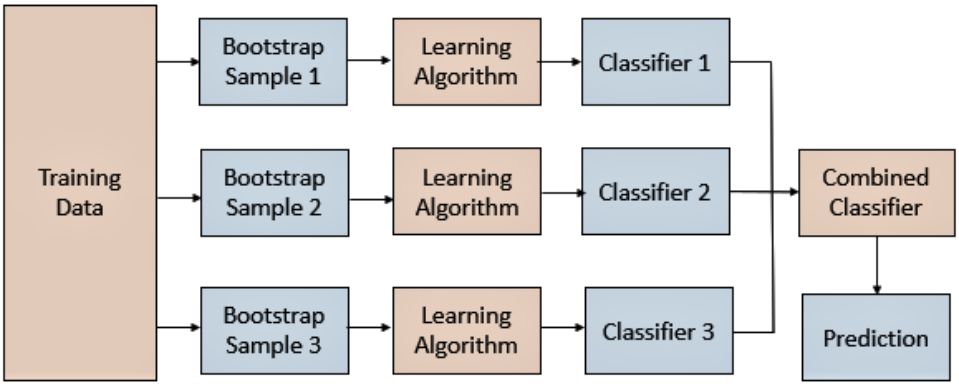
The three classifiers is classifying same 17 examples. By doing so, each classifier came with different examples. The errors can be estimated by VOTING.











Definition

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.

How bagging works

In 1996, Leo Breiman (PDF, 829 KB) (link resides outside IBM) introduced the bagging algorithm, which has three basic steps:

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.

Parallel training:

These bootstrap samples are then trained independently and in parallel with each other using weak or base learners.

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 of bagging

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.

Challenges of bagging:

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

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. More specifically, this research predicts the onset of diabetes based on various risk predictors.

IT:

Bagging can also improve the precision and accuracy in IT systems, such as, network intrusion detection systems.

Environment:

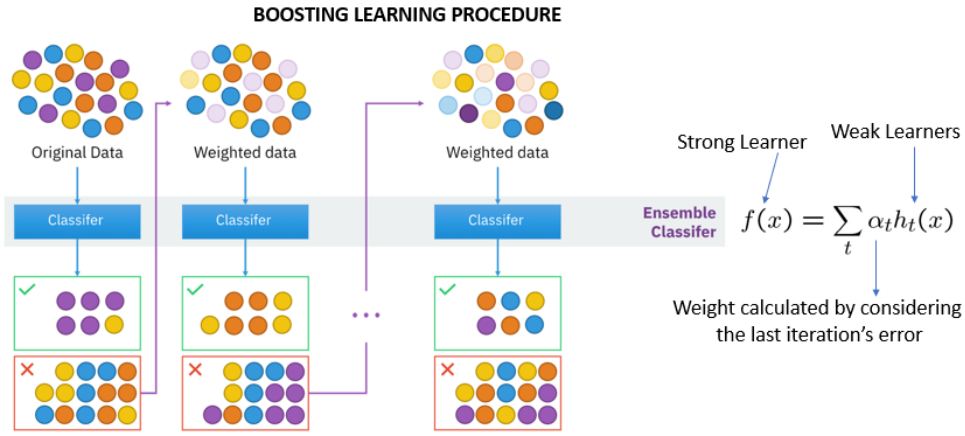
Ensemble methods, such as bagging, have been applied within the field of remote sensing.

Finance:

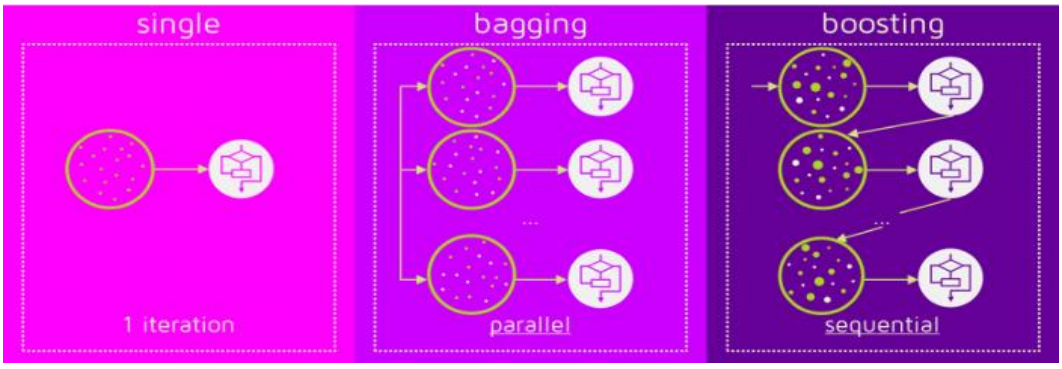
Bagging has also been leveraged with deep learning models in the finance industry, automating critical tasks, including fraud detection, credit risk evaluations, and option pricing problems. The research highlights how bagging helps to minimize risk by to prevent credit card fraud within banking and financial institutions.

BOOSTING

Definition



The term 'Boosting' refers to a family of algorithms which converts weak learner to strong learners. Boosting is an ensemble method for improving the model predictions of any given learning algorithm. The idea of boosting is to train weak learners sequentially, each trying to correct its predecessor.



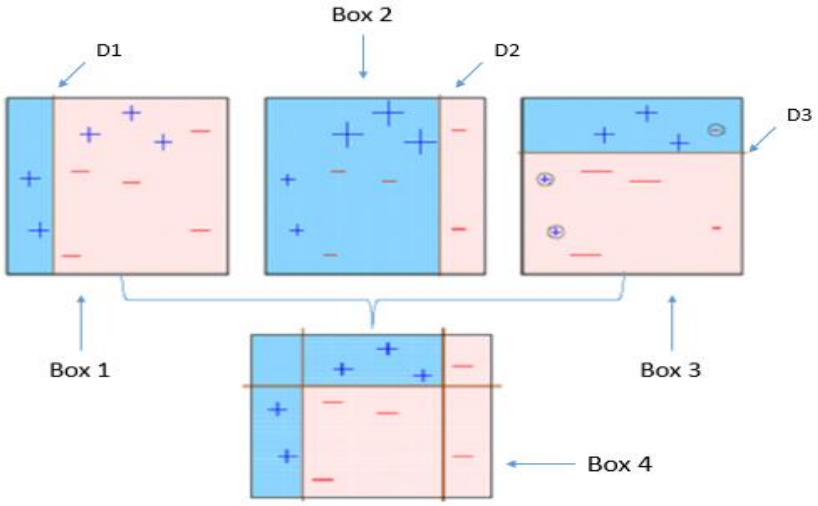
Boosting is an ensemble learning method that combines a set of weak learners into a strong learner to minimize training errors. In boosting, a random sample of data is selected, fitted with a model and then trained sequentially—that is, each model tries to compensate for the weaknesses of its predecessor.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

Types of Boosting

AdaBoost (Adaptive Boosting)

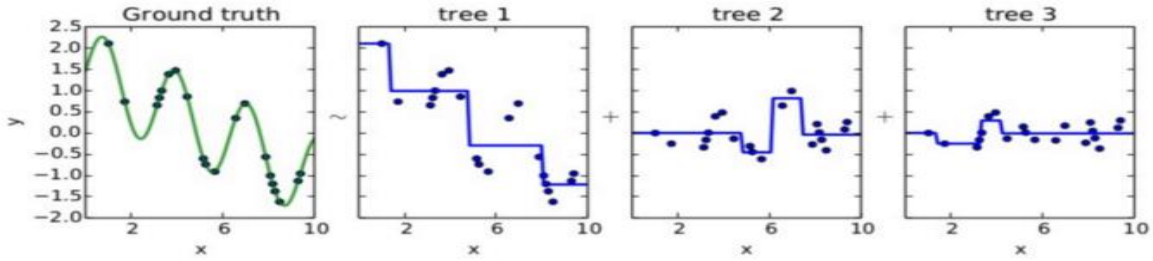
Adaboost combines multiple weak learners into a single strong learner. The weak learners in AdaBoost are decision trees with a single split, called **decision stumps**. When AdaBoost creates its first decision stump, all observations are weighted equally. To correct the previous error, the observations that were incorrectly classified now carry more weight than the observations that were correctly classified. AdaBoost algorithms can be used for both classification and regression problem.

As we see above, the first decision stump(D1) is made separating the (+) blue region from the (—) red region. We notice that D1 has three incorrectly classified (+) in the red region. The incorrect classified (+) will now carry more weight than the other observations and fed to the second learner. The model will continue and adjust the error faced by the previous model until the most accurate predictor is built.



Gradient Boosting

Just like AdaBoost, Gradient Boosting works by sequentially adding predictors to an ensemble, each one correcting its predecessor. However, instead of changing the weights for every incorrect classified observation at every iteration like AdaBoost, Gradient Boosting method tries to fit the new predictor to the residual errors made by the previous predictor.



GBM uses Gradient Descent to find the shortcomings in the previous learner's predictions. GBM algorithm can be given by following steps.

Fit a model to the data, F1(x) = y

Fit a model to the residuals, h1(x) = y−F1(x)

Create a new model, F2(x) = F1(x) + h1(x)

By combining weak learner after weak learner, our final model is able to account for a lot of the error from the original model and reduces this error over time.

ALGORITHM

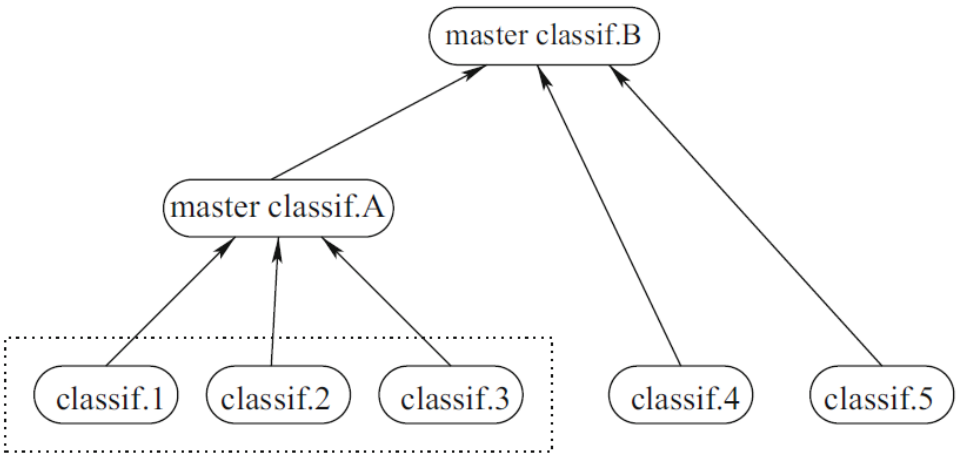
Input: the training set (T) and the user's choice of the induction technique

(1) Create a random training subset, (T_1) and induce from it classifier C_1.

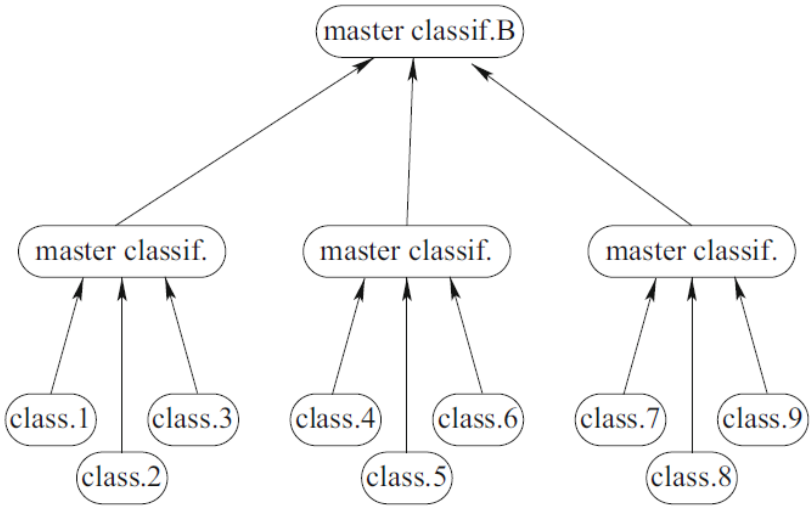
(2) Create a training subset (T_2) in a manner that makes sure that C_1 scores 50% on it. Induce from T_2 classifier C_2.

(3) Create (T_3) such that C_1 and C_2disagree on each of the examples it contains. Induce from (T_3) classifier C_3.

(4) For classification, use plain majority voting.



In the diagram, Master Classifier A combines the votes of classifier 1 – 3. Then, master classifier B combines these votes of master classifier A with those of classifiers 4 and 5.


master classif.B
master classif.
master classif.
master classif.
class.1
class.2
class.3
class.4
class.5
class.6
class.7
class.8
class.9

Q Explain the steps of Adaptive Boosting for the given Dataset.

SL NO	Age	Sex	Chol	Thalach	OldPeak	That	Target
1	63	1	233	150	2·3	1	1
2	37	1	250	187	3·5	2	1
3	41	0	204	172	1·4	2	0
4	56	1	236	178	0·8	2	1
5	57	0	354	163	0·6	2	0

⇒ No of independent variable = Age, Sex, Chol, Thalach, OldPeak, That = 6.
So No of decision stumps = 6. (taking each as root and finding number of correct and miss classification data).

No of Iteration for each weak classifiers = 6 (no of features).

1st Iteration

Step1 No. of data instances = 5 (rows)
So, each row will be assigned with initial weight = 1/5.

So 1st Iteration will be with (Age & Target) columns.

Age	Target (Actual)	Target (Predicted)	Initial weight	Updated weight
63	1	1	1/5	0·167
37	1	1	1/5	0·167
41	0	1 (wrong)	1/5	0·250
56	1	1	1/5	0·167
57	0	1 (wrong)	1/5	0·250

VVI
$ln_e(P) = 2.303 \times Log_{10}(P)$

Step2: Decision stumps for Age (Hage)
if (age ≥ 37) → 1
→ 0
So, 2 miss classification. (row 3, 5).

Weighted error (ϵ_{Age} of Hage):

$$\epsilon_i = \sum_{j=1}^{N} H_i(d_j)\, wt(d_j)$$

$H_j(d_j) = 0$ (correct prediction)
$= 1$ (wrong prediction)

$\epsilon_{age} = 2 \times \frac{1}{5} \doteq 0.4$ (only for wrong).

Step3: Weight of weak classifier

$$\alpha_{age} = \frac{1}{2} \ln\left(\frac{1-\epsilon_{age}}{\epsilon_{age}}\right) = \frac{1}{2} \ln\left(\frac{1-0.4}{0.4}\right) \approx 0.203$$

Step4: Normalize factor Z_{age}

Z_{age} = wt(Correctly classified Instances) * NO of Correctly Classification * $e^{-\lambda age}$

+ wt(wrongly classified Instances) * NO of wrong classification * $e^{+\lambda age}$

$= \frac{1}{5} \times 3 \times e^{-.203} + \frac{1}{5} * 2 * e^{+0.203}$

use in calculator 'e' then '^' sign then value

$= 0.489 + .490 = 0.979.$

Step5: Update weight of all data instances.

$$wt(d_j)_{j+1}^{\text{correct classification}} = \frac{wt(d_j)_{age} \text{ of correct instances} * e^{-\lambda age}}{Z_{age}}$$

$$= \frac{\frac{1}{5} * e^{-0.203}}{.979} = 0.167$$

$$wt(d_j)_{j+1}^{\text{mis classification}} = \frac{wt(d_j)_{age} \text{ of wrong instances} * e^{+\lambda age}}{Z_{age}}$$

$$= \frac{\frac{1}{5} * e^{+0.203}}{0.979} = 0.25$$

Now on the dataset table weight with (1/5) will be updated with (.167) for correct instances and (0.25) with wrong instances.

→ This updation is done w.r.t 'AGE'.

The updated weight table will be used for 2nd Decision Stumps with 'Sex' and continued for all independent variables.

(d)

CGPA	Interactiveness	Practical knowledge	Com" skill	Job Prediction (Actual)
≥9	Y	Good	Good	Y
<9	N	Good	Moderate	Y
≥9	N	Average	Moderate	N
<9	N	Average	Good	N
≥9	Y	Good	Good	Y
≥9	Y	Good	Moderate	Y

Benefits of boosting

Ease of Implementation:

Boosting can be used with several hyper-parameter tuning options to improve fitting. No data pre-processing is required, and boosting algorithms like have built-in routines to handle missing data. In Python, the scikit-learn library of ensemble methods (also known as sklearn.ensemble) 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 only select features that increase its predictive power during training, it can help to reduce dimensionality as well as increase computational efficiency.

Challenges of boosting

Overfitting:

There's some dispute in the research around whether or not boosting can help reduce overfitting or exacerbate it.

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. Boosting algorithms can be slower to train when compared to bagging as a large number of parameters can also influence the behaviour of the model.

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. Boosted classifiers allow for the computations to be stopped sooner when it's clear in which way a prediction is headed. This means that a search engine can stop the evaluation of lower ranked pages, while image scanners will only consider images that actually contains 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 products pricing analysis improve the accuracy of analysing 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 "adaptative 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**.

RANKING

What is scoring and ranking in machine learning?

Scoring is also called prediction, and is the process of generating values based on a trained machine learning model, given some new input data. The values or scores that are created can represent predictions of future values, but they might also represent a likely category or outcome.

Why is ranking data important?

Ranking data sets is useful when statements on the order of observations are more important than the magnitude of their differences and little is known about the underlying distribution of the data. Naturally, information is sacrificed by resorting to ranks, thus, be sure that there is good reason to rank raw data.

What is the use of ranking algorithm?

Page ranking algorithms are used by the search engines to present the search results by considering the relevance, importance and content score and web mining techniques to order them according to the user interest.

How do you present ranking data?

The simplest way to show ranking data is through a column or bar chart, ordered by frequency from greatest to least.