## Bagging

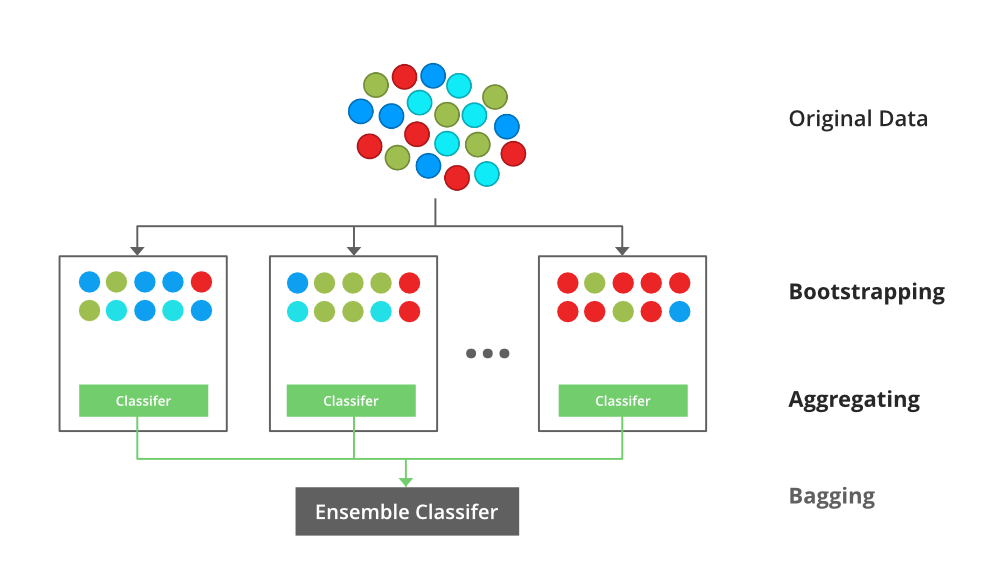
**B**ootstrap **A**ggregating, also known as bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It decreases the [variance](https://www.geeksforgeeks.org/mathematics-mean-variance-and-standard-deviation/)and helps to avoid [overfitting](https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/). It is usually applied to [decision tree methods](https://www.geeksforgeeks.org/decision-tree/). Bagging is a special case of the model averaging approach.

**Description of the Technique**

Suppose a set D of d tuples, at each iteration i, a training set Di of d tuples is selected via row sampling with a replacement method (i.e., there can be repetitive elements from different d tuples) from D (i.e., bootstrap). Then a classifier model Mi is learned for each training set D < i. Each classifier Mi returns its class prediction. The bagged classifier M\* counts the votes and assigns the class with the most votes to X (unknown sample).

**Implementation Steps of Bagging**

* **Step 1:** Multiple subsets are created from the original data set with equal tuples, selecting observations with replacement.
* **Step 2:** A base model is created on each of these subsets.
* **Step 3:**Each model is learned in parallel with each training set and independent of each other.
* **Step 4:**The final predictions are determined by combining the predictions from all the models.



*An illustration for the concept of bootstrap aggregating (Bagging)*

**Example of Bagging**

The [Random Forest model](https://www.geeksforgeeks.org/random-forest-regression-in-python/) uses Bagging, where decision tree models with higher variance are present. It makes random feature selection to grow trees. Several random trees make a Random Forest.

*To read more refer to this article:* [Bagging classifier](https://www.geeksforgeeks.org/ml-bagging-classifier/)

## Boosting

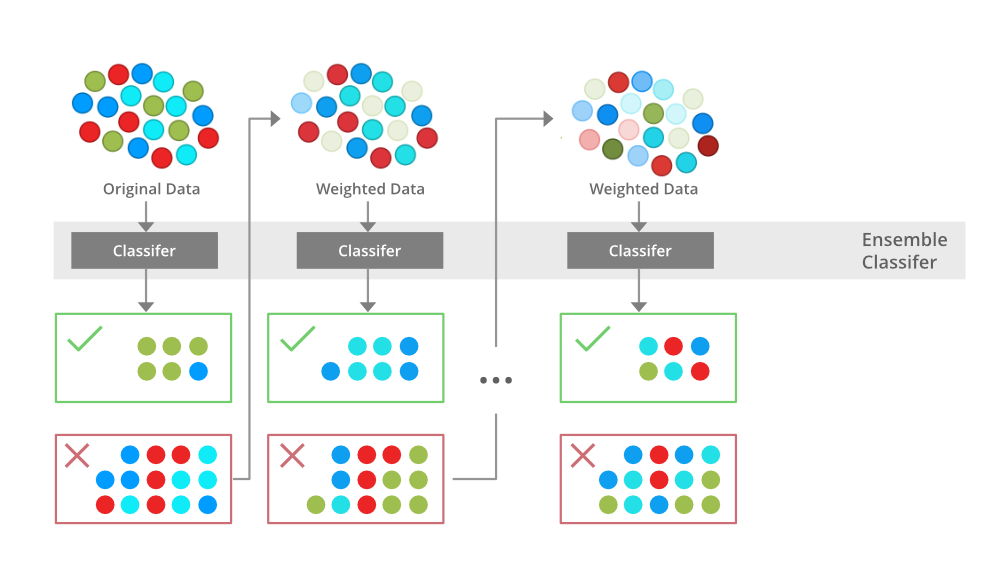
Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models is added.

**Boosting Algorithms**

There are several boosting algorithms. The original ones, proposed by **Robert Schapire** and **Yoav Freund** were not adaptive and could not take full advantage of the weak learners. Schapire and Freund then developed [AdaBoost](https://www.geeksforgeeks.org/implementing-the-adaboost-algorithm-from-scratch/), an adaptive boosting algorithm that won the prestigious Gödel Prize. AdaBoost was the first really successful boosting algorithm developed for the purpose of binary classification. AdaBoost is short for Adaptive Boosting and is a very popular boosting technique that combines multiple “weak classifiers” into a single “strong classifier”.

***Algorithm:***

1. *Initialise the dataset and assign equal weight to each of the data point.*
2. *Provide this as input to the model and identify the wrongly classified data points.*
3. *Increase the weight of the wrongly classified data points and decrease the weights of correctly classified data points. And then normalize the weights of all data points.*
4. *if (got required results)  
     Goto step 5  
   else  
     Goto step 2*
5. *End*



*An illustration presenting the intuition behind the boosting algorithm, consisting of the parallel learners and weighted*

**Data types and measurement scales in Machine Learning**



One of the most confusing aspects when you start working on a Machine Learning project is how to treat your data. Treating your features correctly is absolutely important as it will have a significant effect on your algorithms. The number one thing to remember while performing feature engineering is that all algorithms expect numerical features and hence we cannot feed any characters as a feature.

Data can be classified as:

1. **Qualitative (Categorical)**
2. **Quantitative (Numerical)**

Qualitative or categorical data type as the name suggests will have categorical/ non-numerical values.

Quantitative can be classified into **discreate or continuous.**

* **Discrete**: This will have whole numbers. That means, integer values and not decimals. The most important point to note is that the values will not change with time.  
  Eg: Current number of children in a family, number of calls made in a day, current population of a country, current GDP of a country, age of a person at a particular point in time.
* **Continuous**: Can take any numerical value and has infinite possibilities. Point to note is that the values of a continuous variable can change with time.  
  Eg: Height/ Weight of a person, mileage of a car, age of a person in a 10years dataset, GDP/ population of a country in a 5years dataset.

Note that age, population, GDP can be discrete or continuous based on the scenario where they are being used. To handle both discrete and continuous data types, we can create bins.

It is important to note that, if a feature has just one value throughput the dataset, we will drop it irrespective of whether it is quantitative or qualitative because then the feature is serving as a constant and not a variable.

**Various measurement levels:**

Now that we have classified our data into various data types, let’s have a look at the various measurement levels under which we can classify each data type.

* Nominal
* Ordinal
* Interval
* Ratio

**Nominal:**If the values in a variable does not follow any particular order, we can call it as nominal. Taking a mean or median is meaningless here. Note that sorting the values of a nominal data type does not make any difference. For example, a feature containing various values for colors, brand names, days of a week, types of musical instruments, country names and so on. One hot encoding/ dummy variable creation is the best way to handle these features.

**Ordinal:**If the values in a variable follows a particular order, then we can call it as ordinal. This means a lower value present in the feature holds lesser weight compared to a higher value. Hence, sorting the values of an ordinal data makes sense. For example,

* Ratings on a scale of 10: 1 being lowest and 10 being highest
* Age classification: < 30 years, 30–45 years, 45- 60 years, > 60 years
* Socio-economic classification: low-income earners, average earners top earners
* Purchasing power: low, average, medium, high, very high
* Customer type based on product return behavior: Minimal-Returner, Low-Returner, Normal-Returner, Heavy-Returner

To convert these features into numerical, integer encoding/ label-encoding can be used. Here, each value will be assigned an integer label such as low=0, average=1, medium=2, high=3, very high=4. It is important to note that if an ordinal feature is already label encoded in the dataset, we should not create dummy variables for them as this would destroy the order maintained in the feature.

**Interval:**

In interval type, 0 doesn’t have a true meaning. In the case of temperature, 0 doesn’t mean no temperature. Instead, it is a valid value. A classic example for interval data is temperature. The interval between each value in the interval scale has a meaning and will be equal. Note that calculating percentages, multiplication or division of interval values does not make any sense. But we can add or subtract values. Few other examples for interval data type are geographic coordinates in latitude/ longitude, time measured in a 12 hour clock, temperature measured in Celsius or Fahrenheit. In interval data, we cannot compare 2 values in a ratio. For example, we cannot say 10 am is twice as 5am. It makes no sense.

**Ratio:**If there is a true meaning for 0, then we can call it ratio data type. For example, in the case of length or income, a value 0 means no length or no income. They are of type ratio. We can do addition, subtraction, multiplication, division, percentage calculation, calculate mean/ median. few examples are: income of a person, weight/ height/ length of an object, number of students in a class, number of cars owned by a person. Note that expressing a ratio type data in ratio will have a meaning. For example, we can compare a person of weight 50kg with someone of weight 100kg and say the 2nd person has a weight twice as the first person.

**Other examples:**

**Binary:**If a feature has just 2 possible options, then it is a binary variable. It is a qualitative/ categorical datatype. Examples include, yes/no , male/ female, positive/ negative, east/west, south/north and so on. We can replace the values with 0s and 1s. Note that a binary variable does not follow any order and hence the level of measurement is nominal.

**Time/ date:**As the name suggests, these features will hold a timestamp value or a date value. The data type and measurement level depends upon how you use the date column. We can extract relevant information from a date column such as day of the week, month of the year. We can also use date columns to extract the number of days/ months/ years since the event has happened. For example, current date — date column = age of a person or age of a building or number of years since an event has happened. If we are extracting day of a week or month of a year, we need to one-hot encode the column. Day of the week extracted is categorical and nominal. If we are looking at the number of days between 2 dates, it is quantitative/ numerical and discrete.

Some of the data types which are not covered in this blog include text, image, audio, video.

**Conclusion:**

When we get a dataset and start doing data analysis, we need to not only classify them as categorical or numerical but also understand their level of measurement. This is a very important step in the data pre-processing phase