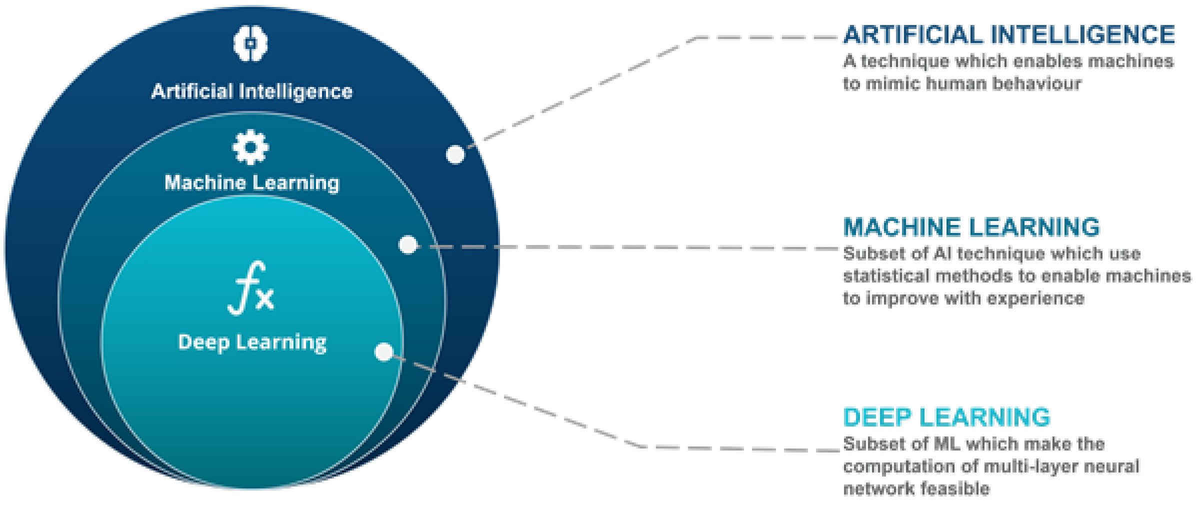
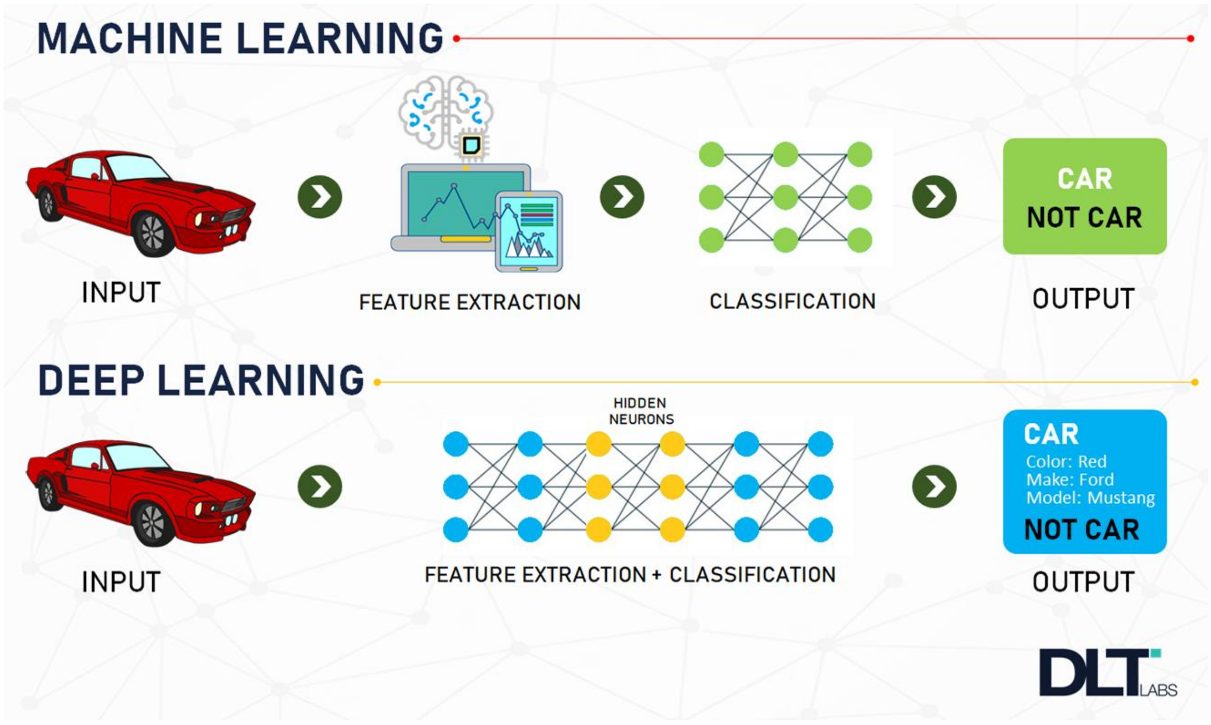

Artificial Intelligence
Machine Learning
Deep Learning
ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour
MACHINE LEARNING
Subset of AI technique which use statistical methods to enable machines to improve with experience
DEEP LEARNING
Subset of ML which make the computation of multi-layer neural network feasible



## Boosting:

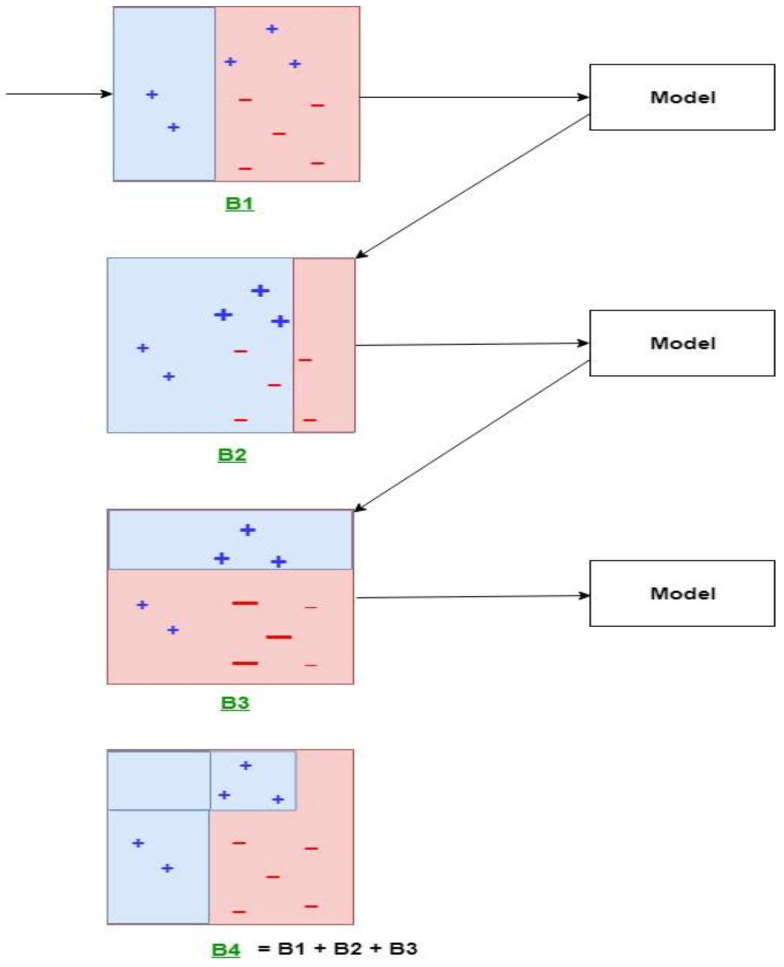
**Boosting** is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors 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 are added.

**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". It was formulated by Yoav Freund and Robert Schapire. They also won the 2003 Gödel Prize for their work.

**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.
4. if (got required results)
     Goto step 5
   else
     Goto step 2

5. End
```



**Explanation:**
The above diagram explains the AdaBoost algorithm in a very simple way. Let's try to understand it in a stepwise process:

- **B1** consists of 10 data points which consist of two types namely plus(+) and minus(-) and 5 of which are plus(+) and the other 5 are minus(-) and each one has been assigned equal weight initially. The first model tries to classify the data points and generates a vertical separator line but it wrongly classifies 3 plus(+) as minus(-).

- **B2** consists of the 10 data points from the previous model in which the 3 wrongly classified plus(+) are weighted more so that the current model tries more to classify these pluses(+) correctly. This model generates a vertical separator line that correctly classifies the previously wrongly classified pluses(+) but in this attempt, it wrongly classifies three minuses(-).
- **B3** consists of the 10 data points from the previous model in which the 3 wrongly classified minus(-) are weighted more so that the current model tries more to classify these minuses(-) correctly. This model generates a horizontal separator line that correctly classifies the previously wrongly classified minuses(-).
- **B4** combines together B1, B2, and B3 in order to build a strong prediction model which is much better than any individual model used.

4 Boosting Algorithms in Machine Learning

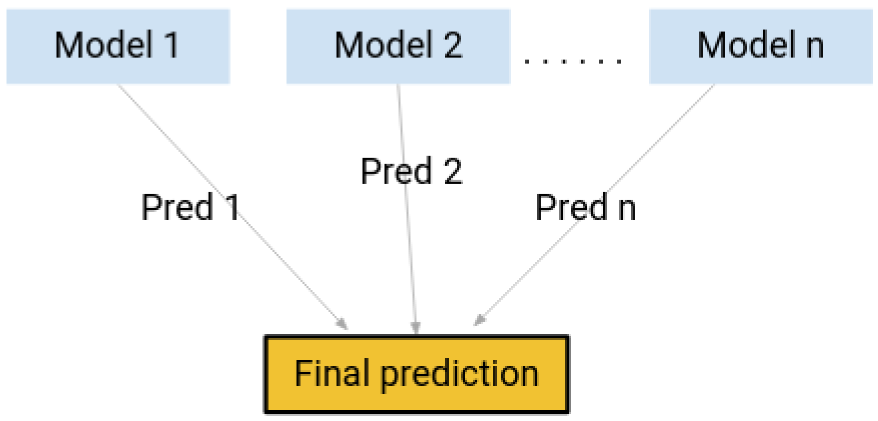
1. Gradient Boosting Machine (GBM)
2. Extreme Gradient Boosting Machine (XGBM)
3. LightGBM
4. CatBoost

## Quick Introduction to Boosting (What is Boosting?)

Picture this scenario:

You've built a linear regression model that gives you a decent 77% accuracy on the validation dataset. Next, you decide to expand your portfolio by building a k-Nearest Neighbour (KNN) model and a decision tree model on the same dataset. These models gave you an accuracy of 62% and 89% on the validation set respectively.

It's obvious that all three models work in completely different ways. For instance, the linear regression model tries to capture linear relationships in the data while the decision tree model attempts to capture the non-linearity in the data.



How about, instead of using any one of these models for making the final predictions, we use a combination of all of these models?

I'm thinking of an average of the predictions from these models. By doing this, we would be able to capture more information from the data, right?

That's primarily the idea behind ensemble learning. And where does boosting come in?

Boosting is one of the techniques that uses the concept of ensemble learning. A boosting algorithm combines multiple simple models (also known as weak learners or base estimators) to generate the final output.

We will look at some of the important boosting algorithms in this article.

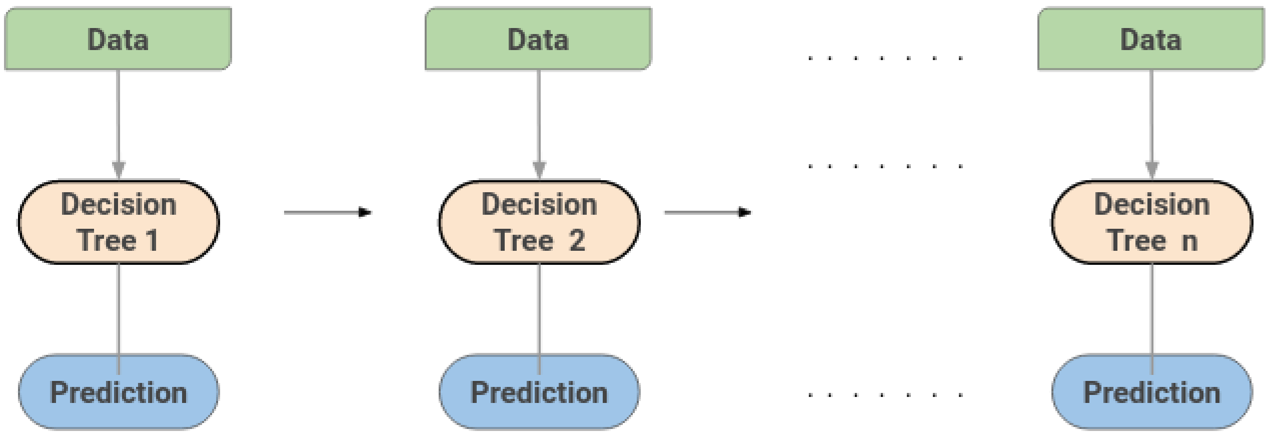
## 1. Gradient Boosting Machine (GBM)

A Gradient Boosting Machine or GBM combines **the predictions from multiple decision trees to generate the final predictions.** Keep in mind that all the weak learners in a gradient boosting machine are decision trees.

But if we are using the same algorithm, then how is using a hundred decision trees better than using a single decision tree? How do different decision trees capture different signals/information from the data?

Here is the trick – **the nodes in every decision tree take a different subset of features for selecting the best split.** This means that the individual trees aren't all the same and hence they are able to capture different signals from the data.

Additionally, each new tree takes into account the errors or mistakes made by the previous trees. So, every successive decision tree is built on the errors of the previous trees. This is how the trees in a gradient boosting machine algorithm are built sequentially.



Here is an article that explains the hyperparameter tuning process for the GBM algorithm:

- Guide to Parameter Tuning for a Gradient Boosting Machine (GBM) in Python

2. Extreme Gradient Boosting Machine (XGBM)

Extreme Gradient Boosting or XGBoost is another popular boosting algorithm. In fact, XGBoost is simply an improvised version of the GBM algorithm! The working procedure of XGBoost is the same as GBM. The trees in XGBoost are built sequentially, trying to correct the errors of the previous trees.

Here is an article that intuitively explains the math behind XGBoost and also implements XGBoost in Python:

- An End-to-End Guide to Understand the Math behind XGBoost

But there are certain features that make XGBoost slightly better than GBM:

- One of the most important points is that XGBM implements **parallel preprocessing (at the node level) which makes it faster than GBM**
- XGBoost also includes a variety of regularization techniques that **reduce overfitting and improve overall performance.** You can select the regularization technique by setting the hyperparameters of the XGBoost algorithm

Learn about the different hyperparameters of XGBoost and how they play a role in the model training process here:

- Guide to Hyperparameter Tuning for XGBoost in Python

Additionally, if you are using the XGBM algorithm, you don't have to worry about imputing missing values in your dataset. **The XGBM model can handle the missing values on its own**. During the training process, the model learns whether missing values should be in the right or left node.

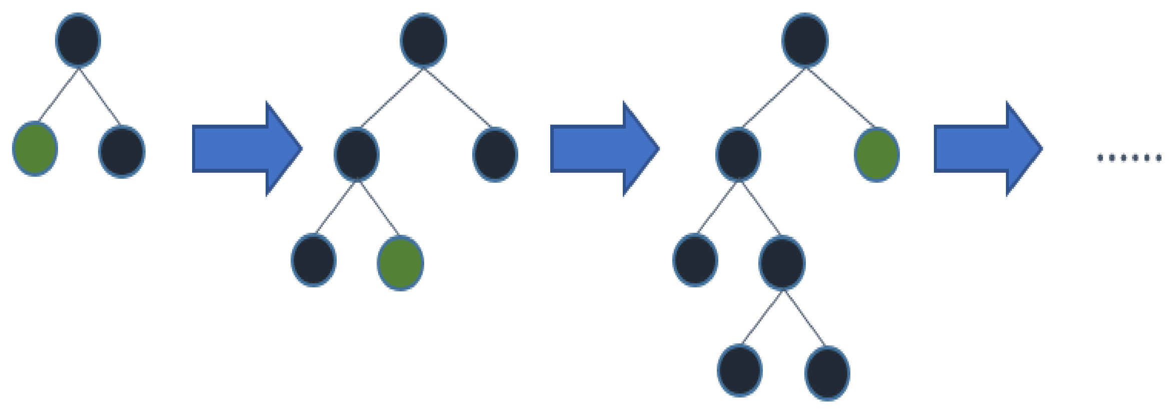
## 3. LightGBM

The LightGBM boosting algorithm is becoming more popular by the day due **to its speed and efficiency.** LightGBM is able to **handle huge amounts of data** with ease. But keep in mind that this algorithm **does not perform well with a small number of data points**.

Let's take a moment to understand why that's the case.

The trees in **LightGBM have a leaf-wise growth, rather than a level-wise growth**. After the first split, the next split is done only on the leaf node that has a higher delta loss.

Consider the example I've illustrated in the below image:



Leaf-wise tree growth

After the first split, the left node had a higher loss and is selected for the next split. Now, we have three leaf nodes, and the middle leaf node had the highest loss. The leaf-wise split of the LightGBM algorithm enables it to work with large datasets.

In order to speed up the training process, **LightGBM uses a histogram-based method for selecting the best split**. For any continuous variable, instead of using the individual values, these are divided into bins or buckets. This makes the training process faster and lowers memory usage.

Here’s an excellent article that compares the LightGBM and XGBoost Algorithms:

- LightGBM vs XGBOOST: Which algorithm takes the crown?

## 4. CatBoost

As the name suggests, CatBoost is a boosting algorithm that can **handle categorical variables in the data.** Most machine learning algorithms cannot work with strings or categories in the data. Thus, converting categorical variables into numerical values is an essential preprocessing step.

CatBoost can internally handle categorical variables in the data. These variables are transformed to numerical ones using various statistics on combinations of features.

If you want to understand the math behind how these categories are converted into numbers, you can go through this article:

- Transforming categorical features to numerical features

Another reason why CatBoost is being widely used is that it works well with the default set of hyperparameters. Hence, as a user, we do not have to spend a lot of time tuning the hyperparameters.

Here is an article that implements CatBoost on a machine learning challenge:

- CatBoost: A Machine Learning Library to Handle Categorical Data Automatically

**Additive-trees** are used to represent objects as “leaves” on a tree, so that the distance on the tree between two leaves reflects the similarity between the objects. Formally, an observed similarity δ is represented by a tree-distance *d*. As such, additive-trees belong to the descriptive multivariate statistic tradition. Additive-tree representations are useful in a wide variety of domains.