What is Boosting in Machine Learning?

Traditionally, building a Machine Learning application consists of taking a **single learner**, like a Logistic Regressor, a Decision Tree, Support Vector Machine, or an Artificial Neural Network, feeding it data, and teaching it to perform a certain task through this data.

Then ***ensemble methods*** were born, which involve using **many learners** to enhance the performance of any single one of them individually. These methods can be described as techniques that use a **group of *weak learners*** (*those who on average achieve only slightly better results than a random model*) together, in order **to create a stronger, aggregated one**.

Generally, ensemble methods are built by **grouping variants of individual Decision Trees**, as we will see later.

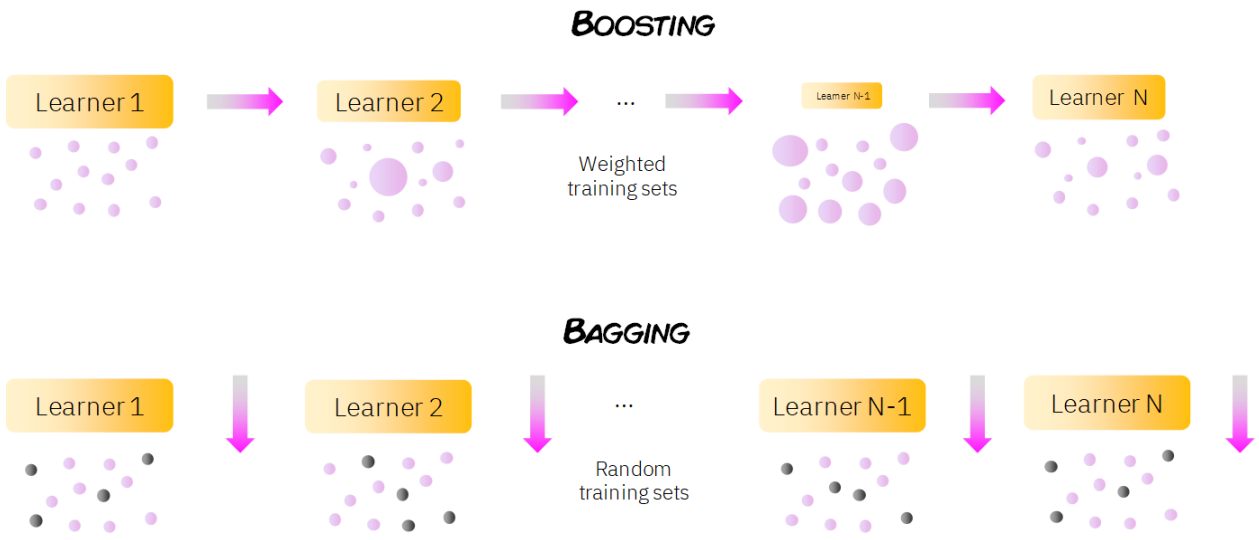
Boosting models fall inside this family of ***ensemble methods.***

Boosting, initially named ***Hypothesis Boosting***, consists of the idea of filtering or weighting the data that is used to train our team of weak learners, so that each new learner gives more weight or is only trained with observations that have been poorly classified by the previous learners.

By doing this **our team** of models **learns to make accurate predictions** on all kinds of data, not just on the most common or easy observations. Also, if one of the individual models is very bad at making predictions on

some kind of observation, it does not matter, as the other N-1 models will most likely make up for it.

Boosting **should not be confused with *Bagging***, which is the other main family of ensemble methods: while in bagging the weak learners are trained in parallel using randomness, **in boosting the learners are trained sequentially**, in order to be able to perform the task of data weighting/filtering described in the previous paragraph.



Bagging vs Boosting

As we can see from the previous image, **in boosting the models can have different importance** or weights (represented in the different sizes of the learners), while **in bagging all learners have the same weight** in the final decision.

Also, **in boosting, the data set is weighted** (represented by the different sizes of the data points), so that observations that were incorrectly classified by classifier n are given more importance in the training of model $n + 1$, while **in bagging the training samples are taken randomly** from the whole population.

Now that we have seen what boosting is, and its differences with bagging, **let's see why it works so well!**

Why is Boosting so effective?

In general, **ensemble methods reduce the bias and variance** of our Machine Learning models. If you don't know what bias and variance are, don't worry, I got you covered with this article.

Ensemble methods help **increase the stability and performance** of machine learning models **by eliminating the dependency of a single estimator**. This can be clearly seen with a Bagging example: *Random Forest*.

Random Forest are, as their name suggests, a group of individual Decision trees that make up a forest. These individual trees are quite prone to overfitting

the data, and despite being very simple and intuitive models, they're **not so great at making predictions.**

However, **if we use many trees these problems disappear**, as each tree is trained with different samples of data and different features, resulting in a stronger and more robust model overall.

With boosting, this works in the same way, but while in bagging each model is trained independently, in boosting the N models are trained sequentially, taking into account the success of the previous model and increasing the weights of the data that this previous model has had the highest error on, which makes the **subsequent models focus on the most difficult data observations.**

Also, the **individual models that perform the best** on the weighted training samples, will become stronger (get a higher weight), and therefore have a higher impact on the final prediction.

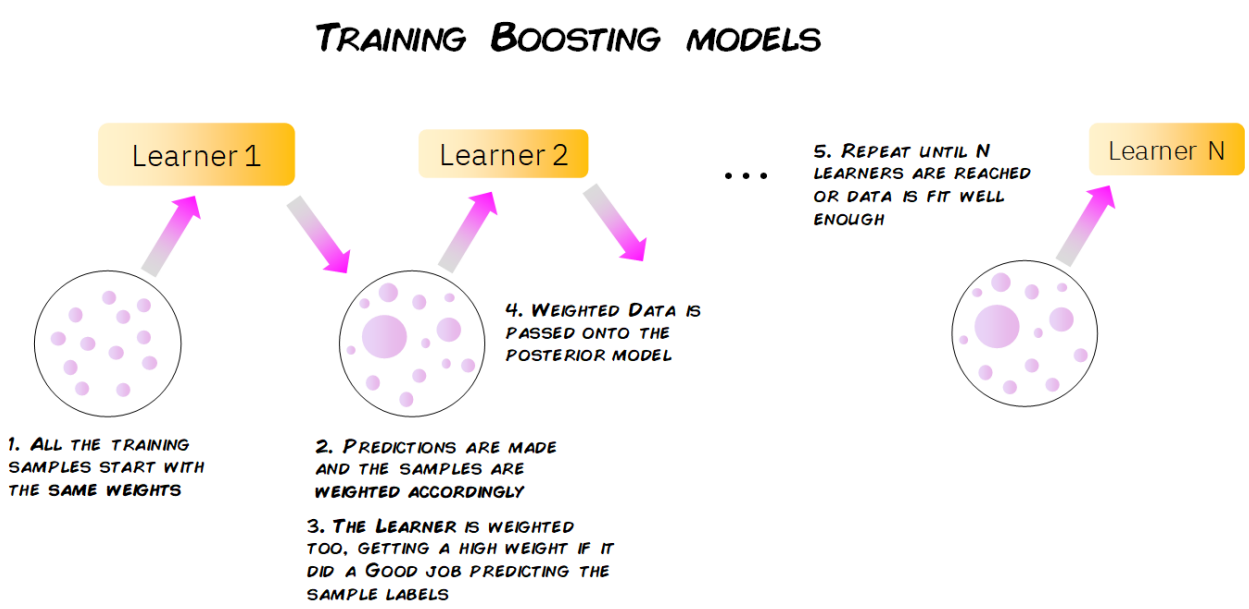
Alright, this sounds lovely, but **how are these models actually trained?**

How is a Boosting Model Trained?

The training process depends on the Boosting algorithm that we are using (Adaboost vs LightGBM vs XGBoost...), but generally it follows this pattern:

1. **All the data samples start with the same weights.** These samples are used to train an individual model (a Decision Tree lets say).

2. The prediction error for each sample is calculated, increasing the **weights of those samples** which have had a greater error, to make them more important for the training of the following individual model.
3. Depending on how well this **individual model** did on its predictions, it **gets assigned an importance/weight or amount of say**. A model that outputs very good predictions will have a high amount of say in the final decision.
4. **The weighted data is passed on to the posterior model,** and 2) and 3) are repeated.
5. Number 4) is repeated until we have reached a certain number of models or until the error is below a certain threshold.



Training boosting models

In some cases, boosting models are trained with a specific **fixed weight for each learner** (called learning rate) and instead of giving each sample an individual weight, the models are trained trying to predict the differences between the previous predictions on the samples and the real values of the objective variable. This difference is what we call ***residuals***.

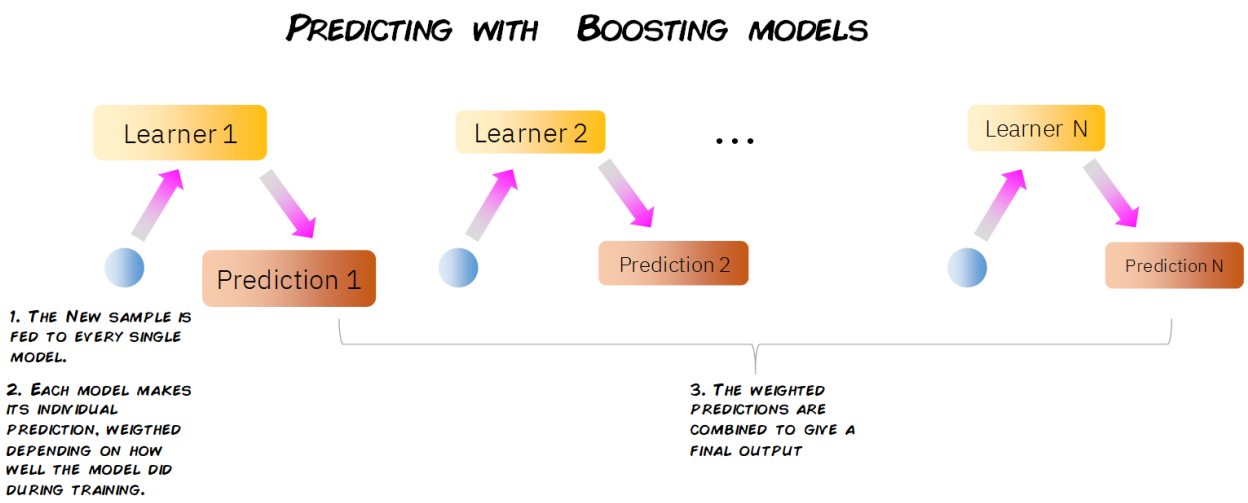
We will speak about this more later, when we see the different kinds of boosting models, however, the main characteristic of the family still remains: **the sequential training on many individual learners to create a more powerful, aggregated model.**

Awesome! Now that we know how Boosting models are trained, let's see **how they are used to make predictions on new data.**

How does a Boosting Model Make Predictions?

They way a boosting model makes predictions on new data is **very simple.** When we get a new observation with its features, it is passed through every one of the individual models, having **each model make its own prediction**.

Then, taking into account the weight of each one of these models, all these **predictions are scaled and combined**, and a final global prediction is give



Predicting with boosting models

To end, let's explore the characteristics of the **most common Boosting models out there.**

Different Boosting Models

AdaBoost

Short for ***Adaptive Boosting***, AdaBoost works by the exact process described before of training sequentially, predicting, and updating the weights of the miss-classified samples and of the corresponding weak models.

It is mostly used with ***Decision Tree Stumps***: decision trees with just a root node and two leaf nodes, where only 1 feature of the data is evaluated. As we can see, by taking into account only 1 feature of our data to make predictions, **each stump is a very very weak model**. However, by

combining many of them, a very robust and accurate ensemble model can be built.

If you want to know more about AdaBoost, check out the following **video by StatQuest.**

Gradient Boosting Machines

Very similar to AdaBoost, ***Gradient Boosting Machines*** train weak learners sequentially, adding more and more estimators, but instead of adapting the weights of the data, **it tries to predict the residual errors made by the previous estimators.**

Because of this, we no longer have sample weights, and **all the weak models have the same amount of say or importance**. Again, most times, Decision trees are used as the base predictors, however, they're not stumps, but

bigger, fixed sized trees. GBMs use a learning rate and take small steps towards better results, in a similar manner conceptually to what is done in Gradient Descent.

Again, if you wanna dive deeper, check out the **video by StatQuest.**

Short for ***eXtreme gradient boosting***, like in Gradient boosting, we fit our trees to the residuals of the previous trees predictions, however, instead of using conventional, fixed size decision trees, XGBoost uses a different kind of trees: ***XGBoost trees*** we could call them.

It builds these trees **by calculating similarity scores** between the observations that end up in a leaf node. Also, XGBoost allows for regularisation, reducing the possible overfitting of our individual trees and therefore of the overall ensemble model.

Lastly, **XGBoost is optimised** to push the limit of the computational resources of boosted tree algorithms, making it a very **high performance** and **fast algorithm** in terms of time and computation.

You can watch the following video ***XGBoost Part 1: Regression***, to get a deeper vision of what XGBoost is all about.

Light Gradient Boosting Machines, known by the short name of ***LightGBM***, are yet another turnaround of improvements for Gradient Boosting algorithms. Instead of using a level-wise growing strategy for the decision trees like in XGBoost, it uses a **leaf-wise growth strategy**, giving it the chance to achieve a higher error reduction per jump than other tree based algorithms. Also, compared to XGBoost, LightGBM is generally faster, especially on large data sets.

Conclusion

That is it! As always, I hope you **enjoyed the post**, and that I managed to help you understand what boosting is, how it works, and why it is so powerful.