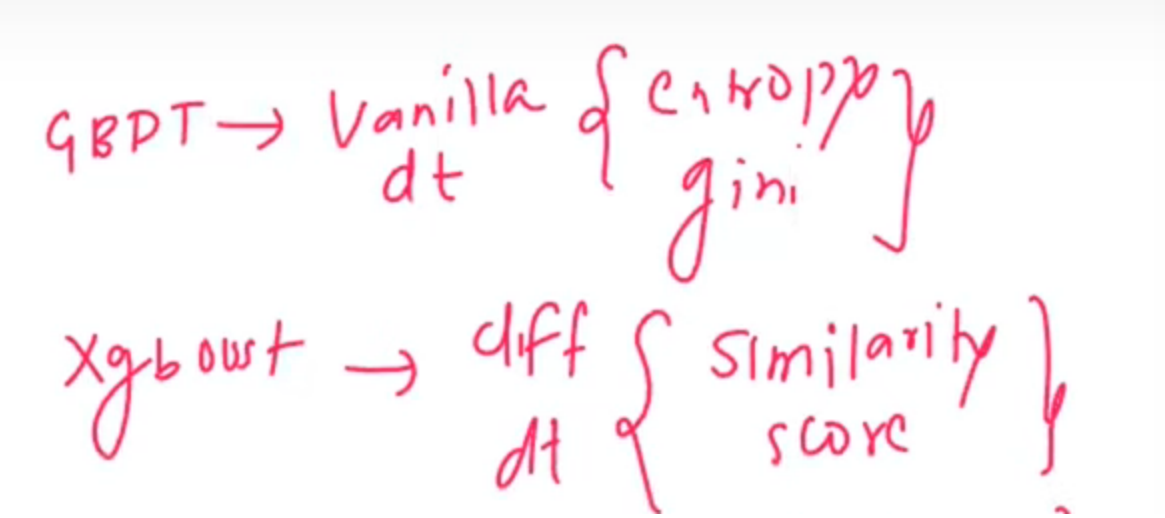
XGBoost (Extreme Gradient Boosting) Introduction

- XGBoost is not an algorithm
- It’s a library built on existing **gradient boosting** algorithms.
- It’s **GradientBoost** + **Software engineering concepts**
- **Performance + Speed**
- Supports multiple languages
- Regression, classification, time-series forecast, ranking

- **Gradient Boosting**: This is an ensemble technique where new models (typically decision trees) are added to correct the errors made by previous models. The goal is to minimize the residuals (errors) by focusing on the harder-to-predict cases.
- **XGBoost**: It's an optimized implementation of gradient boosting, which is faster and more efficient than traditional gradient boosting algorithms due to advanced features like parallelization, regularization, and efficient handling of missing data.

Decision Trees: Gradient Boost vs XGBoost



> **In Gradient boost, the DTs calculate entropy & gini impurity, while in XGBoost, it calculates similarity score.**

Advantages of Gradient Boosting

- You can use any loss function
- Can work with:
  - Regression
  - Classification
  - Ranking
  - Custom defined problems

Speed

- Parallel Processing
  - There is no `njobs` in other boosting algos. So what does Parallel Processing mean?
    - **You do multiple splittings in a tree at the same time.**

        - **Multi-threading** with the `n_jobs` parameter lets you control how many cores to use for training.
        - **Feature Parallelism**: XGBoost divides the feature space (the features available in the dataset) into smaller chunks and computes the optimal split for each chunk in parallel.
- Optimized Data Structures
    - XGBoost stores data in column blocks
- Cache Awareness
- Out of Core computing
    - Big datasets are divided into smaller chunks & you load that chunk in RAM.
    - To activate this, set `tree_method='hist'`
- Distributed Computing
    - You distribute your task in nodes
    - The tasks are carried out parallely
    - You need external libraries link `dask` & `Kubernetes`
- GPU Support
    - `tree_method=gpu_hist` ← This will enable the GPU & speed up the process

Performance

- Regularized Learning Objective
    - Reduces overfitting
    - Regularization term is added in the loss function
- Sparsity Aware Split Finding
    - Sparsity is when your data has too many **zeros** or **missing values**
    - XGBoost handles this 👆 data well
- Handling Missing values

    - It figures out how to fill the missing values
- **Efficient Split Finding (Weighted Quantile Sketch + Approximate Tree Learning)**
    - **Approximate Tree Learning:** You do binning → Convert the data into bins (10-50, 51-100,101-150,etc.)
    - 👆 This is called histogram based training
    - The technique used for creating these bins is called **Weighted Quantile Sketch**
        - You study the distribution of the data & decide bins
        - You do this binning on the basis of quantiles
        - The bins are not uniform. If the region is dense, the bins will be small.
        - By this way, you'll be able to study the data accurately.
- Tree Pruning
    - There's a parameter called `gamma:` It specifies the minimum loss reduction required to make a split.
    - The larger `gamma` is, the more conservative the algorithm will be.
    - Meaning: A new branch will only be created if there;'s a significant reduction in `gamma` (loss)
    - 👆Range: [0,∞]

Alternative to XGBoost:

- **LightGBM:** Free and open-source distributed gradient-boosting framework for machine learning, originally developed by Microsoft.
- **CatBoost:** Supervised machine learning method that is used by the Train Using AutoML tool and uses decision trees for **classification and regression.**
    - USP: **Categorical features support:** Improve your training results with CatBoost that allows you to use non-numeric factors, instead of having to pre-process your data or spend time and effort turning it to numbers.