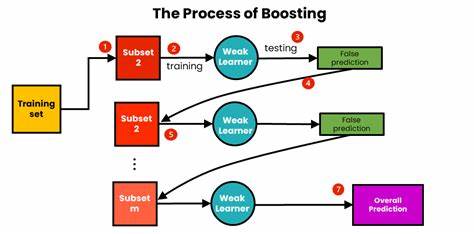
Boosting Assignment

**Boosting:**

It is an ensembling technique which builds a strong learner from the number of weak learners.

It first builds a model from the training data. The second model will be built on the mis-classified data of the first model. This process continues until all the training data’s are predicted correctly.



**Types of Boosting:**

Adaboost

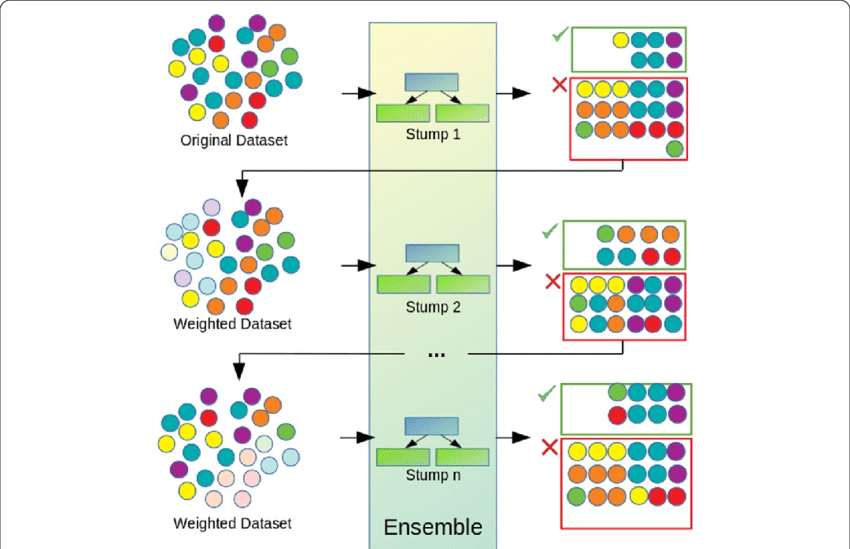
Gradient Boost

XG Boost

Light GBM

**Adaboost:**

An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction.



**Adaboost algorithm**

Step 1: Sample weight creation

A sample weight will be allocated to each data record. Each data points weight ranges between 0 and 1.

Step 2: Stump Creation

A stump is nothing but a decision tree with a depth of 1 or one-level tree

Step 3: Stump Selection

Based on entropy of the stumps it will select a stump with minimum entropy

Step 4: Calculate the total error(TE).

Total error is nothing but the sum of error of misclassified records

Step 5: Calculate the amount of say or performance say

Amount of say=1/2log(1-TE)/TE

Step 6: Update weights

For misclassified data the new weight is calculated as sample weight X e amount of say

For remaining data points the new weight is calculated as sample **weight** X e -amount of say

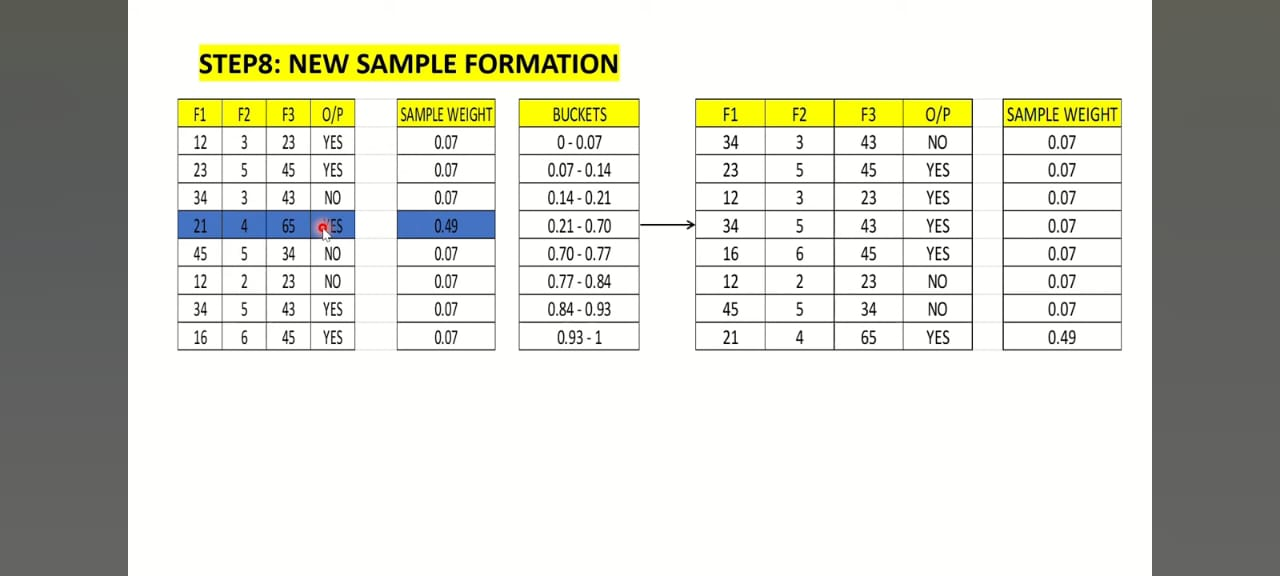
Step 7: Normalise the weight

It is to make the total updated weight ranges between 0 and 1

It is calculated as New weight /sum of New Weights

Step 8: New sample formation

It forms a bucket list from the sample weight and randomly chooses a value and identifies the bucket it falls and changes the arrangement of the accordingly



Step 9: Repeat the step 2 to step 8 until the accuracy has been improved.

**Advantages:**

* Improved accuracy
* Easy implementation
* Versatility
* Robustness to overfitting

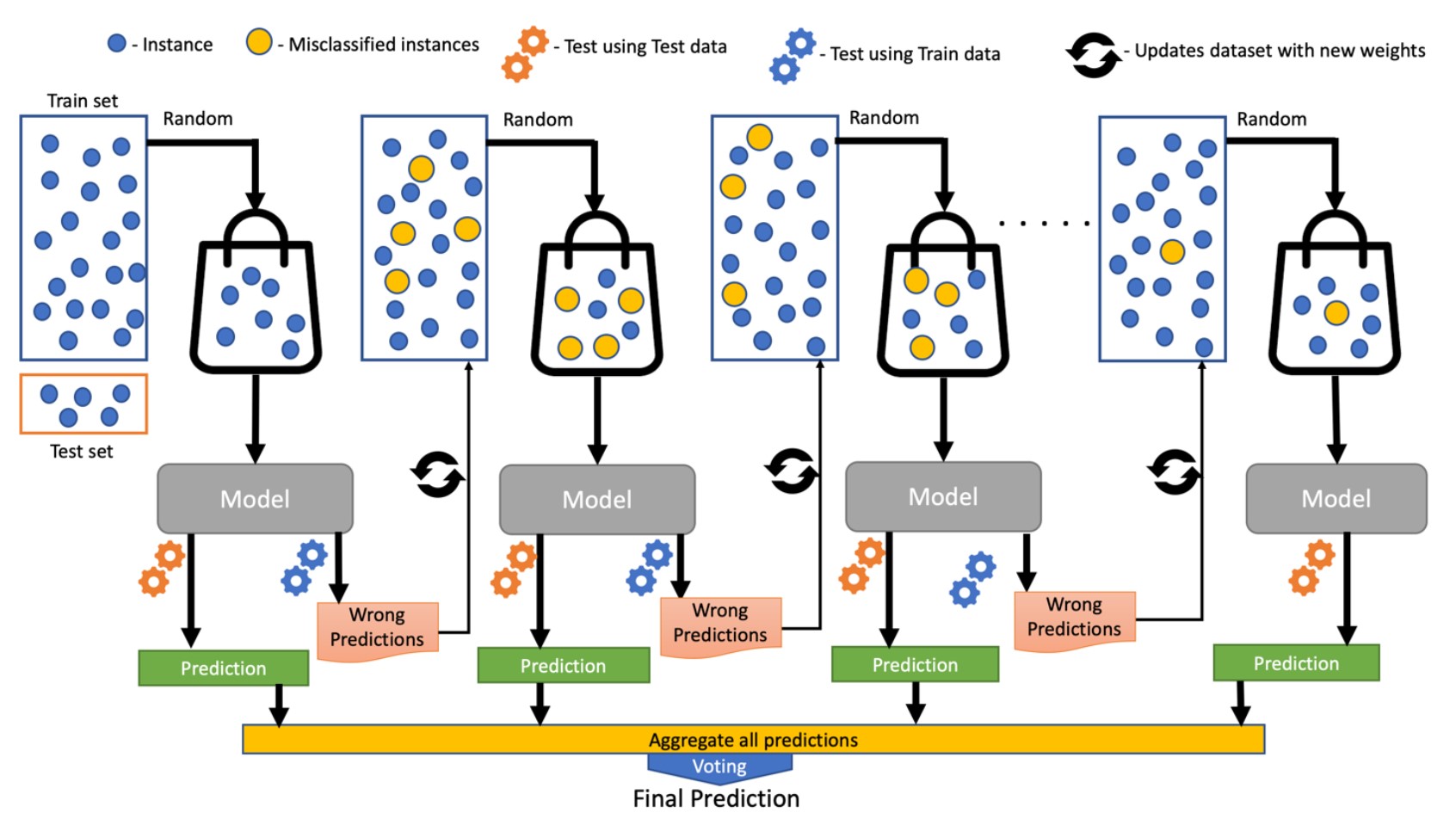
**Disadvantages**

* Sensitivity to noisy data
* Computational cost
* Dependency on weak learners and limited interpretability.

**XG Boost:**

* **XGBoost, or Extreme Gradient Boosting, it builds a strong predictive model by aggregating the predictions of several weak learners, usually decision trees.**
* **It uses a boosting technique to create an extremely accurate ensemble model by having each weak learner after it correct the mistakes of its predecessors.**
* **The optimization method (gradient) minimizes a cost function by repeatedly changing the model’s parameters in response to the gradients of the errors.**

**XG Boost algorithm**



**Step 1: Initialize with a Simple Model**

XGBoost starts with an initial prediction, which is often just the average of all the target values in the dataset. This serves as the initial approximation for the target variable.

**Step 2: Calculate Residuals**

Calculates the residuals (the differences between the actual and predicted values) for each data point in the training set. The residuals represent the errors made by the initial model.

**Step 3: Build a Tree to Predict Residuals**

XGBoost then fits a decision tree to predict these residuals. This tree is usually a small, shallow tree, referred to as a “weak learner”. It’s added to the ensemble in a way that minimizes the loss function.

Step 4: Update Predictions

The predictions from this new tree are then added to the previous predictions, effectively updating the model’s approximation.

**Step 5: Calculate New Residuals**

The residuals are recalculated based on the updated predictions.

**Step 6: Build Another Tree**

A new tree is built to predict these updated residuals. This process is repeated for a predefined number of times, typically referred to as the number of boosting rounds or iterations.

Step 7: Combine Predictions

Finally, the predictions from all the trees are combined to give the final prediction. This combined prediction is typically very accurate due to the ensemble nature of the model.

**Advantages:**

* High performance and accuracy, particularly with structured data
* Efficiently handles missing values and outliers
* Includes built-in regularization to prevent overfitting
* Scales well to large datasets
* Offers flexibility in tuning and optimization
* Provides feature importance scores for interpretability
* Effective pruning

**Disadvantages:**

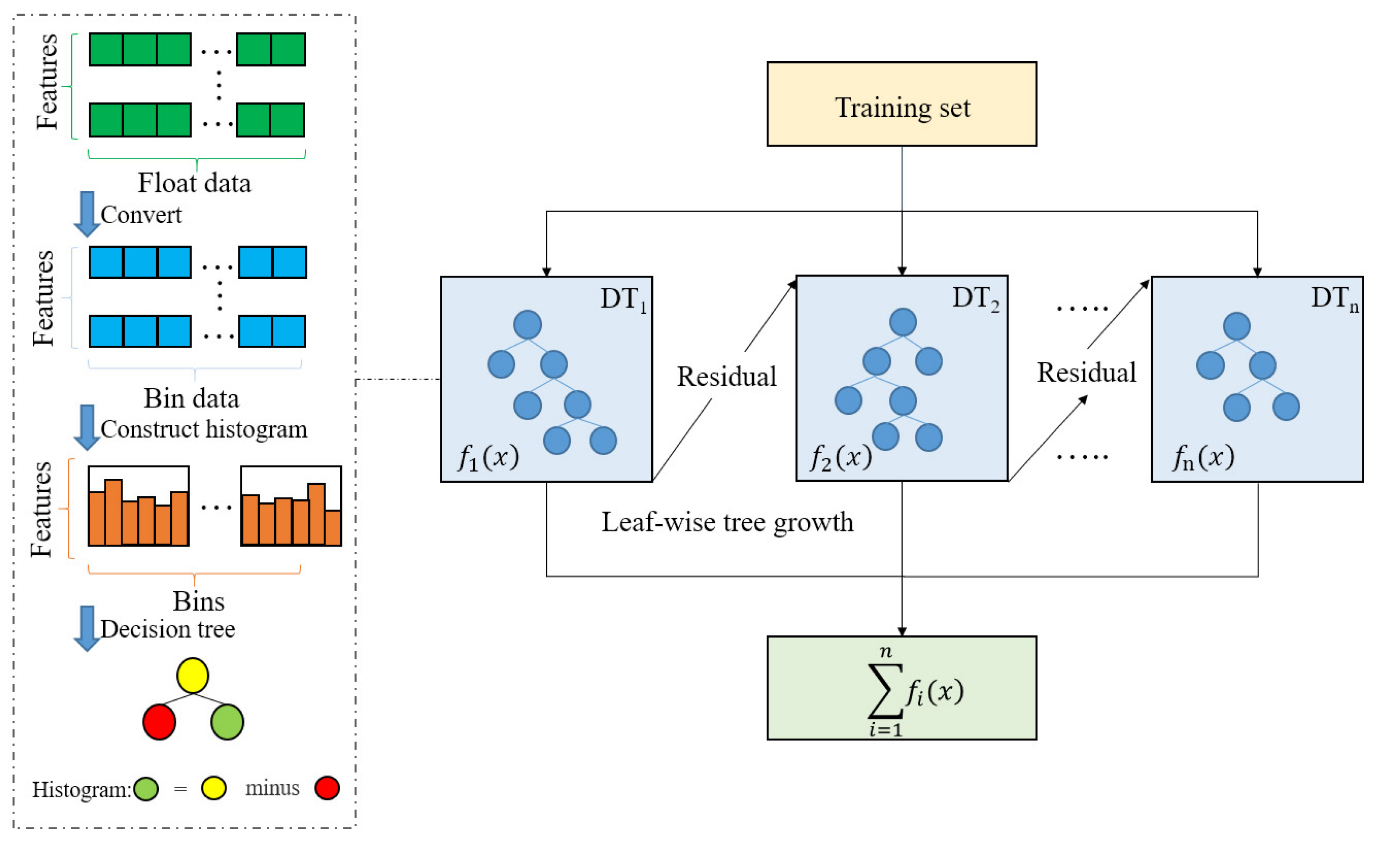
* Requires careful parameter tuning to achieve optimal performance
* Can be prone to overfitting if not properly regularized
* May not perform as well with high-dimensional sparse data
* Training can be computationally expensive, especially with large datasets
* Interpreting the model can be challenging due to its complexity

**LightGBM**

It is a gradient-boosting framework based on decision trees to increase the efficiency of the model and reduces memory usage. It uses two novel techniques:

* Gradient-based One Side Sampling(GOSS)
* Exclusive Feature Bundling (EFB)

LightGBM is a **boosting algorithm** that works by combining many weak decision trees to create a strong model.



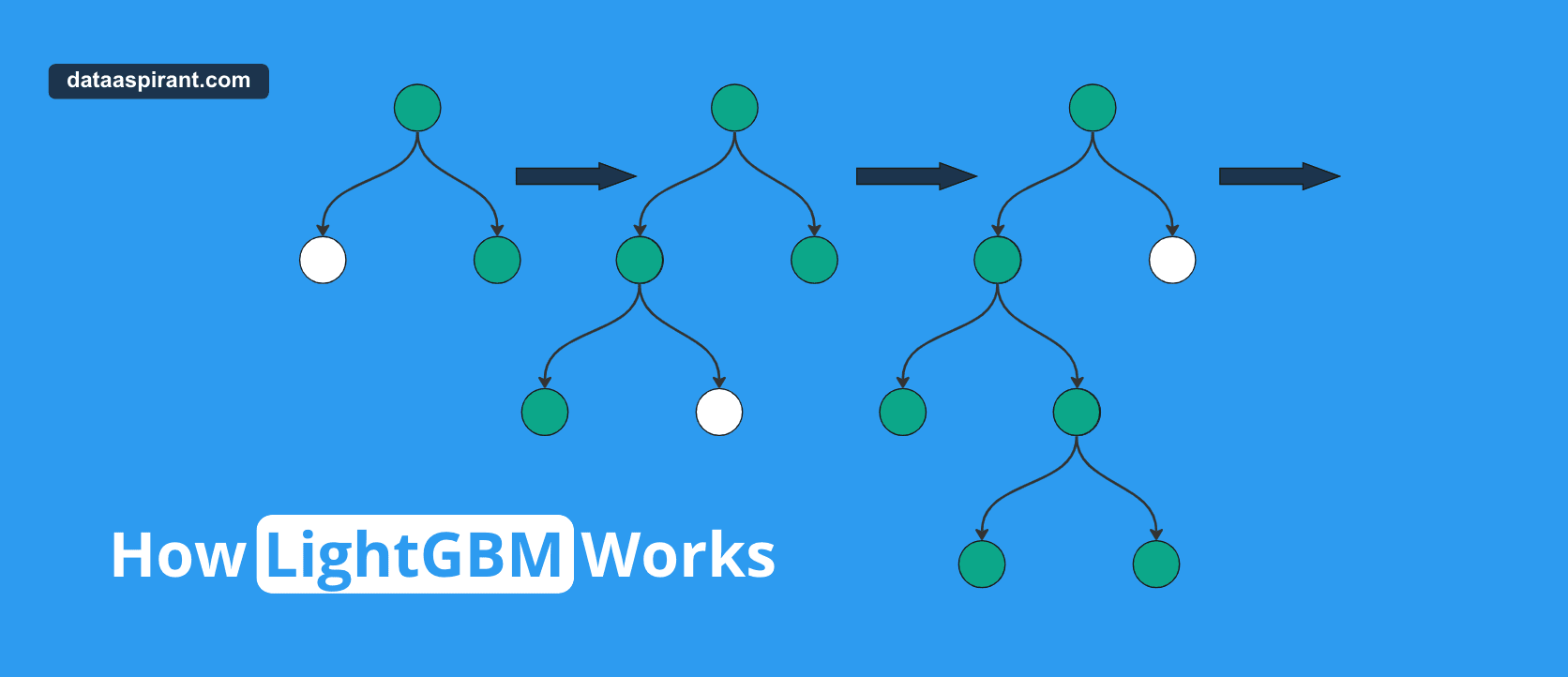
The algorithm starts by creating a [**single decision tree**](https://dataaspirant.com/decision-tree-classifier-implementation-in-r/) that predicts the target variable based on the input features. It then **iteratively adds** more decision trees to the model, with each tree attempting to correct the **errors** of the previous tree.

Histogram-based approach

Grouping the continuous feature values into discrete bins, or histograms, and using these bins to approximate the information gain at each split point.

Leaf-wise tree growth

LightGBM grows the tree by expanding the leaf node that has the highest gain.



Regularization

The regularization techniques used in LightGBM are important for preventing overfitting and improving the accuracy and robustness of the mode

**Advantages**

* High Efficiency
* Faster Learning.
* Scalability

**Disadvantages**

* **Limited Feature Selection**
* **Overfitting**
* **Time and Memory Constraints**