**XGBoost (Extreme Gradient Boosting)** is a high-performance and scalable implementation of gradient-boosted decision trees that has become a standard algorithm for structured data classification and regression tasks. It was introduced by Chen and Guestrin to address limitations in existing boosting algorithms by optimizing both the algorithmic formulation and system-level efficiency [1]. XGBoost implements novel techniques such as second-order gradient approximation, column block data structure, and regularized learning to prevent overfitting. Its parallelization, support for sparse data, and cache-aware access patterns have made it highly competitive in machine learning competitions like Kaggle. Empirical evaluations consistently show that XGBoost outperforms other ensemble methods including AdaBoost and Random Forest on tabular datasets [2], making it a popular choice in real-world applications ranging from bioinformatics to industrial monitoring.

**[1]** T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD ’16), San Francisco, CA, USA, pp. 785–794, 2016, doi: 10.1145/2939672.2939785.  
**[2]** A. Janosi, G. Vaszary, and M. Koncz, “Comparison of Random Forest and XGBoost Algorithms for Supervised Learning,” 2021 29th European Signal Processing Conference (EUSIPCO), Dublin, Ireland, pp. 1516–1520, 2021, doi: 10.23919/Eusipco47968.2021.9615985.

**Overview: How XGBoost is Applied to This Data**

The objective is to **classify spoilage stage** ("Fresh", "Spoiling", "Late Spoilage") based on sensor readings over time. The model learns **patterns in VOC and ethanol evolution** to make these predictions.

**1. Data Input and Preprocessing**

Each batch (Batch 1 = training, Batch 2 = testing) contains minute-by-minute sensor readings:

* BME\_VOC\_Ohm, MQ3\_Top\_PPM, MQ3\_Bottom\_PPM, BME\_Temp, BME\_Humidity

We apply a **3-minute window** over these readings and extract features for each window:

* Statistical features (mean, std)
* Temporal deltas (change from first to last reading)
* VOC-to-ethanol ratios

These become the **input features** X, while the spoilage stage label (determined from average Time\_to\_Spoilage\_Minutes) becomes the **target** y.

**2. XGBoost Training Logic**

XGBoost builds an ensemble of **decision trees** in a stage-wise, additive manner to minimize classification error.

**Key Mechanics:**

* **Initial prediction** starts from a constant (e.g., average class probability).
* At each boosting round:
  + A new tree is trained to predict the **gradient** (error residuals) of the current model.
  + This tree focuses more on **misclassified or hard-to-learn samples**.
  + Trees are added sequentially, each refining the previous one.

**XGBoost Optimization Includes:**

* **Second-order gradient descent** (uses both gradient and Hessian for split decisions)
* **Regularization** (lambda, alpha) to penalize complexity (avoids overfitting)
* **Shrinkage** (learning rate) to reduce overconfidence of each tree
* **Column subsampling** to enhance generalization

**3. Decision Logic Per Tree**

Each decision tree in XGBoost:

* Examines feature thresholds (e.g., VOC\_to\_MQ3\_Top\_Ratio < 2500)
* Makes **binary splits** to partition the data
* Eventually reaches a **leaf node**, which assigns a **score (logit)** for a class

Final prediction is made by:

* Summing contributions of all trees
* Applying **softmax** to convert scores into probabilities
* Selecting the class with highest probability

**Example:**

Let’s say a sample has:

* High VOC,
* Increasing MQ3 PPM,
* Low delta change in VOC over 3 mins.

The model may classify it as "Fresh" because:

* VOC is stable,
* Ethanol (fermentation) has not spiked,
* Ratios fall within thresholds the trees have learned for "Fresh".

**Training Flow Summary**

plaintext

CopyEdit

Raw sensor data ➝ 3-min window ➝ Feature extraction ➝ Label: Spoilage Stage

↓

Training set (X, y) ➝ XGBoost model ➝ Series of trees

↓

Model learns: if VOC ratio ↑ and MQ3 ↑ rapidly → Spoiling or Late Spoilage

**Outcome**

* The model **generalizes spoilage trends** from Batch 1.
* During inference, it predicts spoilage stage on unseen data (Batch 2) based on learned thresholds and interactions.
* Evaluation uses metrics like **accuracy**, **F1-score**, and **ROC curves**.