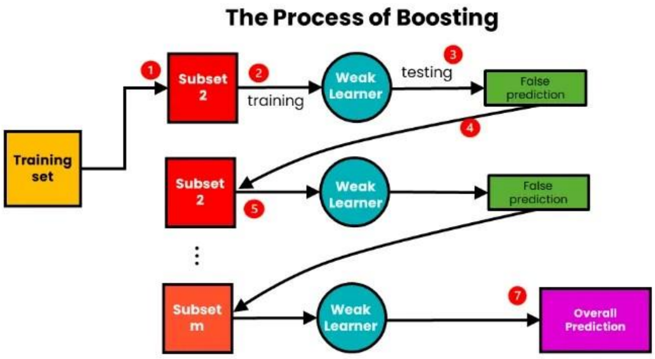
BOOSTING ALGORITHMS

Boosting combines high-bias weak learners through sequential training to reduce both bias and variance by focusing more on previously misclassified data points, unlike bagging which trains models in parallel.




Ada BOOST ALGORITHM

AdaBoost (Adaptive Boosting) is an ensemble learning algorithm that builds a strong classifier by sequentially combining multiple weak learners, typically decision stumps, where each learner focuses more on the samples misclassified by previous ones through adaptive reweighting, and final predictions are made using weighted majority voting.

**Working steps are** –

**Step 1**: Initialize sample weights

At the beginning, all training samples are assigned equal weights:

$$w_i = \frac{1}{N}$$

where $N$is the total number of samples.

**Step 2**: Train a weak learner (Decision Stump)

Train a weak classifier $h_t(x)$(usually a decision stump) on the entire dataset using the current sample weights.
The stump is chosen such that it minimizes the weighted classification error.

**Step 3**: Compute weighted error

Calculate the total weighted error of the weak learner:

$$\epsilon_t = \sum_{i=1}^{N} w_i \cdot \mathbb{1}(y_i \neq h_t(x_i))$$

This represents the sum of weights of the misclassified samples.

**Step 4**: Compute learner weight (Amount of Say)

Calculate the contribution (performance) of the weak learner:

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$$

- Lower error → higher $\alpha_t$
- If $\epsilon_t = 0.5$, the learner has no contribution

**Step 5**: Update sample weights

Update the sample weights to emphasize incorrectly classified samples:

$$w_i \leftarrow w_i \cdot e^{-\alpha_t y_i h_t(x_i)}$$

Which results in:

- Incorrectly classified samples:

$$w_i \leftarrow w_i \cdot e^{+\alpha_t}$$

- Correctly classified samples:

$$w_i \leftarrow w_i \cdot e^{-\alpha_t}$$

**Step 6**: Normalize the weights

Normalize the weights so that they sum to 1:

$$w_i \leftarrow \frac{w_i}{\sum_{i=1}^{N} w_i}$$

**Step 7**: Repeat for multiple rounds

Repeat Steps 2 to 6 for $t = 1$to $T$to build multiple weak learners, where each new learner focuses more on previously misclassified samples.

**Step 8**: Final strong classifier

Combine all weak learners using a weighted majority vote:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \, h_t(x)\right)$$

If problem in understanding above steps read this (https://medium.com/@pingsubhak/boosting-adaboost-gradient-boost-and-xgboost-bdda87eed44e)

Example –

**Problem: Loan Approval Prediction**

**Goal:** Predict whether a loan will be **Approved (+1)** or **Rejected (−1)**
**Weak learner:** Decision stump
**Features:**

- Income (Low / High)
- Credit Score (Poor / Good)

---

## 📊 Dataset

| Sample | Income | Credit Score | Loan (y) |
|---|---|---|---|
| 1 | Low | Poor | −1 |
| 2 | Low | Good | −1 |
| 3 | High | Poor | +1 |
| 4 | High | Good | +1 |
| 5 | Low | Poor | +1 |

👉 This dataset is **not perfectly separable**, making it realistic.

---

## 🔁 Step 1: Initialize Sample Weights

Total samples $N = 5$

$$w_i = \frac{1}{5} = 0.2$$

---

## 🌲 Round 1 – Weak Learner 1

**Choose best decision stump**

**Rule:**

$$h_1(x) = \begin{cases} +1 & \text{if Income = High} \\ -1 & \text{if Income = Low} \end{cases}$$

---

**Predictions**

| Sample | True y | Pred | Correct? | Weight |
|---|---|---|---|---|
| 1 | −1 | −1 | ✓ | 0.2 |
| 2 | −1 | −1 | ✓ | 0.2 |
| 3 | +1 | +1 | ✓ | 0.2 |
| 4 | +1 | +1 | ✓ | 0.2 |
| 5 | +1 | −1 | ✕ | 0.2 |

---

**✕ Weighted Error**

$$\epsilon_1 = 0.2$$

---

**☆ Learner Weight**

$$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-0.2}{0.2}\right) = \frac{1}{2}\ln(4) = 0.693$$

---

**Update Sample Weights**

**Misclassified sample (5):**

$$w_5 = 0.2 \times e^{+0.693} = 0.4$$

**Correct samples:**

$$w = 0.2 \times e^{-0.693} = 0.1$$

### Normalize Weights

Sum = $0.1 \times 4 + 0.4 = 0.8$

| Sample | New Weight |
|---|---|
| 1 | 0.125 |
| 2 | 0.125 |
| 3 | 0.125 |
| 4 | 0.125 |
| 5 | 0.500 |

### 🌲 Round 2 – Weak Learner 2

Focus shifts to **Sample 5**

**New stump:**

$$h_2(x) = \begin{cases} +1 & \text{if Credit Score = Poor} \\ -1 & \text{if Credit Score = Good} \end{cases}$$

### Predictions

| Sample | True y | Pred | Correct? | Weight |
|---|---|---|---|---|
| 1 | −1 | +1 | ❌ | 0.125 |
| 2 | −1 | −1 | ✓ | 0.125 |
| 3 | +1 | +1 | ✓ | 0.125 |
| 4 | +1 | −1 | ❌ | 0.125 |
| 5 | +1 | +1 | ✓ | 0.500 |

### ❌ Weighted Error

$$\epsilon_2 = 0.125 + 0.125 = 0.25$$

---

### ☆ Learner Weight

$$\alpha_2 = \frac{1}{2}\ln\left(\frac{1 - 0.25}{0.25}\right) = 0.549$$

---

### Final Strong Classifier

$$H(x) = \text{sign}(0.693 \cdot h_1(x) + 0.549 \cdot h_2(x))$$

---

### Example Prediction

**New applicant:**
Income = Low, Credit Score = Poor

- $h_1 = -1$
- $h_2 = +1$

$$H(x) = \text{sign}(-0.693 + 0.549) = \text{sign}(-0.144) = -1$$

### → Loan Rejected

**When to use AdaBoost**

- When weak learners perform slightly better than random guessing
- When the dataset is small to medium-sized and relatively clean
- When you want to reduce bias using simple models like decision stumps

**When not to use AdaBoost**

- When the dataset contains heavy noise or many outliers
- When data is very large or highly complex, making training slow or unstable