📘 1.AdaBoost - Adaptive Boosting Algorithm

🔍 Introduction

AdaBoost is a boosting algorithm that combines multiple **weak learners** (typically decision stumps) to form a **strong learner**. It improves model performance by focusing more on the **errors** made by previous models in the sequence.

🧠 Recap: Bagging vs Boosting

- **Bagging** (Bootstrap Aggregation):
  - Multiple base learners are trained **independently**.
  - Example: **Random Forest**.
  - Objective: Reduce variance.
- **Boosting**:
  - Multiple weak learners are trained **sequentially**.
  - Each model focuses on correcting the errors of the previous one.
  - Objective: Reduce both bias and variance.

🌳 What is a Weak Learner?

- A **weak learner** is a model that performs slightly better than random guessing.
- In AdaBoost, we typically use **decision stumps** (decision trees with depth = 1).

🏗️ AdaBoost Workflow

1. Start with equal weights for all samples.

2. Train a weak learner on the dataset.
3. Increase weights for misclassified samples.
4. Train the next weak learner on the updated weights.
5. Repeat steps 2–4 for N iterations.
6. Combine all learners using a **weighted sum**.

📐 Mathematical Formula

If we have N weak learners, the final model is:

$$F(x) = \alpha_1 \cdot h_1(x) + \alpha_2 \cdot h_2(x) + \cdots + \alpha_n \cdot h_n(x)$$

Where:

- $h_1, h_2, ..., h_n$ are weak learners (decision stumps).
- $\alpha_1, \alpha_2, ..., \alpha_n$ are their corresponding weights.
- `F(x)` is the final strong classifier.

🧮 Bias-Variance Trade-off

- A single stump has:
  - **High bias**
  - **Low variance**
- Combined AdaBoost:
  - Reduces bias (by correcting errors)
  - May increase variance slightly
  - Final result: Better generalization

🌱 Decision Stump Explained

- A **decision stump** is a decision tree with **only one split**.
- Simple and fast.
- Used in AdaBoost to act as a weak learner.

- Alone, it **underfits** data.
- In combination (boosting), it contributes to a strong model.

✅ Key Takeaways

- AdaBoost works sequentially to improve prediction by focusing on mistakes.
- It assigns weights to weak learners based on their accuracy.
- Final output is a weighted combination of all weak learners.
- Can be used for **classification** and **regression** problems.

📘 2.AdaBoost Classifier – In-depth Intuition (with Example)

🧮 Sample Dataset

Salary	Credit	Approval
≤ 50K	B	No
≤ 50K	G	Yes
≤ 50K	G	Yes
> 50K	B	No
> 50K	G	Yes
> 50K	N	Yes
≤ 50K	N	No

- **Features**: `Salary` , `Credit`
- **Target**: `Approval` (Yes/No)
- Total samples = **7**

🎯 Goal

Build an **AdaBoost classifier** by combining multiple **weak learners** (here: decision stumps).

🌳 Step 1.0: Create Decision Stumps

🔷 Stump A: `Salary <= 50K`

```
          Salary <= 50K
             /      \
         Yes          No
     2Y / 2N        2Y / 1N
```

- Left Branch (`Salary ≤ 50K`):
  - 4 samples → 2 "Yes", 2 "No"
- Right Branch (`Salary > 50K`):
  - 3 samples → 2 "Yes", 1 "No"

This split has **mixed outcomes** in both branches → Impure

🔷 Stump B: `Credit == G`

```
        Credit == G
          /     \
       Yes       No
     3Y / 1N   1Y / 2N
```

- Left Branch (`Credit = G`):
  - 4 samples → 3 "Yes", 1 "No"
- Right Branch (`Credit ≠ G` → B or N):
  - 3 samples → 1 "Yes", 2 "No"

This stump shows **better separation** of classes.

📊 Step 1.1: Evaluate with Impurity Measures

We choose the **best stump** using:

✔️ Entropy

```
Entropy = -p1 * log2(p1) - p2 * log2(p2)
```

Example (Salary ≤ 50K stump):

- Left: 2 Yes / 2 No → p = 0.5 each
  `Entropy = -0.5*log2(0.5) - 0.5*log2(0.5) = 1`
- Right: 2 Yes / 1 No
  `Entropy = -2/3*log2(2/3) - 1/3*log2(1/3) ≈ 0.918`
- Weighted Average Entropy:
  `= (4/7)*1 + (3/7)*0.918 ≈ 0.965`

✔️ Gini Impurity

```
Gini = 1 - Σ(p²)
```

Example (Credit == G stump):

- Left: 3 Yes / 1 No → p = 0.75 (Yes), 0.25 (No)
  `Gini = 1 - (0.75^2 + 0.25^2) = 1 - 0.625 = 0.375`
- Right: 1 Yes / 2 No → p = 0.33, 0.67
  `Gini = 1 - (1/3^2 + 2/3^2) = 1 - (0.111 + 0.444) = 0.445`
- Weighted Gini:
  `= (4/7)*0.375 + (3/7)*0.445 ≈ 0.405`

✅ Lower than previous stump's entropy/Gini → **Better split**

🔁 How AdaBoost Uses These

1. Initialize **equal weights** on all samples
2. Train **all possible stumps**
3. Choose the one with **lowest weighted error**

4. **Increase weights** of misclassified samples
5. Repeat for next stump
6. Combine all stumps using **weighted majority vote**

📘 Step 2:Sum of the Total Errors and Performance of Stump

🔢 Dataset Overview

Salary	Credit	Approval	Sample Weights
<= 50K	B	No	1/7
<= 50K	G	Yes	1/7
<= 50K	G	Yes	1/7
> 50K	B	No	1/7
> 50K	G	Yes	1/7
> 50K	N	Yes	1/7
<= 50K	N	No	1/7

The weights assigned to each data point are **uniform** (`1/7`), which implies **equal importance** in the decision stump.

🔍 Decision Stump: Credit = G

A **Decision Stump** is a one-level decision tree. In this case, it's trying to make a decision based on the feature `Credit = G` .

Branching Based on Condition:

- **If Credit = G → Predict: Yes**

- **Else → Predict: No**

❌ Total Error from This Split

Out of the data where `Credit ≠ G`, we misclassify one data point:

- 3 values of Credit = G → All classified correctly as "Yes"
- 4 values of Credit ≠ G → 3 "No" and 1 "Yes"
  - The **misclassified point** is:

```
>50K | N | Yes | 1/7
```

✅ **Total Error** = Misclassified weight = **1/7**

📈 Performance of Stump

Performance is measured using the **formula**:

$$\text{Performance (also called as } \alpha) = \frac{1}{2} \ln \left(\frac{1 - \text{TE}}{\text{TE}} \right)$$

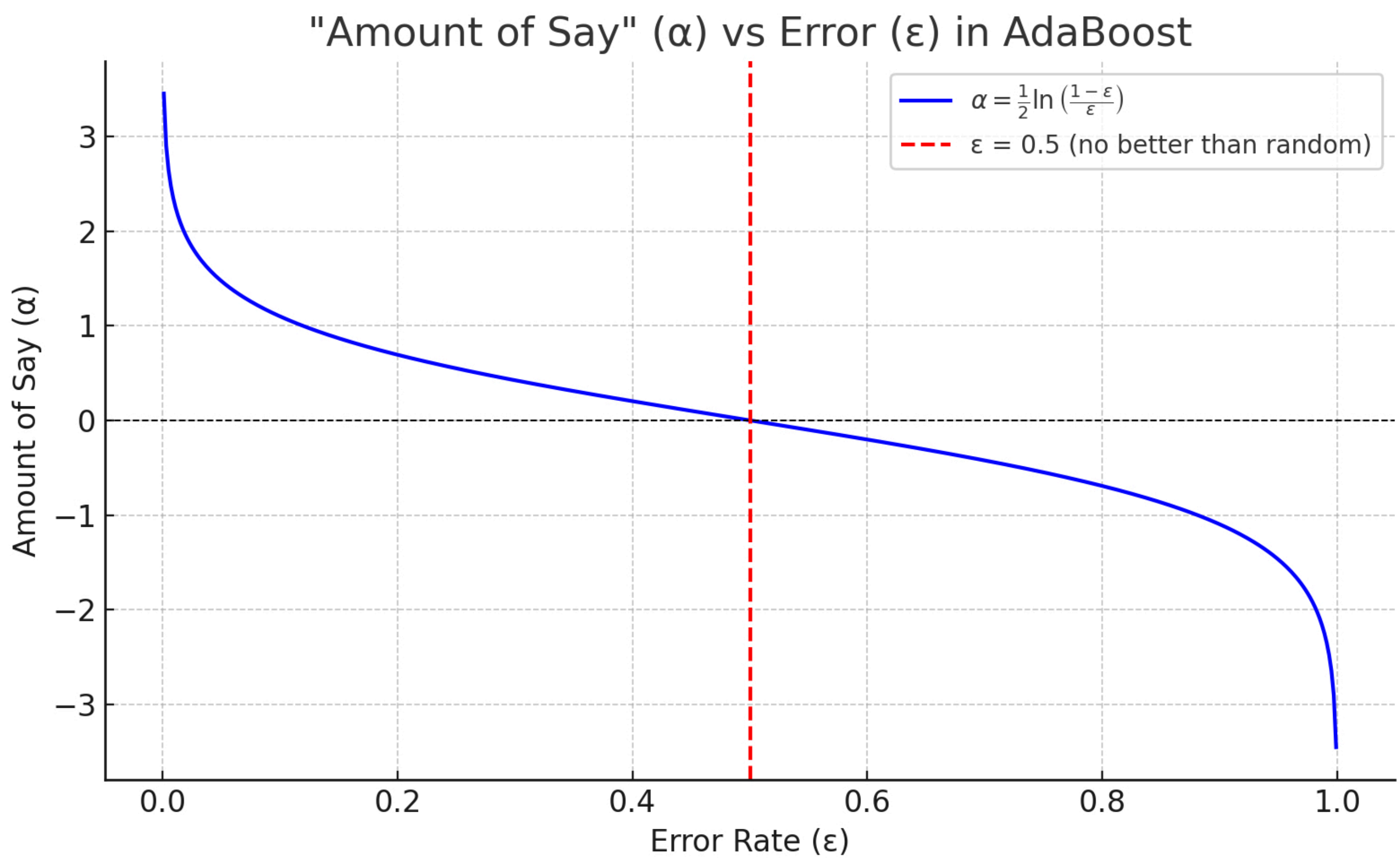
Where:

- **TE** = Total Error = 1/7

Substituting the values:

$$\text{Performance} = \frac{1}{2} \ln \left(\frac{1 - \frac{1}{7}}{\frac{1}{7}} \right) = \frac{1}{2} \ln(6) \approx 0.896$$

This is the **performance score** (also used as a **weight** `α1`) for the weak learner in algorithms like AdaBoost.



AdaBoost: Understanding α Based on Error Rate ε

1. **ε < 0.5 (Better than random) → α > 0**
   A weak learner is doing better than guessing (say, 40% error).

$$\frac{1-\varepsilon}{\varepsilon} > 1 \Rightarrow \ln\left(\frac{1-\varepsilon}{\varepsilon}\right) > 0$$

   - The model assigns positive weight, so its predictions are trusted as-is.
   - The better it performs, the larger α gets.

2. **ε = 0.5 (Random guessing) → α = 0**

$$\frac{1-0.5}{0.5} = 1 \Rightarrow \ln(1) = 0$$

   - This learner adds no useful information—it's like flipping a coin.
   - AdaBoost ignores it entirely.

3. **ε > 0.5 (Worse than random) → α < 0**
   The learner is making more errors than correct predictions.

$$\frac{1-\varepsilon}{\varepsilon} < 1 \Rightarrow \ln\left(\frac{1-\varepsilon}{\varepsilon}\right) < 0$$

- AdaBoost gives it a negative weight, meaning it reverses its prediction (flip the label).
- This is still useful: even a bad learner contains signal if you use it in reverse.

Error Rate ε	Description	α Value
0	Perfect classifier	$+\infty$
< 0.5	Better than random	> 0
$= 0.5$	Random guess	$= 0$
> 0.5	Worse than random	< 0
1	Always wrong	$-\infty$

- **Positive α**: Learner contributes positively to final prediction.
- **Negative α**: Learner contributes inversely (e.g., wrong classifier, flipped decision).
- **Zero α**: Learner is ignored in final prediction.

🧮 Boosting Model Function

Boosting combines multiple weak learners:

$$f(x) = \alpha_1 M_1(x) + \alpha_2 M_2(x) + \cdots + \alpha_n M_n(x)$$

Where:

- α_i is the weight (performance) of stump `Mi` .
- From the example: `α₁ = 0.896` (highlighted in the orange box)

This tells us how important each weak learner is in the final ensemble model.

✅ Conclusion

- A **decision stump** was built using `Credit = G` .
- Only **1 out of 7 samples** was misclassified.
- The **performance** of this stump was calculated using the logarithmic performance formula used in **AdaBoost**.

- This performance value (**0.896**) is then used as the **weight** of the learner in the final boosted model.

📘 Step 3:AdaBoost-Updating Sample Weights

Step: Update the weights for correctly and incorrectly classified points

Salary	Credit	Approval	Sample Weights	Update Wts
<=50K	B	No	1/7 ⬇	→ 0.058
<=50K	G	Yes	1/7 ⬇	→ 0.058
<=50K	G	Yes	1/7 ⬇	→ 0.058
>50K	B	No	1/7 ⬇	→ 0.058
>50K	G	Yes	1/7 ⬇	→ 0.058
>50K	G	Yes	1/7 ⬇	→ 0.058
>50K	N	Yes	1/7 ⬆	→ 0.349
<=50K	N	No	1/7 ⬇	→ 0.058

⬇ means the sample was **correctly classified** (weight decreases).

⬆ means the sample was **misclassified** (weight increases).

📘 Weight Update Rule

✅ For Correctly Classified Points

New Weight = Current Weight × e^(-Performance of Stump)

Example:
= (1/7) × e^(-0.98)
≈ 0.058

❌ For Incorrectly Classified Points

New Weight = Current Weight × e^(+Performance of Stump)

Example:
= (1/7) × e^(0.98)
≈ 0.349

🔍 Notes:

- It gives more importance to incorrectly classified points by increasing their weights.
- It helps the next weak learner focus on the harder examples.

📘 Step 4:Normalized Weights Computation and Assigning Bins in AdaBoost

Table Columns Description

Column Name	Description
Salary	Income level (`<=50K` or `>50K`)
Credit	Credit rating (`B` , `G` , `N`)
Approval	Loan approval outcome (`Yes` or `No`)
Update wts	Updated weights for each data point after applying the weak classifier
Normalized Weights	Weights scaled so they sum to 1
Bins Assignment	Ranges assigned for random sampling proportional to weights

Step 1: Initialize Weights

- Initially, each data point is assigned equal weight.

- In this example, the initial weight is **0.058** for 7 out of 8 samples, and **0.349** for one significant misclassified sample.

Step 2: Update Weights

- Weights are updated based on the classifier's performance.
- The misclassified data point (row: `>50K`, `N`, `Yes`) receives a **higher weight** (0.349), indicating its importance in the next round.
- Remaining samples retain their lower weight.

Step 3: Normalize Weights

- Total weight sum:

```
6 * 0.058 + 0.349 = 0.406 + 0.349 = 0.697
```

- Normalization formula:

```
normalized_weight = weight / total_weight
```

- Example:

```
0.058 / 0.697 ≈ 0.08
0.349 / 0.697 ≈ 0.50
```

- This ensures that all weights now sum up to 1.

Step 4:Assign Bins for Sampling

- Each data point is assigned a **bin range** based on cumulative normalized weights.

Example:

Salary	Credit	Approval	Normalized Weight	Cumulative Bin(Bins Assignment)
≤ 50K	B	No	0.08	0.00 – 0.08

Salary	Credit	Approval	Normalized Weight	Cumulative Bin(Bins Assignment)
≤ 50K	G	Yes	0.08	0.08 – 0.16
≤ 50K	G	Yes	0.08	0.16 – 0.24
> 50K	B	No	0.08	0.24 – 0.32
> 50K	G	Yes	0.08	0.32 – 0.40
> 50K	N	Yes	0.50	0.40 – 0.90
≤ 50K	N	No	0.08	0.90 – 0.98

- This cumulative bin assignment is used to sample data points **proportional to their weights** in the next round.

📘 Step 5:Next Weak Classifier

- A weak classifier (e.g., decision stump) is trained using the **resampled data**.
- Emphasis is on harder (misclassified) samples.
- Boosting continues iteratively by combining multiple such weak classifiers.
- Misclassified points receive higher weight.
- Normalized weights control the sampling.
- AdaBoost focuses future classifiers on previously misclassified examples.

Iterative Process of Data Selection for the Next Decision Tree Stump

This step explains the transition between assigning normalized weights (with bin ranges) and selecting the data for the next decision tree (DT) stump in the AdaBoost algorithm.

🔢 Step 1: Assign Bins to Data Points

Each data point is assigned a bin range based on its normalized weight. For example:

Example:

Salary	Credit	Approval	Normalized Weight	Cumulative Bin(Bins Assignment)
≤ 50K	B	No	0.08	0.00 – 0.08
≤ 50K	G	Yes	0.08	0.08 – 0.16
≤ 50K	G	Yes	0.08	0.16 – 0.24
> 50K	B	No	0.08	0.24 – 0.32
> 50K	G	Yes	0.08	0.32 – 0.40
> 50K	N	Yes	0.50	0.40 – 0.90
≤ 50K	N	No	0.08	0.90 – 0.98

🎲 Step 2: Perform Stochastic Sampling

A random value between 0 and 1 is generated for each data point. Based on this random value, the data point is selected if the value falls within its assigned bin.

Salary (S)	Credit	Approval	Random
>50K	N	Yes	0.50
<=50K	G	Yes	0.10
>50K	N	Yes	0.60
>50K	N	Yes	0.75
<=50K	G	Yes	0.24
>50K	B	No	0.32
>50K	NF	Yes	0.87

We check each random number to see which bin it falls into.

Example Mappings:

- **0.50** falls between `0.40` and `0.90` → Selects: >50K, Normal, Yes

- **0.10** falls below → Selects: <50K, Good, Yes
- **0.60** falls within same bin → Selects: >50K, Normal, Yes (again)
- **0.75** → >50K, Normal, Yes (again)
- **0.24** → <50K, Green, Yes
- **0.32** → >50K, B, No
- **0.87** → >50K, Normal, Yes (again)

We can see some **incorrectly classified records** are being picked repeatedly due to their higher weight.

📦 Step 3: Prepare Dataset for Next DT Stump

All 7 original records are reassembled for the next stump, typically by uniform reweighting or duplication of selected samples as needed to maintain dataset balance.

Salary (S)	Credit	Approval	Sample Weight
>50K	N	Yes	1/7
<=50K	G	Yes	1/7
>50K	N	Yes	1/7
>50K	N	Yes	1/7
<=50K	G	Yes	1/7
>50K	B	No	1/7
>50K	NF	Yes	1/7

This ensures that the training set for the next DT stump contains **all 7 records**, each with equal importance (weight = 1/7).

📈 Performance of New Stump

The newly trained stump's performance is evaluated using a loss function (e.g., exponential loss). In this example, the performance is:

- **TE (Training Error):** 0.65

➕ Final Model Update

The final model is a weighted combination of weak learners:

$$f_t = \alpha_1 h_1(x) + \alpha_2 h_2(x)$$

Where:

- h_1 and h_2 are weak learners
- $\alpha_1 = 0.88$
- $\alpha_2 = 0.65$

📘 Step 6:Final Prediction for Classification

- **Final prediction in AdaBoost (for classification)** is made by combining the predictions of all the weak learners (typically decision tree stumps), each weighted by their accuracy.
- Each weak learner outputs a class label (e.g., *Yes* or *No*), and their contribution to the final prediction is weighted by a coefficient called **alpha (α)**.
- **Alpha (α)** is higher for weak learners with lower error rates and lower (or even negative) for those with higher error rates.
- The final prediction is made by calculating the **weighted sum of predictions** from all weak learners:

$$F(x) = \sum_{i=1}^{N} \alpha_i h_i(x)$$

where $h_i(x)$ is the prediction of the i^{th} weak learner.

- For a new test input (e.g., "salary < 50K and credit score = good"):
  - Decision Tree Stump 1 outputs: *Yes*, $\alpha_1 = 0.896$
  - Decision Tree Stump 2 outputs: *No*, $\alpha_2 = 0.650$
  - Decision Tree Stump 3 outputs: *Yes*, $\alpha_3 = 0.244$
  - Decision Tree Stump 4 outputs: *No*, $\alpha_4 = -0.300$
- Combine weighted votes:
  - For *Yes*: 0.896 + 0.244 = **1.140**

    - For *No*: 0.650 - 0.300 = **0.350**
- Since 1.140 > 0.350, final output = **Yes** → Credit card is approved.
- In classification, **entropy** or **Gini** is used to measure impurity and decide the splits.
- In regression with AdaBoost, **Mean Squared Error (MSE)** is used instead of entropy, and predictions are continuous values.

🔶 Why Not Use Majority Vote Like Random Forest?

- AdaBoost improves accuracy by emphasizing misclassified points via weight adjustment rather than treating all learners equally.

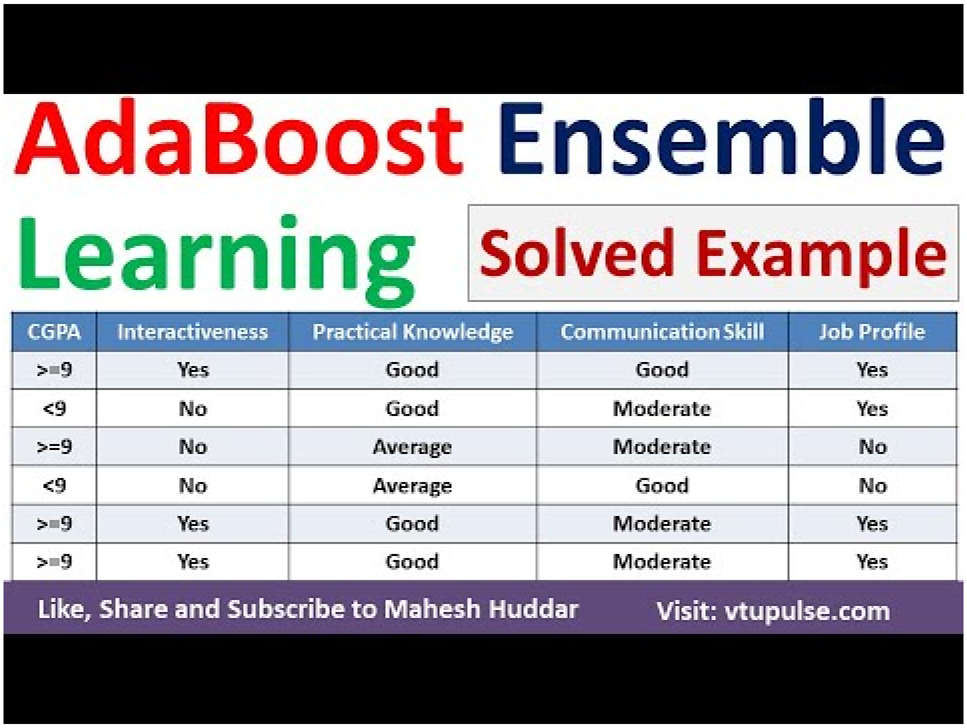
🔶 Weighted Voting Advantage

- Stronger learners influence more, improving robustness and generalization.
- Weak learners are still useful, but their impact is reduced.
- **AdaBoost** is robust to outliers and noisy data, as it focuses on misclassified points

Conclusion

- **AdaBoost** can handle high-dimensional data and non-linear relationships.
- AdaBoost is a powerful ensemble learning algorithm that combines multiple weak learners to - create a strong predictive model.
- It is robust to outliers and noisy data, and it can handle both classification and regression tasks.
- The weighted voting mechanism allows for the emphasis of misclassified points, improving the overall accuracy of the model

Refer https://www.youtube.com/watch?v=l3DzJBb3MaE


AdaBoost Ensemble
Learning
Solved Example

CGPA	Interactiveness	Practical Knowledge	Communication Skill	Job Profile
>=9	Yes	Good	Good	Yes
<9	No	Good	Moderate	Yes
>=9	No	Average	Moderate	No
<9	No	Average	Good	No
>=9	Yes	Good	Moderate	Yes
>=9	Yes	Good	Moderate	Yes

Like, Share and Subscribe to Mahesh Huddar
Visit: vtupulse.com