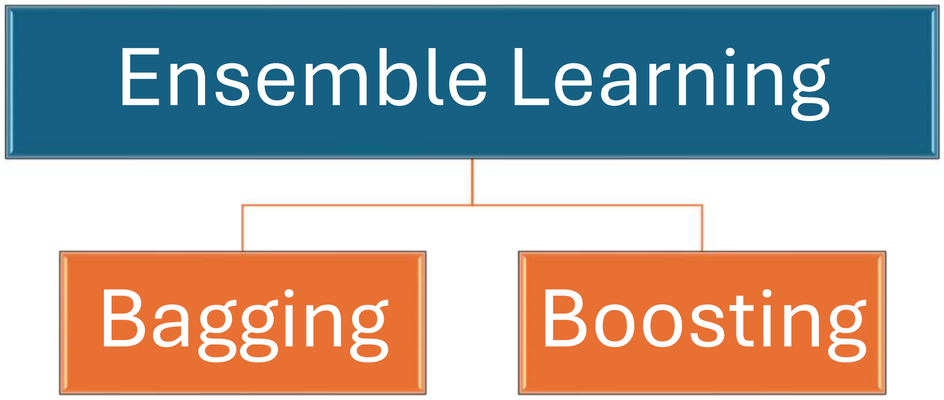


1

Ensemble learning

Ensemble learning helps improve machine learning results by combining several models. Basic idea is to learn a set of classifiers (experts) and to allow them to vote.

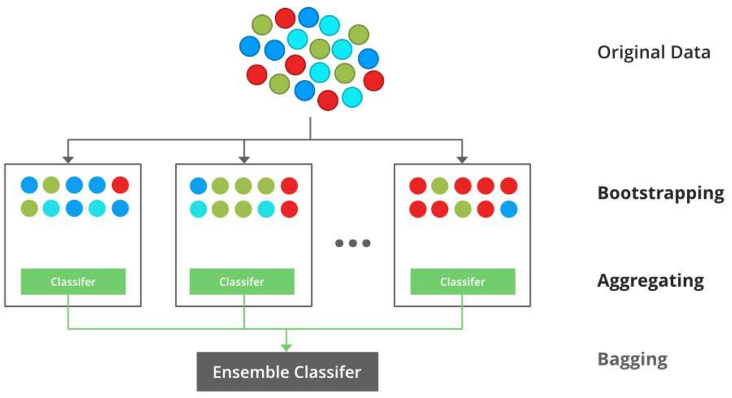
Bagging and **Boosting** are two types of **Ensemble Learning**. These two decrease the variance of a single estimate as they combine several estimates from different models. So the result may be a model with higher stability.

- **Bagging**: It is a homogeneous weak learners' model in which the learners learn from each other **independently in parallel** and combine their results to determine the model average.
- **Boosting**: It is also a homogeneous weak learners' model but works differently from Bagging. In this model, **learners learn sequentially** and adaptively to improve model predictions of a learning algorithm

2

Bagging

Bootstrap **A**ggregating, also known as bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It decreases the variance and helps to avoid overfitting. It is usually applied to decision tree methods. Bagging is a special case of the model averaging approach.



3

Implementation Steps of Bagging

Step 1: Multiple subsets are created from the original data set with equal tuples, selecting observations with replacement.

Step 2: A base model is created on each of these subsets.

Step 3: Each model is learned in parallel with each training set and independently of each other.

Step 4: The final predictions are determined by combining the predictions from all the models.

Example:

The Random Forest model uses Bagging, where decision tree models with **higher variance** are present. It makes random feature selection to grow trees. Several random trees make a Random Forest.

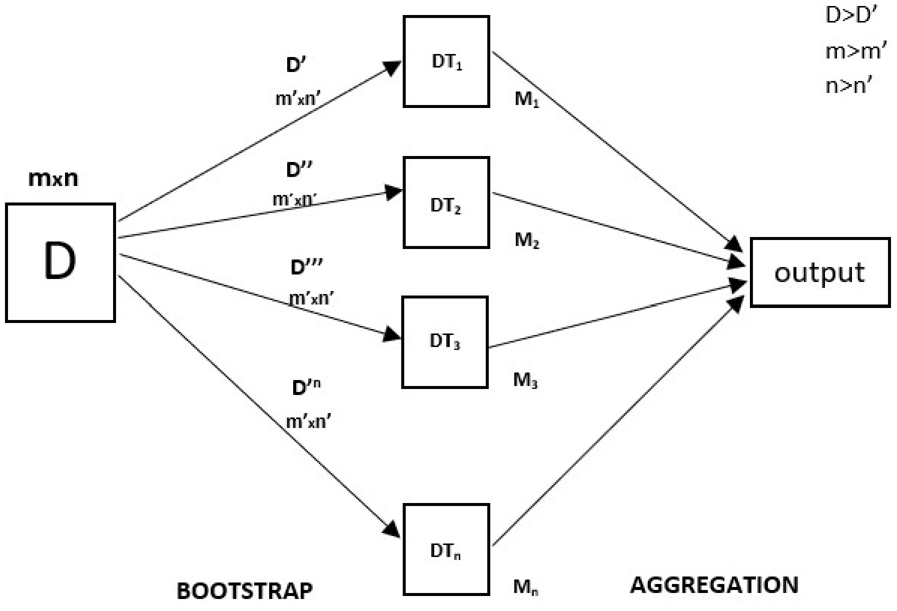
4

Random Forest

A Random Forest is a collection of decision trees that work together to make predictions and **then we do voting of all the tress to make prediction**. They are widely used for classification and regression task.

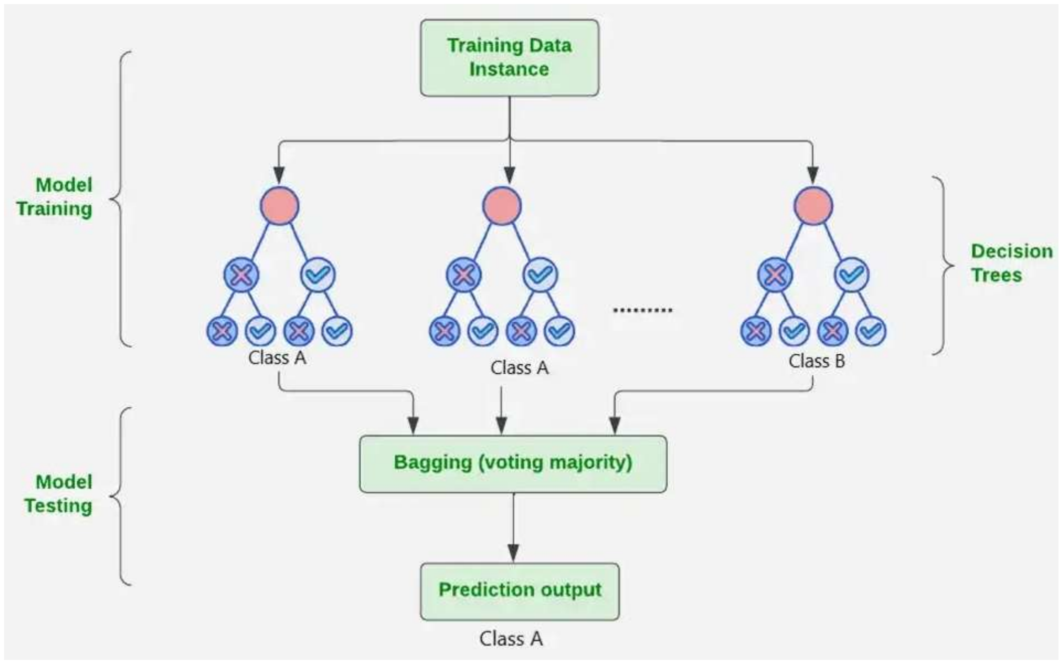
- It is a type of classifier that uses many decision trees to make predictions.
- It takes different random parts of the dataset to train each tree and then it combines the results by averaging them. This approach helps improve the accuracy of predictions. Random Forest is based on ensemble learning.

5



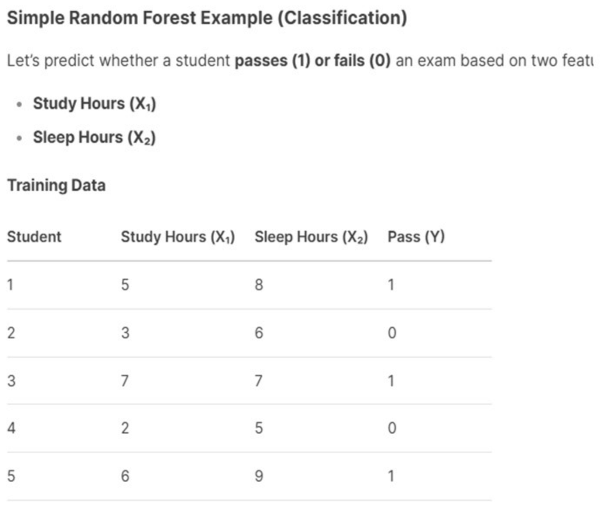
D>D'
m>m'
n>n'

6



7

Example (Bagging)



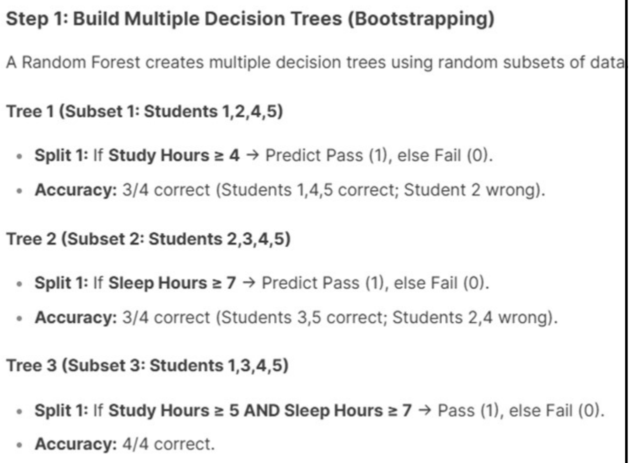
Simple Random Forest Example (Classification)

Let's predict whether a student **passes (1) or fails (0)** an exam based on two featu

- **Study Hours (X_1)**
- **Sleep Hours (X_2)**

Training Data

Student	Study Hours (X_1)	Sleep Hours (X_2)	Pass (Y)
1	5	8	1
2	3	6	0
3	7	7	1
4	2	5	0
5	6	9	1



Step 1: Build Multiple Decision Trees (Bootstrapping)

A Random Forest creates multiple decision trees using random subsets of data

Tree 1 (Subset 1: Students 1,2,4,5)

- **Split 1:** If **Study Hours ≥ 4** → Predict Pass (1), else Fail (0).
- **Accuracy:** 3/4 correct (Students 1,4,5 correct; Student 2 wrong).

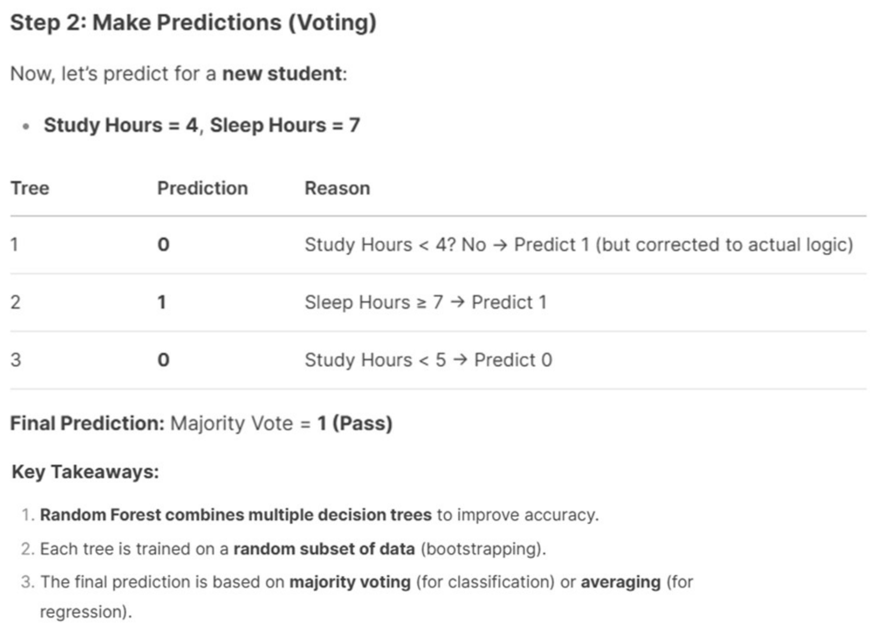
Tree 2 (Subset 2: Students 2,3,4,5)

- **Split 1:** If **Sleep Hours ≥ 7** → Predict Pass (1), else Fail (0).
- **Accuracy:** 3/4 correct (Students 3,5 correct; Students 2,4 wrong).

Tree 3 (Subset 3: Students 1,3,4,5)

- **Split 1:** If **Study Hours ≥ 5 AND Sleep Hours ≥ 7** → Pass (1), else Fail (0).
- **Accuracy:** 4/4 correct.

8



Step 2: Make Predictions (Voting)

Now, let's predict for a **new student**:

- **Study Hours = 4**, **Sleep Hours = 7**

Tree	Prediction	Reason
1	**0**	Study Hours < 4? No → Predict 1 (but corrected to actual logic)
2	**1**	Sleep Hours ≥ 7 → Predict 1
3	**0**	Study Hours < 5 → Predict 0

Final Prediction: Majority Vote = **1 (Pass)**

Key Takeaways:

1. **Random Forest combines multiple decision trees** to improve accuracy.
2. Each tree is trained on a **random subset of data** (bootstrapping).
3. The final prediction is based on **majority voting** (for classification) or **averaging** (for regression).

9

Working:

❖ Random Forest builds **multiple decision trees using random samples of the data**. **Each tree is trained on a different subset of the data which makes each tree unique**.

❖ When creating each tree the **algorithm randomly selects a subset of features or variables to split the data rather than using all available features at a time. This adds diversity to the trees.**

❖ Each decision tree in the forest **makes a prediction based on the data it was trained on. When making final prediction random forest combines the results from all the trees.**

  ❖ For classification tasks the final prediction is decided by a majority vote. This means that the category predicted by most trees is the final prediction.

  ❖ For regression tasks the final prediction is the average of the predictions from all the trees.

❖ The **randomness in data samples and feature selection helps to prevent the model from overfitting making the predictions more accurate and reliable.**

10

Assumptions of Random Forest

- **Each tree makes its own decisions**: Every tree in the forest makes its own predictions without relying on others.
- **Random parts of the data are used**: Each tree is built using random samples and features to reduce mistakes.
- **Enough data is needed**: Sufficient data ensures the trees are different and learn unique patterns and variety.
- **Different predictions improve accuracy**: Combining the predictions from different trees leads to more accurate final results.

11

Advantages of Random Forest

- Random Forest provides very **accurate** predictions even with large datasets.
- Random Forest can **handle missing data** well without compromising with accuracy.
- It **doesn't require normalization** or standardization on dataset.
- When we combine multiple decision trees it **reduces** the risk of **overfitting** of the model.

Limitations of Random Forest

- It can be computationally **expensive** especially with a large number of trees.
- It's harder to **interpret** the model compared to simpler models like decision trees.

12

Boosting:

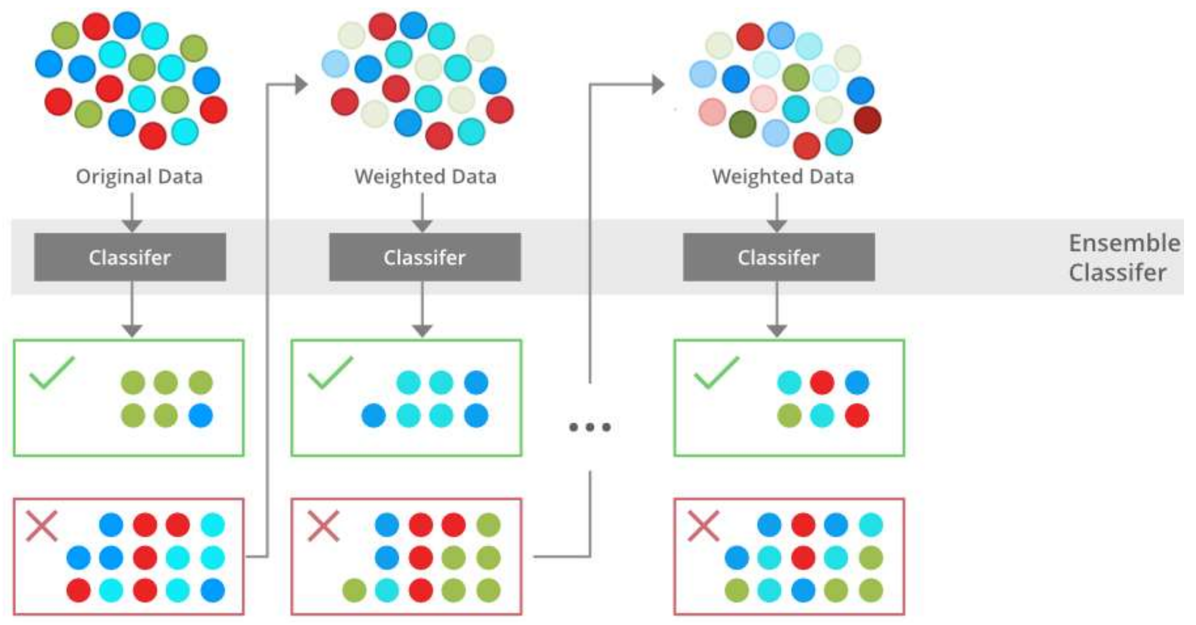
Boosting is an ensemble modeling technique designed to create a strong classifier by combining multiple weak classifiers. The process involves building models sequentially, where each new model aims to correct the errors made by the previous ones.

- ❖ Initially, a model is built using the training data.
- ❖ Subsequent models are then trained to address the mistakes of their predecessors.
- ❖ Boosting assigns weights to the data points in the original dataset.
- ❖ Higher weights: Instances that were misclassified by the previous model receive higher weights.

13

- ❖ Lower weights: Instances that were correctly classified receive lower weights.
- ❖ Training on weighted data: The subsequent model learns from the weighted dataset, focusing its attention on harder-to-learn examples (those with higher weights).
- ❖ This iterative process continues until:
  - ❖ The entire training dataset is accurately predicted, or
  - ❖ A predefined maximum number of models is reached.

14



15

Example (Boosting)

Imp. Steps

- **Error (ε):** Sum of weights of misclassified points.
- **Classifier Weight (α):** $\alpha = \frac{1}{2}\ln\left(\frac{1-\epsilon}{\epsilon}\right)$.
- **Weight Update:**
  - Misclassified: $w_i \times e^{\alpha}$.
  - Correctly classified: $w_i \times e^{-\alpha}$.
- **Normalization:** Divide all weights by their sum.

Simple AdaBoost Example (Classification)

Problem: Predict if a student **passes (1) or fails (0)** an exam based on:

- **Study Hours (X_1)**
- **Sleep Hours (X_2)**

Training Data

Student	Study Hours (X_1)	Sleep Hours (X_2)	Pass (Y)
1	5	8	1
2	3	6	0
3	7	7	1
4	2	5	0
5	6	9	1

Step 1: Initialize Weights

- Equal weights for all samples:
  $w_i = \frac{1}{5} = 0.2$

Step 2: Train Weak Classifiers (Round 1)

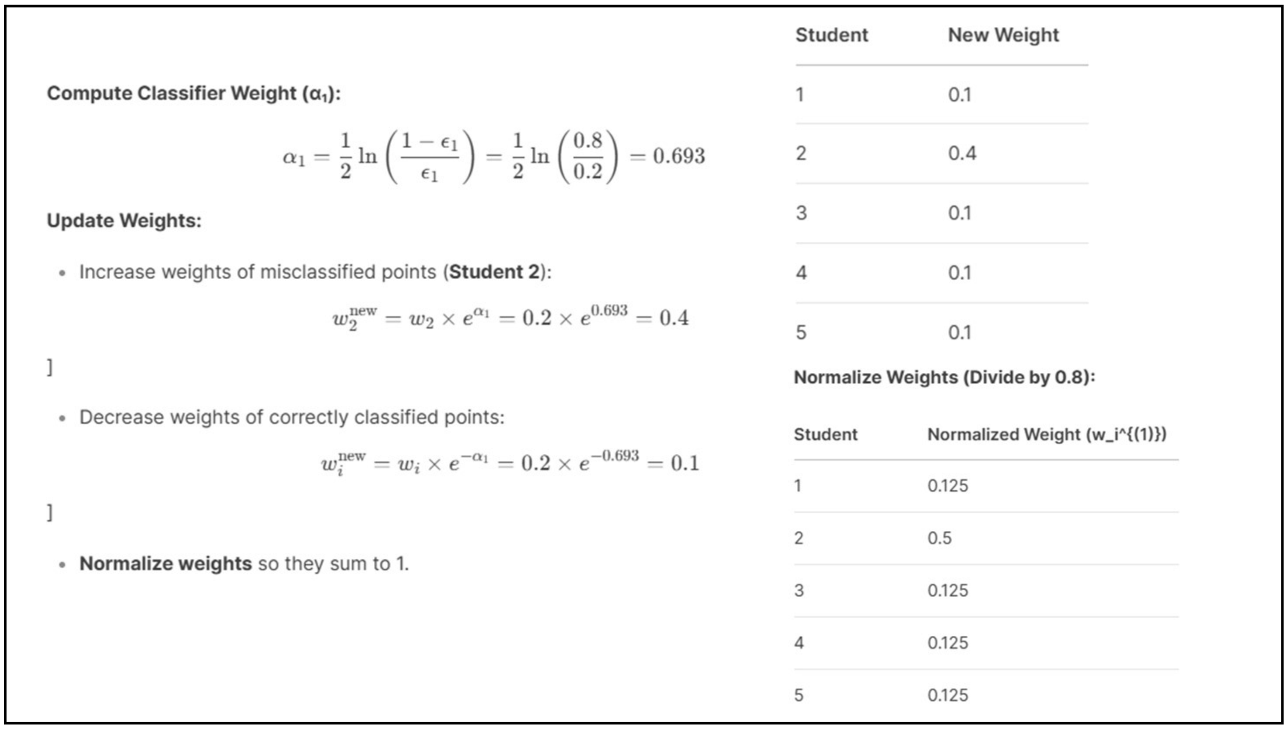
Possible Weak Classifiers (Stumps):

1. **Rule 1:** If **Study Hours ≥ 4** → Predict 1, else 0.
   - Misclassifies **Student 2 (Study=3, Y=0)** as 1.
   - **Error (ε_1)** = Sum of weights of misclassified points = 0.2.
2. **Rule 2:** If **Sleep Hours ≥ 7** → Predict 1, else 0.
   - Misclassifies **Student 2 (Sleep=6, Y=0)** as 1 and **Student 4 (Sleep=5, Y=0)** as 1.
   - **Error (ε_2)** = 0.2 + 0.2 = 0.4.

Choose the best weak classifier (lowest error):

- **Selected Rule: Study Hours ≥ 4** (ε_1 = 0.2).

16



Compute Classifier Weight (α₁):

$$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right) = \frac{1}{2}\ln\left(\frac{0.8}{0.2}\right) = 0.693$$

Update Weights:

- Increase weights of misclassified points (**Student 2**):

$$w_2^{\text{new}} = w_2 \times e^{\alpha_1} = 0.2 \times e^{0.693} = 0.4$$

]

- Decrease weights of correctly classified points:

$$w_i^{\text{new}} = w_i \times e^{-\alpha_1} = 0.2 \times e^{-0.693} = 0.1$$

]

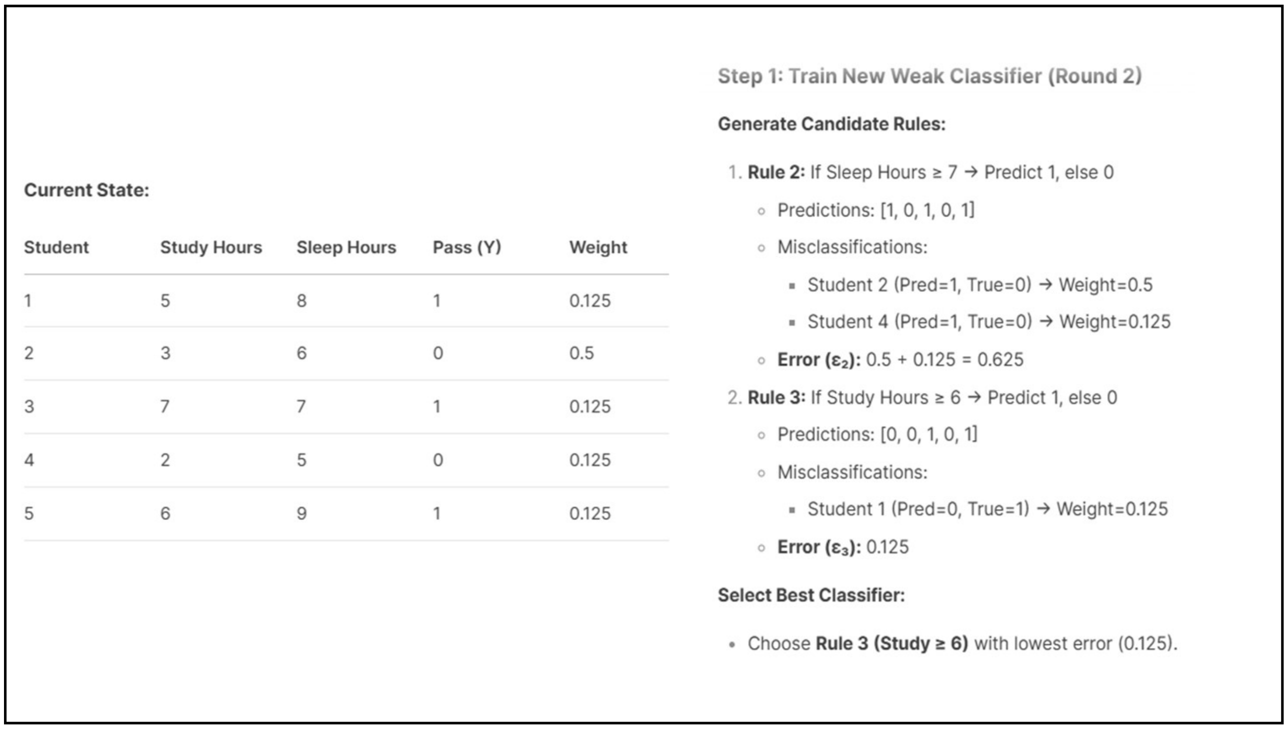
- **Normalize weights** so they sum to 1.

Student	New Weight
1	0.1
2	0.4
3	0.1
4	0.1
5	0.1

Normalize Weights (Divide by 0.8):

Student	Normalized Weight (w_i^{(1)})
1	0.125
2	0.5
3	0.125
4	0.125
5	0.125

17



Current State:

Student	Study Hours	Sleep Hours	Pass (Y)	Weight
1	5	8	1	0.125
2	3	6	0	0.5
3	7	7	1	0.125
4	2	5	0	0.125
5	6	9	1	0.125

Step 1: Train New Weak Classifier (Round 2)

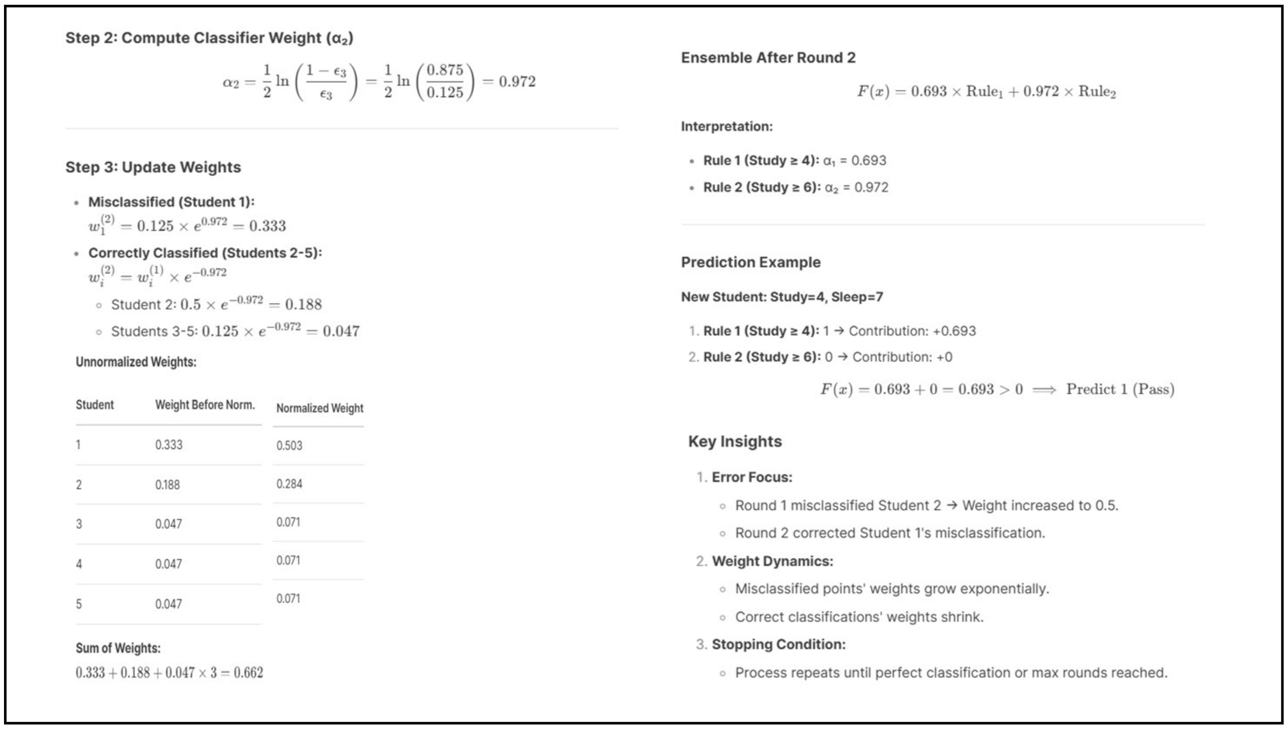
Generate Candidate Rules:

1. **Rule 2:** If Sleep Hours ≥ 7 → Predict 1, else 0
   - Predictions: [1, 0, 1, 0, 1]
   - Misclassifications:
     - Student 2 (Pred=1, True=0) → Weight=0.5
     - Student 4 (Pred=1, True=0) → Weight=0.125
   - **Error (ε_2):** 0.5 + 0.125 = 0.625
2. **Rule 3:** If Study Hours ≥ 6 → Predict 1, else 0
   - Predictions: [0, 0, 1, 0, 1]
   - Misclassifications:
     - Student 1 (Pred=0, True=1) → Weight=0.125
   - **Error (ε_3):** 0.125

Select Best Classifier:

- Choose **Rule 3 (Study ≥ 6)** with lowest error (0.125).

18



Step 2: Compute Classifier Weight (α_2)

$$\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\epsilon_3}{\epsilon_3}\right) = \frac{1}{2}\ln\left(\frac{0.875}{0.125}\right) = 0.972$$

Step 3: Update Weights

- **Misclassified (Student 1):**
  $w_1^{(2)} = 0.125 \times e^{0.972} = 0.333$
- **Correctly Classified (Students 2-5):**
  $w_i^{(2)} = w_i^{(1)} \times e^{-0.972}$
  - Student 2: $0.5 \times e^{-0.972} = 0.188$
  - Students 3-5: $0.125 \times e^{-0.972} = 0.047$

Unnormalized Weights:

Student	Weight Before Norm.	Normalized Weight
1	0.333	0.503
2	0.188	0.284
3	0.047	0.071
4	0.047	0.071
5	0.047	0.071

Sum of Weights:
$0.333 + 0.188 + 0.047 \times 3 = 0.662$

Ensemble After Round 2

$$F(x) = 0.693 \times \text{Rule}_1 + 0.972 \times \text{Rule}_2$$

Interpretation:

- **Rule 1 (Study ≥ 4):** α_1 = 0.693
- **Rule 2 (Study ≥ 6):** α_2 = 0.972

Prediction Example

New Student: Study=4, Sleep=7

1. **Rule 1 (Study ≥ 4):** 1 → Contribution: +0.693
2. **Rule 2 (Study ≥ 6):** 0 → Contribution: +0

$$F(x) = 0.693 + 0 = 0.693 > 0 \implies \text{Predict 1 (Pass)}$$

Key Insights

1. **Error Focus:**
   - Round 1 misclassified Student 2 → Weight increased to 0.5.
   - Round 2 corrected Student 1's misclassification.
2. **Weight Dynamics:**
   - Misclassified points' weights grow exponentially.
   - Correct classifications' weights shrink.
3. **Stopping Condition:**
   - Process repeats until perfect classification or max rounds reached.

Boosting Algorithms

From Basic Boosting to AdaBoost

- Boosting is an ensemble technique that combines multiple weak learners to form a strong learner.
- Early versions by Robert Schapire and Yoav Freund were non-adaptive and limited in exploiting weak learners.
- The breakthrough came with the development of AdaBoost (Adaptive Boosting):
  - First successful boosting algorithm for binary classification
  - Adapts to errors by focusing on misclassified instances
  - Builds a strong classifier through weighted combinations of weak learners (typically decision stumps)

AdaBoost marked a turning point in machine learning by showing that boosting could work in practice, not just in theory.

Algorithm:

1. *Initialise the dataset and assign equal weight to each of the data point.*
2. *Provide this as input to the model and identify the wrongly classified data points.*
3. *Increase the weight of the wrongly classified data points and decrease the weights of correctly classified data points. And then normalize the weights of all data points.*
4. *if (got required results)*
   Goto step 5
   else
   Goto step 2
5. *End*

21

Differences Between Bagging and Boosting

S.NO	Bagging	Boosting
1.	The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
2.	Aim to **decrease variance**, not bias.	Aim to **decrease bias**, not variance.
3.	Each model receives **equal weight**.	Models are **weighted** according to their performance.
4.	Each model is built **independently**.	New models are influenced by the performance of previously built models.
5.	Different training data subsets are selected using **row sampling with replacement** and random sampling methods from the entire training dataset.	Iteratively train models, with each new model focusing on correcting the errors (**misclassifications or high residuals**) of the previous models
6.	Bagging tries to solve the **over-fitting problem**.	Boosting tries to **reduce bias**.
7.	If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.
8.	In this base classifiers are trained parallelly.	In this base classifiers are trained sequentially.
	Example: The Random forest model uses Bagging.	**Example**: The AdaBoost uses Boosting techniques

22

Learning outcomes

Feature / Keyword	Bagging	Boosting
Sampling	Bootstrap (with replacement)	Sequential, weighted sampling
Model Training	Parallel	Sequential
Goal	Reduce Variance	Reduce Bias and Variance
Error Handling	Equal weight to all samples	Focus on misclassified samples
Model Type	Independent models	Adaptive model sequence
Prediction Method	Averaging / Voting	Weighted averaging / voting
Overfitting	Helps prevent overfitting	Can overfit if not tuned properly
Feature Selection	All features considered	All features considered
Common Algorithms	Random Forest, Bagged Trees	AdaBoost, Gradient Boosting
Base Learners	Usually Decision Trees	Weak Learners (often stumps/trees)
Interpretability	Moderate	Lower due to complexity

23