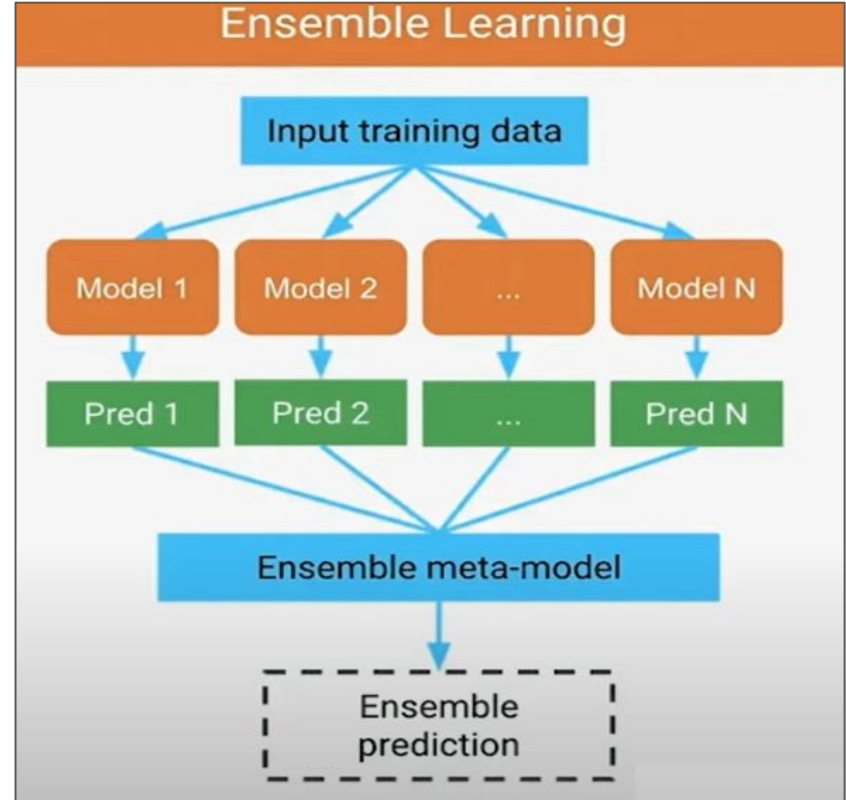


Ensemble Learning

ENSEMBLE TECHNIQUES AND UNSUPERVISED LEARNING Combining multiple learners: Model combination schemes, Voting, Ensemble Learning - bagging, boosting, stacking,

What is Ensemble Learning?

- It is a supervised learning technique used in machine learning to improve the overall performance by combining the predictions from the multiple models
- Each model has its own strength and weakness. By combining the different models, can lead to improved performance and generalization.



Types of Ensemble methods

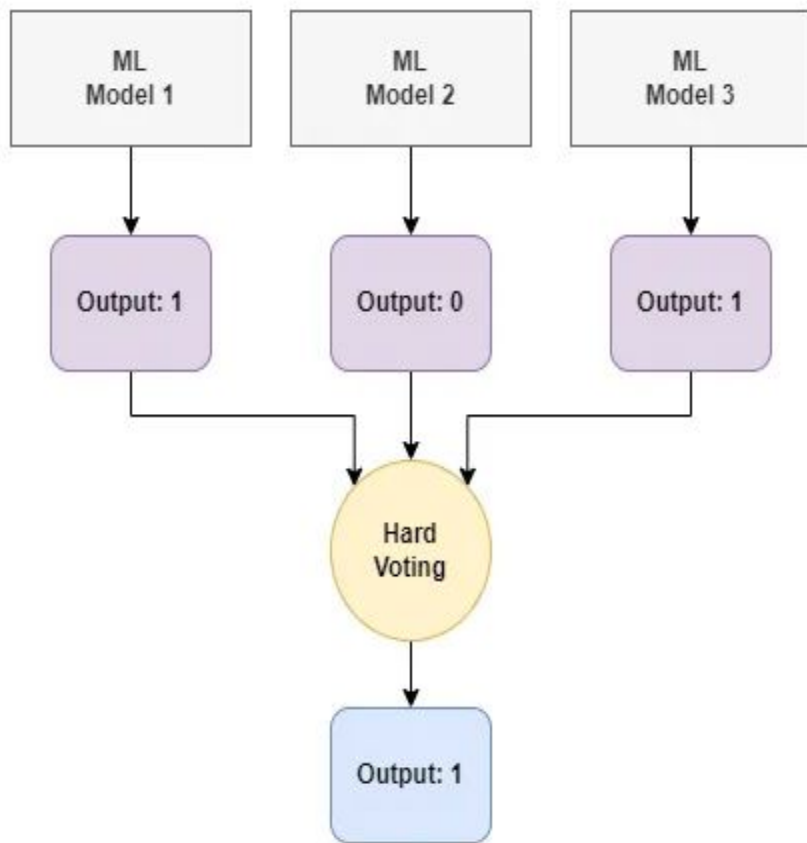
- Voting(Averaging)
- Bootstrap aggregation(Bagging)
- Random forest
- Boosting
- Stacked Generalization(Blending)

Voting

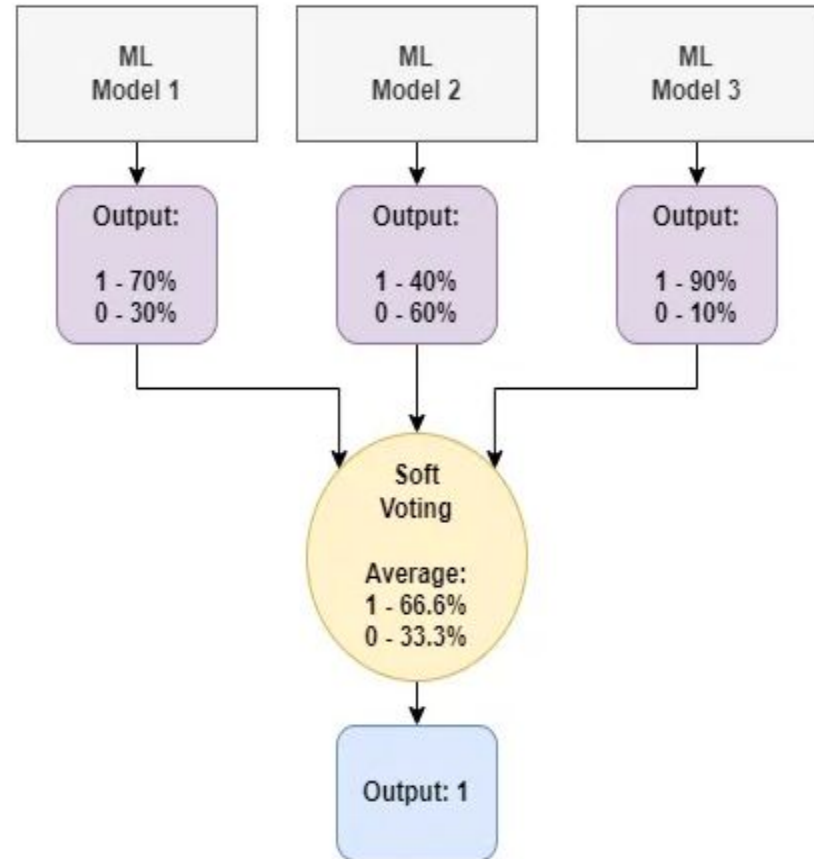
- A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output.
- It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting.
- The idea is instead of creating separate dedicated models and finding the accuracy for each of them, we create a single model which trains by these models and predicts output based on their combined majority of voting for each output class.

Types of voting classifier

- Hard voting classifier
 - i. In hard voting, the predicted output class is a class with the highest majority of votes i.e the class which had the highest probability of being predicted by each of the classifiers.
 - ii. Suppose three classifiers predicted the *output class*(A, A, B), so here the majority predicted A as output. Hence A will be the final prediction.
- Soft voting classifier
 - i. In soft voting, the output class is the prediction based on the average of probability given to that class.
 - ii. Suppose given some input to three models, the prediction probability for class A = (0.30, 0.47, 0.53) and B = (0.20, 0.32, 0.40).
 - iii. So the average for class A is 0.4333 and B is 0.3067, the winner is clearly class A because it had the highest probability averaged by each classifier.



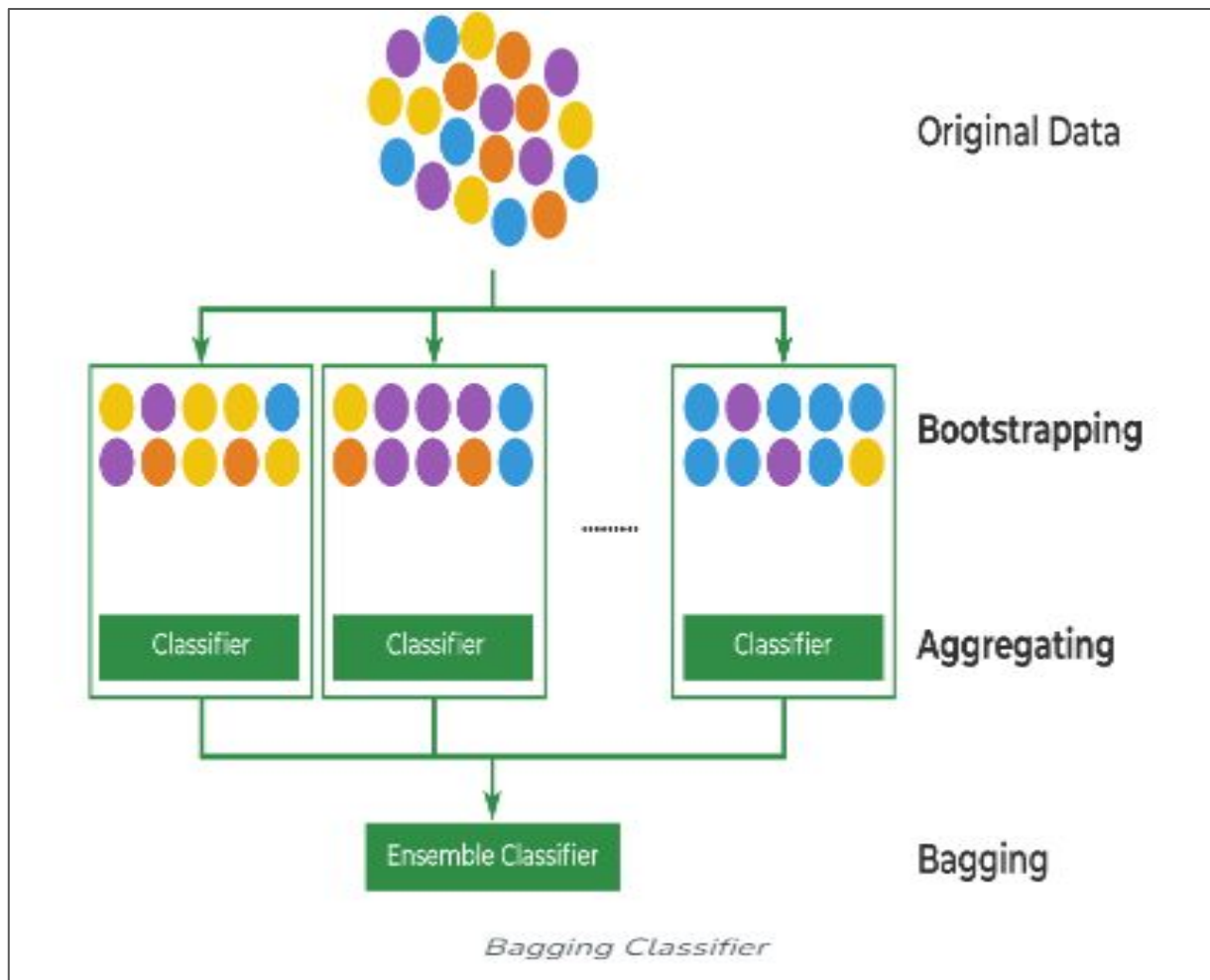
Depiction of Hard Voting in Ensemble Machine Learning



Depiction of Soft Voting in Ensemble Machine Learning

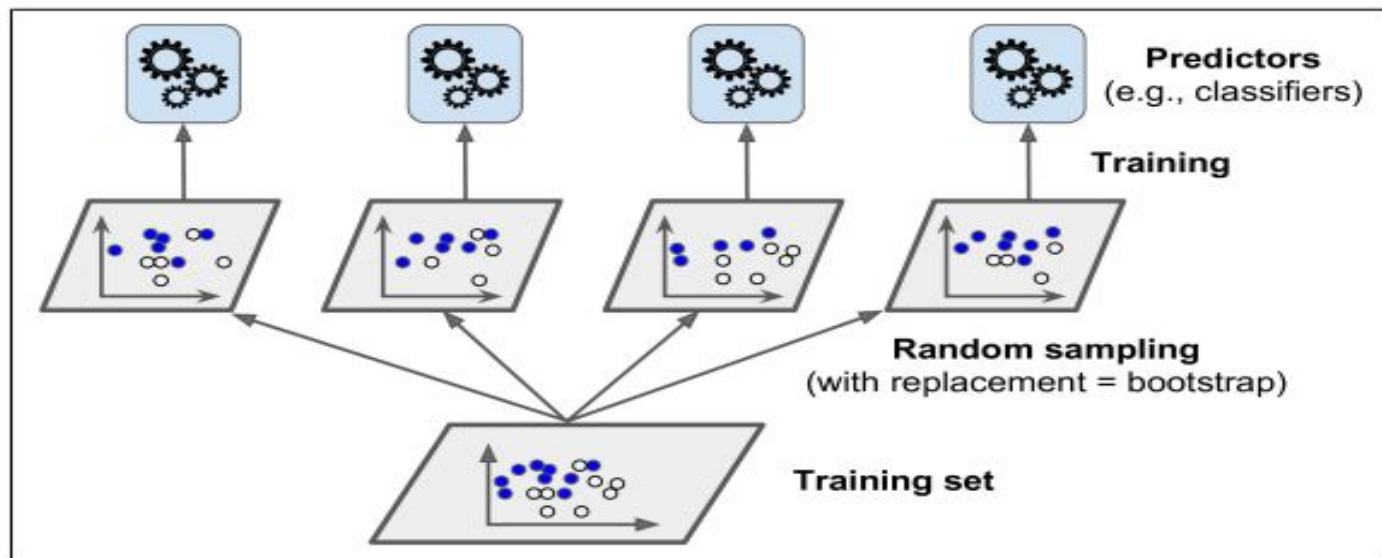
Ensemble Learning-Bagging/Bootstrap Aggregation

- Bagging, an abbreviation for Bootstrap Aggregating, is a machine learning ensemble strategy for enhancing the reliability and precision of predictive models.
- Bagging involves training multiple base learners on different subsets of the training data, typically created through random sampling with replacement.
- It can be used for both classification and regression



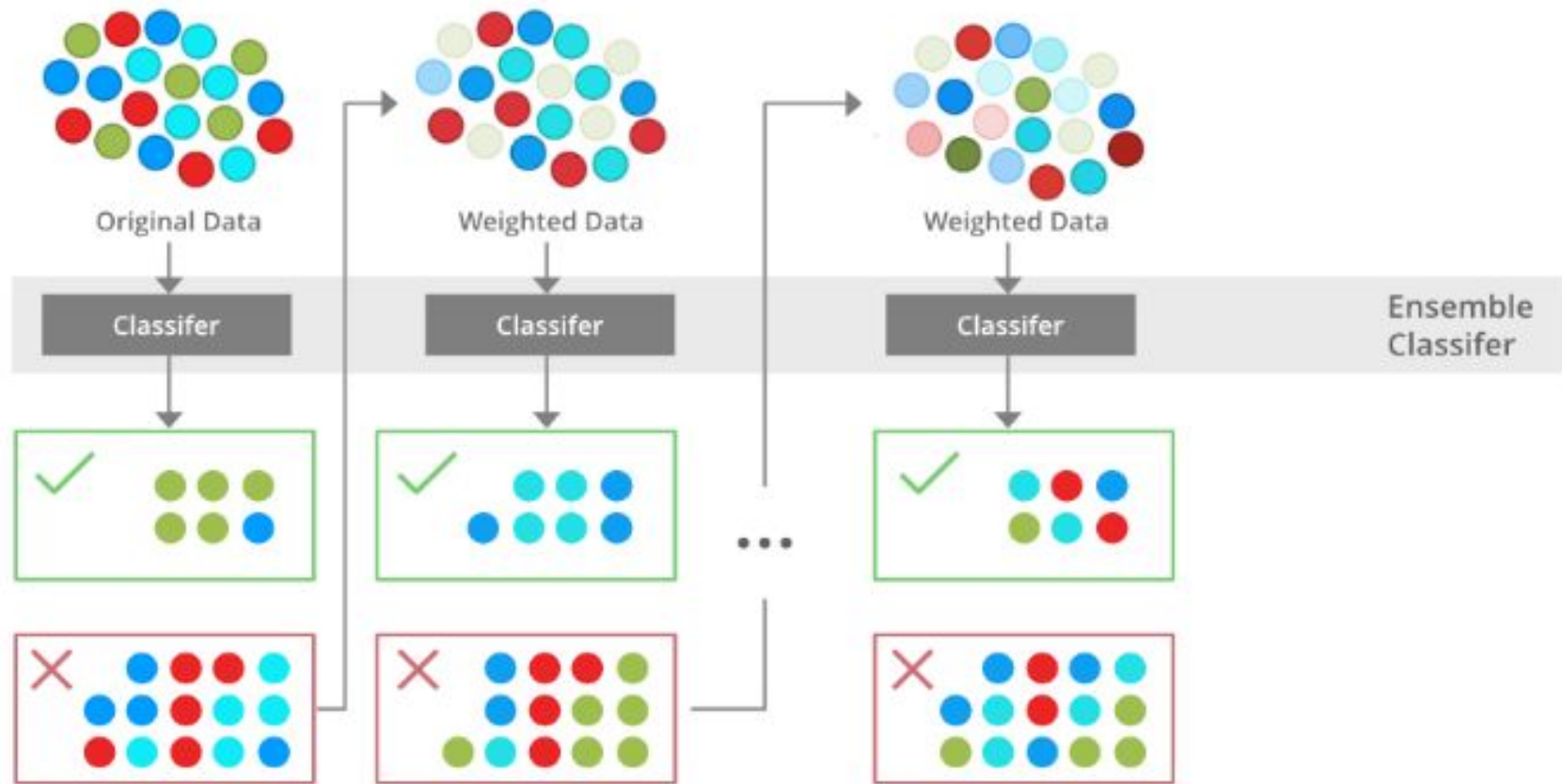
- Multiple base models are trained independently and in parallel on different subsets of the training data.
- Each subset is generated using bootstrap sampling, in which data points are picked at random with replacement.
- In the case of the bagging classifier, the final prediction is made by aggregating the predictions of the all-base model using majority voting.
- In the models of regression, the final prediction is made by averaging the predictions of the all-base model, and that is known as bagging regression.
- Bagging helps improve accuracy and reduce overfitting, especially in models that have high variance.

- Use the same training algorithm for every predictor, but to train them on different random subsets of the training set.
- When sampling is performed with replacement, this method is called bagging (short for bootstrap aggregating).
- When sampling is performed without replacement, it is called pasting.



Ensemble Learning-Boosting

- Boosting originally called Hypothesis Boosting
- **Boosting** is an ensemble modeling technique that attempts to **build a strong classifier from the number of weak classifiers.**
- It is done by building a model by using **weak models in series.**
- Firstly, a **model** is built from the **training data.**
- Then the **second model** is built which tries to **correct the errors present in the first model.**
- This **procedure is continued** and models are added until either the **complete training data set is predicted correctly** or the **maximum number of models are added.**



Training a boosting model

Types of Boosting Algorithms

There are various boosting methods available namely

- Adaboost
- Gradient Boosting
- XGBoost
- CatBoost

AdaBoost

- AdaBoost short for Adaptive Boosting is an ensemble learning used in machine learning for **classification and regression problems**.
- The main idea behind AdaBoost is to iteratively train the weak classifier on the training dataset with each successive classifier giving more weightage to the data points that are misclassified.
- The final AdaBoost model is decided by combining all the weak classifier that has been used for training with the weightage given to the models according to their accuracies.
- The weak model which has the highest accuracy is given the highest weightage while the model which has the lowest accuracy is given a lower weightage.

Step 1 – Initialize the weights

Step 2 – Train weak classifiers

Step 3 – Calculate the error rate and importance of each weak model M_k

Step 4 – Update data point weight for each data point W_i

Step 5 – Normalize the Instance weight

Step 6 – Repeat steps 2-5 for K iterations

Adaboost Problem

- Apply the Adaboost algorithm for the following dataset and classify the dataset with job offer as target attribute.
- Use 4 decision stumps for each of the four attributes

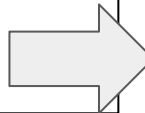
CGPA	Interactiveness	Practical Knowledge	Communication Skill	Job Profile
≥ 9	Yes	Good	Good	Yes
< 9	No	Good	Moderate	Yes
≥ 9	No	Average	Moderate	No
< 9	No	Average	Good	No
≥ 9	Yes	Good	Moderate	Yes
≥ 9	Yes	Good	Moderate	Yes

Step 1: Initial Weight assigned to each item.

- In this given dataset ,we have six training instances,so the initial weight is $\frac{1}{6}$

Step 2: Iterate for each weak classifier

- Decision Stump for CGPA
 - Train the Decision stump H_{CGPA} with a random bootstrap sample from the training dataset T
 - As the considered dataset has very few examples,so we are considering all examples



First Decision Stump

CGPA	Predicted Job offer	Actual Job offer	Weight
≥ 9		Yes	$\frac{1}{6}$
< 9		yes	$\frac{1}{6}$
≥ 9		No	$\frac{1}{6}$
< 9		No	$\frac{1}{6}$
≥ 9		yes	$\frac{1}{6}$
≥ 9		yes	$\frac{1}{6}$

- If **CGPA ≥ 9** , the data instance is predicted to have “**Job Offer**” as “**Yes**” ,Else “**NO**”
- From the table, we observe two instances are wrongly classified ,so we need to **compute the weighted error**

ϵ_{CGPA} of H_{CGPA} on current training dataset T:

$$\epsilon_i = \sum_{j=1}^N H_i(d_j) wt(d_j)$$

Where

$H_i(d_j) = 0$ (Correct Prediction)

$H_i(d_j) = 1$ (Wrong Prediction)

$$\epsilon_{CGPA} = 2 * \frac{1}{6} = 0.333$$

CGPA	Predicted Job offer	Actual Job offer	Weight
≥ 9	YES ✓	Yes	1/6
< 9	NO ✗	yes	1/6
≥ 9	YES ✗	No	1/6
< 9	NO ✓	No	1/6
≥ 9	YES ✓	yes	1/6
≥ 9	YES ✓	yes	1/6

Step 2 (c): Compute the weight of each weak classifier:

$$\underline{\alpha_{CGPA}} = \frac{1}{2} \frac{\ln(1 - \varepsilon_{CGPA})}{\varepsilon_{CGPA}}$$

$$\underline{\alpha_{CGPA}} = \frac{1}{2} \frac{\ln(1 - 0.333)}{0.333}$$

$$\alpha_{CGPA} = 0.347$$

CGPA	Predicted Job offer	Actual Job offer	Weight
≥ 9	YES ✓	Yes	1/6
< 9	NO ✗	yes	1/6
≥ 9	YES ✗	No	1/6
< 9	NO ✓	No	1/6
≥ 9	YES ✓	yes	1/6
≥ 9	YES ✓	yes	1/6

Step 2 (d): Calculate the normalizing factor

$$\underline{Z_{CGPA}}$$

$$\underline{Z_{CGPA}} = \frac{\text{wt}(\text{Correct Classified Instance})}{\text{No of Correct Classification}} * e^{-\alpha_{CGPA}} + \frac{\text{wt}(\text{Wrong Classified Instance})}{\text{No of Wrong Classification}} * e^{+\alpha_{CGPA}}$$

$$Z_{CGPA} = \frac{1}{6} * 4 * e^{-0.347} + \frac{1}{6} * 2 * e^{0.347}$$

$$Z_{CGPA} = 0.9428$$

CGPA	Predicted Job offer	Actual Job offer	Weight
>=9	YES ✓	Yes	1/6
<9	NO ✗	yes	1/6
>=9	YES ✗	No	1/6
<9	NO ✓	No	1/6
>=9	YES ✓	yes	1/6
>=9	YES ✓	yes	1/6

- **Step 2 (e):** Update the weight of all data instances:

- $$\underline{wt(d_j)_{i+1}} = \frac{wt(d_j)_{CGPA} \text{ of correct Instance} * e^{-\alpha CGPA}}{\underline{Z_{CGPA}}}$$

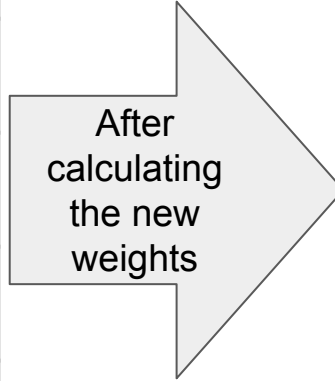
- $$wt(d_j)_{i+1} = \frac{\frac{1}{6} * e^{-0.347}}{0.9428} = \underline{0.1249}$$

- $$wt(d_j)_{i+1} = \frac{wt(d_j)_{CGPA} \text{ of Incorrect Instance} * e^{\alpha CGPA}}{Z_{CGPA}}$$

- $$wt(d_j)_{i+1} = \frac{\overset{\checkmark}{\frac{1}{6}} * e^{0.347}}{0.9428} = \underline{0.2501}$$

CGPA	Predicted Job offer	Actual Job offer	Weight
>=9	YES ✓	Yes	1/6
<9	NO ✗	yes	1/6
>=9	YES ✗	No	1/6
<9	NO ✓	No	1/6
>=9	YES ✓	yes	1/6
>=9	YES ✓	yes	1/6

CGPA	Predicted Job offer	Actual Job offer	Weight
≥ 9	YES	Yes	1/6
< 9	NO	yes	1/6
≥ 9	YES	No	1/6
< 9	NO	No	1/6
≥ 9	YES	yes	1/6
≥ 9	YES	yes	1/6




CGPA	Predicted Job offer	Actual Job offer	Weight
≥ 9	YES	Yes	0.1249
< 9	NO	yes	0.2501
≥ 9	YES	No	0.2501
< 9	NO	No	0.1249
≥ 9	YES	yes	0.1249
≥ 9	YES	yes	0.1249

Decision Stump for “Interactiveness”

Step 2 (a): Train the Decision Stump $H_{Interact}$ with the sample obtained in the previous weak classifier $H_{Interact}$

If **Interactiveness** is **Yes**, the data instance is predicted to have '**Job Offer**' as '**Yes**' else '**No**'.

Second Decision Stump

Interactiveness	Predicted Job offer	Actual Job offer	Weight
Yes	YES	Yes	0.1249
No	NO 	yes	0.2501
No	NO	No	0.2501
No	NO	No	0.1249
Yes	YES	yes	0.1249
Yes	NO	yes	0.1249

- **Step 2 (b):** Compute the **weighted error**

$\epsilon_{Interact}$ of $H_{Interact}$ on current training dataset T:

- $\epsilon_i = \sum_{j=1}^N H_i(d_j)wt(d_j)$

Where

$H_i(d_j) = 0$ for Correct prediction

$H_i(d_j) = 1$ for Wrong prediction

$\epsilon_{Interact} = 1 * 0.2501 = 0.2501$

Interact iveness	Predict ed Job offer	Actual Job offer	Weight
Yes	YES	Yes	0.1249
No	NO	yes	0.2501
No	NO	No	0.2501
No	NO	No	0.1249
Yes	YES	yes	0.1249
Yes	NO	yes	0.1249

Step 2 (c): Compute the weight of
each weak classifier:

$$\alpha_{Interact} = \frac{1}{2} \frac{\ln(1 - \varepsilon_{Interact})}{\varepsilon_{Interact}}$$

$$\alpha_{Interact} = \frac{1}{2} \frac{\ln(1 - 0.2501)}{0.2501}$$

$$\alpha_{Interact} = \underline{0.5490}$$

Interact iveness	Predict ed Job offer	Actual Job offer	Weight
Yes	YES	Yes	0.1249
No	NO	yes	0.2501
No	NO	No	0.2501
No	NO	No	0.1249
Yes	YES	yes	0.1249
Yes	NO	yes	0.1249

- **Step 2 (d):** Calculate the normalizing factor

$Z_{Interact}$

- $\underline{Z_{Interact}} = \text{wt}(\text{Correct Classified Instance}) * \text{No of Correct Classification} * e^{-\alpha_{Interact}} + \text{wt}(\text{Wrong Classified Instance}) * \text{No of Wrong Classification} * e^{+\alpha_{Interact}}$
- $Z_{Interact} = \underline{0.1249} * \underline{4} * \underline{e^{-0.549}} + \underline{0.2501} * \underline{1} * \underline{e^{-0.549}} + \underline{0.2501} * \underline{1} * \underline{e^{0.549}}$
- $Z_{Interact} = \underline{0.866}$

Interact iveness	Predict ed Job offer	Actual Job offer	Weight
Yes	YES	Yes	0.1249
No	NO	yes	0.2501
No	NO	No	0.2501
No	NO	No	0.1249
Yes	YES	yes	0.1249
Yes	NO	yes	0.1249

- **Step 2 (e):** Update the weight of all data instances:

$$wt(d_j)_{i+1} = \frac{wt(d_j)_{Interact}^{of\ correct\ Instance} * e^{-\alpha_{Interact}}}{Z_{Interact}}$$

$$wt(d_j)_{i+1} = \frac{0.1249 * e^{-0.549}}{0.866} = 0.0832$$

$$wt(d_j)_{i+1} = \frac{0.2501 * e^{-0.549}}{0.866} = 0.1667$$

$$wt(d_j)_{i+1} = \frac{wt(d_j)_{Interact}^{of\ Incorrect\ Instance} * e^{\alpha_{Interact}}}{Z_{Interact}}$$

$$wt(d_j)_{i+1} = \frac{0.2501 * e^{0.549}}{0.866} = 0.5001$$

Interact iveness	Predict ed Job offer	Actual Job offer	Weight
Yes	YES	Yes	0.0832
No	NO	yes	0.5001
No	NO	No	0.1667
No	NO	No	0.0832
Yes	YES	yes	0.0832
Yes	NO	yes	0.0832

Decision Stump for “Practical Knowledge”

- **Step 2 (a):** Train the Decision Stump H_{pk} with the sample obtained in the previous weak classifier H_{pk}

If **Practical Knowledge** is **Good**, the data instance is predicted to have '**Job Offer**' as '**Yes**' else '**No**'.

Third Decision Stump

Practical Knowledge	Predicted Job offer	Actual Job offer	Weight
Good	YES	Yes	0.0832
Good	YES	yes	0.5001
Average	NO	No	0.1667
Average	NO	No	0.0832
Good	YES	yes	0.0832
Good	YES	yes	0.0832

No instances are misclassified by this

Decision Stump.

So, no need to change the weights of

the data instances







Practical Knowledge	Predicted Job offer	Actual Job offer	Weight
Good	YES	Yes	0.0832
Good	YES	yes	0.5001
Average	NO	No	0.1667
Average	NO	No	0.0832
Good	YES	yes	0.0832
Good	YES	yes	0.0832

Decision Stump for “Communication Skills”

Step 2 (a): Train the Decision Stump H_{CS} with the sample obtained in the previous weak classifier H_{CS}

If **Communication Skill** is **Good**, the data instance is predicted to have '**Job Offer**' as '**Yes**' else '**No**'.

Fourth Decision Stump

Communi cation Skills	Predict ed Job offer	Actual Job offer	Weight
Good	Yes	Yes 	0.0832
Moderate	No	yes 	0.5001
Moderate	No	No 	0.1667
Good	Yes	No 	0.0832
Moderate	No	yes 	0.0832
Moderate	No	yes 	0.0832

Step 2 (b): Compute the **weighted error**

ε_{CS} of H_{CS} on current training dataset T:







$$\varepsilon_i = \sum_{j=1}^N H_i(d_j) wt(d_j)$$

where , $H_i(d_j) = 0$ for Correct prediction

$H_i(d_j) = 1$ for Wrong prediction

$$\varepsilon_{CS} = \underline{1} * \underline{0.5001} + \underline{3} * \underline{0.0832}$$

$$\underline{\varepsilon_{CS} = 0.7497}$$







Communi cation Skills	Predict ed Job offer	Actual Job offer	Weight
Good	Yes	Yes 	0.0832
Moderate	No	yes 	0.5001
Moderate	No	No 	0.1667
Good	Yes	No 	0.0832
Moderate	No	yes 	0.0832
Moderate	No	yes 	0.0832

Step 2 (c): Compute the weight of each weak classifier:

$$\alpha_{CS} = \frac{1}{2} \frac{\ln(1 - \varepsilon_{CS})}{\varepsilon_{CS}}$$

$$\alpha_{CS} = \frac{1}{2} \frac{\ln(1 - 0.7497)}{0.7497}$$

$$\alpha_{CS} = -0.5485$$

Communi cation Skills	Predict ed Job offer	Actual Job offer	Weight
Good	Yes	Yes 	0.0832
Moderate	No	yes 	0.5001
Moderate	No	No 	0.1667
Good	Yes	No 	0.0832
Moderate	No	yes 	0.0832
Moderate	No	yes 	0.0832

Step 2 (d): Calculate the normalizing factor Z_{CS}

$$Z_{CS} = wt(Correct\ Classified\ Instance) *$$

$$No\ of\ Correct\ Classification * e^{-\alpha_{cs}} +$$

$$wt(Wrong\ Classified\ Instance) *$$







$$No\ of\ Wrong\ Classification * e^{+\alpha_{cs}}$$

$$Z_{CS} = \underline{0.0832} * \underline{1} * \underline{e^{-(-0.5485)}} + \underline{0.1667} * \underline{1} *$$

$$\underline{e^{-(-0.5485)}} + \underline{0.5001} * \underline{1} * \underline{e^{+(-0.5485)}} +$$

$$\underline{0.0832} * \underline{3} * \underline{e^{+(-0.5485)}}$$

$$Z_{CS} = \underline{0.866}$$

Communi cation Skills	Predict ed Job offer	Actual Job offer	Weight
Good	Yes	Yes 	0.0832
Moderate	No	yes 	0.5001
Moderate	No	No 	0.1667
Good	Yes	No 	0.0832
Moderate	No	yes 	0.0832
Moderate	No	yes 	0.0832

Step 2 (e): Update the weight of all data instances:

$$wt(d_j)_{i+1} = \frac{wt(d_j)_{CS} \text{ of correct Instance} * e^{-\alpha CS}}{Z_{CS}}$$







$$wt(d_j)_{i+1} = \frac{0.0832 * e^{-(-0.5485)}}{0.866} = 0.1663$$







$$wt(d_j)_{i+1} = \frac{0.1667 * e^{-(-0.5485)}}{0.866} = 0.3331$$

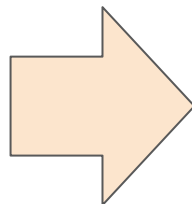
$$wt(d_j)_{i+1} = \frac{wt(d_j)_{CS} \text{ of Incorrect Instance} * e^{+\alpha CS}}{Z_{CS}}$$

$$wt(d_j)_{i+1} = \frac{0.5001 * e^{+(-0.5485)}}{0.866} = 0.3337$$

$$wt(d_j)_{i+1} = \frac{0.0832 * e^{+(-0.5485)}}{0.866} = 0.0555$$

Communi cation Skills	Predict ed Job offer	Actual Job offer	Weight
Good	Yes	Yes 	0.0832
Moderate	No	yes 	0.5001
Moderate	No	No 	0.1667
Good	Yes	No 	0.0832
Moderate	No	yes 	0.0832
Moderate	No	yes 	0.0832

Communi cation Skills	Predict ed Job offer	Actual Job offer	Weight
Good	Yes	Yes 	0.0832
Moderate	No	yes 	0.5001
Moderate	No	No 	0.1667
Good	Yes	No 	0.0832
Moderate	No	yes 	0.0832
Moderate	No	yes 	0.0832



Communi cation Skills	Predict ed Job offer	Actual Job offer	Weight
Good	Yes	Yes	0.1663
Moderate	No	yes	0.3337
Moderate	No	No	0.3331
Good	Yes	No	0.0555
Moderate	No	yes	0.0555
Moderate	No	yes	0.0555

- Step 3: Compute the final predicted value for each data instance:

Sl. No.	$\alpha_{CGPA} = 0.347$	$\alpha_{Interact} = 0.5490$	$\alpha_{CommSkill} = -0.5485$	Weighted Avg	Final Prediction
1	Yes	Yes	Yes	0.3475	
2	No	No	No		
3	Yes	No	No		
4	No	No	Yes		
5	Yes	Yes	No		
6	Yes	Yes	No		

- $H_F(d_j) = \sum_{i=1}^m \alpha_i * H_i(d_j)$

- $H_F(d_j) = \alpha_{CGPA} * Yes + \alpha_{Interact} * Yes + \alpha_{CSL} * Yes = 0.347 * 1 + 0.5490 * 1 - 0.5485 * 1 = \underline{0.3475}$

Sl. No.	$\alpha_{CGPA} = 0.347$	$\alpha_{Interact} = 0.5490$	$\alpha_{CommSkill} = -0.5485$	Weighted Avg	Final Prediction
1	Yes	Yes	Yes	0.3475	
2	No	No	No	0.0	
3	Yes	No	No		
4	No	No	Yes		
5	Yes	Yes	No		
6	Yes	Yes	No		

- $H_F(d_j) = \sum_{i=1}^m \alpha_i * H_i(d_j)$
- $H_F(d_j) = \alpha_{CGPA} * No + \alpha_{Interact} * No + \alpha_{CSL} * No = 0.347 * 0 + 0.5490 * 0 - 0.5485 * 0 = 0.0$

Similarly calculate weighted Avg for other examples

Sl. No.	$\alpha_{CGPA} = 0.347$	$\alpha_{Interact} = 0.5490$	$\alpha_{CommSkill} = -0.5485$	Weighted Avg	Final Prediction
1	Yes	Yes	Yes	0.3475	
2	No	No	No	0.0	
3	Yes	No	No	0.347	
4	No	No	Yes	-0.5485	
5	Yes	Yes	No	0.896	
6	Yes	Yes	No	0.896	

If weighted average is >0 , then predicted value is “yes”
Else “no”

Sl. No.	$\alpha_{CGPA} = 0.347$	$\alpha_{Interact} = 0.5490$	$\alpha_{CommSkill} = -0.5485$	Weighted Avg	Final Prediction
1	Yes	Yes	Yes	0.3475	yes
2	No	No	No	0.0	No
3	Yes	No	No	0.347	yes
4	No	No	Yes	-0.5485	No
5	Yes	Yes	No	0.896	Yes
6	Yes	Yes	No	0.896	Yes

Gradient Boosting

- Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which each **new model** is trained to **minimize the loss function** such as **mean squared error** or **cross-entropy** of the previous model using **gradient descent**.
- In each **iteration**, the algorithm **computes the gradient of the loss function with respect to the predictions of the current ensemble** and **then trains** a new weak model to minimize this gradient.
- The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met.

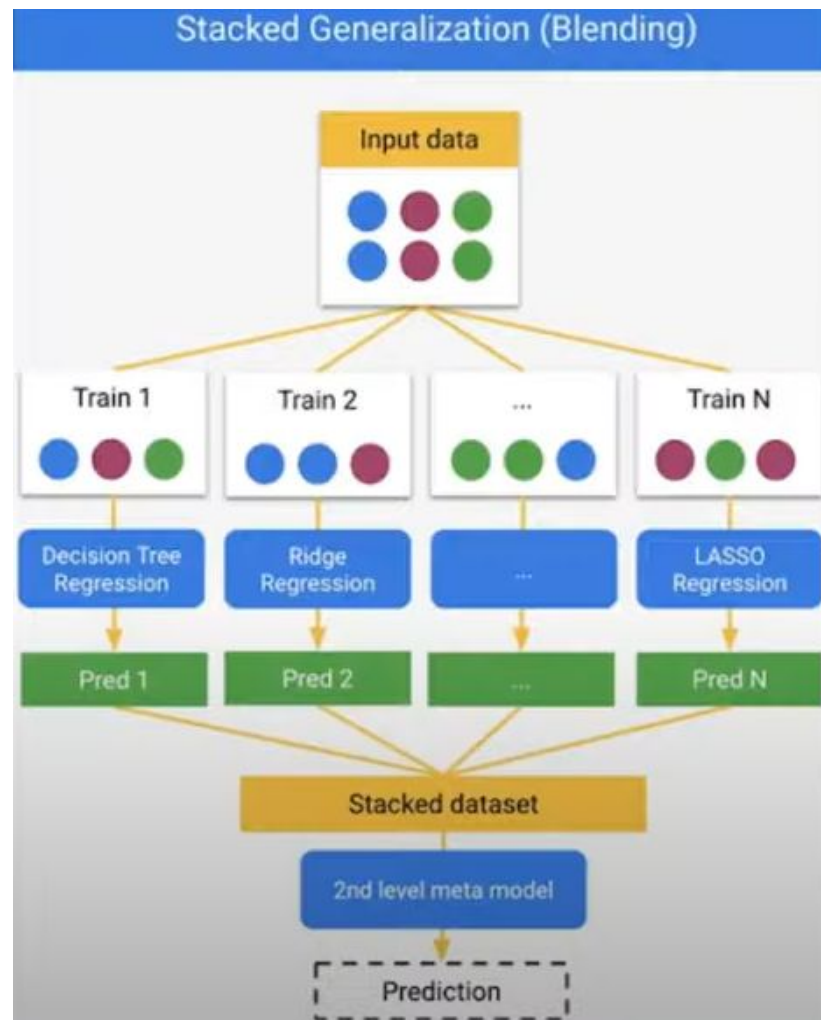
Boosting vs Bagging

Boosting	Bagging
In Boosting we combine predictions that belong to different types	Bagging is a method of combining the same type of prediction
The main aim of boosting is to decrease bias, not variance	The main aim of bagging is to decrease variance not bias
At every successive layer Models are weighted according to their performance.	All the models have the same weightage
New Models are influenced by the accuracy of previous Models	All the models are independent of each other

Stacking

- Stacking, Blending and a Stacked Generalization are all the same thing with different names. It is a kind of ensemble learning
- In traditional ensemble learning, we have multiple classifiers trying to fit to a training set to approximate the target function
- Since, each classifier will have its own output, we will need to find a combining mechanism to combine the results
- This can be through voting (majority wins), weighted voting (some classifier has more authority than the others), averaging the results, etc

- In stacking, the combining mechanism is that the output of the classifiers (Level 0 classifiers) will be used as training data for another classifier (Level 1 classifier) to approximate the same target function.
- Basically, you let the Level 1 classifier to figure out the combining mechanism.



References

<https://www.geeksforgeeks.org/ml-bagging-classifier/>

[Implementing the AdaBoost Algorithm From Scratch - GeeksforGeeks](#)

[GBM in Machine Learning - Javatpoint](#)

<https://www.youtube.com/watch?v=eNyUfpGBLts>

[AdaBoost Ensemble Learning Solved Example Ensemble Learning Solved Numerical Example Mahesh Huddar \(youtube.com\)](#)