LOVELY PROFESSIONAL UNIVERSITY

NAAC GRADE A++

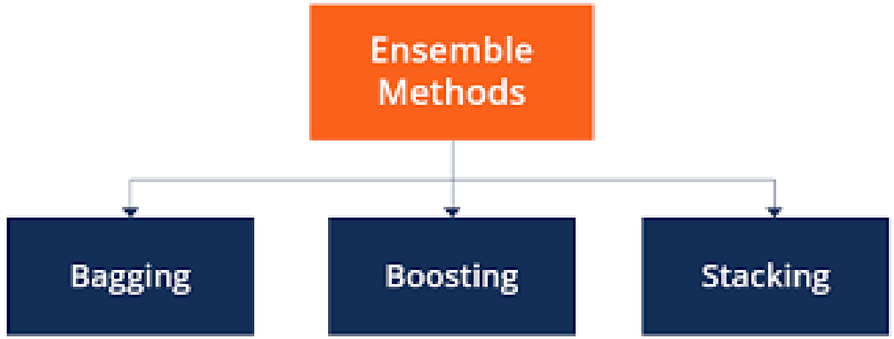
INT395- SUPERVISED ML

Unit 3: Ensemble Methods

Presented By: Blossom Kaler
Assistant Professor
SCSE, LPU

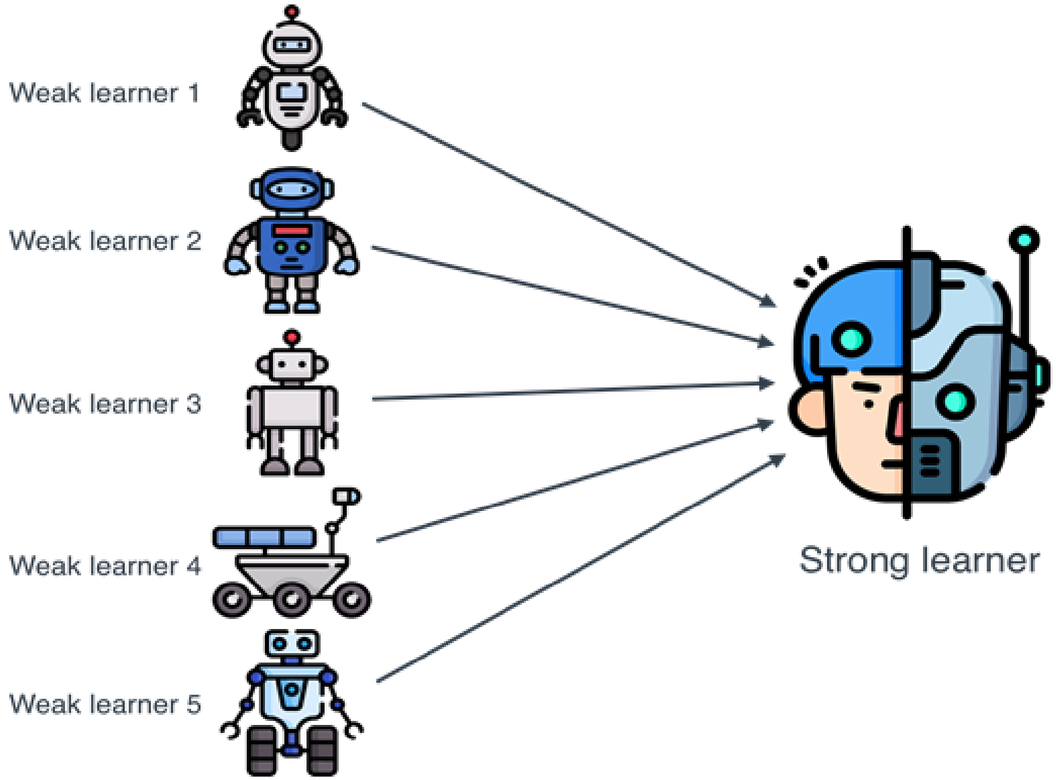
ENSEMBLE MODELS

- Ensemble models combine multiple weak/strong learners to create one powerful model.
- Idea: “Many models together outperform a single model.”
- Reduce variance, bias, and improve generalization.
- Types: Bagging, Boosting, Stacking
- Used widely in competitions (e.g., Random Forest, XGBoost)



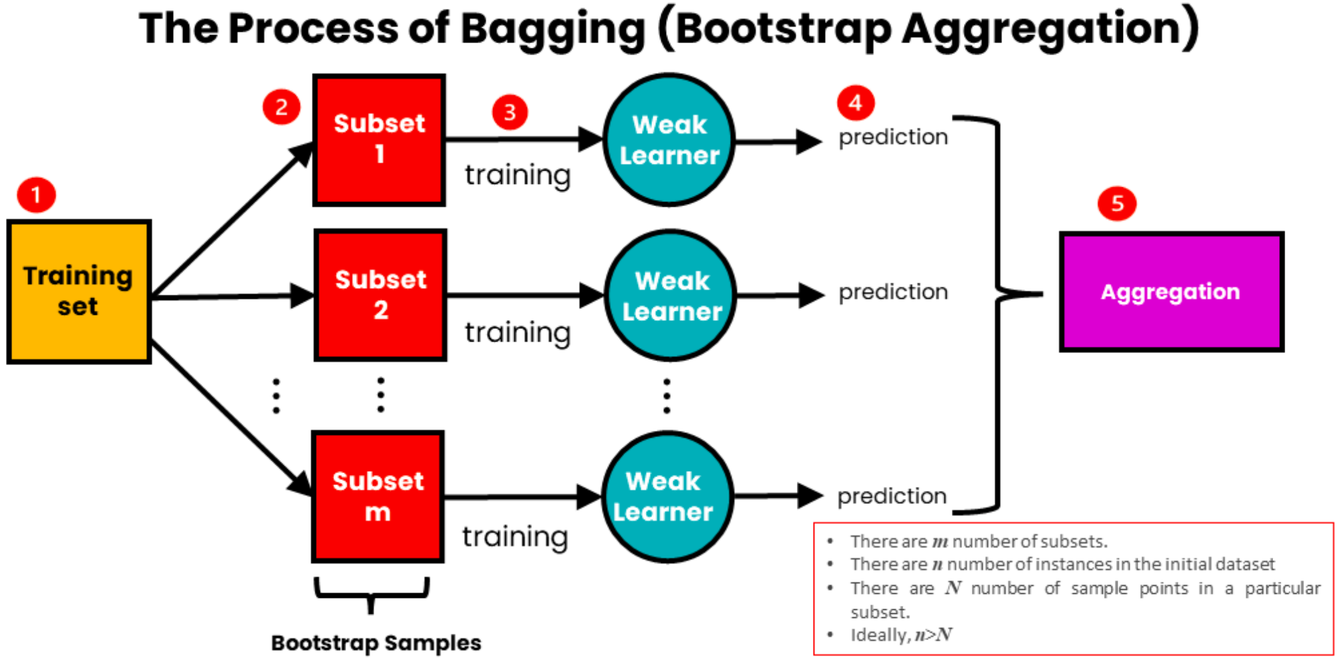
WHY ENSEMBLE MODELS

- Individual models may overfit or underfit.
- Ensembles average out errors → more stable predictions.
- Improve:
- Accuracy
- Robustness
- Resistance to noise
- Work with both classification and regression.



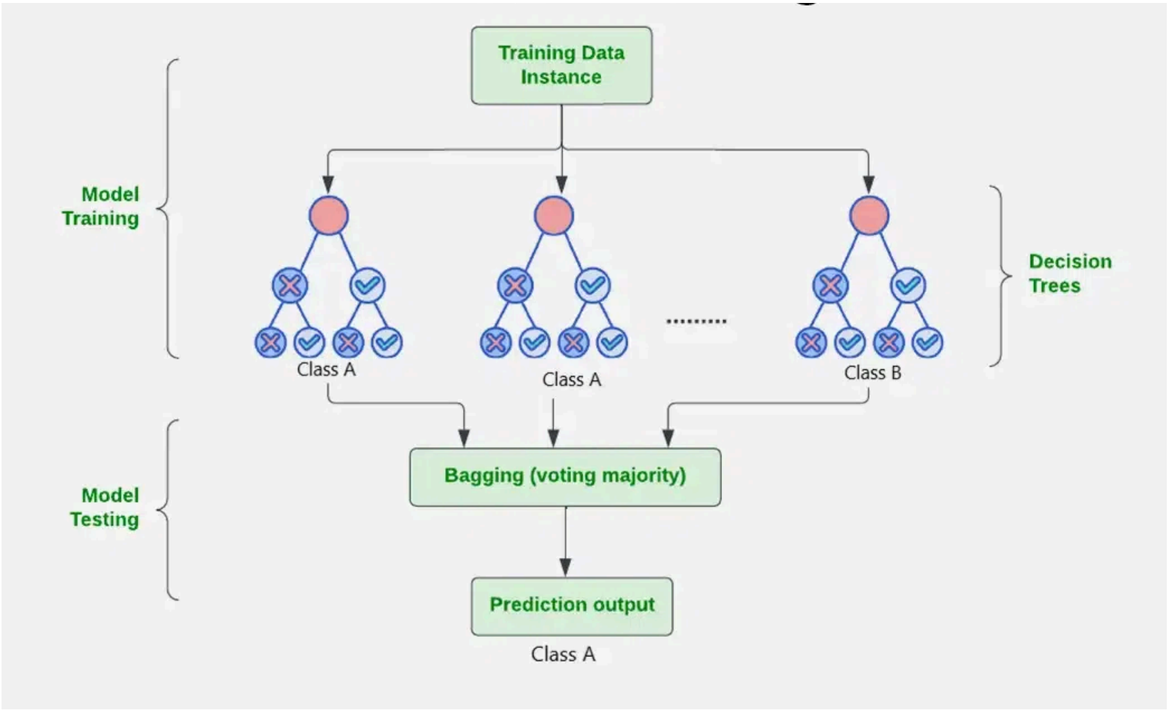
BAGGING

- Bootstrap Aggregating.
- Train many models on different bootstrap samples (random sampling with replacement).
- Each model is independent.
- Final prediction:
- Classification → majority vote
- Regression → average
- Best for reducing variance.



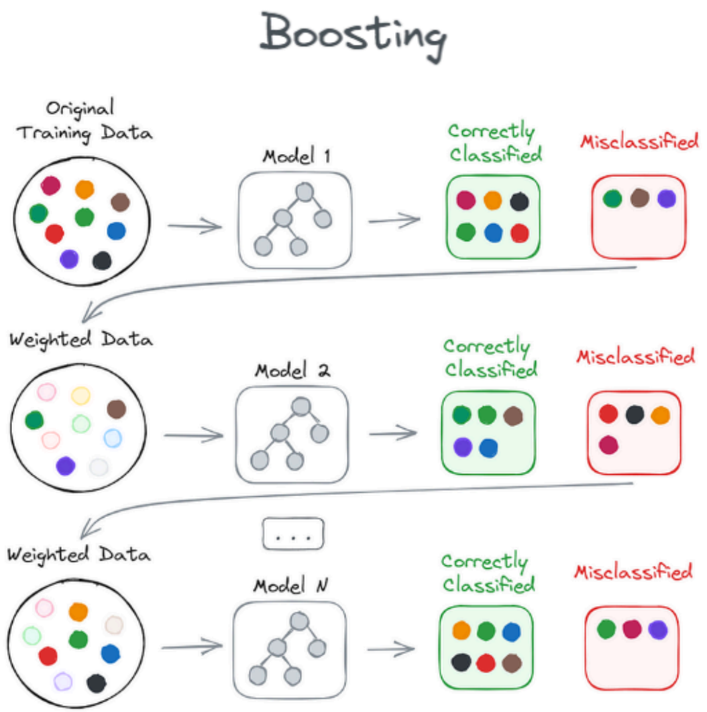
RANDOM FOREST

- Ensemble of multiple decision trees.
- **Extra randomness:**
- Each tree sees a bootstrap sample.
- Each split uses a random subset of features.
- Strong accuracy, less overfitting.



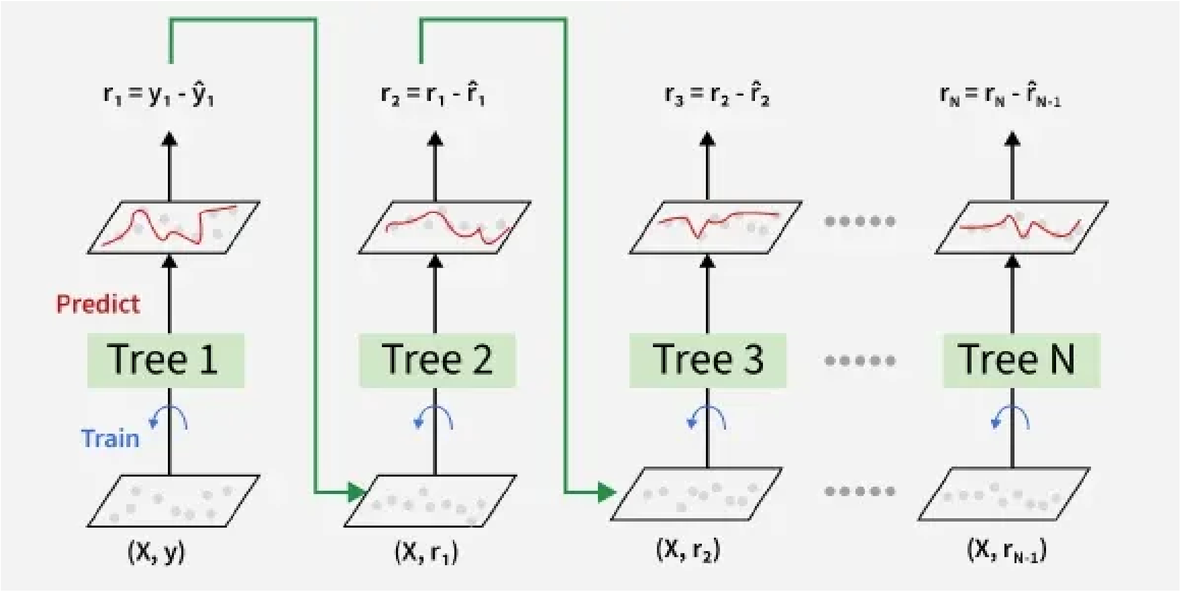
BOOSTING

- Models are trained sequentially, each trying to fix errors of the previous model.
- Turns weak learners into a strong learner.
- Focus on difficult samples.
- Reduces bias.
- **Working:**
- Start with simple model (weak learner).
- Find misclassified / high-error samples.
- Give them higher weight.
- Train next model to correct these mistakes.
- Combine models with weighted voting.



GRADIENT BOOSTING

- Ensemble technique that builds models sequentially
- Each new weak learner (usually a small decision tree) corrects the errors of the previous learners
- Starts with a simple constant prediction (mean of target)
- Computes residuals (actual – predicted)
- New trees are trained to predict these residuals
- Final model = Initial prediction + Sum of all correction trees
- **Shrinkage:** After each tree is trained its predictions are shrunk by multiplying them with the learning rate η which ranges from 0 to 1.



$$y_{pred} = y_1 + \eta \cdot r_1 + \eta \cdot r_2 + \cdots + \eta \cdot r_N$$

Where $r_1, r_2, \ldots, r_N$ are the errors predicted by each tree.

GRADIENT BOOSTING-REGRESSION

Row No.	Cylinder Number	Car Height	Engine Location	Price
1	Four	48.8	Front	12000
2	Six	48.8	Back	16500
3	Five	52.4	Back	15500
4	Four	54.3	Front	14000

Cylinder Number	Car Height	Engine Location	Price	Prediction 1
Four	48.8	Front	12000	14500
Six	48.8	Back	16500	14500
Five	52.4	Back	15500	14500
Four	54.3	Front	14000	14500

Cylinder Number	Car Height	Engine Location	Price	Prediction 1	Residual 1
Four	48.8	Front	12000	14500	-2500
Six	48.8	Back	16500	14500	2000
Five	52.4	Back	15500	14500	1000
Four	54.3	Front	14000	14500	-500

GRADIENT BOOSTING- CLASSIFICATION

$$odds = \frac{p}{q}$$

$$p = probability\ of\ success$$

$$q = 1 - p$$

Logit Function = $\log\left(\frac{p}{1-p}\right)$

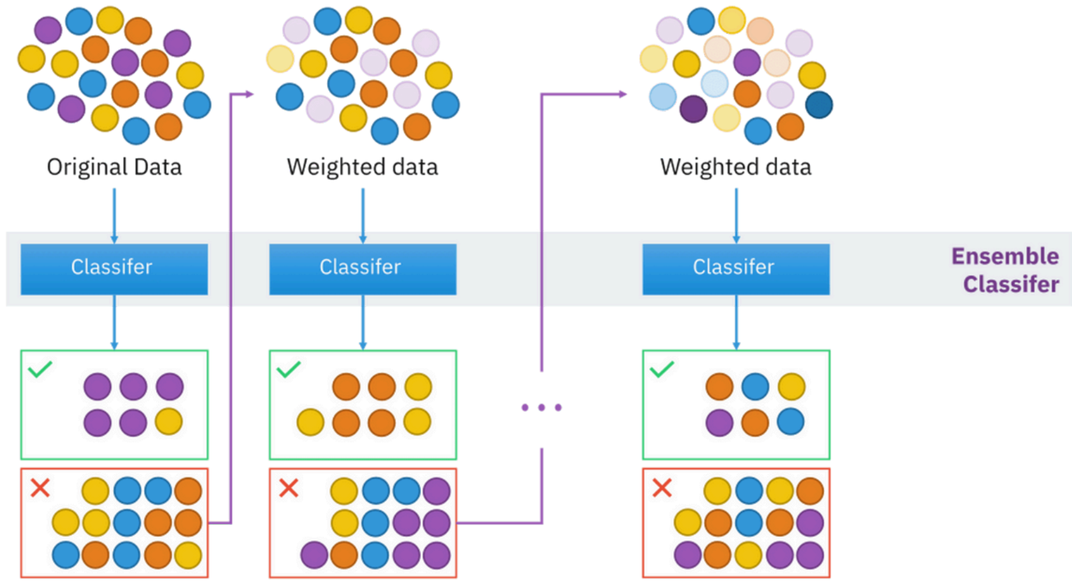
$$\ln\left(\frac{P}{1-P}\right) = a + bX$$

$$\frac{P}{1-P} = e^{a+bX}$$

$$P = \frac{e^{a+bX}}{1+e^{a+bX}}$$

ADABOOST

- AdaBoost (Adaptive Boosting) is an ensemble technique that combines many weak learners to form a strong classifier.
- **Steps:**
- Start with equal weight for all data points
- Train first weak model
- Increase weights of misclassified points
- Decrease weights of correctly classified points
- Train next model using updated weights
- Repeat for T rounds
- Result: Each new learner improves on errors of the previous one.



MATHEMATICAL INTUITION

- Error of model = sum of weights of misclassified points
- Model importance: α
- Weight update idea:
- Wrong prediction → weight increases
- Correct prediction → weight decreases

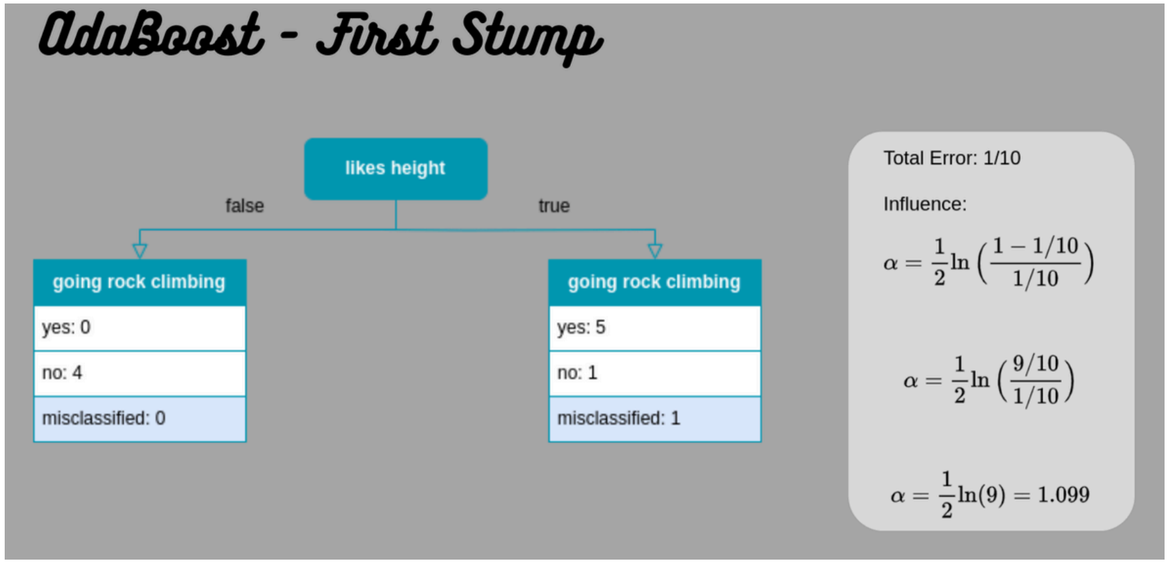
$$\alpha = \frac{1}{2}\ln\left(\frac{1 - TotalError}{TotalError}\right)$$

$$w_{new} = w_{old} \cdot e^{\pm\alpha}.$$

ADABOOST

age	likes height	likes goats	go rock climbing	weight
23	0	0	0	0.1
31	1	1	1	0.1
35	1	0	1	0.1
35	0	0	0	0.1
42	0	0	0	0.1
43	1	1	1	0.1
45	0	1	0	0.1
46	1	1	1	0.1
46	1	0	0	0.1
51	1	1	1	0.1

$$\alpha = \frac{1}{2}\ln\left(\frac{1 - TotalError}{TotalError}\right)$$



ADABOOST

The sign in this equation depends on whether a sample was correctly classified or not. For correctly classified samples, we get

$$w_{new} = 0.1 \cdot e^{-1.099} = 0.0333,$$

and for wrongly classified samples

$$w_{new} = 0.1 \cdot e^{1.099} = 0.3,$$

Build second Model - Data and Weights

age	likes height	likes goats	go rock climbing	weight	normalized weight
23	0	0	0	0.033	0.056
31	1	1	1	0.033	0.056
35	1	0	1	0.033	0.056
35	0	0	0	0.033	0.056
42	0	0	0	0.033	0.056
43	1	1	1	0.033	0.056
45	0	1	0	0.033	0.056
46	1	1	1	0.033	0.056
46	1	0	0	0.3	0.5
51	1	1	1	0.033	0.056

Updated weights, based on the results of the first weak learner. All weights sum up to 1.

ADABOOST

The weights are used to create bins. Let's assume we have the weights $w_1, w_2, \ldots, w_N$, the bin for the first sample is $[0, w_1]$, for the second sample, $[w_1, w_1 + w_2]$, etc. In our example, the bin of the first sample is $[0, 0.056]$, for the second $[0.056, 0.112]$, etc.. The following plot shows all samples with their bins.

age	likes height	likes goats	go rock climbing	bins
23	0	0	0	[0,0.056]
31	1	1	1	[0.056, 0.111]
35	1	0	1	[0.111, 0.178]
35	0	0	0	[0.178, 0.222]
42	0	0	0	[0.222, 0.278]
43	1	1	1	[0.278, 0.333]
45	0	1	0	[0.333, 0.389]
46	1	1	1	[0.389, 0.444]
46	1	0	0	[0.444, 0.944]
51	1	1	1	[0.944, 1]

ADABOOST

- Weighted Resampling

Modified Dataset

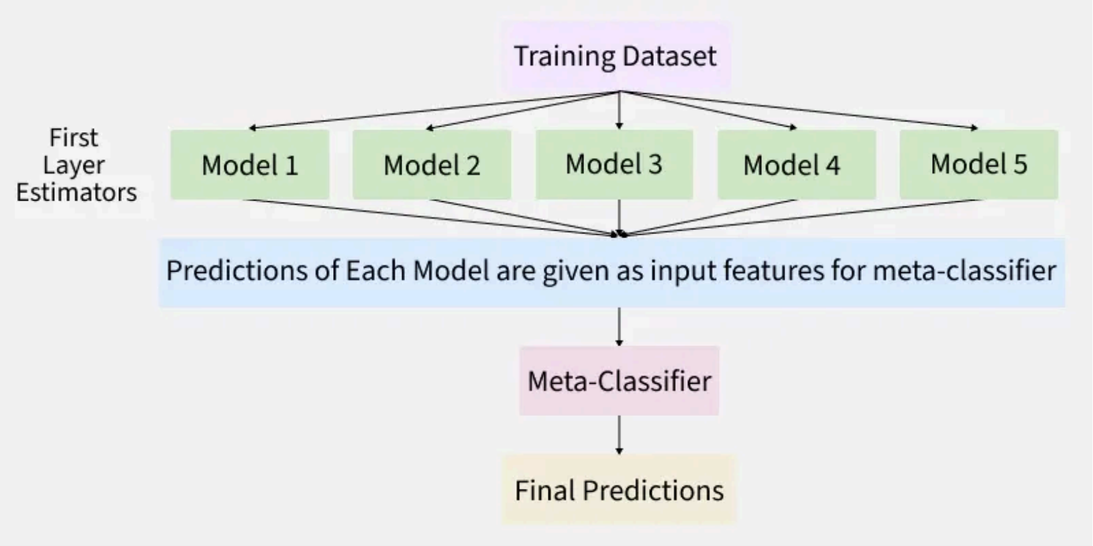
age	likes height	likes goats	go rock climbing
46	1	0	0
46	1	0	0
35	1	0	1
46	1	0	0
51	1	1	1
46	1	0	0
46	1	0	0
35	1	0	0
46	1	0	0
35	1	0	1

Modified dataset based on bootstrapping the previous dataset using the assigned bins.

Note, that this dataset contains duplicates entries.

STACKING

- Stacking is an ensemble learning technique that combines multiple different models to improve prediction performance.
- Instead of simple averaging or voting, stacking learns how to combine models using another model.
- It uses:
- Base models (Level-0 learners)
- Meta-model (Level-1 learner)
- Goal: Use strengths of different models to reduce overall error.
-



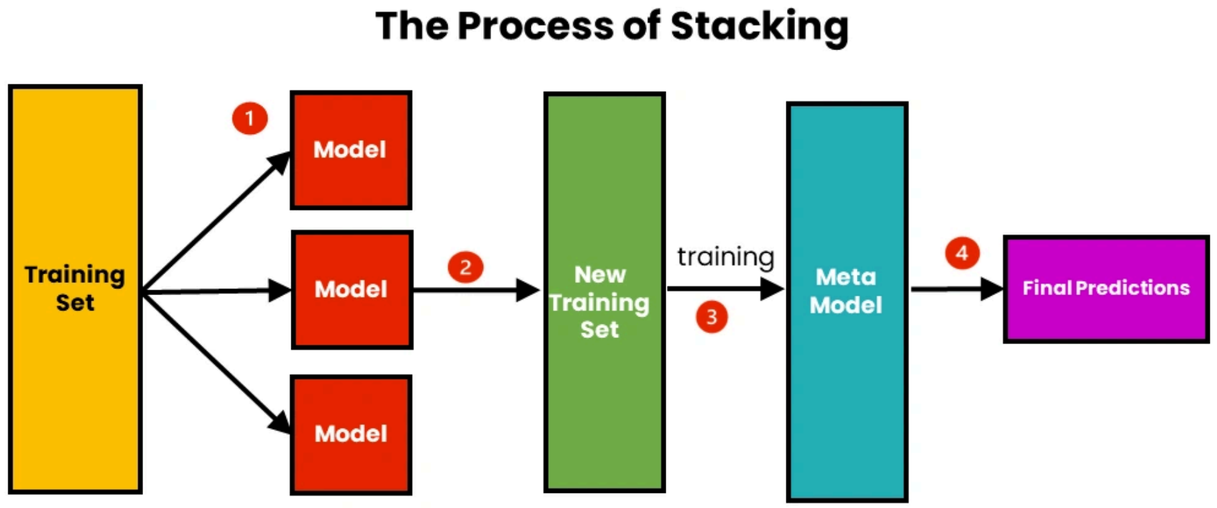
PROCESS OF STACKING

- The process can be summarized in the following steps:
- **Start with training data:** We begin with the usual training data that contains both input features and the target output.
- **Train base models:** The base models are trained on this training data. Each model tries to make predictions based on what it learns.
- **Generate predictions**: After training the base models make predictions on new data called **validation data or out-of-fold data**. These predictions are collected.
- **Train meta-model**: The meta-model is trained using the predictions from the base models as new features. The target output stays the same and the meta-model learns how to combine the base model predictions.
- **Final prediction**: When testing the base models make predictions on new, unseen data. These predictions are passed to the meta-model which then gives the final prediction.
-

WHY STACKING WORKS

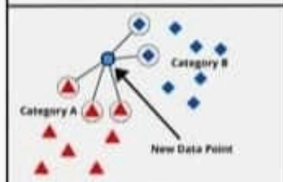
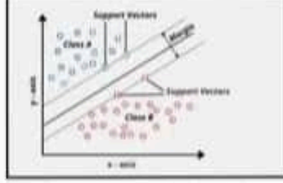
- Different models capture different patterns.
- Errors made by one model may be corrected by another.
- Meta-model learns optimal combination strategy.
- Reduces bias and variance better than single models.
- **Training Process:**

- Split training data into K folds.
- Train base models on K-1 folds.
- Predict on the remaining fold (out-of-fold predictions).
- Collect predictions from all base models.
- Train meta-model on these predictions.



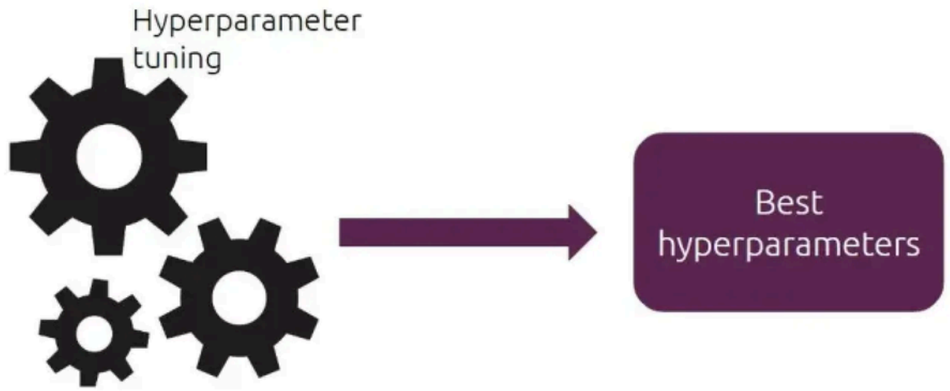
HYPERPARAMETERS

- Hyperparameters are external settings chosen before training a model.
- They control how the model learns (not learned from data).
- Examples:
- Learning rate, batch size (Neural Networks)
- Number of trees, max depth (Random Forest)
- C, kernel, gamma (SVM)
- Correct hyperparameters = faster learning + better accuracy
- Need systematic tuning to find the optimal values

	Decision Tree	Max_depth, min_samples_split, min_samples_leaf, criterion
	K-Nearest Neighbors	n_neighbors, weights, metric
	Support Vector Machines	C, kernel, gamma, degree (for polynomial kernel)

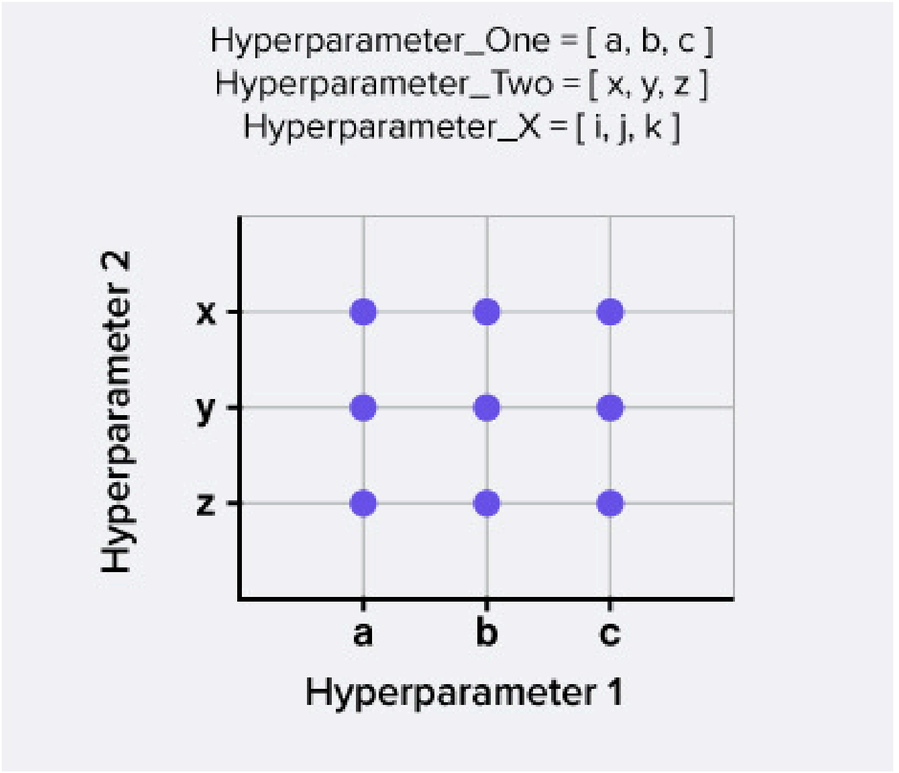
HYPERPARAMETER TUNING?

- Process of systematically searching for the best hyperparameter values for a model.
- Models have settings (hyperparameters) that control learning but are not learned from data.
- Default values rarely give best performance.
- Tuning helps:
- Reduce underfitting/overfitting
- Improve accuracy and generalization
- Optimize training time and model stability
- Goal: find the best combination of hyperparameters for the task.



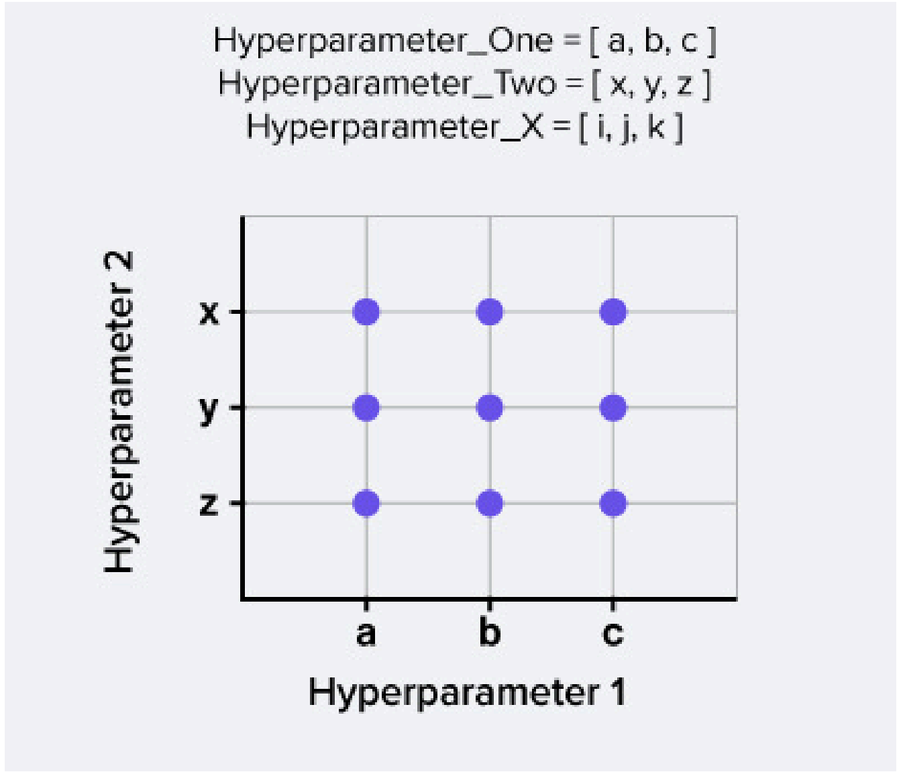
GRID SEARCH

- Exhaustively tries all combinations of hyperparameters.
- Requires defining a parameter grid (list of possible values).
- Uses cross-validation to evaluate each combination.
- Guarantees finding best result within the tested grid.



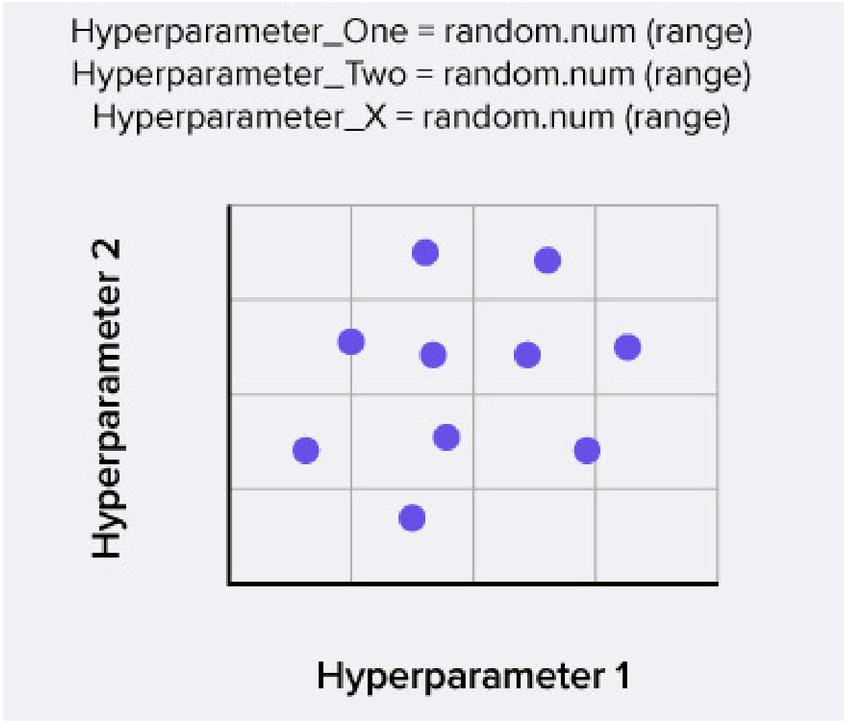
GRID SEARCH – PROS & CONS

- **Pros:**
- Systematic and exhaustive
- Simple to understand and implement
- Reliable for small parameter spaces
- **Cons:**
- Computationally expensive
- Scales poorly with many parameters
- May waste time on unimportant combinations



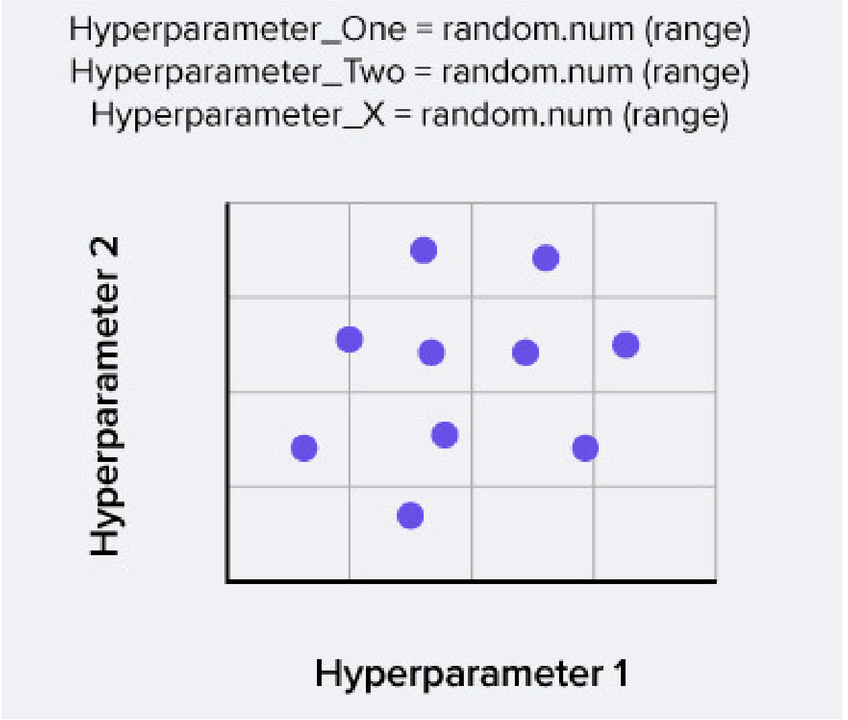
RANDOM SEARCH

- Randomly samples combinations from the hyperparameter space.
- Does not try all possibilities.
- Still uses cross-validation for evaluation.
- Covers space more efficiently when some parameters are less sensitive.



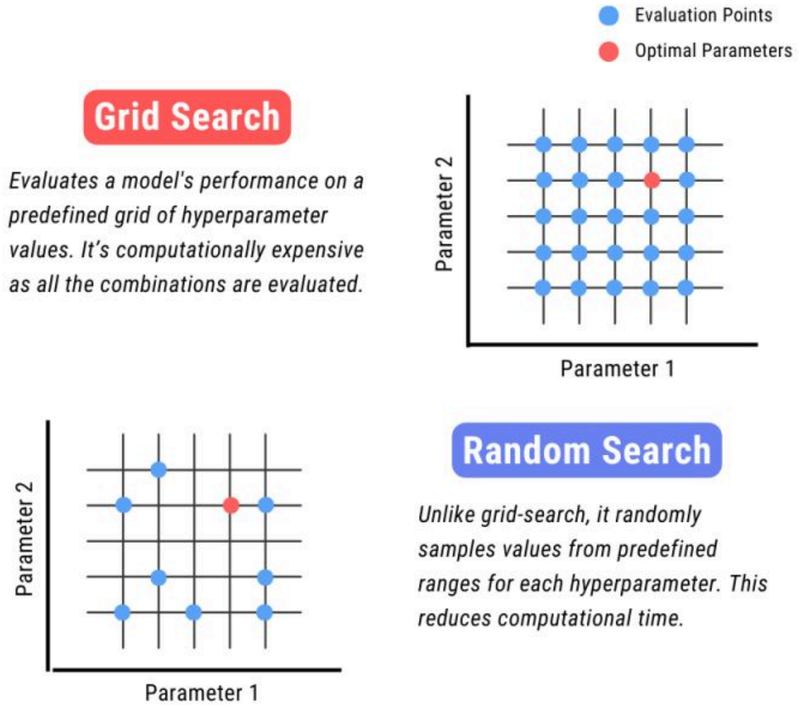
RANDOM SEARCH – PROS & CONS

- **Pros:**
- Much faster for large parameter spaces
- Often finds near-optimal results
- More efficient sampling than grid search
- Suitable for high-dimensional tuning
- **Cons:**
- Does not guarantee best combination
- Results may vary slightly each run



GRID SEARCH VS RANDOM SEARCH

- Grid Search → exhaustive but slow
- Random Search → faster, good coverage
- Random Search preferable when:
- Parameter space is large
- Only a few parameters are highly influential
- Grid Search preferable when:
- Small, well-defined search space



LOVELY PROFESSIONAL UNIVERSITY

NAAC GRADE A++

THANK YOU