Unit – 5

Unit 5 - Syllabus
Ensemble Methods

- ✓ Introduction to Ensemble Methods
- ✓ Bagging
- ✓ Committee Machines and Stacking
- ✓ Boosting :
  - Gradient Boosting,
  - Ada (Adaptive) Boosting
- ✓ Random Forest:
  - Multi Class Classification

- Ensemble Learning is a supervised learning technique.
- Used to improve overall performance by combining the predictions from multiple models (Base Models)
- Each individual model in the ensemble is trained on the same dataset, but they may use different algorithms or *variations of the data.*
- combining multiple individual models (often called “Base Models" or “Base Learners") to create a stronger, more accurate predictive model.

- The idea behind ensemble methods is that by *aggregating the predictions of multiple models, the collective wisdom of the ensemble can often outperform any single model in terms of Predictive Accuracy and Robustness.*

- The main hypothesis is that when base models are correctly combined we can obtain more Accurate and/or Robust Models.

- They can be particularly useful when dealing with complex and noisy datasets, as well as when trying to improve the generalization performance of a model.

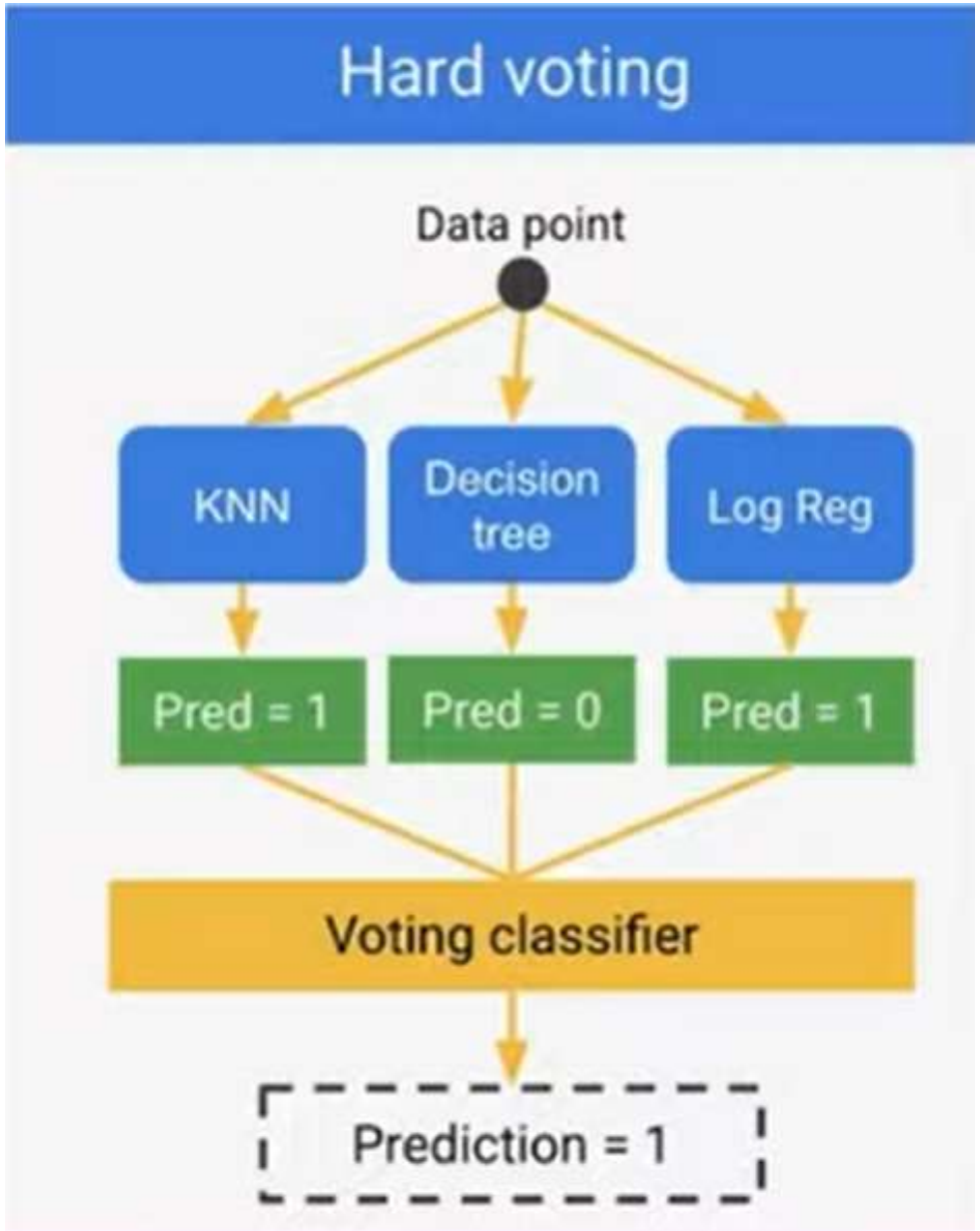
- Base Models:
- These are the individual models that make up the ensemble.
- *Common base models include :*
    - Decision Trees,
    - Support Vector Machines,
    - Neural Networks, and more.
- The choice of Base Models depends on the problem and dataset.

- Aggregation: combine the predictions of base models in different ways, as follows:

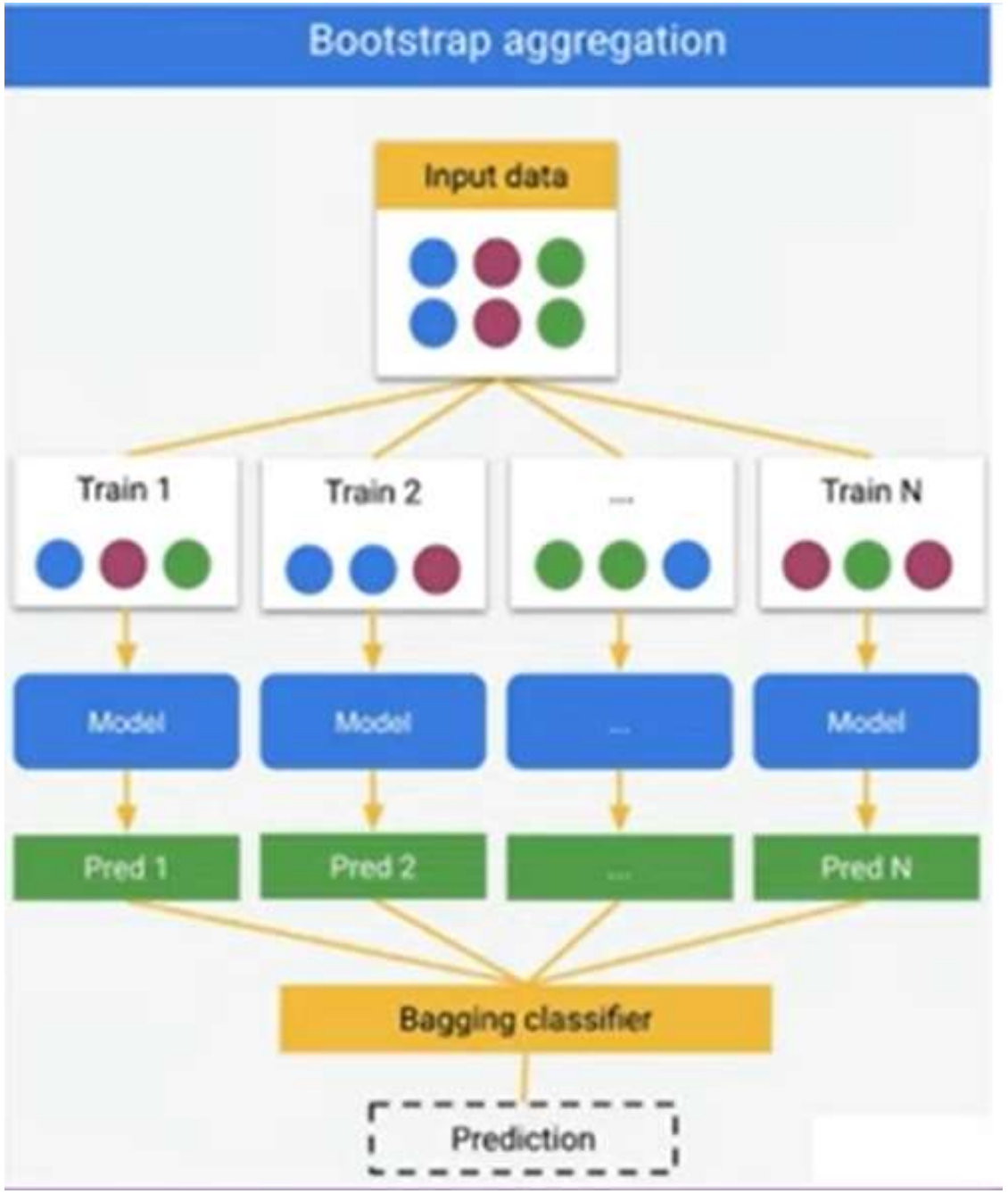
1. Voting (Averaging)
2. Bootstrap Aggregation (OR) Bagging
   - Random Forest
3. Boosting (sub types: Ada Boost, Gradient Boosting, XG Boosting)
4. Stacked Generalization (OR) Blending

✓Voting (Averaging):

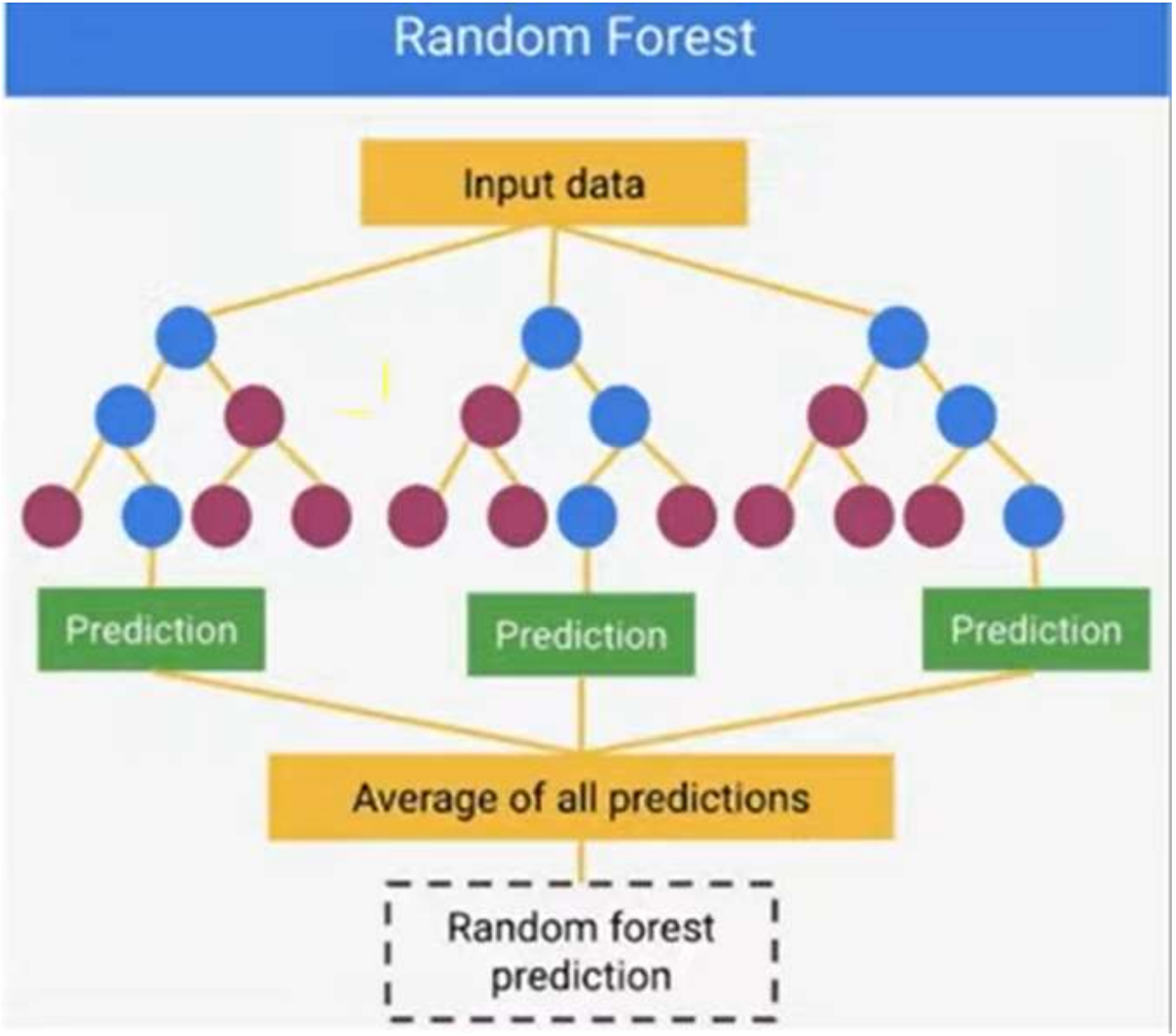
- In classification tasks, the ensemble may use a majority vote (hard voting) or a weighted average of class probabilities (soft voting) to make predictions.
- In regression tasks, ensemble methods often compute the average or weighted average of the predictions from base models.



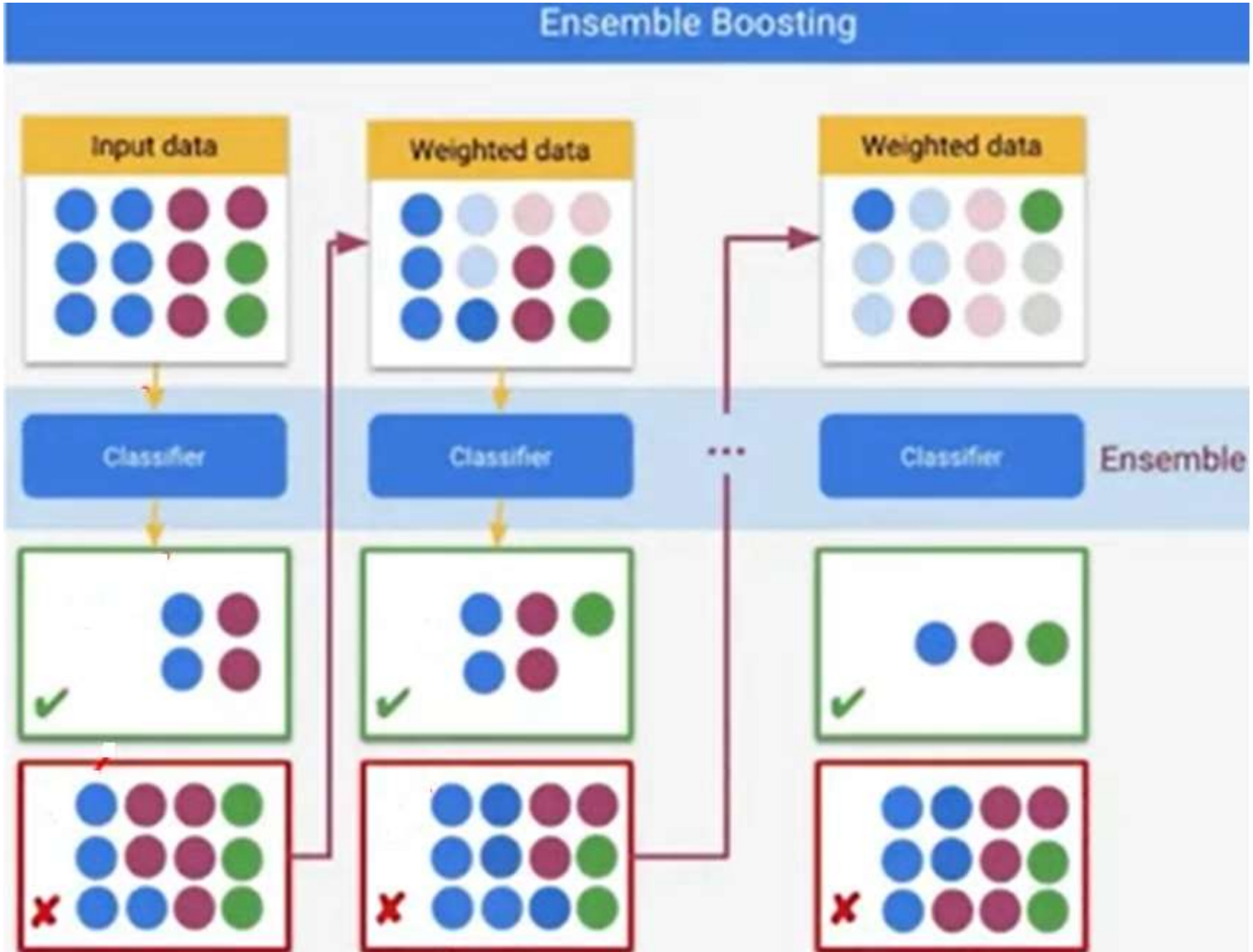
- Bootstrap aggregation (Bagging)
- *Learns independently from each other in parallel and combines them by following some kind of deterministic averaging process*
- It decreases the variance and helps to avoid overfitting.
- The *input data is divided into N no.of sub sets* and trained on different algo.
- It is usually applied to decision tree methods.
- Bagging is a special case of the model averaging approach.



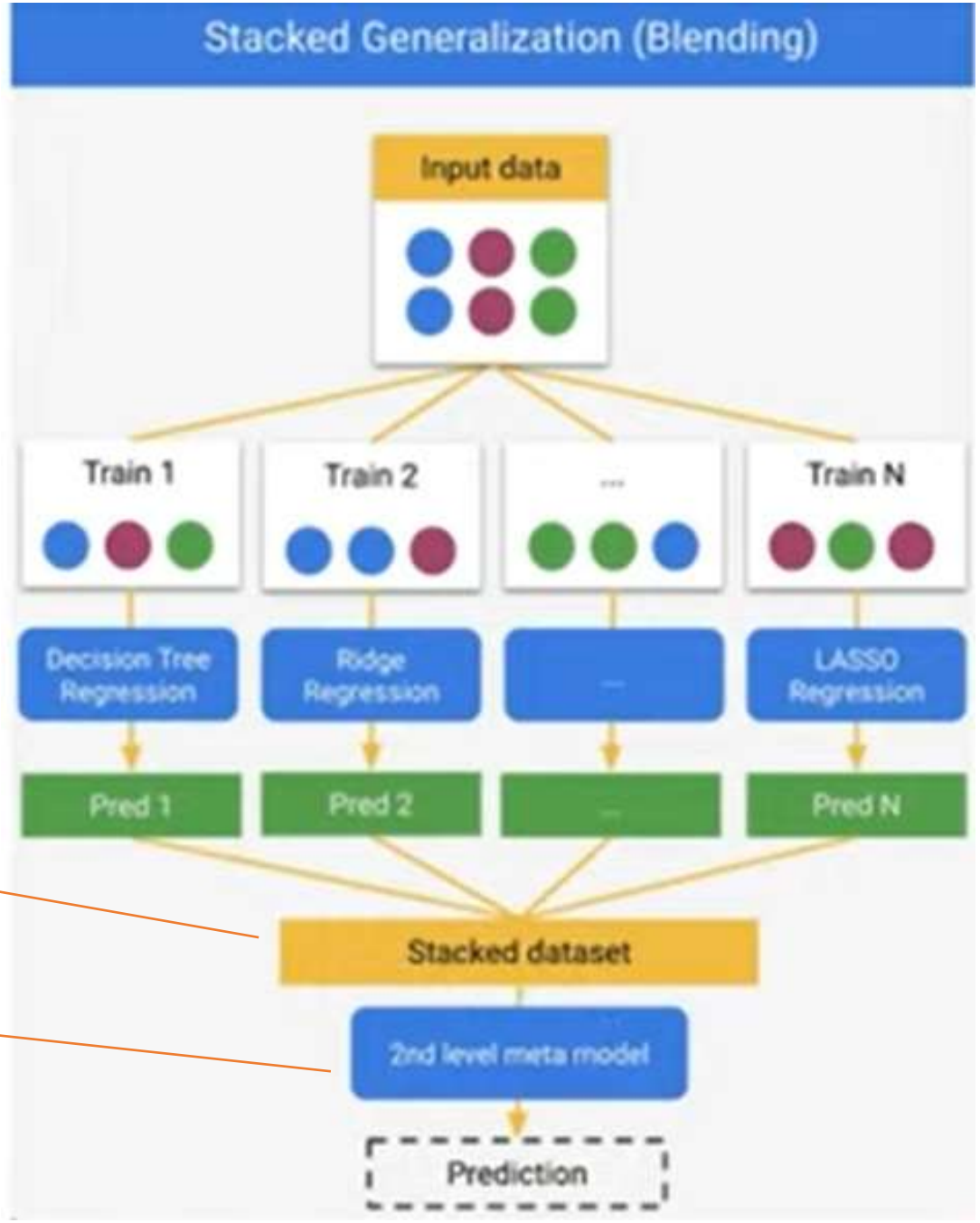
- Random Forest
- Here, multiple Decision Trees are constructed and
- Then they are merged to get a more accurate prediction.
- Divide the input data into multiple groups.



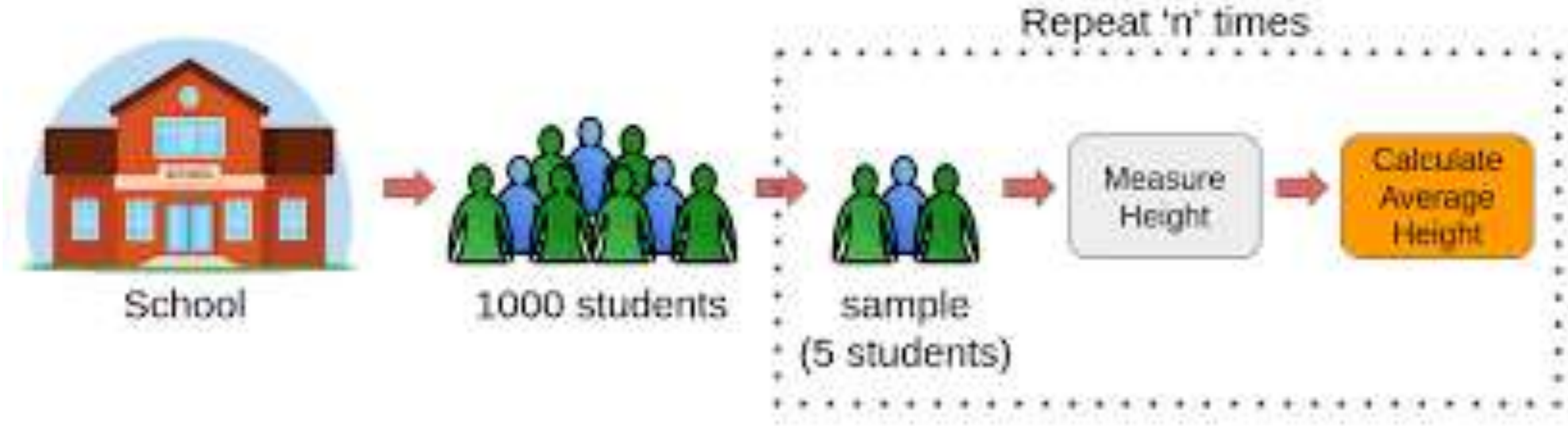
- Boosting:
- *Learns them sequentially in a very adaptative way (a base model depends on the previous ones) and combines them.*
- It is done by building a model by using weak models in a series.
- Firstly, a model is built from the training data
- Then the second model is built which tries to correct 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 is added.


Ensemble Boosting
Input data
Weighted data
Weighted data
Classifier
Classifier
...
Classifier
Ensemble

- Stacked Generalization (Blending):
- *learns them in parallel and combines them by training a meta-model to output a prediction based on the different weak models predictions*
- The output (predictions) of level 1 models will be grouped as a stacked dataset.
- Then 2nd level model will be constructed (input is level 1 models o/p) to get the final prediction.



Bagging
(or)
Bootstraping with Aggregation

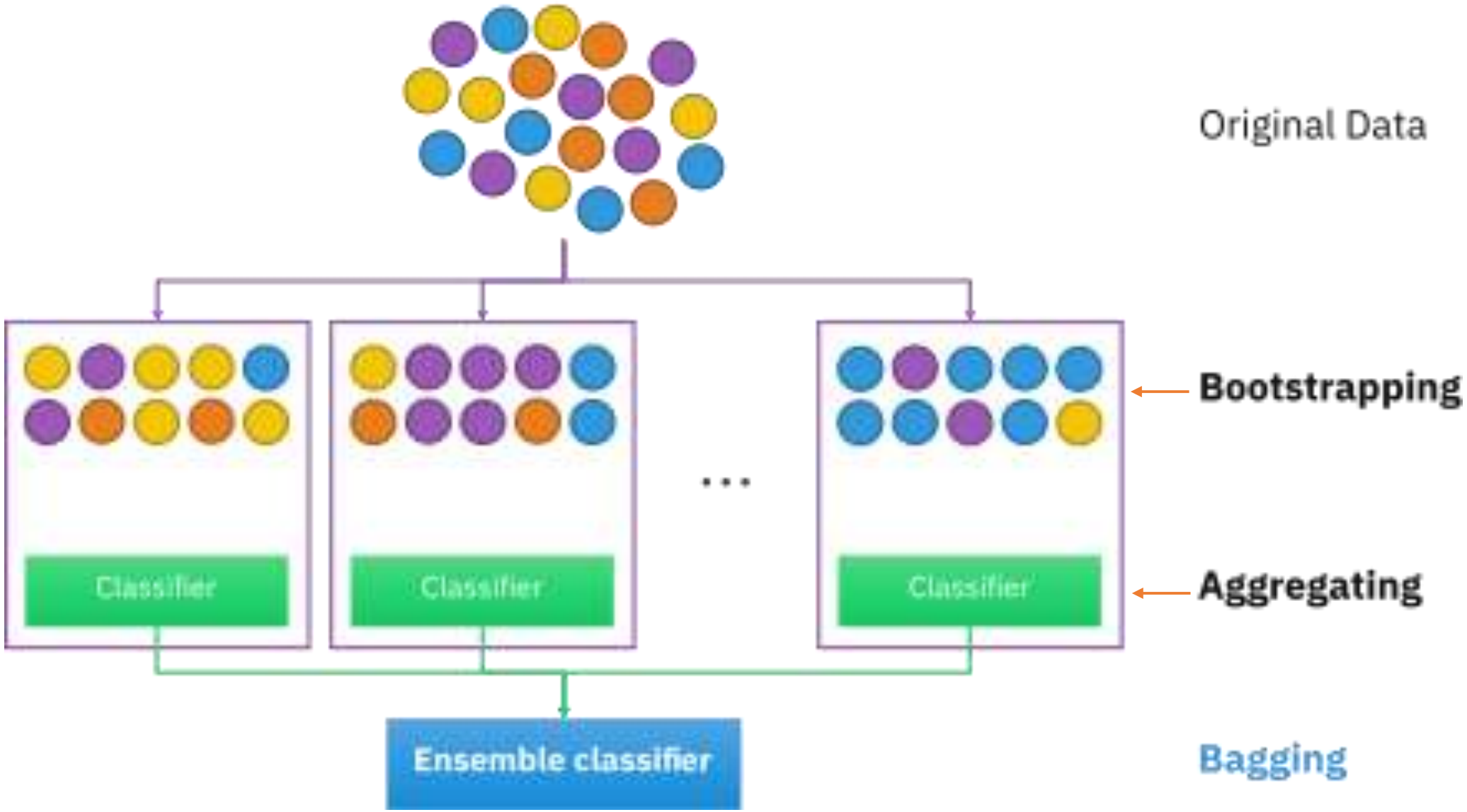


- Bagging also referred as Bootstraping.

- This statistical technique consists in generating samples of size B (called bootstrap samples) from an initial dataset of size N by randomly drawing with replacement B observations.

Row sampling with replacement

Bootstraping + Aggregation = Bagging.

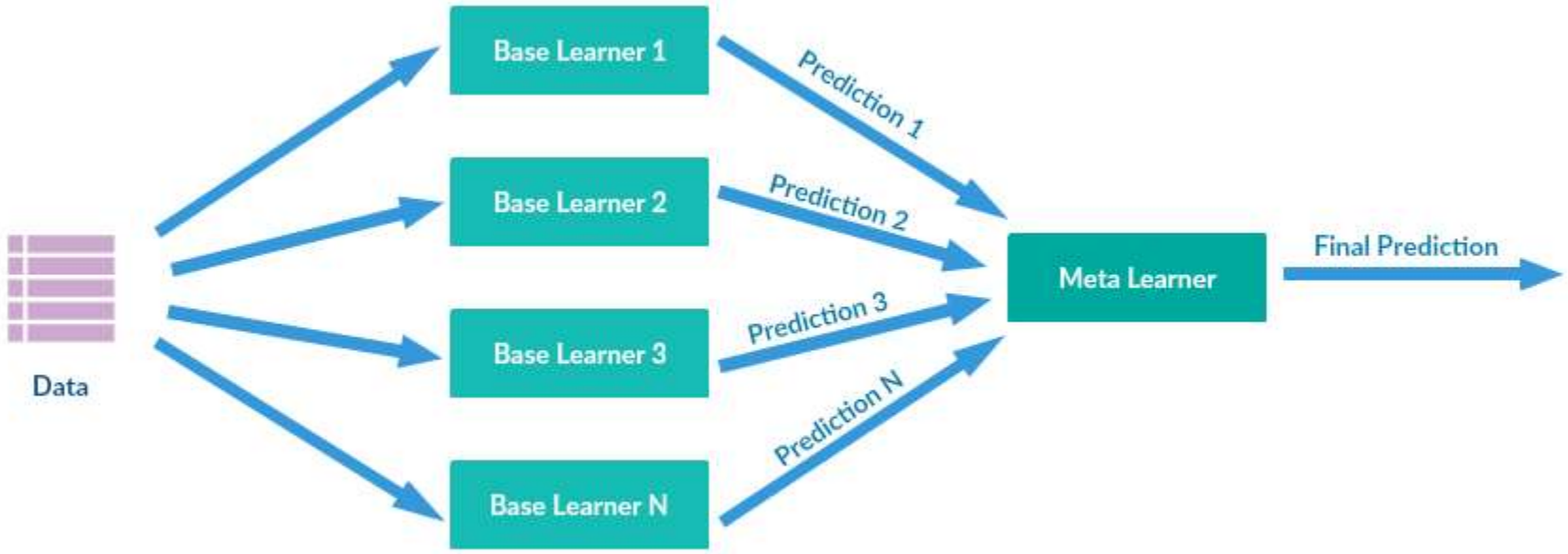


- The idea of bagging is then simple: we want to fit several independent models and “average” their predictions in order to obtain a model with a lower variance.
- First, *we create multiple bootstrap samples so that each new bootstrap sample will act as another (almost) independent dataset drawn from true distribution.*
- Then, we can fit a weak learner for each of these samples and finally aggregate them such that we kind of “average” their outputs and, so, obtain an ensemble model with less variance that its components.

- The bootstrap samples are approximatively independent and identically distributed (i.i.d.), so are the learned base models.
- Then, “averaging” weak learners outputs do not change the expected answer but reduce its variance (just like averaging i.i.d. random variables preserve expected value but reduce variance).
- There are several possible ways to aggregate the multiple models fitted in parallel. For a regression problem, the outputs of individual models can literally be averaged to obtain the output of the ensemble model.

- For classification problem the class outputted by each model can be seen as a vote and the class that receives the majority of the votes is returned by the ensemble model (this is called **hard-voting**).
- Still for a classification problem, we can also consider the probabilities of each classes returned by all the models, average these probabilities and keep the class with the highest average probability (this is called **soft-voting**).
- Averages or votes can either be simple or weighted if any relevant weights can be used.

Committee Machines and Stacking

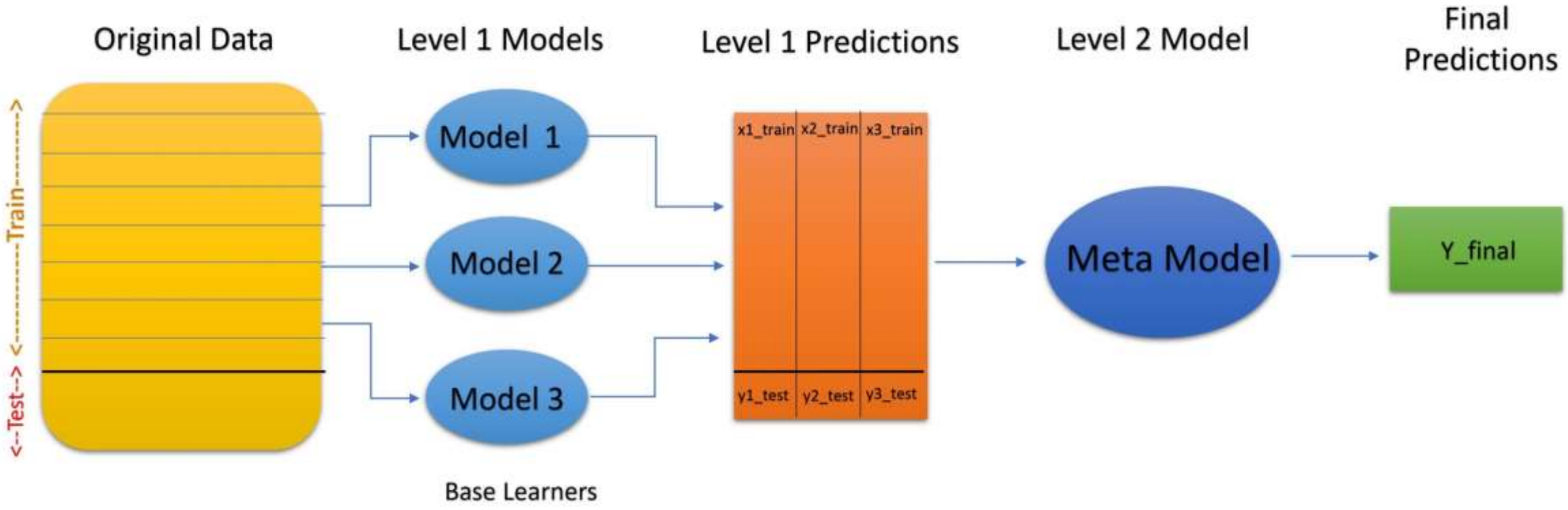


- **Committee Machine (OR) Committee Machines**
- A committee machine is a type of Artificial Intelligence Algorithm that combines the predictions of multiple models to produce a more accurate result.
- Committee Machines and Stacking are both ensemble learning techniques used in ML to improve the performance of predictive models by combining the predictions of multiple base models.
- Multiple models are created using the SAME DATASET.
- Each model provides its own prediction, and the final prediction is typically obtained through some form of Aggregation, such as majority voting or weighted averaging.
- The key idea behind committee machines is that by combining multiple models, you can reduce the impact of individual model errors and increase overall prediction accuracy.

Stacking (OR) Stacked Generalization:

- It is another ensemble learning technique
- It goes a step further than committee machines.
- In stacking, multiple base models (often called level-0 models) are trained on the same dataset to make predictions.
- However, instead of simply aggregating their predictions, a meta-model (level-1 model) is trained on top of these base models to learn how to combine their outputs effectively.
- Stacking allows the ensemble to learn how to best combine the outputs of the base models, potentially capturing more complex relationships in the data and improving overall predictive performance.

STACKING


Original Data
Level 1 Models
Level 1 Predictions
Level 2 Model
Final Predictions
<--Test--> <----------Train---------->
Model 1
Model 2
Model 3
x1_train
x2_train
x3_train
y1_test
y2_test
y3_test
Meta Model
Y_final
Base Learners

Step-by-step process of how stacking works:

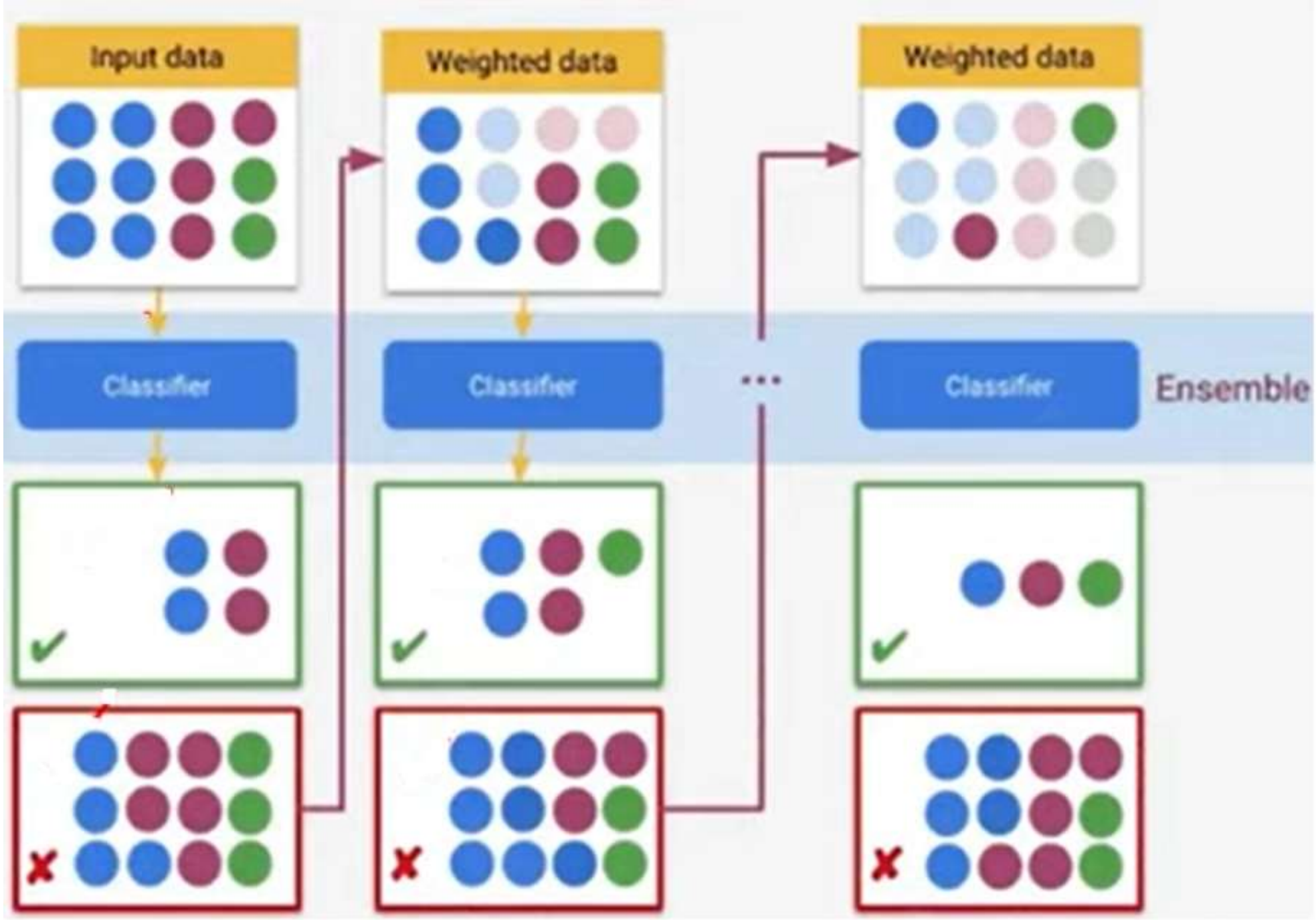
1. Multiple base models are trained on the same dataset, and their predictions are collected.
2. A separate dataset is used to train the meta-model.
   - This dataset contains the same input features as the original data but uses the predictions from the base models as input features.
3. The meta-model is trained to make predictions based on the predictions of the base models.
4. When making predictions on new, unseen data, the base models generate their predictions, which are then used as input features for the trained meta-model, and the meta-model provides the final prediction.

- In Summary ::
- Committee Machines :: focus on aggregating predictions from multiple models,
- Stacking :: takes the step further by training a meta-model to learn how to best combine those predictions (of base models).

Boosting

- ✓ **Gradient Boosting.**
- ✓ **Adaptive (OR) Ada Boosting.**

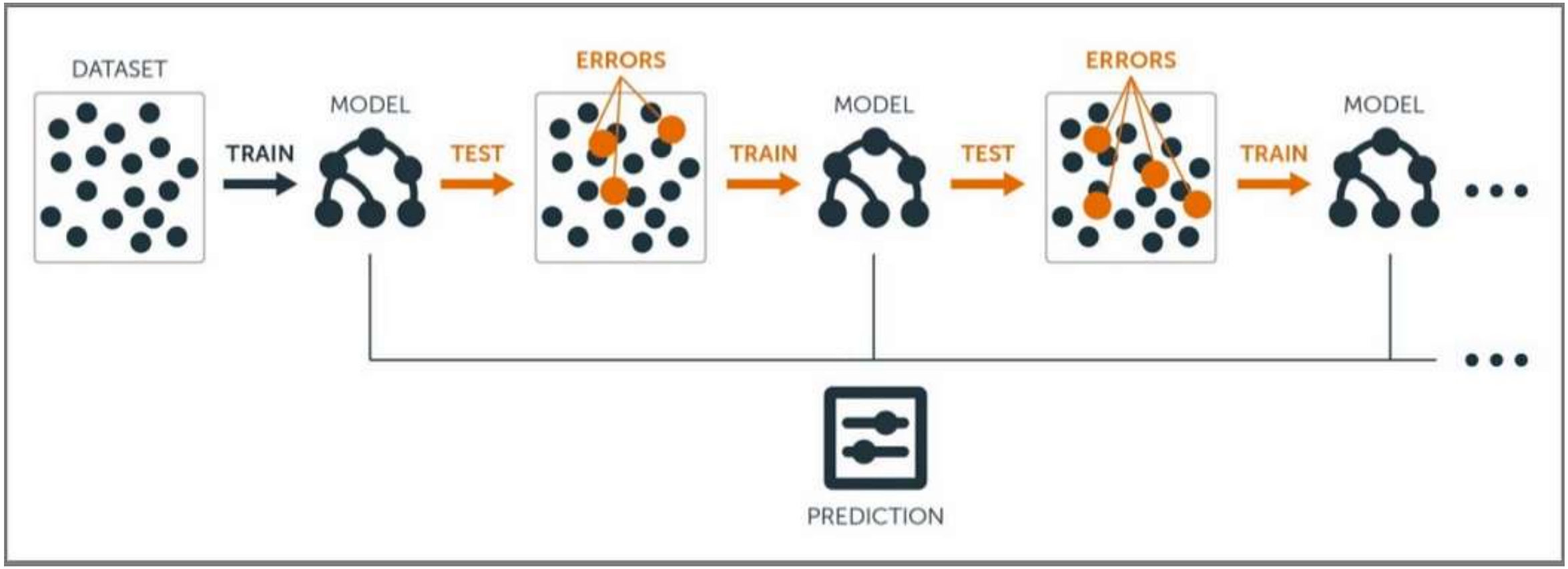
- Boosting:
- *Learns sequentially in a very adaptative way (one base model depends on the previous base model) and combines them.*
- It is done by building a model by using weak models in a series.
- Firstly, a model is built from the training data
- Then the second model is built which *tries to correct 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 is added.


Input data
Weighted data
Weighted data
Classifier
Classifier
...
Classifier
Ensemble

Gradient Boosting – Ensemble Technique

- Gradient Boosting is an Supervised machine learning algorithm.
- It is used for Classification and Regression problems.
- It uses multiple weak learners/models ***to produce a strong model*** for regression and classification.
- Gradient Boosting relies on the intuition that the best possible next model , when combined with the previous models, minimizes the overall prediction errors.
- **The key idea is** to set the target outcomes from the previous models to the next model in order to minimize the errors.

Gradient Boosting



Requirements for Gradient Boosting

1. A Loss Function to Optimize.
2. A Weak Learner (Base Model) to make prediction(Generally Decision tree).
3. An Additive Model to add weak learners to minimize the Loss function.

1. Loss Function : for finding Errors In The Predictions

- The loss function basically tells how my algorithm, Models the dataset.
- In simple terms it is *difference between actual values and predicted values.*
- **Regression Loss functions:**
  - ✓ L1 loss (or) Mean Absolute Errors (MAE)
  - ✓ L2 Loss (or) Mean Square Error(MSE)
  - ✓ Quadratic Loss
- **Binary Classification Loss Functions:**
  - ✓ Binary Cross Entropy Loss
  - ✓ Hinge Loss
  - ✓ A gradient descent procedure is used to minimize the loss when adding trees.

2. Weak Learners (or) Base Models:

- The models which is used sequentially to reduce the error generated from the previous models and to return a strong model on the end.
- Usually Decision Trees are used as base models or weak learners in gradient boosting algorithm.

3. Additive Model

- In gradient boosting, decision trees are added one at a time (in sequence), and existing trees (in the model) are not changed.

- Gradient Boosting builds a strong predictive model by combining the predictions of multiple weak learners (typically decision trees) in an additive manner, where each new learner corrects the errors made by the previous ones.
- Here are the general steps for implementing the Gradient Boosting algorithm:

Algorithm Steps : Gradient Boosting

- Step 1: Initialize the model
- Start with an initial simple model as your base estimator/model.
- This could be a simple decision tree.
- Step 2: Calculate the initial predictions:
- Make predictions using the initial model on the training data.
- These predictions will be used as a starting point for further refinement.
- Step 3: Calculate the initial Residuals/Errors:
- Calculate the difference between the actual target values and the predictions made by the initial model.
- These residuals represent the errors made by the initial model.

- Step 4: Train a weak learner on the residuals:
- Train a new weak learner (typically a decision tree) on the residuals from step 3.
- The goal is to fit the new learner to the errors made by the previous model.
- Step 5: Update the model:
- Add the predictions made by the new weak learner to the predictions of the previous model.
- This update process is typically done with a learning rate (shrinkage parameter) that controls the contribution of the new learner to the ensemble.
- Step 6: Repeat steps 3 to 5:
- Repeat the process of calculating residuals, training a new weak learner, and updating the model for a specified number of iterations or until a convergence criterion is met.

- Step 7: Finalize the model:
- The final ensemble model is the sum of all the individual weak learners' predictions, each weighted by the learning rate.
- Step 8: Predict with the ensemble model:
- To make predictions on new data, apply the final ensemble model, which is the combination of all the weak learners.
- Step 9: Evaluate the model:
- Use appropriate evaluation metrics to assess the performance of your Gradient Boosting model, such as:
  - ❖Mean Squared Error for Regression,
  - ❖Accuracy for classification

- Note:
- Tune hyperparameters:
- Gradient Boosting has various hyperparameters that can be tuned to improve performance, such as the learning rate
- depth of the weak learners (e.g., decision trees), and
- the number of iterations.

- Example:

This is our data set :

Age	Sft.	Location	Price
5	1500	5	480
11	2030	12	1090
14	1442	6	350
8	2501	4	1310
12	1300	9	400
10	1789	11	500

- Age, Sft., Location is independent variables (X) and
- Price is dependent variable or Target variable (y)

Step 1: Calculate the average/mean of the target variable.

- $$\frac{480+1090+350+1310+400+500}{6} = 688$$

Age	Sft.	Location	Price	Average_Price
5	1500	5	480	688
11	2030	12	1090	688
14	1442	6	350	688
8	2501	4	1310	688
12	1300	9	400	688
10	1789	11	500	688

Step 2: Calculate the residuals (errors) for each sample.

•

Residual=Actual Value - Predicted Value

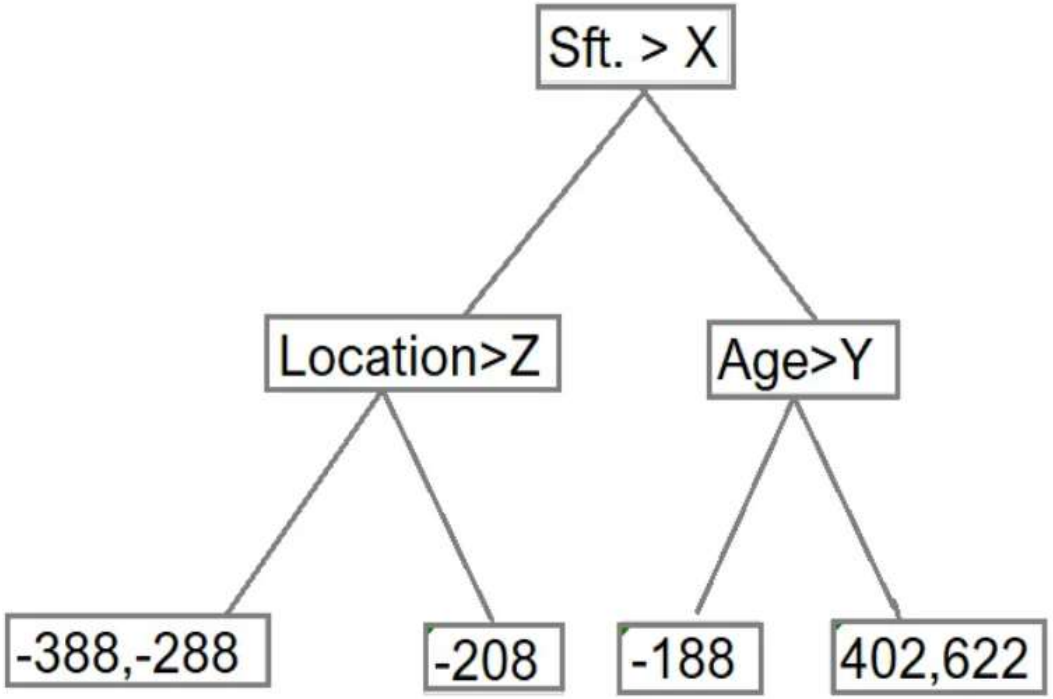
Age	Sft.	Location	Price	Average_Price	Residual
5	1500	5	480	—	688	-208
11	2030	12	1090	—	688	402
14	1442	6	350	—	688	-338
8	2501	4	1310	—	688	622
12	1300	9	400	—	688	-288
10	1789	11	500	—	688	-188

Error

Step 3: Construct a decision tree (Base Model)

- We build a tree with the goal of predicting the Residuals.

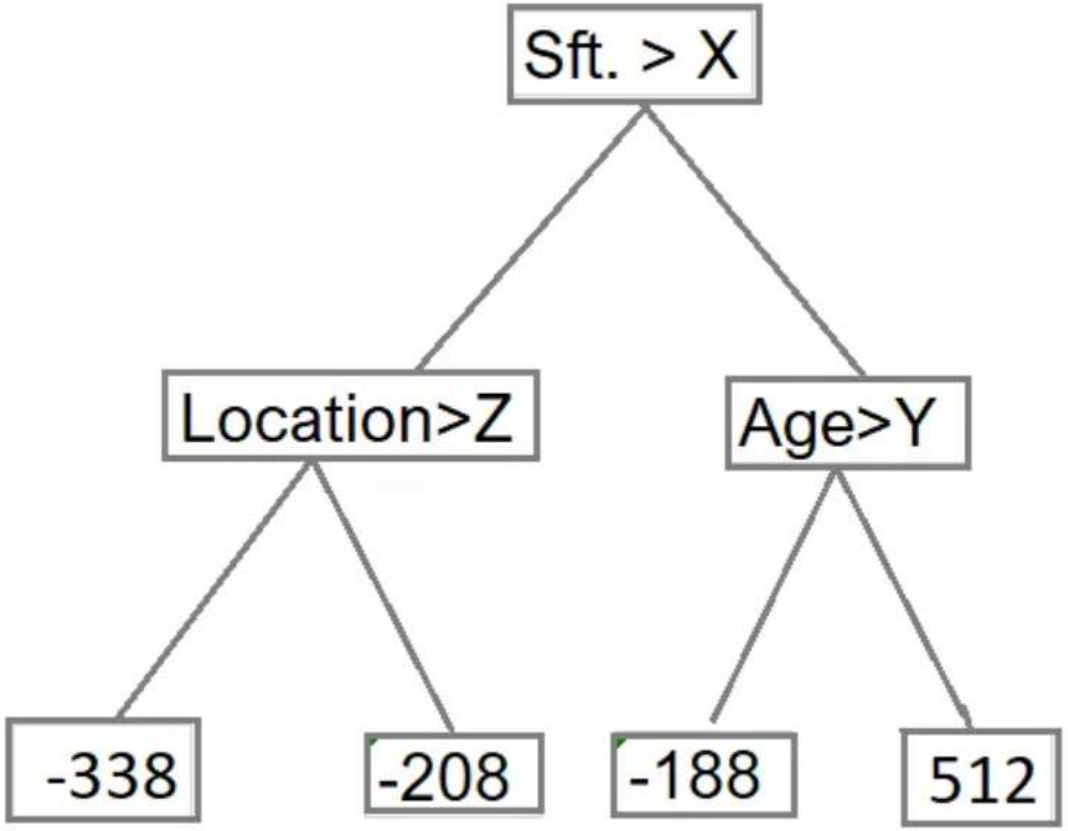
  - In the event if there are more residuals then leaf nodes(here its 6 residuals),some residuals will end up inside the same leaf.
  - When this happens, we compute their average and place that inside the leaf.



$$\frac{(-388)+(-288)= -338}{2}$$

$$\frac{402+622= 512}{2}$$

- After updating the previously calculated values:
- Tree become like this:
-



Step 4: Predict the target label using all the trees within the ensemble.

- Each sample passes through the decision nodes of the newly formed tree until it reaches a given lead.
- The residual in the said leaf is used to predict the house price.
- =>Average Price + Learning Rate * Residual Predicted by decision Tree

=>688 + 0.1 * (-338)

Predicted Price => 654.2

=> 688 + 0.1 * (-208)

Predicted Price=> 667.2

- **Calculation above for Residual value (-338) and (-208) in Step 2**
- Same way we will calculate the **Predicted Price** for other values
- **Note:** We have initially taken 0.1 as learning rate.
- **Step 5** : Compute the new residuals

When Price is 350 and 480 Respectively.

- Residual=Actual Value - Predicted Value

=>350-654.2 = -304.2

=>480-667.2 = -187.2

- With our Single leaf with average value**(688)** we have the below column of Residual.

Age	Sft.	Location	Price	Average_Price	Residual
5	1500	5	480	688	-208
11	2030	12	1090	688	402
14	1442	6	350	688	-338
8	2501	4	1310	688	622
12	1300	9	400	688	-288
10	1789	11	500	688	-188

- With our decision tree ,we ended up the below new residuals.

Age	Sft.	Location	Price	Average_Price	Residual	New Residual
5	1500	5	480	688	-208	-187.2
11	2030	12	1090	688	402	350.8
14	1442	6	350	688	-338	-304.2
8	2501	4	1310	688	622	570.8
12	1300	9	400	688	-288	-254.1
10	1789	11	500	688	-188	-169.2

- **Step 6**: Repeat steps 3 to 5 until the number of iterations matches the number specified by the hyper parameter(numbers of estimators)
- **Step 7**:
- Once trained, use all of the trees in the ensemble to make a final prediction as to value of the target variable.
- The final prediction will be equal to the mean we computed in Step 1 plus all the residuals predicted by the trees that make up the forest multiplied by the learning rate.

=>Average Price + LR * Residual Predicted by DT 1 + LR * Residual Predicted by DT 2 + ...+LR*Residual Predicted by DT N

=> 688 + 0.1 * (-188) + 0.1 *(-169.2) +.....

- **Advantages of Gradient Boosting**

1. Most of the time predictive accuracy of gradient boosting algorithm on Higher side.

2. It provides lots of flexibility and can optimize on different loss functions and provides several hyper parameter tuning options that make the function fit very flexible.

3. Most of the time no data pre-processing required.

4. Gradient Boosting algorithm works great with categorical and numerical data.

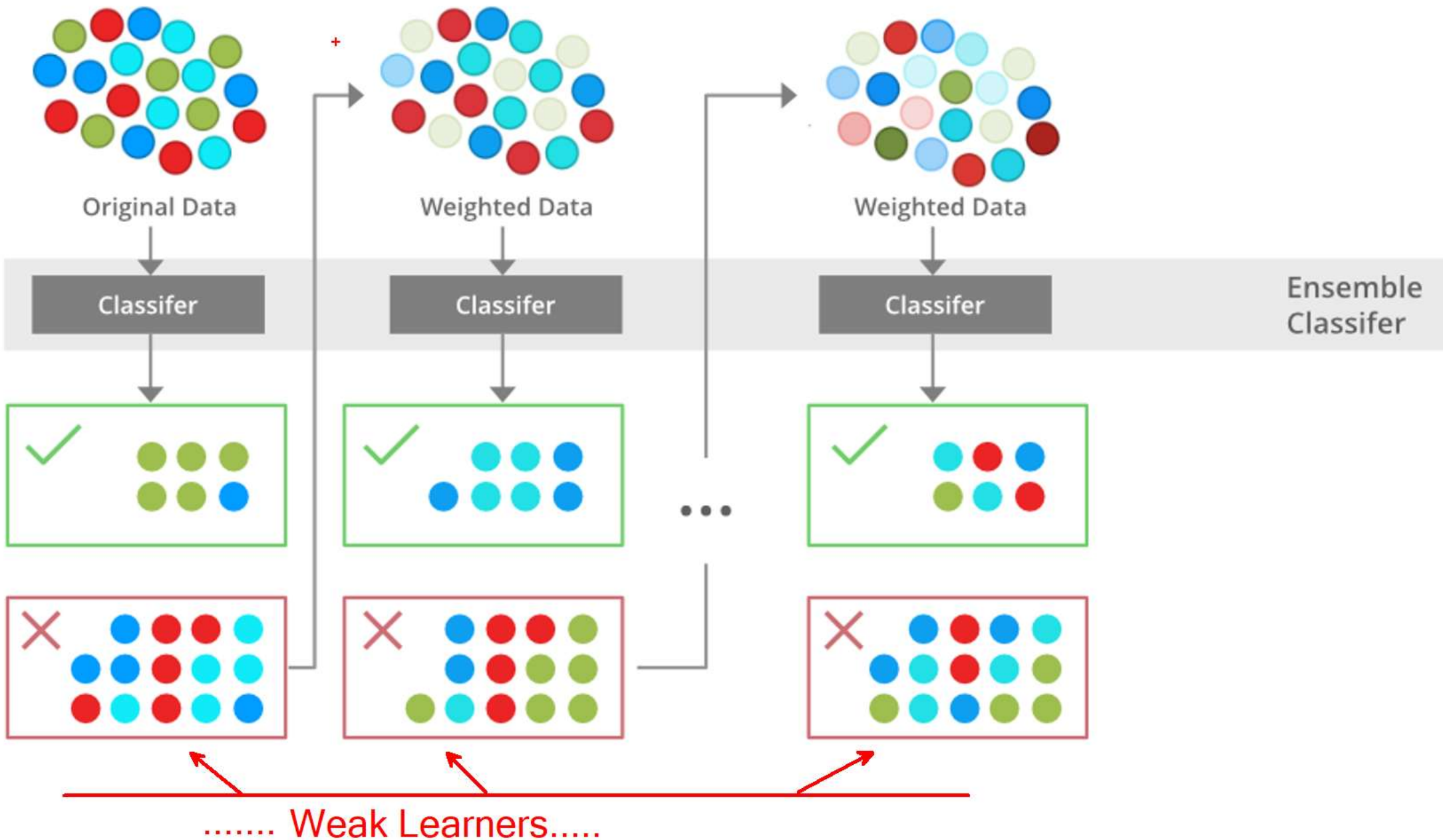
5. Handles missing data — missing value imputation not required.

Adaptive (Ada) Boosting

The key idea is to Focus on the samples that are difficult to classify(Wrongly Classfied), there by improve overall Accuracy.

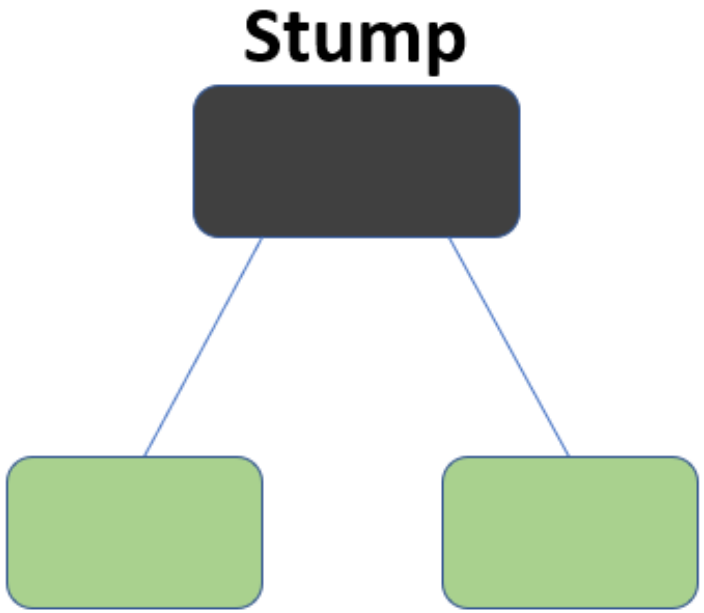
It is called Adaptive Boosting
as
the Weights are re-assigned to each instance, with higher weights to incorrectly classified instances.

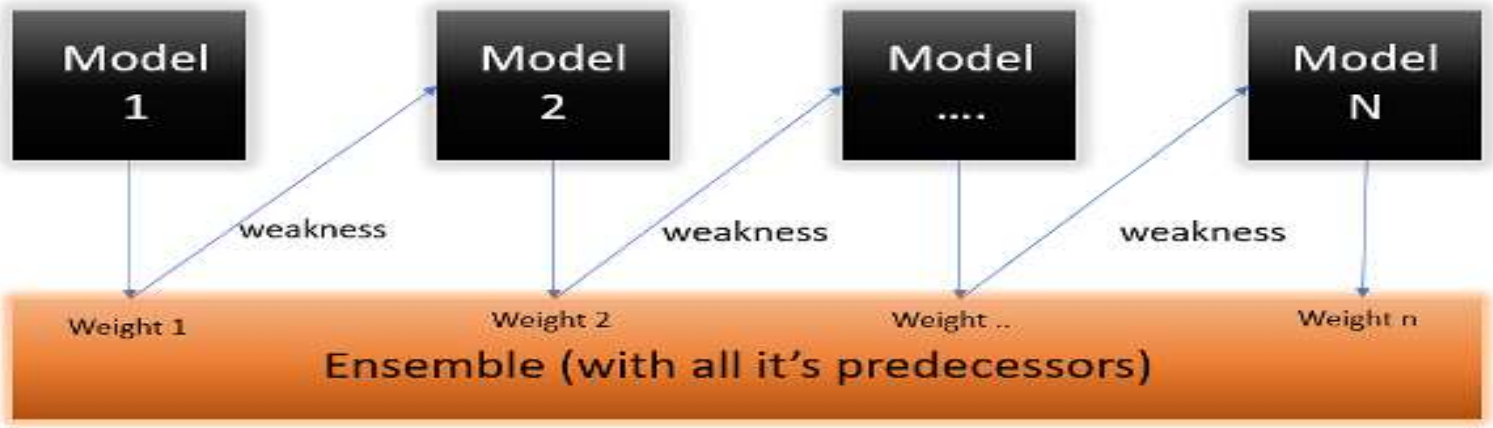
- AdaBoost (or) Adaptive Boosting – Ensemble (Boosting) Technique
- The key idea is to improve the performance of Weak Learners and create a Strong Predictive Model.
- It works by iteratively training a sequence of Weak Classifiers and giving more Weight to the instances that were Misclassified in previous rounds.
- This allows the model to **Focus** on the harder-to-classify samples.
- The *Final Prediction is typically a Weighted combination of the weak classifiers.*
- It's important to note that AdaBoost can be sensitive to noisy data and outliers.
- AdaBoost is a powerful ensemble technique and has been successfully applied in various machine learning tasks, including classification and regression problems.


Original Data
Weighted Data
Weighted Data
Classifer
Classifer
Classifer
Ensemble Classifer
..... Weak Learners.....

- Improve the prediction power by **converting a number of weak learners to strong learners.**
- Combines Multiple models (weak learners) to reach the final output (strong learners)
- The principle behind boosting algorithms is :
- *We first build a model on the training dataset and then build a second model to rectify the errors present in the first model.*
- *This procedure is continued until and unless the errors are minimized and the dataset is predicted correctly.*

- What this algorithm does is :
- it builds a model and gives equal weights to all the data points.
- It then assigns higher weights to points that are wrongly classified.
- Now all the points with higher weights are given more importance in the next model.
- It will keep training models until and unless a lower error is received.





AdaBoost working process (Steps)

1. Initialize weights:

- Assign equal weights to all training samples.
- These weights determine the importance of each sample in the training process.

2. Train a weak classifier:

- Fit a weak learner (usually a decision tree with limited depth or a simple model) on the training data using the current sample weights.

3. Calculate error:

- Calculate the weighted error rate of the weak classifier, where the weights emphasize the misclassified samples.

4. Update classifier weight:

- Calculate a weight for the trained classifier based on its accuracy.
- Better classifiers are assigned higher weights.

5. Update sample weights:

- Increase the weights of the misclassified samples, making them more likely to be chosen in the next round of training.

6. Repeat steps 2-5:

- Continue this process for a predefined number of rounds or until a desired level of accuracy is achieved.

7. Final ensemble prediction:

- Combine the predictions of all weak classifiers by weighting them based on their individual classifier weights.
- Typically, the final prediction is made by majority voting or weighted voting.

Example – Ada Boosting

Maths behind the ADA Boosting

Understanding the Working of the AdaBoost Algorithm

- Step 1: Assigning Weights
-

Row No.	Gender	Age	Income	Illness	Sample Weights
1	Male	41	40000	Yes	1/5
2	Male	54	30000	No	1/5
3	Female	42	25000	No	1/5
4	Female	40	60000	Yes	1/5
5	Male	46	50000	Yes	1/5

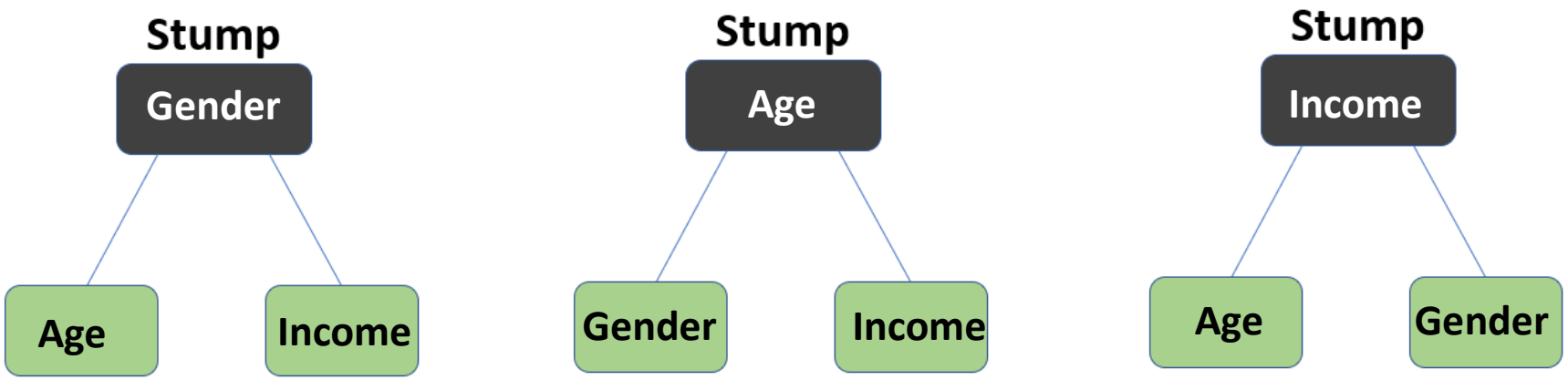
Here since we have 5 data points, the sample weights assigned will be 1/5.

The formula to calculate the sample weights is:

$$w(x_i, y_i) = \frac{1}{N}, \; i = 1, 2, \ldots . n$$

Where N = total number of data points

- Step 2: Classify the Samples
- We start by seeing how well "*Gender*" classifies the samples and
- Similarly, we will see how the variables "Age" and "Income" classify the samples.



- We'll create a multiple decision stump (decision trees) for each of the features.
- then calculate the ***Entropy*** of each stump (tree).
- The tree with the **Lowest Entropy (Stump)** will be our first stump.(first model)
- Here in our dataset, let's assume ***Gender*** has the lowest entropy, so it will be our first stump.

- Step 3: Calculate the Influence/Amount of Say/Importance:
- We'll now calculate the "Influence" for this classifier in classifying the data points.
- It is calculated using the formulae:

$$\frac{1}{2}\log\frac{1 - Total\ Error}{Total\ Error}$$

- Where,
- The total error = Summation of all the sample weights of misclassified data points.
- Assume, in our dataset, there is 1 wrong output (1 wrongly classified)
- So, Total error = 1/5.

- Performance of the stump will be:
- **Note**: Total error will always be between 0 and 1.
- 0 Indicates perfect stump, and 1 indicates horrible stump.
- $$\boxed{Performance\ of\ the\ stump\ =\ \frac{1}{2}\log_e\left(\frac{1-Total\ Error}{Total\ Error}\right)}$$

$$\alpha\ =\ \frac{1}{2}\log_e\left(\frac{1-\frac{1}{5}}{\frac{1}{5}}\right)$$

$$\alpha\ =\ \frac{1}{2}\log_e\left(\frac{0.8}{0.2}\right)$$

$$\alpha\ =\ \frac{1}{2}\log_e(4)\ =\ \frac{1}{2}*(1.38)$$

$$\alpha\ =\ 0.69$$

is called "Influence“ OR "Importance" OR "Amount of Say"

- Step 4: Update Weights
- We need to update the weights because *if the same weights are applied to the next model,* then the output received will be the same as what was received in the first model.
- So, **wrong predictions** will be given more weight,
- whereas the **correct predictions** weights will be decreased.
- Now when we build our next model after updating the weights, *more preference will be given to the points with higher weights.*

- To Update the weights, we use the following formula:

$$New\ sample\ weight\ =\ old\ weight\ *\ e^{\pm Amount\ of\ say\ (\alpha)}$$

- The amount of, say (alpha) will be negative when the sample is correctly classified (Decrease the weights)
- The amount of, say (alpha) will be positive when the sample is miss-classified. (Increase the weights)

- There are : 4 correctly classified samples and 1 wrong.

- Here, the sample weight of that datapoint is 1/5, and the amount of say/performance of the stump of Gender is 0.69.

- New weights for correctly classified samples are: **(Decrease the weights)**

$$New\ sample\ weight\ =\ \frac{1}{5}\ *\ \boxed{\exp(-0.69)}$$

$$New\ sample\ weight\ =\ 0.2\ *\ 0.502\ =\ 0.1004$$

- For *wrongly classified* samples, the updated weights will be: **(Increase the weights)**

-

$$New\ sample\ weight\ =\ \frac{1}{5}\ *\ \boxed{\exp(0.69)}$$

$$New\ sample\ weight\ =\ 0.2\ *\ 1.994\ =\ 0.3988$$

- the alpha is negative when the data point is correctly classified, and this decreases the sample weight from 0.2 to 0.1004.
- It is positive when there is misclassification, and this will increase the sample weight from 0.2 to 0.3988

Row No.	Gender	Age	Income	Illness	Sample Weights	New Sample Weights
1	Male	41	40000	Yes	1/5	0.1004
2	Male	54	30000	No	1/5	0.1004
3	Female	42	25000	No	1/5	0.1004
4	Female	40	60000	Yes	1/5	0.3988
5	Male	46	50000	Yes	1/5	0.1004

0.1004+
0.1004+
0.1004+
0.3988+
0.1004
= 0.8004

- We know that the total sum of the sample weights must be equal to 1,
- but here , the total sum (updated weight) =0.8004.
- To bring this sum equal to 1, we will NORMALIZE THESE WEIGHTS by dividing all the weights by the total sum of updated weights, which is 0.8004.
- So, after normalizing the sample weights, sum is equal to 1.
-

Row No.	Gender	Age	Income	Illness	Sample Weights	New Sample Weights
1	Male	41	40000	Yes	1/5	0.1004/0.8004 =0.1254
2	Male	54	30000	No	1/5	0.1004/0.8004 =0.1254
3	Female	42	25000	No	1/5	0.1004/0.8004 =0.1254
4	Female	40	60000	Yes	1/5	0.3988/0.8004 =0.4982
5	Male	46	50000	Yes	1/5	0.1004/0.8004 =0.1254

- Step 5: Decrease Errors
- Now, we need to make a new dataset to see if the errors decreased or not.
- For this, we will remove the “sample weights” and “new sample weights” columns and then, based on the “new sample weights,” *divide our data points into buckets.*
-

Row No.	Gender	Age	Income	Illness	New Sample Weights	Buckets
1	Male	41	40000	Yes	0.1004/0.8004= 0.1254	0 to 0.1254
2	Male	54	30000	No	0.1004/0.8004= 0.1254	0.1254 to 0.2508
3	Female	42	25000	No	0.1004/0.8004= 0.1254	0.2508 to 0.3762
4	Female	40	60000	Yes	0.3988/0.8004= 0.4982	0.3762 to 0.8744
5	Male	46	50000	Yes	0.1004/0.8004= 0.1254	0.8744 to 0.9998

- Step 6: New Dataset

- Now, what the algorithm selects random numbers from 0-1.

- Suppose the 5 random numbers our algorithm take is 0.38,0.26,0.98,0.40,0.55.

- Now we will see where these random numbers fall in the bucket, and according to it, we'll make our new dataset shown below.

-

Row No.	Gender	Age	Income	Illness
1	Female	40	60000	Yes
2	Male	54	30000	No
3	Female	42	25000	No
4	Female	40	60000	Yes
5	Female	40	60000	Yes

- Step 7: Repeat Previous Steps

- Now this act as our new dataset, and we need to repeat all the above steps i.e.

1. Assign ***equal weights*** to all the data points.

2. Find the stump that does the ***best job classifying*** the new collection of samples by finding their Gini Index and selecting the one with the lowest Gini index.

3. Calculate the ***"Amount of Say"*** and ***"Total error"*** to update the previous sample weights.

4. Normalize the new sample weights.

- Iterate through these steps until and unless a low training error is achieved.

- Suppose, with respect to our dataset:
- For Instance, We have constructed 3 decision trees (DT1, DT2, DT3) in a sequential manner.
- If we send our test data now, it will pass through all the decision trees, and
- Finally, we will see which class has the majority, and based on that, we will do predictions for our test dataset.

Random Forest

- It is a type of supervised learning algorithm that combines the predictions from multiple decision trees to improve overall prediction accuracy and reduce overfitting.
- it can be used for both Classification and Regression problems.
- Radom Forest basically : Bagging Technique (Bootstraping) - reduces model's variance
- A higher variance means that your model is overfitted.
- Predefined number of decision trees are constructed, each tree is constructed independently with its bootstrapped sample and feature subset.
- **Randomness in Training**:
- The "Random" part of Random Forest comes from two main sources of randomness during training:

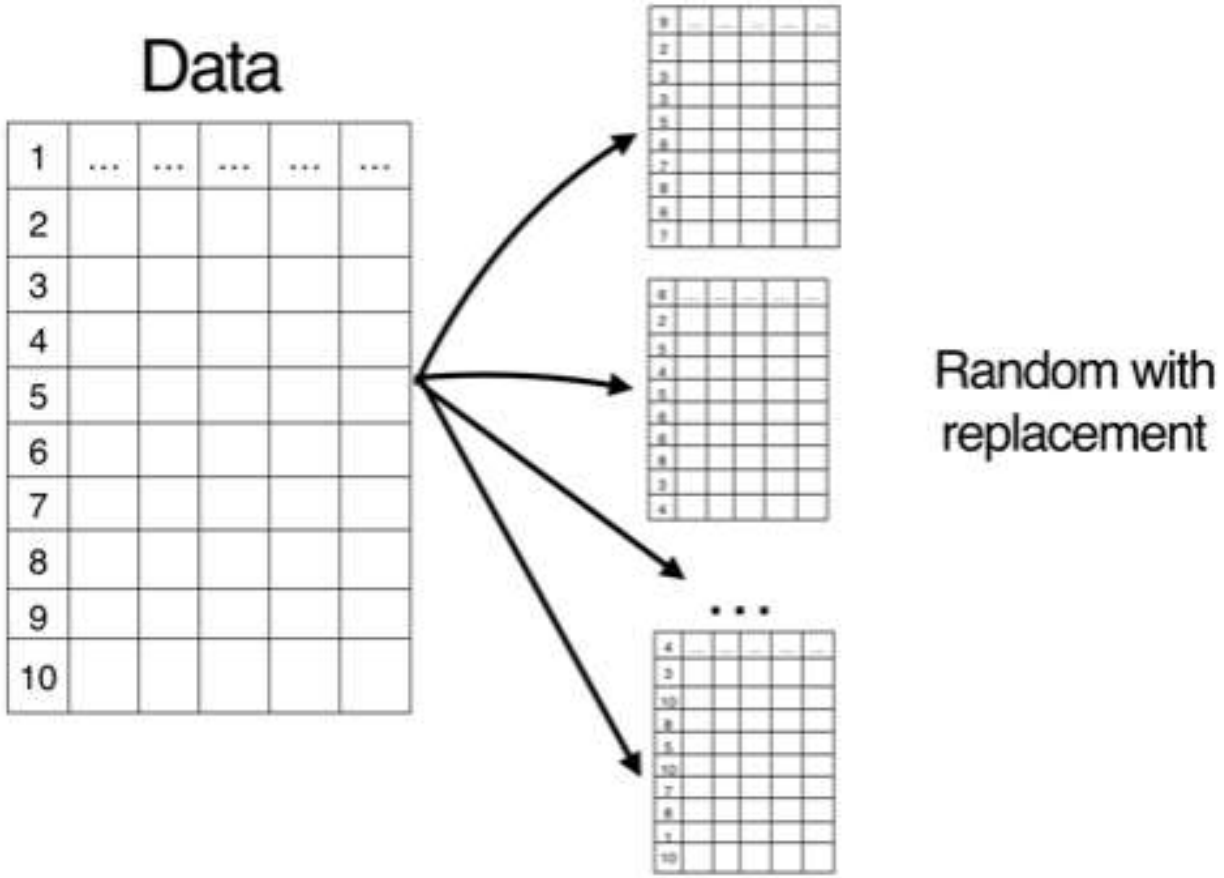
1. Bootstrapped Samples (Bagging):

- Random Forest uses a technique called bootstrapping, where it creates multiple random subsets of the training data with replacement.
- Each decision tree in the forest is trained on one of these bootstrapped samples.
- This process introduces diversity into the individual trees.

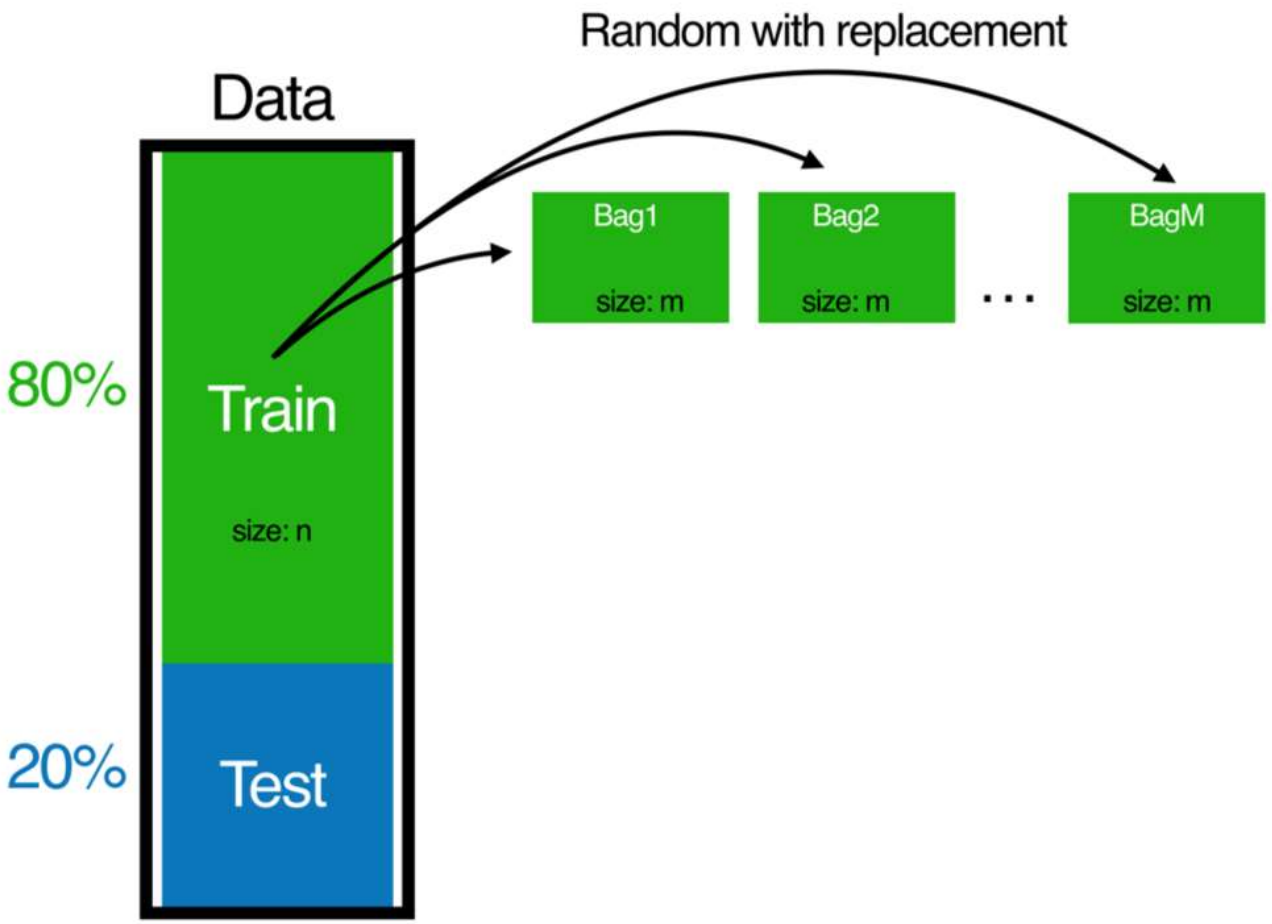
2. Random Subset of Features:

- When splitting each node of a decision tree, Random Forest considers only a random subset of features rather than all available features.
- This helps in decorrelating the trees and preventing them from becoming too similar.

Random Sampling - Bootstrap Sampling



- With Bagging, we use a single training example more than once in different models.
- This results in a modified version of the training set where some rows are represented multiple times and some are absent.
- By doing this, you can fit many different but similar models.

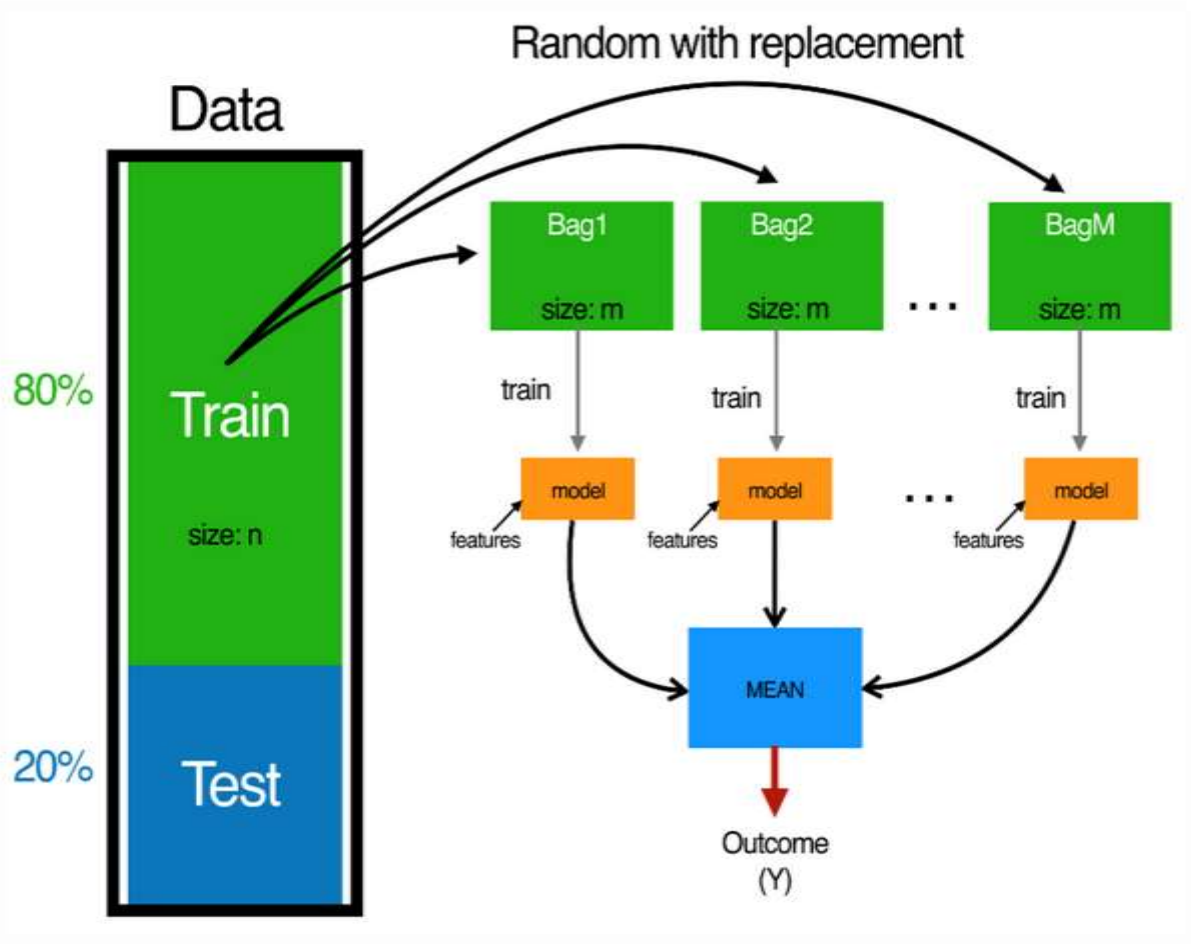


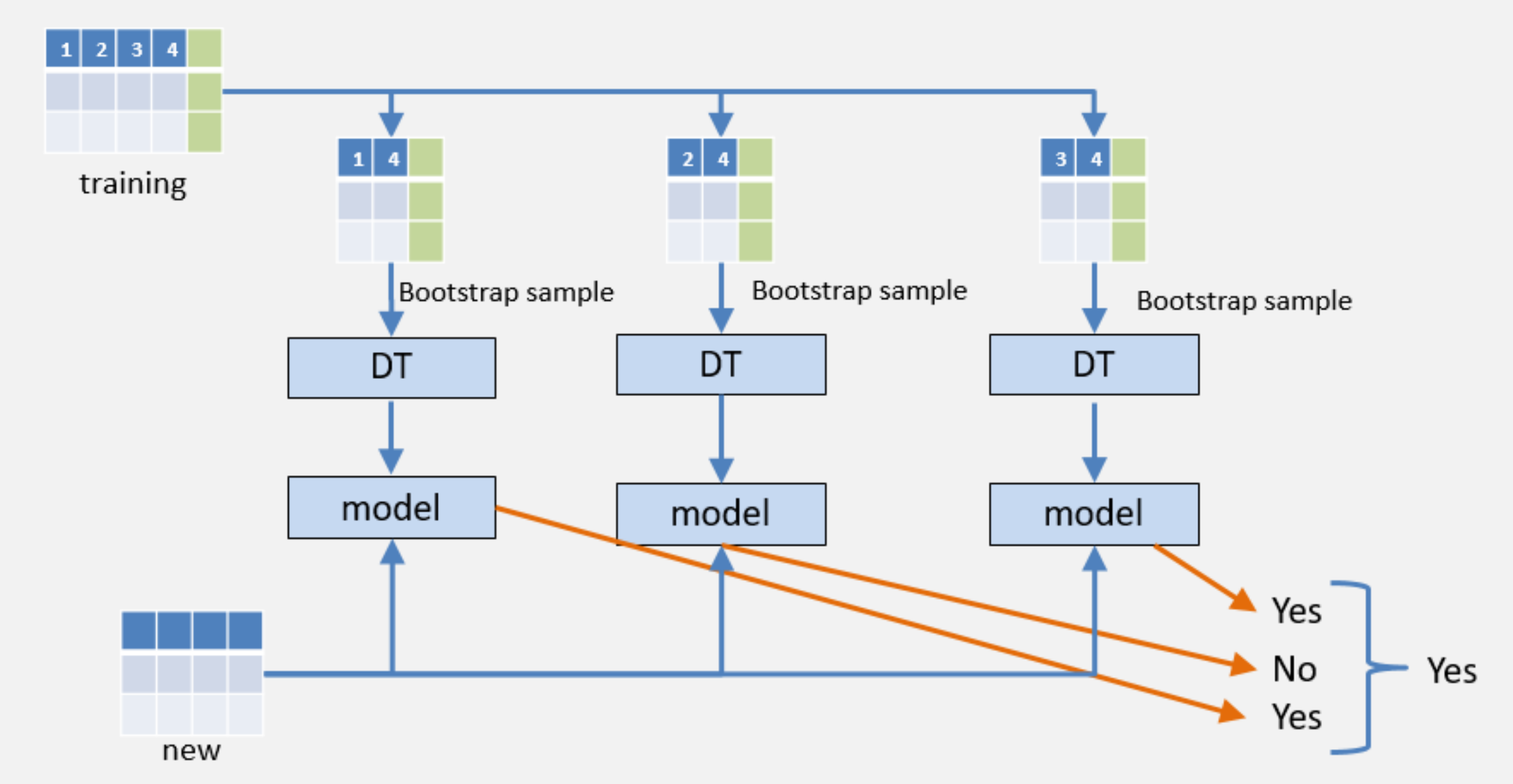
Step 1:

- You draw B samples with replacement from the original data set.
- Where B is a number less than or equal to N, the total number of samples in the training set.

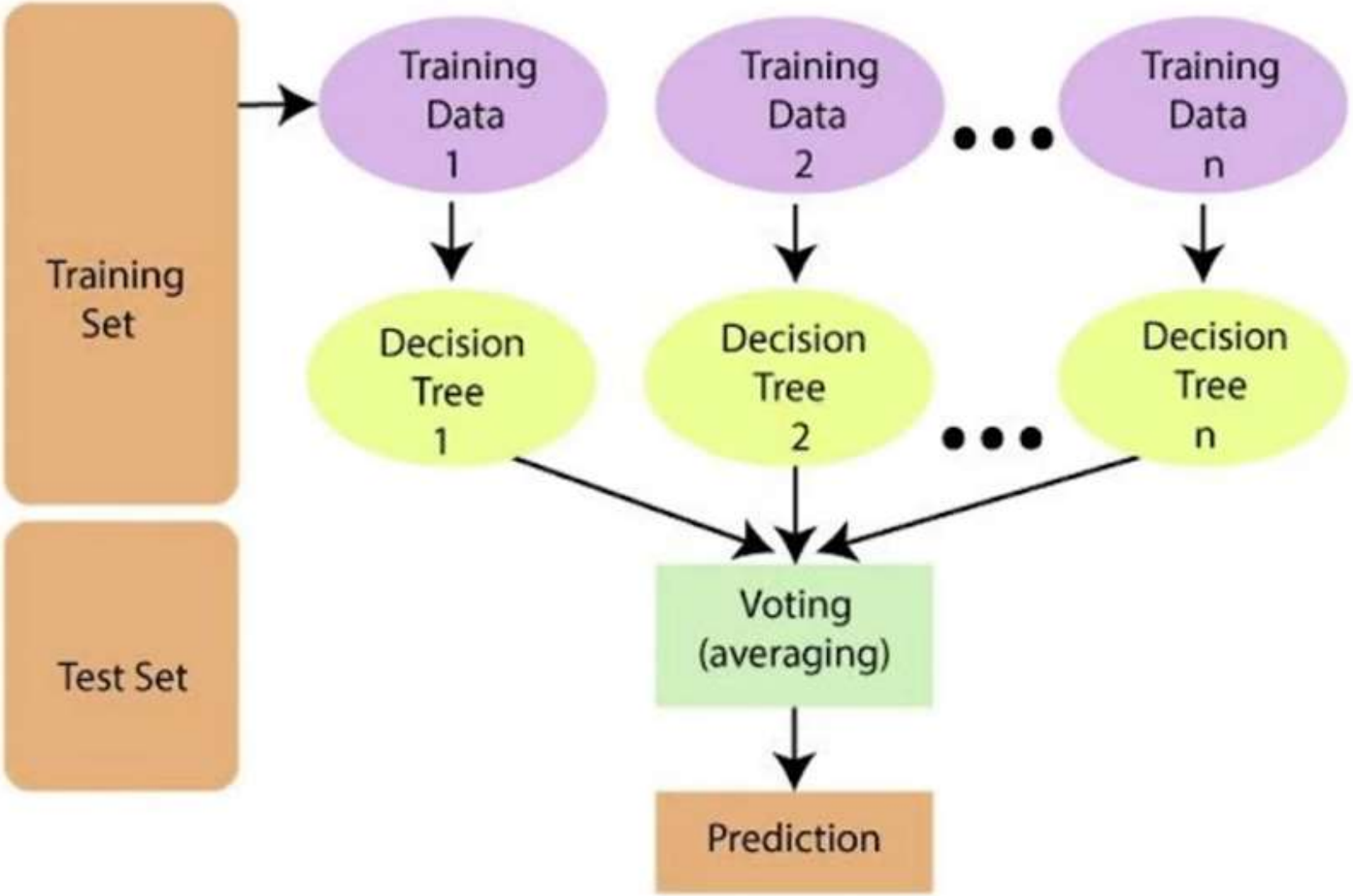
Step 2:

- Train a decision trees on newly created bootstrapped samples.
- Repeat the Step1 and Step2 any number of times that you like.
- Generally, higher the number of trees, the better the model.
- But remember! Excess number of trees can make a model complicated and ultimately lead to overfitting.




1 2 3 4
training
1 4
2 4
3 4
Bootstrap sample
Bootstrap sample
Bootstrap sample
DT
DT
DT
model
model
model
new
Yes
No
Yes
Yes

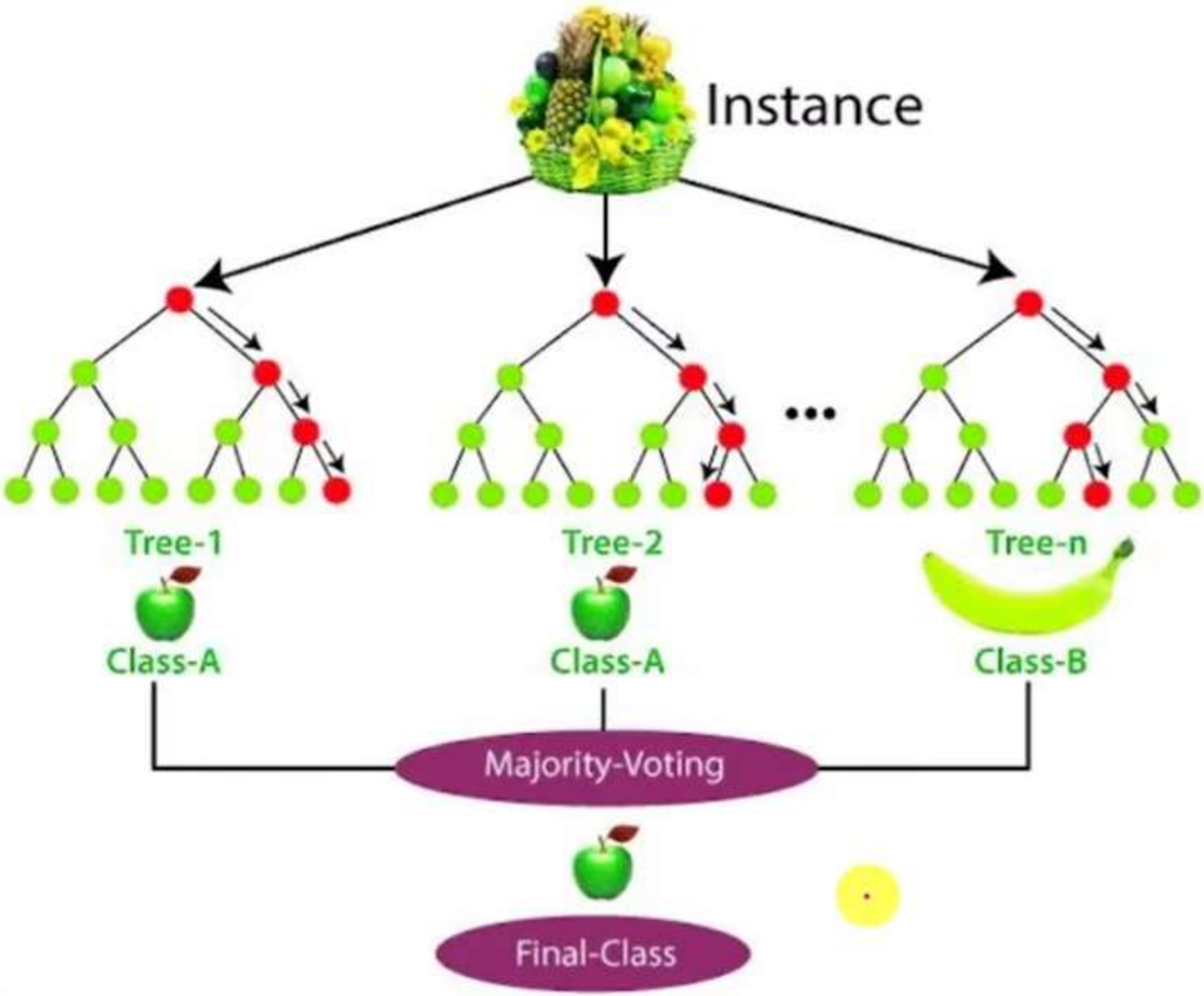
Random Forest



- **Voting (Classification) or Averaging (Regression)**:
- Once the ensemble of decision trees is built, predictions are made by each tree for a given input data point.
- In classification tasks, the most common class predicted by the trees is taken as the final prediction (majority vote).
- In regression tasks, the average of the individual tree predictions is taken as the final prediction.
- Hyperparameters:
- It includes:
  - the number of trees in the forest,
  - the depth of individual trees, and
  - the size of the random feature subsets.

Random Forest Algorithm - Steps

1. Build random forests
   a) If the number of examples in the training set is N, take a sample of n examples at random - but with replacement, from the original data. This sample will be the training set for generating the tree.
   b) If there are M input variables, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the generation of the various trees in the forest.
   c) Each tree is grown to the largest extent possible.


Instance
Tree-1
Tree-2
Tree-n
Class-A
Class-A
Class-B
Majority-Voting
Final-Class

- **Difference Between Decision Tree & Random Forest**

Decision Tree	Random Forest
• They frequently experience the issue of overfitting.	• The issue of overfitting doesn't arise here because they are built from subsets of data, and the outcome is based on average or majority ratings.
• faster to compute.	• It is slower.
• When a feature-rich data collection is used as input, they apply certain criteria.	• Random Forest builds a decision tree from observations that are chosen at random, and then the outcome is determined by majority voting.