Module - 4

Module:4	Combining Multiple Learners	6 hours
Generating Diverse Learners - Model Combination Schemes - Voting - Error Correcting Output Codes - Bagging - Boosting - The Mixture of Experts - Stacking - Random Forest Classifier		

Dr. A. Anitha,
Professor
Department of Software and Systems Engineering,
School of Computer Science Engineering and Information Systems,
Vellore Institute of Technology, 632014

Define these terms

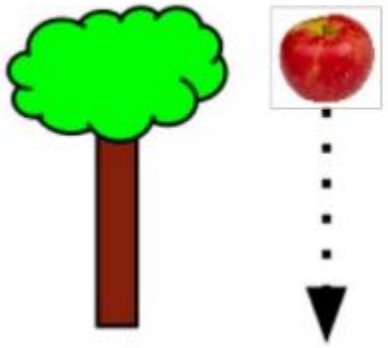
- Learning –

  The process of optimizing the model so that it can predict the correct response based on the training data samples
- Training –
- The process of fitting a particular model to a dataset is known as training.

Why training is not called learning

- First, note that the learning process does not end with the step of data abstraction.
- Learning requires an additional step to generalize the knowledge to future data.
- Second, the term training more accurately describes the actual process undertaken when the model is fitted to the data

Observations → **Data** → **Model**



velocity	time
9.8	1
39.2	2
88.2	3
156.8	4
245	5

$$g = 9.8\ m/s^2$$

Ensemble learning

- An ensemble is a composite model, combines a series of low performing classifiers with the aim of creating an improved classifier.
- Here, individual classifier vote and final prediction label returned that performs majority voting.
- Ensembles offer more accuracy than individual or base classifier.
- Ensemble methods can parallelize by allocating each base learner to different-different machines.
- Finally, you can say Ensemble learning methods are meta-algorithms that combine several machine learning methods into a single predictive model to increase performance.
- Ensemble methods can decrease variance using bagging approach, bias using a boosting approach, or improve predictions using stacking approach.

An Example

- Let's take a real example to build the intuition.
- Suppose, you want to invest in a company XYZ. You are not sure about its performance though.
- So, you look for advice on whether the stock price will increase by more than 6% per annum or not?
- You decide to approach various experts having diverse domain experience:

Survey prediction

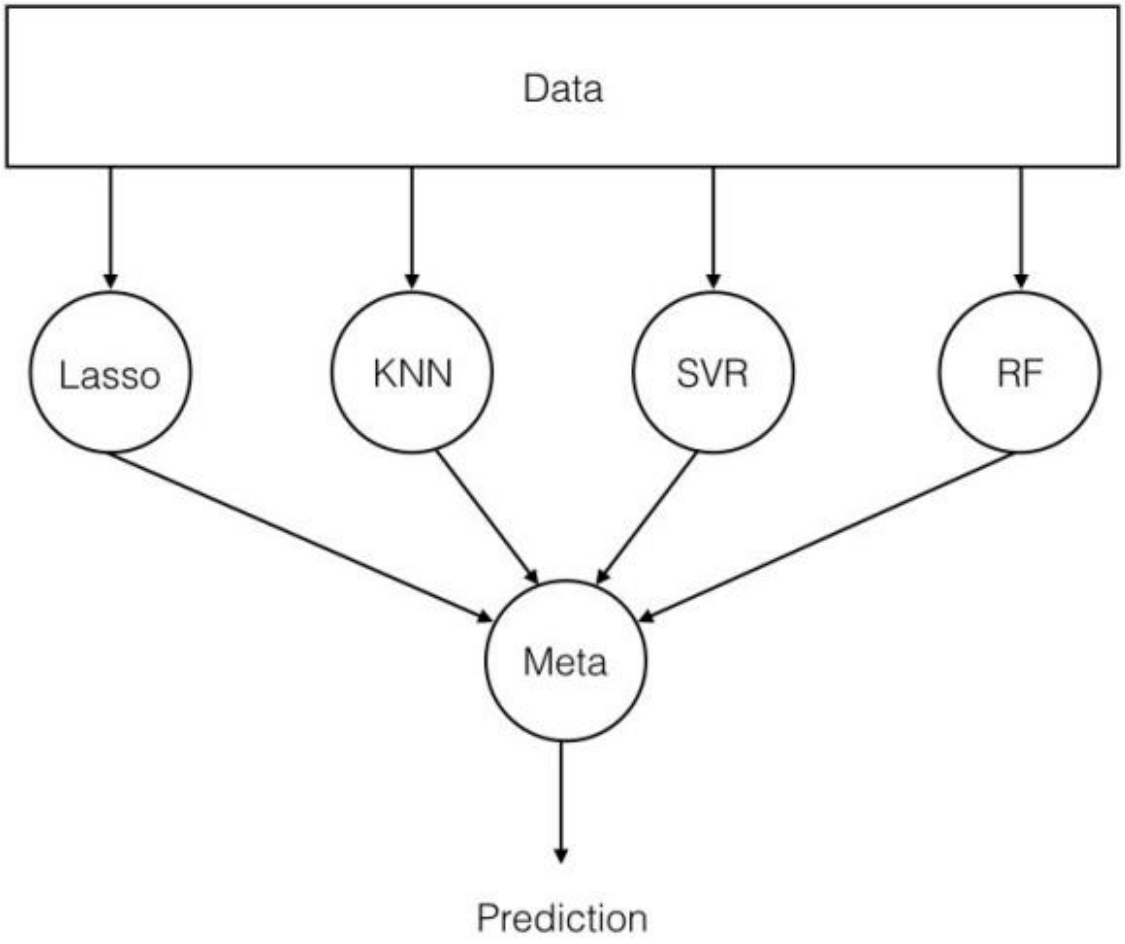
- Employee of Company XYZ: – In the past, he has been right 70% times.
- Financial Advisor of Company XYZ: – In the past, he has been right 75% times.
- Stock Market Trader: – In the past, he has been right 70% times.
- Employee of a competitor: – In the past, he has been right 60% times.
- Market Research team in the same segment: – In the past, he has been right 75% times.
- Social Media Expert: – In the past, he has been right 65% times.

- Given the broad spectrum of access you have, you can probably combine all the information and make an informed decision.
- In a scenario when all the 6 experts/teams verify that it's a good decision(assuming all the predictions are independent of each other), you will get a combined accuracy rate of 1 - (30% . 25% . 30% . 40% . 25% . 35%) = 1 - 0.07875 = 99.92125%
- The assumption used here that all the predictions are completely independent is slightly extreme as they are expected to be correlated. However, you can see how we can be so sure by combining various forecasts together.
- Well, Ensemble learning is no different

Ensembling

- An ensemble is the art of combining a diverse set of learners (individual models) together to improvise on the stability and predictive power of the model.
- In our example, the way we combine all the predictions collectively will be termed as Ensemble learning.
- Moreover, Ensemble-based models can be incorporated in both of the two scenarios, i.e., when data is of large volume and when data is too little.

Basic ensemble



Which components to combine?

- • different learning algorithms
- • same learning algorithm trained in different ways
- • same learning algorithm trained the same way
- There are two steps in ensemble learning: Multiples machine learning models were generated using same or different machine learning algorithm. These are called “base models”. The prediction perform on the basis of base models.

Techniques/Methods in ensemble learning

- Voting
- Error-Correcting Output Codes
- Bagging: KNN or Random Forest Trees
- Boosting: Adaboost
- Stacking.

Model Combination Schemes

- -Combining Multiple Learners
  - Multiexpert combination
  - Multistage combination

Multiexpert combination

- Multiexpert combination methods have base-learners that work in parallel. These methods can in turn be divided into two:
- In the global approach, also called learner fusion, given an input, all base-learners generate an output and all these outputs are used. Examples are voting and stacking.
- In the local approach, or learner selection, for example, in mixture of experts, there is a gating model, which looks at the input and chooses one (or very few) of the learners as responsible for generating the output.

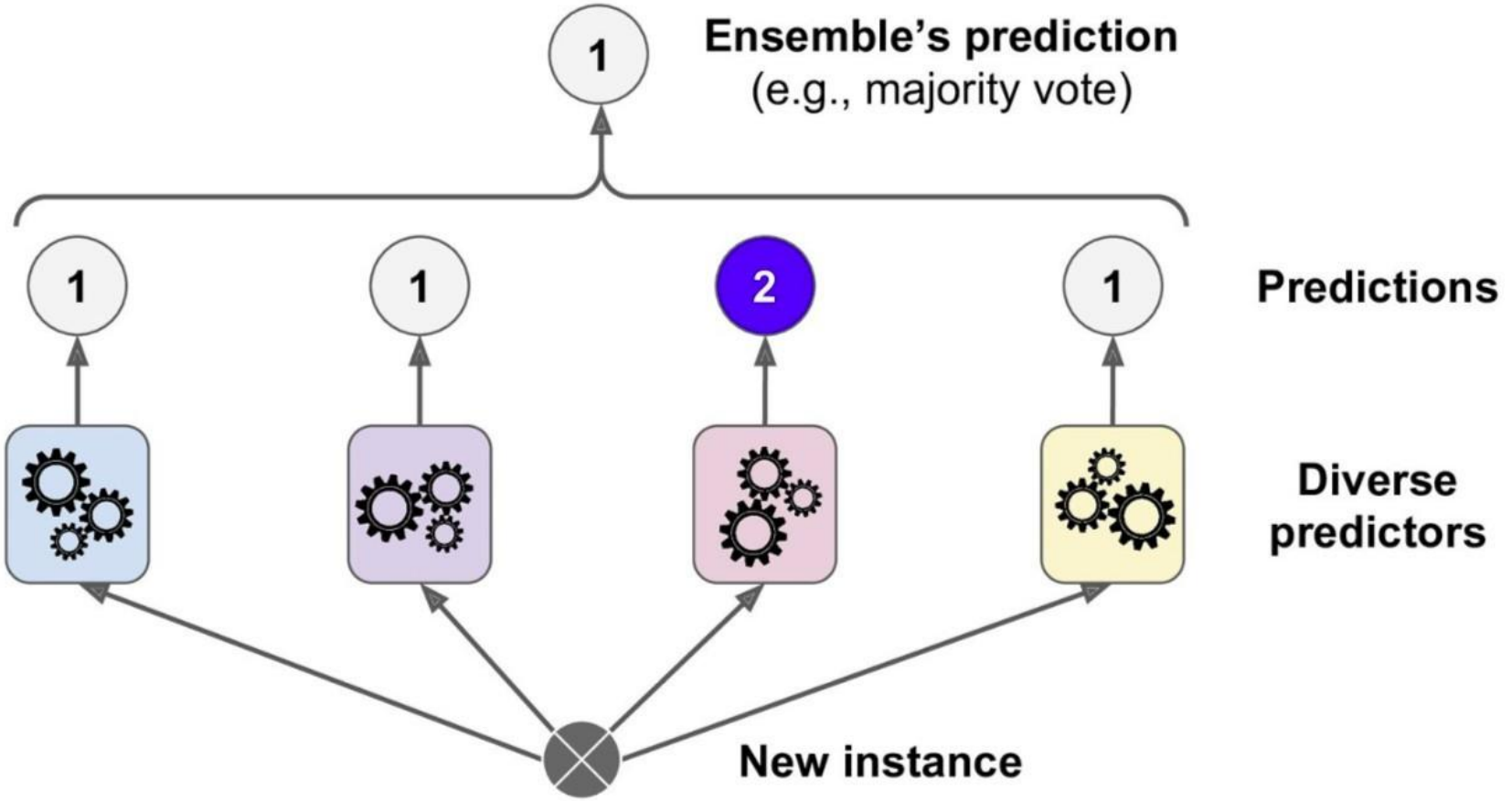
Multistage combination

- Multistage combination methods use a serial approach where the next base-learner is trained with or tested on only the instances where the previous base-learners are not accurate enough. The idea is that the base-learners (or the different representations they use) are sorted in increasing complexity so that a complex base-learner is not used (or its complex representation is not extracted) unless the preceding simpler base-learners are not confident. An example is cascading.

Voting

- Voting is an ensemble machine learning algorithm.
- For regression, a voting ensemble involves making a prediction that is the average of multiple other regression models.
- In classification, a hard voting ensemble involves summing the votes for crisp class labels from other models and predicting the class with the most votes.
- A soft voting ensemble involves summing the predicted probabilities for class labels and predicting the class label with the largest sum probability.

Voting


1
Ensemble's prediction
(e.g., majority vote)
1
1
2
1
Predictions
Diverse predictors
New instance

Voting

- There are two approaches to the majority vote prediction for classification; they are hard voting and soft voting.
- Hard voting involves **summing the predictions** for each class label and predicting the class label with the most votes. Soft voting involves summing the **predicted probabilities** (or probability-like scores) for each class label and predicting the class label with the largest probability.
  - Hard Voting. Predict the class with the largest sum of votes from models
  - Soft Voting. Predict the class with the largest summed probability from models.

Voting

- Use voting ensembles when:
  - All models in the ensemble have generally the same good performance.
  - All models in the ensemble mostly already agree.
- Hard voting is appropriate when the models used in the voting ensemble predict crisp class labels. Soft voting is appropriate when the models used in the voting ensemble predict the probability of class membership.

Voting

- Soft voting can be used for models that do not natively predict a class membership probability, although may require calibration of their probability-like scores prior to being used in the ensemble (e.g. support vector machine, k-nearest neighbors, and decision trees).
  - Hard voting is for models that predict class labels.
  - Soft voting is for models that predict class membership probabilities.
- The voting ensemble is not guaranteed to provide better performance than any single model used in the ensemble.

Voting

- Use a voting ensemble if:
  - It results in better performance than any model used in the ensemble.
  - It results in a lower variance than any model used in the ensemble.
- A voting ensemble is particularly useful for machine learning models that use a stochastic learning algorithm and result in a different final model each time it is trained on the same dataset.
- One example is neural networks that are fit using stochastic gradient descent.

Voting

- In the case of regression, this involves calculating the average of the predictions from the models.
- In the case of classification, the predictions for each label are summed and the label with the majority vote is predicted.
  - Regression Voting Ensemble: Predictions are the average of contributing models.
  - Classification Voting Ensemble: Predictions are the majority vote of contributing models.

Why ensemble

- There are two main reasons to use an ensemble over a single model, and they are related; they are:

– Performance: An ensemble can make better predictions and achieve better performance than any single contributing model.

– Robustness: An ensemble reduces the spread or dispersion of the predictions and model performance

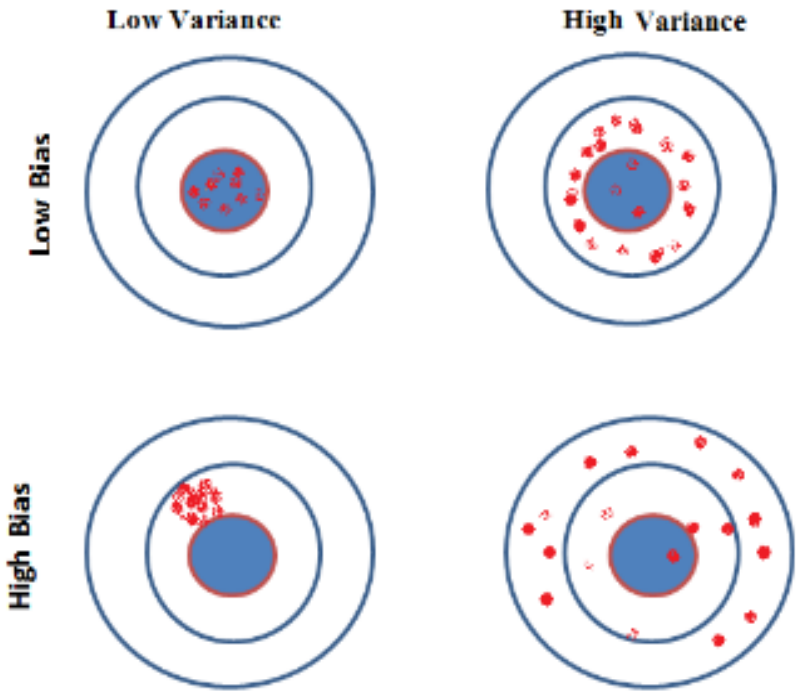
Model error

- The error emerging from any machine model can be broken down into three components mathematically. Following are these component:

$$Err(x) = \Big(E[\hat{f}\,(x)] - f(x)\Big)^2 + E\Big[\hat{f}\,(x) - E[\hat{f}\,(x)]\Big]^2 + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

- **Bias error** is useful to quantify how much on an average are the predicted values different from the actual value. A high bias error means we have an under-performing model which keeps on missing essential trends.
- **Variance** on the other side quantifies how are the prediction made on the same observation different from each other. A high variance model will over-fit on your training population and perform poorly on any observation beyond training.



Low-Bias, Low-Variance:
The combination is an ideal machine learning model. However, it is not possible practically.

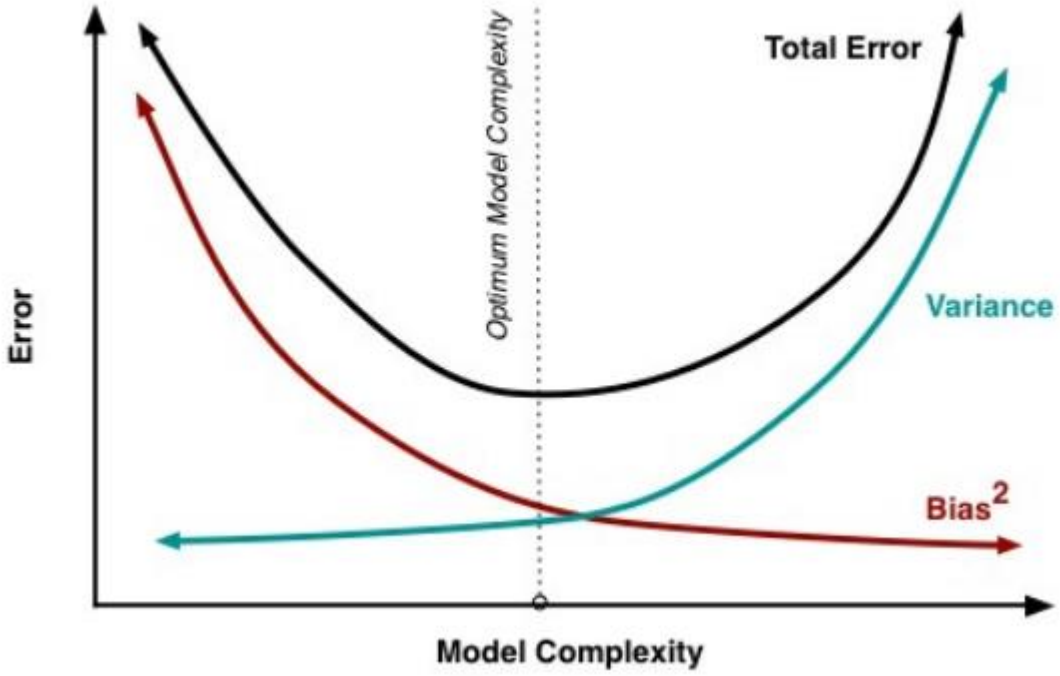
Low-Bias, High-Variance: This is a case of overfitting where model predictions are inconsistent and accurate on average. The predicted values will be accurate(average) but will be scattered.

High-Bias, Low-Variance: This is a case of underfitting where predictions are consistent but inaccurate on average. The predicted values will be inaccurate but will be not scattered.

High-Bias, High-Variance:
With high bias and high variance, predictions are inconsistent and also inaccurate on average.

Bias – variance Tradeoff

- Normally, as you increase the complexity of your model, you will see a reduction in error due to lower bias in the model. However, this only happens till a particular point.
- As you continue to make your model more complex, you end up over-fitting your model and hence your model will start suffering from high variance.
- A champion model should maintain a balance between these two types of errors. This is known as the trade-off management of bias-variance errors. Ensemble learning is one way to execute this trade off analysis.


Error
Optimum Model Complexity
Total Error
Variance
Bias^2
Model Complexity

Ensemble Creation Approaches

- Unweighted Voting (e.g. Bagging)
- Weighted voting – based on accuracy (e.g. Boosting), Expertise, etc.
- Stacking - Learn the combination function

Ensemble Learning Methods

- Bagging
- Boosting
- Stacking

Bootstrap

- The bootstrap is a powerful statistical method for estimating a quantity from a data sample. This is easiest to understand if the quantity is a descriptive statistic such as a mean or a standard deviation.
- Let's assume we have a sample of 100 values (x) and we'd like to get an estimate of the mean of the sample.
- We can calculate the mean directly from the sample as:

$$\text{mean}(x) = 1/100 * \text{sum}(x)$$

Bootstrap

- We know that our sample is small and that our mean has error in it. We can improve the estimate of our mean using the bootstrap procedure:
- Create many (e.g. 1000) random sub-samples of our dataset with replacement (meaning we can select the same value multiple times).
- Calculate the mean of each sub-sample.
- Calculate the average of all of our collected means and use that as our estimated mean for the data.

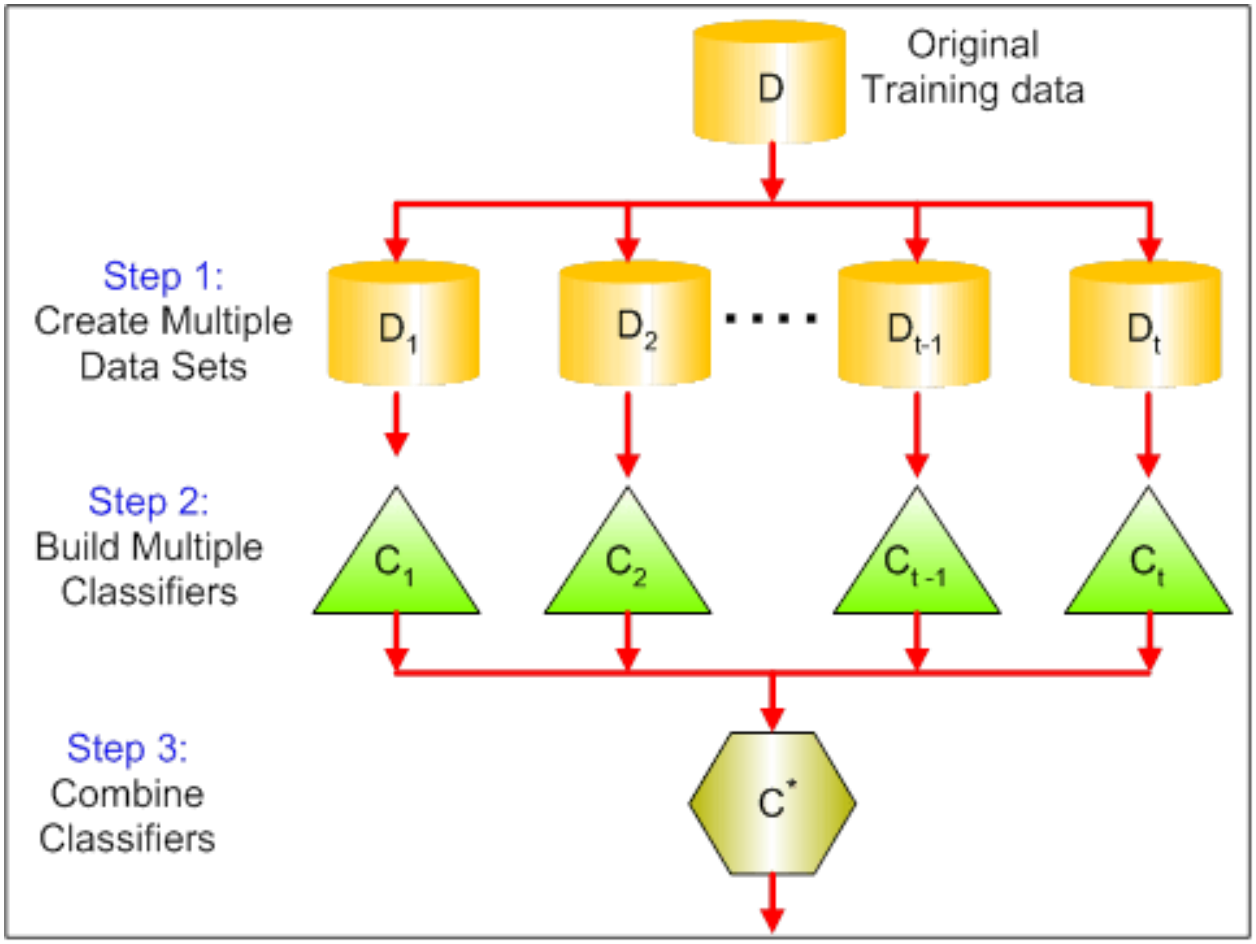
Bootstrap

- For example, let's say we used 3 resamples and got the mean values 2.3, 4.5 and 3.3.
- Taking the average of these we could take the estimated mean of the data to be 3.367.
- This process can be used to estimate other quantities like the standard deviation and even quantities used in machine learning algorithms, like learned coefficients.

Bagging

- Bagging stands for bootstrap aggregation.
- It combines multiple learners in a way to reduce the variance of estimates.
- For example, random forest / Decision tree trains M Decision Tree, you can train M different trees on different random subsets of the data and perform voting for final prediction.
- Example:
  - Random Forest
  - Extra Trees.

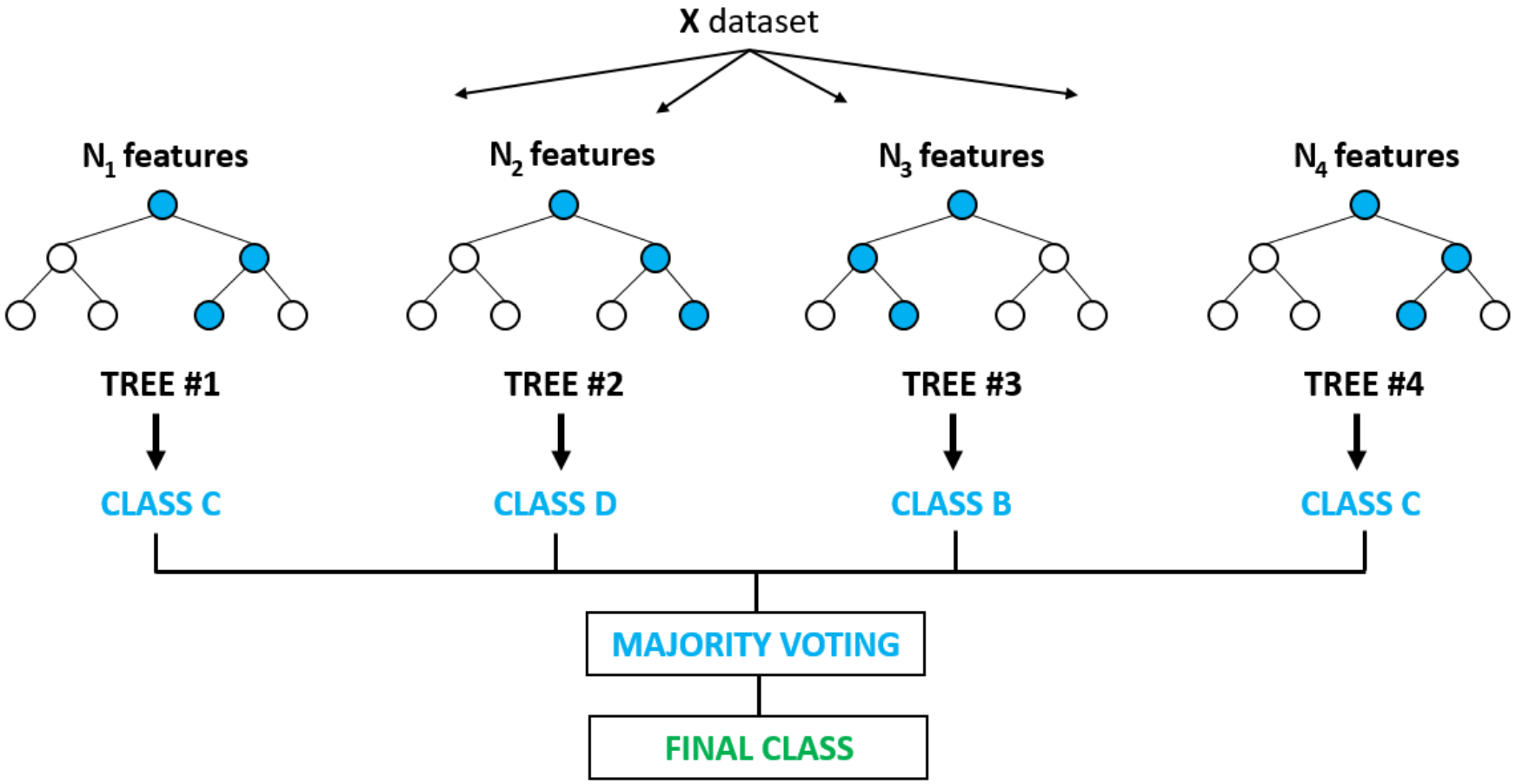
Bagging


D
Original
Training data
Step 1:
Create Multiple
Data Sets
D1
D2
Dt-1
Dt
Step 2:
Build Multiple
Classifiers
C1
C2
Ct-1
Ct
Step 3:
Combine
Classifiers
C*

How it works ?

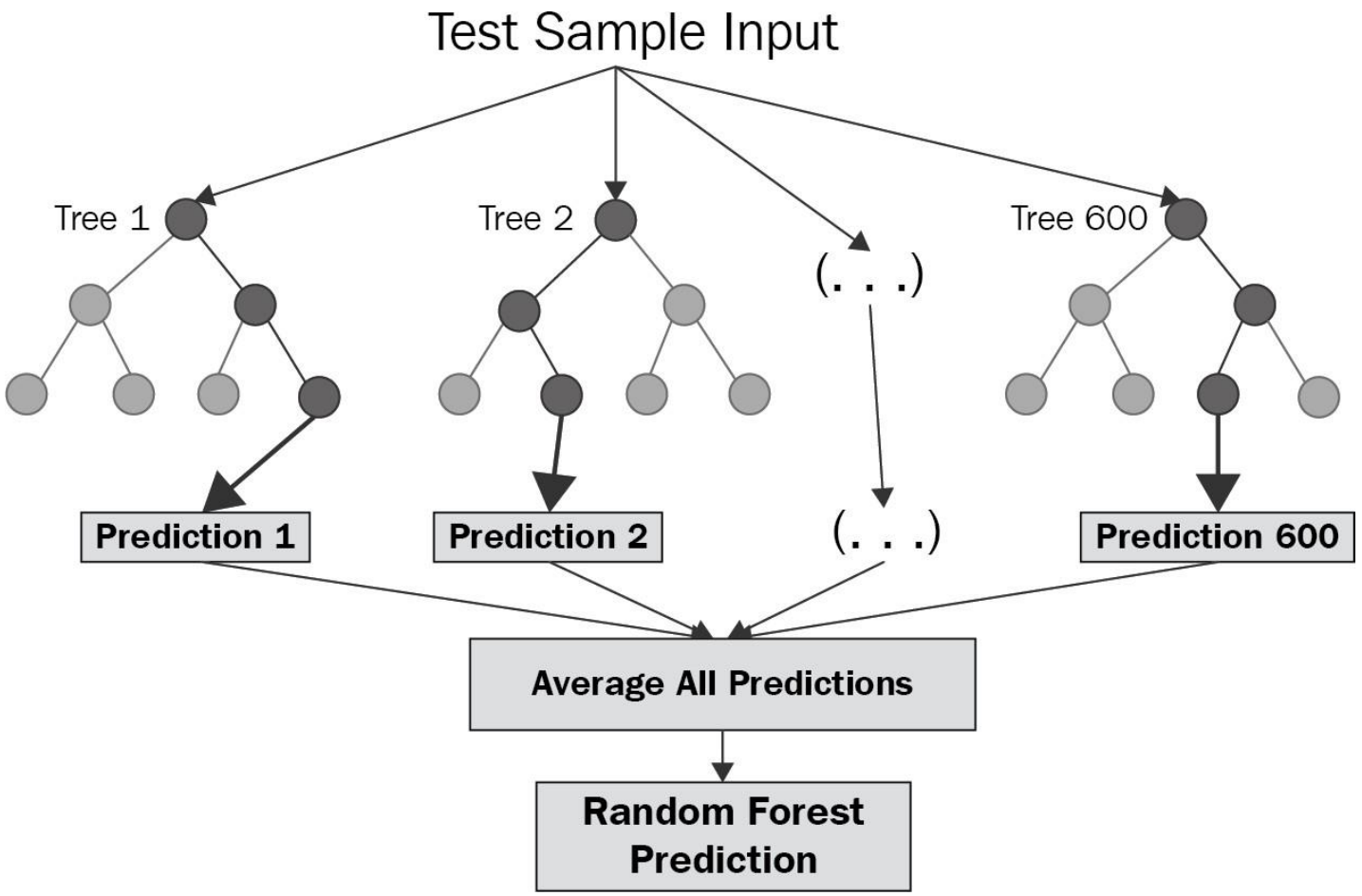
- Pick N random records from the dataset.
- Build a decision tree based on these N records.
- Choose the number of trees you want in your algorithm and repeat steps 1 and 2.
- In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.
-

Majority Voting



Source: medium.com

Regressor Output



use **Bagging with K-Nearest Neighbours (KNN) as the classifier.**

Sample	Feature X1	Feature X2	Feature X3	Target Y
1	2.0	3.1	1.5	0
2	3.2	4.3	2.1	1
3	1.5	2.8	1.2	0
4	4.1	3.9	2.5	1
5	5.3	6.7	3.2	1
6	3.8	5.1	2.8	1
7	2.7	3.5	1.8	0
8	4.5	4.9	2.7	1
9	1.9	3.0	1.4	0
10	5.1	6.2	3.0	1

For a new test sample **X1=4.0,X2=4.5,X3=2.2** each KNN classifier makes a predections as

Bootstrap Sample 1

Sample	Feature X1	Feature X2	Feature X3	Target Y
1	2.0	3.1	1.5	0
2	3.2	4.3	2.1	1
5	5.3	6.7	3.2	1
6	3.8	5.1	2.8	1
7	2.7	3.5	1.8	0

Bootstrap Sample 2

Sample	Feature X1	Feature X2	Feature X3	Target Y
3	1.5	2.8	1.2	0
4	4.1	3.9	2.5	1
5	5.3	6.7	3.2	1
8	4.5	4.9	2.7	1
10	5.1	6.2	3.0	1

Bootstrap Sample 3

Sample	Feature X1	Feature X2	Feature X3	Target Y
1	2.0	3.1	1.5	0
3	1.5	2.8	1.2	0
6	3.8	5.1	2.8	1
7	2.7	3.5	1.8	0
9	1.9	3.0	1.4	0

Train KNN Classifiers on Each Bootstrap Sample

Each bootstrap sample is used to train a **K-Nearest Neighbors (KNN) classifier**.

Classifier 1 (Trained on Sample 1)

Classifier 2 (Trained on Sample 2)

Classifier 3 (Trained on Sample 3)

For a new test sample **X1=4.0,X2=4.5,X3=2.2**, each KNN classifier makes a prediction:

Classifier 1 predicts: ?

Classifier 2 predicts: ?

Classifier 3 predicts: ?

For classification – the majority takes the value of prediction

For regression – the average of all the predictions are considered

Boosting

- Boosting is an ensemble modelling technique that attempts to build a strong classifier from the number of weak classifiers.

- It is done by building a model by using the weak models in the set of data

- First, the model is built from the training data
- -Here some of the instances are correctly classified, where as some are incorrectly classified.

- The incorrectly classified instances are collected as second model and the model is built which tries to correct the errors present in the first model.

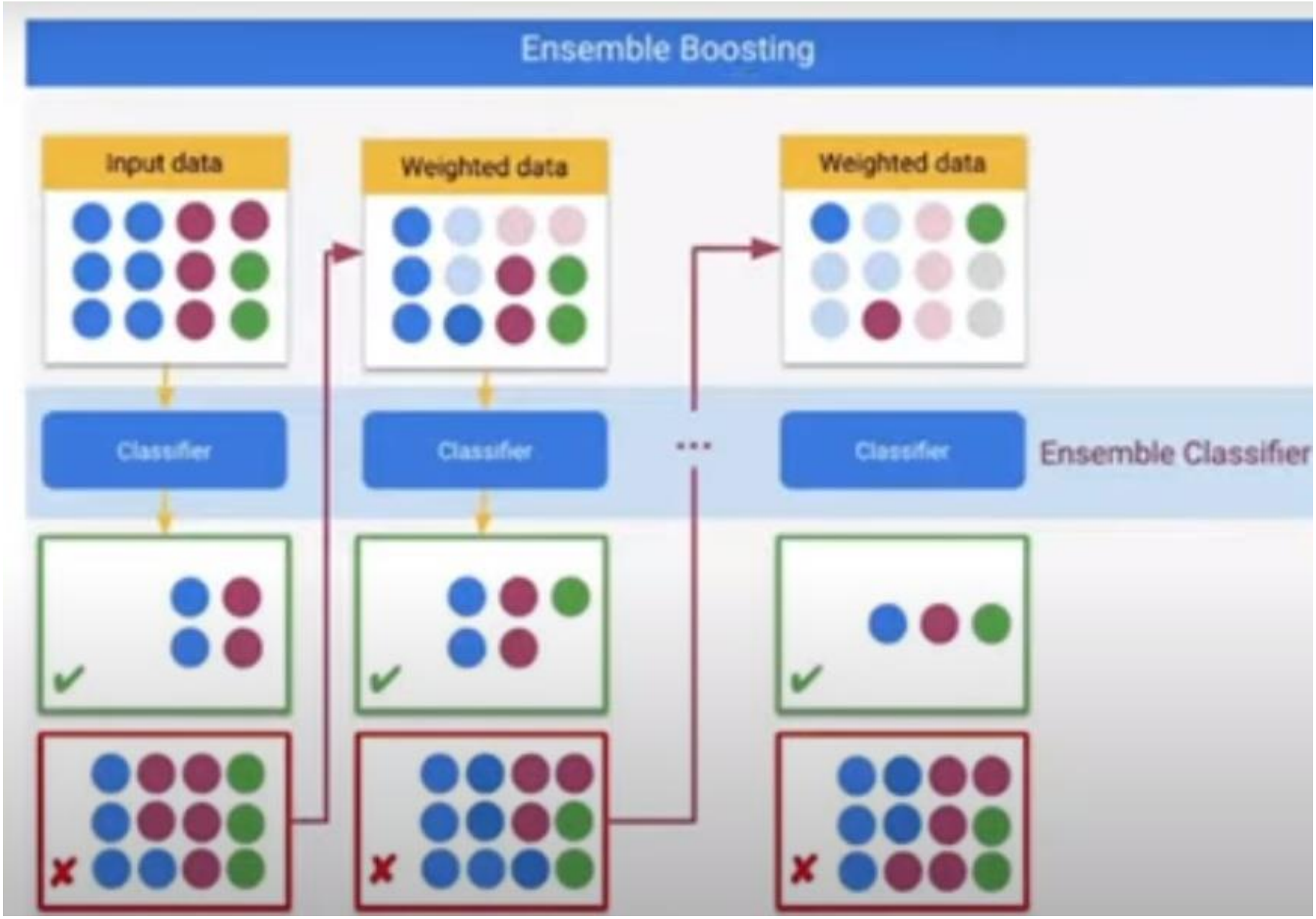
- This process is repeated untill all the instances are correctly classified or maximum models are added.

Boosting

- Boosting algorithms are a set of the low accurate classifier to create a highly accurate classifier.
- Low accuracy classifier (or weak classifier) offers the accuracy better than the flipping of a coin.
- This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added.
- Highly accurate classifier(or strong classifier) offer error rate close to 0. Boosting algorithm can track the model who failed the accurate prediction.
- Boosting algorithms are less affected by the overfitting problem.

- In simple terms

- Models are trained sequentially, where each new model focuses on **correcting the mistakes** of the previous one.More weight is given to misclassified data points.The final prediction is a weighted combination of all models.


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

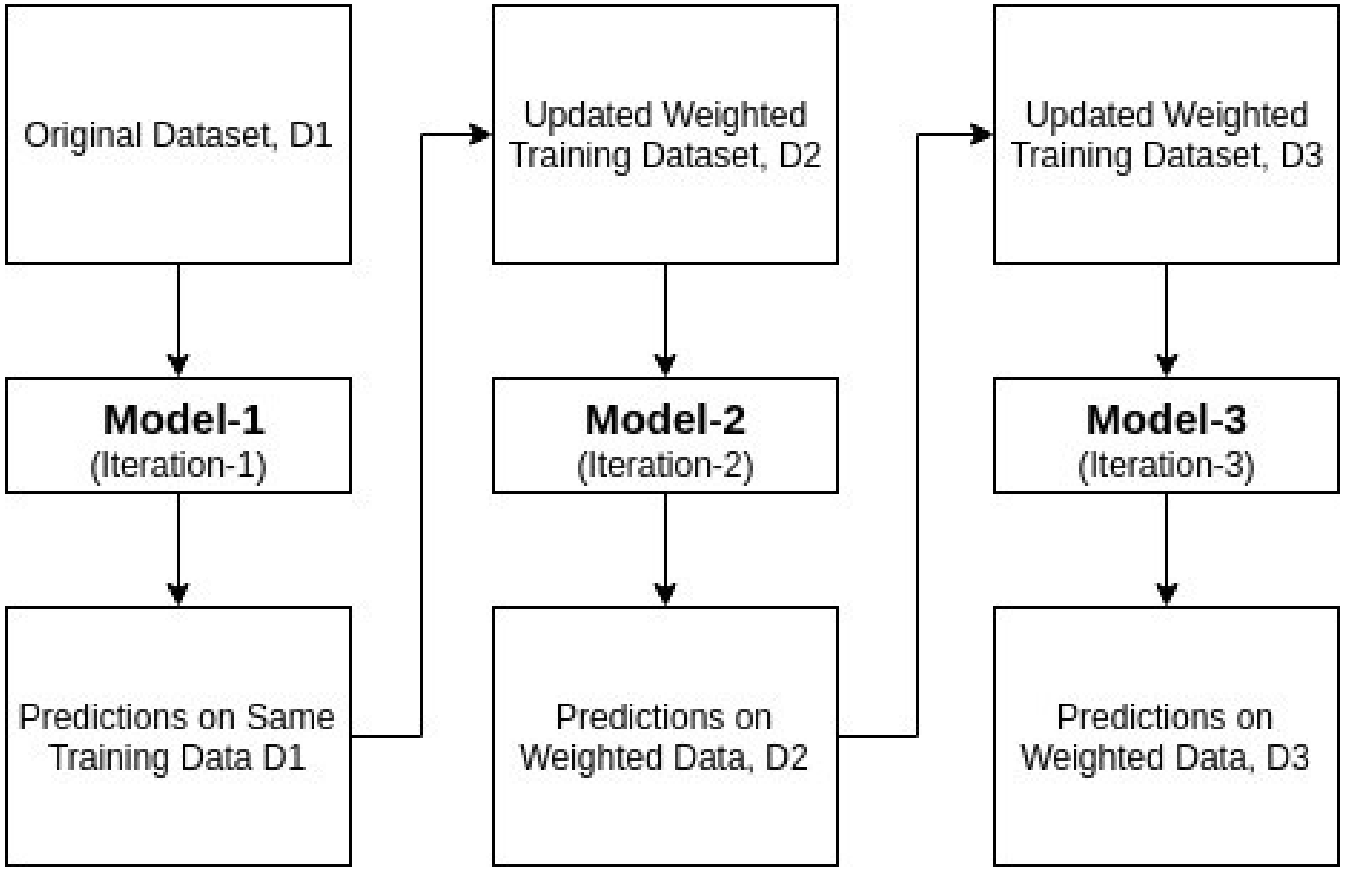
Boosting Models

- Models that are typically used in Boosting technique are:
  - XGBoost (Extreme Gradient Boosting)
  - GBM (Gradient Boosting Machine)
  - ADABoost (Adaptive Boosting)

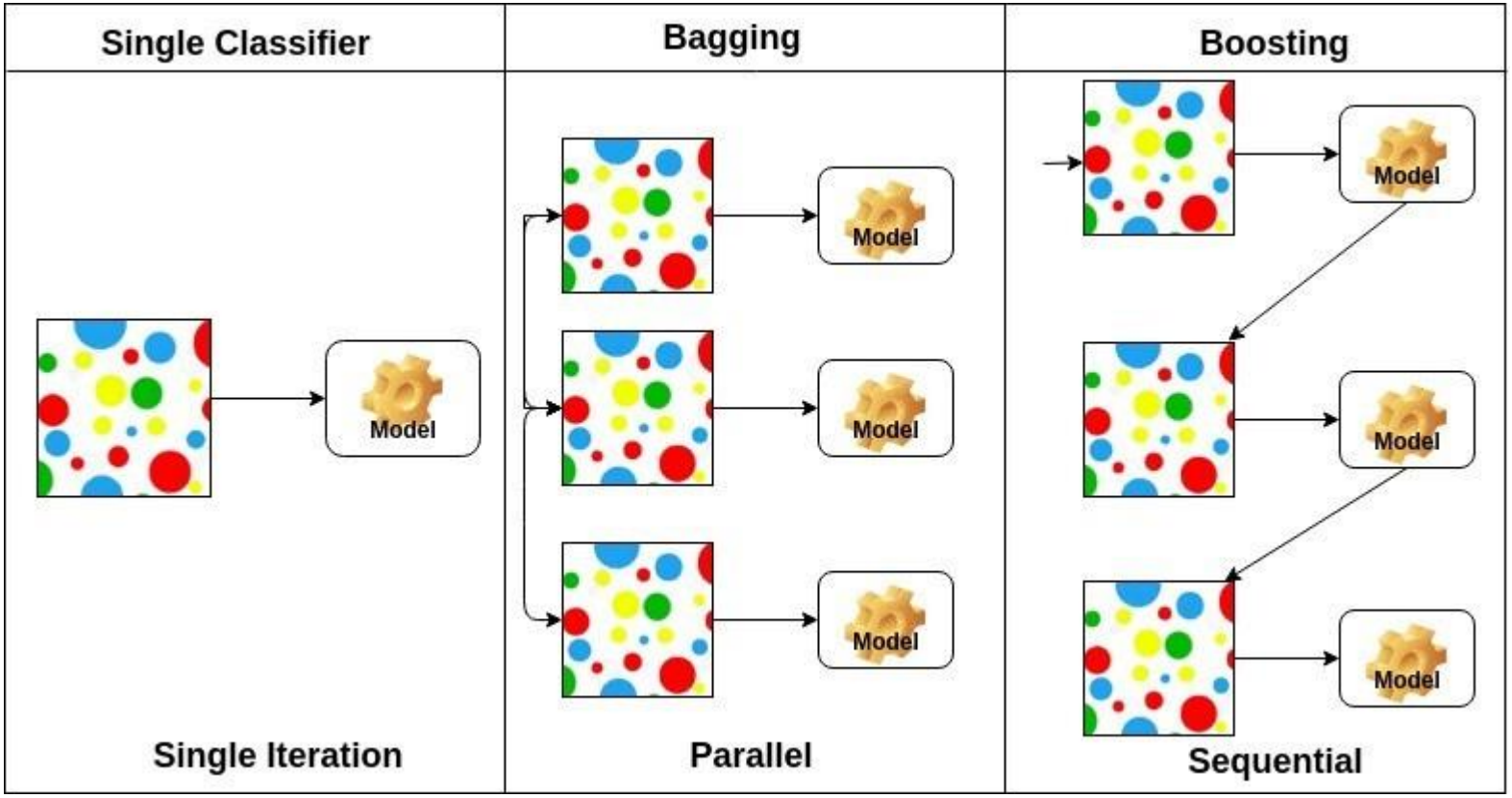
Adaboost Summary

- Initially, Adaboost selects a training subset randomly.
- It iteratively trains the AdaBoost machine learning model by selecting the training set based on the accurate prediction of the last training.
- It assigns the higher weight to wrong classified observations so that in the next iteration these observations will get the high probability for classification.
- Also, It assigns the weight to the trained classifier in each iteration according to the accuracy of the classifier. The more accurate classifier will get high weight.
- This process iterate until the complete training data fits without any error or until reached to the specified maximum number of estimators.
- To classify, perform a "vote" across all of the learning algorithms you built.

How Adaboost Works?



Comparison


Single Classifier
Bagging
Boosting
Model
Model
Model
Model
Model
Model
Model
Single Iteration
Parallel
Sequential

Stacking

- Stacked Generalization or “Stacking” for short is an ensemble machine learning algorithm.
- It involves combining the predictions from multiple machine learning models on the same dataset, like bagging and boosting.
- Stacking addresses the question:
  - Given multiple machine learning models that are skillful on a problem, but in different ways, how do you choose which model to use (trust)?

Stacking

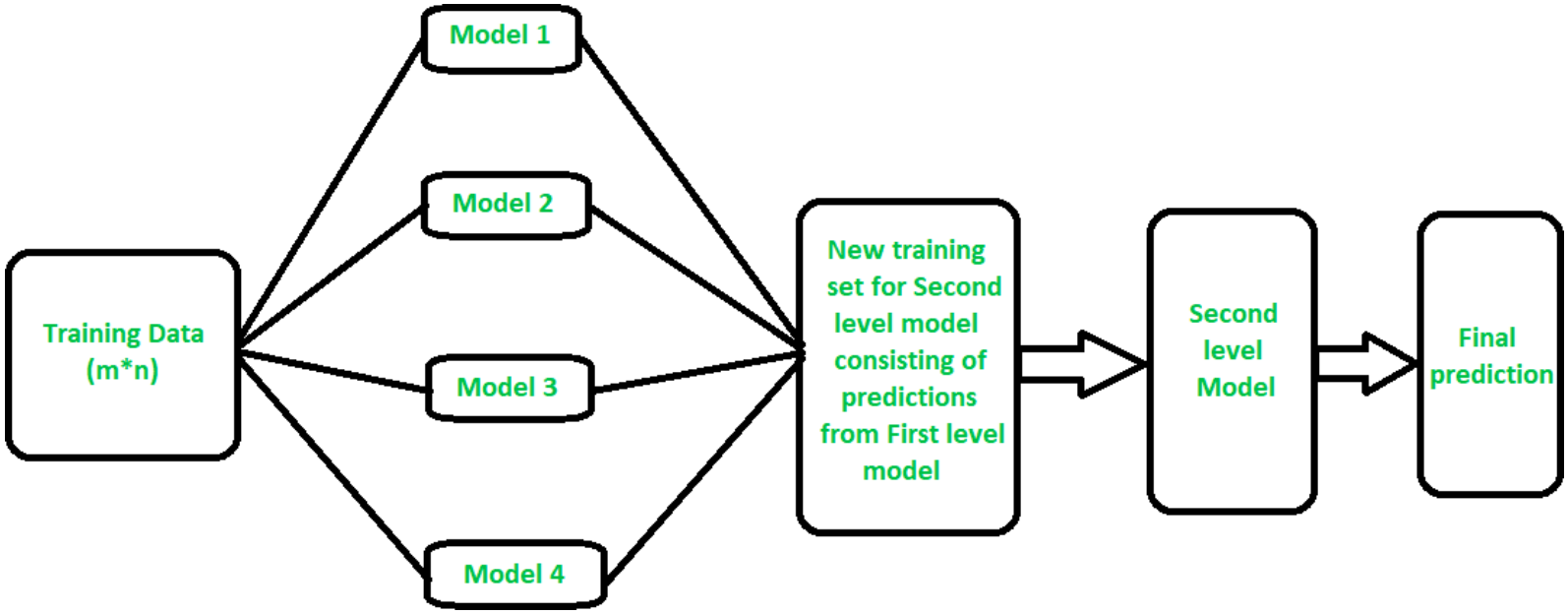
- The approach to this question is to use another machine learning model that learns when to use or trust each model in the ensemble.
  - Unlike bagging, in stacking, the models are typically different (e.g. not all decision trees) and fit on the same dataset (e.g. instead of samples of the training dataset).
  - Unlike boosting, in stacking, a single model is used to learn how to best combine the predictions from the contributing models (e.g. instead of a sequence of models that correct the predictions of prior models).

Stacking

- The architecture of a stacking model involves two or more base models, often referred to as level-0 models, and a meta-model that combines the predictions of the base models, referred to as a level-1 model.
  - Level-0 Models (Base-Models): Models fit on the training data and whose predictions are compiled.
  - Level-1 Model (Meta-Model): Model that learns how to best combine the predictions of the base models.

- In short
- Multiple base models (which can be different algorithms) are trained on the same dataset.Their predictions are fed into a **meta-model** (or second-level model), which learns to **combine their outputs** optimally.

Stacking


Model 1
Model 2
Model 3
Model 4
Training Data
(m*n)
New training set for Second level model consisting of predictions from First level model
Second level Model
Final prediction

Stacking

- The meta-model is trained on the predictions made by base models on out-of-sample data.
- That is, data not used to train the base models is fed to the base models, predictions are made, and these predictions, along with the expected outputs, provide the input and output pairs of the training dataset used to fit the meta-model.
- The outputs from the base models used as input to the meta-model may be real value in the case of regression, and probability values, probability like values, or class labels in the case of classification.

Useful Journal Article

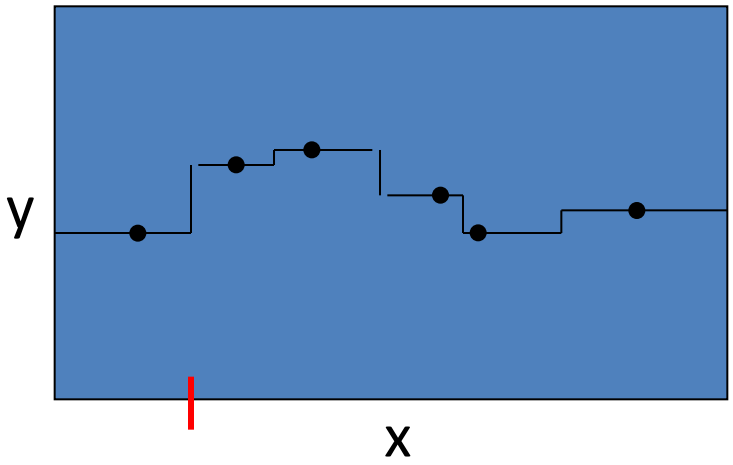
- https://www.mdpi.com/2076-3417/13/5/3210

A spectrum of models

Very local models

- e.g. Nearest neighbors

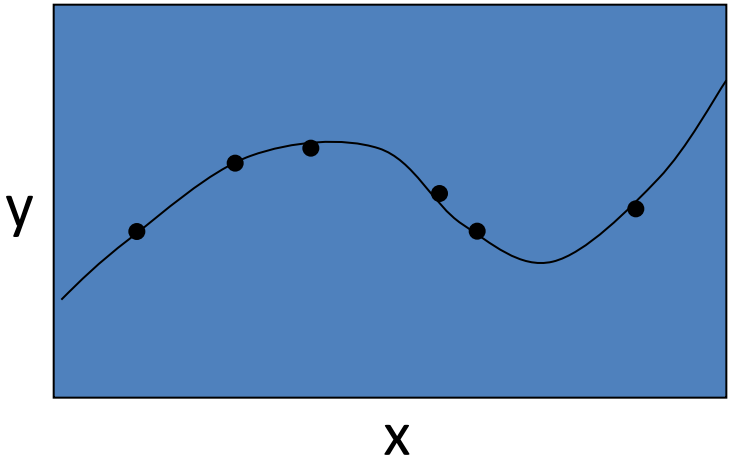
- Very fast to fit
  - Just store training cases
- Local smoothing obviously improves things



Fully global models

- e. g. Polynomial

- May be slow to fit
  - Each parameter depends on all the data

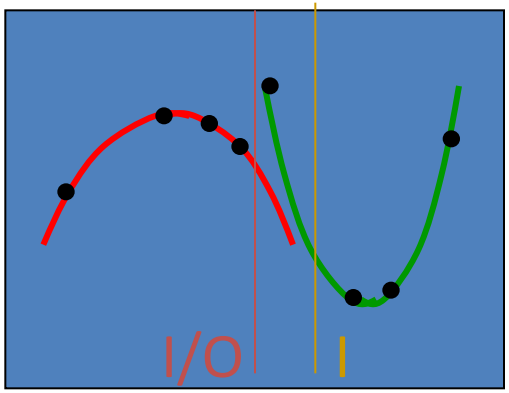


Multiple local models

- Instead of using a single global model or lots of very local models, use several models of intermediate complexity.
  - Good if the dataset contains several different regimes which have different relationships between input and output.
  - But how do we partition the dataset into subsets for each expert?

Partitioning based on input alone versus partitioning based on input-output relationship

- We need to cluster the training cases into subsets, one for each local model.
  - The aim of the clustering is NOT to find clusters of similar input vectors.
  - We want each cluster to have a relationship between input and output that can be well-modeled by one local model



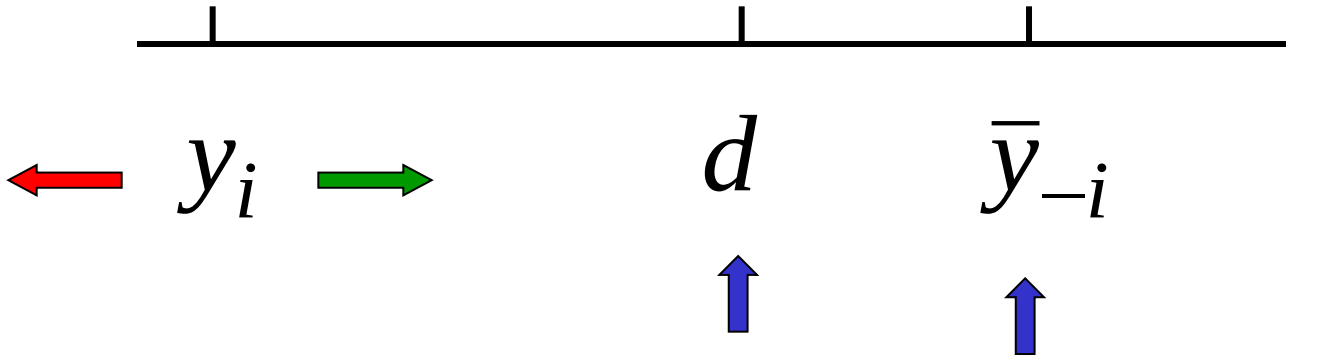
which partition is best:

I=input alone or I/O=input→output mapping?

Mixtures of Experts

- Can we do better that just averaging predictors in a way that does not depend on the particular training case?
  - Maybe we can look at the input data for a particular case to help us decide which model to rely on.
    - This may allow particular models to specialize in a subset of the training cases. They do not learn on cases for which they are not picked. So they can ignore stuff they are not good at modeling.
- The key idea is to make each expert focus on predicting the right answer for the cases where it is already doing better than the other experts.
  - This causes specialization.
  - If we always average all the predictors, each model is trying to compensate for the combined error made by all the other models.

A picture of why averaging is bad



Do we really want to move the output of predictor i away from the target value?

Making an error function that encourages specialization instead of cooperation

- If we want to encourage cooperation, we compare the average of all the predictors with the target and train to reduce the discrepancy.
  - This can overfit badly. It makes the model much more powerful than training each predictor separately.

Average of all the predictors

$$E = (d - < y_i >_i)^2$$

- If we want to encourage specialization we compare each predictor separately with the target and train to reduce the average of all these discrepancies.
  - **Its best to use a weighted average, where the weights, p, are the probabilities of picking that "expert" for the particular training case.**

$$E = < p_i (d - y_i)^2 >_i$$

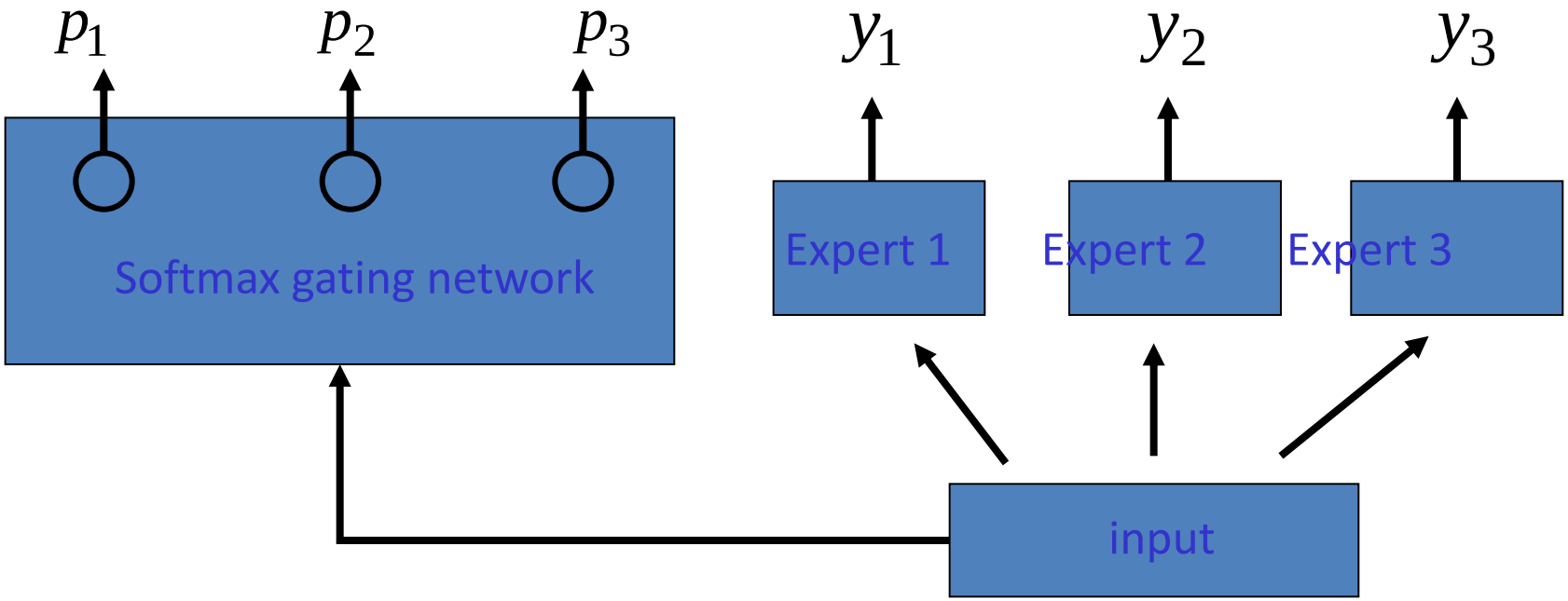
probability of picking expert i for this case

The mixture of experts architecture

Combined predictor: $y = \sum_i p_i \, y_i$

Simple error function for training: $E = \sum_i p_i (d - y_i)^2$

(There is a better error function)



The derivatives of the simple cost function

- If we differentiate w.r.t. the outputs of the experts we get a signal for training each expert.

$$E = \sum_i p_i (d - y_i)^2$$

$$\frac{\partial E}{\partial y_i} = p_i (d - y_i)$$

- If we differentiate w.r.t. the outputs of the gating network we get a signal for training the gating net.
  - **We want to raise p for all experts that give less than the average squared error of all the experts (weighted by p)**

$$\frac{\partial E}{\partial x_i} = p_i [(d - y_i)^2 - E]$$

$$where \quad p_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

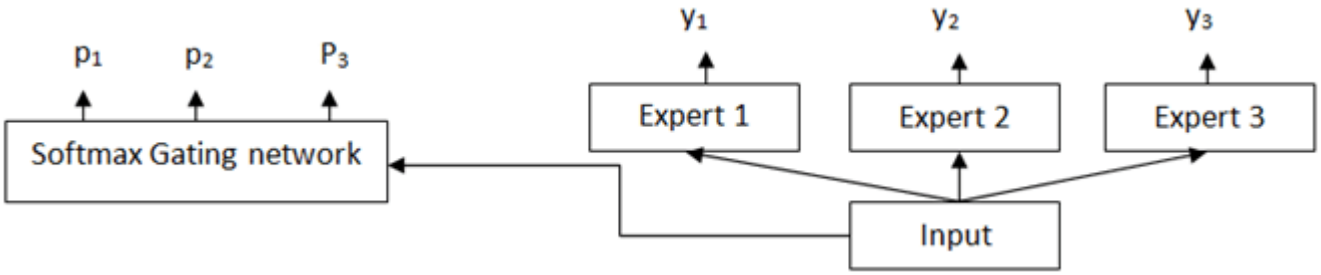
Another view of mixtures of experts

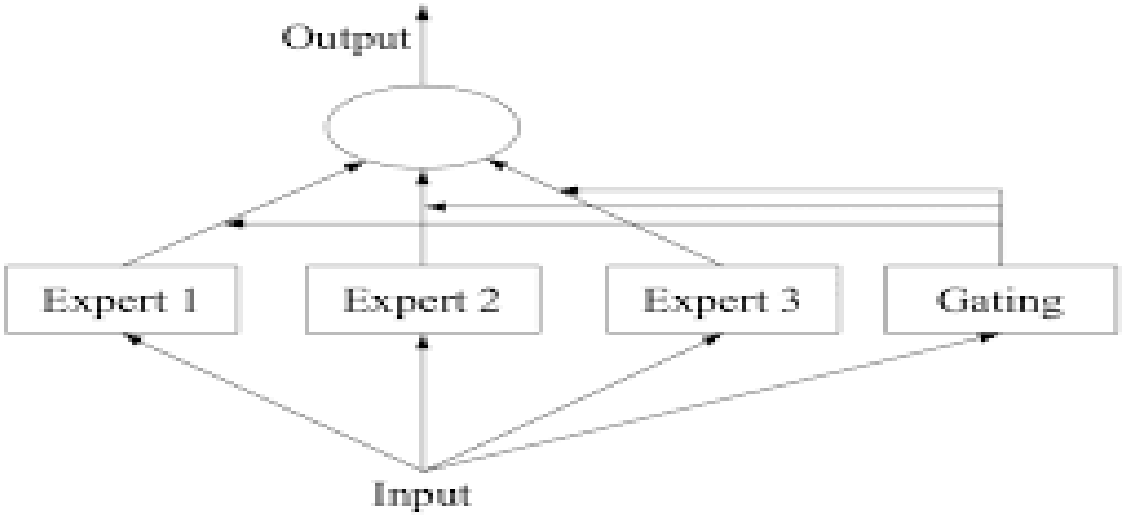
- One way to combine the outputs of the experts is to take a weighted average, using the gating network to decide how much weight to place on each expert.

What is gating mechanism in ensemble learning?

In ensemble learning, "gating" refers to a mechanism where a separate network, called a "gating network," determines which individual model within an ensemble should be most trusted for a given input, essentially acting as a selector that dynamically weights the predictions of different models based on the specific data point being evaluated; this is most commonly seen in "Mixture of Experts" (MoE) architectures where each model specializes in a particular region of the input space, and the gating network decides which expert to consult for a given input

Example - MOE

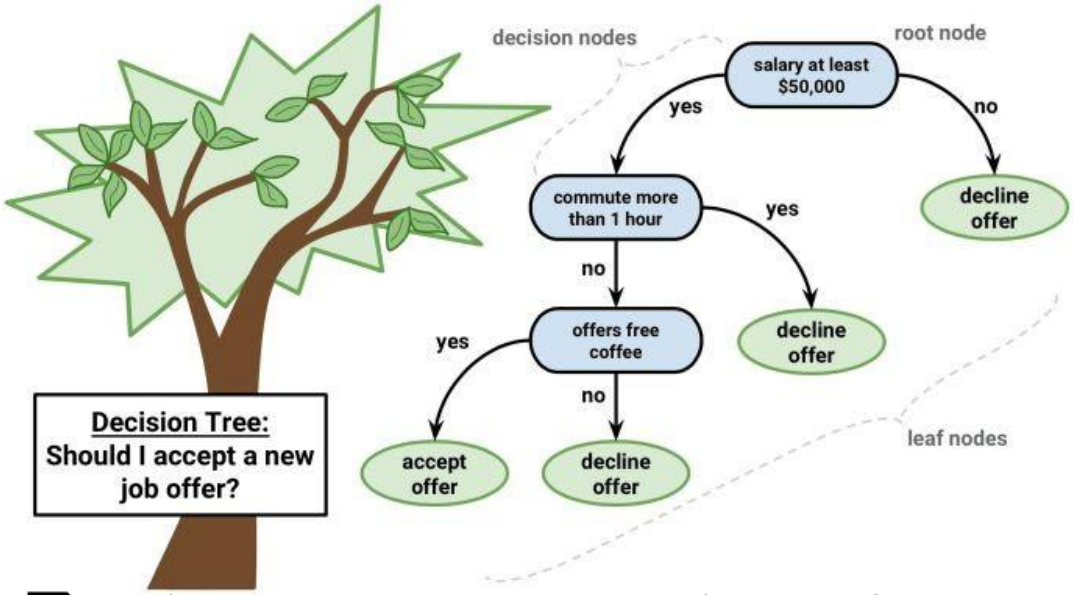

p1
p2
P3
Softmax Gating network
y1
y2
y3
Expert 1
Expert 2
Expert 3
Input


Output
Expert 1
Expert 2
Expert 3
Gating
Input

Random Forest Classifier

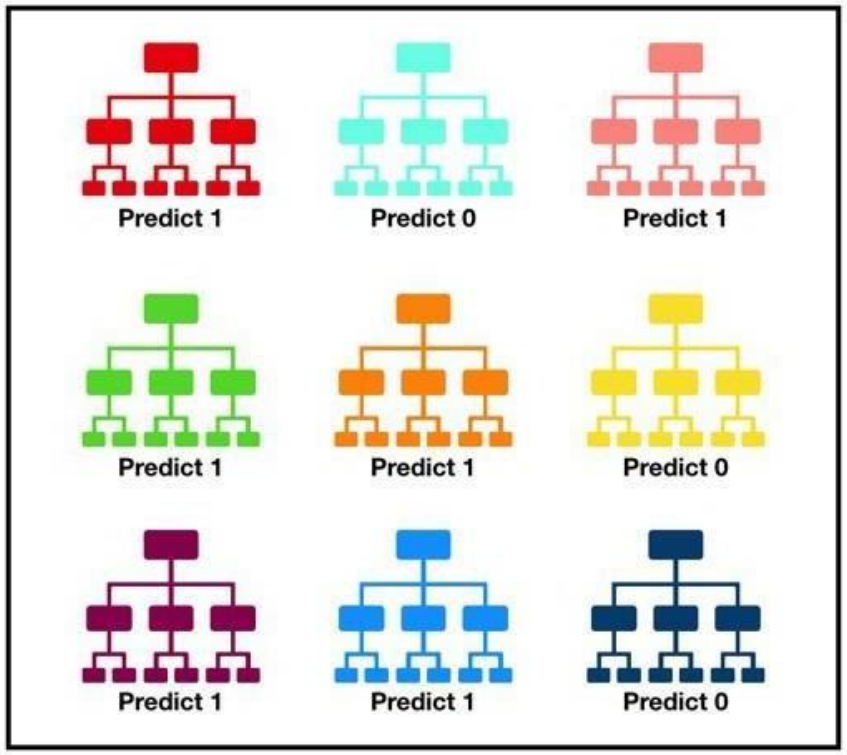
- Random Forest grows multiple decision trees which are merged together for a more accurate prediction.
- The logic behind the Random Forest model is that multiple uncorrelated models (the individual decision trees) perform much better as a group than they do alone. When using Random Forest for classification, each tree gives a classification or a “vote.” The forest chooses the classification with the majority of the “votes.” When using Random Forest for regression, the forest picks the average of the outputs of all trees.

Decision Trees



- A decision tree is a predictor that predicts the label associated with a sample $\boldsymbol{x}$ by traveling along a tree from the root to a leaf
  - We'll focus on binary trees
- At each internal node we made a decision based on features of $\boldsymbol{x}$
  - It corresponds to splitting the input space
  - Simplest idea, split on the basis of a threshold on one of the features, i.e., $x_i < \theta$ or $x_i \geq \theta$
- Each leaf is associated to a label

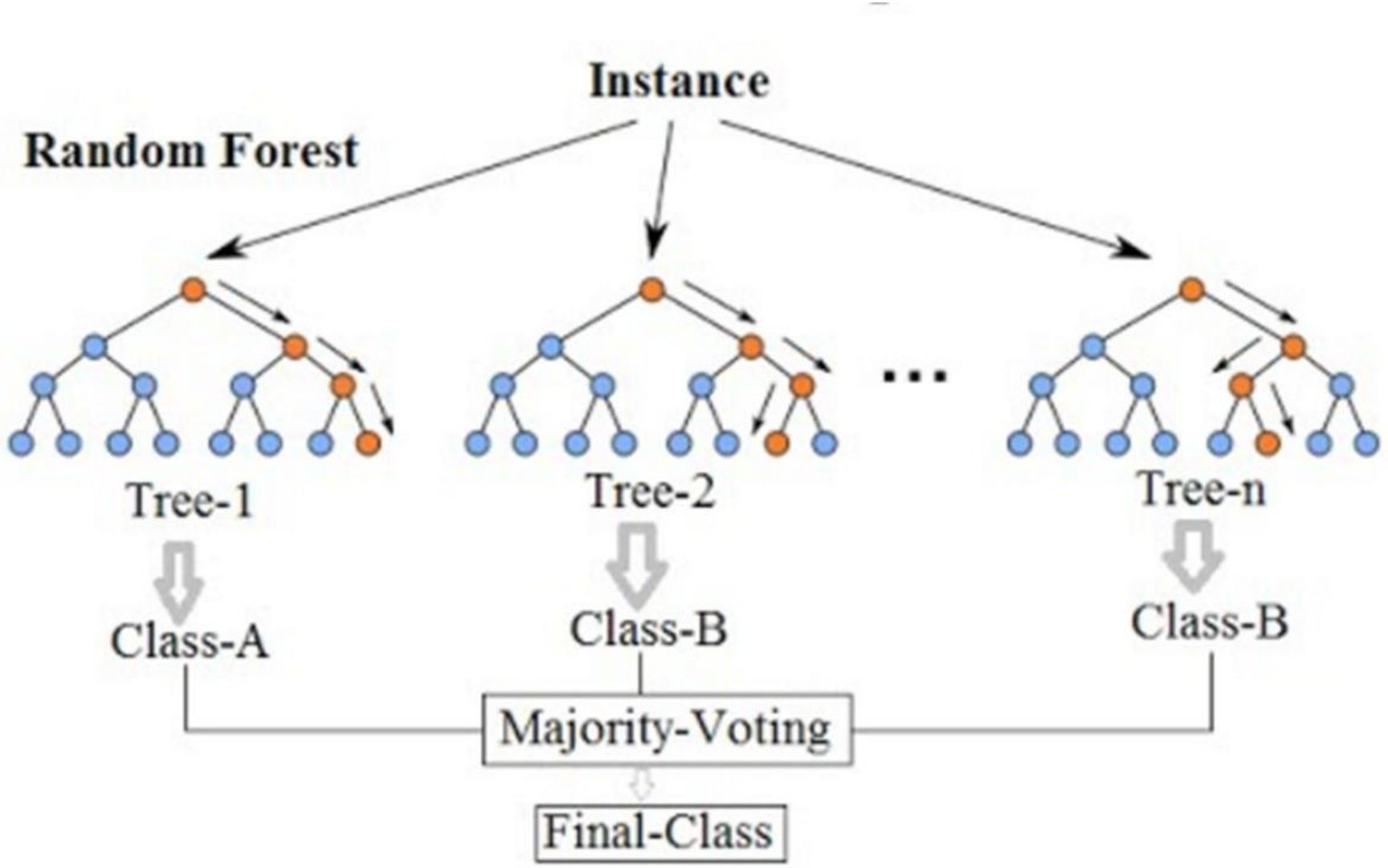
Random Forests (RF)



Tally: Six 1s and Three 0s
Prediction: 1

- Introduced by Leo Breiman in 2001
- Instead of using a single large tree construct an ensemble of simpler trees
- A Random Forest (RF) is a classifier consisting of a collection of decision trees
- The prediction is obtained by a majority voting over the prediction of the single trees

Random Forest: Example



Random Forest Classifier

A classification problem where we predict whether a person will buy a product based on three attributes:
Age (Young, Middle, Old)
Salary (Low, Medium, High)
Online Ads Clicked (Few, Moderate, Many)
Target Variable (Buy Product?): Yes or No

ID	Age	Salary	Clicks	Buy?
1	Young	Low	Few	No
2	Young	High	Many	Yes
3	Middle	Medium	Moderate	Yes
4	Old	Low	Few	No
5	Old	Medium	Many	Yes
6	Middle	High	Few	No
7	Young	Medium	Moderate	Yes
8	Middle	Low	Few	No
9	Old	High	Many	Yes
10	Young	Low	Many	Yes

Step 2: Bootstrap Sampling (Bagging)
We'll create 3 different bootstrap samples (randomly selecting instances with replacement):
Tree 1:
Sample: {1, 3, 4, 5, 7, 9}
Tree 2:
Sample: {2, 4, 6, 8, 9, 10}
Tree 3:
Sample: {1, 3, 5, 6, 7, 10}

Step 3: Build Decision Trees (Manually Splitting by Gini Impurity or Entropy)

Step 4: Making Predictions
Each tree makes its prediction, and we use **majority voting**:

Final predicted output

ID	Tree 1	Tree 2	Tree 3	Final Prediction
1	No	Yes	No	No
2	Yes	Yes	Yes	Yes
3	Yes	No	Yes	Yes
4	No	No	No	No
5	Yes	Yes	Yes	Yes
6	No	No	No	No
7	Yes	Yes	Yes	Yes
8	No	No	No	No
9	Yes	Yes	Yes	Yes
10	Yes	Yes	Yes	Yes

Final Result

Our Random Forest classifier predicts "Buy = Yes" for instances {2, 3, 5, 7, 9, 10} and "Buy = No" for {1, 4, 6, 8}.