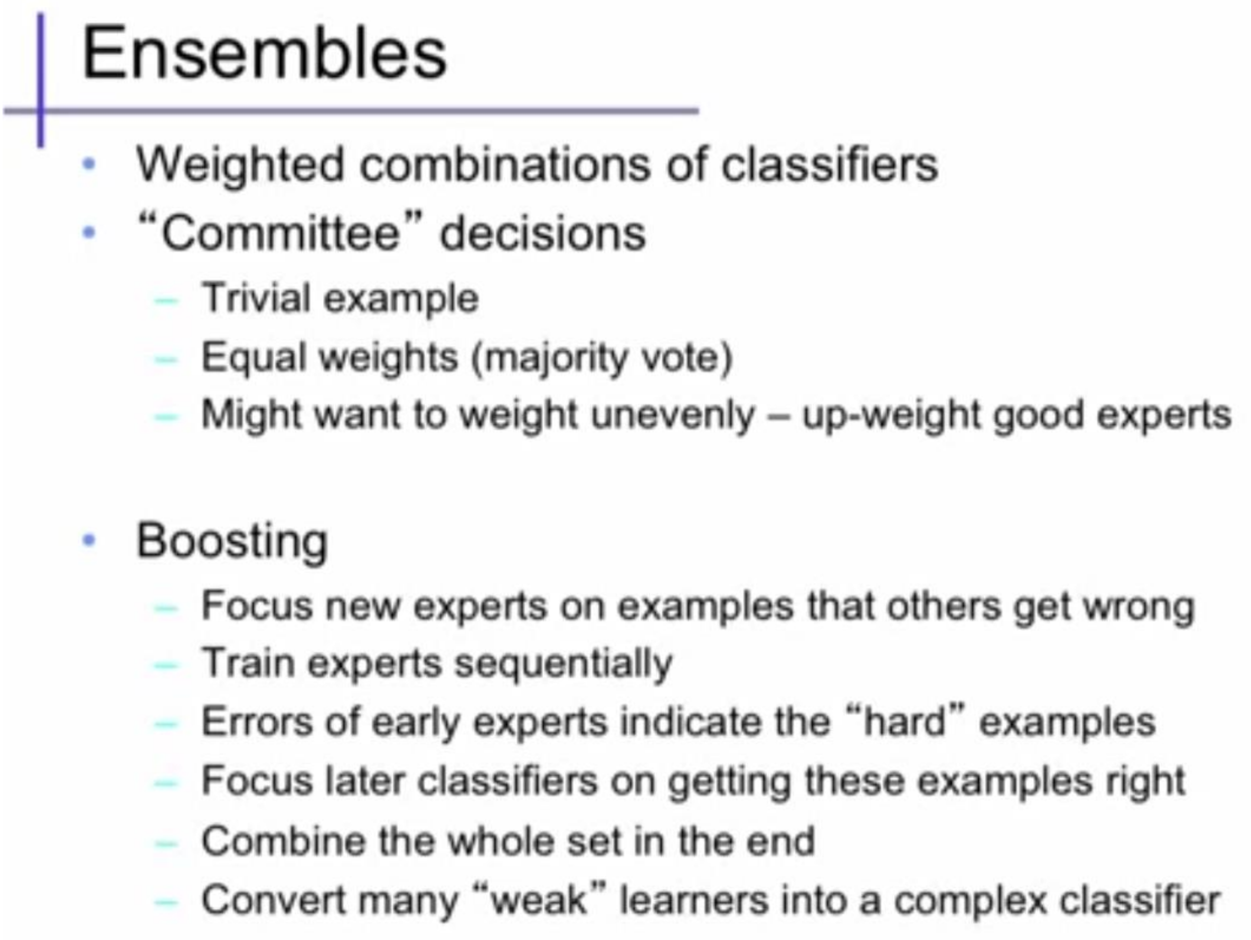


Ensembles

- Weighted combinations of classifiers
- “Committee” decisions
  - Trivial example
  - Equal weights (majority vote)
  - Might want to weight unevenly – up-weight good experts
- Boosting
  - Focus new experts on examples that others get wrong
  - Train experts sequentially
  - Errors of early experts indicate the “hard” examples
  - Focus later classifiers on getting these examples right
  - Combine the whole set in the end
  - Convert many “weak” learners into a complex classifier

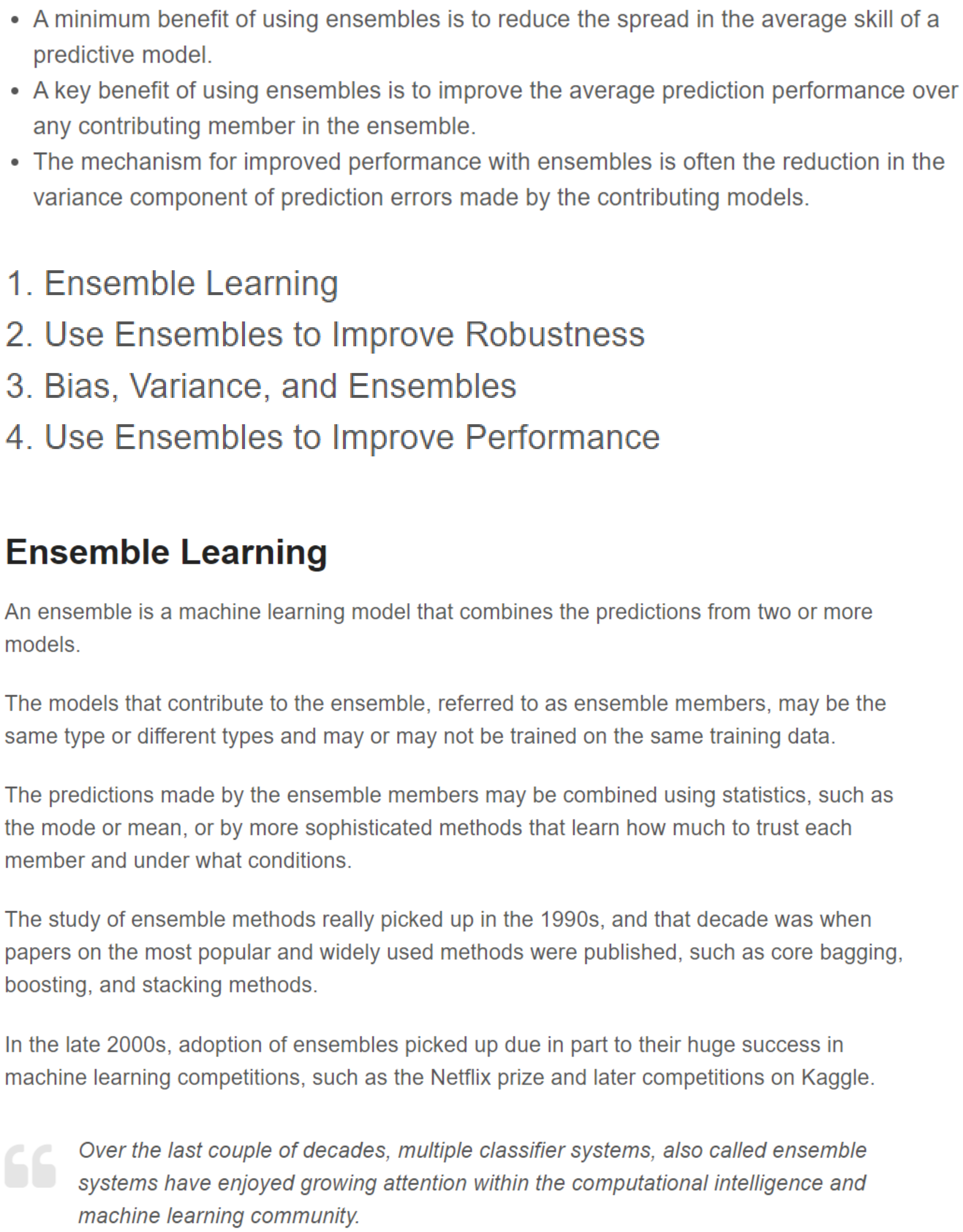
What are the Benefits of Ensemble Methods for Machine Learning?

Ensembles are predictive models that combine predictions from two or more other models.

Ensemble learning methods are popular and the go-to technique when the best performance on a predictive modeling project is the most important outcome.

Nevertheless, they are not always the most appropriate technique to use and beginners the field of applied machine learning have the expectation that ensembles or a specific ensemble method are always the best method to use.

Ensembles offer two specific benefits on a predictive modeling project, and it is important to know what these benefits are and how to measure them to ensure that using an ensemble is the right decision on your project.



- A minimum benefit of using ensembles is to reduce the spread in the average skill of a predictive model.
- A key benefit of using ensembles is to improve the average prediction performance over any contributing member in the ensemble.
- The mechanism for improved performance with ensembles is often the reduction in the variance component of prediction errors made by the contributing models.

Ensemble Learning

An ensemble is a machine learning model that combines the predictions from two or more models.

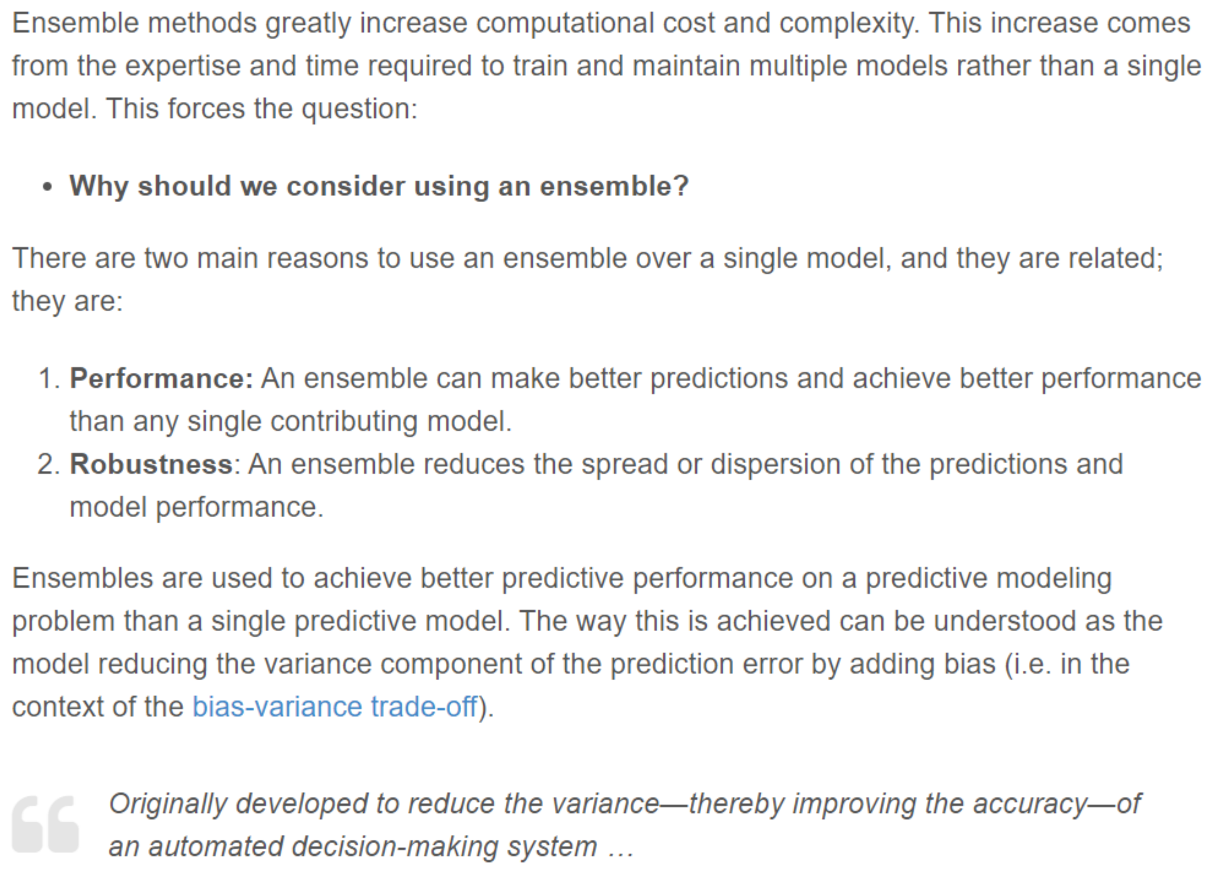
The models that contribute to the ensemble, referred to as ensemble members, may be the same type or different types and may or may not be trained on the same training data.

The predictions made by the ensemble members may be combined using statistics, such as the mode or mean, or by more sophisticated methods that learn how much to trust each member and under what conditions.

The study of ensemble methods really picked up in the 1990s, and that decade was when papers on the most popular and widely used methods were published, such as core bagging, boosting, and stacking methods.

In the late 2000s, adoption of ensembles picked up due in part to their huge success in machine learning competitions, such as the Netflix prize and later competitions on Kaggle.

> *Over the last couple of decades, multiple classifier systems, also called ensemble systems have enjoyed growing attention within the computational intelligence and machine learning community.*



Ensemble methods greatly increase computational cost and complexity. This increase comes from the expertise and time required to train and maintain multiple models rather than a single model. This forces the question:

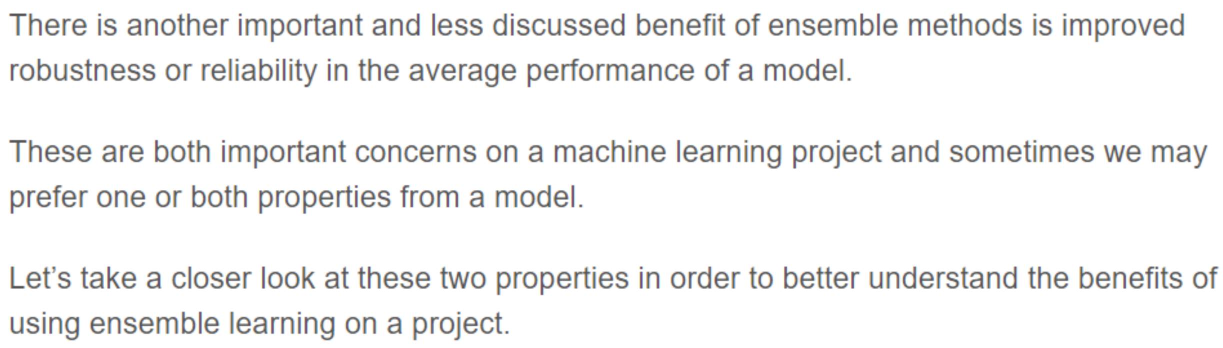
- **Why should we consider using an ensemble?**

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

1. **Performance:** An ensemble can make better predictions and achieve better performance than any single contributing model.
2. **Robustness**: An ensemble reduces the spread or dispersion of the predictions and model performance.

Ensembles are used to achieve better predictive performance on a predictive modeling problem than a single predictive model. The way this is achieved can be understood as the model reducing the variance component of the prediction error by adding bias (i.e. in the context of the bias-variance trade-off).

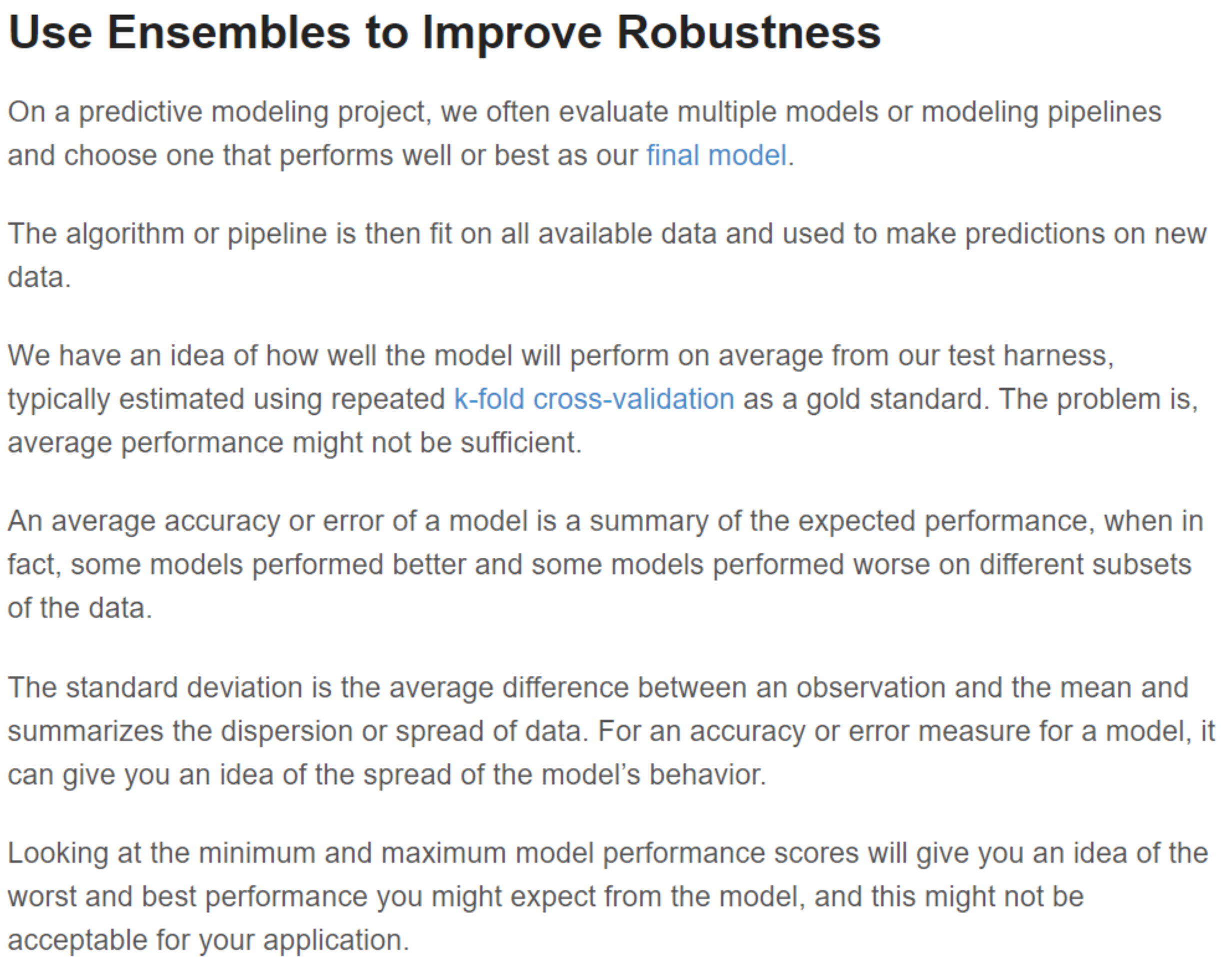
> *Originally developed to reduce the variance—thereby improving the accuracy—of an automated decision-making system …*



There is another important and less discussed benefit of ensemble methods is improved robustness or reliability in the average performance of a model.

These are both important concerns on a machine learning project and sometimes we may prefer one or both properties from a model.

Let’s take a closer look at these two properties in order to better understand the benefits of using ensemble learning on a project.



Use Ensembles to Improve Robustness

On a predictive modeling project, we often evaluate multiple models or modeling pipelines and choose one that performs well or best as our final model.

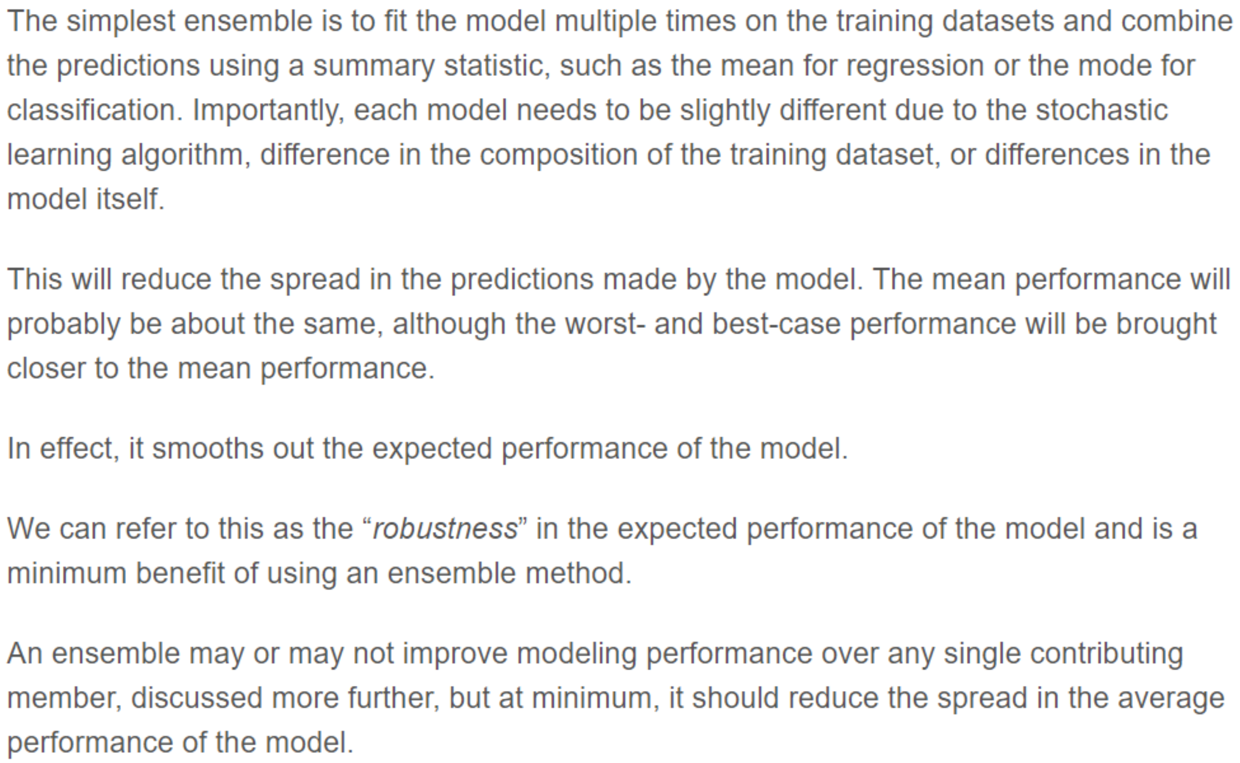
The algorithm or pipeline is then fit on all available data and used to make predictions on new data.

We have an idea of how well the model will perform on average from our test harness, typically estimated using repeated k-fold cross-validation as a gold standard. The problem is, average performance might not be sufficient.

An average accuracy or error of a model is a summary of the expected performance, when in fact, some models performed better and some models performed worse on different subsets of the data.

The standard deviation is the average difference between an observation and the mean and summarizes the dispersion or spread of data. For an accuracy or error measure for a model, it can give you an idea of the spread of the model’s behavior.

Looking at the minimum and maximum model performance scores will give you an idea of the worst and best performance you might expect from the model, and this might not be acceptable for your application.



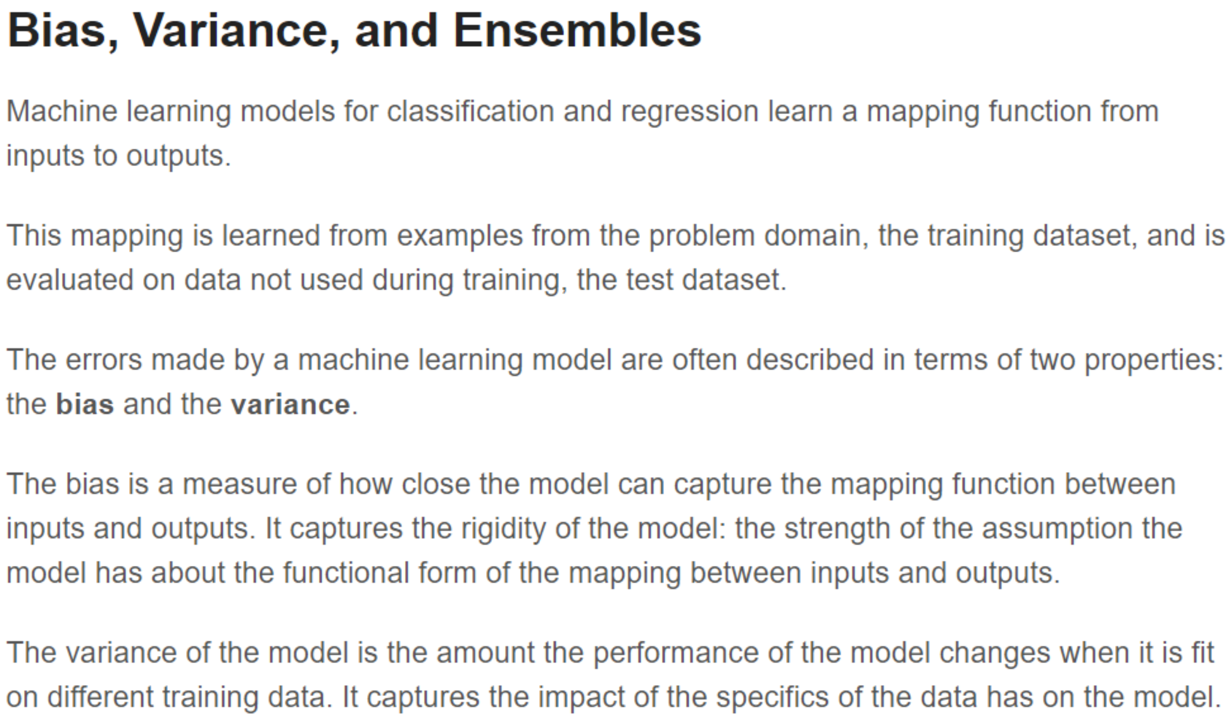
The simplest ensemble is to fit the model multiple times on the training datasets and combine the predictions using a summary statistic, such as the mean for regression or the mode for classification. Importantly, each model needs to be slightly different due to the stochastic learning algorithm, difference in the composition of the training dataset, or differences in the model itself.

This will reduce the spread in the predictions made by the model. The mean performance will probably be about the same, although the worst- and best-case performance will be brought closer to the mean performance.

In effect, it smooths out the expected performance of the model.

We can refer to this as the “*robustness*” in the expected performance of the model and is a minimum benefit of using an ensemble method.

An ensemble may or may not improve modeling performance over any single contributing member, discussed more further, but at minimum, it should reduce the spread in the average performance of the model.



Bias, Variance, and Ensembles

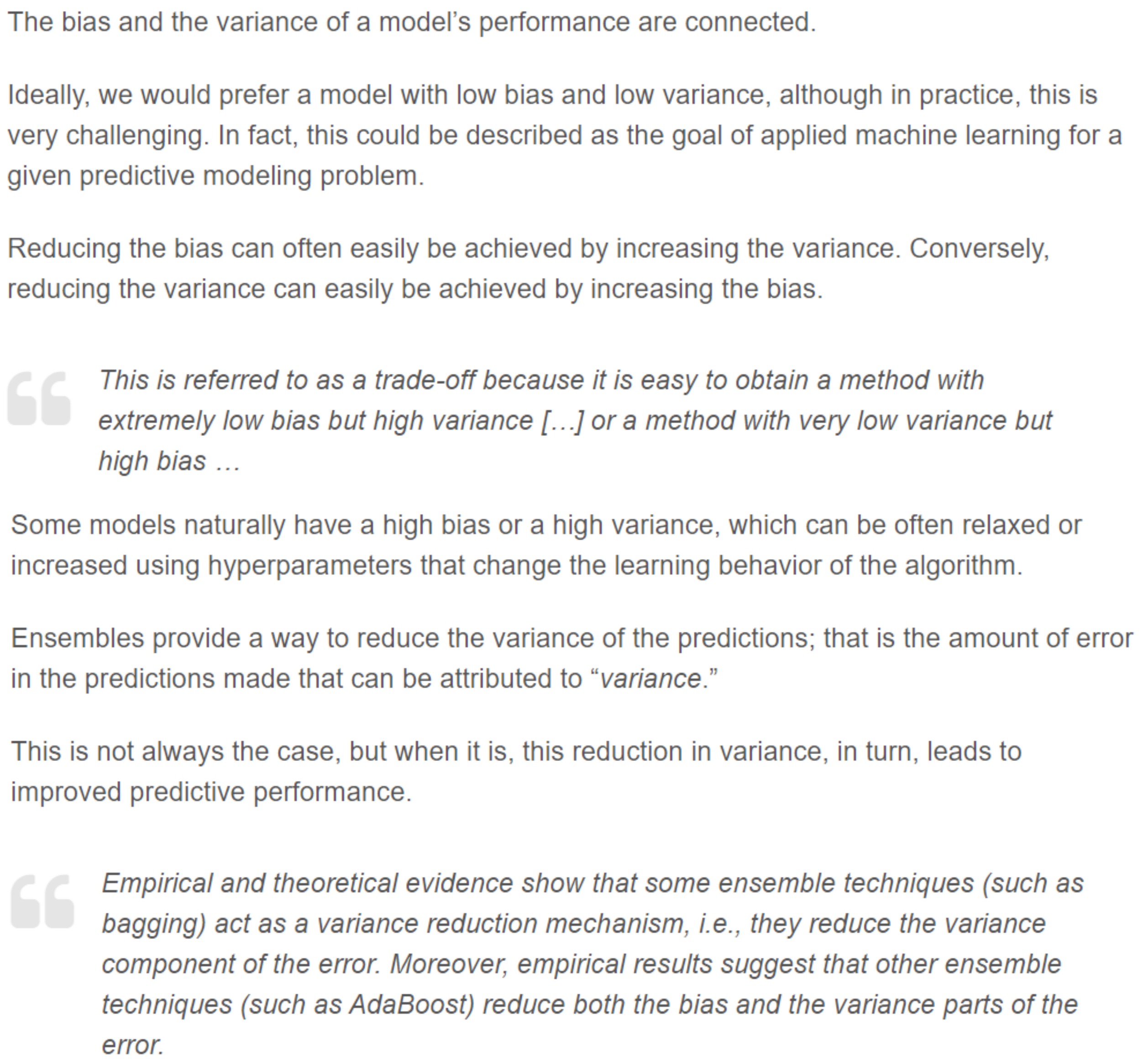
Machine learning models for classification and regression learn a mapping function from inputs to outputs.

This mapping is learned from examples from the problem domain, the training dataset, and is evaluated on data not used during training, the test dataset.

The errors made by a machine learning model are often described in terms of two properties: the **bias** and the **variance**.

The bias is a measure of how close the model can capture the mapping function between inputs and outputs. It captures the rigidity of the model: the strength of the assumption the model has about the functional form of the mapping between inputs and outputs.

The variance of the model is the amount the performance of the model changes when it is fit on different training data. It captures the impact of the specifics of the data has on the model.



The bias and the variance of a model's performance are connected.

Ideally, we would prefer a model with low bias and low variance, although in practice, this is very challenging. In fact, this could be described as the goal of applied machine learning for a given predictive modeling problem.

Reducing the bias can often easily be achieved by increasing the variance. Conversely, reducing the variance can easily be achieved by increasing the bias.

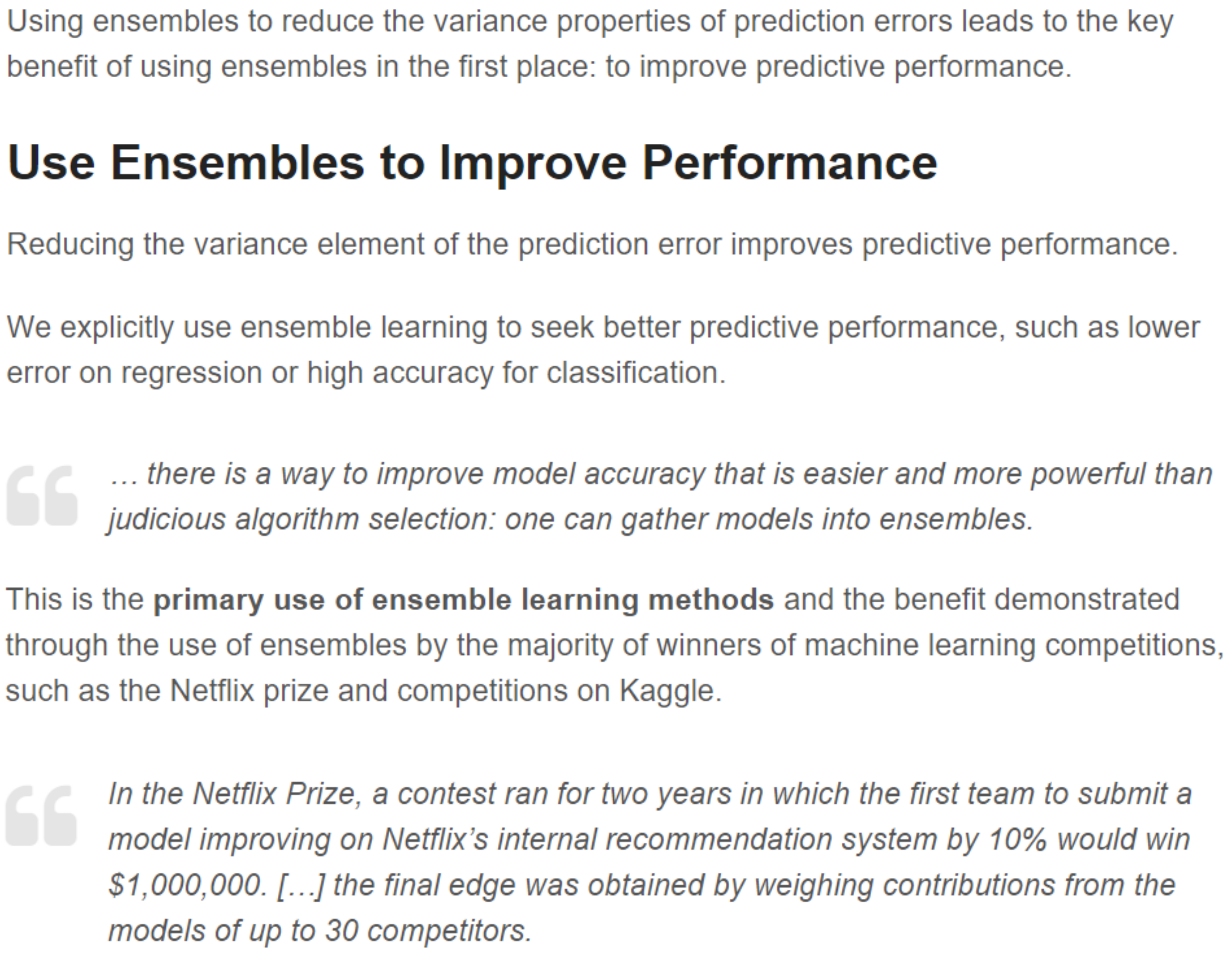
> *This is referred to as a trade-off because it is easy to obtain a method with extremely low bias but high variance […] or a method with very low variance but high bias …*

Some models naturally have a high bias or a high variance, which can be often relaxed or increased using hyperparameters that change the learning behavior of the algorithm.

Ensembles provide a way to reduce the variance of the predictions; that is the amount of error in the predictions made that can be attributed to "*variance*."

This is not always the case, but when it is, this reduction in variance, in turn, leads to improved predictive performance.

> *Empirical and theoretical evidence show that some ensemble techniques (such as bagging) act as a variance reduction mechanism, i.e., they reduce the variance component of the error. Moreover, empirical results suggest that other ensemble techniques (such as AdaBoost) reduce both the bias and the variance parts of the error.*



Using ensembles to reduce the variance properties of prediction errors leads to the key benefit of using ensembles in the first place: to improve predictive performance.

Use Ensembles to Improve Performance

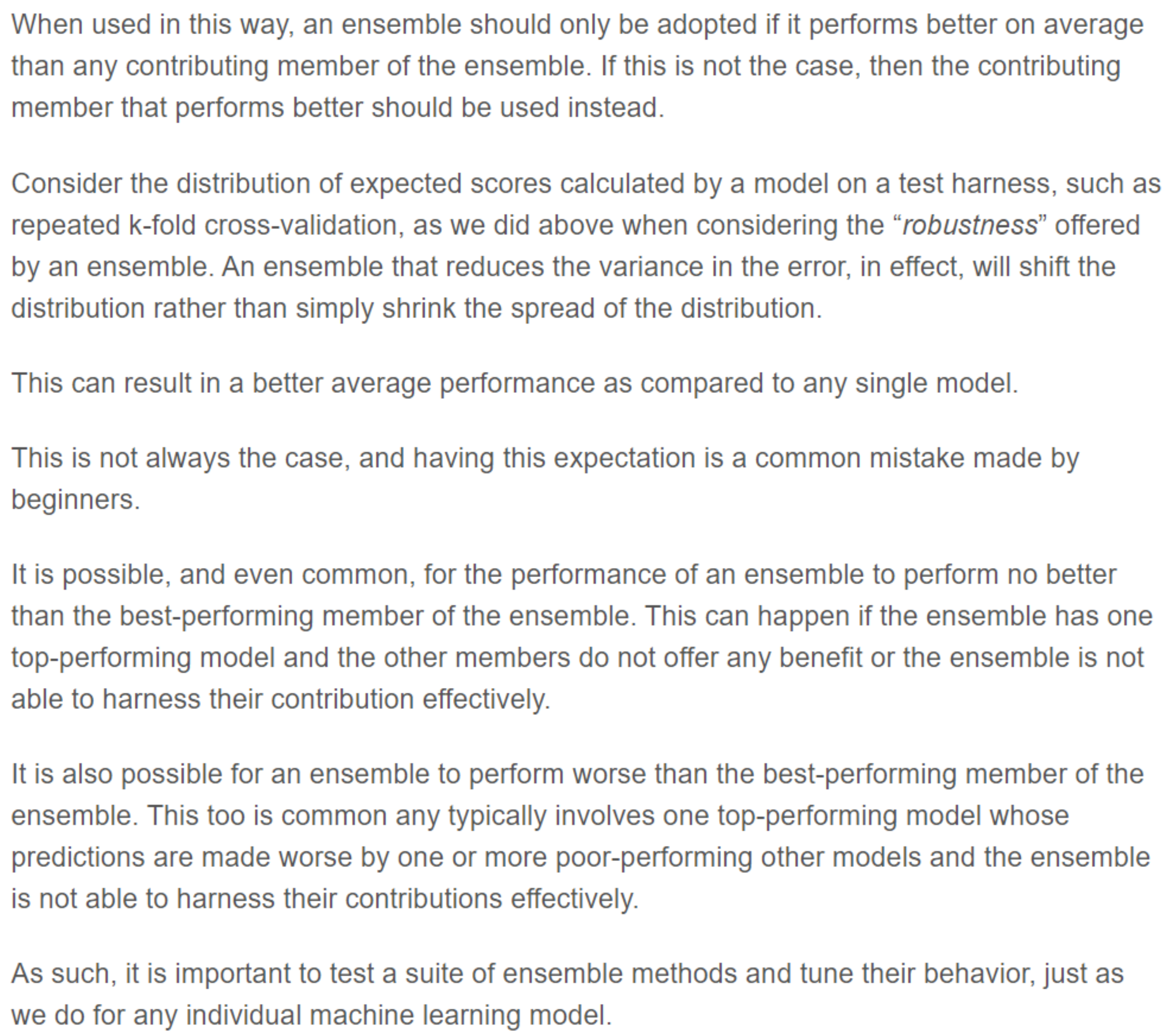
Reducing the variance element of the prediction error improves predictive performance.

We explicitly use ensemble learning to seek better predictive performance, such as lower error on regression or high accuracy for classification.

> *… there is a way to improve model accuracy that is easier and more powerful than judicious algorithm selection: one can gather models into ensembles.*

This is the **primary use of ensemble learning methods** and the benefit demonstrated through the use of ensembles by the majority of winners of machine learning competitions, such as the Netflix prize and competitions on Kaggle.

> *In the Netflix Prize, a contest ran for two years in which the first team to submit a model improving on Netflix's internal recommendation system by 10% would win $1,000,000. […] the final edge was obtained by weighing contributions from the models of up to 30 competitors.*



When used in this way, an ensemble should only be adopted if it performs better on average than any contributing member of the ensemble. If this is not the case, then the contributing member that performs better should be used instead.

Consider the distribution of expected scores calculated by a model on a test harness, such as repeated k-fold cross-validation, as we did above when considering the “*robustness*” offered by an ensemble. An ensemble that reduces the variance in the error, in effect, will shift the distribution rather than simply shrink the spread of the distribution.

This can result in a better average performance as compared to any single model.

This is not always the case, and having this expectation is a common mistake made by beginners.

It is possible, and even common, for the performance of an ensemble to perform no better than the best-performing member of the ensemble. This can happen if the ensemble has one top-performing model and the other members do not offer any benefit or the ensemble is not able to harness their contribution effectively.

It is also possible for an ensemble to perform worse than the best-performing member of the ensemble. This too is common any typically involves one top-performing model whose predictions are made worse by one or more poor-performing other models and the ensemble is not able to harness their contributions effectively.

As such, it is important to test a suite of ensemble methods and tune their behavior, just as we do for any individual machine learning model.

To apply the boosting approach, we start with a method or algorithm for finding the rough rules of thumb.

The boosting algorithm calls this “weak” or “base” learning algorithm repeatedly, each time feeding it a different subset of the training examples (or, to be more precise, a different distribution or weighting over the training examples).

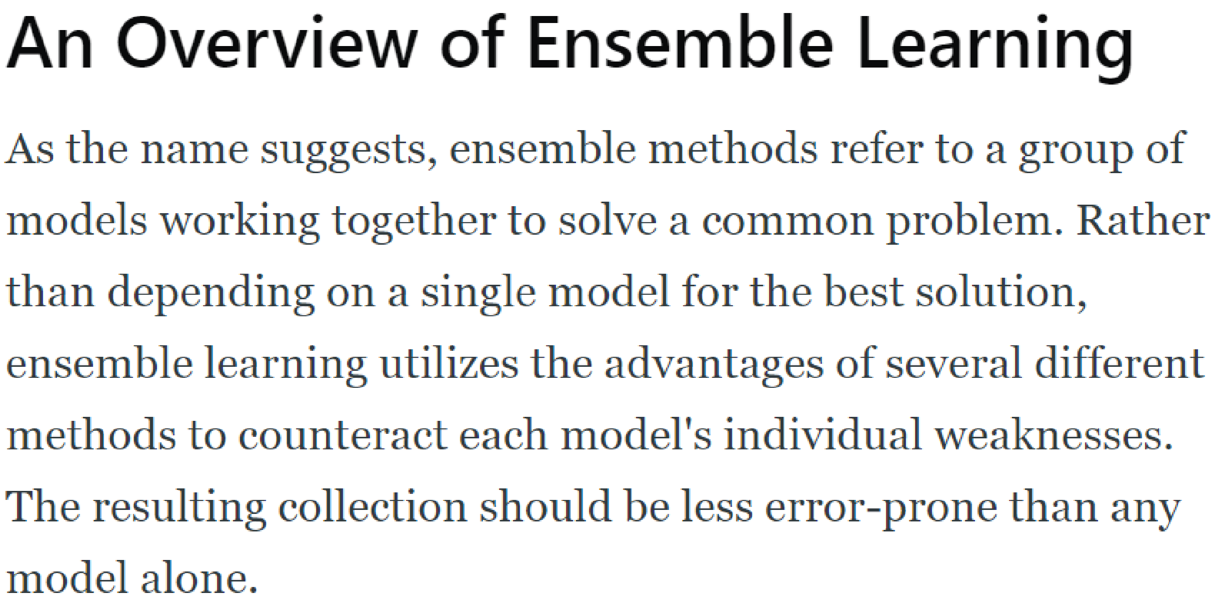
Each time it is called, the base learning algorithm generates a new weak prediction rule, and after many rounds, the boosting algorithm must combine these weak rules into a single prediction rule that, hopefully, will be much more accurate than any one of the weak rules.

To make this approach work, there are two fundamental questions that must be answered: (i) How should each distribution be chosen on each round, and second, how should the weak rules be combined into a single rule?

Regarding the choice of distribution, the technique that we advocate is to place the most weight on the examples most often misclassified by the preceding weak rules; this has the effect of forcing the base learner to focus its attention on the “hardest” examples.

(ii) As for combining the weak rules, simply taking a (weighted) majority vote of their predictions is natural and effective.

Boosting refers to a general and provably effective method of producing a very accurate prediction rule by combining rough and moderately inaccurate rules of thumb.



An Overview of Ensemble Learning

As the name suggests, ensemble methods refer to a group of models working together to solve a common problem. Rather than depending on a single model for the best solution, ensemble learning utilizes the advantages of several different methods to counteract each model's individual weaknesses. The resulting collection should be less error-prone than any model alone.

Why Use Ensemble Learning?

Combining several weak models can result in what we call a *strong learner*.

We can use ensemble methods to combine different models in two ways: either using a single base learning algorithm that remains the same across all models (a *homogeneous ensemble model*), or using multiple base learning algorithms that differ for each model (a *heterogeneous ensemble model*).

Generally speaking, ensemble learning is used with decision trees, since they're a reliable way of achieving regularization. Usually, as the number of levels increases in a decision tree, the model becomes vulnerable to high-variance and might overfit (resulting in high error on test data). We use ensemble techniques with general rules (vs. highly specific rules) in order to implement regularization and prevent overfitting.

This puts forth the advantages of ensemble learning:

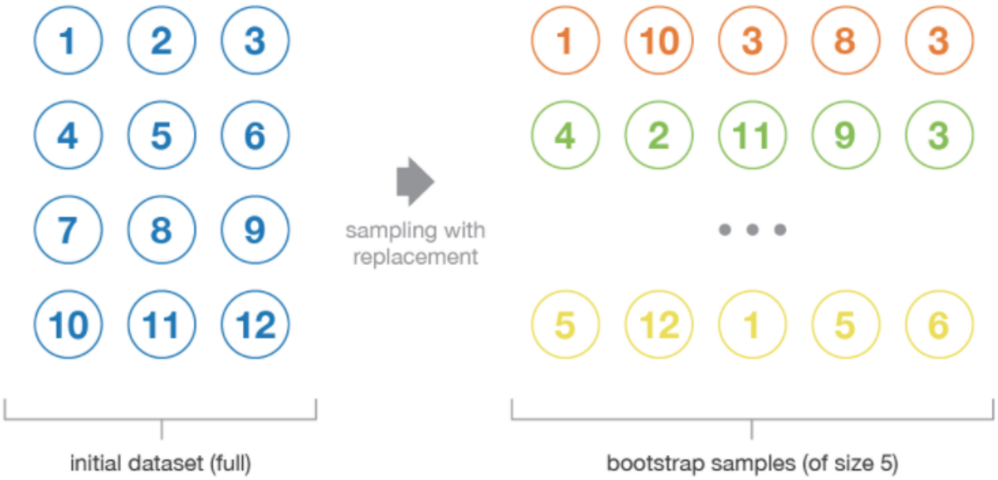
- Ensures reliability of the predictions.
- Ensures the stability/robustness of the model.

What Is Bagging?

Bagging, also known as **bootstrap aggregating**, is the aggregation of multiple versions of a predicted model. Each model is trained individually, and combined using an averaging process. The primary focus of bagging is to achieve less variance than any model has individually. To understand bagging, let's first understand the term **bootstrapping**.

Bootstrapping

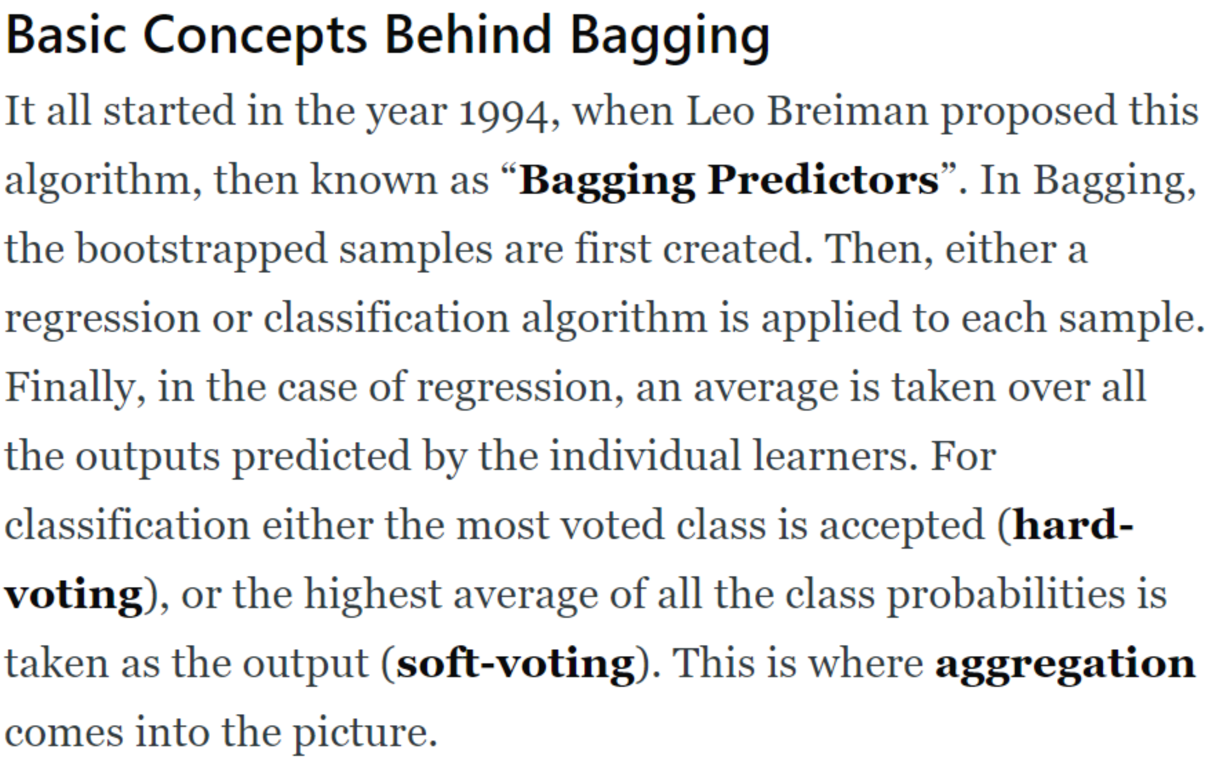
Bootstrapping is the process of generating bootstrapped samples from the given dataset. The samples are formulated by randomly drawing the data points with replacement.



The resampled data contains different characteristics that are as a whole in the original data. It draws the distribution present in the data points, and also tend to remain different from each other, i.e. the data distribution has to remain intact whilst maintaining the dissimilarity among the bootstrapped samples. This, in turn, helps in developing robust models.

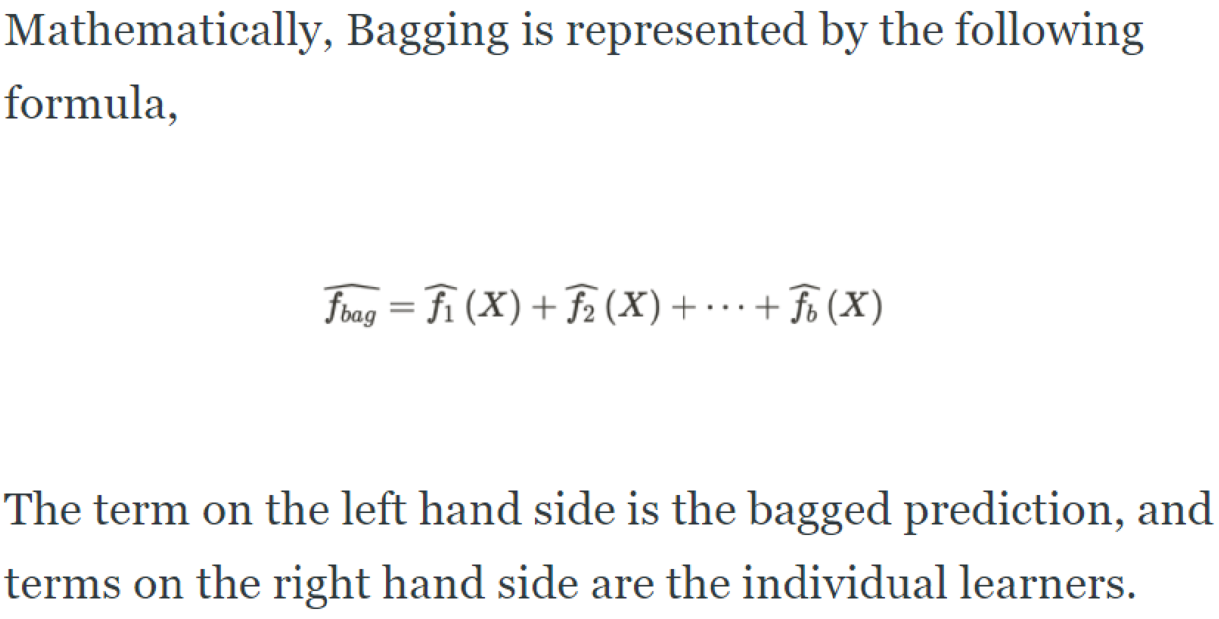
Bootstrapping also helps to avoid the problem of overfitting. When different sets of training data are used in the model, the model becomes resilient to generating errors, and therefore, performs well with the test data, and thus reduces the variance by maintaining its strong foothold in the test data. Testing with different variations doesn’t bias the model towards an incorrect solution.

Now, when we try to build fully independent models, it requires huge amounts of data. Thus, Bootstrapping is the option that we opt for by utilizing the approximate properties of the dataset under consideration.



Basic Concepts Behind Bagging

It all started in the year 1994, when Leo Breiman proposed this algorithm, then known as "**Bagging Predictors**". In Bagging, the bootstrapped samples are first created. Then, either a regression or classification algorithm is applied to each sample. Finally, in the case of regression, an average is taken over all the outputs predicted by the individual learners. For classification either the most voted class is accepted (**hard-voting**), or the highest average of all the class probabilities is taken as the output (**soft-voting**). This is where **aggregation** comes into the picture.



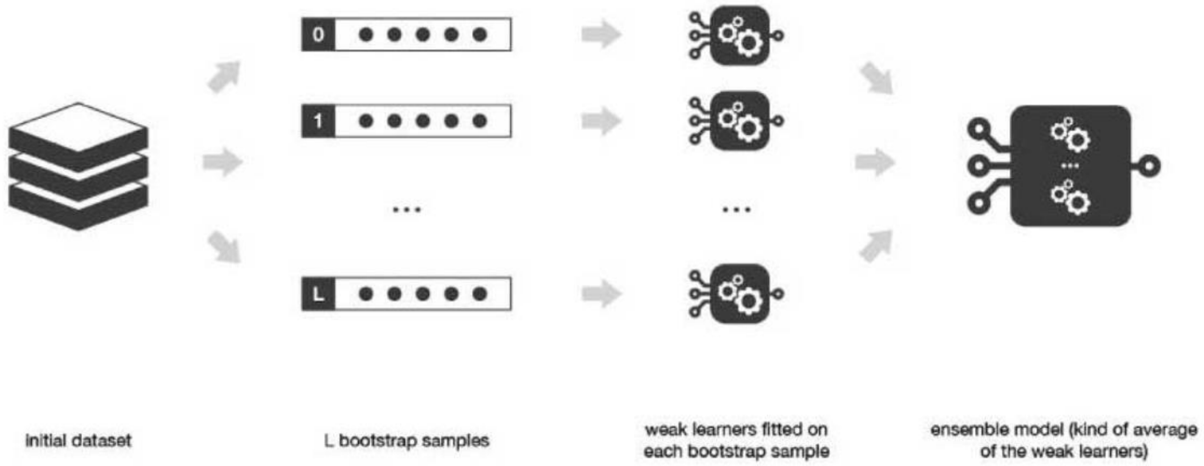
Mathematically, Bagging is represented by the following formula,

$$\widehat{f_{bag}} = \widehat{f_1}(X) + \widehat{f_2}(X) + \cdots + \widehat{f_b}(X)$$

The term on the left hand side is the bagged prediction, and terms on the right hand side are the individual learners.

Bagging works especially well when the learners are unstable and tend to overfit, i.e. small changes in the training data lead to major changes in the predicted output. It effectively reduces the variance by aggregating the individual learners composed of different statistical properties, such as different standard deviations, means, etc. It works well for high variance models such as Decision Trees. When used with low variance models such as linear regression, it doesn't really affect the learning process. The number of base learners (trees) to be chosen depends on the characteristics of the dataset. Using too many trees doesn't lead to overfitting, but can consume a lot of computational power.

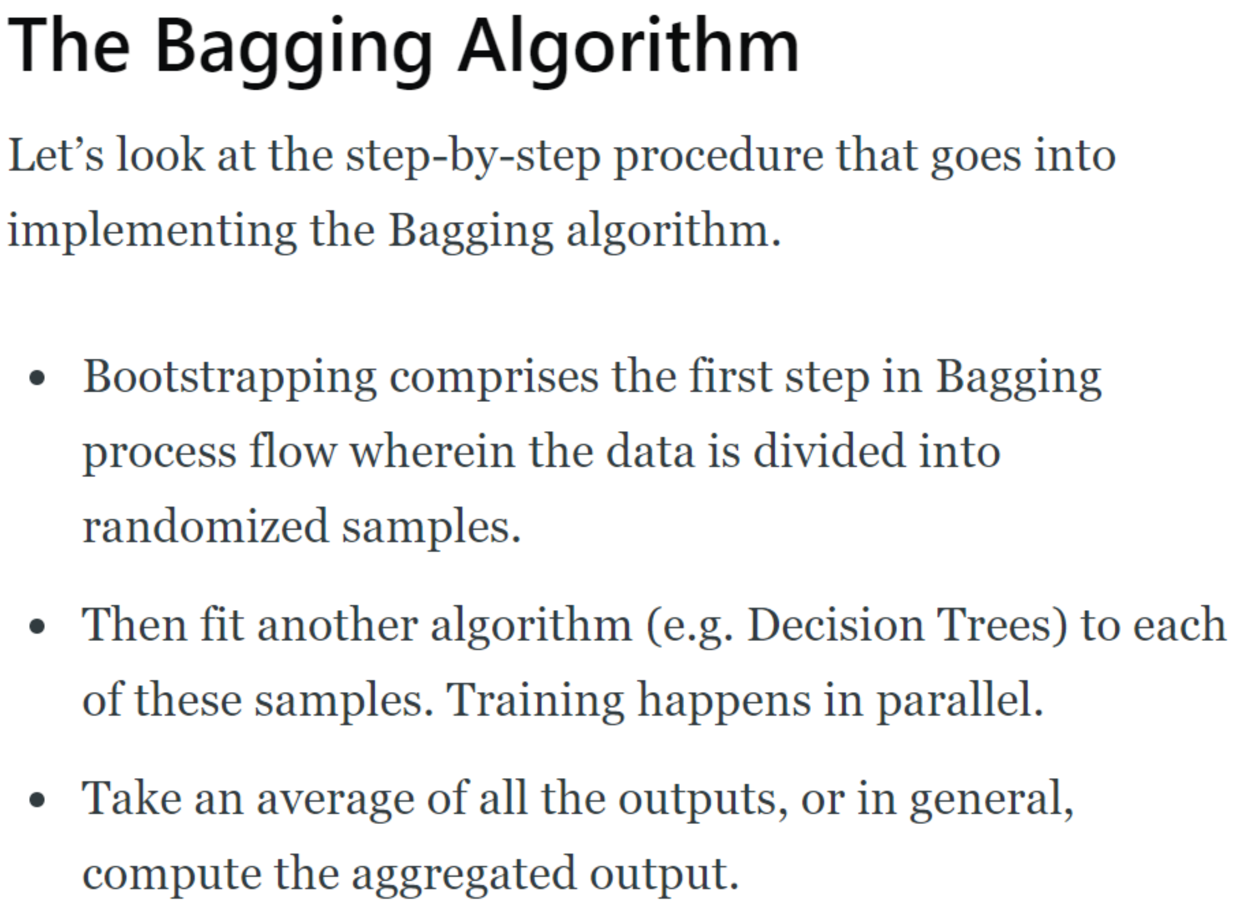
Bagging can be done in parallel to keep a check on excessive computational resources. This is a one good advantages that comes with it, and often is a booster to increase the usage of the algorithm in a variety of areas.


0
1
...
L
...
initial dataset
L bootstrap samples
weak learners fitted on each bootstrap sample
ensemble model (kind of average of the weak learners)

Applications of Bagging

Bagging technique is used in a variety of applications. One main advantage is that it reduces the variance in prediction by generating additional data whilst applying different combinations and repetitions (replacements in the bootstrapped samples) in the training data. Below are some applications where the bagging algorithm is widely used:

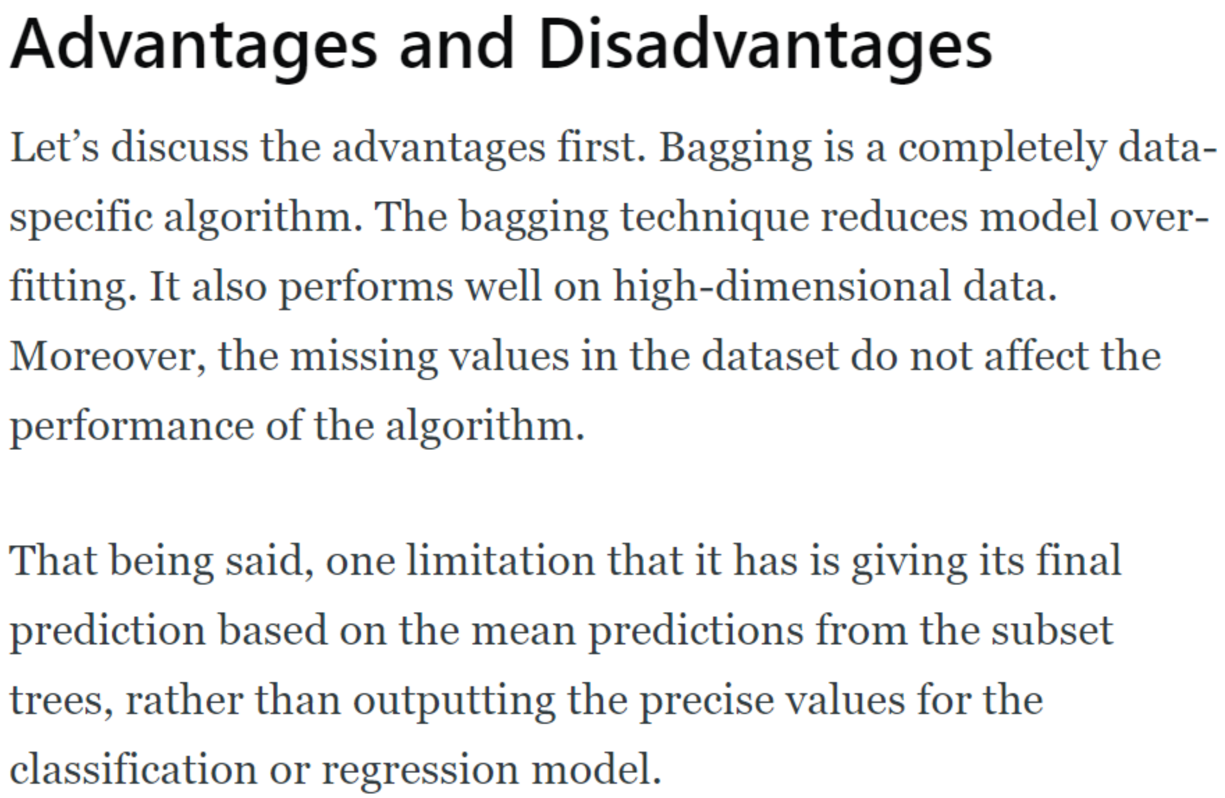
- Banking Sector
- Medical Data Predictions
- High-dimensional data
- Land cover mapping
- Fraud detection
- Network Intrusion Detection Systems
- Medical fields like neuroscience, prosthetics, etc.



The Bagging Algorithm

Let’s look at the step-by-step procedure that goes into implementing the Bagging algorithm.

- Bootstrapping comprises the first step in Bagging process flow wherein the data is divided into randomized samples.
- Then fit another algorithm (e.g. Decision Trees) to each of these samples. Training happens in parallel.
- Take an average of all the outputs, or in general, compute the aggregated output.



Advantages and Disadvantages

Let’s discuss the advantages first. Bagging is a completely data-specific algorithm. The bagging technique reduces model over-fitting. It also performs well on high-dimensional data. Moreover, the missing values in the dataset do not affect the performance of the algorithm.

That being said, one limitation that it has is giving its final prediction based on the mean predictions from the subset trees, rather than outputting the precise values for the classification or regression model.

CLUSTERING ENSEMBLES

Finding a consensus clustering from multiple partitions is a different problem that can be approached from: graph-based, combinatorial, or statistical perspectives.

Problems/Challenges in Clustering Ensembles:

- Major difficulty in finding a consensus partition from the output partitions of various clustering algorithms.
- The partitions are unlabeled and therefore, there is no explicit correspondence between labels delivered by different clusterings.
- An extra complexity arises when different partitions contain different number of clusters, often resulting in intractable label correspondence problem.
- Choice of clustering algorithms for the ensemble.
- Bootstrap methods can be used to generate clustering ensembles.

3 Major issues in clustering combination design:

(1) Consensus Function:

- How to combine different clusterings?
- How to resolve label correspondence problem?
- How to ensure symmetrical and unbiased consensus w.r.t. all component partitions?

(2) Diversity of Clustering:

- How to generate different partitions?
- What is the source of diversity in components?

(3) Strength of Constituents/Components:

- How weak each input is?

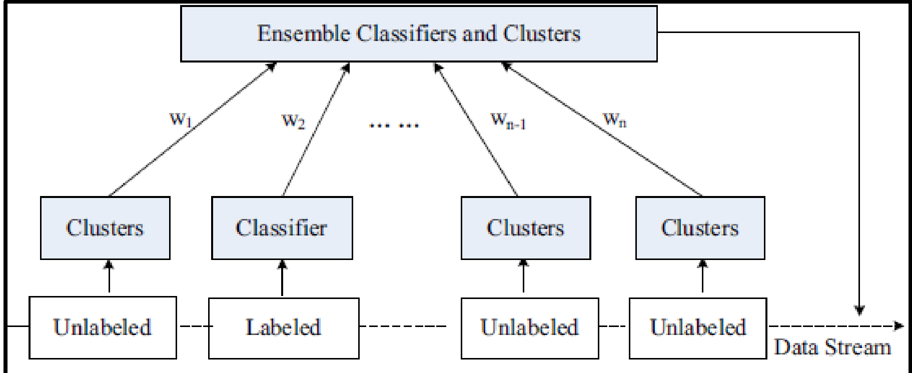
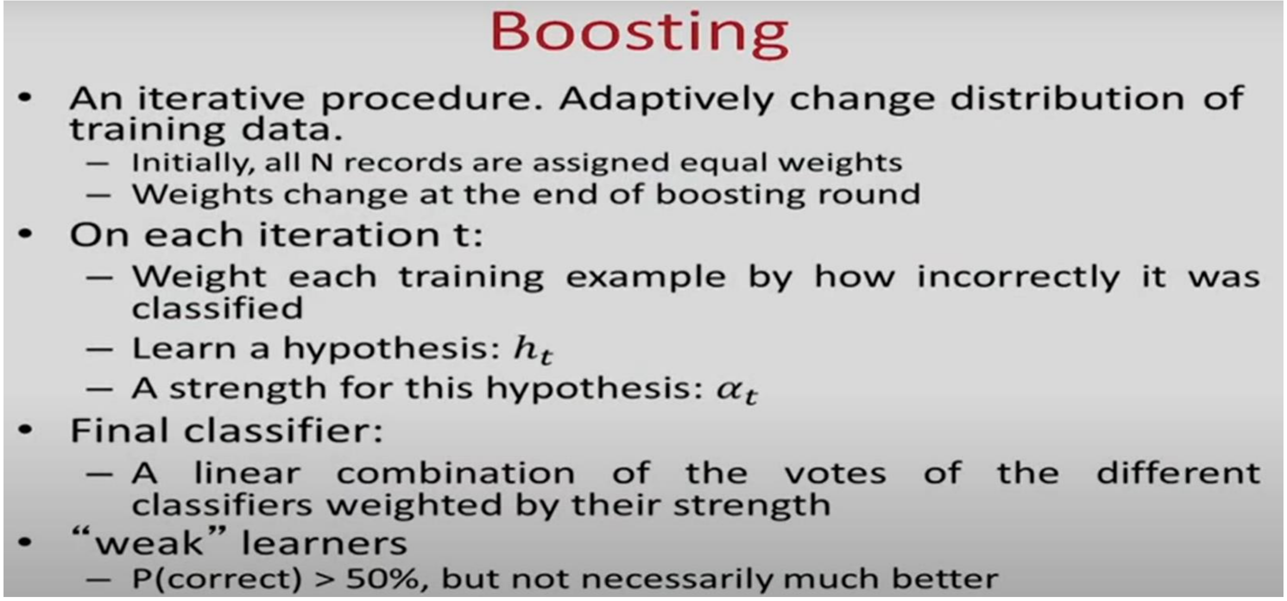
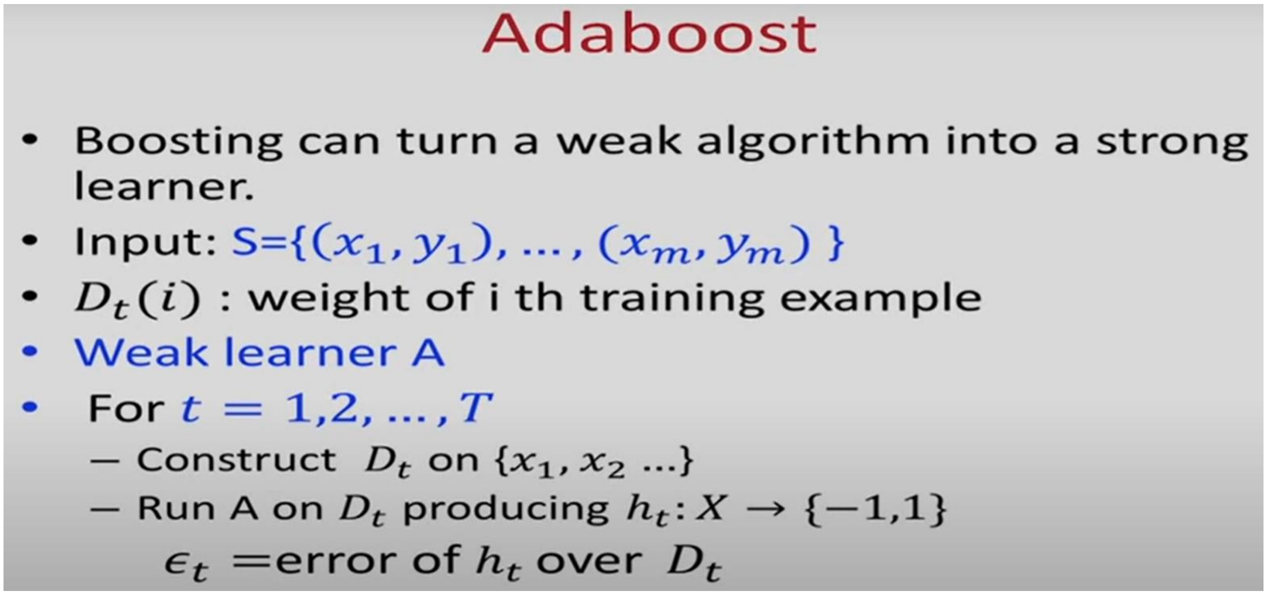


Figure 1. An illustration of the ensemble classifier and cluster model.



Boosting

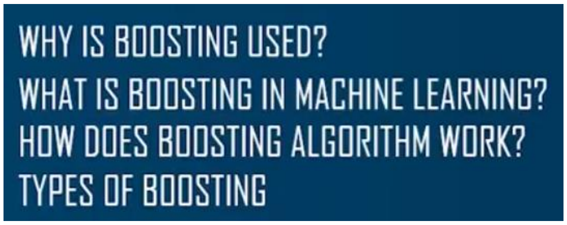
- An iterative procedure. Adaptively change distribution of training data.
  - Initially, all N records are assigned equal weights
  - Weights change at the end of boosting round
- On each iteration t:
  - Weight each training example by how incorrectly it was classified
  - Learn a hypothesis: h_t
  - A strength for this hypothesis: α_t
- Final classifier:
  - A linear combination of the votes of the different classifiers weighted by their strength
- “weak” learners
  - P(correct) > 50%, but not necessarily much better



Adaboost

- Boosting can turn a weak algorithm into a strong learner.
- Input: S={$(x_1, y_1), \dots, (x_m, y_m)$ }
- $D_t(i)$: weight of i th training example
- Weak learner A
- For $t = 1,2, \dots, T$
  - Construct D_t on $\{x_1, x_2 \dots\}$
  - Run A on D_t producing $h_t: X \to \{-1,1\}$

    ϵ_t =error of h_t over D_t



WHY IS BOOSTING USED?

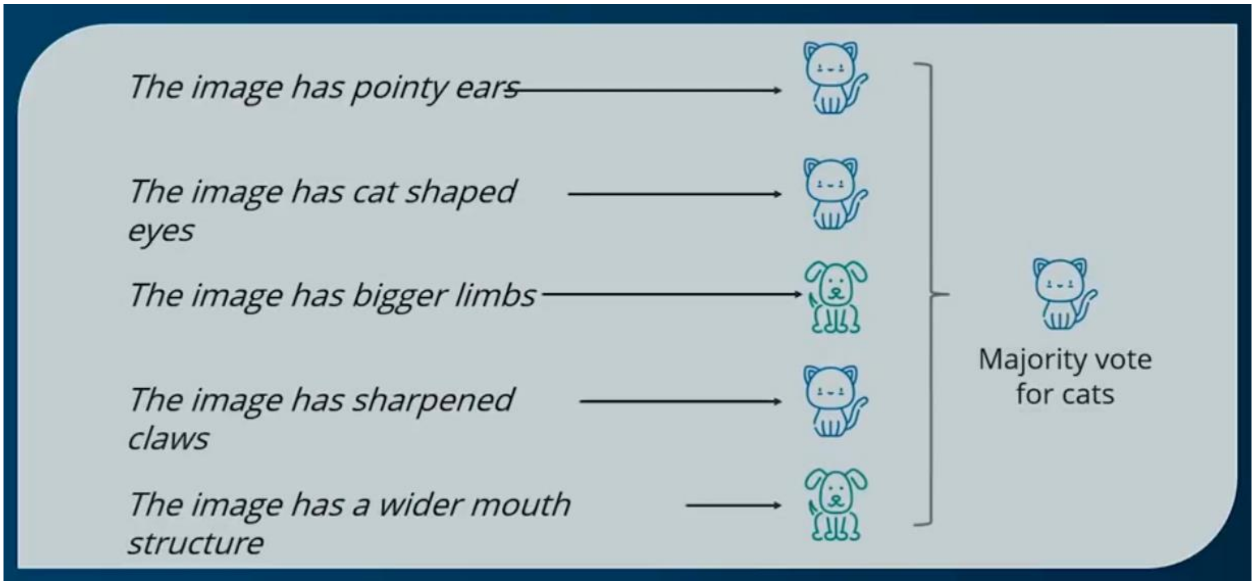
WHAT IS BOOSTING IN MACHINE LEARNING?

HOW DOES BOOSTING ALGORITHM WORK?

TYPES OF BOOSTING

01

WHY IS BOOSTING USED?



The image has pointy ears

The image has cat shaped eyes

The image has bigger limbs

The image has sharpened claws

The image has a wider mouth structure

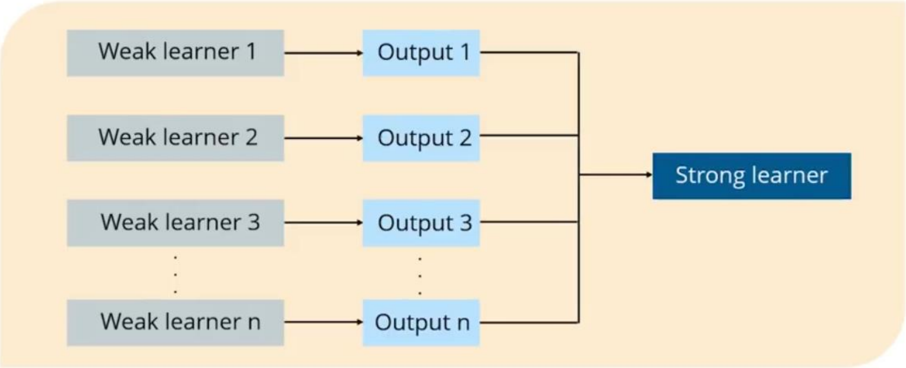
Majority vote for cats

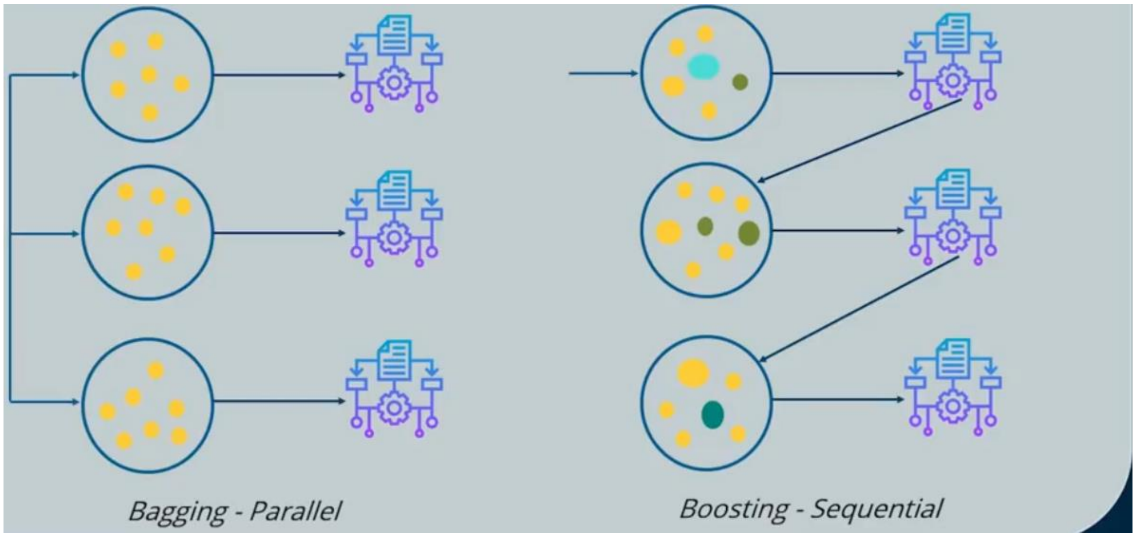
Why Is Boosting Used?

These rules help us identify whether an image is a Dog or a cat but, if we were to classify an image based on an individual (single) rule, the prediction would be flawed. Each of these rules, individually, are called weak learners because these rules are not strong enough to make predictions.

WHAT IS BOOSTING?

Boosting is a process that uses a set of Machine Learning algorithms to combine weak learner to form strong learners in order to increase the accuracy of the model.



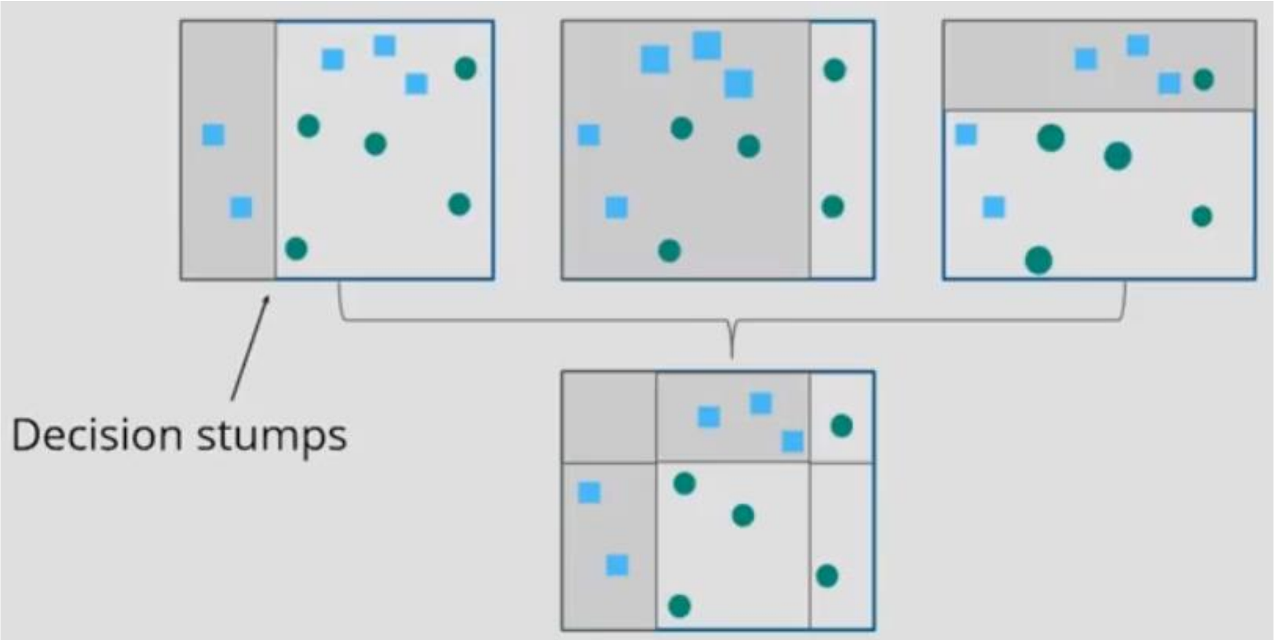


What Is Ensemble Learning?

Ensemble learning is a method that is used to enhance the performance of Machine Learning model by combining several learners. When compared to a single model, this type of learning builds models with improved efficiency and accuracy

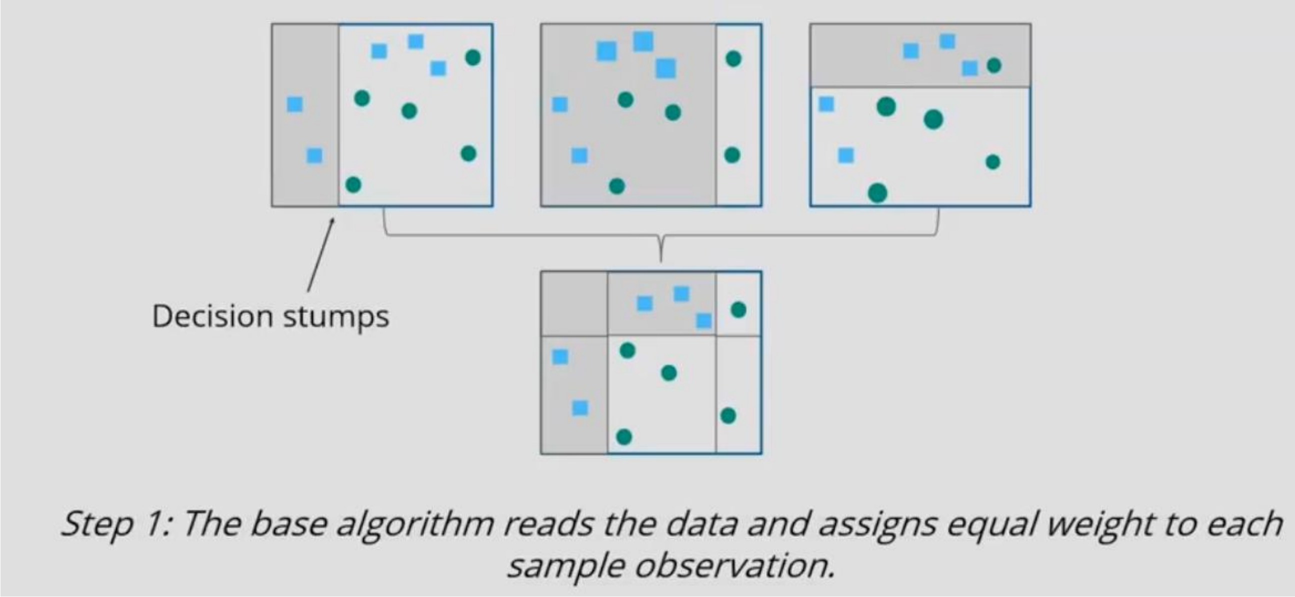
03

HOW DOES BOOSTING ALGORITHM WORK?

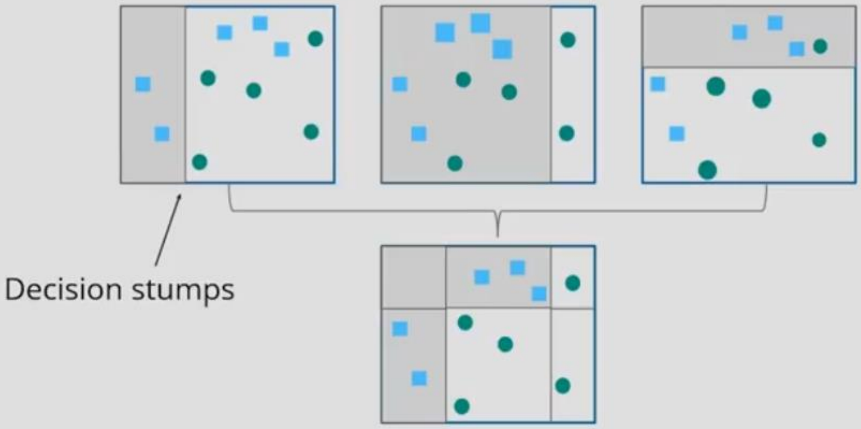


How Does Boosting Algorithm Work?

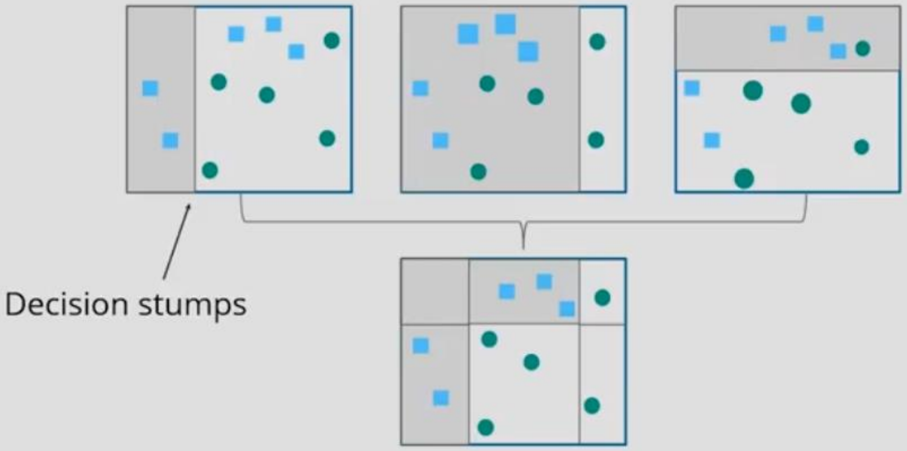
The basic principle behind the working of the boosting algorithm is to generate multiple weak learners and combine their predictions to form one strong rule



Step 1: The base algorithm reads the data and assigns equal weight to each sample observation.



Step 2: False predictions are assigned to the next base learner with a higher weightage on these incorrect predictions.

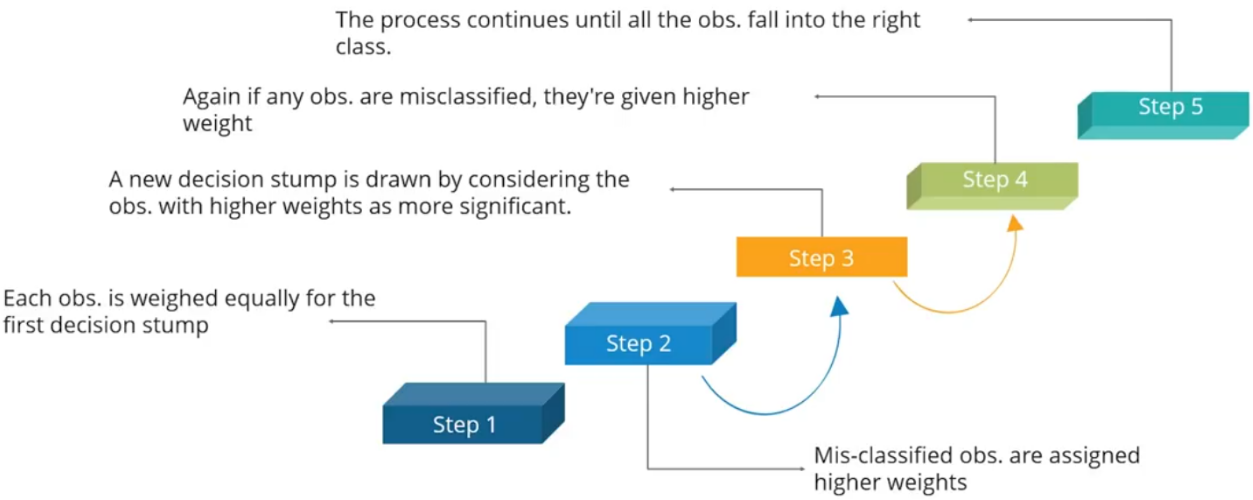


Step 3: Repeat step 2 until the algorithm can correctly classify the output.

04

TYPES OF BOOSTING

ADAPTIVE BOOSTING



Each obs. is weighed equally for the first decision stump

Mis-classified obs. are assigned higher weights

A new decision stump is drawn by considering the obs. with higher weights as more significant.

Again if any obs. are misclassified, they're given higher weight

The process continues until all the obs. fall into the right class.

GRADIENT BOOSTING

In Gradient Boosting, base learners are generated sequentially in such a way that the present base learner is always more effective than the previous one.

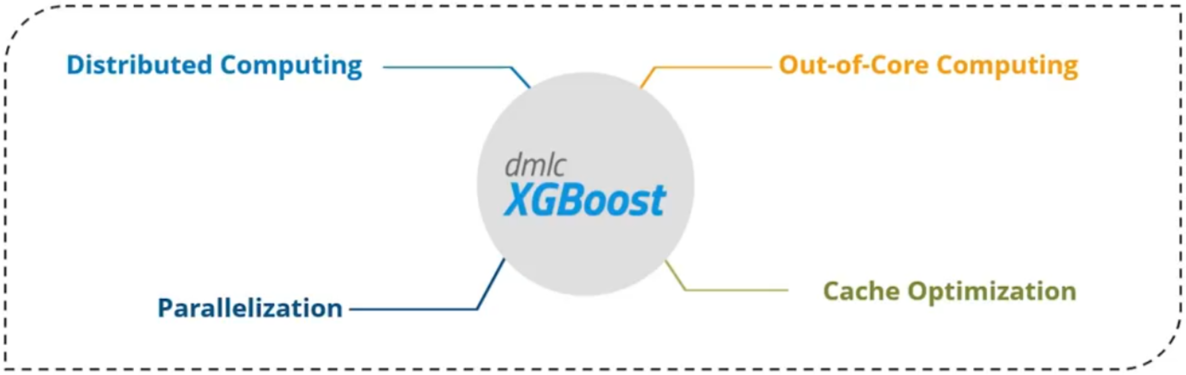
optimize the loss function of the previous learner

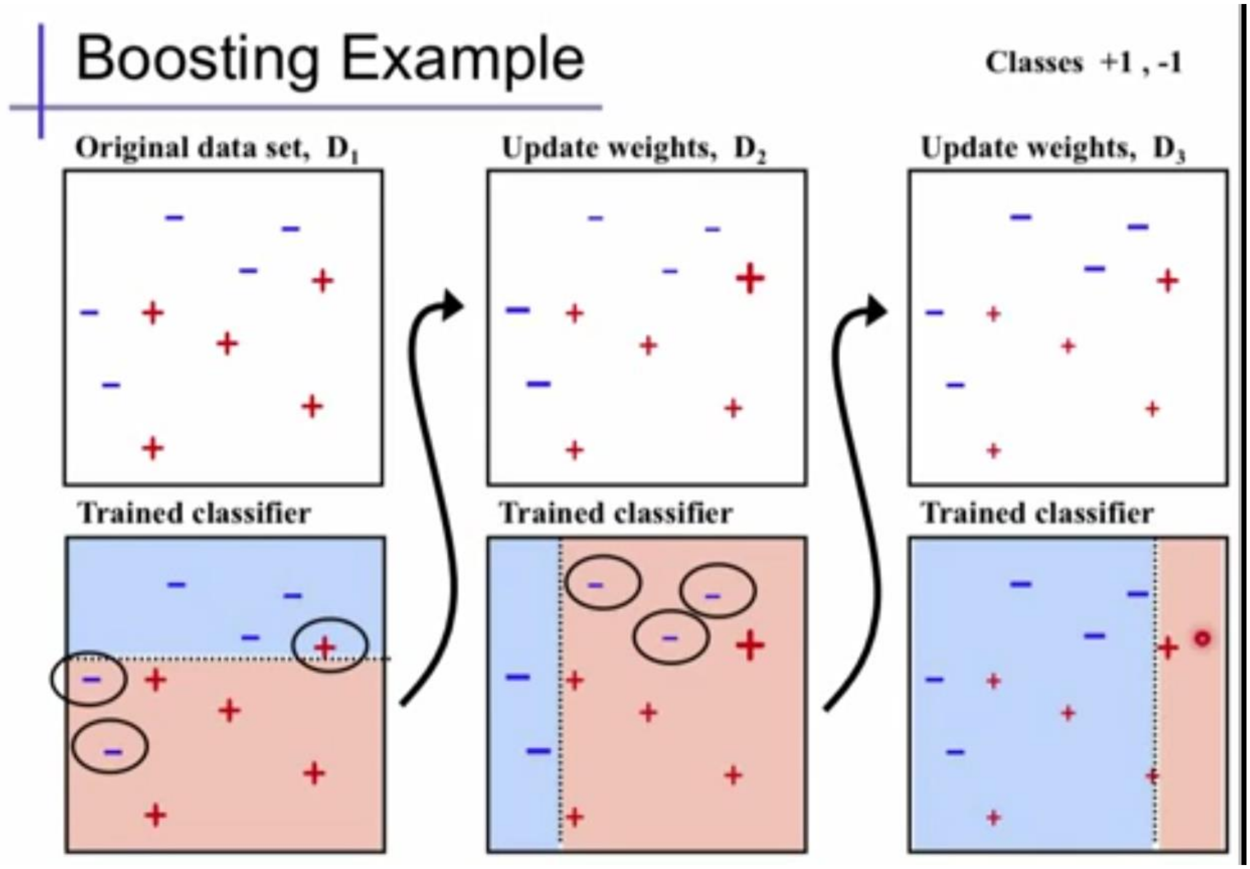
Three main components:

- ***Loss function** that needs to be ameliorated.*
- ***Weak learner** for computing predictions and forming strong learners.*
- *An **Additive Model** that will regularize the loss function.*

XGBOOST

XGBoost is an advanced version of Gradient boosting method that is designed to focus on computational speed and model efficiency.





Boosting Example

Classes +1 , -1

Original data set, D_1

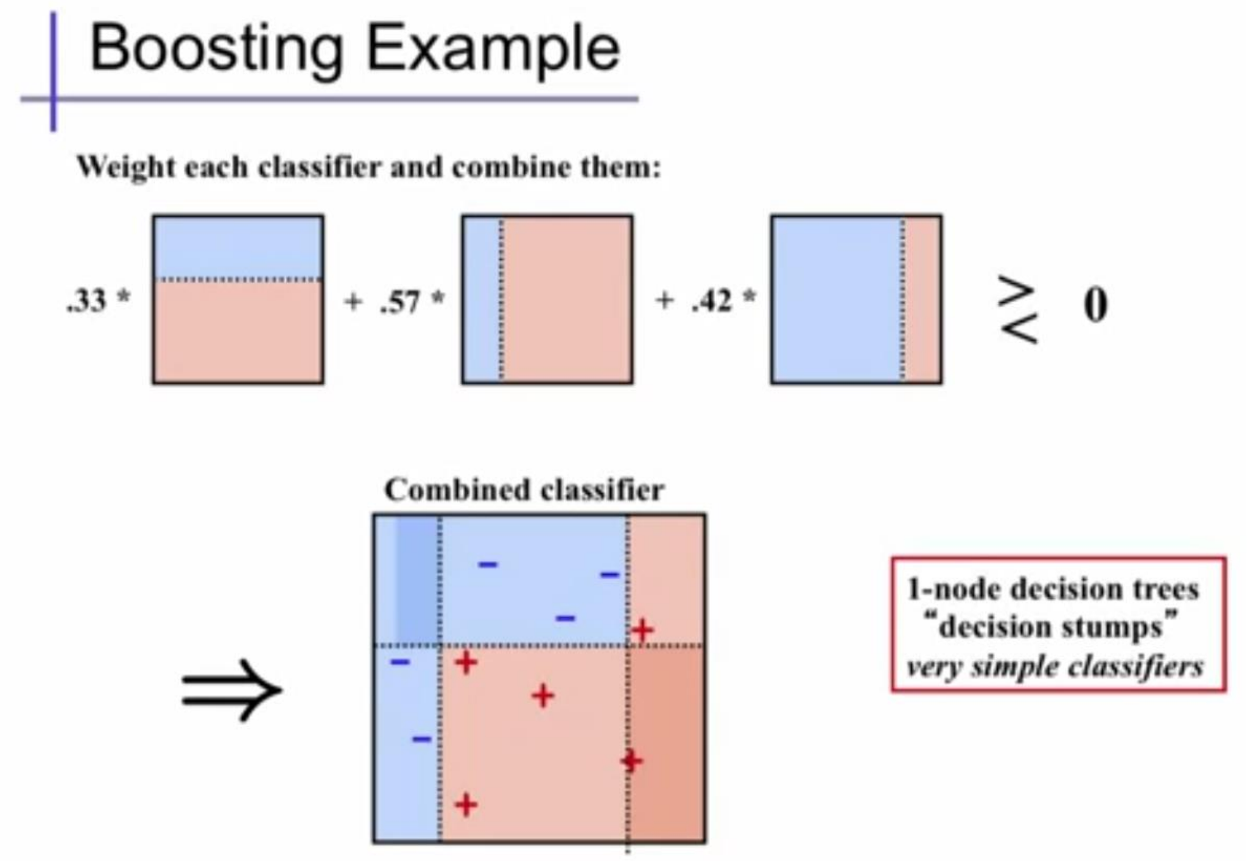
Update weights, D_2

Update weights, D_3

Trained classifier

Trained classifier

Trained classifier

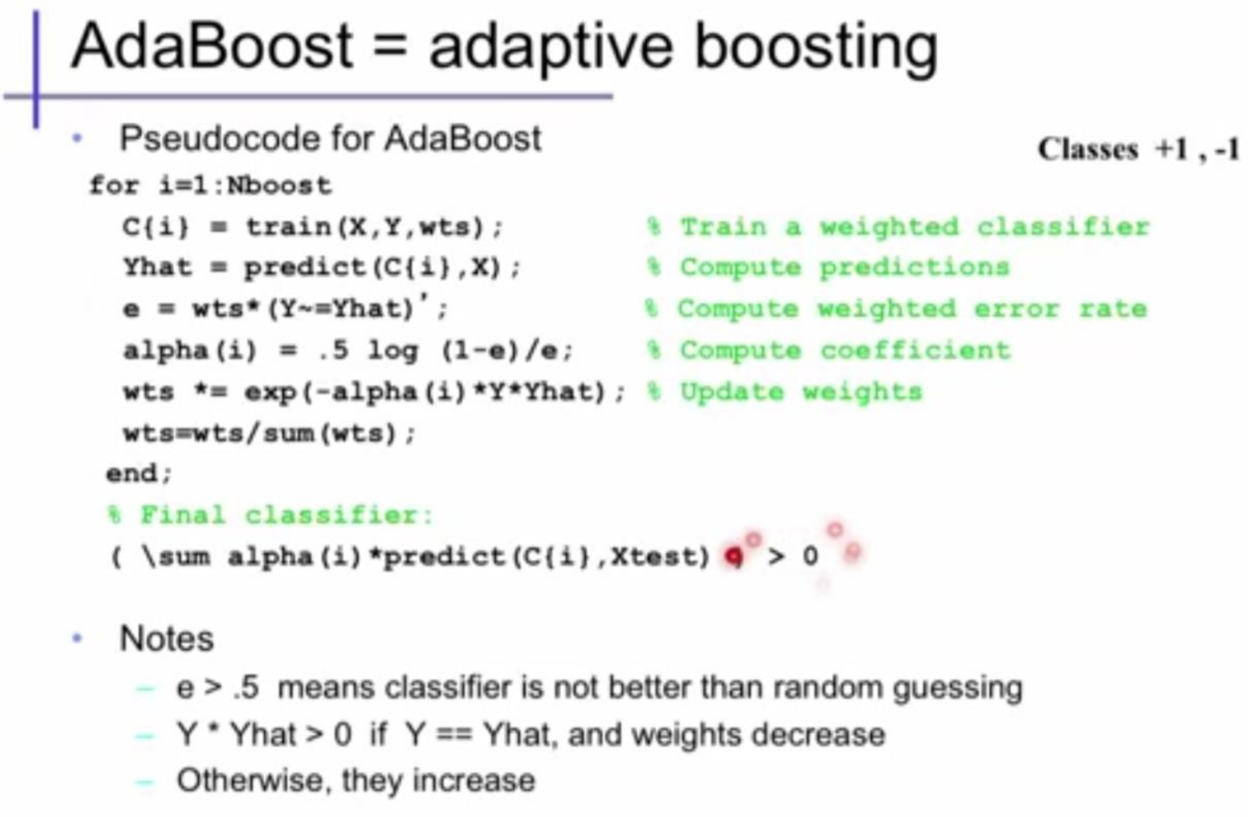


Boosting Example

Weight each classifier and combine them:

Combined classifier

1-node decision trees
"decision stumps"
very simple classifiers



AdaBoost = adaptive boosting

- Pseudocode for AdaBoost

Classes +1 , -1

```
for i=1:Nboost
  C{i} = train(X,Y,wts);         % Train a weighted classifier
  Yhat = predict(C{i},X);        % Compute predictions
  e = wts*(Y~=Yhat)';            % Compute weighted error rate
  alpha(i) = .5 log (1-e)/e;     % Compute coefficient
  wts *= exp(-alpha(i)*Y*Yhat);  % Update weights
  wts=wts/sum(wts);
 end;
 % Final classifier:
 ( \sum alpha(i)*predict(C{i},Xtest) ) > 0
```

- Notes
  - e > .5 means classifier is not better than random guessing
  - Y * Yhat > 0 if Y == Yhat, and weights decrease
  - Otherwise, they increase