

ENSEMBLE LEARNING  
BAGGING AND BOOSTING

**Fida Salim and Soumik Bhattacharyya**

National School of Science Education and Research  
Bhubaneswar, Odisha

March 1, 2023

# PART I: ENSEMBLE LEARNING

- 1 Introduction . . . . . 4**
  - 1.1 Bias and Variance . . . . . 4
  - 1.2 Ensemble Learning: Definition . . . . . 6
  - 1.3 Ensemble Learning: Example . . . . . 7

## PART II: BAGGING AND BOOSTING

<b>1</b>	<b>Bagging</b>	<b>9</b>
1.1	Definition and Idea	9
1.2	Vanilla Bagging	10
1.2.1	Algorithm	10
1.3	Random Forest	14
1.3.1	Algorithm	15
1.3.2	Key Benefits and Challenges	16
1.3.3	Applications	17
<b>2</b>	<b>Boosting</b>	<b>18</b>
2.1	Definition and Idea	18
2.2	Gradient Boosting	19
2.2.1	Regressor Algorithm	20
2.2.2	Example	23
2.2.3	Key Benefits and Challenges	26
2.2.4	Applications	27
<b>3</b>	<b>Summary</b>	<b>28</b>
3.1	Bagging Vs Boosting	28

# Part I

## ENSEMBLE LEARNING

# INTRODUCTION

## BIAS AND VARIANCE

- ▶ Machine learning models are designed to generalize from training data to new, unseen data. However, there is a fundamental trade-off between the bias and variance of a model that can affect its ability to generalize.
- ▶ Bias of a model is the error in predicting the training dataset, while variance is the change in model with respect to change in training dataset.
- ▶ A model with high bias tends to underfit the training data, i.e. the model is too simple to capture the underlying patterns in the data. In contrast, a model with high variance is prone to memorize the noise in the training data rather than the underlying patterns. In other words, it overfit the training data.
- ▶ Therefore, we look for a model which is both low bias and low variance, however it is difficult to construct a single model that simultaneously achieves both of them.

## WHY IS IT DIFFICULT TO CONSTRUCT A LOW-BIAS LOW-VARIANCE MACHINE?

The bias-variance trade-off can be mathematically demonstrated using the concept of the mean squared error (MSE) of a machine learning model. The MSE measures the average squared difference between the predicted values and the true values of the target variable in the training data. It can be decomposed into three components: the squared bias, the variance, and the irreducible error:

$$MSE = Bias^2 + Variance + Irreducible\ error^1$$

For a single model, once the minima of MSE is reached, it is impossible to lower both bias and variance simultaneously. However, ensemble learning can be used to tune the bias-variance trade-off by combining models with different levels of bias and variance. Subsequently, combining multiple models can achieve better performance than any individual model.

---

<sup>1</sup>Irreducible Error represents the minimum achievable error of any model on the given data, due to the inherent noise or variability in the data. The proof of the equation is as follows,

$$\begin{aligned} E[(Y - \hat{Y})^2] &= E[(Y - E[\hat{Y}] + E[\hat{Y}] - \hat{Y})^2] \\ &= E[(Y - E[\hat{Y}])^2] + E[(E[\hat{Y}] - \hat{Y})^2] + 2E[(Y - E[\hat{Y}])(E[\hat{Y}] - \hat{Y})] \\ &= Bias^2 + Variance + Irreducible\ error \end{aligned}$$

# INTRODUCTION

## ENSEMBLE LEARNING: DEFINITION

- ▶ In machine learning, ensemble learning methods are techniques used to combine multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.
- ▶ It works on the hypothesis that certain models do well in one aspect of the data, while others do well in modeling another. The idea is to leverage the strengths of different models and reduce their individual weaknesses.
- ▶ There are different types of ensemble learning, each with its own approach to combining models. The most common types are bagging, boosting, and stacking. In bagging, multiple models are trained on different subsets of the training data, which reduces the variance by averaging the predictions of the individual models. Boosting, on the other hand, focuses on reducing bias by sequentially training models on misclassified instances. Stacking combines multiple models by training a meta-model on their predictions. The meta-model learns to combine the strengths of the individual models and generate a final prediction. We'll discuss Bagging and Boosting in detail in later slides.

## THE NETFLIX PRIZE: A FUN EXAMPLE

Ensemble learning has been successfully applied in various real-world scenarios, one such example is the Netflix Prize Challenge.

- ▶ In 2006, Netflix launched a competition to improve their recommendation algorithm, which recommends movies and TV shows to users based on their viewing history.
- ▶ The goal was to reduce the root mean square error (RMSE) of their existing algorithm by 10%. The competition attracted more than 40,000 teams from around the world, and the winning team, BellKor's Pragmatic Chaos, used an ensemble of multiple machine learning models to achieve a 10.06% improvement in RMSE.
- ▶ The winning solution was a blend of 107 different models, each using different combinations of features and algorithms. By combining the predictions of these individual models, the ensemble was able to achieve a much lower RMSE than any individual model.



## Part II

# BAGGING AND BOOSTING

# BAGGING

## DEFINITION AND IDEA

Bagging stands for Bootstrap Aggregating, which is a technique used in ensemble learning to reduce the variance of machine learning models. The idea behind bagging is to train multiple models on different subsets of the training data, and then combine their predictions to make the final prediction.

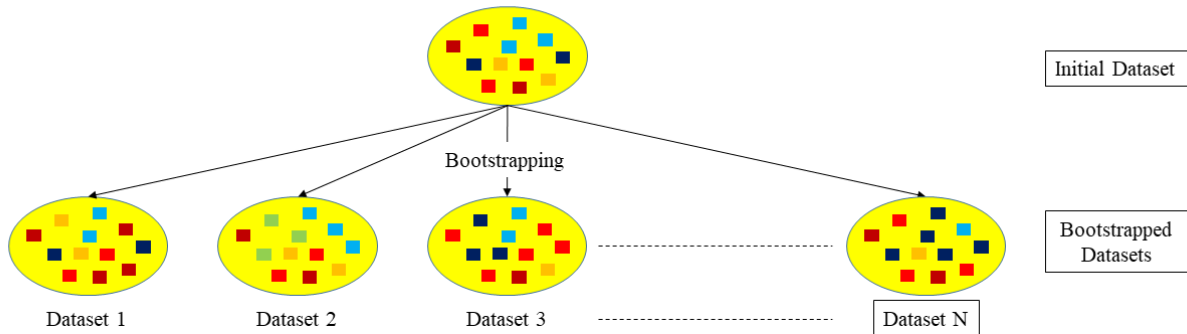
- ▶ In bagging, we randomly select a subset of the training data with replacement (bootstrap) to create a new training set for each model. This means that some data points may be present multiple times in a single subset, while others may be left out entirely.
- ▶ By training models on these different subsets, we can reduce the variance of the overall prediction, since each model is making its predictions based on a slightly different set of data. Bagging is particularly useful in applications where the individual models are prone to overfitting the training data, as bagging can help to reduce this overfitting by creating more diverse models.
- ▶ It is commonly used in applications such as classification, regression, and time series forecasting. We will discuss the detailed algorithm of Vanilla Bagging (and Random Forest, which is a stretch of this algorithm) in the next slides.

# VANILLA BAGGING

Vanilla Bagging is widely used in real-world applications including credit risk assessment, fraud detection, stock price prediction etc.

## ► Algorithm

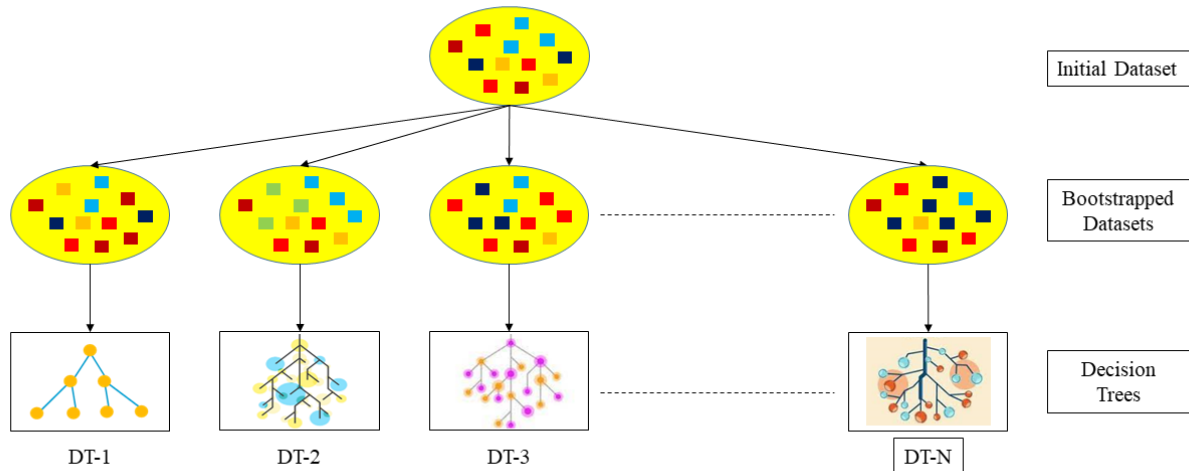
- In Vanilla Bagging, we use bootstrap method (randomly choosing an element with replacement) to make subset datasets of equal length as the initial dataset.
- Even though the number of elements in subsets are equal to the number of elements in the original dataset, they are not the same dataset as the draw has been done with replacements.
- **The number of subsets,  $N$ , is a hyperparameter.**



# VANILLA BAGGING ALGORITHM

CONTINUED

- After bootstrapping, a separate decision tree is built on each subset of data.
- It is not mandatory to use decision trees, Vanilla Bagging can be used with pretty much any algorithm that can handle multiple inputs, such as decision trees, neural networks, and support vector machines.



# VANILLA BAGGING ALGORITHM

## CONTINUED

- Each of the models in each subset predicts its own result.

### ▶ **Regression Problem:**

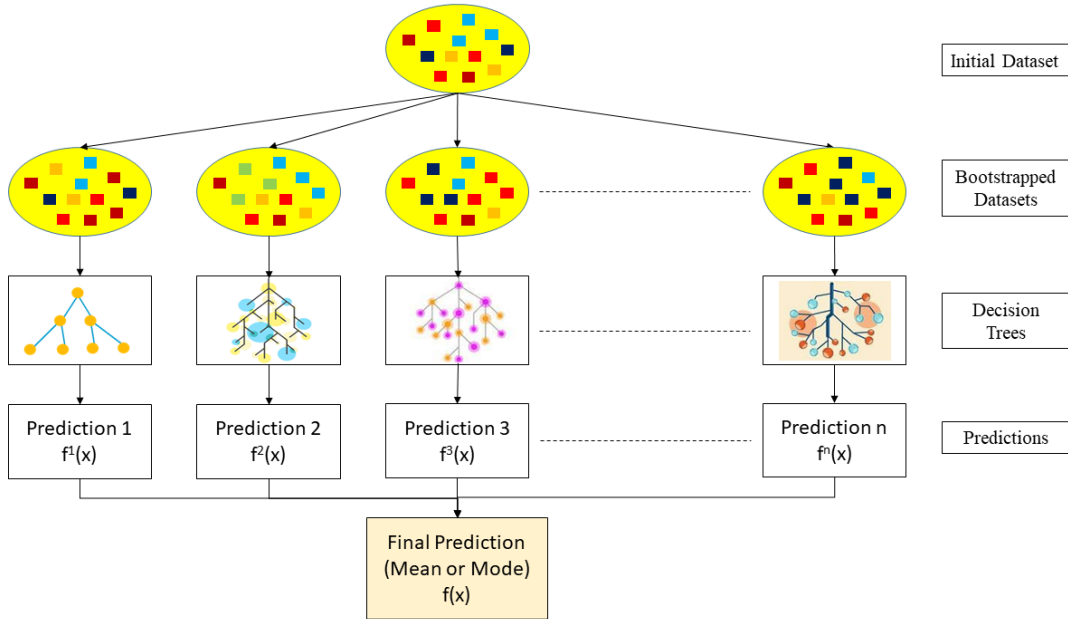
- In regression problems, the final output is calculated as the average of all the predicted results from each tree (or other model).

### ▶ **Classification Problem:**

- For binary classification problems, each model from each subset predicts either 'yes' or 'no'. The final output is determined by aggregating the predictions of all models, with a majority vote.
- For general cases, the mode of all models is taken as the final output.
- However, complications may arise in case of a tie. These cases can be handled with domain knowledge, classification with a confidence level or other complicated methods.

# VANILLA BAGGING ALGORITHM

CONTINUED



**Figure.** Algorithm for Vanilla Bagging

## RANDOM FOREST

- ▶ Random Forest is a popular ensemble learning algorithm, which is an extension of the vanilla bagging algorithm.
- ▶ The first algorithm for random decision forests was created in 1995 by Tin Kam Ho. In this algorithm, he introduced the idea of random feature selection for a high cardinality of feature space, which is the key difference between vanilla bagging and random forest.
- ▶ The algorithm for random forests is similar to that of bagging methods. However, in random forests, a subset of pre-decided length is formed from original feature space for each of the bootstrapped dataset.
- ▶ The feature subset is chosen randomly without replacements. **The length of the feature subsets,  $\xi_f$ , is a hyperparameter.**
- ▶ A decision tree is formed for each dataset and corresponding feature space, leading to a prediction from each. Final prediction is made following the same rules as of bagging, i.e. taking mean for regression and mode for classification.

# RANDOM FOREST ALGORITHM

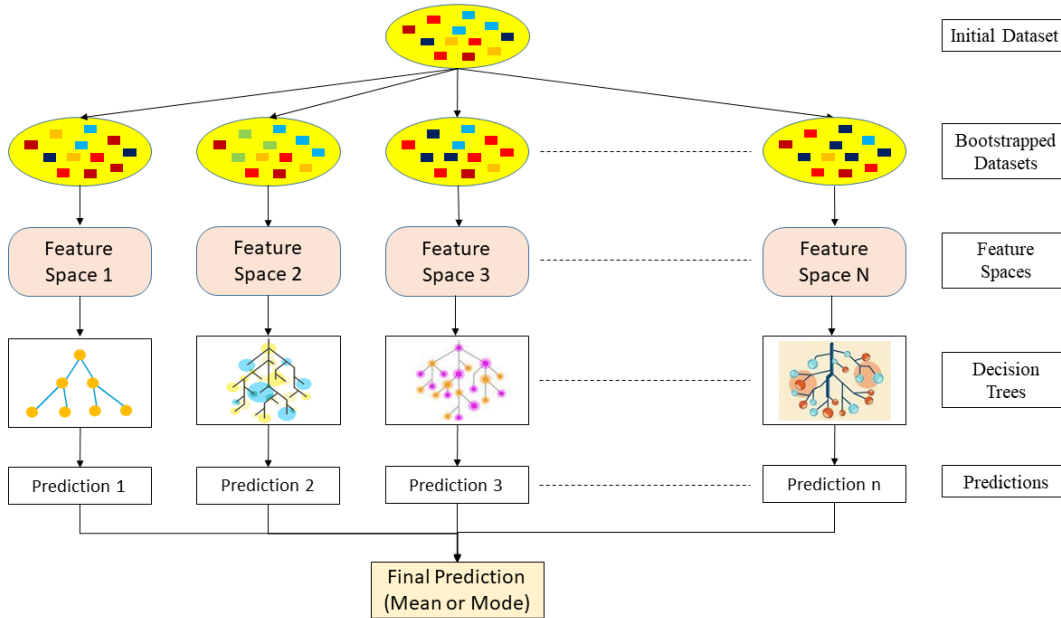


Figure. Algorithm for Random Forest



# RANDOM FOREST: KEY BENEFITS AND CHALLENGES

## ► Benefits:

- **Reduced risk of overfitting:** Decision trees are prone to overfitting because they try to fit all the samples in the training data too closely. Random Forests with a large number of decision trees reduce the risk of overfitting because the averaging of independent trees lowers the overall variance and prediction error .
- **High accuracy:** Random Forest can handle both regression and classification problems with high accuracy, making it a popular method among data scientists.
- **Robustness:** Random Forest is robust to noise and missing values, which makes it a good choice for data with missing or incomplete values.

## ► Challenges:

- **Time and Resource Demanding:** As Random Forest computes a separate decision tree with different feature space for each subset of data, it takes a lot more time and computation power compared to a simple decision tree.
- **Interpretability:** Even though Random Forest provides a measure of feature importance, prediction of a single decision tree is easier to interpret since it shows the path of the decision-making process for that specific sample.

## RANDOM FOREST: APPLICATIONS

Random Forest, in general, is a good choice when we want a high-performing model with low variance and low bias. It is particularly useful when we have a large number of strongly correlated features, as the feature subsampling helps to decorrelate the models. Although, in instances, when we don't have a large sample-space or feature-space, or we need to find co-dependencies or strong interpretation, it is more useful to use a simpler algorithm such as decision tree or support vector machine. Nevertheless, Random Forest is one of the most powerful machine learning algorithms we have and it's been used in several complicated real life problems.

- ▶ Random Forest is used in the **banking and finance** industry for tasks such as credit risk analysis, fraud detection, and loan approval processes.
- ▶ In **e-commerce**, it is used for tasks such as customer segmentation, personalized product recommendations, and fraud detection.
- ▶ Random Forest is used in **image and speech recognition** applications, such as facial recognition, voice recognition, and emotion detection.
- ▶ Random Forest is used in **social media analysis** for tasks such as sentiment analysis, predicting user behavior, and recommending content.
- ▶ In **healthcare**, Random Forest can be used for tasks such as predicting disease diagnoses, identifying high-risk patients, and determining treatment plans.

# BOOSTING

## DEFINITION AND IDEA

- ▶ Boosting is an ensemble learning method that combines a set of weak learners into a strong learner to minimize training errors.
- ▶ In boosting, a random sample of data is chosen, fitted with a model, and then sequentially trained. In other words, each model aims to make up for the shortcomings of the one before it.
- ▶ Intuitively, we do not have a super learner, but many bad learners. These bad learners are combined to obtain a strong learner with lower bias.
- ▶ So, Boosting is used to shift the models from **high** bias  $\rightarrow$  **low** bias.

# BOOSTING

## GRADIENT BOOSTING

- ▶ Boosting algorithms can differ in how they create and aggregate weak learners during the sequential process.
- ▶ There are three popular types of boosting: AdaBoost, Gradient Boosting and XGBoost.
- ▶ The one discussed in the class is **Gradient Boosting**. It works by sequentially adding predictors to an ensemble, each correcting for its predecessor's errors. Each predictor is trained on the residual errors of the previous predictor.
- ▶ When the target column is continuous, we use **Gradient Boosting Regressor**; when it is a classification problem, we use **Gradient Boosting Classifier**. The difference between the two is the *Loss function* used.

# GRADIENT BOOSTING REGRESSOR

## ALGORITHM

- ▶ Let us consider a decision tree with regression. We will choose an ensemble consists of  $N$  decision trees (DT's). The  $N$  **here is a hyperparameter**.
- ▶ Here, Tree-1 is trained using the feature matrix  $X$  and the labels  $y$ . The predictions labelled  $\hat{y}_1$  are used to determine the training set residual errors  $r_1$ .
- ▶ Tree-2 is then trained using the feature matrix  $X$  and the residual errors  $r_1$  of Tree-1 as labels. The predicted results  $\hat{r}_1$  are then used to determine the residual  $r_2$ .
- ▶ The process is repeated until all the ensemble's  $N$  trees are trained.

# GRADIENT BOOSTING REGRESSOR

ALGORITHM: CONTINUED

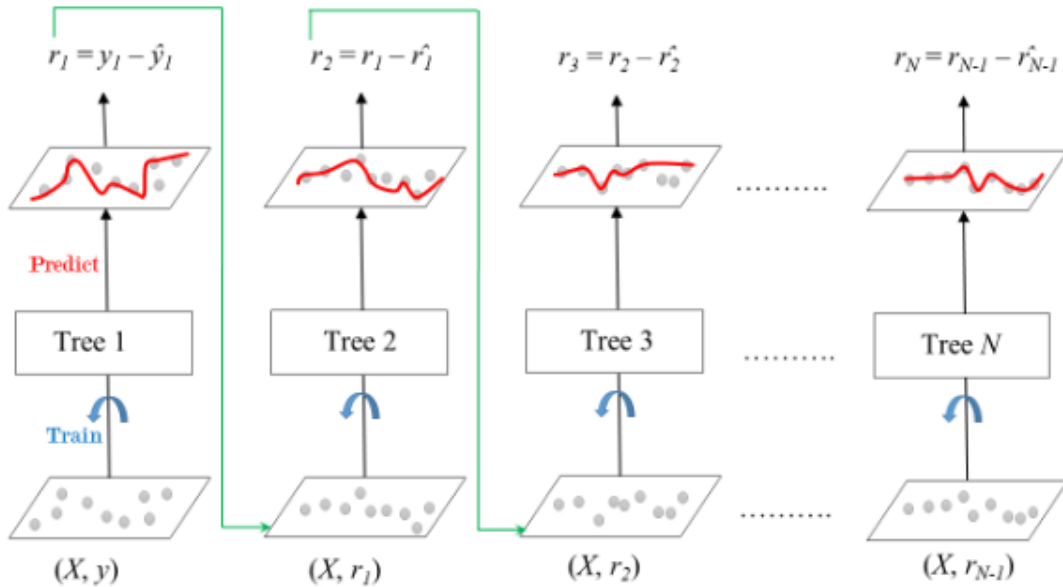


Figure. Gradient Boosted Trees for Regression

# GRADIENT BOOSTING REGRESSOR

## SHRINKAGE

- ▶ A learning rate  $\eta$ , which ranges from  $(0, 1)$ , is being introduced, which will be multiplied with the prediction of each tree in the ensemble. This process is known as Shrinkage.
- ▶ Such shrinking of the prediction is introduced to avoid over-fitting, and thus  $\eta$  can be considered a damping factor.
- ▶ There is a trade-off between  $\eta$  and  $\mathbf{N}$  (the number of DT's / estimators). Decreasing learning rate ( $\eta \downarrow$ ) needs to be compensated with increasing estimators ( $\mathbf{N} \uparrow$ ) to reach specific model performance.

Each tree predicts labels, and this formula gives the final prediction,

$$y_{pred} = \hat{y}_1 + \eta_1 r_1 + \eta_2 r_2 + \dots + \eta_N r_N \quad (1)$$

where  $\eta_i$  are the learning rate for the estimators  $DT_i$  with residual error  $r_i$ .

# GRADIENT BOOSTING REGRESSOR

## EXAMPLE

### Example 2.1

Suppose, we were trying to predict the price of a house given their age, square footage and location.

**Table.** Price of the house

Age	Square Footage	Location	Price
5	1500	5	480
11	2030	12	1090
14	1442	6	350
8	2501	4	1310
12	1300	9	400

- ▶ **Step 1:** A decision tree  $DT_0$  with a leaf is created, the predicted value for the variable of interest. Here, it is the price of the house. Then this leaf is made as the baseline to approach the correct solution sequentially.



# GRADIENT BOOSTING REGRESSOR

## EXAMPLE-CONTINUED

- ▶ **Step 2:** The residual  $r_i$  is calculated for every sample.  
Residual ( $r$ ) = Actual Value( $\hat{y}$ ) - Predicted Value( $y_{pred}$ )

**Table.** GradBoost  $DT_1$

Age	Square Footage	Location	Price	Predicted $\hat{y}_1$	Residuals $r_1$
5	1500	5	480	440	40
11	2030	12	1090	880	210
14	1442	6	350	400	-50
8	2501	4	1310	1010	300
12	1300	9	400	600	-200

- ▶ **Step 3:** We then build a decision tree ( $DT_1$ ) that makes predictions on the residuals  $r_1$  and we obtain  $\hat{y}_2$ .
- ▶ **Step 4:** The process is continued by developing  $N$  decision trees and finally attaining the predicted value for each sample with suitable weightage as a learning parameter, as mentioned in 1.
- ▶ So, the final prediction of the price of the house  $y_{pred}$  is  
$$y_{pred} = \text{Price predicted from } DT_0 + \eta_1 * \text{Residual}(r_1) \text{ from } DT_1 + \dots + \eta_N * \text{Residual}(r_N) \text{ from } DT_N$$
where  $\eta$  are the learning rate.

# GRADIENT BOOSTING REGRESSOR

## EXAMPLE-CONTINUED

**Table.** Boosting  $DT_2$

Age	Square Footage	Location	Price	Predicted $\hat{y}_1$	Residuals $r_1$	Predicted $\hat{y}_2$	Residuals $r_2$
5	1500	5	480	440	40	20	20
11	2030	12	1090	880	210	200	10
14	1442	6	350	400	-50	0	-50
8	2501	4	1310	1010	300	240	60
12	1300	9	400	600	-200	-50	-150

**Table.** Boosting final prediction after  $N$  Decision Trees

Age	Square Footage	Location	Price	Predicted $\hat{y}_1$	Residuals $r_1$	Residuals $r_2$	...	Price Prediction $y_{pred}$
5	1500	5	480	440	40	20		478
11	2030	12	1090	880	210	10		1091
14	1442	6	350	400	-50	-50		335
8	2501	4	1310	1010	300	60		1311
12	1300	9	400	600	-200	-150		400

# BOOSTING: KEY BENEFITS AND CHALLENGES

## ► Benefits:

- **Reduction of bias:** Boosting algorithms sequentially combine several weak learners, iteratively improving on observations. This method can aid in lowering high bias, which is frequently present in logistic regression models and shallow decision trees.
- **Computational Efficiency:** Boosting algorithms can aid in reducing dimensionality and improving computational efficiency because they only choose features during training that improve their predictive power.

## ► Challenges:

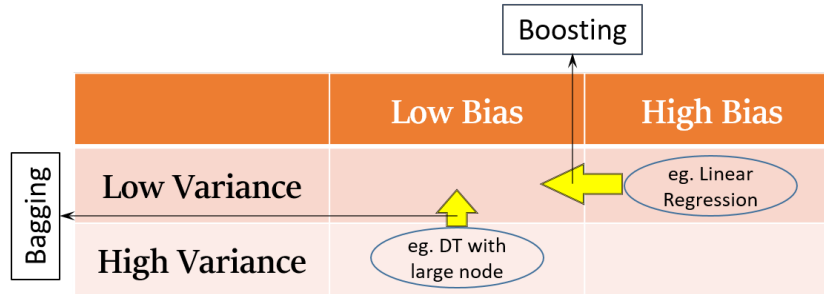
- **Overfitting:** There has been evidences boosting sometimes lead to overfitting.
- **Intense computation:** Scaling sequential training in boosting is challenging. Boosting models can be computationally expensive because each estimator builds on its predecessors.

## BOOSTING: APPLICATIONS

- ▶ When **predicting medical data**, such as cardiovascular risk factors and cancer patient survival rates, boosting is used to reduce prediction errors.
- ▶ The Viola-Jones boosting algorithm is used for **image retrieval**, while search engines use gradient boosted regression trees for **page rankings**.
- ▶ Deep learning models and boosting are combined to automate important tasks like **fraud detection, pricing analysis**, and more.

# BAGGING VS BOOSTING

## SUMMARY









### Bagging

- ▶ Used on weak learners that exhibit high variance and low bias (HVLB)
- ▶ The weak learners are trained in parallel
- ▶ Each learner receives equal weight.
- ▶ Bagging can be used to avoid overfitting.

### Boosting

- ▶ Used on weak learners that exhibit low variance and high bias (LVHB)
- ▶ The weak Learners are learned Sequentially.
- ▶ Each learner is weighted differently according to their performance.
- ▶ Boosting can be used to lower bias.

## REFERENCES I

-  *A Gentle Introduction to Ensemble Learning Algorithms* (Apr. 2021). <https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/>. Accessed: 2023-02-25.
-  Josh Starmer, StatQuest with (2019). *Gradient Boost*. <https://www.youtube.com/watch?v=3CC4N4z3GJc>.
-  *ML – Gradient Boosting* (2020). <https://www.geeksforgeeks.org/ml-gradient-boosting/>.
-  *What is bagging?* (2020). <https://www.ibm.com/in-en/topics/bagging>. Accessed: 2023-02-25.
-  *What is Boosting?* (2020). <https://www.ibm.com/in-en/topics/boosting>. Accessed: 2023-02-25.
-  *What is Random Forest?* (2020). <https://www.ibm.com/in-en/topics/random-forest>. Accessed: 2023-03-01.