Gradient Boosting

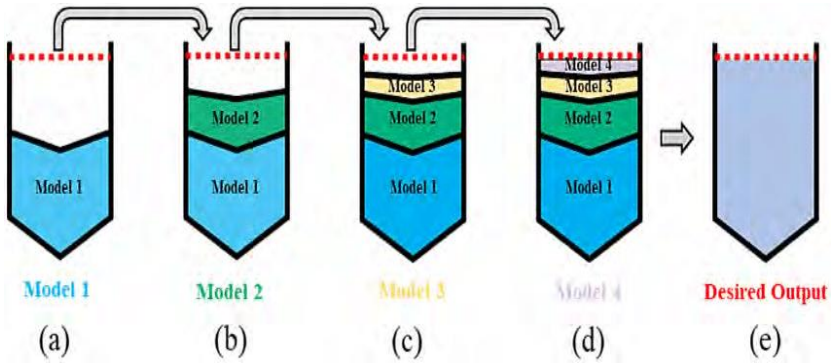
Presented by: Muhammad Zaqeem

Overview:

- Gradient Boosting
- How it Works
- Mathematical Intuition of Gradient Boosting (Regression)
- Mathematical Intuition Gradient Boosting (Classification)
- Hyperparameter in Gradient Boosting
- Gradient Boosting vs Bagging(Random Forest)
- Advantage of Gradient Boosting

Gradient Boosting:

- Gradient Boosting is an ensemble machine learning technique primarily used for regression and classification problems.
- It builds models in a sequence, with each model focusing on correcting the errors made by the previous models.
- Gradient boosting methods like XGBoost and LightGBM are popular in machine learning competitions (e.g., Kaggle) due to their high performance.
- builds a model by combining multiple weak models, typically decision trees, in a sequential manner.
- Used extensively in business, finance, healthcare, and more to make predictions about future events based on historical data.



How it Works:

Start with a simple model: The first model is usually a very basic prediction. For **regression**, this is often just the mean of the target variable (average of all target values). For **classification**, it can be the log-odds of the target class.

Compute the Residuals:

- **Residuals** represent the difference between the true values (target) and the predictions made by the model.

Residual = Actual value - Predicted value.

Train a weak learner (typically a decision tree): To predict these residuals. This means that the new model is not trying to predict the target directly, but instead learning to predict the errors or the discrepancies between the actual and predicted values.

The tree's job is to model the residuals, i.e., correct the errors made by the previous model.

Continue…

Add the New Model's Predictions to the Existing Model:

- The predictions from the new model are added to the previous model's predictions to improve the overall prediction.

$$F_m(x) = F_{m-1}(x) + \nu_m h_m(x)$$

Repeat the Process (Iterate).

- This process is repeated for a fixed number of iterations (or until some stopping condition is met, like a desired accuracy or minimum error).

Update the model:

$$F_m(x) = F_{m-1}(x) + \nu_m h_m(x)$$

Final Model:

- After the desired number of models (trees) have been added, the final model is a weighted combination of all the trees. The predictions are more accurate as each tree has learned to correct the errors from the previous models.
- The model is now trained and can be used to make predictions on new, unseen data.

Mathematical Intuition in Regression:

Build a Base Model:

- The first step in gradient boosting is to build a base model to predict the observations in the training dataset.
- In regression, this could be the average of all target values.

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

$$L = (y_i - \gamma)^2$$

$$\frac{\partial}{\partial \gamma} \sum_{i=1}^{n} L = \frac{\partial}{\partial \gamma} \sum_{i=1}^{n} (y_i - \gamma)^2$$

$$= -2 \sum_{i=1}^{n} (y_i - \gamma)$$

$$= -2 \sum_{i=1}^{n} y_i + 2n\gamma$$

$$-2 \sum_{i=1}^{n} y_i + 2n\gamma = 0$$

$$n\gamma = \sum_{i=1}^{n} y_i$$

$$\gamma = \frac{1}{n} \sum_{i=1}^{n} y_i = \bar{y}$$

Iterative Training (For m=1to M)

- The main part of Gradient Boosting involves multiple iterations. In each iteration, we improve the model by adding a new tree that corrects the errors of the previous model.

Continue…

- **Compute Residuals:** it represent the **errors** from the previous model (the difference between the actual and predicted values).

2-1. Compute residuals $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ $for\ i = 1,\ldots, n$

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$
$$= -\frac{\partial (y_i - F_{m-1})^2}{\partial F_{m-1}}$$
$$= 2(y_i - F_{m-1})$$

2-2. Train regression tree with features x against r and create terminal node reasions R_{jm} for $j = 1,\ldots, J_m$

j represents a terminal node (i.e. leave) in the tree, m denotes the tree index, and capital J means the total number of leaves

Continue…

2-3. Compute $\gamma_{jm} = \underset{\gamma}{argmin} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma) \quad for\ j = 1,\ldots, J_m$

- We are searching for $\gamma_j m$ that minimizes the loss function on each terminal node j. $\Sigma x_i \in R_j m$ L means we are aggregating the loss on all the sample x_is that belong to the terminal node $R_j m$.

2-4. Update the model:

$$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} 1(x \in R_{jm})$$

Mathematical Intuition in Classification:

Gathering and Analysing Our Data: In the table we are using the **training data** that we have gathered from six patients. The data shows patients' presence of chest pain, their pulse (beats per minute), weight (underweight, normal, and overweight), and a history of heart disease. Our aim here is to understand how **gradient boost** fits a model to this training data.

Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease	Actual	Residual
No	No	No	No	0	-0.7
Yes	Yes	Yes	Yes	1	0.3
Yes	Yes	No	Yes	1	0.3
Yes	No	No	Yes	1	0.3
Yes	No	Yes	Yes	1	0.3
No	Yes	No	No	0	-0.7

Odds and Probability Calculating:

- Using **gradient boost for classification** we discover the initial prediction for every patient in the **log (odds)**.
- Then convert the **log (odds)** to **probability**. The trick here is to use the **logistic function**.

$$ln(odds) = ln\left(\frac{numberOf\ YesHeartDisease}{numberOf\ NoHeartDisease}\right) = ln\left(\frac{4}{2}\right) = 0.69$$

$$F_0(x) = \text{logit}\left(\frac{p}{1-p}\right) = \log\left(\frac{p}{1-p}\right)$$

$$probability = \frac{odds}{1+odds} = \frac{e^{ln(odds)}}{1+e^{ln(odds)}} = \frac{e^{0.69}}{1+e^{0.69}} = 0.67$$

Compute the Pseudo-Residuals:

The residual r_{im} at iteration m is given by the negative gradient of the loss function:

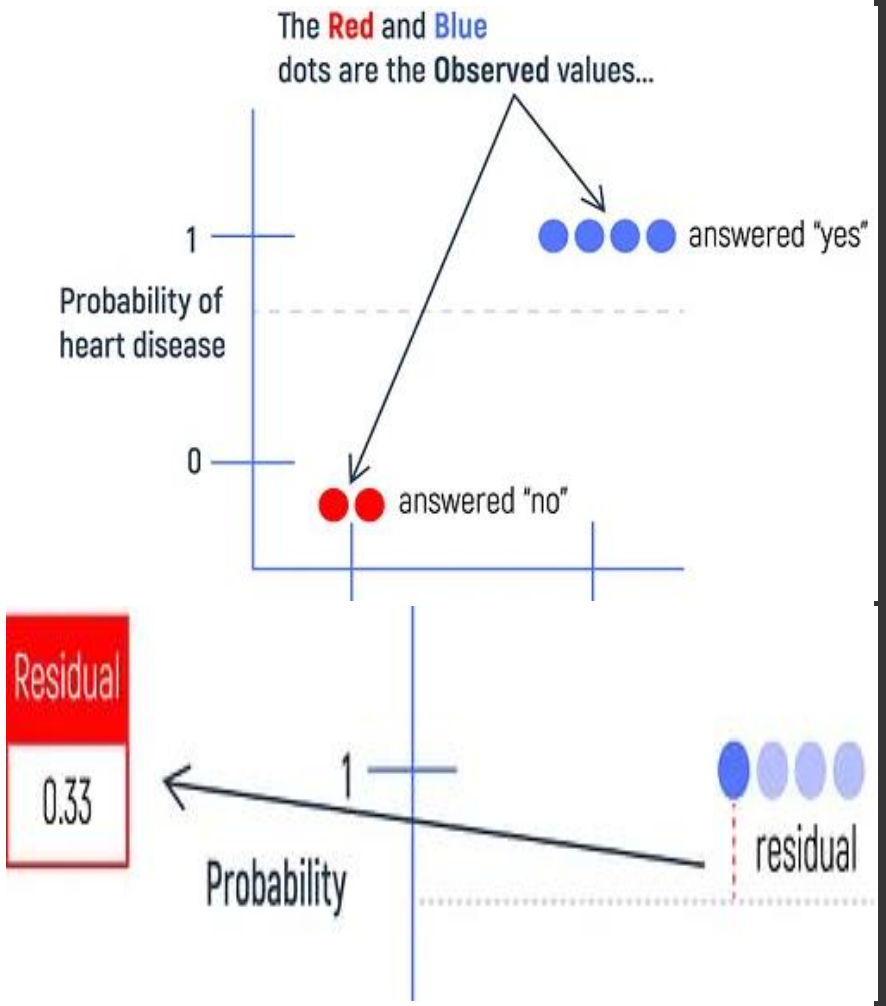
$$r_{im} = -\frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}$$

- For logistic loss,

$$r_{im} = y_i - p_i$$

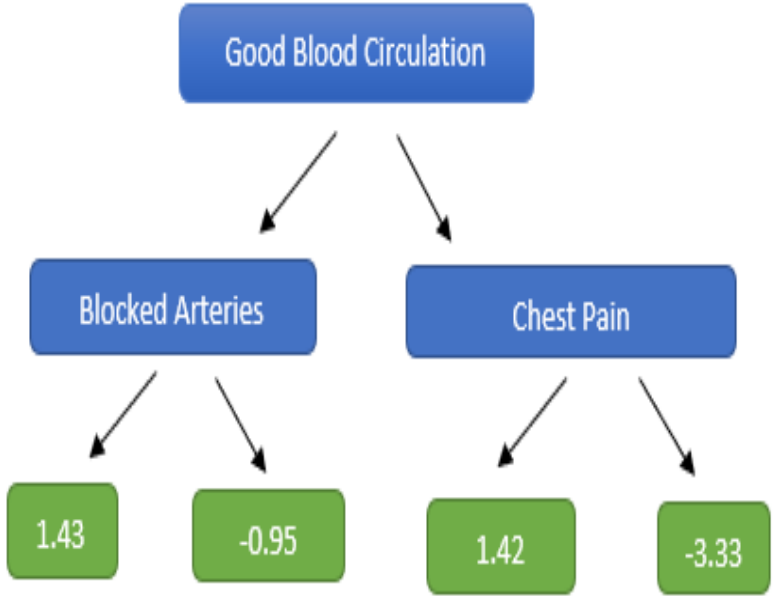
- where pi is the predicted probability of i at iteration.

$$p_i = \frac{1}{1 + \exp(-F_{m-1}(x_i))}$$



Predicting Residuals with a Decision Tree

- Next, we build a decision tree that tries to predict these residuals based on features (like "Chest Pain" or “Pluse").
- This tree will split data points into groups based on the features, trying to capture patterns in the residuals.
- The initial prediction was in terms of log(odds) and the leaves are derived from a probability. Hence, we need to do some transformation to get the predicted residuals in terms of log(odds). The most common transformation is done using the following formula



$$\frac{\sum Residual_i}{\sum[Previous\ Probability_i \times (1 - Previous\ Probability_i)]}$$

$$\frac{0.3}{0.7 \times (1-0.7)} \approx 1.43$$

Update the model

- The model is updated by adding the output of the tree to the previous prediction.

$$F_m(x) = F_{m-1}(x) + \nu_m h_m(x)$$

$$\textit{Initial prediction} + \textit{Learning Rate} \times \textit{Predicted Residual}$$

$$= 0.7 + (0.2 \times 1.43) = 0.99$$

$$\frac{e^{\log(odds)}}{1+e^{\log(odds)}} = \frac{e^{0.99}}{1+e^{0.99}} = 0.7289 \approx 0.73$$

Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease	Actual	New Prediction	New Residual
No	No	No	No	0	0.51	-0.51
Yes	Yes	Yes	Yes	1	0.73	0.27
Yes	Yes	No	Yes	1	0.73	0.27
Yes	No	No	Yes	1	0.72	0.28
Yes	No	Yes	Yes	1	0.72	0.28
No	Yes	No	No	0	0.63	-0.63

- Repeat steps III to V until the residuals converge to a value close to 0 or the number of iterations matches the value given as hyperparameter while running the algorithm.

Hyperparameter in Gradient Boosting:

Boosting vs Bagging:

Use Gradient Boosting if:

- **Accuracy is Crucial**: If you need the highest possible accuracy, Gradient Boosting generally outperforms Random Forests when carefully tuned.
- **You Have Imbalanced Data**: Gradient Boosting is more effective for handling imbalanced data, as it pays more attention to difficult examples.
- **You Have Time for Hyperparameter Tuning**: Gradient Boosting requires careful tuning, so if you have the time and computational resources to optimize your model, it can yield better results.
- **Complex Problem Spaces**: If you're working with complex data where each decision depends on previous ones, Gradient Boosting can capture subtle patterns better than Random Forest.

Use Random Forest if:

- **You Need a Fast and Robust Model**: Random Forests are great for quickly building strong baseline models with minimal tuning.
- **You Have a Large Dataset**: Random Forest is highly scalable and can handle large datasets with ease.
- **Interpretability is Important**: If interpretability is a priority, Random Forests offer a simpler structure than Gradient Boosting, especially for smaller forests.
- **You Need to Avoid Overfitting**: Random Forests are less prone to overfitting due to bagging, making them ideal for datasets with noisy features.

Advantages

- Gradient Boosting typically achieves higher accuracy than Random Forests and other ensemble methods because it's designed to continuously minimize prediction errors.
- Gradient Boosting allows for fine-grained control over model complexity through hyperparameters like the learning rate, the number of iterations (trees), and the maximum depth of each tree.
- Gradient Boosting can be adapted to handle imbalanced datasets (where one class is much more common than another) by adjusting the weights of misclassified samples.
- Gradient Boosting has built-in regularization techniques, like the learning rate and shrinkage (scaling of tree outputs), which help reduce overfitting.

Thank you