XGBOOST

BYD2060

Jiayi Zhou, Yanshu Li, Ziqi Zhang, Xinye Yang

01

Overview

Introduction to XGBoost

- Extreme Gradient Boosting
- Efficient and widely used ML algorithm for classification & regression.
- Key idea: Ensemble of weak learners (decision trees).

Objective Function

$$\text{Obj} = \sum L(y_i, \hat{y}_i) + \sum \Omega(f_k)$$

- $L(y_i, \hat{y}_i)$: Loss function.
- $\Omega(f_k)$: Regularization term controlling complexity.

Taylor Expansion for Optimization

$$L(y_i, \hat{y}_i + f_t(x_i)) \approx L(y_i, \hat{y}_i) + g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2$$

- $g_i = \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}$: First-order gradient of the loss.
- $h_i = \frac{\partial^2 L(y_i, \hat{y}_i)}{\partial \hat{y}_i^2}$: Second-order gradient (Hessian).
- $f_t(x_i)$: The output of the current tree (weak learner) for the input xi.
- $\hat{y}_i + f_t(x_i)$: The updated prediction.

Structure of Tree Models

$$f(x) = w_{q(x)}, \quad \Omega(f) = \gamma N + \frac{1}{2}\lambda \sum_{j=1}^{N} w_j^2$$

- $q(x)$: Maps input x to a leaf.
- $w_{q(x)}$: Weight of the corresponding leaf.
- N: Number of leaf nodes.

Optimal Leaf Weights and Gain

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

$$\text{Gain} = \frac{1}{2} \sum_{j=1}^{N} \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_j} h_i + \lambda} - \gamma N$$

- **Gain**: Improvement in split quality.
- λ, γ: Regularization terms for leaf weights and control the complexity of the tree.

Prevent Overfitting

1. **Shrinkage**: Applies a learning rate η to scale the contribution of each tree.
2. **Column Subsampling**: Randomly selects subsets of features for each tree to increase diversity and reduce variance.

Advantages & Applications

- **Speed**: Parallelized computations.
- **Accuracy**: Regularization and second-order optimization.
- **Scalability**: Handles large datasets efficiently.
- **Flexibility**: Supports custom loss functions.

- **Competitions**: Kaggle and ML challenges.
- **Finance**: Fraud detection, credit scoring.
- **Healthcare**: Disease prediction, risk stratification.
- **E-commerce**: Recommendations, click-through prediction.

02

Implementation

Build Decision Tree

Algorithm 1 Build Decision Tree

Input:
Training data $X \in \mathbb{R}^{m \times n}$
Gradient $g \in \mathbb{R}^m$
Hessian $h \in \mathbb{R}^m$
Parameters λ, γ
Function BuildTree(X, g, h, depth):
if depth = max_depth or n_samples < min_samples **then**
    return $w = -\frac{\sum g_i}{\sum h_i + \lambda}$
end if
for each feature j and split value s **do**
    $I_L \leftarrow \{i | x_{ij} < s\}$
    $I_R \leftarrow \{i | x_{ij} \geq s\}$
    $G_L \leftarrow \sum_{i \in I_L} g_i$, $G_R \leftarrow \sum_{i \in I_R} g_i$
    $H_L \leftarrow \sum_{i \in I_L} h_i$, $H_R \leftarrow \sum_{i \in I_R} h_i$
    Gain $\leftarrow \frac{1}{2}[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}] - \gamma$
    Update best split if Gain is larger
end for
if best_gain > 0 **then**
    Split data using (j^*, s^*)
    left_tree $\leftarrow$ BuildTree(X_{I_L}, g_{I_L}, h_{I_L}, depth + 1)
    right_tree $\leftarrow$ BuildTree(X_{I_R}, g_{I_R}, h_{I_R}, depth + 1)
    return Node with split (j^*, s^*) and subtrees
else
    return $w = -\frac{\sum g_i}{\sum h_i + \lambda}$
end if

The base case for recursion - when reaching maximum depth or minimum required samples, return the optimal weight for leaf node.

Divide the data into left and right subsets based on the split value, then calculate the aggregate gradient statistics and Hessian statistics.

Calculate the gain of the split using first and second order gradients, incorporating L2 regularization and minimum split gain threshold.

Recursively build the left and right subtrees using the split data and updated depth, forming the complete tree structure.

XGBoost

Algorithm 2 XGBoost Training

Input:
Training set $S = \{(x_1, y_1), ..., (x_m, y_m)\}$
Number of trees T
Learning rate η
Initialize $\hat{y}_i^{(0)} = 0$ for all i
for $t = 1$ to T **do**
  Calculate gradients:
  $g_i \leftarrow \hat{y}_i^{(t-1)} - y_i$
  $h_i \leftarrow 1$
  Build tree f_t using BuildTree(X, g, h, 0)
  Update predictions:
  $\hat{y}_i^{(t)} \leftarrow \hat{y}_i^{(t-1)} + \eta f_t(x_i)$
end for
Final model:
$f(x) = \sum_{t=1}^{T} \eta f_t(x)$
$P(y = 1|x) = \frac{1}{1+e^{-f(x)}}$

Calculate the first and second order derivatives for the current iteration, where the gradient is the difference between current prediction and true label.

Build a new decision tree using the computed gradients and update the model's predictions.

03

Previous Work

Previous Work

- Breast Cancer Diagnosis
- Customer Churn Prediction
- Titanic Survival Prediction

Model Parameters

- Number of trees = 5
  - Balance between accuracy and efficiency
- Maximum depth = 5
  - Avoid overfitting
- Learning rate = 0.3
  - balance between convergence speed and stability

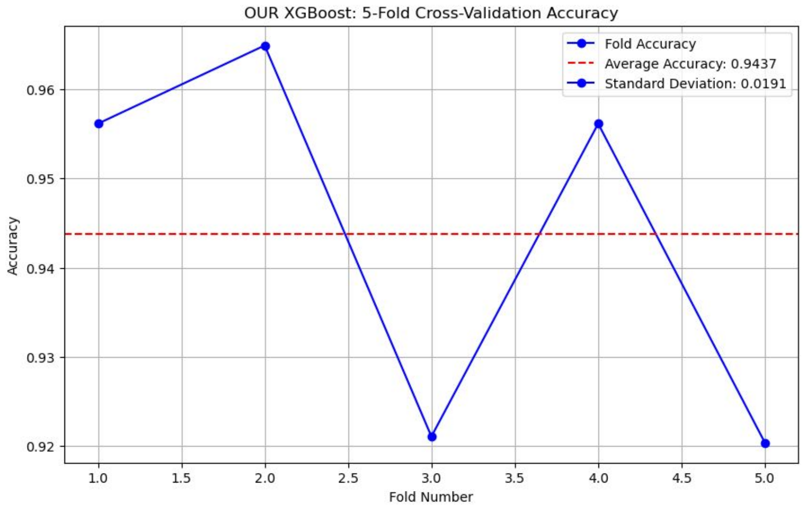
Breast Cancer Diagnosis

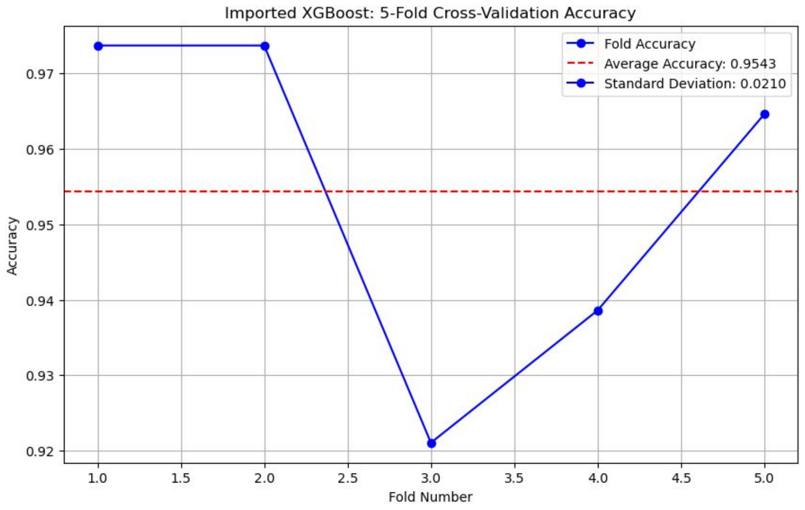
- diagnose whether a breast mass is benign or malignant.

Feature	Description
ID number	Identifier for each case
Diagnosis	M = malignant, B = benign
radius	Mean of distances from center to points on the perimeter
texture	Standard deviation of gray-scale values
perimeter	Perimeter of the contour
area	Area within the contour
smoothness	Local variation in radius lengths
compactness	Perimeter^2 / Area - 1.0
concavity	Severity of concave portions of the contour
concave points	Number of concave portions of the contour
symmetry	Symmetry of the contour
fractal dimension	"Coastline approximation" - 1

Photo by National Cancer Institute on Unsplash

Breast Cancer Diagnosis





Our work

- Average Accuracy: **94.37%**
- Standard Deviation: 1.91%

Previous work

- Average Accuracy: **95.43%**
- Standard Deviation: 2.10%

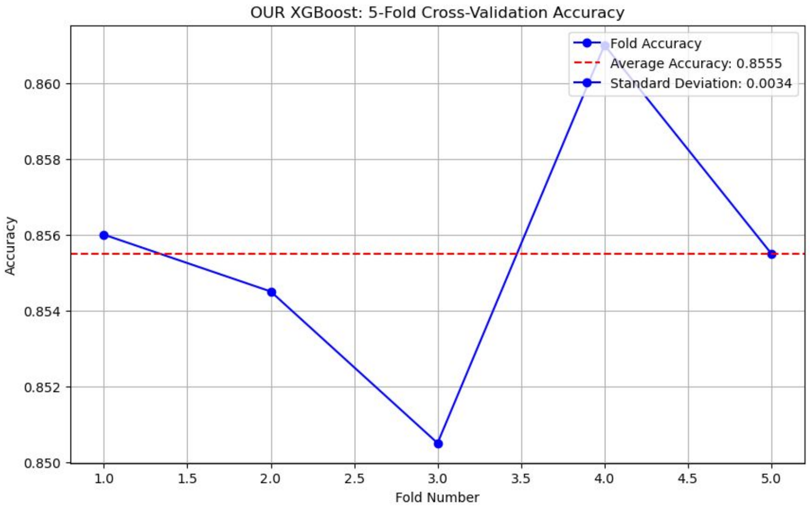
Customer Churn Prediction

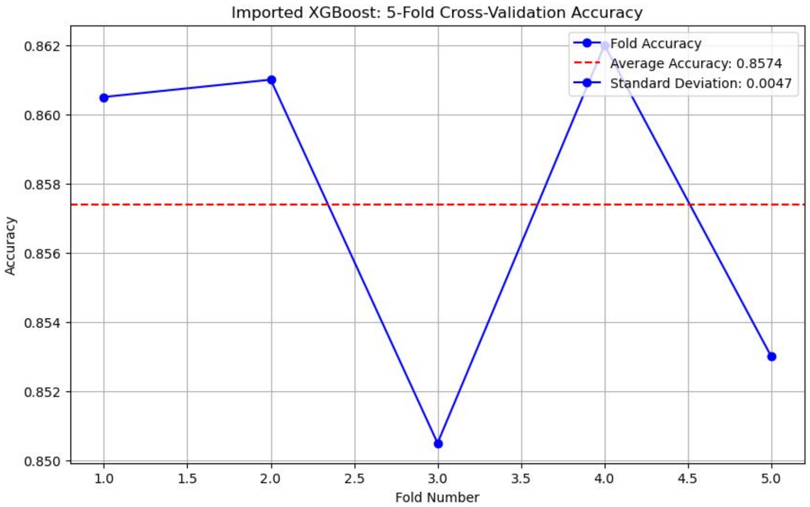
- Predicts customer churn based on customer information.

Feature	Description
RowNumber	The row number in the dataset (used as an index).
CustomerId	A unique identifier for each customer.
Surname	The last name of the customer.
CreditScore	The credit score of the customer.
Geography	The location or country of the customer.
Gender	The gender of the customer (e.g., Male, Female).
Age	The age of the customer.
Tenure	The number of years the customer has been with the provider.
Balance	The account balance of the customer.
NumOfProducts	The number of products the customer has with the provider.
HasCrCard	Whether the customer has a credit card (1 = Yes, 0 = No).
IsActiveMember	Whether the customer is an active member (1 = Yes, 0 = No).
EstimatedSalary	The estimated salary of the customer.
Exited	Whether the customer exited the bank (1 = Yes, 0 = No).

Photo by Patrick Tomasso on Unsplash

Customer Churn Prediction





Our work

- Average Accuracy: **85.55%**
- Standard Deviation: 0.34%

Previous work

- Average Accuracy: **85.74%**
- Standard Deviation: 0.47%

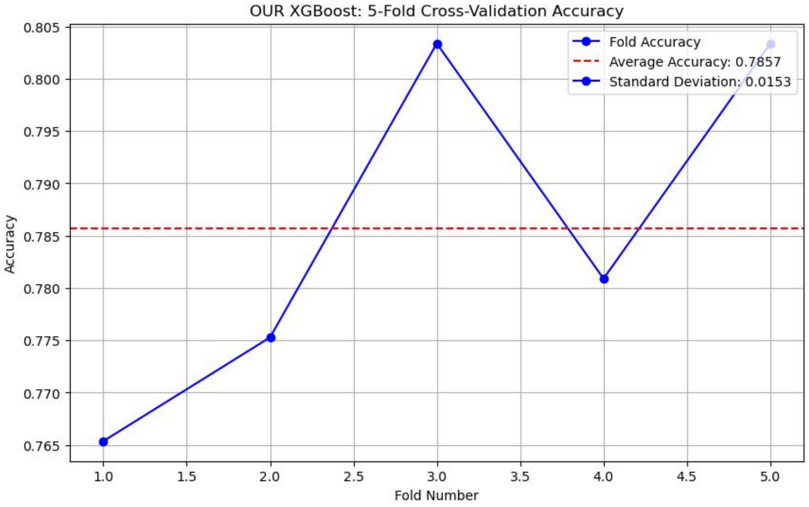
Titanic Survival Prediction

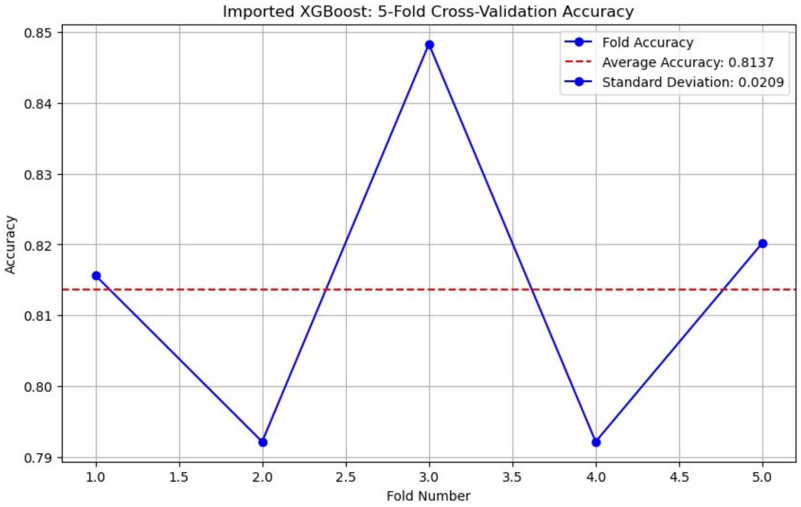
- Predict which passengers would survive the Titanic shipwreck.

Feature	Description
PassengerId	Unique identifier for each Passenger.
Survival	Survival status (0 = No, 1 = Yes).
Pclass	Ticket class (1 = 1st, 2 = 2nd, 3 = 3rd).
Sex	Sex of the passenger.
Age	Age of the passenger in years.
Sibsp	Number of siblings/spouses aboard the Titanic.
Parch	Number of parents/children aboard the Titanic.
Ticket	Ticket number.
Fare	Passenger fare.
Cabin	Cabin number.
Embarked	Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton).

Photo by Torsten Dederichs on Unsplash

Titanic Survival Prediction





Our work

- Average Accuracy: **78.57%**
- Standard Deviation: 1.53%

Previous work

- Average Accuracy: **81.37%**
- Standard Deviation: 2.09%

04

Summary

Highlights

Two classes

Class Decision Tree

Class XGBoost

1. **Clear design:** Separate modules for decision tree and XGBoost ensure clear responsibilities and easy maintenance.
2. **Detailed implementation:** Incorporates gradient and Hessian calculations, supports cross-entropy loss, and uses recursion for tree building.
3. **Strong scalability:** Designed for easy extension to other loss functions and optimization methods.

Challenges

1. **Robust XGBoost:** Handles suboptimal data distributions, but dataset quality remains critical for achieving reliable and accurate predictions.

2. **Hyperparameter Dependency of Model:** Model performance is highly dependent on hyperparameters like tree depth and split size, which directly affect gain function regularization.

Solutions

1. **Preprocessing and Pruning:** Apply preprocessing to enhance dataset quality and implement adaptive pruning strategies for dynamic parameter adjustment during training.

2. **Joint Optimization:** Coordinate hyperparameter tuning to balance complexity and generalization, ensuring strong performance on both training and unseen data.

Analysis of the performance gaps

1. **Split Function:** Pre-implemented XGBoost optimizes splits with a histogram algorithm, while manual methods with linear scans lead to relatively inaccurate splits and lower performance.
2. **High-precision calculations:** Pre-implemented XGBoost avoids floating-point errors affecting gradients and optimization.
3. **Diverse tree growth strategies:** Pre-implemented XGBoost supports diverse tree growth strategies, unlike fixed strategies in manual implementation.

Questions?

THANK YOU

BYD2060