XGBoost for Classification

Overview:
XGBoost stands for Extreme Gradient Boosting. The term *"extreme"* refers to a series of advanced optimizations incorporated into the model to enhance both performance and computational speed.
Consider a classification problem where the dataset includes a single input feature, CGPA, and an output label Placed (1 if placed, 0 otherwise). Our goal is to build an XGBoost classification model that predicts whether a new student will be placed based on their CGPA.

CGPA	Placed
5.70	0
6.25	1
7.10	0
8.15	1
9.60	1

XGBoost follows the same foundational principles as Gradient Boosting. The core idea involves stage-wise additive modeling, where multiple weak learners (typically decision trees) are combined to form a strong model. Initially, a base prediction is made; then, residual errors (mistakes) are calculated and passed to the next model. This process continues iteratively, with each new model aiming to correct the errors made by the ensemble so far.
In classification, the base prediction starts not from the mean (as in regression), but from the log-odds of the target. Based on these log-odds, initial predictions are made, and the corresponding errors are computed. A decision tree is then trained on these errors (gradients), and the model is updated. This iterative process continues until the overall prediction error is minimized.

While XGBoost and traditional Gradient Boosting share a similar workflow, the primary difference lies in how the decision trees are constructed:

- In Gradient Boosting, trees are built using metrics like Gini Impurity or Entropy (for classification tasks).
- In XGBoost, trees are constructed using a similarity score (also called gain), which measures how much a split improves the performance based on the gradients and Hessians of the loss function.

Thus, although the overall boosting methodology is consistent, the internal mechanism of tree building and optimization is more advanced in XGBoost, leading to better efficiency and accuracy.

Step-by-Step Solution: XGBoost for Classification

Stage 1: Base Estimator (Initial Prediction)

In the first stage, we begin with a base estimator, which serves as the initial prediction for all data points. In XGBoost for classification, this base prediction is made using the log-odds of the positive class.
The formula for log-odds is:
log odds = $log\left(\frac{p}{1-p}\right)$
Where:

- p is the probability of the positive class (i.e., class = 1)

Given the dataset:

CGPA	Placed
5.70	0
6.25	1
7.10	0
8.15	1
9.60	1

There are 3 positive labels out of 5 samples, so:
p = 3/5 , 1-p = 2/5

log odds = $log\left(\frac{\frac{3}{5}}{\frac{2}{5}}\right) = \log\left(\frac{3}{2}\right) = 0.405$

So, the initial log-odds prediction for all samples is:

CGPA	Placed	pred1 (log odds)
5.70	0	0.405
6.25	1	0.405
7.10	0	0.405
8.15	1	0.405
9.60	1	0.405

Stage 2: Converting Log-Odds to Probability

To interpret the output, we must convert log-odds into probability using the sigmoid function:

p = $\frac{e^{log\ odds}}{1+e^{log\ odds}} = \frac{e^{0.405}}{1+e^{0.405}} \approx 0.60$

Hence, the probability prediction for all instances is 0.6 :

CGPA	Placed	pred1 (log odds)	pred1 (prob)
5.70	0	0.405	0.6
6.25	1	0.405	0.6
7.10	0	0.405	0.6
8.15	1	0.405	0.6
9.60	1	0.405	0.6

At this point, since the threshold for classification is 0.5, all instances would be predicted as class 1 (placed). However, this is just the initial model (Stage 1), and it lacks personalization based on features (like CGPA).

Stage 3: Compute Pseudo-Residuals

Next, we compute the pseudo-residuals, which are the difference between the true label and predicted probability:
Residual (res1) = Placed − pred1 (prob)

CGPA	Placed	pred1 (log odds)	pred1 (prob)	res1
5.70	0	0.405	0.6	-0.6
6.25	1	0.405	0.6	0.4
7.10	0	0.405	0.6	-0.6
8.15	1	0.405	0.6	0.4
9.60	1	0.405	0.6	0.4

These residuals indicate how far the prediction is from the actual value. The goal of the next model is to minimize these residuals.

Stage 4: Fit a Decision Tree to the Residuals

A new decision tree (Model 2) is trained using the input feature CGPA and the pseudo-residuals res1 as the target. This tree attempts to predict the residuals/errors made by the previous model.

- The predicted residuals from this tree are added to the current model's output (in log-odds space).
- This updated model gives a new prediction, and the residuals are recalculated.

- The process is repeated iteratively, gradually reducing the overall error.

The overall goal is to incrementally improve the model by correcting errors at each stage using additive boosting.

How the Decision Tree is Formed in XGBoost (Classification)

To build the decision tree, we follow the below steps using the CGPA feature and corresponding residuals (res1).

Step 1: Sort the Data by CGPA

CGPA	res1
5.70	-0.6
6.25	0.4
7.10	-0.6
8.15	0.4
9.60	0.4

Step 2: Calculate Similarity Score for the Root Node

All residuals are initially placed in one leaf node. The similarity score (SS) for a node is calculated using the formula for classification:

$$SS = \frac{(\sum_{i=1}^{n} residual_i)^2}{\sum_{i=1}^{n} prev\ prob_i\ (1 - prev\ prob_i) + \lambda}$$

Assumptions:

- prev $prob_i$ = 0.6 for all observations (as calculated earlier)
- λ = 0 (no regularization)

Substitute residuals:

residuals = [−0.6,0.4,−0.6,0.4,0.4]

$$SS_{Root} = \frac{(-0.6 + 0.4 - 0.6 + 0.4 + 0.4)^2}{5 \times (0.6 \times 0.4) + 0} = \frac{0^2}{1.2} = 0$$

The similarity score for the root node is 0.

Step 3: Determine Splitting Criteria

We calculate split points as the average of consecutive CGPA values:

- Between 5.70 & 6.25 → (5.70 + 6.25)/2 = 5.97

- Between 6.25 & 7.10 → 6.67
- Between 7.10 & 8.15 → 7.62
- Between 8.15 & 9.60 → 8.87

Step 4: Evaluate All Splits

For each split, calculate the left and right node similarity scores, then the gain:

$$Gain = (SS_L + SS_R) - SS_{Root}$$

Split 1: CGPA < 5.97

- Left: [-0.6]
- Right: [0.4, -0.6, 0.4, 0.4]

$$SS_L = \frac{(-0.6)^2}{0.6 \times 0.4} = \frac{0.36}{0.24} = 1.5$$

$$SS_R = \frac{(0.4 - 0.6 + 0.4 + 0.4)^2}{4 \times 0.24} = \frac{0.6^2}{0.96} = \frac{0.36}{0.96} \approx 0.375$$

Gain = 1.5 + 0.375 − 0 = 1.87

Split 2: CGPA < 6.67

- Left: [-0.6, 0.4]
- Right: [-0.6, 0.4, 0.4]

$$SS_L = \frac{(-0.6 + 0.4)^2}{2 \times 0.24} = \frac{(-0.2)^2}{0.48} = \frac{0.04}{0.48} \approx 0.08$$

$$SS_R = \frac{(-0.6 + 0.4 + 0.4)^2}{3 \times 0.24} = \frac{0.2^2}{0.72} = \frac{0.04}{0.72} \approx 0.05$$

Gain = 0.08 + 0.05 − 0 = 0.13

Split 3: CGPA < 7.62

- Left: [-0.6, 0.4, -0.6]
- Right: [0.4, 0.4]

$$SS_L = \frac{(-0.6 + 0.4 - 0.6)^2}{3 \times 0.24} = \frac{(-0.8)^2}{0.72} = \frac{0.64}{0.72} \approx 0.88$$

$$SS_R = \frac{(0.4 + 0.4)^2}{2 \times 0.24} = \frac{0.8^2}{0.48} = \frac{0.64}{0.48} \approx 1.33$$

Gain = 0.88 + 1.33 − 0 = 2.22

Split 4: CGPA < 8.87

- Left: [-0.6, 0.4, -0.6, 0.4]
- Right: [0.4]

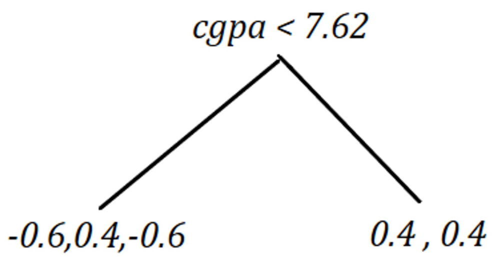
$$SS_L = \frac{(-0.6+0.4-0.6+0.4)^2}{4 \times 0.24} = \frac{(-0.4)^2}{0.96} = \frac{0.16}{0.96} \approx 0.16$$

$$SS_R = \frac{(0.4)^2}{0.24} = \frac{0.16}{0.24} \approx 0.66$$

Gain = 0.16 + 0.66 − 0 = 0.82

Step 5: Select the Best Split

The highest gain is 2.22 at splitting criteria CGPA < 7.62.
Thus, this becomes the root node of our Stage-2 tree.



This is how XGBoost forms the decision tree for classification in Stage 2.

Calculating Output Values for Leaf Nodes in XGBoost (Classification)

To make predictions from the decision tree, we must compute the output values for each leaf node. These outputs are used to update the model in the additive boosting process.

Formula for Output at a Leaf Node

For classification, the output value at a leaf node is given by:

$$output = \frac{\sum_{i=1}^{n} residual_i}{\sum_{i=1}^{n} prev\ prob_i \,.\, (1 - prev\ prob_i) + \lambda}$$

Where:

- $residual_i = y_i - \hat{p}_i$
- $prev\ prob_i = \hat{p}_i$, the prediction probability from the previous stage
- λ = regularization parameter (set to 0 here)

Output Calculation for Each Leaf Node

From the previous tree split (CGPA < 7.62):

- Left node contains residuals: -0.6, 0.4, -0.6

$$Output_{left} = \frac{-0.6+0.4-0.6}{3*0.6*0.4} = -\frac{0.8}{0.72} \approx -1.1111$$

- Right node contains residuals: 0.4, 0.4

$$Output_{right} = \frac{0.4+0.4}{2*0.6*0.4} = \frac{0.8}{0.48} \approx 1.66$$

Update the Combined Model Prediction

We now update our prediction using the outputs of both Stage 1 (log odds) and Stage 2 (decision tree):

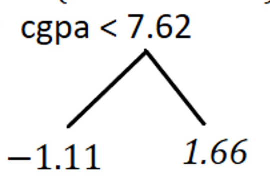
Combined Log Odds = $m_1 + \eta \cdot m_2$

Where:

- m_1 = 0.405 (initial log-odds from stage 1)
- m_2 = leaf output based on CGPA split
- η = 0.3 (learning rate)

$$m1\ (log\ odds)\ +\ 0.3 * m2\ (decision\ tree)$$

$$0.405\ +\ 0.3\ *$$



Final Table: Stage 2 Predictions and Residuals

CGPA	Placed ?	pred1 (log odds)	pred1 (prob)	res1	pred2 (log odds)	pred2 (prob)	res2
5.70	0	0.405	0.600	-0.6	0.072	0.518	-0.51
6.25	1	0.405	0.600	0.4	0.072	0.518	0.48
7.10	0	0.405	0.600	-0.6	0.072	0.518	-0.51
8.15	1	0.405	0.600	0.4	0.903	0.712	0.28
9.60	1	0.405	0.600	0.4	0.903	0.712	0.28

Where:

- pred2 (log odds) = 0.405 + 0.3 × m2

- pred2 (prob) = sigmoid(pred2 log odds)
- res2 = y - pred2 (prob)

Observations

- Residuals (res2) are lower than previous residuals (res1), indicating improvement in model performance.
- This shows that the combined model is learning well through boosting.

Next Steps: Move to Stage 3

In Stage 3:

- Use CGPA and res2 to build the next decision tree.
- The combined model becomes:

$$\text{Prediction} = m_1 + 0.3 \cdot m_2 + 0.3 \cdot m_3$$

- Compute new predictions, convert log-odds to probabilities, and calculate new residuals.
- Continue this process until residuals converge close to zero.

Conclusion

XGBoost builds trees sequentially by:

- Using residuals as pseudo-targets
- Calculating output values at leaves using a formal gain formula
- Updating predictions with scaled tree outputs
- Iterating until the model learns the pattern

While similar to gradient boosting, XGBoost improves the process through:

- Better handling of regularization
- Greedy optimal split finding
- Use of second-order approximation for loss