GridSearchCV vs RandomizedSearchCV

GridSearchCV:

- Tries all combinations of parameters in a grid.
- Exhaustive, but can be very slow if the grid is large.
- Best when the number of parameters and possible values is small.

RandomizedSearchCV:

- Randomly samples combinations from the parameter space.
- Much faster and scalable for large or continuous spaces.
- Ideal for initial search and tuning over a wide range.

Random Forest:

- Recommended: RandomizedSearchCV (due to many less-sensitive parameters).
- Good ranges: n_estimators (100-1000), max_depth (None, 10-50), etc.

XGBoost:

- Recommended: RandomizedSearchCV for initial search, then fine-tune with GridSearchCV.
- Important hyperparameters: learning_rate, max_depth, n_estimators, subsample, colsample_bytree.

Handling Class Imbalance During Evaluation:

- Avoid accuracy.
- Use: precision, recall, F1, ROC-AUC, PR-AUC, confusion matrix.
- Use stratified splits, class weighting, resampling (SMOTE/undersampling), and threshold tuning.

ROC Curve:

- Plots TPR vs FPR at various thresholds.

- AUC-ROC measures overall model ranking ability.

- Higher AUC = better classifier.

- Use with balanced datasets; use PR Curve for high imbalance.

XGBoost Hyperparameters:

- learning_rate (eta): [0.01 - 0.3], lower = better generalization.

- max_depth: [3 - 10], deeper = more complex model.

- n_estimators: [100 - 1000+], number of boosting rounds.

- subsample: [0.5 - 1.0], row sampling for each tree.

- colsample_bytree: [0.3 - 1.0], feature sampling per tree.

Does XGBoost Use Only One Tree?

- No, XGBoost uses many trees built sequentially.

- Controlled by n_estimators.

- Each tree corrects the previous one's errors using gradient boosting.