1 Hyperparameter Optimization (HPO)

1.1 What is Hyperparameter Optimization?

Why do some machine learning models outperform others, even with the same algorithms and data? The answer often lies in hyperparameters, critical settings like learning rate, model complexity, and regularization, adjusted before training begins. These settings act as tuning knobs that significantly impact a model's ability to learn and generalize to new data.

Finding optimal hyperparameters is challenging. The search space is vast, testing configurations is computationally expensive, and manual tuning is slow and imprecise. Exhaustive search is impractical due to the large number of combinations.

Hyperparameter Optimization (HPO) addresses this challenge by developing efficient, automated algorithms to find optimal configurations without excessive manual effort or computation. HPO accelerates model training, reduces costs, and enhances model performance by eliminating guesswork and unlocking the true potential of machine learning algorithms.

At its core, hyperparameter tuning is simply an optimization problem. The goal is to find the set of hyperparameters that maximizes a model's performance, typically measured by accuracy or another metric, on a validation dataset. Effective Hyperparameter Optimization (HPO) seeks to identify these ideal settings efficiently.

1.2 Popular Hyperparameter Optimization Methods

- **Grid Search**: Exhaustively tests all possible combinations within a predefined search space.
- **Random Search**: Samples random combinations of hyperparameters, often more efficient than Grid Search.
- **Bayesian Optimization**: Uses probabilistic models to guide the search toward promising regions of the search space.
- **Evolutionary Algorithms**: Applies principles of natural selection to evolve optimal hyperparameter configurations over generations.

1.3 Popular Hyperparameter Optimization Libraries

- Optuna
- Ray Tune
- SMAC3

1.4 What is Gradient Boosting?

Gradient Boosting is an **ensemble learning technique** where multiple decision trees are trained **sequentially**. Each tree attempts to correct the mistakes made by the previous trees. The final prediction is the combined output of all the trees.

If you know how Decision Trees work, that's great! Gradient Boosting just involves training several trees in sequence to gradually improve the performance.

*Note: Don't worry if this concept is not entirely clear right now. It's not crucial for this lab, as the goal is to practice **hyperparameter tuning**. Most machine learning models have some hyperparameters to tune.*

1.5 Popular Gradient Boosting Libraries

- XGBoost: Highly optimized, popular for competitions.
- LightGBM: Fast and efficient, especially with large datasets.
- CatBoost: Great for categorical data handling, requires minimal preprocessing.

Note: Gradient Boosting models (like CatBoost, XGBoost, LightGBM) have several important hyperparameters (like number of trees, learning rate, tree depth) that significantly impact performance. Therefore, they are excellent candidates for applying HPO techniques, which is what we'll practice in this lab.

1.6 Why is Gradient Boosting Important?

- It is often considered **state-of-the-art for tabular problems** (datasets with structured rows and columns).
- Works well on both regression and classification tasks.

1.7 Recommended Reading

1. When do neural nets outperform boosted trees on tabular data?
2. M5 Forecasting - Accuracy
3. Tuning the hyper-parameters of an estimator
4. When to Choose CatBoost Over XGBoost or LightGBM (Practical Guide)
5. The State of Machine Learning Competitions