Problem Statement:

Using genetic programming, hyperparameter optimization involved in the optimization of machine learning pipelines was performed. In machine learning, hyperparameter optimization is the search for the most suitable among all combinations of n different hyperparameters that take values at certain intervals. This search is based on minimizing the experiment cost, optimization time, and therefore the computational cost and effort spent, by using algorithms that ideally require high convergence performance. In this study, I tested whether the population produced in each generation could converge more effectively and successfully using genetic functions like mutation and crossover. As a case study, LightGBM hyperparameter set was optimized for California House Pricing Prediction Dataset.

Methodology:

Before starting the experiment, according to the selected experiment configurations (population size, number of parameters, parameter ranges (lower and upper limits), the number of discrete intervals per parameter is determined.

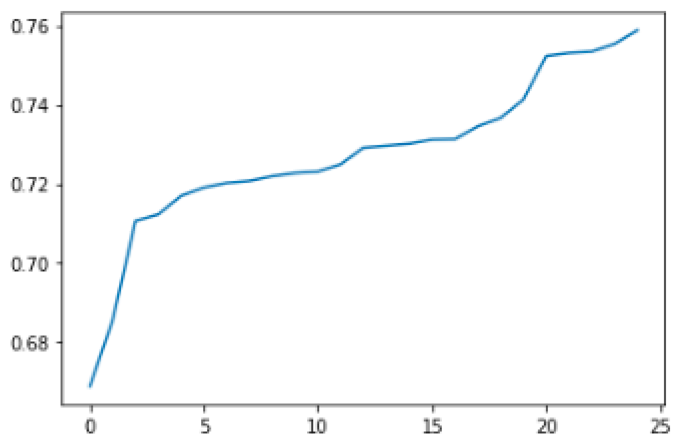
The population generates initial values according to the above configurations. A chromosome is expressed as a binary of the discrete intervals included in this experiment. For example, if the max_depth parameter is between 0-8, if these values are grouped by 2 (For example [0,1], [2,3]), if [0,1] is selected, it will be binarized as 1000, the same formulation is applied for the remaining parameters. These binarized parts are then added together to form the phenotype of the chromosome.

The mutation operation is used to reach the global maximum by expanding to different values of the same parameter. Different values of the parameter (for example 0) affect the selection of the mutation operator by weighting it according to the mean fitness values of the chromosomes that receive that value.

The cross-over operation is used to select different parameters, whichever parameter is selected, for example max_depth or max_leaf, min_samples_split, to determine which chromosomes to cross over. Each round, which parameters for chromosomes will be determined to be optimized in this round, if the probability value determined(specified by switch_prop) is not above the specified parameter, it chooses the parameter with the highest standard deviation (which affects the model the most if it changes), otherwise it chooses randomly among the rest. This allows it to focus on combinations of parameters, not just one parameter.

By training the existing combinations with sample data on the model, the r2_score metric in the training result was determined as the fitness values of the chromosomes.

Results:



Convergence plot of GA methodology
(X axis represents the iteration number, Y axis represents R2 Score)

I found that the genetic operations reached the same convergence point as the other methods, but the convergence performance was not as fast as the efficiency of the bayesian optimization method. I found that the convergence that Bayesian optimization does in 1 unit of time, that genetic algorithms do in my experiments is between 1.5~2 units. Optina was used as a Bayesian optimization benchmark. At the same time, parametric optimizations of genetic operations (mutation) (how many percent probability mutations and cross-overs will be applied) and how to apply them are more important than standard optimization problems and need to be approached specifically, and I have seen that they complicate the problem. But I think that genetic algorithms will be successful when the literature for these problems is enriched and specialized.

Conclusion:

This course added skills for me to approach optimization problems (model selection, hyperparameter optimization, etc.) that I use in my daily life in data science from different directions. In addition, this course directed me to academic literature research and encouraged me to learn how to summarize an article by breaking it into semantic parts and how to present it to an audience. During the course, he also contributed to the situations in which different genetic operators and genetic libraries might have advantages and disadvantages over others. In addition to the theoretical background with its practical applications, I learned the requirements for creating a genetic algorithm framework by working on the use cases in which I applied genetic operators.