Hyperparameter Optimization using Half Grid Search, Genetic Algorithm, Simulated Annealing and Comparison with Grid Search

Wong Yan Jian[a], Associate Prof. Ts. Dr. Sri Devi A/P Ravana[a]

[a]*Faculty of Computer Science and Information Technology, Universiti Malaya, 50603 Kuala Lumpur, Malaysia*
u2102753@siswa.um.edu.my, sdevi@um.edu.my

Abstract

This report aims to conduct a comparative analysis between Simulated Annealing, Genetic Algorithm, Half Grid Search, and Grid Search by evaluating them from three aspects: computation time, solution quality, and diversity in the hyperparameters found by each method. To achieve this, these four hyperparameter tuning methods were used to tune classifiers for predicting credit card churning, and their performances were evaluated. The results found that the Genetic Algorithm is the hyperparameter tuning method that performs best across all three metrics. The models, findings, and results from the experiments were published in a web application to share the outcomes of the experiment worldwide.

Keywords: Hyperparameter Optimization, Machine Learning, Half Grid Search, Genetic Algorithm, Simulated Annealing

1. INTRODUCTION

The number of data generated has been growing from time to time since the beginning of the Internet age, whether from online transactions, emails, social media interactions, sensors and other sources (27). In order to tap into the value that could be gained from these data, machine learning (ML) is the most intelligent way to do so. Various types of ML methods, such as supervised, unsupervised, semi-supervised, deep learning and reinforcement learning have been used to analyze and search the hidden patterns, trends and outliers in the datasets. By using these techniques, we can obtain valuable insights from the data to solve business problems, support data-driven decision-making and promote the industry from machine automation to knowledge automation (10). Therefore, ML techniques have been applied in many real-world application domains, including predictive analytics and intelligent decision-making, Internet of Things (IoT) and smart cities, E-commerce and product recommendations, as well as other business problems (28). However, there are many challenges that have to be tackled when building an ML model with a high level of accuracy and precision. One such challenge is hyperparameter tuning.

Hyperparameters are a set of settings that are used to configure an ML model, such as the learning rate for training a deep learning network or the regularization strength (L1 or L2 penalties) in linear models. In addition, they define the algorithm used for minimizing the loss function, like the number of trees in random forests as well as the activation function and optimizer types in neural networks (37). Hyperparameters must be carefully chosen as they could affect the performance of the model to various degrees (14). Thus, a range of possibilities have to be explored in order to build the desirable and usable ML model. The process of finding an optimal set of hyperparameter configurations for the model is known as hyperparameter tuning. It is a part of the key process, especially in designing tree-based ML models and deep neural networks as these types of models have many hyper-parameters (16).

Traditionally, hyperparameter selection has been guided by heuristic rules or experimental evaluation of various combinations within a predefined grid (5). Conventional automated approaches such as Grid Search (GS) have been prevalent for hyperparameter optimization (HPO). However, these methods often prove to be exhaustive and inefficient, particularly when navigating vast HPO search spaces, leading to the curse of dimensionality (6). To mitigate these challenges, advanced algorithmic strategies like Half Grid Search (HGS), Genetic Algorithm (GA) and Simulated Annealing (SA) have emerged as promising alternatives, offering more sophisticated solutions to the complexities of HPO.

2. LITERATURE REVIEW

In recent years, researchers have conducted various experiments to compare different HPO methods. Lin et al. (2008) compared the performance of a Support Vector Machine (SVM) optimized with SA and GS across 11 classification datasets. The results showed that SVM with SA achieved higher classification accuracy rates in all cases compared to GS, particularly after performing feature selection, which highlighted a significant difference in accuracy between SA and GS. Specifically, SVM with SA achieved an accuracy score higher than 90% for 8 out of 11 datasets, while GS only achieved this in 4 datasets.

Polgár et al. (2000) compared Random Search, GA, SA, Random Hill Climbing, and Combinative Hill Climbing methods in evaluating ellipsometric measurements, using average Root Mean Square Error (RMSE) as the metric. The experiment revealed that GA had a lower total average RMSE score of 0.15 compared to SA's 0.19. The researchers also noted that

SA is efficient when the parameter space dimension is large, but it is significantly affected by the initial point's location in the search space and the roughness of the error surface between the initial point and the solution.

Blume et al. (2021) compared Hyperband, GA, Random Search, and Bayesian Optimization in designing a roll angle estimator based on an artificial neural network. The objective function in this case was RMSE, and GA showed better results compared to other methods, generally achieving lower RMSE scores below 0.02 for all 5 solutions found, in contrast to Hyperband, which obtained 4 solutions with RMSE scores higher than 0.02. The research suggested that Hyperband Algorithms do not exhibit continuous improvement during the optimization process, whereas knowledge-based methods like GA provide promising solutions in such applications and are preferable for optimizing a large number of hyperparameters.

In a study by Petro and Pavlo (2019) comparing GS, Random Search, and GA in optimizing neural architectures for CIFAR-10 dataset, a convolutional neural network was built. The best accuracy for GS was 83%, achieved in 4.3 hours, while GA achieved a best accuracy of 86% in 4.13 hours. This indicates that GA can achieve slightly better accuracy than GS in a marginally shorter time. The paper suggests that GS is preferable when the model and search space are not too large, whereas GA is suitable for models with many layers and a large search space.

Ghazanfari et al. (2007) focused on comparing SA and GA in learning Fuzzy Cognitive Map (FCM), directed graphs showing the relations between essential components in complex systems. In this study, SA was found to learn faster than GA, finding related solutions in less computational time (approximately 2 seconds compared to GA's 40 seconds). However, GA achieved lower error scores when the number of nodes was relatively small, while the opposite occurred when the number of nodes was larger. Therefore, the findings concluded that FCMs with fewer nodes are better suited to GA, and SA is more effective for a larger number of nodes.

Tsai and Fang (2021) conducted a comparative analysis between GS, SA, and Particle Swarm Optimization in hyperparameter optimization for Deep Neural Networks (DNNs) predicting passenger numbers at Taipei Main Station of Taipei Metro. They used Mean Absolute Percentage Error (MAPE) as the performance metric. The results indicated that DNN with SA had a lower MAPE score at 7.920 compared to DNN with GS at 8.084 for weekday predictions. However, for weekend predictions, DNN with GS outperformed DNN with SA, having a lower MAPE score of 9.128 compared to 9.175. This study highlighted that advanced algorithms like SA do not always guarantee better performance than traditional methods like GS.

Vincent and Jidesh (2023) discussed different HPO techniques in their work on building an improved hyperparameter optimization framework for AutoML systems using evolutionary algorithms. They specifically compared Hyperband and GA in image classification problems. AutoKeras with Hyperband and Tree-based Optimization Tool (TPOT) with GA were compared using CIFAR, MNIST, SVHN and CALTECH datasets. In all cases except for the CALTECH dataset, where Hyperband had an error rate of 0.7, Hyperband performed better than GA. The early stopping criteria in Hyperband also resulted in significantly reduced optimization times. This study demonstrated that HPO performance is highly dependent on the dataset, leading to varying results for different HPO methods.

Kozarovytska and Kucherenko (2023) focused on comparing the performance of Hyperband, Random Search, Evolution Algorithm, and Bayesian Optimization in optimizing a Convolutional Neural Network for classifying the Fashion MNIST dataset. Their performance was evaluated using accuracy scores. The Evolution Algorithm showed minimal changes from initially sampled configurations, achieving a validation accuracy of 92.4% and a testing accuracy of 91.3%. In contrast, Hyperband, which sampled all configurations randomly, helped the model achieve a higher validation accuracy of 94% and testing accuracy of 93.5%.

Kerr and Mullen (2019) compared GA and SA in maximizing thermal conductance of harmonic lattices. Their experiment showed that GA consistently outperformed SA in terms of final solution strengths, with GA achieving better results in every given scenario and lower error rates. This suggests that in complex problems such as biomolecules, GA can offer superior performance compared to SA.

Coroiu (2016) conducted research comparing GS, GA, and Random Forest using six datasets related to the medical field and involving classification problems. The experimental results indicated that GA optimized several ML models, including K-Nearest Neighbor, Decision Tree, and Random Forest, faster than GS, often in less than a minute. In contrast, the GS approach typically took several minutes or even an hour to complete. Additionally, GA offered good results in terms of accuracy, with 11 out of 18 cases achieving accuracy higher than 80%, compared to GS's 9 cases. In almost all cases, GA's accuracy scores were higher than those of GS. The author suggested that performance might increase with a customized version of GA instead of the standard implementation used in the experiment.

These studies collectively show that while there is extensive research comparing HPO methods, the optimal choice is contingent upon specific factors such as model complexity, dataset, and problem complexity. There is no definitive answer as to which method is universally the best, underscoring the importance of context in selecting HPO techniques.

3. PROBLEM FORMULATION

3.1. Problem Statement

One of the standard approaches to fine-tune the hyperparameters is by using GS, an algorithm that conducts an exhaustive search across a specified subset of the hyperparameter space of the training algorithm (23). Although GS is capable of evaluating all possible combinations to select the best-performing one, it is highly time-consuming and computationally expensive. The issue becomes worse when there are numerous hyperparameters to test across a broad range of values, leading to increased processing time as it iterates through every possible

hyperparameter combination (29). This phenomenon is often referred to as the curse of dimensionality. Consequently, a variety of advanced optimization techniques have been developed to tackle this problem. Each technique employs different strategies aimed at improving efficiency and reducing the resources required to identify optimal hyperparameters.

Novel methods such as evolution strategies like GA, Hyperband like HGS and adaptive explore-exploit tradeoff methods like SA can be used in place of GS when solving optimization problems (3). These methods have their own strategies that can utilise exploration vs exploitation trade-offs when finding the suitable set of hyperparameters. It started with exploring a vast hyperparameter search space to seek potential solutions, and then slowly reducing the search space around the solutions to find the better one (26). Even though there are research papers that are focused on using these techniques to improve HPO performance for a certain ML model, there is still a lack of research on comparing HGS, GA, and SA with GS in HPO.

Therefore, this project aims to fill the gap in the research of HPO by addressing the following problems when comparing GS against HGS, GA and SA:

1. Lack of **assessment** into **computation efficiency** required by each optimization method
2. Limited **evaluation of hyperparameter solution** found by each optimization method
3. Lack of **quantification of the diversity** in hyperparameter solutions found by each optimization method

3.2. Objectives

In order to explore and compare novel optimization methods, specifically HGS, GA and SA, for hyperparameter tuning in ML models against traditional GS method, the following objectives have to be achieved:

1. To **measure** and **compare** the average computation time taken by each method to find the optimal set of hyperparameters.
2. To **assess** the quality of hyperparameters found by each method by evaluating the performance of the resulting models using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.
3. To **quantify** the diversity of hyperparameters evaluated by each method to assess their exploratory capabilities.

By doing so, the study can uncover insights into the relative advantages and limitations of each method, which provides a nuanced understanding of their applicability and performance in searching for suitable hyperparameters.

4. METHODOLOGY

Cross Industry Standard Process for Data Mining (CRISP-DM) is a conventional process model that serves as a foundation for a data science project. It provides step-by-step guidance on how to conduct the project from start to finish, which can systematize the process flow and ensure a high-quality result from the project (35). The model consists of six iterative phases, which are Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation and Deployment. The methodology of the project will be implemented following CRISP-DM.

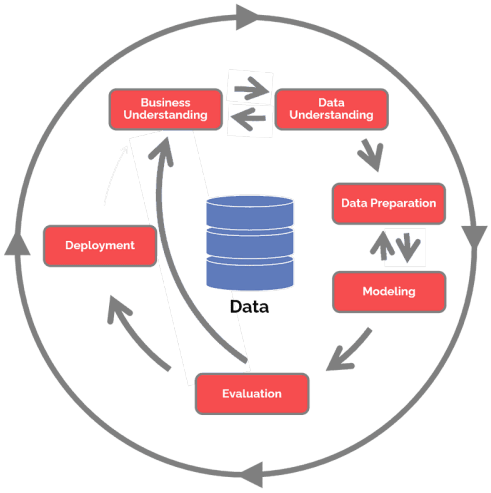


Figure 1: An overview of CRISP-DM model (15)

4.1. Business Understanding

Understanding the business context is the critical first step in any project. It involves a thorough comprehension of the project's goals and requirements, as well as establishing clear success criteria (35). This foundational stage is facilitated by defining the problem statement and objectives.

The focus of this project is to conduct a comparative analysis of HGS, GA and SA against traditional GS within the domain of HPO. By addressing the known challenges associated with Grid Search in HPO, this project aims to enhance efficiency and effectiveness in model tuning. Success will be measured by the project's ability to meet its objectives and demonstrate the comparative advantages of these HPO techniques through various metrics.

4.2. Data Understanding

Data understanding is a process that consists of data collection, exploration and verification to ensure comprehensive knowledge of the dataset and to confirm that there are no issues with its quality. (30).

4.2.1. Dataset Description

A credit card dataset (12) was obtained from Kaggle, comprising sociodemographic and behavioural data of credit card customers, which is useful for predicting customer churn. This dataset includes information from 10,127 credit card customers, encompassing 20 features and 1 class variable. Figure 2 illustrates samples from the dataset.

The features are categorized into three groups: 14 numerical features (including age, number of dependents, total transaction amount, among others), 4 ordinal features (such as attrition flag, education level, card category, and income category, and

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_book
0	768805383	Existing Customer	45	M	3	High School	Married	$60K-80K	Blue	39
1	818770008	Existing Customer	49	F	5	Graduate	Single	Less than $40K	Blue	44
2	713982108	Existing Customer	51	M	3	Graduate	Married	$80K-120K	Blue	36
3	769911858	Existing Customer	40	F	4	High School	Unknown	Less than $40K	Blue	34
4	709106358	Existing Customer	40	M	3	Uneducated	Married	$60K-80K	Blue	21

Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit	Total_Revolving_Bal	Avg_Open_To_Buy	Total_Amt_Chng_Q4_Q1
5	1	3	12691.0	777	11914.0	1.335
6	1	2	8256.0	864	7392.0	1.541
4	1	0	3418.0	0	3418.0	2.594
3	4	1	3313.0	2517	796.0	1.405
5	1	0	4716.0	0	4716.0	2.175

Avg_Open_To_Buy	Total_Amt_Chng_Q4_Q1	Total_Trans_Amt	Total_Trans_Ct	Total_Ct_Chng_Q4_Q1	Avg_Utilization_Ratio
11914.0	1.335	1144	42	1.625	0.061
7392.0	1.541	1291	33	3.714	0.105
3418.0	2.594	1887	20	2.333	0.000
796.0	1.405	1171	20	2.333	0.760
4716.0	2.175	816	28	2.500	0.000

Figure 2: Sample data

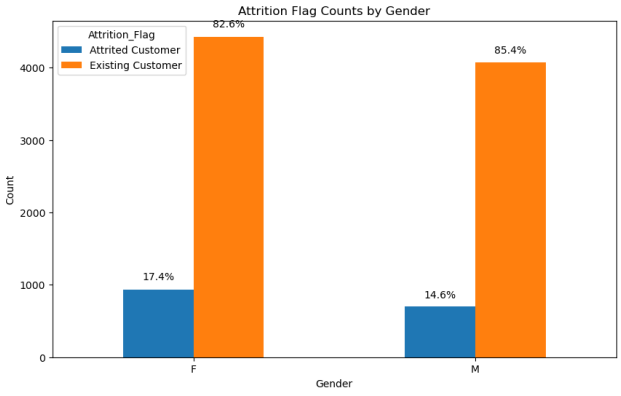


Figure 3: Bar Chart of Attrition Flag Breakdown by Gender

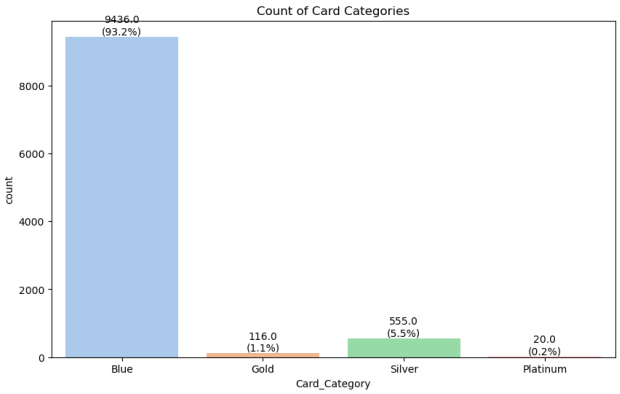


Figure 4: Bar Chart of Card Categories

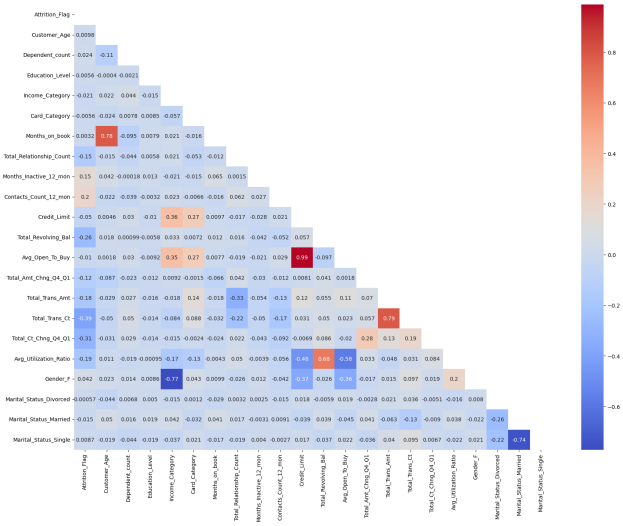


Figure 5: Correlation Graph

3 nominal features (client number, gender, and marital status). The description of each variable is presented in Table 1.

4.2.2. Exploratory Data Analysis

Subsequently, Exploratory Data Analysis (EDA) was conducted to acquire a comprehensive understanding of the data. This phase is vital for evaluating data quality and guiding the subsequent data-cleaning processes.

Figure 3 illustrates that the numbers of female and male customers are approximately closer to each other. However, the attrition flag target is imbalanced in both genders, with only about 17.4% for female customers and 14.6% for male customers. Examining the number reveals 8,500 records with 'Existing Customer' and 1,627 records for 'Attrited Customer'. In scenarios where a large portion of instances belong to one class compared to others, traditional classifiers may encounter an imbalanced learning problem, as they tend to classify data into the majority class. Therefore, a resampling method such as the Synthetic Minority Oversampling Technique (SMOTE) is required to address this imbalanced dataset.

Figure 4 displays the distribution across different card categories. Notably, there are only 116 'Gold' and 20 'Platinum' cards among the 10,127 data points, making them rare values compared to other categories. Therefore, these cards could be merged into a single category for mapping in later stages.

Figure 5 presents the correlation between various variables. A significant observation from this graph is the strong correlation (0.99) between 'Avg_Open_To_Buy' and 'Credit_Limit'. This correlation is logical, as 'Avg_Open_To_Buy' represents the difference between the credit limit assigned to the cardholder's account and the current balance. Therefore, this particular feature could be omitted in later stages, as it does not provide additional value in building classification models.

4.3. Data Preparation

Data preparation is often a time-consuming phase in which raw data undergo preprocessing and further analysis to be formatted appropriately for modelling (2).

4.3.1. Data Preprocessing

The dataset used is of decent quality, as it contains no missing values or duplications. Initially, records exhibiting outliers in more than one column were removed. Outliers were identified by determining if the a variable's value exceeded 1.5 times

Variable Name	Description
CLIENTNUM	Client Number. Unique identifier for the customer holding one credit card
Attrition_Flag	Class variable or the target variable. Consist of Existing Customer and Attrited Customer
Customer_Age	Age of the customer
Gender	Gender of the customer. Consists of M and F
Dependent_Count	Number of dependents for the customer
Education_Level	Education level of the customer. Consists of Unknown, Uneducated, High School, College, Post-Graduate, Graduate, and Doctorate
Marital_Status	Marital status of the customer. Consists of Married, Single, Divorced, and Unknown
Income_Category	Income category of the customer. Consists of Unknown, Less than $40K, $40K - $60K, $60K - $80K, $80K - $120K, and $120K +
Card_Category	Type of credit card held by the customer. Consists of Blue, Silver, Gold, and Platinum
Months_on_Book	Number of months the customer has been with the bank
Total_Relationship_Count	Total number of products the customer has with the bank
Months_Inactive_12_mon	Number of months the customer has been inactive in the last 12 months
Contacts_Count_12_mon	Number of contacts between the customer and the bank in the last 12 months
Credit_Limit	Credit limit of the customer
Total_Revolving_Bal	Total revolving balance on the customer's credit card
Avg_Open_To_Buy	Average open to buy credit line
Total_Amt_Chng_Q4_Q1	Change in transaction amount from Q4 to Q1
Total_Trans_Amt	Total transaction amount in the last 12 months
Total_Trans_Ct	Total number of transactions in the last 12 months
Total_Ct_Chng_Q4_Q1	Change in transaction count from Q4 to Q1
Avg_Utilization_Ratio	Average card utilization ratio

Table 1: Description for each variable in dataset

the interquartile range from the first to the third quartile. This process resulted in the removal of 1,274 records, leaving a total of 8,853 records. Subsequently, ordinal columns were mapped according to Table 2. One-hot encoding was then applied to nominal columns, specifically 'Gender' and 'Marital_Status'. Following this, one column from each set of one-hot encoded variables ('Gender_M' and 'Marital_Status_Unknown') was removed. This is because the value of the last one-hot encoded column can be inferred from the others. Removing these columns helps reduce the Variance Inflation Factor (VIF) score among these features, addressing multicollinearity issues and simplifying the feature selection process. Additionally, unused variables like 'CLIENTNUM' and highly correlated variable, 'Avg_Open_To_Buy' were removed. Ultimately, 21 variables remained for each record.

4.3.2. Train Test Splitting and Standardization

A standard 80/20 split was employed, resulting in 7,082 records for training and 1,771 for testing. All 13 numerical variables in the training set were standardized based on their corresponding z-values. The means and standard deviations of the numerical variables from the training set were also utilized to standardize numerical variables in the test set.

4.3.3. Feature Selection

Feature extraction was conducted following the approach used by Wu and Wang (2022), which consists of: (1) SelectKBest and (2) Permutation Importance with an Extra Tree Classifier. The selected features from these two methods were compared by fitting them into a simple logistic regression model. The features selected by the method that achieved a higher accuracy score were used as predictors in modelling.

SelectKBest, a function provided by Scikit-learn, selects a specified number (k) features based on the highest scores. In this approach, the scoring function 'f_classif' was employed,

Variable	Value	Mapped Value
Attrition_Flag	Existing Customer	0
	Attrited Customer	1
Education_Level	Unknown	0
	Uneducated	1
	High School	2
	College	3
	Post-Graduate	4
	Graduate	5
	Doctorate	6
Income Category	Unknown	0
	Less than $40K	1
	$40K - $60K	2
	$60K - $80K	3
	$80K - $120K	4
	$120K +	5
Card_Category	Blue	0
	Silver	1
	Gold	2
	Platinum	2

Table 2: Mapping table for ordinal variables

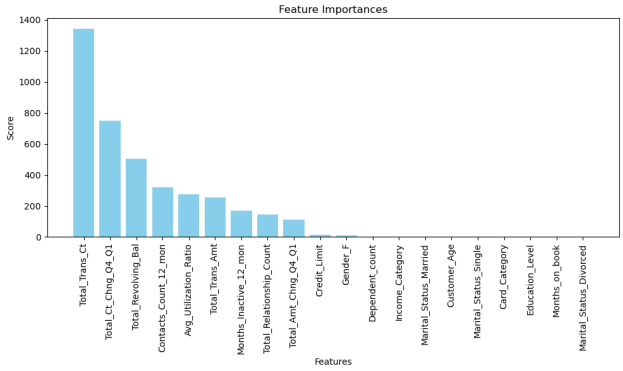


Figure 6: Feature Importance Score from SelectKBest

Model	Hyperparameter	Search Space
LR	penalty	elasticnet, l1, l2, None
	C	$2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, 2^{3}, 2^{5}, 2^{7}, 2^{9}, 2^{1}3$
	solver	sag, saga, liblinear, lbfgs, newton-cg, newton-cholesky
	l1_ratio	0, 0.2, 0.4, 0.6, 0.8, 1
SVC	kernel	linear, poly, rbf, sigmoid
	C	$2^{-5}, 2^{-3}, 2^{-1}, 1, 2^{1}, 5, 2^{3}, 10$
	degree	2, 3, 4, 5
	gamma	scale, auto
RFC	n_estimators	80, 100, 150, 200, 250, 500
	criterion	gini, entropy, log_loss
	max_depth	3, 4, 5, 7, 10, 15, 20
	min_samples_split	2, 3, 5, 7, 11, 17, 26, 40
GBC	learning_rate	np.logspace(-3, 0, num=10)
	n_estimators	80, 100, 150, 150, 200, 250, 500
	subsample	0.5, 0.6, 0.7, 0.8, 0.9, 1
	max_depth	3, 4, 5, 7, 10, 15, 20

Table 3: Search Space for each Machine Learning Model

which computes the ANOVA F-value between the label and features. Here, k was set to 15 to identify the top 15 features.

Permutation importance, on the other hand, is a feature selection method that measures feature importance by observing the effect of randomly shuffling each feature on model accuracy. It begins with a trained machine learning model to calculate the baseline accuracy on the dataset. Then, it randomly shuffles the values in one feature at a time and calculates the prediction accuracy on the shuffled dataset. The difference between the baseline accuracy and the shuffled dataset accuracy represents the feature importance score. Features causing a significant drop from the baseline accuracy are deemed important. This process is repeated for every feature, and the results are ranked based on the feature importance score for each variable. In this approach, the Extra Tree Classifier was used as the baseline model. This classifier trains decision trees on the entire dataset, randomly selecting the values at which to split a feature and create child nodes, differing from the Random Forest that uses bagging techniques (31).

Ultimately, the top 15 features from SelectKBest achieved a slightly higher accuracy score of 0.8938 compared to 0.8961 for Permutation Importance with Extra Tree Classifier. Figure 6 displays the feature importance score for each variable evaluated using SelectKBest.

Subsequently, a Wald Test was conducted to determine the significance of the selected variables. A p-value threshold of 0.05 was set, with a 95% confidence interval. Variables with a p-value greater than 0.05 were deemed insignificant and removed from the selected features. Based on the results, 'Customer_Age' was removed due to its p-value of approximately 0.8295. Consequently, 14 features remained: 'Total_Trans_Ct', 'Total_Ct_Chng_Q4_Q1', 'Total_Revolving_Bal', 'Contacts_Count_12_mon', 'Avg_Utilization_Ratio', 'Total_Trans_Amt', 'Months_Inactive_12_mon', 'Total_Relationship_Count', 'Total_Amt_Chng_Q4_Q1', 'Credit_Limit', 'Gender_F', 'Dependent_Count', 'Income_Category', and 'Marital_Status_Married'.

4.3.4. Resampling

SMOTE was applied at the end of the process, as it is more effective when used later rather than earlier (25). It is a technique that oversamples the minority class in the training set, effectively making the minority class more general. This resulted in an equal number of records for positive and negative examples. Finally, the training set was shuffled to ensure a random mix of real and synthetic samples, promoting more effective and unbiased learning.

4.4. Modelling

Four classification models were selected for classifying the target variables: Logistic Regression (LR), Support Vector Classifier (SVC), Random Forest Classifier (RFC), and Gradient Boosting Classifier (GBC). These ML models vary in the number of hyperparameters requiring tuning, making them ideal for testing the capabilities of HGS, GA, SA, and GS in the HPO experiment. The selection of the search space for each hyperparameter is based on the works of Nie et al. (2011), Coussement and Van Den Poel (2008), Konuksal (2018), and Al-Shourbaji et al. (2023), as shown in Table 3.

The HPO experiment was designed so that each of the four ML models would undergo the four hyperparameter tuning methods, resulting in a total of 16 model-building instances for each round of the experiment. Additionally, five seeds (7, 13, 42, 404, 1001) were chosen to repeat the entire HPO experiment, each introduced at different phases. The first experiment involved introducing a seed during data splitting, focusing on assessing computation time and the quality of the hyperparameters found. For the second experiment, using the same training and testing splits with seed 42, the five seeds were introduced during the initialization of the model. This was done to minimize the effect of data splitting and to assess the exploratory capabilities of each HPO method. In total, 9 rounds of experiments were conducted, resulting in the construction of 144 models.

4.5. Evaluation

To ensure a fair evaluation of the different HPO techniques, each round of the experiment was conducted in the same Google Collaboratory environment to guarantee identical computational resources. Each round of the experiment was repeated five times to account for variability in performance. Ad-

ditionally, specific conditions were imposed on the HPO methods to further ensure a fair evaluation, including a fixed 100 iterations for SA, 10 rounds for HGS, and a total of 100 populations for GA.

A baseline model using default hyperparameters was run to serve as a reference point for the enhancements provided by each HPO technique. All models, whether baseline or optimized, were evaluated using the same approach. A 3-fold cross-validation was utilized to assess the models with five metrics: accuracy, precision (macro), recall (macro), F1-score (macro), and ROC-AUC (macro) (13). These metrics enabled a comprehensive evaluation of the model performance from different perspectives.

Three metrics were used to evaluate the performance of HGS, GA, SA, and GS. The computation efficiency of each HPO method was assessed based on their average computation time to complete the optimization process. The quality of the hyperparameter solutions found by each optimization method was assessed based on their ROC-AUC improvement for the testing set. This was chosen because the testing set is imbalanced, and ROC-AUC is insensitive to this imbalance. Finally, the diversity of hyperparameter solutions was quantified by calculating the average Manhattan distance variations between each point for each HPO method. This involved computing all pairwise distances for each point using Manhattan distance and then averaging them. To facilitate this, the hyperparameter solutions were represented in a higher-dimensional space as points, achieved by performing min-max normalization on numerical parameters and one-hot encoding on categorical parameters (17). By combining the results from these three metrics, the strengths and weaknesses of each HPO method could be discerned based on different aspects.

4.6. Deployment

Streamlit, an open-source framework was used to create web applications for hosting the ML models and findings from the HPO experiment. Users can input required information to predict their likelihood of churn using the four top-performing models from different ML algorithms. Experiment results and visualizations were published on the web application, allowing users to browse them. In addition, the web application serves as a resource for users to learn about HGS, GA, SA, and GS, and to generate code for utilizing these HPO methods in their local environments. A user manual for the web application is included in the Appendix. The code for each HPO method has been published on GitHub, enabling users worldwide to fork and modify it according to their needs.

5. RESULT

5.1. Model Performance

This section presents a comparison of different models using various optimization methods. It includes the average results for performance metrics such as accuracy, F1-score, precision, Rrecall, and ROC-AUC for all five seeds when predicting the testing set.

Methods	Models	Accuracy	F1-score	Precision	Recall	ROC-AUC
Baseline	LR	0.8494	0.8623	0.8925	0.8494	0.9208
	SVC	0.9127	0.9166	0.9248	0.9127	0.9597
	RFC	0.9557	0.9559	0.9562	0.9557	0.9854
	GBC	0.9539	0.9547	0.9563	0.9539	0.9856
SA	LR	0.8487	0.8617	0.8923	0.8487	0.9206
	SVC	0.9223	0.9247	0.9274	0.9230	0.9610
	RFC	0.9577	0.9578	0.9581	0.9577	0.9861
	GBC	0.9651	0.9651	0.9651	0.9651	0.9885
HGS	LR	0.8494	0.8623	0.8926	0.8494	0.9208
	SVC	0.8746	0.8838	0.9043	0.8746	0.9338
	RFC	0.9562	0.9563	0.9566	0.9562	0.9859
	GBC	0.9651	0.9651	0.9653	0.9651	0.9907
GS	LR	0.8480	0.8612	0.8922	0.8480	0.9210
	SVC	0.9231	0.9245	0.9267	0.9231	0.9608
	RFC	0.9564	0.9565	0.9568	0.9564	0.9856
	GBC	0.9651	0.9651	0.9651	0.9651	0.9904
GA	LR	0.8490	0.8620	0.8923	0.8490	0.9207
	SVC	0.9217	0.9232	0.9253	0.9217	0.9549
	RFC	0.9592	0.9596	0.9602	0.9592	0.9866
	GBC	0.9653	0.9655	0.9657	0.9653	0.9912

Table 4: Performance of Model by HPO Methods

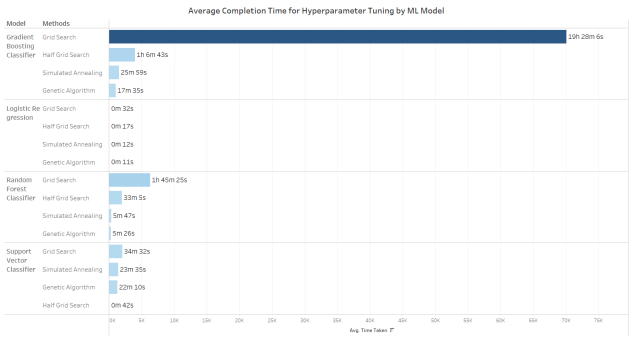


Figure 7: Bar Chart of Average Completion Time

Table 4 displays the performance of each model using different HPO optimization methods. Overall, LR exhibited the poorest performance across all metrics, even after optimization. GBC and RFC showed similar performance, with RFC slightly leading in the baseline model. However, after optimization, GBC managed to outperform RFC, indicating that it benefits more from hyperparameter optimization compared to RFC. The worst-performing model was LR optimized with GS, with the lowest accuracy and recall score of 0.8480, precision of 0.8922, F1-score of 0.8612, and ROC-AUC of 0.9210. Conversely, the best-performing model was GBC optimized with GA, achieving the highest accuracy and recall of 0.9653, precision of 0.9657, F1-score of 0.9655, and an astonishing ROC-AUC of 0.9912.

5.2. Computation Time Evaluation of HPO Methods

Figure 7 illustrates the average completion time for each HPO method. Notably, GS took the longest, particularly for GBC, which required approximately 19 hours and 28 minutes to complete. This extended duration is attributed to GBC's numerous hyperparameters needing tuning, causing GS to take a significant amount of time. Another interesting observation is that HGS required only 42 seconds to complete tuning for SVC,

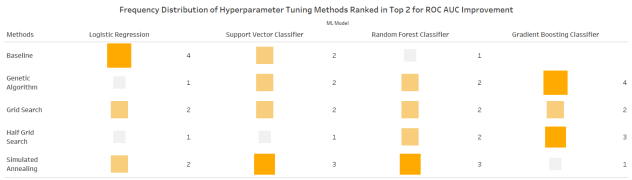


Figure 8: Frequency Distribution of HPO Methods Ranked in Top 2 for ROC-AUC Improvement

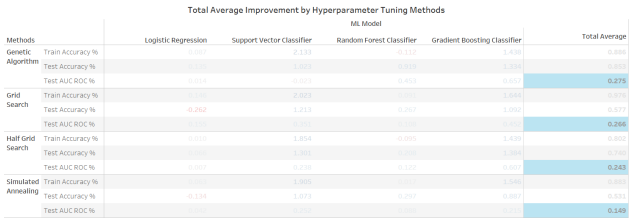


Figure 9: Average Improvement by HPO Methods

due to the selection of simpler SVC models with linear kernels in the early iterations. Ranking the completion time, GA was the fastest, followed by SA, HGS, and lastly GS, which suffers from the curse of dimensionality.

5.3. Quality Assessment of Hyperparameter Solutions in HPO Methods

Figure 8 shows the frequency distribution of HPO methods ranked in the top 2 for ROC-AUC improvement. The results indicate that the baseline model is effective for simpler models like LR, as it ranked in the top 2 for ROC-AUC improvement 4 out of 10 times. GA was highly effective for complex models like GBC, being selected 4 times. GS performed averagely across all models, while HGS showed improvement in solution quality as model complexity increased. SA demonstrated its strength for moderately complex models like SVC and RFC, ranking in the top 2 for ROC-AUC improvement 3 times.

Figure 9 depicts the average improvement by each HPO method. GA provided the largest overall ROC-AUC improvements at 0.275%, followed by GS at 0.266%, HGS at 0.243%, and SA at 0.149%. This suggests that GA can offer higher-quality hyperparameter solutions compared to other methods, albeit by a narrow margin.

5.4. Diversity of Hyperparameter Solutions in HPO Methods

Figure 10 presents the average Manhattan distance variations for each HPO method. A higher average distance variation infers greater exploratory capabilities of an HPO method. GA had

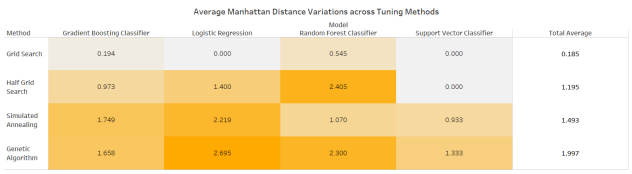


Figure 10: Average Manhattan Distance Variations across HPO Methods

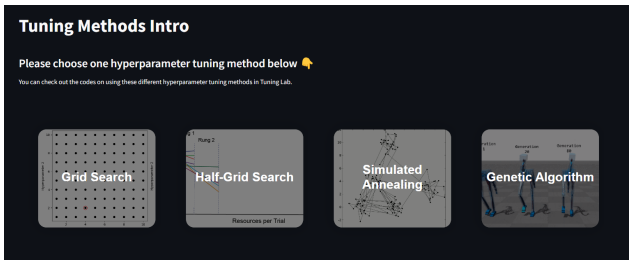


Figure 11: HyperTune 101 Page

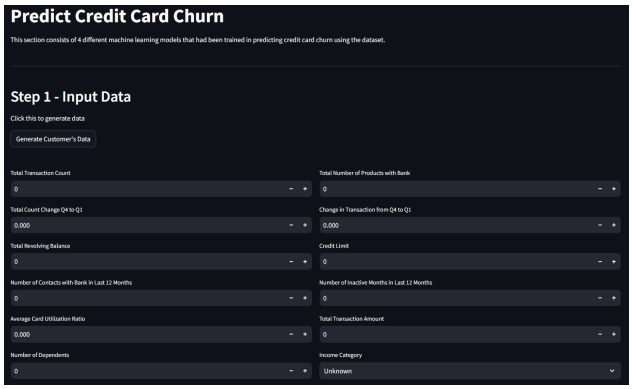


Figure 12: Predict Churn Page

the highest values at 1.997, followed by SA at 1.493, HGS at 1.195, and GS at 0.185. This indicates that GA has the highest exploratory capability, while GS has limited exploratory capability. GS evaluates every possible hyperparameter solution and returns the one with the highest accuracy or user-defined score, limiting the diversity of its solutions. In contrast, GA employs mechanisms inspired by natural selection and genetics to explore the hyperparameter space more effectively, allowing for a broader range of solutions.

5.5. Application

The web application was developed using the Streamlit framework and is accessible via the link: https://hpoexperiment-8wze5k5k6qpuscgwk7qmst.streamlit.app/. The application features six pages: 'Home', 'Dataset Overview and EDA', 'HyperTune 101', 'Predict Churn', 'Tuning Lab', and 'Results and Findings'. Among these, the key pages include:

- HyperTune 101: This page, as shown in Figure 11, provides comprehensive resources related to GS, HGS, SA, and GA. Users can select any of these methods to learn about their processes, parameters, advantages, disadvantages, and more.
- Predict Churn: As depicted in Figure 12, this page allows users to input 14 pieces of data or generate synthetic data for churn prediction by utilizing LR, SVC, RFC, and GBC models.
- Tuning Lab: Illustrated in Figure 13, this page enables users to generate template codes for using the HPO meth-

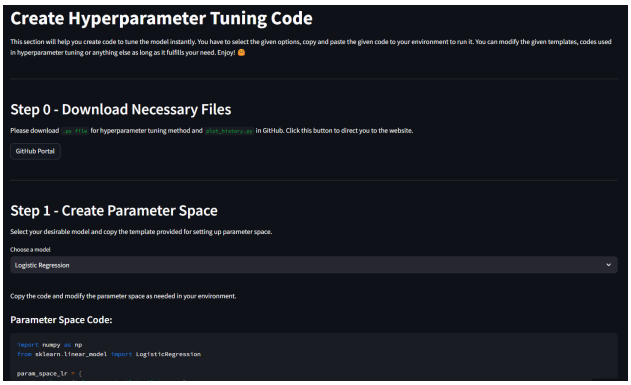


Figure 13: Tuning Lab Page

ods in their local environments. It provides suitable code based on user selections and includes step-by-step instructions to facilitate ease of use.

The details of the web application can be found in the user manual which is attached in Appendix.

6. DISCUSSION

The findings from the HPO experiment are detailed in the previously mentioned figures and tables. Overall, GBC benefited the most from optimization using various HPO methods, while LR continued to perform poorly even after optimization. Among the four hyperparameter tuning methods, GA showed promising results in terms of computation time, solution quality, and exploratory capability. GA completed the optimization process in the shortest time, thanks to its population-based approach that simultaneously explores multiple areas of the search space, leading to a more efficient search process. The mutation process and its balanced exploration-exploitation strategy allow GA to explore beyond immediate neighbourhoods, avoiding local optima and increasing the likelihood of discovering global optima. This results in GA converging to high-quality solutions faster than other methods and exhibiting higher exploratory capabilities. Other methods like HGS and SA also showed promising results compared to GS, requiring less time to complete the tuning process and yielding high-quality solutions depending on the model complexity. Overall, these advanced HPO methods are suitable to be used as an alternative to GS, particularly when the model complexity is high and requires more hyperparameters to tune with.

However, there are some limitations to this HPO experiment. First, the findings are constrained to the use of a credit card churning dataset and do not encompass a diverse range of problems, such as datasets from other domains or regression problems. Therefore, there is a need to conduct similar HPO experiments on various datasets to assess the performance of these HPO methods more broadly. Second, the experiment did not include other types of ML models and deep neural networks, which are popular in a variety of applications. This limitation restricts the applicability of the findings to only LR, SVC, RFC, and GBC. Future experiments should focus on using a wider range of ML models to thoroughly test the capabilities of these HPO methods. Lastly, the use of CPUs for building the ML models contributed to the lengthy duration of certain tuning processes. With the current trend where GPUs offer superior performance in model building, future research could utilize libraries like cuML for GPU-accelerated ML models, potentially saving significant time and computational resources.

7. CONCLUSION

In conclusion, this study utilized a credit card churning dataset to conduct a comparative analysis between Half Grid Search, Simulated Annealing, Genetic Algorithm, and Grid Search. The implementation of the CRISP-DM methodology facilitated a smooth progression of the Hyperparameter Optimization experiment. Four models were developed for binary classification: Logistic Regression, Support Vector Classifier, Random Forest Classifier, and Gradient Boosting Classifier. These models were then optimized using the four hyperparameter tuning methods to evaluate them in terms of computation time, quality of hyperparameter solutions, and diversity in the hyperparameters found. The results indicated that the Genetic Algorithm outperformed the other methods in all three aspects, demonstrating the shortest optimization time, high solution quality, and greater diversity in hyperparameter findings. The Gradient Boosting Classifier, in particular, showed the most significant improvement after optimization with the Genetic Algorithm, achieving a remarkable ROC-AUC score of 0.9912. Ultimately, all models and experimental findings were published on a web application, enabling users worldwide to test the prediction models and utilize the hyperparameter tuning code in their local environments. The findings from the HPO experiment are intended to guide Machine Learning practitioners in selecting appropriate hyperparameter optimization methods for their models and to contribute to filling the research gaps in the comparative analysis of these four hyperparameter tuning methods.

8. ACKNOWLEDGEMENT

I would like to express my gratitude to all those who contributed to this project. This study aims to provide Machine Learning practitioners with a fundamental understanding of the potential of advanced hyperparameter optimization methods and to offer insights into their strengths and weaknesses compared to traditional Grid Search through this comparative analysis. Special thanks go to my supervisor, Associate Prof. Ts. Dr. Sri Devi A/P Ravana, and the collaborators from Ernst & Young (EY) for their invaluable guidance and advice during this HPO experiment. Lastly, I am grateful for the feedback and encouragement from my friends and family, which helped sustain my motivation until the completion of this project.

References

[1] Al-Shourbaji, I., Helian, N., Sun, Y., Hussien, A. G., Abualigah, L., & Elnaim, B. M. E. (2023). An Efficient Churn Prediction Model Using Gradi-

ent Boosting Machine and Metaheuristic Optimization. *Scientific Reports, 13*(1). https://doi.org/10.1038/s41598-023-41093-6

[2] Bandgar, S. (2021, May 28). *Data Preparation in Data Science*. Medium. https://medium.com/analytics-vidhya/data-preparation-in-data-science-16f9311760

[3] Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A., Deng, D., & Lindauer, M. (2023). Hyperparameter Optimization: Foundations, Algorithms, Best Practices, and Open Challenges. *WIREs Data Mining and Knowledge Discovery, 13*(2), e1484. https://doi.org/10.1002/widm.1484

[4] Blume, S., Benedens, T., & Schramm, D. (2021). Hyperparameter optimization techniques for designing software sensors based on artificial neural networks. *Sensors, 21*(24), 8435. https://doi.org/10.3390/s21248435

[5] Claesen, M., & De Moor, Bart. (2015). *Hyperparameter Search in Machine Learning*. ArXiv.org. https://arxiv.org/abs/1502.02127

[6] Cooper, A. F., Lu, Y., Forde, J. Z., & De Sa, C. (2021). *Hyperparameter Optimization Is Deceiving Us, and How to Stop It.* ArXiv.org. https://doi.org/10.48550/arXiv.2102.03034

[7] Coroiu, A. M. (2016). Tuning Model Parameters Through a Genetic Algorithm Approach. *2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP)*, 135-140. https://doi.org/10.1109/ICCP.2016.7737135

[8] Coussement, K., & Van Den Poel, D. (2008). Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. *Expert Systems With Applications, 34*(1), 313–327. https://doi.org/10.1016/j.eswa.2006.09.038

[9] Du, X., Xu, H., & Zhu, F. (2021). Understanding the Effect of Hyperparameter Optimization on Machine Learning Models for Structure Design Problems. *Computer-Aided Design*, *135*, 103013. https://doi.org/10.1016/j.cad.2021.103013

[10] Ge, Z., Song, Z., Ding, S. X., & Huang, B. (2017). Data Mining and Analytics in the Process Industry: The Role of Machine Learning. *IEEE Access, 5*, 20590–20616. https://doi.org/10.1109/access.2017.2756872

[11] Ghazanfari, M., Alizadeh, S., Fathian, M., Koulouriotis, D. E. (2007). Comparing Simulated Annealing and Genetic Algorithm in Learning FCM. *Applied Mathematics and Computation, 192*(1), 56–68. https://doi.org/10.1016/j.amc.2007.02.144

[12] Goyal, S. (2020). *Credit Card Customers* [Data set]. https://www.kaggle.com/datasets/sakshigoyal7/credit-card-customers

[13] Guo, B., Hu, J., Wu, W., Peng, Q., & Wu, F. (2019). The Tabu_Genetic Algorithm: A Novel Method for Hyper-Parameter Optimization of Learning Algorithms. *Electronics, 8*(5), 579. https://doi.org/10.3390/electronics8050579

[14] Hoque, K. E., & Aljamaan, H. (2021). Impact of Hyperparameter Tuning on Machine Learning Models in Stock Price Forecasting. *IEEE Access, 9*, 163815-163830. https://doi.org/10.1109/access.2021.3134138

[15] Hotz, N. (2023). CRISP-DM Diagram [Online image]. Data Science Process Alliance. https://www.datascience-pm.com/wp-content/uploads/2021/02/CRISP-DM.png

[16] Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). *Automated Machine Learning: Methods, Systems, Challenges. The Springer Series on Challenges in Machine Learning*. Springer International Publishing. https://doi.org/10.1007/978-3-030-05318-5

[17] Jeroen, V. H., Joaquin, V. (2021, January 6). *Hyperboost: Hyperparameter Optimization by Gradient Boosting Surrogate Models*. arXiv.org. https://arxiv.org/abs/2101.02289

[18] Kerr, A., & Mullen, K. (2019). A Comparison of Genetic Algorithms and Simulated Annealing in Maximizing the Thermal Conductance of Harmonic Lattices. *Computational Materials Science, 157*, 31–36. https://doi.org/10.1016/j.commatsci.2018.10.007

[19] Konuksal, S. (2018). Credit Card Churn Prediction with Machine Learning Algorithms. https://openaccess.mef.edu.tr/xmlui/handle/20.500.11779/1183

[20] Kozarovytska, P. & Kucherenko, T. (2023). Empirical Comparison of Hyperparameter Optimization Methods for Neural Networks. *Master's Symposium on Advances in Data Mining, Machine Learning, and Computer Vision.* https://www.researchgate.net/publication/369597344_Empirical_comparison_of_hyperparameter_optimization_methods_for_neural_networks

[24] Lin, S., Lee, Z., Chen, S., & Tseng, T. (2008). Parameter Determination of Support Vector Machine and Feature Selection using Simulated Annealing Approach. *Applied Soft Computing, 8*(4), 1505–1512. https://doi.org/10.1016/j.asoc.2007.10.012

[22] Nie, G., Rowe, W., Zhang, L., Tian, Y., & Shi, Y. (2011). Credit card churn forecasting by logistic regression and decision tree. *Expert Systems With Applications, 38*(12), 15273–15285. https://doi.org/10.1016/j.eswa.2011.06.028

[23] Petro, L., & Pavlo, L. (2019). *Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS*. ArXiv.org. https://doi.org/10.48550/arXiv.1912.06059

[24] Polgár, O., Fried, M., Lohner, T., & Bársony, I. (2000). Comparison of Algorithms used for Evaluation of Ellipsometric Measurements Random Search, Genetic Algorithms, Simulated Annealing and Hill Climbing Graph-Searches. *Surface Science, 457*(1–2), 157–177. https://doi.org/10.1016/s0039-6028(00)00352-6

[25] Ramos-Pérez, I., Arnaiz-González, Á., RodrıGuez, J. J., & García-Osorio, C. (2022). When is Resampling Beneficial for Feature Selection with Imbalanced Wide Data? *Expert Systems With Applications, 188*, 116015. https://doi.org/10.1016/j.eswa.2021.116015

[26] Rocca, J. (2021, May 23). *The Exploration-exploitation Trade-off: Intuitions and Strategies*. Towards Data Science. https://towardsdatascience.com/the-exploration-exploitation-dilemma-f5622fbe1e82

[27] Sagiroglu, S., & Sinanc, D. (2013). Big Data: A Review. *2013 International Conference on Collaboration Technologies and Systems (CTS)*, 42–47. https://doi.org/10.1109/cts.2013.6567202

[28] Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science, 2*(3), 160. https://doi.org/10.1007/s42979-021-00592-x

[29] Soleymani, A. (2023, February 13). *Grid Search and Random Search are Outdated. This Approach Outperforms Both*. Medium. https://medium.com/@ali.soleymani.co/stop-using-grid-search-or-random-search-for-hyperparameter-tuning-c2468a2ff887

[30] Schröer, C., Kruse, F., & Gómez, J. M. (2021). A Systematic Literature Review on Applying CRISP-DM Process Model. *Procedia Computer Science, 181*, 526–534. https://doi.org/10.1016/j.procs.2021.01.199

[31] Thankachan, K. (2022, August 9). What? When? How?: ExtraTrees Classifier - Towards Data science. *Medium*. https://towardsdatascience.com/what-when-how-extratrees-classifier-c939f905851c

[32] Tsai, C., & Fang, Z. (2021). An Effective Hyperparameter Optimization Algorithm for DNN to Predict Passengers at a Metro Station. *ACM Transactions on Internet Technology, 21*(2), 1–24. https://doi.org/10.1145/3410156

[33] Vincent, A. M., & Jidesh, P. (2023). An Improved Hyperparameter Optimization Framework for AutoML Systems using Evolutionary Algorithms. *Scientific Reports, 13*(1). https://doi.org/10.1038/s41598-023-32027-3

[34] Wainer, J., & Fonseca, P. (2021). How to tune the RBF SVM hyperparameters? An empirical evaluation of 18 search algorithms. Artificial Intelligence Review, 54(6), 4771–4797. https://doi.org/10.1007/s10462-021-10011-5

[35] Wirth, R. & Hipp, J. (2000, April 11-13). *CRISP-DM: Towards a Standard Process Model for Data Mining* [Paper presentation]. Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining, Manchester. https://cs.unibo.it/ danilo.montesi/CBD/Beatriz/10.1.1.198.5133.pdf

[36] Wu, C., & Wang, L. (2022). A Comparative Analysis of Churn Prediction Models: A Case Study in Bank Credit Card. *Journal of Supply Chain and Operations Management, 20*(2), 120-138. https://www.csupom.com/uploads/1/1/4/8/114895679/n20p7formatted.pdf

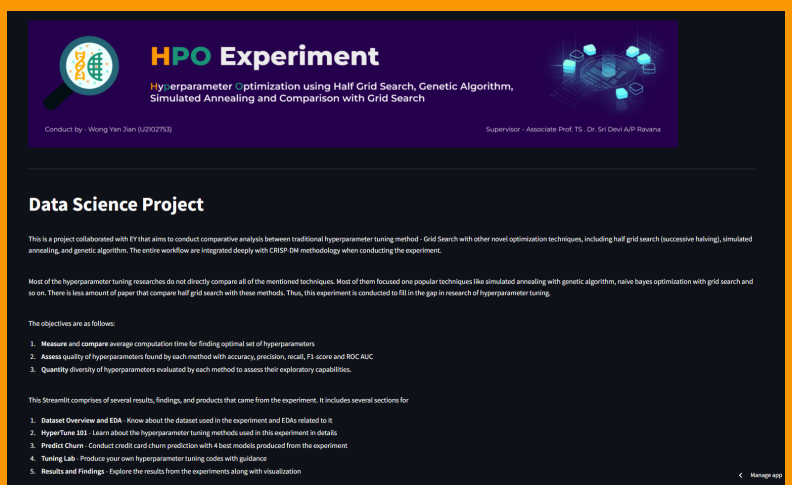
[37] Yang, L., & Shami, A. (2020). On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice. *Neurocomputing, 415*, 295–316. https://doi.org/10.1016/j.neucom.2020.07.061

HPO Experiment Streamlit User Manual

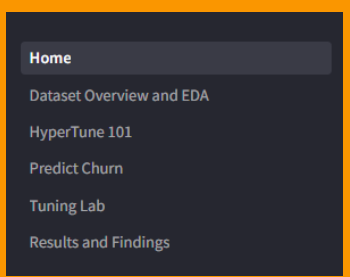
Prepared by - Wong Yan Jian (U2102753)

https://hpoexperiment-8wze5k5k6qpuscgwk7qmst.streamlit.app/

1. Home Page



This is homepage where you first landing to the web application. It introduces about the HPO experiment, including its problem statement, objectives, and other webpages usages

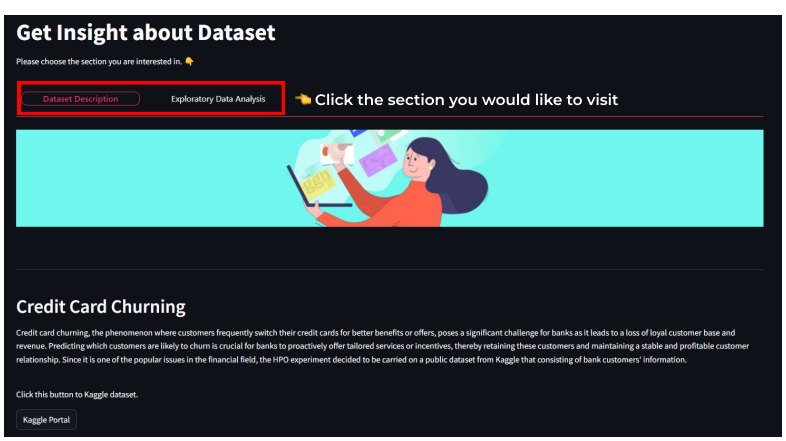


On the left side, you will see the navigation bar

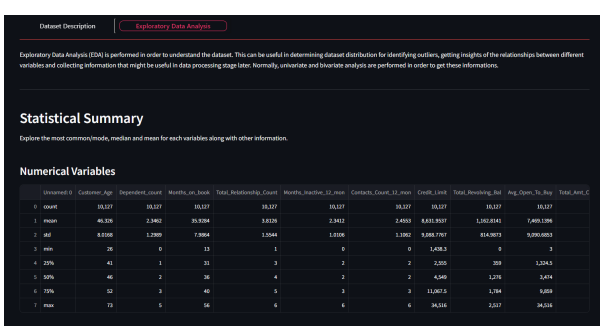
You can select the page you interested in and check them out 👀

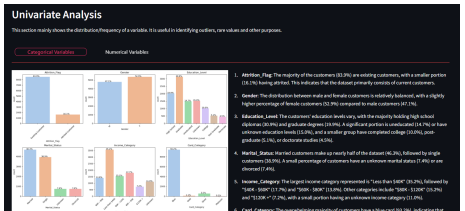
2. Dataset Overview and EDA Page

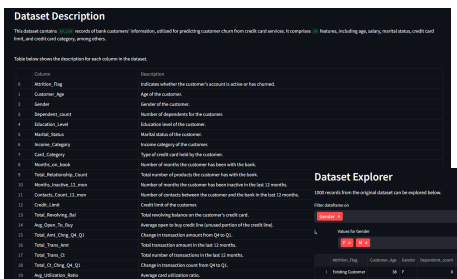
This page allows you to understand, explore and see the results from EDA. It contains 2 sections, which are Dataset Description and Exploratory Data Analysis



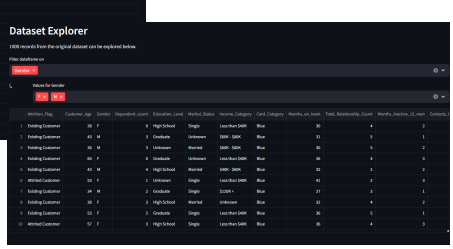
EDA section will display the result from statistical summary, univariate and bivariate analysis



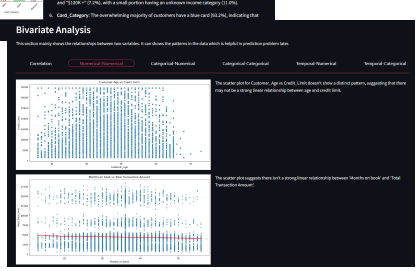




👈 These are the description for each column in the dataset



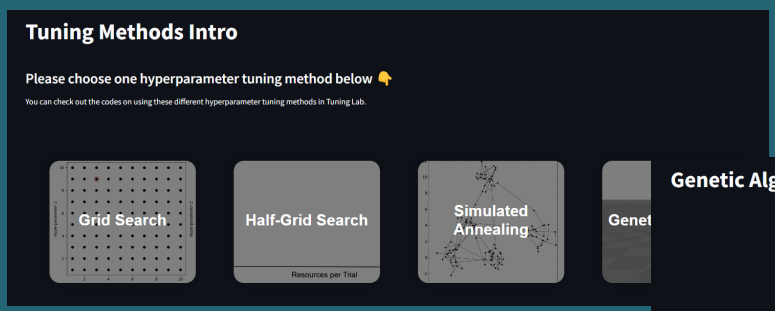
Further scrolling down, you can explore the dataset by filter the columns and values 🔎



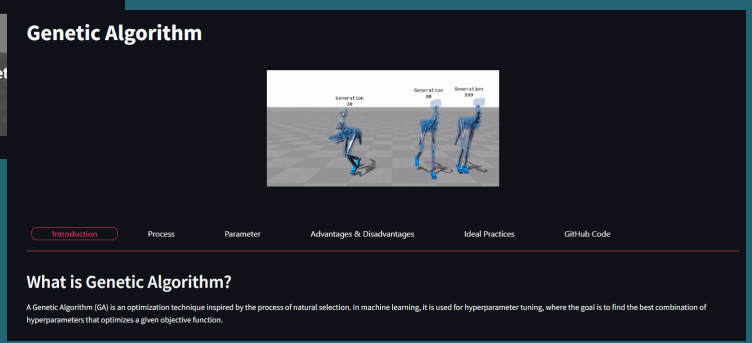
📊 Each graph is shown with its interpretation

3. HyperTune 101 Page

This page will teach you about the hyperparameter optimization method



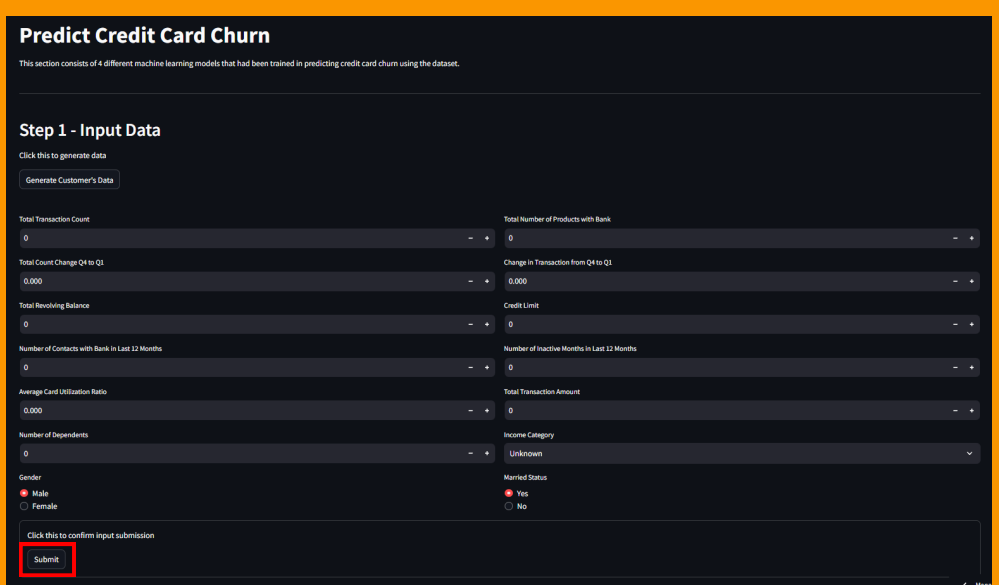
It consists of contents such as introduction, process, codes, and others for the selected method



Select one hyperparameter optimization method to start your learning journey 💫

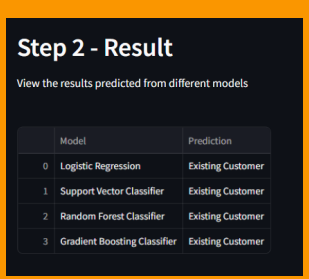
4. Predict Churn Page

This page allows user to input or generate 14 data pieces for predicting credit card churning with four machine learning models



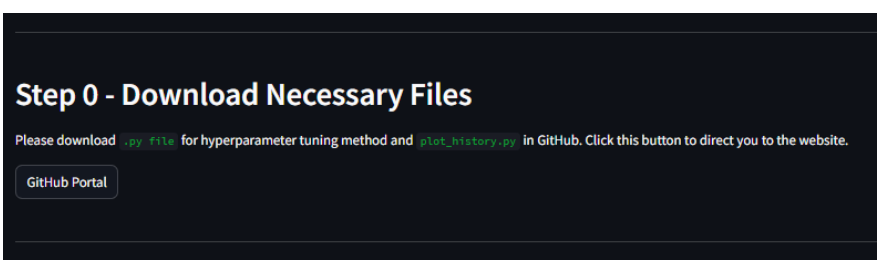
Instructions

1. Fill in the data or click 'Generate Customer's Data' button
2. Click 'Submit' button
3. Check your result 🤗



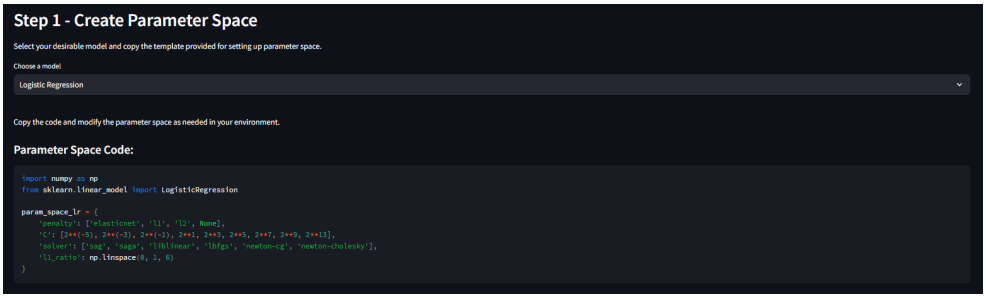
5. Tuning Lab Page

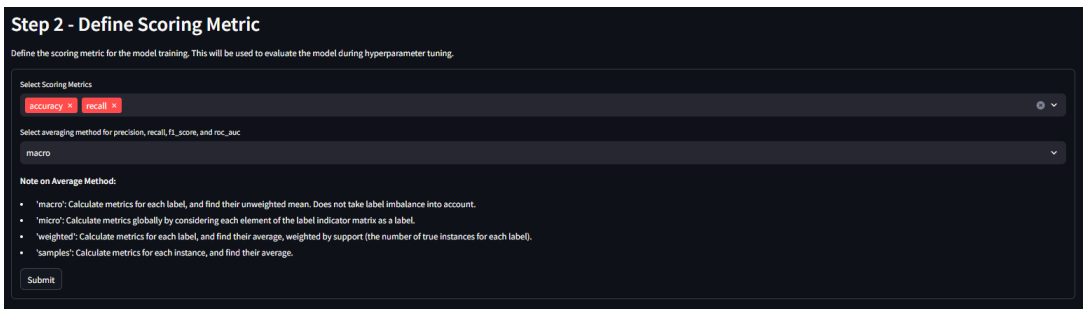
This page will generate template codes to help you using hyperparameter optimization method in your local environments



👈 First, you need to download python files from GitHub

Next, choose your preferable model. It will automatically generate the parameter space code 🤫



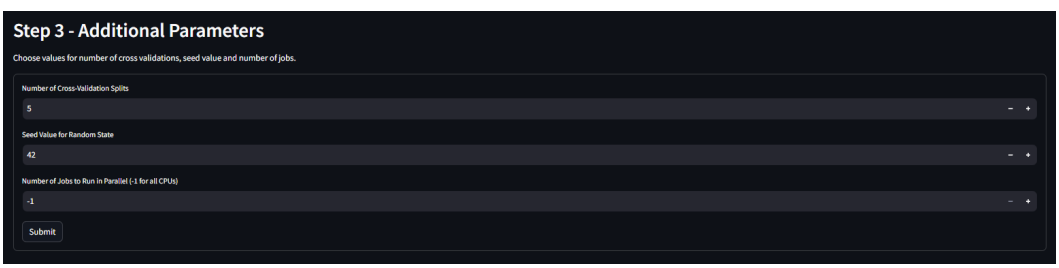


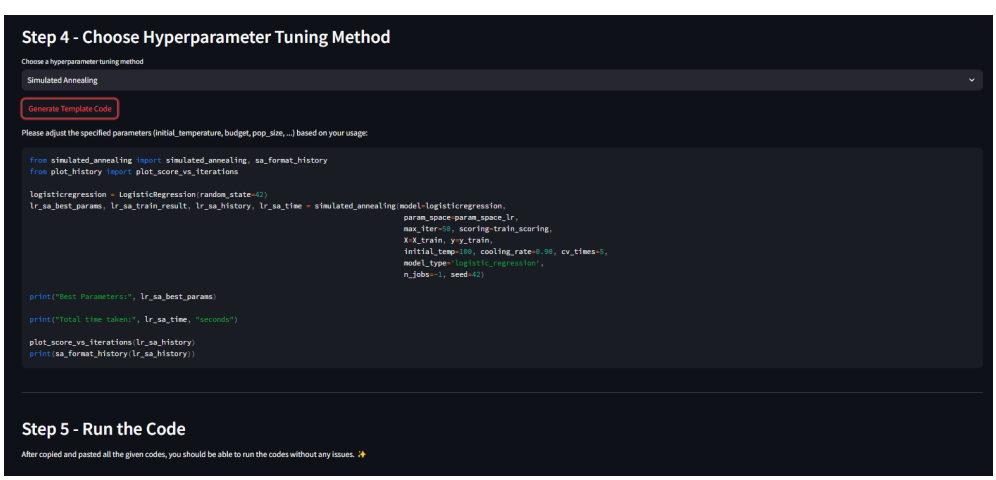
After that, specify your scoring metrics and click 'Submit' button 💯

It will generate code for selected scoring metric

Then, define the additional parameters and don't forget to click 'Submit' button

Again, code will be generated



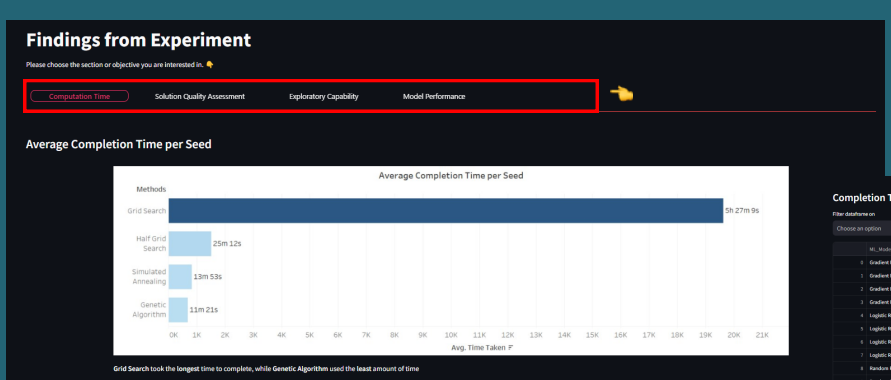


Lastly, select your preferable hyperparameter tuning method and click 'Generate Template Code' 🤖

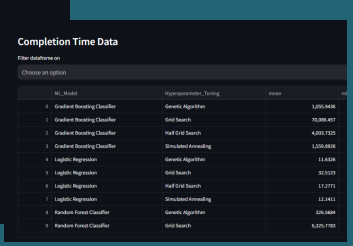
Copy all of them into your local environment and now you can run the hyperparameter optimization code 🤩

6. Results and Findings Page

This page contains the results from HPO experiment and interpretation for each graph



Users can explore different findings by selecting the section



At the end of page, the data gathered from each finding can be explored