USER MANUAL

OPTIMA

(OPTimized Interpretable

Model Building and Analysis Toolkit)

Laboratory of drug design and discovery,

Dept. of Pharmaceutical Technology,

Jadavpur University, Kol-700032.

Table of Contents

OPTIMA (OPTimized Interpretable Model Building and Analysis Toolkit) is a comprehensive toolkit designed to streamline and accelerate the development of interpretable ML-based classification models. The toolkit was developed entirely based on the Python programming language, and it provides a user-friendly GUI that significantly reduces the time and effort required to optimize, build, and interpret ML models. The primary goal behind the development of the Optima toolkit is to support the creation of robust and optimized ML-based classification models, along with their interpretation. The present version of OPTIMA comprises five core modules, each representing an important stage in the ML workflow: (1) Dataset Division algorithm, (2) Feature selection strategy, (3) Optimization of necessary hyperparameters, (4) Construction of different ML-based classification models, and (5) Model Interpretability (using SHAP Analysis). In addition to these core functionalities, we have also developed a settings panel that enables complete customization of all the generated plots according to user requirements. This panel supports modifications in parameters like colour schemes, axis labels, font sizes, plot dimensions, etc. Such flexibility not only enhances the visual clarity and interpretability of the results but also enables seamless integration into academic publications, reports, or presentations.

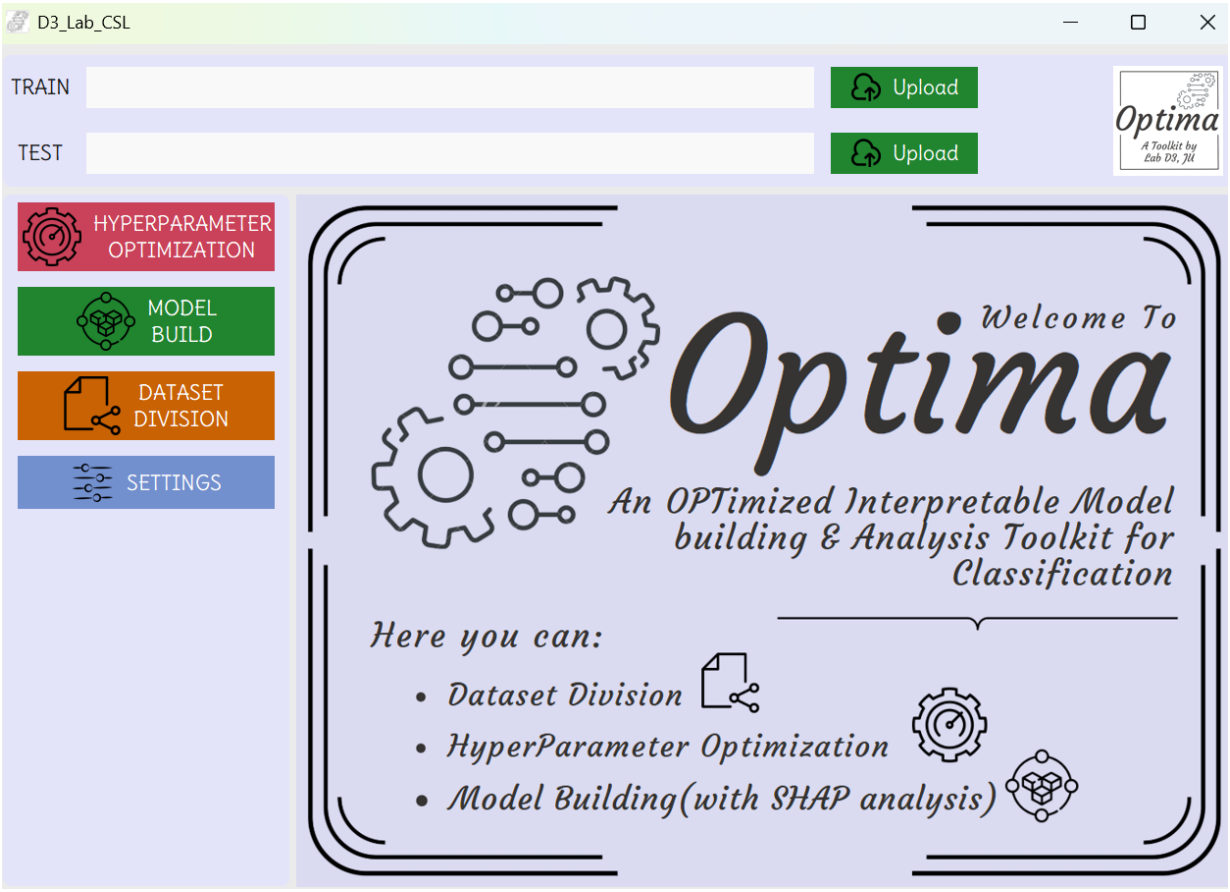

Fig 1. Homepage interface of the Optima Toolkit

Installation

To download the latest version of the Optima toolkit, users can refer to this link: https://github.com/Rahul-Roy-21/OPTIMA

Features

Dataset Division panel

The Optima toolkit provides users with flexible options for dataset partitioning, including both random and rational approaches such as Kennard-Stone and activity/property-based division, along with the capability to incorporate user-defined training and test sets. In scenarios where predefined training and test sets are available, users can directly tag individual samples as "TRAIN" or "TEST" within the interface. Alternatively, users may choose to perform automated dataset splitting based on either random sampling or rational selection algorithms.

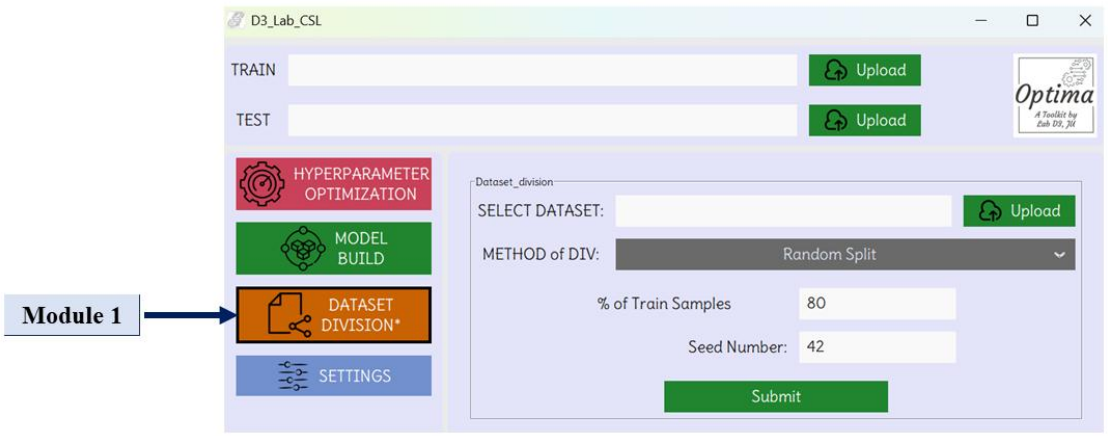

Fig 2. Dataset division algorithm panel for Random Splitting

Random division:

In the case of random splitting, the user specifies the desired percentage for the training set in the "% of Train samples" option, and various partitions can be generated by adjusting the seed number value, thereby enhancing robustness through repeatability.

For more controlled and chemically meaningful dataset partitioning, the Optima toolkit incorporates two rational division methods: the Kennard–Stone (KS) algorithm and the activity/property/toxicity-based splitting approach.

Kennard–Stone (KS) algorithm

In the case of the Kennard–Stone algorithm, the division is based on Euclidean distances between two data points. Initially, the two most dissimilar compounds (i.e., those with the greatest Euclidean distance) are selected as seeds for the training set. Subsequently, additional training samples are iteratively selected to maximize the minimum distance between already chosen training compounds and the remaining pool. This procedure ensures the inclusion of chemically diverse samples in the training set until the user-specified proportion is reached. The remaining samples are allocated to the test set. This method is particularly effective for constructing representative and diverse training sets in small to moderate-sized datasets, where the preservation of structural and activity space diversity is critical. Users can input their desired percentage of training samples in the "% of Train Samples" option under the Kennard-Stone division algorithm panel.

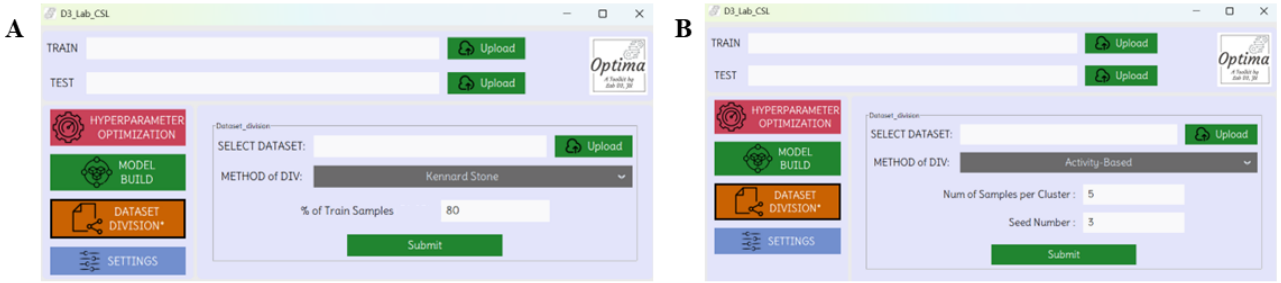

Fig 3. Dataset division algorithm panel for (A) Kennard–Stone (KS) algorithm and (B) Activity-based approach

Activity/Property/Toxicity-based splitting approach

The activity-based division approach is another rational method for dataset partitioning, which is available in our Optima toolkit. This method of dataset division ensures a representative distribution of biological activity/response values across both the training and test sets. This method is particularly useful for small to moderate-sized datasets where preserving the diversity and range of activity is crucial for developing robust and generalizable models. In this approach, compounds are first systematically sorted in ascending order based on their biological activity/response values. The sorted dataset is then divided into n clusters, where each cluster contains a fixed/predetermined number of compounds. This clustering ensures that the entire activity range is evenly segmented. Once the clusters are formed, compounds from each cluster are systematically and alternately assigned to either the training or test set. This stratified assignment ensures that both subsets capture the full spectrum of activity values, thereby preventing bias of specific activity ranges in either set.

Guidelines for the Input Dataset

In this software, users must provide an input dataset in .xlsx format containing the following mandatory information: a chemical ID or serial number (user-defined) in the first column, a response variable in the last column, and a set of descriptors/features in between the serial number and response columns. The dataset must be properly formatted before use. The prepared file can be conveniently uploaded using the "Upload" button located within the panel.

Guidelines for the Input Train and Test Set Files

Once the dataset has been properly divided into training and test sets, users can upload the individual .xlsx files through the interface by tagging them as "TRAIN" or "TEST" accordingly. The format of both files must follow the previously described input dataset structure, including mandatory columns such as Chemical ID, response variable, and descriptors/features.

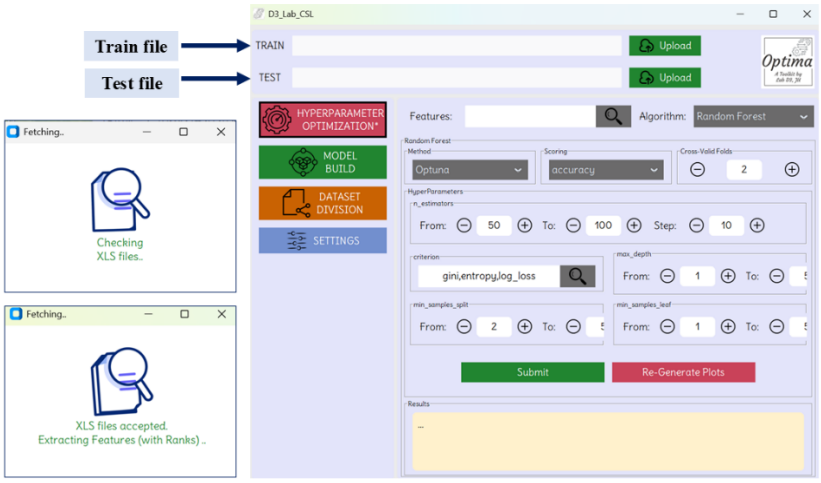

Fig 4. Panel for uploading the training and test sets.

Upon upload, the system will automatically verify that the feature (descriptor) sets in both the TRAIN and TEST files are identical. This verification includes if both file have:

- Same feature names
- Same number of features
- Consistent data types and formats

If the validation is successful, the files will be accepted, and the system will proceed to the feature selection stage, based on the method selected by the user. If any inconsistency is

detected during the check, an error message will prompt the user to correct the discrepancies before proceeding.

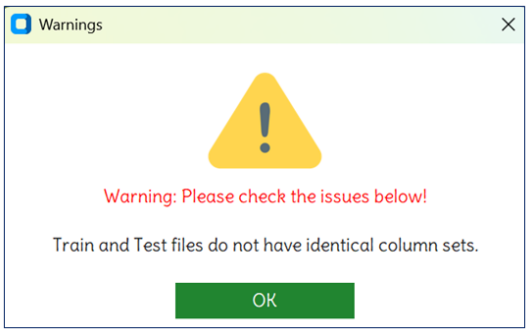

Fig 5. Warning box

References:

1. Kennard RW, Stone LA. Computer aided design of experiments. Technometrics. 1969 Feb 1;11(1):137-48.
2. Martin TM, Harten P, Young DM, Muratov EN, Golbraikh A, Zhu H, Tropsha A. Does rational selection of training and test sets improve the outcome of QSAR modeling?. Journal of chemical information and modeling. 2012 Oct 22;52(10):2570-8.

Feature selection panel

Before successfully uploading both the TRAIN and TEST .xlsx files, users can proceed to the Settings panel to configure feature selection options. Within the settings panel, an option labeled feature_selection.ranking_method is available. This allows users to choose the desired method for ranking features. The two available options are:

- MDF – Most discriminating feature selection algorithm
- MIS – Mutual Information Score

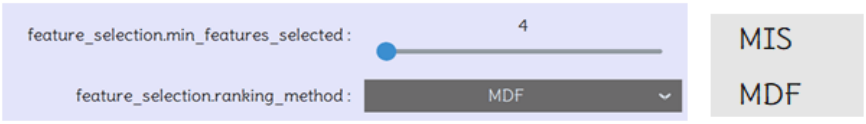

Fig 6. Algorithm selection panel for feature identification

Users can select either method based on their preference or the specific requirements of their model. Additionally, users can specify how many top-ranked features should be selected by the tool using the feature_selection.min_features_selected option in the settings panel. This option allows users to define the minimum number of features to retain after the feature selection process, based on the chosen ranking method. The tool will

automatically select the specified number of highest-ranked features accordingly. Once the method is selected, the system will automatically perform feature ranking accordingly.

NOTE: Once the system successfully validates the format and consistency of the uploaded TRAIN and TEST .xlsx files, it immediately proceeds to the feature selection process. Therefore, users are strongly advised to select the desired feature selection method in the settings panel (under feature_selection.ranking_method) before uploading the TRAIN and TEST files. This ensures that the appropriate ranking method- either MDF or MIS is applied during the feature selection step without interruption. Failure to select a method in advance may result in the default selection of the tool.

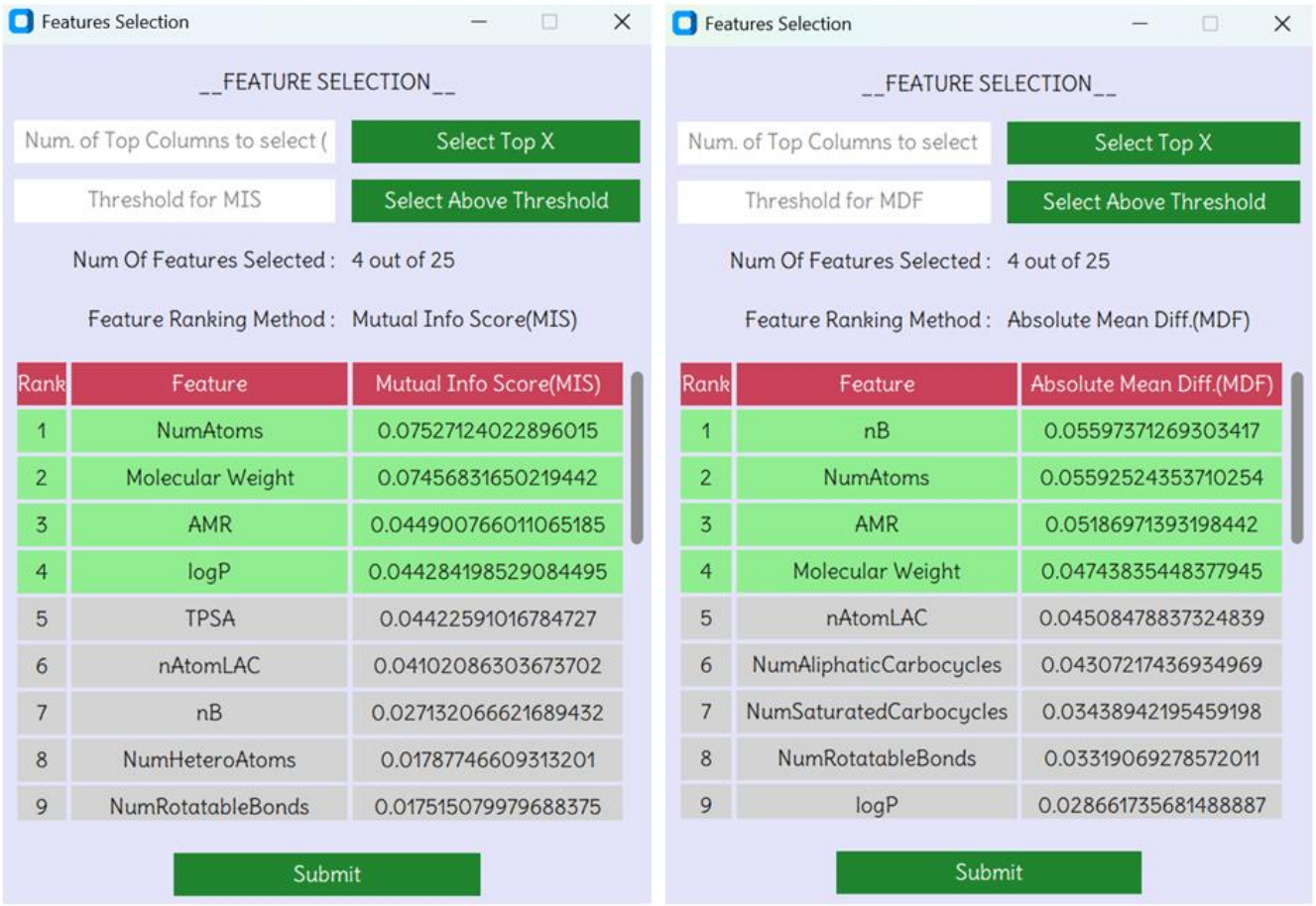

Fig 7. Feature selection panel (MDF/ MIS)

MDF feature selection strategy

The Most discriminating feature (MDF) selection algorithm systematically identifies features that exhibit the highest discriminatory power between two predefined classes. Initially, all feature values in the training dataset are normalized to the range [0, 1], ensuring consistent scaling across different feature magnitudes. The dataset is then partitioned into

two classes based on their corresponding response values. For each feature, the algorithm computes the class-wise mean i.e., the average value of the feature within each class. The absolute difference between these class means is then calculated for every feature. This absolute difference reflects the degree to which a feature can differentiate between the two classes. Features with larger absolute differences are considered as more discriminative and are prioritized for selection.

MIS feature selection strategy

The Mutual Information Score strategy is a widely used approach for the identification of relevant features. It quantifies the mutual dependence between individual features and the target variable, offering a non-linear measure of association. For each feature, the mutual information (MI) score with respect to the class labels is computed. This score reflects how much information about the target variable is gained by knowing the value of a particular feature. Once all MI scores are calculated, the features are ranked in descending order based on their MI values. Features with the highest mutual information scores are considered the most informative, as they contribute significantly to reducing uncertainty about the target class. This method is robust to non-linear relationships and applies to both binary and multiclass classification scenarios

Based on the selected cut-off in either of the feature selection methods- 'MDF' or 'MIS'- the user can choose to input a threshold value in the 'Select Above Threshold' panel or directly specify the desired number of top features using the 'Select Top X' option. Once the cut-off is properly selected, the panel will display the number of selected features, indicating how many will be used in the subsequent hyperparameter optimization or model development process.

References:

1. Khatun S, Dasgupta I, Sen S, Amin SA, Qureshi IA, Jha T, Gayen S. Histone deacetylase 8 in focus: Decoding structural prerequisites for innovative epigenetic intervention beyond hydroxamates. International Journal of Biological Macromolecules. 2025 Jan 1;284:138119.
2. Dasgupta I, Das T, Das B, Gayen S. Identification of structural features of surface modifiers in engineered nanostructured metal oxides regarding cell uptake through ML-based classification. Beilstein Journal of Nanotechnology. 2024 Jul 22;15(1):909-24.

3. Wang X, Guo B, Shen Y, Zhou C, Duan X. Input feature selection method based on feature set equivalence and mutual information gain maximization. IEEe Access. 2019 Oct 17;7:151525-38.

Customizable Objective Function

An important feature of the toolkit is that users can define their preferred objective function, selecting from a wide array of scoring metrics according to the nature of their classification task. The toolkit offers a comprehensive range of scoring methods to evaluate model performance across various classification tasks. These include commonly used metrics such as accuracy, balanced accuracy, and average precision. It also supports a suite of F1 scores, namely f1, f1_micro, f1_macro, and f1_weighted, to provide a balanced measure of precision and recall across different class distributions. Additionally, the toolkit includes metrics like precision, recall, Jaccard index, and several forms of ROC AUC scores, including roc_auc, roc_auc_ovr (one-vs-rest), roc_auc_ovo (one-vs-one), and their weighted variants (roc_auc_ovr_weighted and roc_auc_ovo_weighted).

Cross-Validation Configuration

The Optima toolkit also allows users to specify the preferred number of cross-validation folds (CV) to be used during hyperparameter optimization. This flexibility ensures that the model's generalizability and robustness are thoroughly evaluated on different subsets of the training data.

Hyperparameter optimization panel

Optimizing the necessary hyperparameters is a crucial step in developing any ML-based classification model, as it significantly impacts model performance. If hyperparameters are chosen improperly, they can result in models that either underfit the data or overfit it, thereby reducing the model's effectiveness on validation data. Different techniques, such as Grid Search CV, Randomized Search CV, and more advanced methods like Bayesian Optimization, systematically explore the hyperparameter space to identify optimal configurations. Our toolkit, Optima, offers three distinct techniques for hyperparameter optimization: Grid Search CV, Randomized Search CV, and specially advanced methods like Optuna. These approaches systematically explore the hyperparameter space to identify the most effective configurations for model performance.

Grid Search CV method

Grid Search CV is one of the traditional and exhaustive techniques for hyperparameter tuning. It works by evaluating the model on every possible combination of hyperparameters defined in a user-specified grid. For instance, if a user wants to tune two parameters and provides three values for each, Grid Search will evaluate all 3×3 = 9 combinations. The major advantage of the Grid Search algorithm is that it guarantees that all possible combinations are tested, so the global optimum (within the grid) will be found if it exists. Its simplicity and transparency make it particularly appealing for beginners or for tackling smaller-scale problems, where interpretability and ease of implementation are just as important as performance. But there are some disadvantages also, which include high computational cost. The number of combinations increases exponentially with more hyperparameters or values, making grid search computationally unfavourable for large search spaces.

However, this challenge can be effectively addressed using advanced hyperparameter optimization techniques such as the Optuna method, which reduces computational cost by pruning unpromising trials during the search process. A more detailed explanation of this approach is provided later.

Randomized Search CV method

Another hyperparameter optimization strategy, Randomized Search CV, improves upon Grid Search CV by sampling a fixed number of hyperparameter combinations at random from specified distributions. This stochastic approach allows users to explore a wider range of values without exhaustively testing every possibility. It can handle large and complex search spaces better because it avoids evaluating every single possibility. Since the combinations are chosen at random, there is a chance that the optimal configuration could be missed, especially if the number of iterations is too low. As this is a randomized search approach, results can vary between runs, making it harder to reproduce outcomes.

These limitations can be effectively addressed using an Optuna-based hyperparameter optimization framework. Optuna intelligently explores the search space and utilizes techniques like pruning and efficient sampling to focus on promising regions. Further, increasing the number of trials enhances the possibility of identifying the most accurate and optimal hyperparameter configuration.

Optuna-based hyperparameter optimization

Optuna is an advanced framework for hyperparameter optimization based on Bayesian optimization principles. It builds a probabilistic model of the objective function and uses this model to select the set of hyperparameters. The goal is to balance exploration (trying new areas of the hyperparameter space) and exploitation (refining areas known to perform well). One of the major advantages of using this strategy is that Optuna significantly reduces the number of trials needed to find a near-optimal solution by focusing the search on promising regions of the space, instead of wasting resources on poor-performing combinations. Additionally, it supports dynamic search spaces, conditional hyperparameters (where the choice of one parameter influences another), and pruning of unpromising trials, which saves time.

These capabilities enable Optuna to explore the search space more intelligently and efficiently than Grid Search CV and Randomized Search CV, ultimately leading to faster convergence toward optimal hyperparameter configurations. The current algorithm is well-suited for high-dimensional and complex models.

Another notable advantage of the Optuna-based hyperparameter optimization in the toolkit is its ability to save the complete trial history into a SQLite database file. This feature ensures that all trial results are systematically recorded and can be easily accessed later for further analysis, visualization, or reporting. It greatly facilitates the generation of plots at any stage of the workflow and plays a crucial role in maintaining reproducibility and transparency throughout the optimization process.

By offering all three strategies, the Optima tool equips users with flexible ways for model tuning, allowing them to choose the most appropriate approach depending on the specific context and requirements of their machine learning workflow. This toolkit stands out as a comprehensive and user-friendly solution for machine learning model optimization, offering support for seven widely used algorithms: Random Forest (RF), Support Vector Machine (SVM), k-nearest neighbors (KNN), Gradient Boosting (GB), Logistic Regression (LR), Linear Discriminant Analysis (LDA), and Multi-Layer Perceptron (MLP).

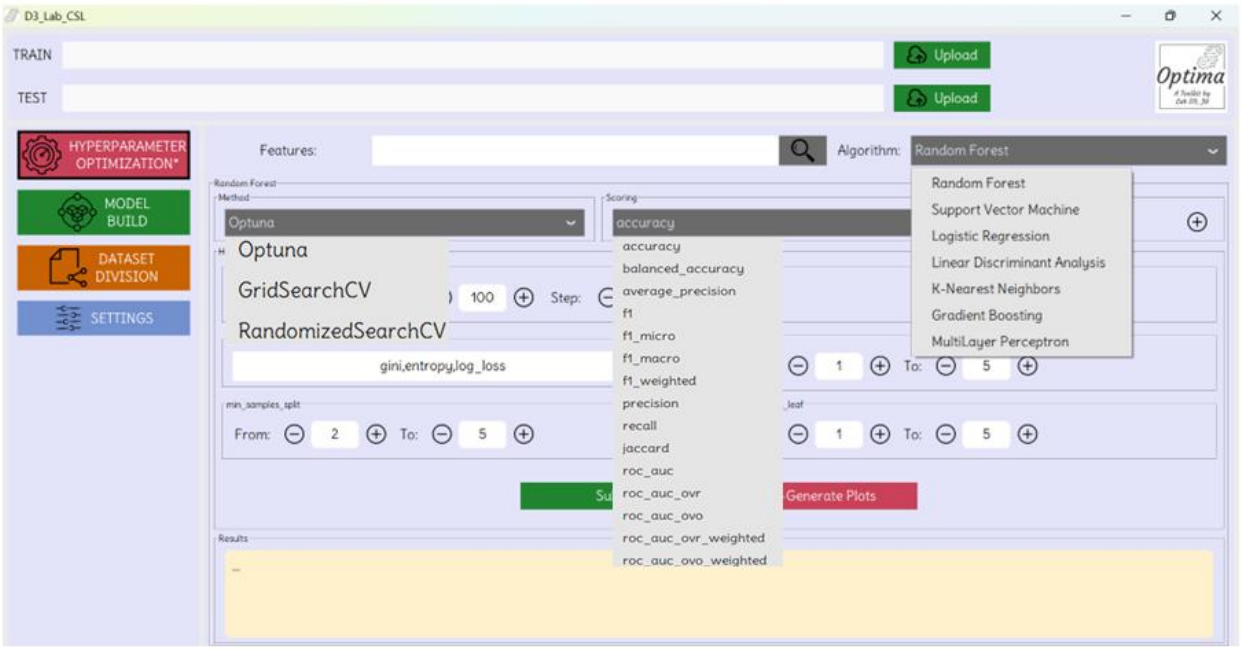

Fig 8. Hyperparameter optimization panel of the Optima toolkit

Visualization support for Optuna

One of the standout features of the Optima toolkit is its clear visualization support for the Optuna-based hyperparameter optimization process, offering an interactive and user-friendly interface to effectively track, analyze, and interpret the entire optimization workflow. This visualization module primarily generates five types of plots, which include optimization history plot, hyperparameter importance plot, parallel coordinate plot, slice plot, and contour plot.

i. **Optimization history plot**: This plot displays the progression of the selected objective value across all optimization trials. It provides a clear visual trend of how the model is improving with each iteration.

ii. **Hyperparameter importance plot**: This plot visualizes the relative impact of each hyperparameter on the optimization process. Ranking hyperparameters based on their contribution helps users to prioritize the tuning procedure for the most influential parameters, thereby reducing both time and computational cost.

iii. **Parallel coordinate plot**: This multidimensional visualization shows how different combinations of hyperparameters correlate with objective values. Each line represents a trial, passing through axes for each hyperparameter and the corresponding objective value. Patterns emerging from this plot reveal optimal parameter ranges and potential interdependencies between hyperparameters.

iv. **Slice plot**: Slice plots isolate individual hyperparameters to show how variations in a single parameter affect the objective value, keeping other parameters constant. These plots are especially useful for examining the local sensitivity of the model.

v. **Contour plot**: This two-dimensional heatmap visualizes the interaction between two selected hyperparameters and their combined effect on model performance. The contour plot helps users to identify synergistic effects between hyperparameters, guiding more informed optimization strategies.

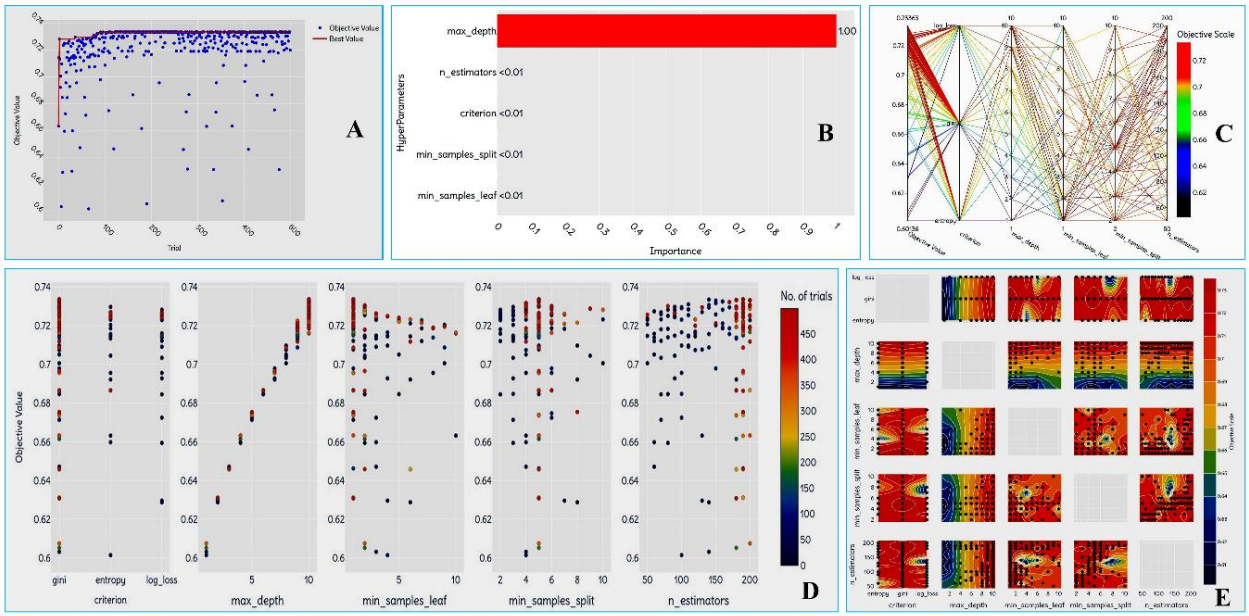

Fig 9. Hyperparameter optimization summary of the RF model in the Optima toolkit. Figs (A–E) represent various optimization plots generated during hyperparameter tuning of the ML-based classification models. (A) Optimization history plot, (B) Hyperparameter importance plot, (C) Parallel coordinate plot, (D) Slice plot, and (E) Contour plot

Together, these visualizations provide a comprehensive and intuitive framework for understanding the dynamics of hyperparameter tuning, enabling users to make data-driven decisions.

One notable advantage for users of the Optima toolkit is the flexibility to fully customize the generated optimization plots after the completion of all successful trials. If there are any issues with the colour scheme, plot labels, or other visual elements, users can access the "SETTINGS" panel to make adjustments. This panel allows for complete customization, including the selection of preferred colour schemes and the adjustment of plot height and width for each visualization plot individually. Once the desired changes are made, users should save the updated settings. Then, by simply clicking the "Re-Generate Plots" option

and uploading the corresponding study, users can recreate all five optimization plots. This functionality is enabled by the fact that the entire Optuna study is saved during the trial process, allowing users to regenerate plots at any time without the need to rerun the optimization. This also ensures reproducibility of the optimization results, as the saved study retains all trial data and parameter configurations.

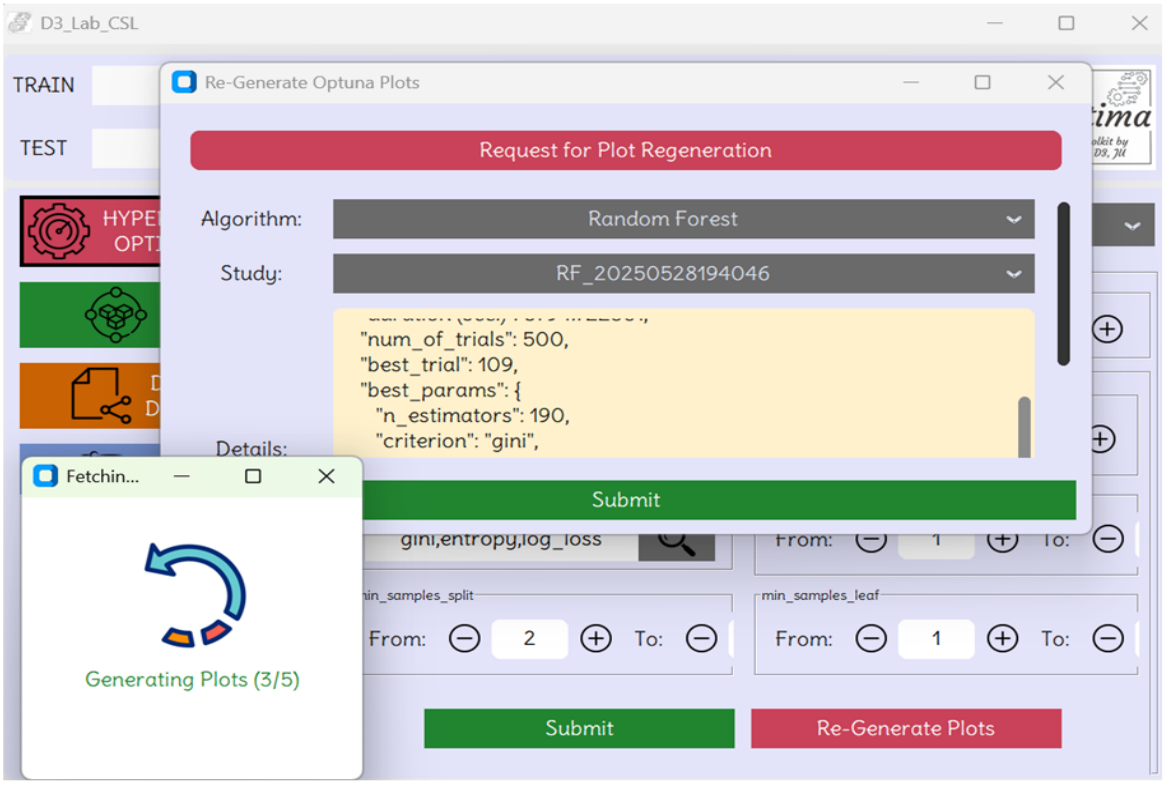

Fig 10. Panel for regenerating the visualization plots from Optuna-based hyperparameter optimization.

References

1. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html
3. https://optuna.org/
4. Akiba T, Sano S, Yanase T, et al. Optuna: a next-generation hyperparameter optimizationframework. In:Teredesai A, Kumar V, editors. Proceedings of the 25th ACM SIGKDDInternational Conference on Knowledge Discovery & Data Mining. (NY): Association forComputing Machinery; 2019. p. 2623–2631

Development of various ML-based classification models

The current module of the Optima toolkit is designed to simplify the development of different ML-based classification models. This toolkit supports seven of the most widely used ML algorithms, each accessible via a dedicated panel tailored to that method. The supported algorithms include Random Forest (RF), Support Vector Machine (SVM), k-nearest neighbors (KNN), Gradient Boosting (GB), Logistic Regression (LR), Linear Discriminant Analysis (LDA), and Multi-Layer Perceptron (MLP). After model training and evaluation, users receive a detailed performance report that includes key classification metrics such as accuracy, precision, recall, F1 score, Matthews correlation coefficient (MCC), and Cohen's kappa, along with ROC curves for visual performance assessment.

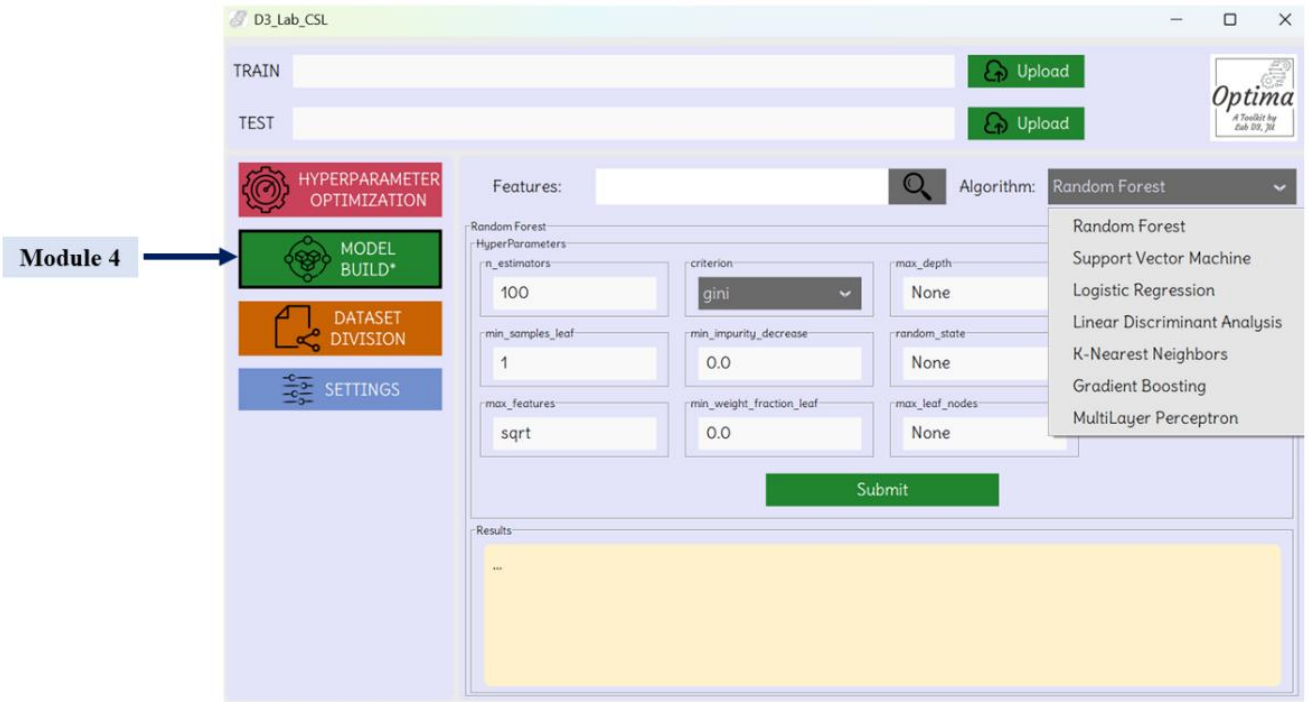

Fig 11. Panel for generating different ML-based classification models.

To enhance the visual clarity and publication quality of ROC (Receiver Operating Characteristic) curves, users are encouraged to customize the plot settings using the following controls available in the SETTINGS interface:

(a) Figure DPI (plot_properties.RC_PARAMS.FIGURE_DPI) - Adjust the resolution of the generated figure according to the choice of users.

(b) Font Size (plot_properties.RC_PARAMS.FONT_SIZE)- Set the size of all text elements in the plot (e.g., labels, titles, legends).

(c) Font Style (plot_properties.RC_PARAMS.FONT_STYLE)- Choose a preferred font family such as Times New Roman, Arial, etc.

(d) Font Weight (plot_properties.RC_PARAMS.FONT_WEIGHT)-Define the thickness of the font (e.g., normal, bold).

(e) ROC Curve Colour (plot_properties.ROC_CURVE.COLOR)- Select the colour of the ROC curve line.

(f) ROC Curve Line Width (plot_properties.ROC_CURVE.LW)- Customize the line thickness of the ROC curve.

SHAP plot

SHAP (SHapley Additive exPlanations) is an interpretability framework grounded in Shapley values from cooperative game theory, developed to quantify the contribution of each feature to a machine learning model's prediction. In the context of classification models, a SHAP summary plot provides a comprehensive visualization of feature importance and the direction of their influence on the predicted outcome. Features are ranked according to their mean absolute SHAP values, reflecting their overall contribution to model predictions. The plot employs a colour gradient to represent the actual feature values (e.g., high or low) for individual instances, while the horizontal dispersion indicates the magnitude of influence exerted by each feature across the dataset. The colour distribution along the horizontal axis further reveals whether higher or lower feature values are associated with an increased likelihood of a specific class prediction. This integrated visualization enables the rapid identification of the most influential features and facilitates a deeper understanding of the relationship between feature variation and model decision-making. In the Optima toolkit, SHAP summary plots for all classification models are automatically generated alongside the evaluation metrics and results, eliminating the need for any additional steps to produce these visualizations

However, we recommend that users select their preferred colour scheme in the "SETTINGS" panel before clicking the Submit button on the "Model Build" panel. This ensures the SHAP visualizations are not only informative but also aligned with users' visual preferences.

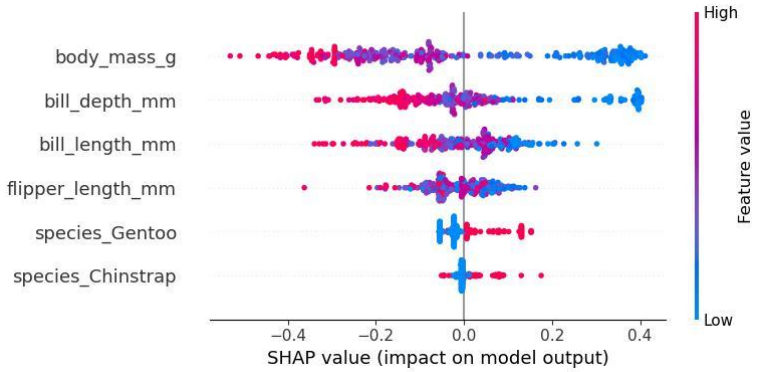

Fig 12. SHAP summary plot illustrating feature importance.

NOTE:

- We strongly recommend that all users thoroughly review and adhere to the official scikit-learn documentation before starting any model optimization or development. Following these guidelines ensures the proper application of ML algorithms, the selection of appropriate parameters, and improved reproducibility of results.
- If you encounter any issues or if the GUI displays an error, it is recommended to check the terminal for detailed messages. This can help identify the root cause of the problem and assist users in resolving it more efficiently.

References:

1. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
3. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
4. https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html
5. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
6. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
7. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
8. https://shap.readthedocs.io/