AutoBioLearn: An Automated Data Science Framework for eXplainable Analyses (XAI) of Clinical Datasets

Lucas P. B. Moreira[1], **Maria L. G. Kuniyoshi** [2], **Zofia Wicik**[3,4], **David C. Martins-Jr**[5], **Helena Brentani**[2], **Sérgio N. Simões**[1]

[1]PPCOMP – Instituto Federal do Espírito Santo – Brazil

[2]IPq – Faculdade de Medicina da Universidade de São Paulo – Brazil

[3]Dept. of Exp. & Clin. Pharmacology, CEPT, Med. Univ. Warsaw, Poland

[4]Dept. of Neurochemistry, Inst. of Psychiatry & Neurology, Warsaw, Poland

[5]CMCC – Universidade Federal do ABC – Brazil

lpbadaro55@gmail.com, sergio@ifes.edu.br

Abstract. *With the increasing volume of biological and medical data, the application of efficient data science techniques has become essential for analysis. However, healthcare data scientists often need to integrate and analyze multiple datasets simultaneously. Although these analyses share similarities, they require adjustments to various parameters, delaying development and further hindering knowledge discovery. In this paper, we propose a framework that encapsulates all stages of typical data science analyses, from data pre-processing, execution, and evaluation to the interpretation of models. In addition, the framework includes XAI analyses. In tests involving a clinical dataset, the framework achieved a reduction of 92% in lines of code.*

1. Introduction

1.1. Contextualization

During the past century, researchers in the health sciences have relied mainly on classical data analysis approaches. Statistical analyses have significantly expanded our understanding of diagnosis and treatment, informing critical decisions such as medication prescriptions and surgical interventions [MacEachern and Forkert 2021]. However, while these methods have significantly contributed to medical advancements, they frequently fail to achieve the level of precision required for accurate diagnoses and rapid decision-making, as they tend to overlook individual patient characteristics. With the increase in computational power and the growing volume of biological data collected from patients, recent studies have begun to apply novel methodologies to propose treatments with a higher degree of personalization [Duffy 2016]. These advancements have enabled the simultaneous and systematic analysis of diverse datasets [Lee et al. 2014], facilitating the integration and fusion of various types of data – including genomic profiling, medical imaging, laboratory tests, and demographic information [Alber et al. 2019].

Such methodology includes Artificial Intelligence (AI), particularly Machine Learning (ML) models [MacEachern and Forkert 2021, Ahmed et al. 2020], which have shown great potential in assisting with disease diagnosis and treatment by revealing complex patterns in patient data that may initially be hidden [Shailaja et al. 2018]. Some ML

models are considered black-boxes – i.e., they are too complex for human understanding. To address this, tools such as eXplainable Artificial Intelligence (XAI) have been developed to aid in model interpretation [Petch et al. 2022, Srinivasu et al. 2022]. The XAI approach complements ML data analysis by providing further insights into model behavior [Neto 2021]. However, even though XAI makes ML models more comprehensible, implementing these frameworks requires programming knowledge that often falls outside the skillset of many health sciences researchers. Additionally, even for trained ML and XAI researchers, building extensive pipelines demands a significant amount of time and is prone to human error [Alanazi 2022]. Furthermore, when the same pipeline must be applied to multiple inputs and hyperparameter configurations, the repetitive nature of the process increases both the likelihood of human error and the time required for setting up the experiments.

1.2. Related Works

Researchers have already proposed tools to mitigate the issues mentioned in the previous section. One example is graphical user interfaces, but none adequately integrates multiple datasets, compares the performance of ML models, or performs XAI to explain the results. For instance, Orange and KNIME are tools with drag-and-drop interfaces for constructing ML pipelines with preprocessing, exploratory analysis, model tuning, and training [Berthold et al. 2009, Demšar et al. 2013]. However, both lack features for XAI analysis, dataset balancing, and comparing the performance of models trained with different algorithms and hyperparameters. Additionally, Orange does not support recent ML models such as XGBoost, CatBoost, and LightGBM [Demšar et al. 2013, Chen and Guestrin 2016, Dorogush et al. 2018, Ke et al. 2017].

There are other tools that can perform analyses on biological data, such as: JADBio, H2O AutoML and TPOT. JADBio [Tsamardinos et al. 2022] is an AutoML platform focused on biomedical data, ideal for users without experience in data science. It facilitates the creation of predictive models, offering effective feature selection and model interpretability. However, JADBio has some limitations regarding the data science pipeline. It does not perform exploratory data analysis, nor does it allow for in-depth customized preprocessing. Additionally, it provides limited model explainability, lacking detailed tools like SHAP, and it does not offer comprehensive graphical comparisons between generated models. TPOT [Le et al. 2020] is an AutoML library that uses genetic algorithms to explore and optimize machine learning pipelines. It automates the preprocessing, feature selection, and hyperparameter tuning processes to find the best solution. However, TPOT does not perform exploratory data analysis, the data preprocessing is not fully automated, and does not offer model explainability or tools for graphical comparison between generated models.

H2O AutoML [LeDell and Poirier 2020] is a solution designed to automate the training and tuning of models, supporting a broad array of algorithms such as Random Forest and neural networks. It is well-suited for both novice and experienced data scientists, offering strong integration with Python and R. However, H2O AutoML is not tailored specifically for certain needs in biological data analysis, as it lacks features for handling hierarchical data structures, and systematically analyzing multiple input-output configurations. Additionally, while it covers many algorithms, it does not currently support some state-of-the-art models like LightGBM and CatBoost or provide consolidated

explainability through SHAP averaging.

In this paper, we propose ***AutoBioLearn***[1], a comprehensive framework that encompasses data preprocessing, model and hyperparameter selection, model training with cross-validation, performance evaluation, visualization, and XAI analysis. Notably, this framework simplifies the design of ML pipelines for data scientists by providing automated support for hierarchical data structures, input-output experiment configurations, and consolidated SHAP analysis – each tailored to meet common demands in biological research. *AutoBioLearn* also aids bioinformaticians who require systematic and repetitive analyses, minimizing human error through automation and reduced code volume while offering robust and interpretable insights into data. These features, offered by *AutoBioLearn*, can be beneficial for more complex, segmented datasets typical in biological research.

2. Key Concepts in Data Science

2.1. Data Preparation

Database preparation is crucial for making a valid input to the ML models. This step allows the identification and correction of data errors, removal of outliers, and dataset balancing, all of which contribute to the model achieving optimal results. *Imbalanced datasets* have a suboptimal representation of one class, meaning there are more instances of one class than another [Pecorelli et al. 2020]. The most common data balancing methods are Oversampling, which artificially increases instances of the minority class, and Undersampling, which reduces instances of the majority class [Batista et al. 2004]. *Outliers* are data points that significantly deviate from the range of other values and may skew the model. To address this issue, such values can be removed or replaced by alternatives such as the mean, the median or the boundaries of the other records' range [Bonthu 2021]. There are two approaches for handling *missing data* are their removal or imputation with techniques such as replacing by the mean, linear regression and specific predictive models (ex: KNN-imputer) [de Souza et al. 2018].

2.2. Supervised Models, Evaluation Metrics, and Tuning

Classification and Regression are a supervised learning tasks in which a set of labelled examples (training data) is given to the algorithm, resulting in a model that makes predictions for unseen data points [Mohri et al. 2018]. Classification learns to map inputs x to outputs y, where $y \in 1, ..., C$, where C represents the number of classes. The resulting model takes unlabelled x inputs and attributes them to a y value. Regression is similar to classification, except that the response variable is continuous, unlike classification where the classes are discrete values. [Murphy 2012]. *AutoBioLearn* supports the methods SVM, Random Forest, XGBoost, CatBoost, and LightGBM for both classification and regression tasks. Additionally, for classification, it also includes support for Logistic Regression.

Cross-validation is a crucial technique in model evaluation, providing a robust estimation of a model's performance by partitioning the dataset into multiple subsets, or folds, and training and testing the model on each fold. This process mitigates overfitting

[1]https://github.com/AutoBioLearn

and offers a more reliable assessment of the model's ability to generalize to new data. Model evaluation metrics – such as accuracy, precision, recall, F1-score, and the area under the ROC curve for classification tasks, or R^2 and root mean squared error (RMSE) for regression tasks – provide quantitative insights into model performance. Hyperparameter tuning further enhances model accuracy by systematically adjusting parameters to identify the optimal configuration. Visualization tools, including boxplots of metric distributions, learning curves, and SHAP (Shapley Additive Explanations) values, facilitate the interpretation of model behavior and feature importance, offering a comprehensive understanding of the model's strengths and limitations.

2.3. Explainable Artificial Intelligence (XAI)

With the emergence of new ML models, their complexity has increased, some even being called "black-box". This urged the need for tools that interpret ML models. [Srinivasu et al. 2022, Antoniadi et al. 2021]. This is relevant in the medical field, since some regulations require the explainability of ML models since some studies can be biased, leading to incorrect results [Adadi and Berrada 2018]. In this context, XAI has emerged, which is a branch of ML engineering aimed at explaining results, exposing biases, and making ML models understandable to humans [Srinivasu et al. 2022, Antoniadi et al. 2021]. Even though the XAI concept dates back to the 1970s, Van Lent coined the term in 2004 to describe how his algorithm could explain the behaviour of AI-controlled game characters. Today, the widespread AI/ML use turned XAI into a widely studied area. Several XAI approaches have emerged, especially for ML [Adadi and Berrada 2018], and they can be classified according to three criteria:

- Complexity of Interpretability: degree of abstraction and simplicity of the explanation results. Methods with lower complexity use linear models or symbolic representations, while complex ones may require neural representations or advanced data analysis techniques.
- Scope of Interpretability: which aspects are explained. Local interpretability explains a specific decision and addresses individual explanations, typically to justify a particular decision for a given instance. Global interpretability explains the overall logic behind the model's results.
- Level of Dependence on the ML Model: concerns the relationship between the explanation method and the specific ML model. Methods can be either model-agnostic, i.e. applicable to various ML algorithms, or model-specific, i.e. tailored to a particular class of algorithms.

One of the most well-known methods in XAI is SHAP (SHapley Additive exPlanations). It is model-agnostic, works with black-box models, and provides both local and global explanations [Lundberg and Lee 2017, Slack et al. 2020]. SHAP is based on Shapley Values, a mathematical concept from game theory that defines how cooperative players impact the possible outcomes of a game [Shapley 1953]. In the context of XAI, the model proposed by [Lundberg and Lee 2017] assigns Shapley Value to each feature as a measure of importance, representing its contribution to the model's outcome.

3. Methodology

Here we present the *AutoBioLearn* framework, whose main advantages include an emphasis on enhancing interpretability through Explainable AI (XAI) methods, which are essential for robust biological data analysis.

Additionally, *AutoBioLearn* is designed to handle complex data structures commonly encountered in biological research, such as multi-layered headers organized into sections. Leveraging Pandas' multi-index functionality, *AutoBioLearn* efficiently imports and manipulates these hierarchical data formats. Furthermore, it enables automated analyses across different sections and combinations of data relative to the target variable. This capability allows for systematic and detailed exploration, facilitating the discovery of relevant patterns in segmented biological data.

AutoBioLearn also supports complex experimental setups, involving multiple combinations of input variable groups and target variables. Data scientists often need to perform numerous analyses across various input-output dataset combinations, which traditionally requires running the framework multiple times manually. To overcome this limitation, we introduced a functionality that allows *AutoBioLearn* to ingest a table listing all desired input-output combinations and execute all experiments automatically and systematically. This enhancement streamlines the analysis process, saving time and reducing the likelihood of human error during setup.

Finally, *AutoBioLearn* distinguishes itself through its focus on enhancing interpretability via Explainable AI (XAI) methods. When performing SHAP analyses multiple times – for instance, varying train-test partitions – the ranking of feature importance can fluctuate, potentially affecting result robustness. To mitigate this issue, we implemented a feature called **consolidated SHAP**, which aggregates the results from multiple SHAP runs and computes the average feature importance across iterations. This consolidation increases robustness, providing a more stable view of feature impacts and yielding a single, aggregated chart that offers a clearer and more reliable interpretation of variable importance. Figure 1 provides an overview of the framework, illustrating the complete workflow divided into five stages of an ML pipeline:

- Module 1 - Data preparation: describes the dataset with exploratory analysis, preprocesses the data, balances the dataset, and handles outliers.
- Module 2 - Data Splitting and Balancing: splits the dataset into training and test sets, and applies validation techniques for later model evaluation.
- Module 3 - Classification / Regression: performs training and prediction for the selected classifiers or predictors, while calculating performance metrics for model comparison.
- Module 4 - Explainability (XAI): Trains the selected models and uses SHAP to demonstrate the variables of greater influence on the model.
- Module 5 - Visualization: Displays the results of the model metrics, the XAI analysis, technique in graphical form, and provides the option to save them as image files.

We have developed a Python framework that leverages widely used libraries such as Scikit-Learn, Pandas, and NumPy. As illustrated in Figure 1, the framework can train and explain models for both binary and multiclass classification, as well as regression problems. It organizes functions to perform the complex tasks of each module, thereby simplifying the machine learning process, including class instantiation, metric analysis, and explainable artificial intelligence (XAI). To utilize the framework, it is necessary to instantiate the *AutoBioLearn* class, which is responsible for automating the pipeline and is located within the framework's namespace.

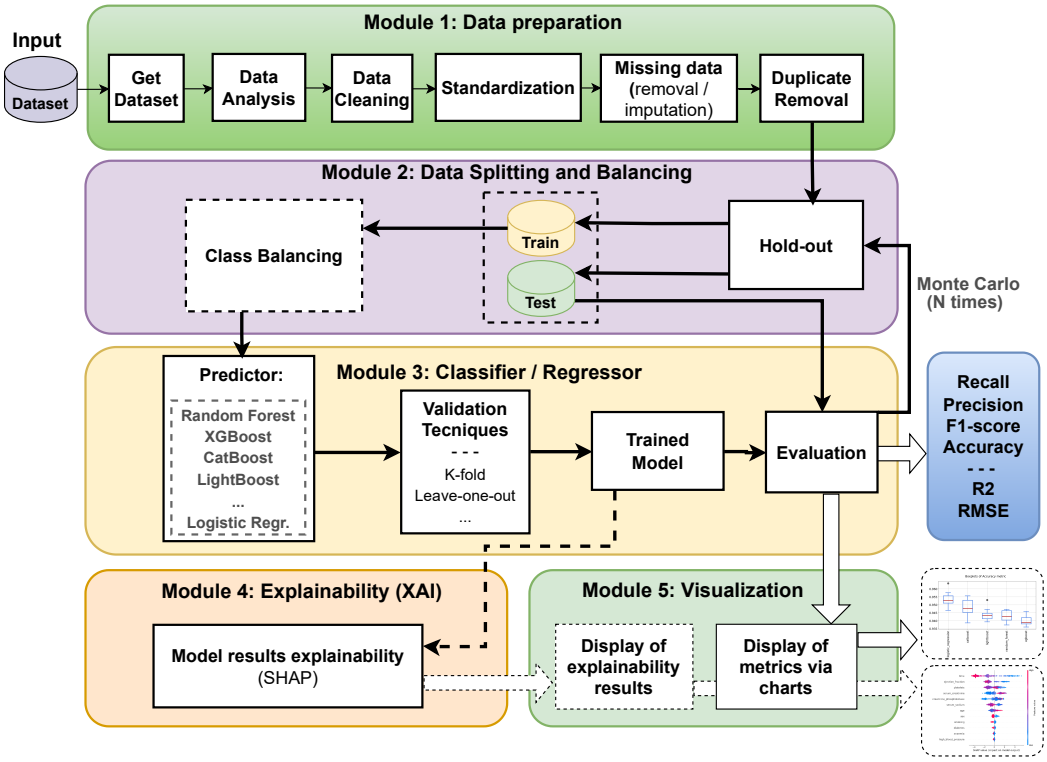

Figure 1. *AutoBioLearn* Framework Overview flowchart.

3.1. Module 1: Data Preparation

This module contains the methods that were encapsulated for data input and processing. Table 1 details the methods of this module. There are methods for encapsulating the dataset inputting, exploratory data analyses, handling missing data, type conversion, data cleaning and data removal.

3.2. Module 2: Data Splitting and Balancing

This module executes the hold-out method, which involves splitting the data into training and test sets. Optionally, it can also perform class balancing. The methods in this module work in conjunction with those in module 3. Finally, the training data (balanced or unbalanced) are passed to the subsequent module responsible for training the predictive model (either a classifier or regressor). The test data are set aside for the final evaluation of the trained model.

3.3. Module 3: Validation and Evaluation of the Predictor Models

This module provides two methods for models training, validation, testing and evaluation. The method *set_validations* configures the specifications of which validations will be employed. Optionally it can also perform the class balancing. In its turn, the method *run_models* encapsulates the holdout, training, validation and testing. Finally, the method

Table 1. Module 1: Data preparation

Data Input and Analysis Methods	
load_dataset	Encapsulates the code responsible for data input, processing based on file extension, and column name management. The user provides the file path via the `path` parameter, specifies the target variable column with `target`, and can optionally set the column delimiter through `delimiter`.
perform_eda	This method assists users in identifying the data processing methods implemented by the class. Additionally, it encapsulates the use of the *ydata-profile* library to generate an exploratory data analysis (EDA) dashboard. The dashboard can optionally be saved as a file, with the file name specified as a parameter.
Methods for handling missing data	
show_cols_na *show_rows_na*	Show the percentage of missing data in rows and columns, helping to define the parameters for *drop_rows_na* and *drop_cols_na*.
drop_rows_na	Removes rows with missing data above a user-specified percentage specified by the user through the parameter percent, or uses the default value of 10%, returning the indices of the removed rows.
drop_cols_na	Removes columns with missing data above a user-specified percentage specified by the user through the parameter percent, or uses the default value of 30%, returning the indices of the removed rows.
impute_cols_na	Performs missing data imputation using one of the following approaches: mean, mode, median, or KNN.
Data type conversion methods	
encode_categorical	converts categorical data into numerical values. Before the conversion, it excludes missing records to avoid incorrect assignments. To use it, the user specifies a list of names of the columns with categorical data, passing them as the `cols` parameter.
encode_datetime	Converts date values into numerical values by passing the columns to be converted as a list through the `cols` parameter.
Dataset cleaning methods	
remove_cols	Removes user-specified columns from the dataset based on an input list
remove_duplicates	Automatically eliminates duplicate records without the need for parameters

eval_models returns the predictor (classifier or regressor) metrics in tabular mode. The details of these methods are provided in Table 2.

Table 2. Modules 2 and 3: Methods for execution and validation of the models

set_validations	Defines which validations will be executed in the models and the parameters of each one. As default, the *hold-out*validation is configured to consider 70% for training and 30% for testing, but it also considers user-defined possibilities. For this, the method *set_validations* has three parameters: • `validations`: a list which specifies the validations to be used; • `num_folds`: number of folds which validations of type *fold* will perform in the dataset; • `train_size`: percentage of the dataset used for training in *hold-out* models.
run_models	Trains and validates the models, applying the configured validation settings for each, including Monte Carlo validation. Additionally, it evaluates each model's metrics. For validation, the user can select personalized or default values. The method allows configuration of the following parameters: • `models`: a list of predictor models (regressor or classifier) to be executed, which may include SVM, Random Forest, XGBoost, CatBoost, LightGBM, or Logistic Regression. • `times_repeats`: specifies the number of repetitions for Monte Carlo validation for each model. • `params`: a dictionary for defining the hyperparameters of each model to be executed.
evaluate_models	Returns the predictor (classifier or regressor) metrics in tabular mode. It also returns a dictionary where the metrics' names are keys and the values are the output of the Pandas *describe* function. In addition, it returns a table with the raw values of the chosen metrics, the executed models, information about Monte Carlo validation, and details about k-fold validation, if applicable. All metrics are returned by default.

3.4. Modules 4 and 5: Explainability (XAI) and Visualization

The module 4 conducts the eXplainable Artificial Intelligence (XAI) analysis of the predictor model selected in the previous module. The explainability of the chosen model is achieved using the SHAP (SHapley Additive exPlanations) technique. A method that encapsulates the use of SHAP is provided, which generates the explainability data to be utilized as input for graphical visualization in the subsequent module. The method is detailed in Table 3.

The module 5 presents the results by means of tables and charts. It includes metrics tables and boxplots for the various trained predictor models to facilitate comparison. For classifiers, the metrics shown in the boxplots are Accuracy, Recall, Precision, F1-score, and ROC-AUC. For regressors, the metrics include R2-score (coefficient of determination) and RMSE (Root Mean Square Error). Additionally, SHAP visualizations are optionally provided, which encompass global explanations, local explanations, interaction effects, and a consolidated SHAP analysis that enhances robustness by averaging feature importances across multiple runs. These methods are described in Table 3.

Table 3. Methods for Explainability Analysis and Visualization

plot_metrics	Shows results in graphical mode, generating a boxplot comparing the models for each metric (For classification: Recall, Precision, Accuracy, F1-Score, ROC-AUC. For Regression: R^2, RMSE). A chart is created for each selected metric.
perform_xai_analysis	This method performs XAI analyses, currently using SHAP. By default, it analyzes all generated models mentioned in Section 3.3, with options to filter by model, validation method, Monte Carlo execution, or fold number. The analyses are conducted for the specified models, and the results are stored in an internal class property, ensuring a seamless and transparent process for the user. Future updates will extend support to additional XAI techniques.
plot_xai_analysis	This method graphically displays the results generated by the *perform_xai_analysis* method. It provides both global and local explanations for the selected predictors (classifiers or regressors), including additional visualizations such as dependence plots, waterfall plots, and others for all analyzed models. Furthermore, it generates a consolidated chart summarizing the results across all executions of the same predictor.

4. Case Study

We compared *AutoBioLearn* a Python script which applied the same methodology to analyze medical data and the framework was able to substantially reduce the number of lines of code. We have run both *AutoBioLearn* and this previous script to the Heart Failure Prediction Dataset from Kaggle[2] [Chicco and Jurman 2020]. This dataset contains 299 patient samples (rows), 12 features (columns) and an additional column called `DEATH_EVENT`, which is a binary output representing two classes (203 not deceased and 96 deceased patients). The features include age, anaemia, creatinine_phosphokinase, diabetes, ejection_fraction, high_blood_pressure, platelets, serum_creatinine, serum_sodium, sex, smoking and time. *AutoBioLearn* performed exploratory analysis, executed classification models, and conducted XAI analysis. All classification models supported by the tool were executed and hyperparameterized. We split the data into 80% for training and 20% for testing, using Monte Carlo validation 10 times for each model.

AutoBioLearn reduced the number of lines of code by 92%, from 189 lines to 15, without considering lines related to imports, comments, blank lines, and data file loading.

[2]https://www.kaggle.com/datasets/andrewmvd/heart-failure-clinical-data/data

In addition, the number of characters was reduced from 7,785 to 820 characters, also disregarding comment characters or spaces, resulting in a reduction of approximately 89% of written characters. It maintained the same results as the non-encapsulated code. Some examples of charts that the framework's visualization module can display are illustrated in Figure 2 and Figure3.

Figure 2 provides an illustrative example of boxplots automatically generated by *AutoBioLearn* to compare classifiers based on their F1-scores (left) and ROC-AUC (right) metrics. The framework can also automatically generate similar plots for additional metrics, including Recall, Precision, and Accuracy.

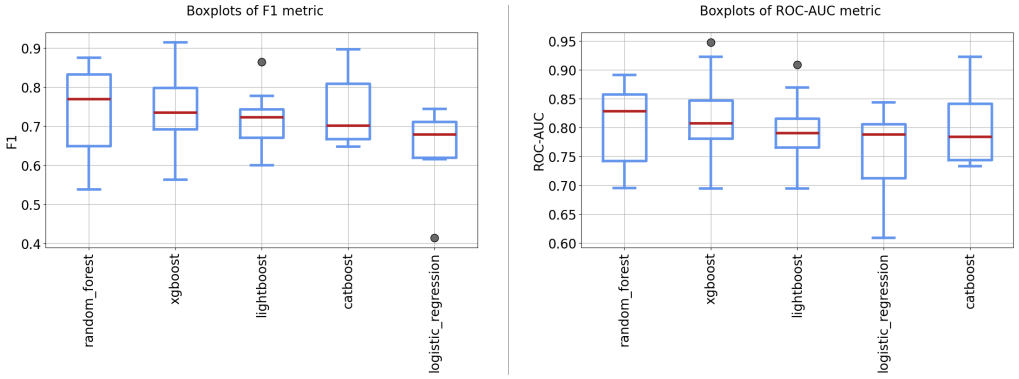

Figure 2. Illustrative example of F1-score (left) and ROC-AUC (right) boxplots comparing the implemented models, generated by *AutoBioLearn*. This figure demonstrates sample metric results produced by the framework. For classification tasks, the framework can also generate boxplots for additional metrics, including Accuracy, Recall, and Precision. Additionally, it is capable of performing similar analyses for regression models.

On the other hand, Figures 3-(A) and 3-(C) depict the results of global and local explainability modules, respectively. To make the text concise, we show only the XAI applied to the XGBoost models, but it is possible to generate these plots for each instance and classifier. As for the global XAI, Figure 3-(A) shows the order of importance of the variables for classifying the samples with the model generated at this instance. Additionally, there is an option to generate a consolidated global explainability plot for each model, which consists of all instances of a classifier displayed in a single plot. In its turn, Figure 3-(C) illustrates the local XAI for a single record. In other words, it shows how features influence the final classification outcome of a single record and instance, allowing the record-specific explanation. This local analysis can also be performed for all executions of any chosen classifier or regressor.

Figure 3-(B) illustrates a SHAP dependence plot, which displays the relationship between a feature and its SHAP values, highlighting how variations in the feature influence the model's predictions. Each point in the plot represents a data instance, with the x-axis showing the feature's value and the y-axis displaying the corresponding SHAP value, which indicates that feature's contribution to the prediction. The color of the points often corresponds to another feature, providing multi-dimensional insights. This plot helps identify trends, feature interactions, and the overall influence of each feature on the model's output.

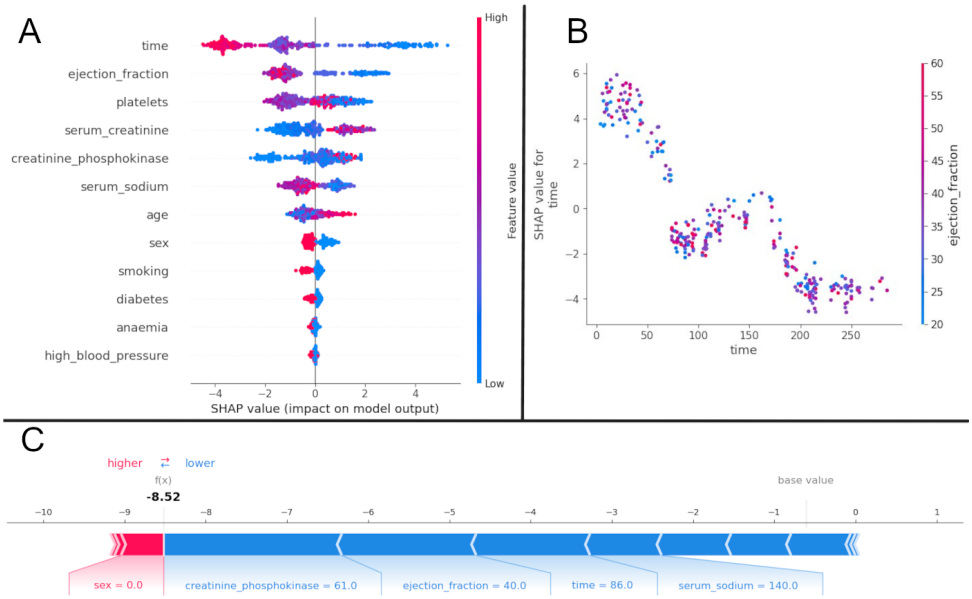

Figure 3. Illustrative example of SHAP analysis outputs automatically generated by *AutoBioLearn*. (A) Global explanation plot, where the horizontal axis represents SHAP values, indicating each feature's contribution to the model. The vertical axis lists features by importance, with blue dots for low feature values and red dots for high values. (B) Dependency plot for a specific feature pair, illustrating the relationship between SHAP values and the variable values. Each point represents an observation, allowing trend identification, such as whether increases in the variable lead to higher or lower predictions. (C) Local explanation plot showing individual feature impacts on a specific prediction. The horizontal axis represents SHAP values for a single sample, with blue segments indicating features that decrease the prediction and red segments indicating features that increase it.

5. Conclusion

This article presented a framework that assists data scientists in performing ML and XAI analyses on biological data. In our tests, the framework reduced the code size needed for analysis by approximately tenfold. Therefore, we consider that *AutoBioLearn* enables more systematized analyses in less time, thus accelerating knowledge discovery. By simplifying the programming steps, our tool frees up time for other scientific activities.

For future versions of this project, we plan to incorporate generative AI methods to enhance data augmentation and model performance, enabling the generation of synthetic data to improve model robustness. With these additions, our work supports the streamlined creation of supervised models with improved performance and integrated explainability analysis, allowing data scientists to efficiently conduct multiple sophisticated analyses.

References

Adadi, A. and Berrada, M. (2018). Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160.

Ahmed, Z., Mohamed, K., Zeeshan, S., and Dong, X. (2020). Artificial intelligence with multi-functional machine learning platform development for better healthcare and precision medicine. *Database*, 2020.

Alanazi, A. (2022). Using machine learning for healthcare challenges and opportunities. *Informatics in Medicine Unlocked*, 30:100924.

Alber, M., Buganza Tepole, A., Cannon, W. R., De, S., Dura-Bernal, S., Garikipati, K., Karniadakis, G., Lytton, W. W., Perdikaris, P., Petzold, L., et al. (2019). Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences. *NPJ digital medicine*, 2(1):1–11.

Antoniadi, A. M., Du, Y., Guendouz, Y., Wei, L., Mazo, C., Becker, B. A., and Mooney, C. (2021). Current challenges and future opportunities for xai in machine learning-based clinical decision support systems: a systematic review. *Applied Sciences*, 11(11):5088.

Batista, G. E., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29.

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., Ohl, P., Thiel, K., and Wiswedel, B. (2009). Knime - the konstanz information miner: Version 2.0 and beyond. *SIGKDD Explor. Newsl.*, 11(1):26–31.

Bonthu, H. (2021). Detecting and treating outliers — treating the odd one out!

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA. Association for Computing Machinery.

Chicco, D. and Jurman, G. (2020). Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. *BMC Medical Information Decision Making*, 20(16).

de Souza, R., Ribeiro, R., Ferlin, C., Goldschmidt, R., Carvalho, L., and Soares, J. (2018). Apoiando o processo de imputação com técnicas de aprendizado de máquina. In *Anais do XXXIII Simpósio Brasileiro de Banco de Dados*, pages 259–264, Porto Alegre, RS, Brasil. SBC.

Demšar, J., Curk, T., Erjavec, A., Črt Gorup, Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., and Zupan, B. (2013). Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353.

Dorogush, A. V., Ershov, V., and Gulin, A. (2018). Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*.

Duffy, D. J. (2016). Problems, challenges and promises: perspectives on precision medicine. *Briefings in bioinformatics*, 17(3):494–504.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Le, T. T., Fu, W., and Moore, J. H. (2020). Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics*, 36(1):250–256.

LeDell, E. and Poirier, S. (2020). H2O AutoML: Scalable automatic machine learning. *7th ICML Workshop on Automated Machine Learning (AutoML)*.

Lee, G., Singanamalli, A., Wang, H., Feldman, M. D., Master, S. R., Shih, N. N., Spangler, E., Rebbeck, T., Tomaszewski, J. E., and Madabhushi, A. (2014). Supervised multi-view canonical correlation analysis (smvcca): Integrating histologic and proteomic features for predicting recurrent prostate cancer. *IEEE transactions on medical imaging*, 34(1):284–297.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

MacEachern, S. J. and Forkert, N. D. (2021). Machine learning for precision medicine. *Genome*, 64(4):416–425.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Neto, M. G. (2021). Explainable ai: extraindo explicações e aumentando a confiança dos modelos de ml.

Pecorelli, F., Di Nucci, D., De Roover, C., and De Lucia, A. (2020). A large empirical assessment of the role of data balancing in machine-learning-based code smell detection. *Journal of Systems and Software*, 169:110693.

Petch, J., Di, S., and Nelson, W. (2022). Opening the black box: The promise and limitations of explainable machine learning in cardiology. *Canadian Journal of Cardiology*, 38(2):204–213. Focus Issue: New Digital Technologies in Cardiology.

Shailaja, K., Seetharamulu, B., and Jabbar, M. (2018). Machine learning in healthcare: A review. In *2018 Second international conference on electronics, communication and aerospace technology (ICECA)*, pages 910–914. IEEE.

Shapley, L. S. (1953). *17. A Value for n-Person Games*, pages 307–318. Princeton University Press, Princeton.

Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. (2020). Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, page 180–186, New York, NY, USA. Association for Computing Machinery.

Srinivasu, P. N., Sandhya, N., Jhaveri, R. H., and Raut, R. (2022). From blackbox to explainable ai in healthcare: existing tools and case studies. *Mobile Information Systems*, 2022:1–20.

Tsamardinos, I., Charonyktakis, P., Papoutsoglou, G., Borboudakis, G., Lakiotaki, K., Zenklusen, J. C., Juhl, H., Chatzaki, E., and Lagani, V. (2022). Just add data: automated predictive modeling for knowledge discovery and feature selection. *NPJ precision oncology*, 6(1):38.