AI

MDPI

Article

SIBILA: Automated Machine-Learning-Based Development of Interpretable Machine-Learning Models on High-Performance Computing Platforms

Antonio Jesús Banegas-Luna *,† **and Horacio Pérez-Sánchez** †

Structural Bioinformatics and High-Performance Computing (BIO-HPC), Campus de los Jerónimos, Universidad Católica de Murcia (UCAM), Guadalupe, 30107 Murcia, Spain; hperez@ucam.edu

* Correspondence: ajbanegas@ucam.edu; Tel.: +34-968278-821

† These authors contributed equally to this work.

Abstract: As machine learning (ML) transforms industries, the need for efficient model development tools using high-performance computing (HPC) and ensuring interpretability is crucial. This paper presents SIBILA, an AutoML approach designed for HPC environments, focusing on the interpretation of ML models. SIBILA simplifies model development by allowing users to set objectives and preferences before automating the search for optimal ML pipelines. Unlike traditional AutoML frameworks, SIBILA is specifically designed to exploit the computational capabilities of HPC platforms, thereby accelerating the model search and evaluation phases. The emphasis on interpretability is particularly crucial when model transparency is mandated by regulations or desired for stakeholder understanding. SIBILA has been validated in different tasks with public datasets. The results demonstrate that SIBILA consistently produces models with competitive accuracy while significantly reducing computational overhead. This makes it an ideal choice for practitioners seeking efficient and transparent ML solutions on HPC infrastructures. SIBILA is a major advancement in AutoML, addressing the rising demand for explainable ML models on HPC platforms. Its integration of interpretability constraints alongside automated model development processes marks a substantial step forward in bridging the gap between computational efficiency and model transparency in ML applications. The tool is available as a web service at no charge.

Keywords: explainable machine learning; data fusion; automated machine learning; high-performance computing; deep learning; consensus

Citation: Banegas-Luna, A.J.; Pérez-Sánchez, H. SIBILA: Automated Machine-Learning-Based Development of Interpretable Machine-Learning Models on High-Performance Computing Platforms. *AI* **2024**, *5*, 2353–2374. https://doi.org/10.3390/ai5040116

Academic Editor: Gianni D'Angelo

Received: 27 September 2024
Revised: 2 November 2024
Accepted: 11 November 2024
Published: 14 November 2024

1. Introduction

The rapid development of technologies has helped artificial intelligence (AI) become a well-known and reliable tool for researchers in academia and industry [1]. Its ability to analyze vast amounts of data has become a powerful tool in science and business [2]. Looking for repetitive patterns among such data collections is a complex but necessary task that needs to be done to extract knowledge from past events. By exploring several samples, AI models can learn the internal relationships among data and use that information to forecast future events or unexplored samples. Machine learning (ML) and its subtype, deep learning (DL), are two typical approaches to AI [3]. Both types of model are flexible enough to analyze a range of datasets, including tabular data, text, time series, and images. This adaptability to different contexts has propelled their application into traditional and fundamental areas of science, such as biology [4,5], chemistry [6,7], and medicine [8]. Classical scientific areas can profit from ML and DL, as can new and related multidisciplinary fields. This is the case with genomics [9,10], bioinformatics [11], and drug discovery [12–14], to name a few. Medicine probably has the greatest visibility in society of all the scientific areas mentioned. Advances in medicine are frequently considered highly relevant, meaning any help is always welcome. Consequently, applying ML and DL to treat patients or give

insight into a disease is a significantly relevant topic. As a result, more than a few examples of ML and DL applied to medicine can be found in the literature [15–22]. Unfortunately, ML research requires following an iterative process that can be very time-consuming and needs some expertise. First, an exploratory data analysis (EDA) is performed to clean and organize the data to allow the models to work. The researchers need some knowledge of the data to identify what changes can be made. For example, knowing the input features' meaning can help encode categorical features properly. Next, one or more models have to be developed and fed with data for training. Once the model is trained, it is evaluated, and based on the results its hyperparameters may need to be adjusted. Hyperparameter tuning is an essential but complex task that strongly determines the accuracy of the models. This workflow is repeated until the accuracy of the models reaches the desired quality level or the scientist gives up. However, when another dataset has to be processed, all this work is frequently disregarded because each dataset needs particular transformations. Many automated machine-learning (AutoML) frameworks have arisen in the last decades to automate that process and make ML and DL more accessible for researchers [23–25]. Furthermore, ML and DL models are considered unreliable in critical contexts such as medicine and still require human validation. Consequently, it is crucial to understand how models make decisions and objectively explain such understanding to the final users. Thus, a post-training process must be performed to transform the probabilities the models estimate into something easily understandable by the general public. This is known as explainable artificial intelligence (XAI). However, the explainability of the models is not easy and has two main limitations. First, there are several ways of interpreting models, ranging from inherently self-explanatory models, such as decision trees, to model agnostic algorithms like LIME or SHAP values. This diversity of explanations often leads to contradictory interpretations, making it difficult to distinguish the correct one. Secondly, the model explanation can be a time-consuming task and may require high-performance computing (HPC) platforms to carry out the calculations [26].

SIBILA is a novel software developed to automate the model development process, build several explanations, and save time for researchers. SIBILA can train, evaluate, and explain ML and DL models all at once and test various configurations in a single command line. It can be run on HPC platforms without configuration to address performance issues, resulting in competitive response times. SIBILA is containerized to cope with HPC platforms with different configurations. In consequence, it can be run on any host supporting Singularity. Moreover, SIBILA applies a diversity of XAI algorithms to provide the final user with a deep insight into the decisions that led the model to make the prediction. The users are provided with many global and sample-wise plots depicting all the metrics and explanations of the selected models. With this tool, researchers may save a great deal of time looking for models to mine their datasets and understanding the prediction process. To make it more accessible, SIBILA functionalities are also available through a freely accessible web server at https://bio-hpc.ucam.edu/sibila (accessed on 13 November 2024).

This manuscript is organized as follows. Section 2 summarizes some similar works and explains SIBILA's usefulness against competitors. Next, Section 3 introduces the main features of SIBILA, including the supported models and metrics, the way interpretability is calculated, and containerization. Section 4 presents four case studies to show how to use SIBILA effectively. In Section 5, the advantages and limitations of the tool are discussed. Finally, the main conclusions and future works are available in Section 6.

2. Related Work

Building ML algorithms can be time-consuming due to the disparity of hyperparameters that must be tuned. Moreover, reusing already tuned models is not always possible when a new dataset comes along. In order to facilitate the model-building task and alleviate the tuning efforts, many AutoML frameworks have been developed in the last few years. In Truong et al. [27], the authors extensively reviewed the available AutoML frameworks to date. The authors summarize the main tools available for researchers,

including H2O-AutoML [28] and Auto-sklearn [29]. Such tools are compared with others like AutoML-Zero [30] and FLAML [31], which have been developed recently. Similarly, Ferreira et al. [32] carried out another comparison of AutoML tools, including AutoGluon-Tabular [33]. On the contrary, Alsharef et al. [24] focused their review on ML models to forecast time-series data. In any case, the results of these, and other similar works, demonstrate the potential of the general-purpose AutoML tools to fully automate the selection and tuning of ML models. Among the most frequently used tools, it is worth mentioning the following:

- Auto-SKLearn [29] is an automated machine-learning tool built upon the scikit-learn library. It relieves users from the tasks of algorithm selection and hyperparameter tuning. The package also integrates feature engineering techniques such as one-hot encoding, numerical feature standardization, and principal component analysis (PCA). It leverages SKLearn estimators to handle both classification and regression tasks. Auto-SKLearn constructs a pipeline and employs Bayesian search to optimize it. Within this machine-learning framework, two components are introduced to refine hyperparameter tuning using Bayesian inference: meta-learning is applied to initialize the optimizers through Bayesian methods and the automatic configuration setup is evaluated throughout the optimization process.
- FLAML [31] identifies accurate models or configurations for common ML/AI tasks while minimizing computational resource use. It eliminates the need for users to manually choose models or hyperparameters for training and inference, while still allowing for easy customization. By automatically adapting large language models (LLMs) to specific applications, FLAML maximizes the advantages of these resource-intensive models while reducing associated costs. It allows users to create and deploy adaptive AI agents with minimal effort. FLAML also provides a rapid auto-tuning tool driven by a novel, cost-efficient approach, capable of managing large search spaces with varying evaluation costs, complex constraints, guidance, and early stopping mechanisms.
- H2O-AutoML [28] is an open-source, distributed in-memory, machine-learning platform. It is compatible with both R and Python and supports a wide range of commonly used statistical and machine-learning algorithms, such as gradient boosted machines, generalized linear models, and deep learning. H2O features an automated machine-learning module that utilizes its proprietary algorithms to build pipelines. It employs exhaustive search techniques for feature engineering and hyperparameter optimization to enhance pipeline performance. The platform automates various complex tasks in data science and machine learning, including feature engineering, model validation, tuning, selection, and deployment. Additionally, it offers automated visualization tools and machine-learning interpretation.
- AutoGluon [33] can generate models that predict the values in one column based on the other columns for standard tabular datasets (such as those stored in CSV files or extracted from databases). With a single call, it delivers high accuracy in typical supervised learning tasks, including both classification and regression, while automatically handling tasks like data cleaning, feature engineering, hyperparameter tuning, and model selection.

It can be observed that the presented frameworks implement several features, such as manipulating different data types (time series, images, or text), data pre-processing, and hyperparameter optimization. However, they still have some limitations. Firstly, commercial tools often require the use of the vendor platform, which is expensive. This is the case of Google AutoML and Azure ML. Although service providers offer money-saving plans, training many models with large amounts of data can be very costly in economic terms. Even more, the users are tied to the platform configuration and will have to adapt their code to make it run with the available packages. This is often a major limitation for users who have no experience with these types of tools because their field of knowledge is not related to AI or even computer science. Therefore, this problem can be a brake on the

democratization of ML. In addition, despite the fact that users do not have to worry about the configuration and maintenance of the HPC infrastructure, they will have difficulties moving their code to other platforms.

Finally, it should also be noted that, although some frameworks offer interpretability features, as a general rule they are not focused on the interpretation of ML models. Current interpretability algorithms often result in contradictory explanations of the same model, the so-called Rashomon effect, which induces the use of data fusion techniques to resolve inconsistency in explanations. This situation is rarely addressed by AutoML tools, as they usually pay more attention to the choice of models.

3. Materials and Methods

This section describes SIBILA's main features, including the list of models and interpretability algorithms available, the way tasks are parallelized, and how the data fusion works.

3.1. Architecture

SIBILA has been programmed in Python3 and relies on widely used libraries to implement both the models and interpretability algorithms. In addition, all packages are included within a Singularity container to make it portable. SIBILA has been designed to be modular and flexible, so that it can be extended in the future. Based on this idea, different modules have been implemented: models, evaluation, interpretability, and consensus (Figure 1).

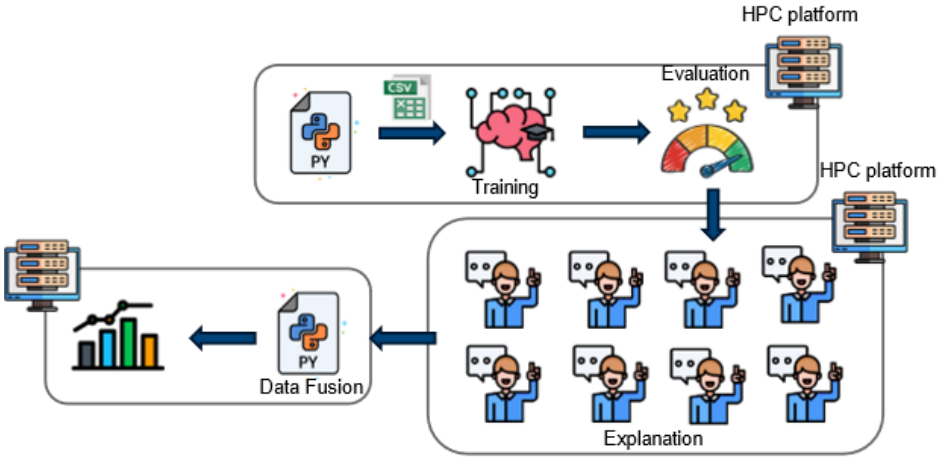

Figure 1. Architecture of SIBILA. First, an input file is received and data cleaning is performed. Next, the selected models are trained and evaluated on an HPC platform. If needed, the interpretability algorithms are run on separate jobs. Finally, the user can apply data fusion to combine the attributions calculated in the interpretation stage.

Before running a task in SIBILA, users may need to customize the hyperparameter space search to fit their needs. This process can be done by simply modifying the configuration files of each model. Next, when the experiment is launched, SIBILA processes the input parameters and loads the dataset. Once the data have been loaded, pre-processing tasks such as data analysis, balancing, and normalization are performed. Then, the Python modules that implement each of the chosen models are dynamically loaded. Note that, to add a new model, it is sufficient to implement the corresponding module to train it and perform predictions and enable the new model in the input parameter settings. In addition, any number of models can be trained in a single run, for both classification and regression tasks. Once the models are trained, they are evaluated using the appropriate metrics according to the type of problem addressed. The complete state of the SIBILA run up to this point is persisted in a file. This allows one to stop the pipeline after the evaluation

of the models and to perform the interpretation step later. The models are then interpreted with the interpretability algorithms supported by SIBILA. This process can be performed sequentially or in parallel. If SIBILA is run on an HPC cluster, the training and evaluation of the models will be executed in a single job, while the interpretability algorithms can be executed in that same job or in multiple separate jobs. Finally, SIBILA provides a command-line (CLI) script to combine the attributions computed by the interpretability algorithms into a single global attribution.

Execution through the web interface is exactly the same, the only difference being that the parameters are entered via the interface instead of by manually manipulating the model configuration files.

3.2. Machine-Learning and Deep-Learning Models

SIBILA provides a collection of ML and DL models that could easily be extended in the future. Selected models cover a wide range of approaches, including typical ML models (i.e., support vector machine, decision trees), ensemble models (bagging and random forest), DL models (neural networks), and rule-based models (RIPPERk and RuleFit). Table 1 summarizes the available models and libraries and whether or not they support regression and classification. All the implemented models support their own set of hyperparameters, which may have to be adjusted for each dataset. Hyperparameter tuning often requires code changes, which makes collaboration between researchers difficult. This problem has been addressed with a hyperparameter search system configured through external JSON (JavaScript Object Notation) files, allowing each researcher to have a custom configuration file for each model and dataset. Although JSON format is frequently used in web development, it is also helpful in other types of application due to its flexible syntax and portability. SIBILA uses JSON formatted files to simplify the configuration of the models and avoid code changes.

Table 1. List of classification and regression models implemented by SIBILA.

Model	Name	Libraries	Class./Reg.	Ref.
ANN	Artificial Neural Network	Tensorflow 2, Keras Tuner	Both	[34,35]
BAG	Bagging	scikit-learn	Both	[36]
DT	Decision Tree	scikit-learn	Both	[36]
LR	Linear/Logistic Regression	scikit-learn	Both	[36]
KNN	K-Nearest Neighbours	scikit-learn	Both	[36]
RF	Random Forest	scikit-learn	Both	[36]
RLF	RuleFit	rulefit	Classification	[37]
RP	Repated Incremental Pruning to Pruduce Error Reduction	wittgenstein	Classification	[38]
SVM	Support Vector Machine	scikit-learn	Both	[36]
XGBOOST	eXtreme Gradient Boosting Machine	xgboost	Both	[39]

3.3. Evaluation Metrics

To assess the accuracy of the models, SIBILA implements the typical metrics for the different types of problem it supports. Table 2 lists the available metrics.

Table 2. Evaluation metrics for classification and regression problems implemented in SIBILA.

Problem	Metrics
Classification	Accuracy, Area Under the Curve (AUC), Confusion Matrix, F1 Score, Matthews Correlation Coefficient (MCC), Precision, Recall, Specificity
Regression	Coefficient of Determination (R^2), Mean Absolute Error (MAE), Pearson Coefficient, Root Mean Squared Error (RMSE)

3.4. Data Cleaning

Data cleaning is a crucial step in obtaining accurate models. SIBILA provides two mechanisms to perform data pre-processing on the fly. A frequent problem encountered when analyzing datasets is unbalanced data. This problem happens when there are more samples of one class than the others, which generates a bias when training the model. Data imbalance can be tackled in SIBILA through various approaches. For example, the weighting of classes. This technique assigns a weight to each class based on the number of samples it contributes, which makes the model pay more attention to the errors of the minority class [40]. The random oversampling [41] technique is also available in SIBILA. This approach generates new random samples from perturbations in the minority class samples. Additionally, other more complex approaches, such as ADASYN (Adaptative Synthetic) [42] and SMOTE (Synthetic Minority Oversampling Technique) [43], can also be used through SIBILA. Another common problem is data denormalization. If the data for a feature are in a very sparse range, the stability of the model may be impaired [44]. This results in models that take longer to converge and whose accuracy is lower. SIBILA provides numerous algorithms for normalizing data, such as the min–max, standard scaler, or binary scaler.

3.5. Interpretability Algorithms

Machine learning and deep learning models can be very accurate in making predictions. However, there are some contexts where high accuracy is insufficient, and the models are still seen as black boxes [45]. Humans tend to rely more on the things they can easily understand, and this is a crucial factor in adopting AI in some critical contexts, such as medicine and engineering [46]. Based on that premise, XAI has emerged in recent decades. Aiming to bring light to how AI models make decisions, XAI embraces a diversity of techniques to explain to humans the decision-making process followed by AI models. Although the terms explainability and interpretability are often used interchangeably, some works make a slight difference [45,47,48]. It is not the purpose of this work to dig into the interpretability concept; thus, we will use both terms indistinctly. Aiming to contrast the different ways of explaining predictions, many interpretability algorithms are employed to detect the most relevant input features. Interpretability algorithms often assign a numerical value to each input feature to represent how much it contributed to the prediction—the more positive or negative the attribution, the more relevant the feature. SIBILA stores the attributions in text files (CSV format, comma-separated values) to perform consensus afterward. SIBILA implements a range of interpretability approaches. First, the global model-agnostic algorithms are those able to explain the entire dataset. This is the case of permutation feature importance, random forest, partial dependence plots (PDP), and accumulated local effects (ALE). These approaches take the entire dataset and assign a global attribution to each feature. Secondly, local model-agnostic methods explain every sample individually. The local model-agnostic approaches implemented in SIBILA are Shapley values, local interpretable model-agnostic explanations (LIME), integrated gradients, counterfactual explanations, and scoped rules. To better understand the local explanations, SIBILA averages the individual attributions into an overall explanation to give the user a summary of the feature's importance at a glance. Table 3 shows the supported algorithms.

Table 3. List of interpretability algorithms available in SIBILA.

Algorithm	Library	Ref.
Accumulated Local Effects (ALE)	alibi	[49,50]
Anchors (Scopes rules)	alibi	[51]
Diverse Counterfactual Explanations (DiCE)	dice-ml	[52]
Integrated Gradients	alibi	[53]
Local Interpretable Model-Agnostic Explanations (LIME)	lime	[54]
Partial Dependence Plots (PDP) + Individual Conditional Expectation (ICE)	scikit-learn	[55]
Permutation Importance	scikit-learn	[56]
Random Forest Feature Importance	scikit-learn	[57]
Shapley Values	shap	[58,59]

3.6. Consensus

Interpretability algorithms often produce contradictory explanations [60], leading to confusion and incoherence. Data fusion can be an excellent choice to reduce uncertainty in interpreting results. As SIBILA provides diverse interpretability approaches, it also provides a way of carrying out data fusion through the consensus of the attributions. SIBILA incorporates five predefined consensus methodologies: the arithmetic, geometric, and harmonic means of feature attributions; the average positional rank of a feature within a sorted list based on its attribution; and the frequency with which a feature appears among the top-10 most attributed. Through a streamlined command-line script that leverages information extracted from generated CSV files, users can effortlessly switch between these methods. Additionally, users retain the flexibility to devise and integrate their own consensus approaches. Consequently, each model's interpretability is enhanced by assigning a unified attribution score to each input feature, derived from applying the chosen consensus function to the attribution scores produced by all interpretability algorithms.

3.7. Scalability and Performance

Training and explaining models are time-consuming tasks. SIBILA can perform both tasks on various models while testing various configurations of each. To speed up the process and deliver results in a short time, SIBILA makes use of HPC. Training is accelerated by graphics processing units (GPUs) and parallel computing. The ANN model is implemented with the Tensorflow framework, which automatically detects the available GPUs on the host machine and parallelizes the training. On the other hand, some models, such as RF and XGB, can perform multi-threaded training. The number of threads is configured by the hyperparameter "n_jobs". If its value is set to -1, as many jobs as possible will be created. Model interpretability is another bottleneck in terms of speed. Many interpretability approaches perturb the trained model or the dataset to assess the impact of changes in model accuracy. This process is often computationally expensive, especially when dealing with large datasets or complex models. Since SIBILA explains models with nine different approaches, the process can take longer than training. This problem is overcome by using HPC clusters. SIBILA launches each interpretability algorithm in a separate job and waits until they are all finished before delivering the results. GPU computing is not essential for this task because the implemented algorithms do not support parallelization at the GPU level.

3.8. Containerization

SIBILA can train, evaluate, and explain several models in one go. This can become a time-consuming process with increasing dataset sizes. Hence, running SIBILA on a regular computer may not be feasible. To deal with performance issues, SIBILA has been implemented to run in a local environment or on HPC platforms. When it is run in a local environment, it is the responsibility of the user to set up the environment. To efficiently perform this task, the user is facilitated with a file with all the required dependencies.

On the contrary, users may opt to build their models on an HPC platform. Nevertheless, each platform may have a different configuration, making it challenging to run SIBILA. Aiming to avoid users struggling with the configuration of several HPC machines or clusters, a Singularity [61] container has been created. This way, the entire ecosystem of SIBILA can be run on any computer that supports Singularity and Simple Linux Utility for Resource Management (SLURM) queues. The choice of Singularity as the container instead of Docker [62] was based on two premises. Firstly, the current trend is to use Singularity instead of Docker in most HPC infrastructures [63]. This makes SIBILA a highly portable tool with zero configuration. Secondly, it is available on most of the HPC platforms we have access to. Although Docker and other containers have been extensively used, they are not always available to scientists. Additionally, Singularity is transparent to the access to GPUs, which helps to accelerate the calculations and dramatically reduces the computing times.

4. Results

This section evaluates SIBILA's ability to automatically build and explain ML and DL models. The tool has been used to build models for four different toy datasets, including binary classification, multiclass classification, and regression problems. All experiments have been run on an HPC cluster provided with Nvidia A100-SXM GPUs.

4.1. Model Search

The main purpose of SIBILA is to search for ML and DL models. In this experiment, its ability to find suitable models for four different tasks is tested, including two binary classification problems and multiclass and regression tasks. All the datasets are publicly available at the UCI Machine Learning Repository [64]. Table 4 summarizes the most relevant information about the datasets. A hyperparameter random search was carried out with all the models. This approach speeds up the search of a base model while leaving the exhaustive search for later with a reduced set of hyperparameter values. SIBILA supports extensive hyperparametrization of all models. Tables A1 and A2 detail the hyperparametrization space configured for classification and regression experiments. In either case, the user can easily customize the search values by manually editing the parameter files. No cross-validation or class weighting was applied to preserve the default values of the search. Tables 5 and 6 show each task's best model and main metrics. In classification problems, the best model was chosen based on the area under the curve (AUC) metric, while in the regression task the coefficient of determination was the reference metric.

Table 4. Description of the four toy datasets employed in the experiments.

Dataset	Description	Task	Samples	Features	Ref.
Cancer	Prediction of indicators of cervical cancer	Binary classification	858	40	[65]
Spam	Identification of spam emails	Binary classification	4601	57	[66]
Wine	Classification of three types of Italian wine	Multiclass classification	178	13	[67]
Crime	Prediction of the number of crimes in the USA	Regression	1994	4091	[68]

Table 5. Best models built for classification tasks.

Dataset	Task	Model	Specificity	Precision	Recall	AUC
Cancer	Binary classification	XGB	100.000	100.000	100.000	1.000
Spam	Binary classification	ANN	95.390	92.818	94.118	0.948
Wine	Multiclass classification	XGB	33.095	97.619	97.619	0.536

Table 6. Best models built for regression task.

Dataset	Task	Model	MAE	MSE	RMSE	R^2
Crime	Regression	RF	0.087	0.018	0.133	0.628

The results demonstrate that SIBILA can find suitable base models for different tasks and datasets. It must be pointed out that the displayed results were obtained with the default search space used by SIBILA. However, such values can be easily customized by modifying the JSON configuration files. The final hyperparameters of the best models (Tables A3–A6) and the complete list of metrics (Tables A7–A10) are provided in Appendix A.

4.2. Interpretability and Data Fusion

SIBILA interprets each model with a collection of interpretability algorithms (Figures 2–6). All interpretability algorithms supported by SIBILA calculate the contribution of each input feature to the model prediction. This contribution, known as attribution, is a numerical value whose range of values ranges from one algorithm to another. Local methods assign an attribution to each feature in the prediction of each individual sample, while global methods assign a single overall feature attribution for the entire dataset. Interpreting the graphs created by SIBILA is very simple since all algorithms assign higher attribution values to the most important features. In addition, the standard deviation of the individual attributions has been included in the graphs created to summarize the local methods in order to quickly understand the stability of the algorithm and the dispersion of the data. The attributions of the local methods are averaged into a global attribution that summarizes the interpretations of all samples. This facilitates the explanation of models with local methods by users.

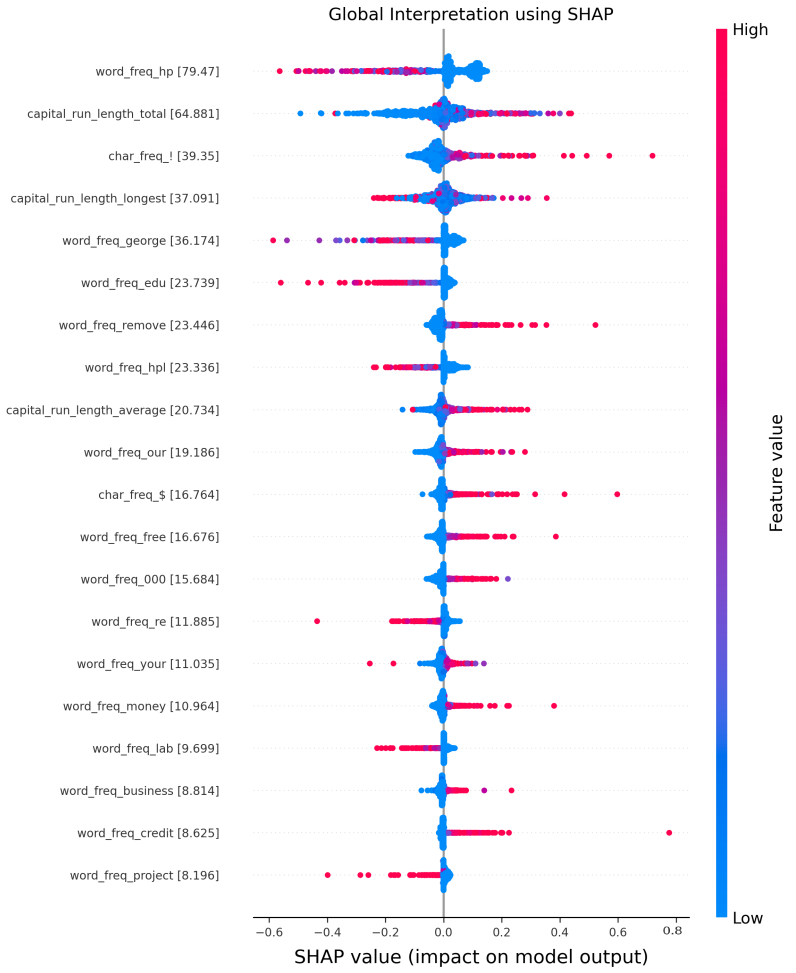

Figure 2. Average attribution plots created by SIBILA. Shapley values obtained for the ANN model after training the spam dataset.

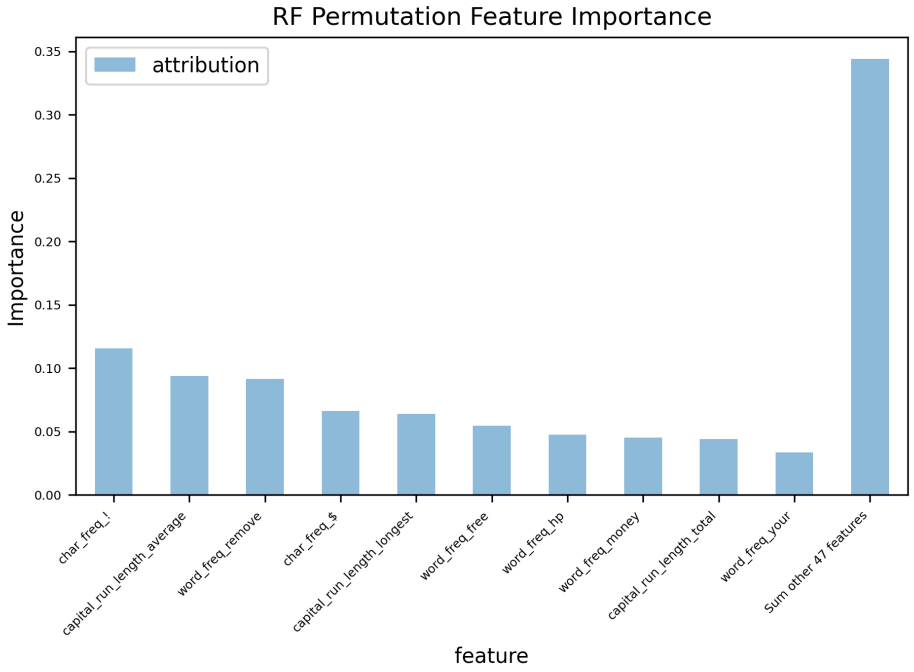

Figure 3. Average attribution plots created by SIBILA. Random-forest-based permutation importance obtained for the ANN model after training the spam dataset.

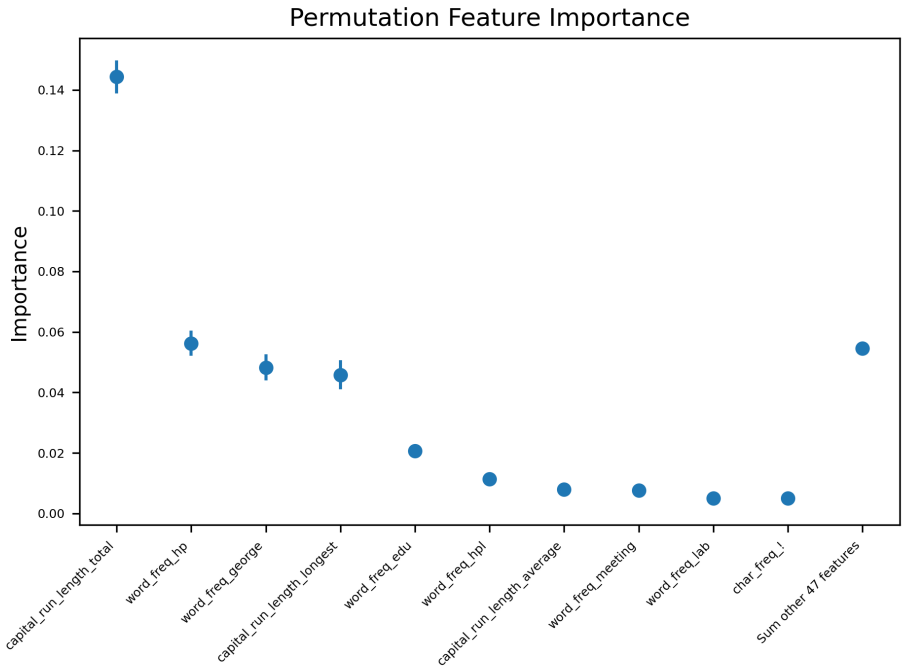

Figure 4. Average attribution plots created by SIBILA. Permutation importance obtained for the ANN model after training the spam dataset.

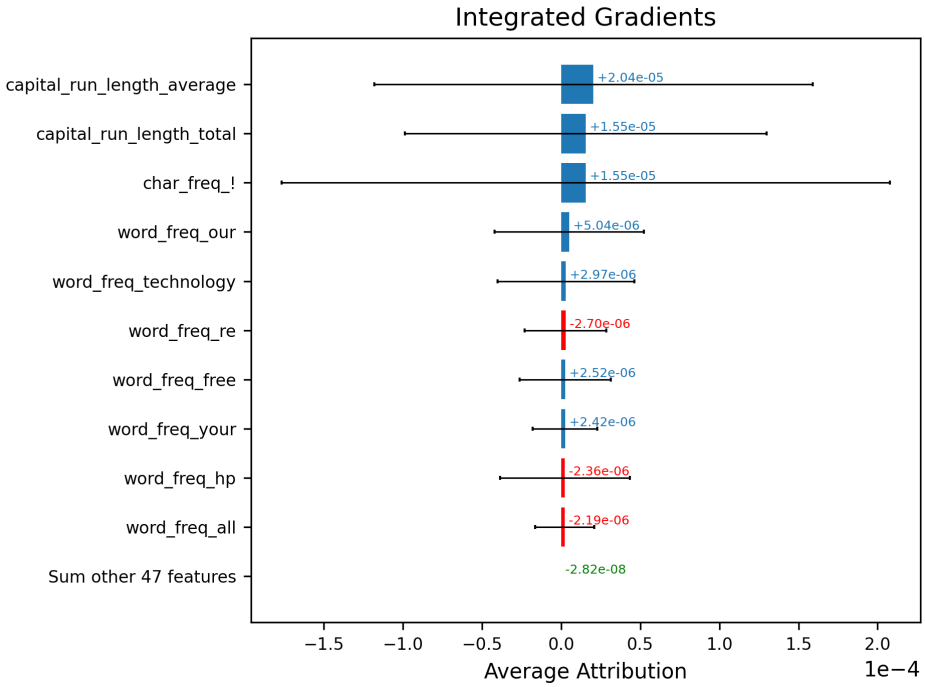

Figure 5. Average attribution plots created by SIBILA. Integrated gradients attributions obtained for the ANN model after training the spam dataset.

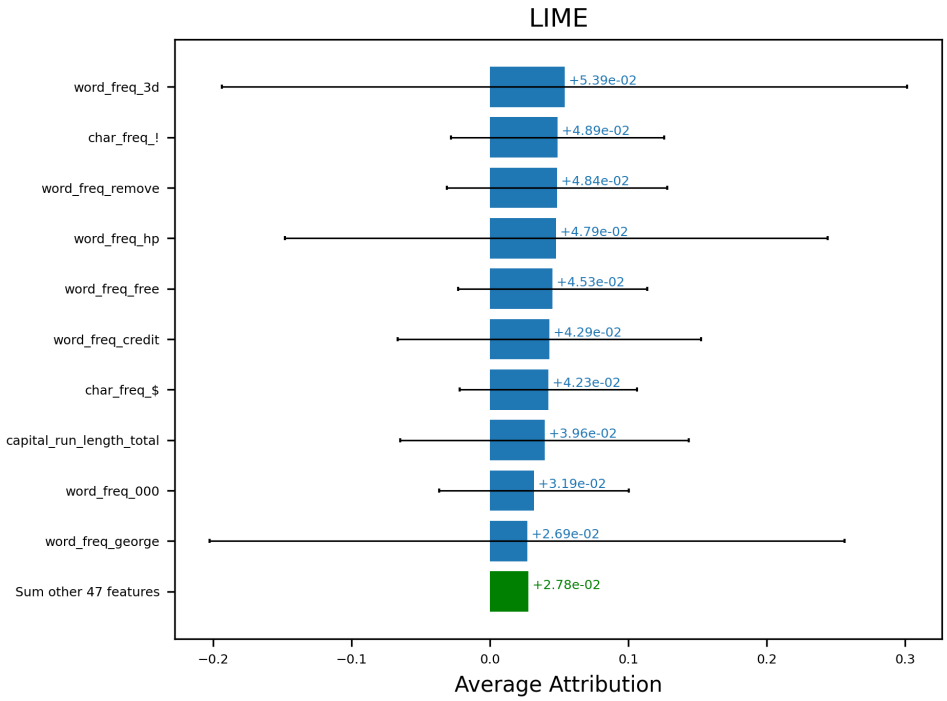

Figure 6. Average attribution plots created by SIBILA. LIME attributions obtained for the ANN model after training the spam dataset.

The interpretations obtained for the same model can vary substantially depending on the algorithm used to generate them. SIBILA addresses this problem by employing data fusion, the so-called consensus. Consensus consists of combining the attributions assigned by each algorithm to each of the input features. Thus, a combined attribution value per feature is obtained for the same model that has been interpreted with different algorithms. This mechanism leads to more robust global interpretations in which possible deviations are minimized. SIBILA implements a collection of consensus functions. Table 7 shows each dataset's five most relevant features after applying data fusion with the arithmetic mean. Note that the features are sorted by the absolute value of their attribution, as this represents the total importance of that feature in explaining the model. Figure 7 shows the consensus plots created by SIBILA for the best model found for each dataset. The length of the bars represents the attribution score; thus, the longer the bar, the more important the feature.

Table 7. Top-5 ranked features per dataset after data fusion.

Cancer		Spam		Wine		Crime	
Feature	**Attribution**	**Feature**	**Attribution**	**Feature**	**Attribution**	**Feature**	**Attribution**
Dx:HPV	0.129	Word_freq_remove	0.067	Proline	0.130	NumStreet	−0.004
Dx	0.043	Word_freq_hp	0.021	Color intensity	0.045	PctKids2Par	−0.003
Smokes (years)	0.004	Word_freq_free	0.019	Flavanoids	0.035	racePctWhite	−0.002
Hinselmann	0.001	Char_freq_$	0.018	Hue	0.025	pctWInvInc	−0.001
Biopsy	0.001	Char_freq_!	0.015	Alcohol	0.017	PctFam2Par	−0.001

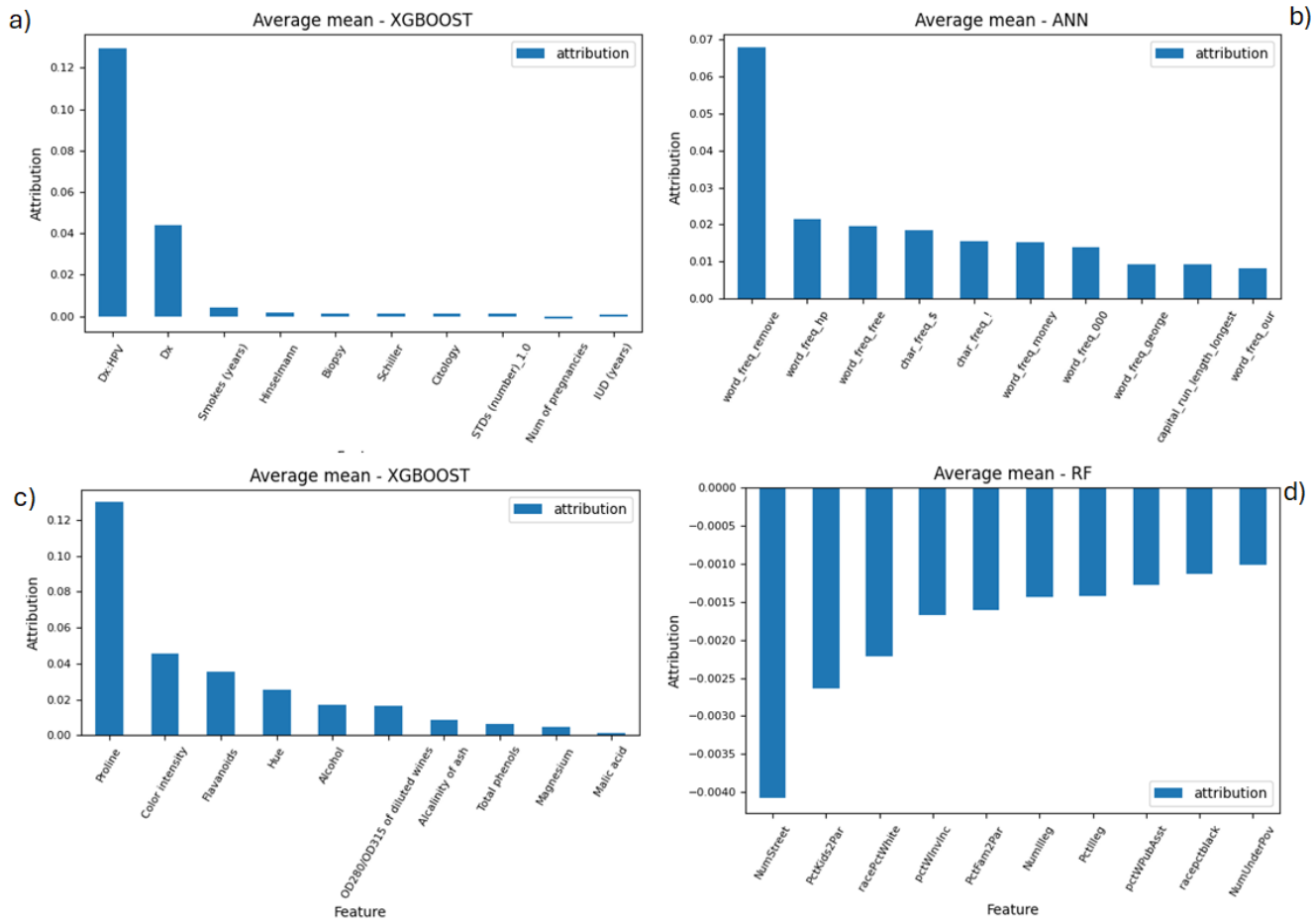

Figure 7. Top 10 most important features of each dataset according to SIBILA consensus: (**a**) Cancer dataset; (**b**) Spam dataset; (**c**) Wine dataset; (**d**) Crime dataset.

4.3. Parallelization and GPU Usage

Using cluster and GPU parallelization is crucial when training and explaining ML/DL models. SIBILA uses GPU processing for model training and HPC clusters to explain the models. This test aims to compare SIBILA's performance when GPU and HPC clusters are used. Table 8 shows the training and interpretation times, in seconds, with and without GPU and HPC clusters of the most accurate models. Note that the interpretation of the models does not use GPU because the implemented algorithms do not support that sort of parallelization.

Table 8. SIBILA performance in training and interpretation tasks.

		Training			Interpretation		
Dataset	**Task**	**CPU**	**GPU**	**Improve**	**Sequential**	**HPC Cluster**	**Improve**
Cancer	Binary clf.	17.35	8.64	50.20%	2267.10	1245.87	45.05%
Spam	Binary clf.	27.53	986.87	−3484.71%	12,841.32	6257.19	51.27%
Wine	Multiclass clf.	19.28	19.25	0.16%	167.60	49.86	70.25%
Crime	Regression	29.12	20.20	30.63%	116,986.10	103,696.92	11.36%

Regarding interpretability, the time with cluster parallelization is the time taken by the slowest algorithm to interpret the model. In contrast, the sequential time is the addition of the time taken to interpret a model with all the individual algorithms. It can be observed that the reduction ranged from 11.36% in the crime dataset to 70.25% in the wine dataset. On the other hand, the use of GPU to train the models accelerates the process by up to 50.20% with the cancer dataset. SIBILA depicts this information for every individual model, as shown in Figure 8.

Figure 8. Execution times of the XGB model after training and interpreting the wine dataset.

5. Discussion

SIBILA has been benchmarked in three different tasks with four toy datasets. The primary purpose of SIBILA is to search for ML and DL models for a given dataset. In this regard, a complete search was performed across all the available models and three types of problem. The first search was carried out with the cervical cancer dataset. Some models, such as XGB, reached the maximum accuracy (AUC = 1). This test proved SIBILA to be helpful in binary classification tasks. However, the unusually high scores obtained for many of the models may indicate that cervical cancer is a too-simple dataset. Hence, another binary classification task was carried out with the spam classifier dataset. This dataset is bigger than the previous one (4601 samples and 57 features), but the accuracy of the models is also very high. The chosen model was the deep-learning approach (ANN) in this case.

Although the ML models also reached high accuracy, the ANN model outperformed them. This may be due to the complexity of the data that makes the DL approach better than other simpler models. Regarding the multiclass classification task, the preferred model was again XGB. However, the accuracy was much lower than in binary classification (AUC = 0.536). The poor accuracy of the model suggests two improvements: (i) Pre-processing the data with more techniques (handle outliers, standardization...), (ii) training the models with advanced options like cross-validation and class weighting. In any case, XGB seems to be a good choice as a baseline. The last task was to predict the number of crimes in different cities in the USA. The suggested model was RF, which is an ensemble model. The metrics obtained for this model were acceptable (MAE = 0.087, MSE = 0.018), but the coefficient of determination was not so high (R^2 = 0.628). R^2 is a correlation metric that indicates if there is a linear relationship between the inputs and the target feature. The obtained value suggests a slight correlation, which is confirmed by the plot created by SIBILA (Figure 9).

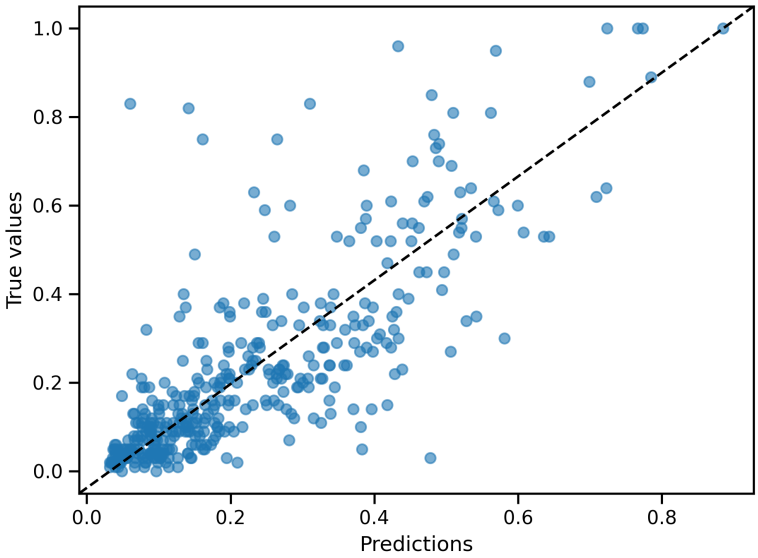

Figure 9. Correlation between the crime dataset's inputs and output with the random-forest model.

The second experiment involves identifying the most relevant features for the models to make decisions. Figure 7 shows the top-10 attributed features after the data fusion. Our consensus function, the average mean, detects an evident influence of the "Dx:HPV" feature when explaining the XGB model for the cancer dataset. The HPV test examines cells to find infections with the high-risk types of HPV that sometimes cause cervical cancer. It is recommended every five years starting at age 30. Hence, it seems logical to consider it a relevant biomarker for our model. Concerning the spam dataset, the ANN model focuses on the appearance of the word "remove" so that an email containing this word frequently should be tagged as spam. Although this is the best discriminant, other words like "hp" and "free" or the character "$" could also serve to classify an email as spam. Again, the explanation of this model makes sense because the word "remove" could appear once or twice but not several times in an email. The explanation of the wine dataset targets "proline" as the most relevant feature. It has been proven that proline concentration in grapes is related to maturity and cannot be exploited by yeasts; as such, its concentration can be managed through specific viticultural practices. On the other hand, proline is easy to determine in wine and could be used to identify desirable sensory profiles [69]. Finally, the regression with the crime dataset was the hardest to explain because all the features received similar attention, and almost all were negatively attributed. The negative attribution values mean that the higher the feature values, the lower the probability of classifying the sample as the predicted class. SIBILA sorts the features by the absolute value of their attributions; thus, the most important ones may not be aligned with the

highest attributions. Although the attributions were very similar, "NumStreet" received more attention than others. The negative importance of NumStreet indicates that this feature is associated with a reduction in the crime rate, i.e., the higher the number, the lower the crime rate. If a community has more streets, it could be interpreted as having better infrastructure, greater connectivity, or being better developed. This could correlate with lower crime rates, as indicated by the model. On the contrary, the features with the highest positive attribution were "community" and "county", which indicates that specific communities and certain counties are more prone to high crime rates.

The last experiment aimed to demonstrate the ability of SIBILA to accelerate the training and interpretation of the models by using HPC platforms and parallelization. The training of the models is accelerated by GPU computing. The results show that GPU-intensive training outperforms the CPU in most cases. Training times can be reduced by up to 50.20% in the cervical cancer dataset. The wine dataset suffered a slight time reduction (0.16%) because it is a small dataset with insufficient data to take advantage of GPU. It is worth pointing out that the spam dataset took longer on GPU than on CPU. Different reasons might be behind this result, but the exclusiveness of cluster nodes should be considered first. Our configuration does not ensure that an entire cluster node is reserved for our jobs; consequently, if the GPU nodes were overloaded with other tasks but the CPU nodes were underutilized, the computation time on the GPU nodes may be higher than on the CPU. In any case, a deeper analysis would be required to explain such behavior. On the other hand, the interpretability algorithms do not profit from GPU parallelization; thus, HPC clustering is used instead. In HPC clustering, each interpretability algorithm was launched in a separate job. Thus, the time taken by the longest job was used as a reference because all the other jobs were finished before. Thanks to this approach, the computing time for interpretability tasks was reduced by between 11.36% and 70.25%.

6. Conclusions

Machine learning offers robust capabilities for decision-support systems, though its adoption is frequently hindered by time-consuming model development and interpretability issues. This becomes particularly critical in fields such as healthcare, where decision transparency is paramount. In the absence of interpretability, professionals in medicine, law, finance, and politics may hesitate to rely on these predictions, thereby forfeiting the potential benefits of these advanced techniques. To address these challenges, we have developed SIBILA, an AutoML tool that trains and explains ML/DL models by leveraging HPC infrastructures. SIBILA's automated computation of interpretability significantly reduces the time scientists would traditionally spend on building bespoke models for individual datasets [70]. Additionally, it presents results in a user-friendly format accessible to professionals outside the IT sphere. Acknowledging the wide range of potential users, SIBILA is freely available at https://bio-hpc.ucam.edu/sibila (accessed on 10 November 2024). The results confirm that SIBILA can efficiently train, evaluate, and elucidate multiple models for regression, binary, and multiclass classification problems. It has found accurate base models for four datasets containing data of very dissimilar contexts. In addition, SIBILA could provide users with coherent explanations of the selected models. For each dataset, a few input features were clearly identified as the most influential in the model's decision-making. The final subset was the result of interpreting the models with many explainability approaches and then applying a data-fusion process yielding a unique global explanation. Using GPU parallelization and HPC clustering, SIBILA drastically reduced the computation time required for the training and interpretability steps. The time reduction reached 70% when interpreting the wine dataset. Notwithstanding its many functionalities, SIBILA does have areas for future development. Integrating consensus calculation into the pipeline rather than conducting it through separate scripts could ease its usage. The set of models could be extended to include stacking or even unsupervised-learning models (i.e., clustering). Not only the model set but also the input data is susceptible to being extended. Image processing, natural language processing (NLP), and time series could

benefit from SIBILA when looking for baseline models. Despite these limitations, we firmly believe that SIBILA holds promise in advancing the accessibility and applicability of interpretable machine-learning models, thus contributing to more transparent and trustworthy artificial-intelligence systems across various domains.

Author Contributions: A.J.B.-L., conceptualization, investigation, data curation, software, writing; H.P.-S., funding, conceptualization, investigation, validation, supervision, writing, funding. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the European Project Horizon 2020 SC1-BHC-02-2019 [REVERT, ID:848098].

Data Availability Statement: The data used in this manuscript are freely accessible. The authors indicate from where the data can be obtained in the document. The source code is available at https://github.com/bio-hpc/sibila.git (accessed on 10 November 2024).

Acknowledgments: Supercomputing resources in this work were supported by the Poznan Supercomputing Center's infrastructures, the e-infrastructure program of the Research Council of Norway, and the supercomputing center of UiT—the Arctic University of Norway, the Plataforma Andaluza de Bioinformática at the University of Málaga, the supercomputing infrastructure of the NLHPC (ECM-02, Powered@NLHPC), and the Extremadura Research Centre for Advanced Technologies (CETA-CIEMAT), funded by the European Regional Development Fund (ERDF). CETA-CIEMAT is part of CIEMAT and the Government of Spain.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADASYN	Adaptative Synthetic
AI	Artificial Intelligence
ALE	Accumulated Local Effects
AUC	Area Under the Curve
AutoML	Automated Machine Learning
CSV	Comma-Separated Values
DL	Deep Learning
EDA	Exploratory Data Analysis
GPU	Graphics Processing Unit
HPC	High-Performance Computing
ICE	Individual Conditional Expectation
JSON	JavaScript Object Notation
LIME	Local Interpretable Model-agnostic Explanations
LLM	Large Language Models
MAE	Mean Absolute Error
MCC	Matthews Correlation Coefficient
ML	Machine Learning
MSE	Mean Squared Error
PCA	Principal Component Analysis
PDP	Partial Dependence Plot
RMSE	Root Mean Squared Error
SLURM	Simple Linux Utility for Resource Management
SMOTE	Synthetic Minority Oversampling Technique
XAI	eXplainable Artificial Intelligence

Appendix A

Appendix A.1. Hyperparameter Values Used in the Random Search Process

Table A1. Tested hyperparameters for the classification problems.

Model	Hyperparameter Values
ANN	batch_size: 128, objective: accuracy, activate: [relu, elu, tanh, sigmoid, softmax, linear, exponential], dropout_rate: 0.15, optimizer: [Adam, RMSprop, SGD, Adagrad], loss: sparse_categorical_crossentropy, epochs: 100
BAG	n_estimators: [10, 20, 50], max_features: [0.25, 0.5, 1.0], oob_score: [true, false], bootstrap: [true, false], max_samples: [0.25, 0.5, 1.0]
DT	criterion: [gini, entropy], splitter: [best, random], max_depth: [2, 4, 6, 8, 10, 12], min_samples_split: [0.1, 0.2, 0.4, 0.8, 0.9], min_samples_leaf: [1, 2, 3, 4], max_features: [auto, sqrt, log2], max_leaf_nodes: [50, 100, 200, 300], min_impurity_decrease: [0, 0.1, 0.2, 0.3, 0.4, 0.5], ccp_alpha: [0, 0.1, 0.2, 0.3, 0.4, 0.5]
KNN	n_neighbors: [3, 4, 5, 6, 7], algorithm: [auto, ball_tree, kd_tree, brute], leaf_size: [10, 20, 30, 50], metric: [minkowski, euclidean, manhattan, chebyshev], p: [1, 2, 3]
LR	penalty: [l1, l2, elasticnet], tol: [0.001, 0.0001, 1×10^{-5}], C: [0.25, 0.3, 0.5, 0.6, 0.75, 0.9, 1], fit_intercept: [true, false], solver: [liblinear, newton-cg, sag, saga], max_iter: [50, 100, 500, 1000], l1_ratio: [0.1, 0.25, 0.5, 0.75, 1]
RF	n_estimators: [50, 100, 400, 800], criterion: [gini, entropy], max_depth: [25, 50, 250], min_samples_split: [2, 5, 10], min_samples_leaf: [2, 5, 10], max_features: [auto, sqrt, log2], oob_score: [true, false], bootstrap: [true, false]
RLF	tree_size: [4, 16, 32], max_rules: [50, 100, 500, 1000], memory_par: [0.01, 0.05, 0.1], lin_trim_quantile: [0.025, 0.05, 0.1], lin_standardise: [true, false], exp_rand_tree_size: [true, false], cv: [3, 5]
RP	n_discretize_bins: [10, 20, 50], k: [1, 2], prune_size: [0.25, 0.33, 0.5]
SVM	C: [0.5, 1, 1.5], kernel: [linear, poly, rbf, sigmoid], degree: [1, 2, 3, 4, 5], gamma: [scale, auto], coef0: [0, 0.5, 1], shrinking: [true, false], tol: [1×10^{-4}, 5×10^{-4}, 1×10^{-3}, 2×10^{-3}], cache_size: [100, 200, 300], max_iter: [−1, 100, 150, 200, 500], decision_function_shape: [ovo, ovr]
XGBOOST	n_estimators: [50, 100, 300, 600], booster: [gbtree, gblinear, dart], eta: [0.1, 0.3, 0.5], gamma: [0, 0.5], max_depth: [4, 6, 8], min_child_weight: [1, 2], max_delta_step: [0, 5, 10], subsample: [0.1, 0.5, 1], lambda: [0.5, 1, 1.5], alpha: [0, 0.5, 1], tree_method: [auto, exact, approx,hist], grow_policy: [depthwise, lossguide], max_leaves: [0, 5, 15, 25], max_bin: [128, 256], sketch_eps: [0.01, 0.03, 0.05], refresh_leaf: [0, 1], scale_pos_weight: [1, 10, 25, 50, 75, 99, 100, 1000]

Table A2. Tested hyperparameters for the regression problem.

Model	Hyperparameter Values
ANN	batch_size: 128, objective: loss, activate: [relu, elu, tanh, sigmoid, softmax, linear,exponential], optimizer: [Adam, RMSprop, SGD, Adagrad], loss: mean_absolute_error, epochs: 120
BAG	n_estimators: [10, 20, 50], max_features: [0.25, 0.5, 1.0], oob_score: [true, false], bootstrap: [true, false], max_samples: [0.25, 0.5, 1.0]
DT	splitter: [best, random], max_depth: [2, 4, 6, 8, 10, 12], min_samples_split: [0.1, 0.2, 0.4, 0.8, 0.9], min_samples_leaf: [1, 2, 3, 4], max_features: [auto, sqrt, log2], max_leaf_nodes: [50, 100, 200, 300], min_impurity_decrease: [0, 0.1, 0.2, 0.3, 0.4, 0.5], ccp_alpha: [0, 0.1, 0.2, 0.3, 0.4, 0.5]
KNN	n_neighbors: [3, 4, 5, 6, 7], algorithm: [auto, ball_tree, kd_tree, brute], leaf_size: [10, 20, 30, 50], metric: [minkowski, euclidean, manhattan, chebyshev], p: [1, 2, 3]
LR	penalty: [l1, l2, elasticnet], tol: [1×10^{-3}, 1×10^{-4}, 1×10^{-5}], C: [0.25, 0.3, 0.5, 0.6, 0.75, 0.9, 1], fit_intercept: [true, false], solver: [liblinear, newton-cg, sag, saga], max_iter: [50, 100, 500, 1000], l1_ratio: [0.1, 0.25, 0.5, 0.75, 1]
RF	n_estimators: [50, 100, 400, 800, 2000], max_depth: [25, 50, 250, 500], min_samples_split: [2, 5, 10, 20], min_samples_leaf: [2, 5, 10], max_features: [auto, sqrt, log2], oob_score: [true, false], bootstrap: [true, false], min_weight_fraction_leaf: [0, 0.5]
SVM	C: [0.5, 1, 1.5], kernel: [linear, poly, rbf, sigmoid], degree: [1, 2, 3, 4, 5], gamma: [scale, auto], coef0: [0, 0.5, 1], shrinking: [true, false], tol: [1×10^{-4}, 5×10^{-4}, 1×10^{-3}, 2×10^{-3}], cache_size: [100, 200, 300], max_iter: [−1, 100, 150, 200, 500]
XGBOOST	n_estimators: [50, 100, 300, 600, 1000], booster: [gbtree, gblinear, dart], eta: [0.1, 0.3, 0.5], gamma: [0, 0.25, 0.5], max_depth: [4, 6, 8, 12, 20], min_child_weight: [1, 2], max_delta_step: [0, 5, 10], subsample: [0.1, 0.5, 1], lambda: [0.5, 1, 1.5], alpha: [0, 0.5, 1], tree_method: [auto, exact, approx, hist], grow_policy: [depthwise, lossguide], max_leaves: [0, 5, 15, 25, 35, 50], max_bin: [128, 256, 512], sketch_eps: [0.01, 0.03, 0.05], refresh_leaf: [0, 1], scale_pos_weight: [1, 10, 25, 50, 75, 99, 100, 1000]

Appendix A.2. Best Hyperparameters of Each Model Selected for Each Dataset

Table A3. Hyperparameters of the XGB model for the cancer dataset.

Hyperparameter	Value
N_estimators	100
Booster	Dart
Eta	0.1
Gamma	0.5
Max_depth	8
Min_child_weight	2
Max_delta_step	10
Subsample	1
Lambda	1
Alpha	0
Tree_method	Approx
Grow_policy	Depthwise
Max_leaves	25
Max_bin	256
Sketch_eps	0.05
Refresh_leaf	1
Scale_pos_weight	100

Table A4. Hyperparameters of the ANN model for the spam dataset.

Hyperparameter	Value
Num_layers	4
Units	[248, 12, 12, 184]
Output_units	2
Activation	Softmax
Dropout	False
Optimizer	Adam
Learning_rate	0.001131
Loss_function	Sparse_categorical_crosstentropy
Epochs	100

Table A5. Hyperparameters of the XGB model for the wine dataset.

Hyperparameter	Value
N_estimators	600
Booster	Gbtree
Eta	0.5
Gamma	0.25
Max_depth	12
Min_child_weight	1
Max_delta_step	0
Subsample	0.1
Lambda	1.5
Alpha	0.5
Tree_method	Hist
Grow_policy	Depthwise
Max_leaves	25
Max_bin	256
Sketch_eps	0.03
Refresh_leaf	1
Scale_pos_weight	25

Table A6. Hyperparameters of the RF model for the crime dataset.

Hyperparameter	Value
N_estimators	400
Criterion	mse
Max_depth	50
Min_samples_split	20
Min_samples_leaf	2
Min_weight_fraction_leaf	0
Max_features	sqrt
Oob_score	False
Bootstrap	False

Appendix A.3. Metrics Obtained for Every Model with All the Datasets

Table A7. Evaluation metrics obtained for the cancer dataset.

Model	Accuracy	Precision	F1 Score	Recall	Specificity	AUC	Matthews
ANN	97.674	100.0	50.0	33.333	100.0	0.667	0.571
BAG	98.837	100.0	80.0	66.667	100.0	0.833	0.812
DT	96.512	0.0	0.0	0.0	100.0	0.500	0.0
KNN	96.512	0.0	0.0	0.0	100.0	0.500	0.0
LR	97.093	100.0	28.571	16.667	100.0	0.583	0.402
RF	98.256	100.0	66.667	50.000	100.0	0.750	0.701
RLF	99.419	100.0	90.909	83.333	100.0	0.917	0.910
RP	100.0	100.0	100.0	100.0	100.0	1.0	1.0
SVM	96.512	0.0	0.0	0.0	100.0	0.500	0.0
XGB	100.0	100.0	100.0	100.0	100.0	1.0	1.0

Table A8. Evaluation metrics obtained for the spam dataset.

Model	Accuracy	Precision	F1 Score	Recall	Specificity	AUC	Matthews
ANN	94.897	92.818	93.463	94.118	95.390	0.948	0.893
BAG	94.571	94.236	92.898	91.597	96.454	0.940	0.885
DT	77.199	94.012	59.924	43.978	98.227	0.711	0.534
KNN	87.079	84.0	83.168	82.353	90.071	0.862	0.727
LR	92.725	92.151	90.442	88.796	95.213	0.920	0.846
RF	95.223	95.101	93.750	92.437	96.986	0.947	0.899
RLF	94.137	94.169	92.286	90.476	96.454	0.935	0.876
RP	88.708	93.471	83.951	76.190	96.631	0.864	0.763
SVM	78.284	67.253	75.369	85.714	73.582	0.796	0.578
XGB	89.794	80.510	88.071	97.199	85.106	0.912	0.804

Table A9. Evaluation metrics obtained for the wine dataset.

Model	Accuracy	Precision	F1 Score	Recall	Specificity	AUC	Matthews
ANN	69.444	51.389	54.193	61.905	29.098	0.518	0.600
BAG	94.444	94.103	94.447	95.055	32.717	0.535	0.917
DT	52.778	43.452	46.963	56.695	28.123	0.525	0.402
KNN	80.556	78.974	78.792	78.816	30.389	0.526	0.706
LR	94.444	94.103	94.447	95.055	32.717	0.535	0.917
RF	97.222	96.667	96.912	97.436	32.963	0.535	0.959
RLF	97.222	96.667	96.912	97.436	32.963	0.535	0.959
SVM	61.111	53.968	54.325	63.146	26.167	0.529	0.454
XGB	97.222	97.619	97.531	97.619	33.095	0.536	0.959

Table A10. Evaluation metrics obtained for the crime dataset.

Model	Pearson	R^2	MAE	MSE	RMSE
ANN	40.340	0.114	0.139	0.042	0.205
BAG	77.408	0.599	0.091	0.019	0.138
DT	nan	−0.004	0.170	0.048	0.218
KNN	44.353	0.162	0.141	0.040	0.199
LR	76.354	0.582	0.100	0.020	0.141
RF	79.333	0.628	0.087	0.018	0.133
SVM	0.116	−0.169	0.196	0.055	0.235
XGB	73.440	0.533	0.100	0.022	0.149

References

1. Misra, N.; Dixit, Y.; Al-Mallahi, A.; Bhullar, M.S.; Upadhyay, R.; Martynenko, A. IoT, big data, and artificial intelligence in agriculture and food industry. *IEEE Internet Things J.* **2020**, *9*, 6305–6324. [CrossRef]
2. Duan, Y.; Edwards, J.S.; Dwivedi, Y.K. Artificial intelligence for decision making in the era of Big Data–evolution, challenges and research agenda. *Int. J. Inf. Manag.* **2019**, *48*, 63–71. [CrossRef]
3. Choi, R.Y.; Coyner, A.S.; Kalpathy-Cramer, J.; Chiang, M.F.; Campbell, J.P. Introduction to machine learning, neural networks, and deep learning. *Transl. Vis. Sci. Technol.* **2020**, *9*, 14. [PubMed]
4. Ching, T.; Himmelstein, D.S.; Beaulieu-Jones, B.K.; Kalinin, A.A.; Do, B.T.; Way, G.P.; Ferrero, E.; Agapow, P.M.; Zietz, M.; Hoffman, M.M.; et al. Opportunities and obstacles for deep learning in biology and medicine. *J. R. Soc. Interface* **2018**, *15*, 20170387. [CrossRef] [PubMed]
5. Bai, Q.; Ma, J.; Liu, S.; Xu, T.; Banegas-Luna, A.J.; Pérez-Sánchez, H.; Tian, Y.; Huang, J.; Liu, H.; Yao, X. WADDAICA: A webserver for aiding protein drug design by artificial intelligence and classical algorithm. *Comput. Struct. Biotechnol. J.* **2021**, *19*, 3573–3579. [CrossRef]
6. Mater, A.C.; Coote, M.L. Deep learning in chemistry. *J. Chem. Inf. Model.* **2019**, *59*, 2545–2559. [CrossRef]
7. Goh, G.B.; Hodas, N.O.; Vishnu, A. Deep learning for computational chemistry. *J. Comput. Chem.* **2017**, *38*, 1291–1307. [CrossRef]
8. Miotto, R.; Wang, F.; Wang, S.; Jiang, X.; Dudley, J.T. Deep learning for healthcare: Review, opportunities and challenges. *Briefings Bioinform.* **2018**, *19*, 1236–1246. [CrossRef]
9. Zou, J.; Huss, M.; Abid, A.; Mohammadi, P.; Torkamani, A.; Telenti, A. A primer on deep learning in genomics. *Nat. Genet.* **2019**, *51*, 12–18. [CrossRef]
10. Eraslan, G.; Avsec, Ž.; Gagneur, J.; Theis, F.J. Deep learning: New computational modelling techniques for genomics. *Nat. Rev. Genet.* **2019**, *20*, 389–403. [CrossRef]
11. Li, H.; Tian, S.; Li, Y.; Fang, Q.; Tan, R.; Pan, Y.; Huang, C.; Xu, Y.; Gao, X. Modern deep learning in bioinformatics. *J. Mol. Cell Biol.* **2020**, *12*, 823–827. [CrossRef] [PubMed]
12. Gawehn, E.; Hiss, J.A.; Schneider, G. Deep learning in drug discovery. *Mol. Inform.* **2016**, *35*, 3–14. [CrossRef] [PubMed]
13. Maia, E.H.B.; Assis, L.C.; De Oliveira, T.A.; Da Silva, A.M.; Taranto, A.G. Structure-based virtual screening: From classical to artificial intelligence. *Front. Chem.* **2020**, *8*, 343. [CrossRef] [PubMed]
14. Patel, L.; Shukla, T.; Huang, X.; Ussery, D.W.; Wang, S. Machine learning methods in drug discovery. *Molecules* **2020**, *25*, 5277. [CrossRef] [PubMed]
15. Pérez-Gandía, C.; García-Sáez, G.; Subías, D.; Rodríguez-Herrero, A.; Gómez, E.J.; Rigla, M.; Hernando, M.E. Decision support in diabetes care: The challenge of supporting patients in their daily living using a mobile glucose predictor. *J. Diabetes Sci. Technol.* **2018**, *12*, 243–250. [CrossRef]
16. Lee, Y.; Ragguett, R.M.; Mansur, R.B.; Boutilier, J.J.; Rosenblat, J.D.; Trevizol, A.; Brietzke, E.; Lin, K.; Pan, Z.; Subramaniapillai, M.; et al. Applications of machine learning algorithms to predict therapeutic outcomes in depression: A meta-analysis and systematic review. *J. Affect. Disord.* **2018**, *241*, 519–532. [CrossRef]
17. Misawa, M.; Kudo, S.E.; Mori, Y.; Cho, T.; Kataoka, S.; Yamauchi, A.; Ogawa, Y.; Maeda, Y.; Takeda, K.; Ichimasa, K.; et al. Artificial intelligence-assisted polyp detection for colonoscopy: Initial experience. *Gastroenterology* **2018**, *154*, 2027–2029. [CrossRef]
18. Ichimasa, K.; Kudo, S.E.; Mori, Y.; Misawa, M.; Matsudaira, S.; Kouyama, Y.; Baba, T.; Hidaka, E.; Wakamura, K.; Hayashi, T.; et al. Artificial intelligence may help in predicting the need for additional surgery after endoscopic resection of T1 colorectal cancer. *Endoscopy* **2018**, *50*, 230–240.
19. Hamet, P.; Tremblay, J. Artificial intelligence in medicine. *Metabolism* **2017**, *69*, S36–S40. [CrossRef]
20. Schork, N.J. Artificial intelligence and personalized medicine. In *Precision Medicine in Cancer Therapy*; Springer: Cham, Switzerland, 2019; pp. 265–283.
21. Khan, O.; Badhiwala, J.H.; Grasso, G.; Fehlings, M.G. Use of machine learning and artificial intelligence to drive personalized medicine approaches for spine care. *World Neurosurg.* **2020**, *140*, 512–518. [CrossRef]
22. Handelman, G.S.; Kok, H.K.; Chandra, R.V.; Razavi, A.H.; Lee, M.J.; Asadi, H. eD octor: Machine learning and the future of medicine. *J. Intern. Med.* **2018**, *284*, 603–619. [CrossRef] [PubMed]

23. Bahri, M.; Salutari, F.; Putina, A.; Sozio, M. AutoML: State of the art with a focus on anomaly detection, challenges, and research directions. *Int. J. Data Sci. Anal.* **2022**, *14*, 113–126. [CrossRef]
24. Alsharef, A.; Aggarwal, K.; Sonia; Kumar, M.; Mishra, A. Review of ML and AutoML solutions to forecast time-series data. *Arch. Comput. Methods Eng.* **2022**, *29*, 5297–5311. [CrossRef] [PubMed]
25. Karmaker, S.K.; Hassan, M.M.; Smith, M.J.; Xu, L.; Zhai, C.; Veeramachaneni, K. Automl to date and beyond: Challenges and opportunities. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–36. [CrossRef]
26. Thiyagalingam, J.; Shankar, M.; Fox, G.; Hey, T. Scientific machine learning benchmarks. *Nat. Rev. Phys.* **2022**, *4*, 413–420. [CrossRef]
27. Truong, A.; Walters, A.; Goodsitt, J.; Hines, K.; Bruss, C.B.; Farivar, R. Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools. In Proceedings of the 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), Portland, OR, USA, 4–6 November 2019; pp. 1471–1479.
28. LeDell, E.; Poirier, S. H2O AutoML: Scalable Automatic Machine Learning. In *Proceedings of the 7th ICML Workshop on Automated Machine Learning (AutoML)*; ICML: San Diego, CA, USA, 2020.
29. Feurer, M.; Klein, A.; Eggensperger, K.; Springenberg, J.; Blum, M.; Hutter, F. Efficient and Robust Automated Machine Learning. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 2962–2970.
30. Real, E.; Liang, C.; So, D.; Le, Q. Automl-zero: Evolving machine learning algorithms from scratch. *Int. Conf. Mach. Learn.* **2020**, *119*, 8007–8019.
31. Wang, C.; Wu, Q.; Weimer, M.; Zhu, E. Flaml: A fast and lightweight automl library. *Proc. Mach. Learn. Syst.* **2021**, *3*, 434–447.
32. Ferreira, L.; Pilastri, A.; Martins, C.M.; Pires, P.M.; Cortez, P. A Comparison of AutoML Tools for Machine Learning, Deep Learning and XGBoost. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; pp. 1–8. [CrossRef]
33. Erickson, N.; Mueller, J.; Shirkov, A.; Zhang, H.; Larroy, P.; Li, M.; Smola, A. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv* **2020**, arXiv:2003.06505.
34. Team, T. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: https://tensorflow.org/ (accessed on 10 November 2024).
35. O'Malley, T.; Bursztein, E.; Long, J.; Chollet, F.; Jin, H.; Invernizzi, L. KerasTuner. 2019. Available online: https://github.com/keras-team/keras-tuner (accessed on 10 November 2024).
36. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
37. Molnar, C. Python Implementation of the Rulefit Algorithm. Available online: https://github.com/christophM/rulefit (accessed on 21 September 2024).
38. Imoscovitz. Ruleset Covering Algorithms for Transparent Machine Learning. Available online: https://github.com/imoscovitz/wittgenstein (accessed on 21 September 2024).
39. xgboost. XGBoost Documentation. Available online: https://xgboost.readthedocs.io/en/stable (accessed on 21 September 2024).
40. King, G.; Zeng, L. Logistic regression in rare events data. *Political Anal.* **2001**, *9*, 137–163. [CrossRef]
41. Menardi, G.; Torelli, N. Training and assessing classification rules with imbalanced data. *Data Min. Knowl. Discov.* **2014**, *28*, 92–122. [CrossRef]
42. He, H.; Bai, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, 1–6 June 2008; pp. 1322–1328.
43. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
44. Singh, D.; Singh, B. Investigating the impact of data normalization on classification performance. *Appl. Soft Comput.* **2020**, *97*, 105524. [CrossRef]
45. Doshi-Velez, F.; Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv* **2017**, arXiv:1702.08608.
46. Carter, A.; Imtiaz, S.; Naterer, G. Review of interpretable machine learning for process industries. *Process Saf. Environ. Prot.* **2023**, *170*, 647–659. [CrossRef]
47. Lipton, Z.C. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue* **2018**, *16*, 31–57. [CrossRef]
48. Gilpin, L.H.; Bau, D.; Yuan, B.Z.; Bajwa, A.; Specter, M.; Kagal, L. Explaining explanations: An overview of interpretability of machine learning. In Proceedings of the 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), Turin, Italy, 1–3 October 2018; pp. 80–89.
49. ALIBI EXPLAIN, Version 0.9.5 Accumulated Local Effects. Available online: https://docs.seldon.io/projects/alibi/en/stable/methods/ALE.html (accessed on 21 September 2024).
50. Apley, D.; Zhu, J. Visualizing the effects of predictor variables in black box supervised learning models. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **2020**, *82*, 1059–1086. [CrossRef]
51. Ribeiro, M.T.; Singh, S.; Guestrin, C. Anchors: High-precision model-agnostic explanations. *Proc. AAAI Conf. Artif. Intell.* **2018**, *32*, 1527–1535. [CrossRef]

52. Mothilal, R.K.; Sharma, A.; Tan, C. Explaining machine learning classifiers through diverse counterfactual explanations. In Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, 27–30 January 2020; pp. 607–617.
53. ALIBI EXPLAIN, Version 0.9.5 Integrated Gradients. Available online: https://docs.seldon.io/projects/alibi/en/latest/methods/IntegratedGradients.html (accessed on 21 September 2024).
54. Ribeiro, M.T. Lime: Explaining the Predictions of Any Machine Learning Classifier. Available online: https://github.com/marcotcr/lime (accessed on 21 September 2024).
55. Scikit-learn. Partial Dependence and Individual Conditional Expectation Plots. Available online: https://scikit-learn.org/stable/modules/partial_dependence.html (accessed on 21 September 2024).
56. Scikit-learn. Permutation Feature Importance. Available online: https://scikit-learn.org/stable/modules/permutation_importance.html (accessed on 21 September 2024).
57. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
58. SHAP. SHAP Documentation. Available online: https://shap.readthedocs.io/en/latest/index.html (accessed on 21 September 2024).
59. Lundberg, S. A unified approach to interpreting model predictions. *arXiv* **2017**, arXiv:1705.07874.
60. Krishna, S.; Han, T.; Gu, A.; Pombra, J.; Jabbari, S.; Wu, S.; Lakkaraju, H. The disagreement problem in explainable machine learning: A practitioner's perspective. *arXiv* **2022**, arXiv:2202.01602.
61. Kurtzer, G.M.; Sochat, V.; Bauer, M.W. Singularity: Scientific containers for mobility of compute. *PLoS ONE* **2017**, *12*, e0177459. [CrossRef] [PubMed]
62. Merkel, D. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.* **2014**, *239*, 2.
63. Hu, G.; Zhang, Y.; Chen, W. Exploring the performance of singularity for high performance computing scenarios. In Proceedings of the 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Zhangjiajie, China, 10–12 August 2019; pp. 2587–2593.
64. Kelly, M.; Longjohn, R.; Nottingham, K. The UCI Machine Learning Repository. Available online: https://archive.ics.uci.edu (accessed on 10 November 2024).
65. Fernandes, K.; Cardoso, J.; Fernandes, J. Cervical Cancer (Risk Factors). Available online: https://archive.ics.uci.edu/dataset/383/cervical+cancer+risk+factors (accessed on 10 November 2024). [CrossRef]
66. Hopkins, M.; Reeber, E.; Forman, G.; Suermondt, J. Spambase. Available online: https://archive.ics.uci.edu/dataset/94/spambase (accessed on 10 November 2024). [CrossRef]
67. Aeberhard, S.; Forina, M. Wine. Available online: https://archive.ics.uci.edu/dataset/109/wine (accessed on 10 November 2024). [CrossRef]
68. Redmond, M. Communities and Crime. Available online: https://archive.ics.uci.edu/dataset/183/communities+and+crime (accessed on 10 November 2024). [CrossRef]
69. Espinase Nandorfy, D.; Watson, F.; Likos, D.; Siebert, T.; Bindon, K.; Kassara, S.; Shellie, R.; Keast, R.; Francis, I. Influence of amino acids, and their interaction with volatiles and polyphenols, on the sensory properties of red wine. *Aust. J. Grape Wine Res.* **2022**, *28*, 621–637. [CrossRef]
70. Pérez-Sánchez, H.; Banegas-Luna, A.J. 164. SIBILA: Investigación y Desarrollo en Aprendizaje Máquina Interpretable Mediante Supercomputación para la Medicina Personalizada [Audio Podcast]. In Investigando la Investigación. Spotify. Available online: https://open.spotify.com/episode/3oRXe7PLpCeK86AT3izn7W (accessed on 10 November 2024).