

VisEvol: Visual Analytics to Support Hyperparameter Search through Evolutionary Optimization

A. Chatzimparmpas¹ , R. M. Martins¹ , K. Kucher¹ , and A. Kerren^{1,2} 

¹Department of Computer Science and Media Technology, Linnaeus University, Sweden

²Department of Science and Technology, Linköping University, Sweden

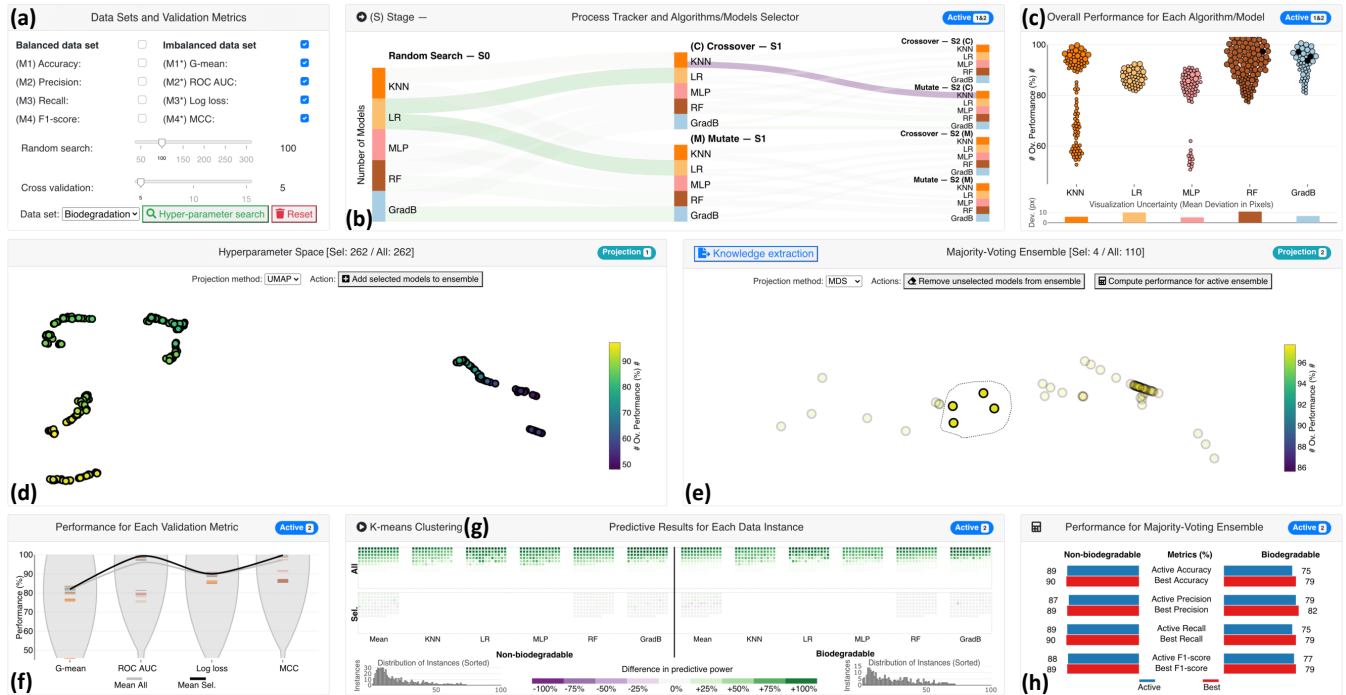


Figure 1: Controlling the evolutionary optimization process for hyperparameter search with VisEvol: (a) the panel for the selection of validation metrics and initialization of random search; (b) the Sankey diagram for the management and analysis of the crossover and mutation procedure; (c) the beeswarm plot with sorted algorithms/models according to overall performance; (d) the projection-based visualization of hyperparameters that aggregates the results of the chosen metrics for all models; (e) the visual embedding of ensembles that includes the handpicked models; (f) the bean plot that presents the performance of models for each metric; (g) the grid-based visualization that displays the predictive power in each instance; and (h) the horizontal bar chart for showing the results of the active vs. the best voting ensemble.

Abstract

During the training phase of machine learning (ML) models, it is usually necessary to configure several hyperparameters. This process is computationally intensive and requires an extensive search to infer the best hyperparameter set for the given problem. The challenge is exacerbated by the fact that most ML models are complex internally, and training involves trial-and-error processes that could remarkably affect the predictive result. Moreover, each hyperparameter of an ML algorithm is potentially intertwined with the others, and changing it might result in unforeseeable impacts on the remaining hyperparameters. Evolutionary optimization is a promising method to try and address those issues. According to this method, performant models are stored, while the remainder are improved through crossover and mutation processes inspired by genetic algorithms. We present VisEvol, a visual analytics tool that supports interactive exploration of hyperparameters and intervention in this evolutionary procedure. In summary, our proposed tool helps the user to generate new models through evolution and eventually explore powerful hyperparameter combinations in diverse regions of the extensive hyperparameter space. The outcome is a voting ensemble (with equal rights) that boosts the final predictive performance. The utility and applicability of VisEvol are demonstrated with two use cases and interviews with ML experts who evaluated the effectiveness of the tool.

CCS Concepts

• **Human-centered computing** → Visualization; Visual analytics; • **Machine learning** → Supervised learning;

1. Introduction

Hyperparameter optimization (also called *hyperparameter tuning*) is the process of selecting appropriate values of hyperparameters for machine learning (ML) models, often independently for each data set, to achieve their best possible results. Although time consuming, this process is required for the vast majority of ML models before their deployment into production [vRH17, vRH18]. Numerous techniques exist that try to solve this challenge, such as the well-known *grid search*, *random search* [BB12], and *Bayesian optimization* that belong to the generic type of *sequential-based methods* [BBBK11, SSW*16]. Other proposed methods include *bandit-based approaches* [FKH18, LJD*17], *population-based methods* [JDO*17], and *evolutionary optimization* [DDF*18, YRK*15], which is our focus in this paper.

Inspired by the biological concept of evolution, one strategy of *evolutionary optimization* keeps the top-ranked models and replaces the remaining, worst-performing models using new hyperparameter sets generated through *crossover* and *mutation* [CK05]. With *crossover*, random pairs of underperforming models (originating from the same algorithm) are picked and their hyperparameters are fused with the goal of creating a better model. As a result, internal regions of the solution space are further explored, and better local optima are investigated. On the other hand, *mutation* randomly generates new values for the hyperparameters to substitute old values. It facilitates scanning for external regions of the solution space to discover additional local optima. These unexplored areas of the hyperparameter space may offer a fresh start to the search for hyperparameters. The synergy of combining both techniques can be beneficial in finding distinctive local optima that generalize to a better result in the end. Hence, the problem of getting stuck in local optima of the hyperparameter space is addressed. However, one question that emerges is: (RQ1) how to choose which models (and algorithms) should crossover and/or mutate, and to what extent, considering we have limited computational resources?

Various automatic ML methods [FH19] and practical frameworks [Com, NNI] have been proposed to deal with the challenge of hyperparameter search. However, their output is usually a single model, which is frequently underpowered when compared to an ensemble of ML models [SR18]. Ensemble methods—such as *bagging* and *boosting*—could be combined in a majority-voting ensemble [CGW13] with a democratic voting system that summarizes the decisions among models. The authors of a recent survey [SR18] state that users should understand how to tune models and, in extension, choose hyperparameters for selecting the appropriate ML ensemble. Consequently, another open question is: (RQ2) how to find which particular hyperparameter set is suitable for each model in a majority-voting ensemble of diverse models?

The optimization of hyperparameters is often performed with the support of a single, specific validation metric (e.g., in Bayesian optimization) [SLA12]. The selection of a proper metric for a task is related to the particular data set, the problem, and the tasks at hand. Thus, the use of a single metric for every data set (such as, e.g., accuracy [MRK06, Stu13]) may result in several problems. The use of multiple metrics, however, poses an extra challenge to such an automatic optimization procedure [FHOM09, PN17, SL09, Tha18], and the comparison and selection between multi-

ple performance indicators are not trivial, even for widely used metrics [DG06, SR15]. Alternatives, such as Matthews correlation coefficient (MCC), might be more informative for imbalanced classification data [CJ20], but even advanced metrics are not the holy grail, and additional challenges can be found in the literature [LJVR08, Pow11]. This leads to one further question: (RQ3) is there any performance improvement from employing several validation metrics that fit better to a specific data set's inherent characteristics?

Evolutionary optimization and majority-voting ensembles inspired us to focus on the three aforementioned questions that constitute open research challenges. In this paper, we present a visual analytics (VA) tool, called *VisEvol* (see Figure 1), that addresses the three research questions described above by supporting the exploratory combination of five different ML algorithms. *VisEvol* uses validation metrics for balanced and imbalanced data sets, and it involves an initial stage of random search (S_0) and two evolutionary generation stages (S_1 and S_2) of hyperparameter settings (see the details in Section 4). To address the three research questions (RQ1–RQ3), *VisEvol* supports the following **workflow** (cf. Figure 2 described in Section 4): (i) the selection and combination of appropriate validation metrics, (ii) the overall exploration of different algorithms and models using diverse hyperparameters, (iii) the inspection of predictive impact for each data instance, (iv) the control of the evolutionary process, and (v) a final phase where the performance of the current best ML ensemble is compared to the currently active majority-voting ensemble. In summary, our contributions consist of the following:

- the systematization of hyperparameter search using evolutionary optimization with a coherent visual analytic workflow;
- an implementation of the aforementioned conceptual proposal, our VA tool called *VisEvol*, that consists of a novel combination of interactive coordinated views—which control the crossover and mutation processes—and supports the visual exploration of the most performant/diverse models for the creation of a powerful majority-voting ensemble;
- a demonstration of the applicability of our proposed system with two use cases, using real-world data, that confirm the effectiveness of controlling the process of evolutionary optimization of hyperparameters and testing different ML ensembles; and
- the discussion of the methodology of the interviews and the positive and supportive feedback received from three ML experts.

The rest of this paper is organized as follows. In Section 2, we discuss relevant techniques for visualizing hyperparameter tuning and existent automatic approaches. Afterwards, in Section 3, we describe the analytical requirements and design goals for attaching VA to evolutionary optimization and combining VA with ensemble learning. Section 4 presents the functionalities of the tool and, at the same time, describes the first use case with the goal of selecting a composition of models (with specific hyperparameters) for the creation of a majority-voting ensemble using medical data. Thereafter in Section 5, we demonstrate the applicability and usefulness of *VisEvol* with another real-world data set focusing on biodegradation of molecules. Next, in Section 6, we review the feedback our VA tool obtained during the interview sessions by summing up the experts' opinions and the limitations that guide us to possible future directions for *VisEvol*. Finally, Section 7 concludes our paper.

2. Related Work

Visualization tools have been implemented for sequential-based, bandit-based, and population-based approaches [PNKC21], and for more straightforward techniques such as grid and random search [LCW*18]. Evolutionary optimization, however, has not experienced similar consideration by the InfoVis and VA communities, with the exception of more general visualization approaches such as EAVis [KE05, Ker06] and interactive evolutionary computation (IEC) [Tak01]. To the best of our knowledge, there is no literature describing the use of VA in hyperparameter tuning of evolutionary optimization (as defined in Section 1) with the improvement of performance based on majority-voting ensembles. In this section, we review prior work on automatic approaches, visual hyperparameter search, and tools with which users may tune ML ensembles. Finally, we discuss the differences of such systems when compared to VisEvol in order to clarify the novelty of our tool.

Automatic Approaches. In the ML community, most of the research is geared towards fully-automated hyperparameter search with no human interaction [CSP*14]. It is true that automatic techniques present encouraging results and are successful on tuning hyperparameters of some models [BBKS13, YM14], for example, by automatically finding optimal deep learning hyperparameters using genetic algorithms [FBO*07, YRK*15]. Important contributions of this research include the formalization of primary concepts [CDM15], the identification of methods for assessing hyperparameter importance [JWXY16, PBB19, vRH17, HHLB13, HHLB14, vRH18], and resulting libraries and frameworks for specific hyperparameter optimization methods [KGG*18, THHLB13]. Indeed, several packages exist that focus on automatically optimizing Bayesian methods with the use of a single performance measurement [Bay, HHLB11, HHLBS09, SSW*16], and there are popular commercial platforms developed for hyperparameter optimization [Dat, Aut]. This widespread automation does not stop in supervised classification problems, but also includes dimensionality reduction (DR) algorithms (e.g., t-SNE) [BCA*19, KB19].

Despite the success of automatic approaches and their advancement through the years, it is important to note that such approaches require extensive computing power and may lack critical features. Automatically (or manually) set thresholds may discard different models which could be informative but theoretically seem to perform worse than the rest. Moreover, the ranking of models is often based on a single validation metric, leading to the risks discussed in Section 1. The aforementioned works that make use of genetic algorithms contain similar mechanisms as in VisEvol, but without VA support for (1) the exploration of the interconnected hyperparameters, and (2) the selection of the proper number of models that should crossover and mutate.

Visual Hyperparameter Searching. ATMSeer [WMJ*19] implements a multi-granularity visualization for model selection and hyperparameter tuning. It is a visualization tool coupled with a backend framework, called ATM [SDC*17], that allows the users to interact with the middle steps of an AutoML process and control them by adjusting the search space dynamically during execution time. In contrast to VisEvol, it only supports a single performance measurement, and the output is a single optimized model.

One common focus of related work is the hyperparameter search for deep learning models. HyperTuner [LCW*18] is an interactive VA system that enables hyperparameter search by using a multi-class confusion matrix for summarizing the predictions and setting user-defined ranges for multiple validation metrics to filter out and evaluate the hyperparameters. Users can intervene in the running procedure to anchor a few hyperparameters and modify others. However, this could be hard to generalize for more than one algorithm at the same time. In our case, we combine the power of diverse algorithms, with one of them being a neural network (NN). HyperTendrill [PNKC21] is a visualization tool that supports random search, population-based training [JDO*17], Bayesian optimization, HyperBand [LJD*17], and the last two methods joined together [FKH18]. It enables the users to set an initial budget, search the space for the best configuration, and select suitable algorithms. However, its effectiveness is only tested in scenarios specifically designed for NNs. Other examples of publications which work explicitly with deep learning only, and do not support evolutionary optimization, are VisualHyperTuner [PKK*19], Jönsson et al. [JES*20], and Hamid et al. [HDK*19].

The use of parallel coordinates plots [ID87] is rather prominent for the visualization of automatic hyperparameter tuners such as HyperOpt [BKE*15]. Most of the time, less interactive visualizations have been developed for monitoring automatic frameworks [ASY*19, GSM*17, KKP*18, LLN*18, LTKS19, TBCT*18]. Visualizations arranged into dashboard-styled interfaces are the preferred norm for managing ML experiments and their associated models [SKJ*17, TMB*18, WRW*20, WWO*20]. Automated approaches exclude the user from exploring and refining the hyperparameter search, and the visual representation of automation happens in the form of visualization of the already computed results.

Human-in-the-Loop Ensemble Learning. There are relevant works that involve the human in interpreting, debugging, refining, and comparing ensembles of models [DCCE19, LXL*18, NP20, SJS*18, XXM*19, ZWLC19]. These papers use bagging [Bre01] and boosting [CG16, FSA99, KMF*17] techniques for ranking and identifying the best combination of models in different application scenarios. StackGenVis [CMKK21] is a VA system for composing powerful and diverse stacking ensembles [Wol92] from a pool of pre-trained models. On the one hand, we also enable the user to assess the various models and build his/her own ensemble of models. On the other hand, we support the process of generating new models through genetic algorithms and highlight the necessity for the best and most diverse models in the simplest possible voting ensemble. Finally, our approach is model-agnostic and generalizable, since we use both bagging and boosting techniques along with both NNs and simpler models [LXL*18, NP20, ZWLC19].

VA tools have also been developed to visualize buckets of models [CAA*19, TLKT09, ZWM*19], where the best model for a specific problem is automatically chosen from a set of available options. These works feature exploration of the space in search for a final model, but the best model might not be the optimal when compared to a set of models (i.e., multiple hyperparameters) from several algorithms. Additionally, the models are already generated before the exploration, and there is no involvement of an optimization method.

3. Analytical Requirements and Design Goals

In this section, we define the main analytical requirements that a VA system should tackle for supporting evolutionary optimization of hyperparameters. Then, we describe the corresponding design goals that directed the development of our proposed VisEvol tool.

3.1. Analytical Requirements for Evolutionary Optimization

The analytical requirements (**R1–R5**) originate from the analysis of the related work in Section 2, including the three analytical needs from Park et al. [PNKC21], the three key decisions from Wang et al. [WMJ*19], and the five sub-steps from Li et al. [LCW*18]. Also, our own experiences played a vital role, for instance, VA tools for ML such as t-viSNE [CMK20] and StackGenVis [CMKK21], and recently-conducted literature reviews [CMJK20, CMJ*20].

R1: Identify effective hyperparameters. Interviews performed by Park et al. [PNKC21] showed that users usually sort the models based on a validation metric and then check the hyperparameters of the most performant models (commonly less than 10) for the generated outcomes. Next, they select the hyperparameter spaces close to the already explored ones to find more effective hyperparameters using more computational resources. However, they cannot be sure whether the updated solution spaces would produce better models, as they might have missed searching for a critical space with better hyperparameters. Another crucial step is to drop underperforming hyperparameter settings from the candidates or, even better, to re-process them to become more robust. With crossover and mutation, the final task can be effectively accomplished (see **R3**).

R2: Build an initial ensemble of performant and diverse models. Wang et al. [WMJ*19] stated that automatic ML approaches yield the model with the highest performance score by default. Nevertheless, the users could prefer more stable algorithms regarding the adaptations of hyperparameters. There might even be a case of an underperforming model that could perform better when coupled with another model. Those patterns could be lost if only the single best model is available for preview by the user.

R3: Send the remaining models for improvement and handle crossover and mutation procedures. Configuring hyperparameter optimization methods was found unpredictable and disturbing by the interviewees of the investigation by Park et al. [PNKC21]. Participants from the interview by Wang et al. [WMJ*19] stated that they revised the hyperparameter space based on previous knowledge. For this requirement and **R2**, the users should be able to analyze different algorithms thoroughly and decide to what degree they are going to crossover and mutate each algorithm according to prior knowledge and feedback from VA tools.

R4: Contrast the results of all model-generation stages and update the majority-voting ensemble. In evolutionary optimization, a crossover and mutation phase leads to a propagation of more crossover and mutation phases with exponential growth (cf. Figure 1(b)). Li et al. [LCW*18] found that once the ML expert has acquired all the results from an execution stage, he/she should analyze them with various perspectives and decide if the previously explored models' performance match his/her needs. If not, then more stages should be involved in the process until his/her expectations

are met. This entire process should be trackable and manageable from the user's side. The best models (according to the user) are accumulated in a final bucket, forming a majority-voting ensemble.

R5: Validate and select a final configuration (single model or combination of models). Automatic ML commonly delivers to users a model with the highest performance according to a single metric (e.g., accuracy), but fails to take into account other characteristics of models [PNKC21]. In practice, users want to consider several model features and validation metrics for selecting a model (or models). Thus, the users want to examine collections of validation metrics and explore model ensembles in various granularities.

3.2. Design Goals for VisEvol

To fulfill the described analytical requirements (**R1–R5**), specifically in the context of VA and ensemble learning, we have derived five design goals (**G1–G5**) to be tackled by our tool. The implementation of these design goals is described in Section 4.

G1: Analysis of predictions and validation metrics for the identification of effective hyperparameters. We aim to support the exploration of algorithms and models with various hyperparameters (**R1**) as follows: (1) illustrate the performance of each algorithm and model based on multiple validation metrics chosen by the user; (2) project the models into a hyperparameter embedding according to the previous overall performance using DR methods; (3) compare the mean performance of all algorithms and models vs. a selection of models for every metric; and (4) analyze the predictive results for each instance and for all models against a selection of models with regard to the difference in predictive power.

G2: Migration of powerful and alternative models to the majority-voting ensemble. In continuation of the preceding goal, our VA tool should allow the users to pick the best (and most diverse) models for the ensemble from different areas in the projection (**R2**). Using the other coordinated views, the user can compare the selected models against all models and act accordingly.

G3: Transformation of underperforming models via crossover and mutation. Users are able to conduct similarity-based analyses relevant to the algorithms' predictions, which are the initial indications for tuning crossover and mutation of Stage 1 (S_1). The various perspectives that multiple views have to offer for several algorithms further support those prior analyses. We aim to provide explicit visual feedback from S_1 of crossover and mutation toward S_2 for users to select appropriate numbers for model optimization and test their hypotheses (**R3**).

G4: Comparison of multi-stage generated hyperparameter sets in various granularities. An addition to **G1** is that the positive or negative impact of performance should be measured during the creation of models through the multi-stage crossover and mutation procedure. VisEvol should thus display both successful and underperforming paths for every crossover and mutation stage (**R4**).

G5: Extraction of an ultimate model or a voting ensemble with a side-by-side performance comparison. A comparison between the currently active ensemble against the optimal solution found until that point in time should be established in our tool to assist the extraction of a competitive and effective ensemble (**R5**).

4. VisEvol: System Overview and Application

Following our analytical requirements and derived design goals, we have developed VisEvol, an interactive web-based VA tool that allows users to utilize evolutionary optimization in order to search for effective hyperparameters. It is implemented in JavaScript using the Vue.js [vue14] framework and a combination of D3.js [D311] and Plotly.js [plo10] visualization libraries for the frontend. The backend is implemented in Python using Flask [Fla10] as the web framework and Scikit-Learn [PVG* 11] as the ML library.

The tool consists of eight main interactive visualization panels (Figure 1): (a) data sets and validation metrics (\rightarrow G1), (b) process tracker and algorithms/models selector (\rightarrow G3), (c) overall performance for each algorithm/model, (d) hyperparameter space, (e) majority-voting ensemble (\rightarrow G2), (f) performance for each validation metric, (g) predictive results for each data instance (\rightarrow G4), and (h) performance for majority-voting ensemble (\rightarrow G5). We propose the following **workflow** for the integrated use of these panels (cf. Figure 2): (i) choose suitable validation metrics for the data set, which are then used for validation during the entire process (Figure 1(a)); (ii) in the next exploration phase, compare and choose specific ML algorithms for the ensemble and then proceed with their particular instantiations, i.e., the models (see Figure 1(c–e)); (iii) during the detailed examination phase, zoom in into interesting clusters already explored in the previous phase, and focus on indications that confirm either their approval in the ensemble or their need for transformation through the evolutionary process (cf. Figure 1(f and g)); (iv) control the evolutionary process by setting the number of models that will be used for crossover and mutation in each algorithm (Figure 1(b)); and (v) compare the performances of the *best* so far identified ensemble against the *active* majority-voting ensemble in Figure 1(h). This is an iterative process with a final composition of the most performant and most diverse majority-voting ensemble. The generated knowledge regarding hyperparameters is fed back to the user, whose trust in the results increases, and he/she stops when his or her expectations are met. The individual panels and the workflow are discussed in more detail below.

The user interface of VisEvol is structured as follows: (1) two projection-based views, referred to as *Projections 1 and 2*, occupy the central UI area (cf. Figure 1(d and e)); (2) active views relevant for *both* projections are positioned on the top (cf. Figure 1(b and c)); and (3) commonly-shared views that update on the exploration of *either* Projection 1 or 2 are placed at the bottom (see Figure 1(f and g)). Thus, Figure 1(h) is always active for Projection 2, as it is related to the majority-voting ensemble. *Soft majority voting* strategy (i.e., predicted probabilities) is always applied.

To exploit the model-agnostic nature of our proposed workflow, VisEvol supports five different supervised ML algorithms (any could have been used): (1) a neighbor classifier (*k-nearest neighbor (KNN)*), (2) a linear classifier (*logistic regression (LR)*), (3) an NN classifier (*multilayer perceptron (MLP)*), and (4) two ensemble classifiers (*random forest (RF)* and *gradient boosting (GradB)*). The primary hyperparameters used for mutation: *number of neighbors* for KNN, *inverse of regularization strength* for LR, *hidden layer sizes* for MLP, *number of decision trees* for RF, and *number of boosting stages* for GradB.

In the following subsections, we explain the system by using a

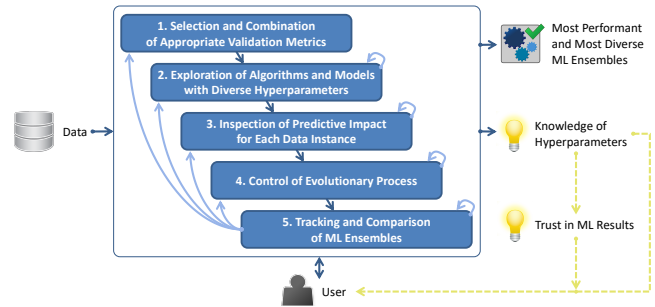


Figure 2: The VisEvol workflow allows the users to construct performant and diverse ML ensembles, gain knowledge about the hyperparameters chosen via the evolutionary optimization process, and thus gain trust in the respective ML results. The users are capable of interacting with all phases iteratively, represented by the multiple arrows inside the box.

running example with the *heart disease* healthcare data set obtained from the UCI Machine Learning repository [DG17]. The data set represents a binary classification problem and consists of 13 numerical features/attributes and 303 instances. It is rather balanced, with 138 patients being healthy and 165 having a diseased heart.

4.1. Data Sets and Validation Metrics

Support for (1) selecting proper validation metrics for balanced and imbalanced data sets and (2) directing the experts' attention to different classes for the given problem constitute two of the critical open challenges in ML. For instance, accuracy is preferred to the *g-mean* metric for a balanced data set [BDA13]. In another example, a medical expert might focus more on eliminating false-negative predictions than false-positives (e.g., a patient being actually ill but predicted as healthy) with a bad impact on the latter. However, this trade-off is necessary when considering a person's life.

In VisEvol, up to eight different metrics can be used simultaneously, depending on the number of instances falling into each class of a binary classification problem. The available metrics are divided into two groups: balanced data sets (\rightarrow *accuracy*, *precision*, *recall*, and *f1-score*) and imbalanced data sets (\rightarrow *g-mean*, *ROC AUC*, *log loss*, and *MCC*). For the initialization of VisEvol, the user should direct his/her attention to the top-left panel shown in Figure 1(a); the preferable group of validation metrics will depend on the distribution of instances in the two classes for each individual data set (Step 1 in Figure 2). Then he/she sets a number of models n with the slider shown in Figure 1(a), from 50 to 300, which will impact the initial search of random hyperparameters. Choosing a value n is a matter of finding a balance between spending computational resources and time against scanning more accurately the solution space for better hyperparameter tuples. The k value is used for the k -fold cross-validation, with the options of 5, 10, or 15 folds.

In our running example, knowing that the data set is balanced, we use the first group of metrics (currently unselected in Figure 1(a)), and leave the default random search to 100 models per algorithm, leading to 500 models in total. The k value is set to 10 because we want to precisely compare our results with recent work [LJ19].

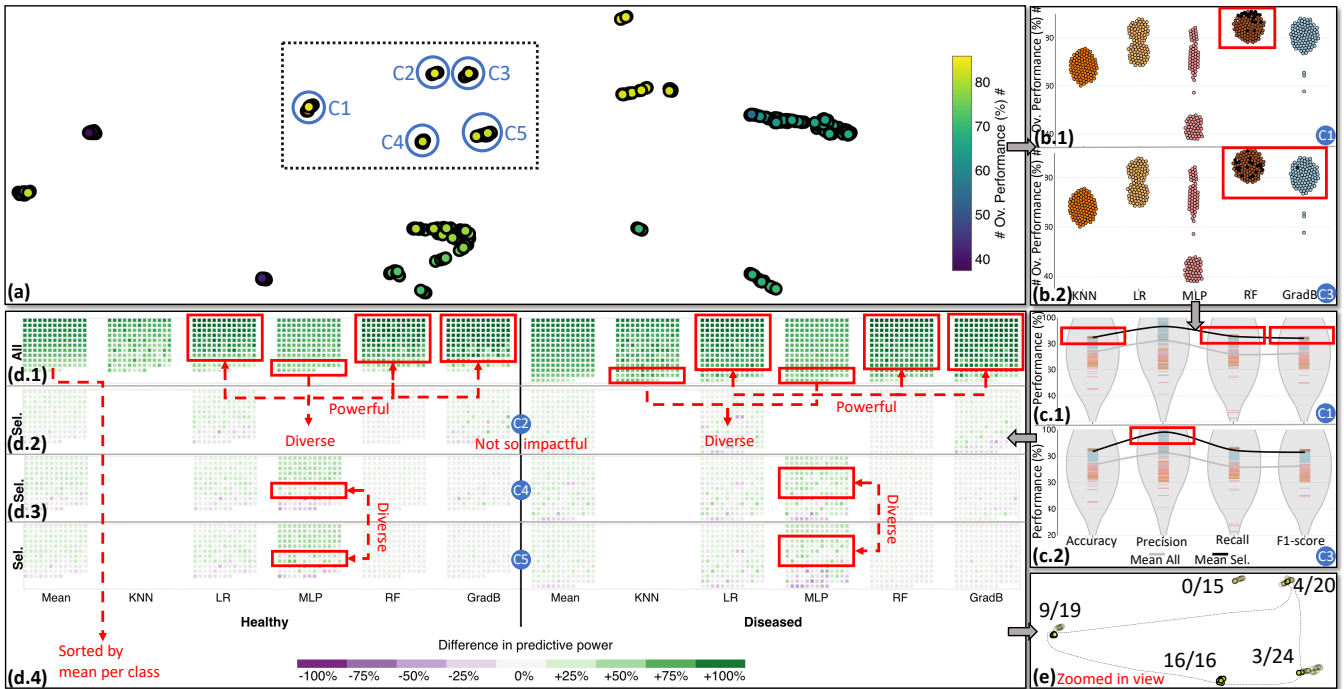


Figure 3: Exploration of ML models with VisEvol. View (a) presents a selection of similar and better-performing models in several clusters. (b.1) indicates that $\textcircled{C1}$ contains well-performing RF models, in contrast to (b.2), in which $\textcircled{C2}$ includes more diverse RF models and a GradB model. For the accuracy, recall, and f1-score metrics, $\textcircled{C1}$ performs much better than the average, based on the bean plots in (c.1). However, $\textcircled{C2}$ achieves better results for the precision metric. In the grid-based view (d.1), LR, RF, and GradB algorithms appear more powerful than other algorithms that are more diverse due to the good predictions of hard-to-classify instances. $\textcircled{C2}$ seems redundant because of the $\textcircled{C1}$ and $\textcircled{C3}$ that improve similar cases (d.2). If we look at (d.3) and (d.4), both visualizations display MLP models that enhance the predictive power of different instances in both classes. Finally in view (e), we mix models from the multiple explored clusters to create the first voting ensemble.

4.2. Hyperparameter Space

To provide a holistic view on the performance of the models for the selected validation metrics, we use a UMAP [MHM18] projection, as seen in Figure 3(a), that consists of the 500 randomly-sampled models (MDS [Kru64] and t-SNE [vdMH08] are also available). Each model uses a set of particular hyperparameters, and it is projected from the space of validation metric values (here 4 dimensions, but could be more). Thus, groups of points represent clusters of models that perform similarly according to all the metrics. The plot uses the Viridis colormap [LH18] to show the average performance of each model according to all selected metrics. This view provides the user with an overview of the hyperparameter space and ability to look for previously-unknown patterns. We can now select high-performing clusters and proceed with deciding which models to include in our ensemble (Step 2 in Figure 2).

At this phase, we want to confirm precisely the cluster affiliation and the relationship with the overall performance (here, the average of 4 validation metrics) for all the models. To achieve that, the beeswarm plots in Figure 3(b.1 and b.2) arrange the models according to the distinct algorithms in the x-axis and sort the models based on the overall performance along the y-axis (abbreviated to *Ov. Performance*). Similarly-performing models can overlap in this view (due to the y-axis values), so we apply a force-based layout

algorithm to make sure they are visible. However, moving models around introduce uncertainty in this view. Thus, we visualize the mean deviation in pixels for every algorithm (cf. Figure 1(c), bar chart at the bottom) to minimize any misleading visualization bias. Figure 3(b.1) suggests that $\textcircled{C1}$ contains the most performant RF models, while Figure 3(b.2) presents less performant RF and GradB models situated in $\textcircled{C2}$ that may, however, be more diverse (black dots represent selected models).

For clusters $\textcircled{C1}$ and $\textcircled{C2}$, we want to find out the relation with the different validation metrics. This task is supported by the bean plots in Figure 3(c.1 and c.2), which are separated on the x-axis by the selected metrics from the validation metrics panel (see Figure 1(a)). The *beans* (lines inside each bean plot) represent all the individual models. However, they were not designed with a goal to accurately represent each individual model, since there are cluttering problems due to the thickness of the line for each bean and the large number of models per bean plot. The main purpose of this plot is to check if the mean values for a selection are better when compared to the overall mean, as shown in Figure 3(c.1). For $\textcircled{C1}$ (Figure 3(c.2)), the precision metric is further improved compared to $\textcircled{C2}$.

At this point, the importance of $\textcircled{C1}$ and $\textcircled{C2}$ is clear, so we decide to gradually scan for the in-depth connections of the models belonging to the remaining clusters and the data instances (Step 3

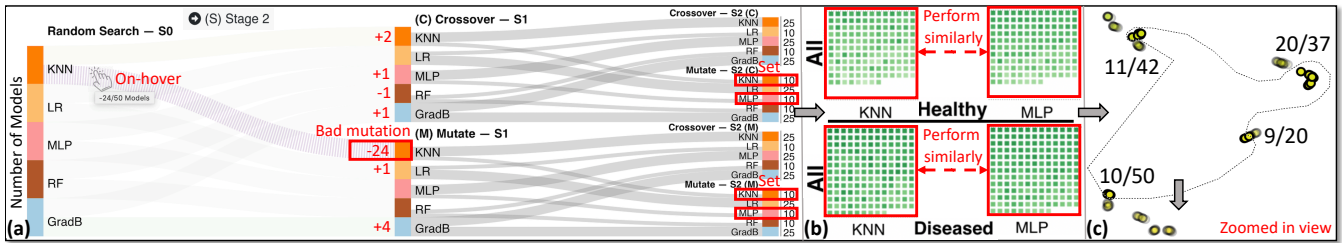


Figure 4: Tuning the crossover and mutation process toward S_2 . In (a), we set fewer models for mutation and more for crossover for both KNN and MLP algorithms. Our choice is motivated by the feedback received from the bad KNN mutation in S_1 and the fact that KNN and MLP perform almost identically for both independent classes (as illustrated in (b)). Similar to Figure 3, we investigate clusters in the projection, select a few models from each explored cluster shown in (c), and send the rest for crossover and mutation.

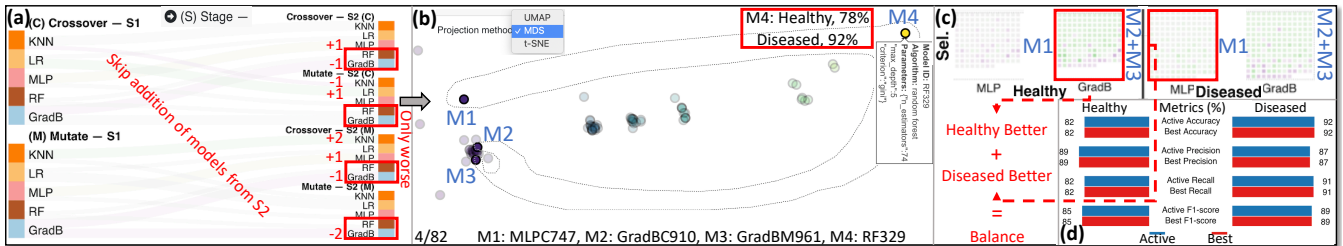


Figure 5: The outcome of the S_2 evolutionary optimization procedure and the final voting ensemble: (a) highlights that we have reached an impactful solution, since the models are not getting significantly better. Thus, we skip the addition of models from S_2 . In (b), after an extensive exploration of the majority-voting ensemble, we end up with the selection of four models: M_4 originating from the initial random search (the most performant when used individually) and M_1 – M_3 from the crossover and mutation processes at S_1 . We narrow down this selection even further by examining (c), where one MLP model appears to perform better for the Diseased class and two GradB models for the Healthy class. The active performance matches the best performance found so far (d). Hence, this is the most powerful majority-voting ensemble.

in Figure 2). The grid-based visualization in Figure 3(d.1) focuses on the exploration of the predictive power (0%–100%) over the data set’s instances represented with white to dark green colors. If a data set contains fewer than 169 instances per class, then we display all of them in the grid. Otherwise, in order to scale the visualization to larger data sets (e.g., our next use case), we first build a grid with a fixed number of cells (100). Then, we use *K-means clustering* to place the data set’s instances within these cells, i.e., we create 100 clusters, one per cell of the grid. This action is noticeable due to specific play/stop glyphs, as illustrated in Figure 1(g), top-left corner. There is one grid per different ML algorithm, plus one grid for the overall values (leftmost grid in Figure 3(d.1)), and all the instances are sorted according to the mean performance for all the explored models. Each cell of the grid (as shown in Figure 3(d.2–d.4)) then presents the computed difference in predictive power for all its instances (from –100% to +100%) for the selected against all models. The color-encoding diverges from purple to green for negative to positive difference. In case the *K-means clustering* functionality is active, we use bar charts to depict the distribution of instances in the 100 individual cells (see Figure 1(g), bottom). Afterwards, the predictive power for every cell is computed on average from all the instances that belong to it.

From Figure 3(d.1), we observe that KNN and MLP contain more diverse models (darker green color for instances at the bottom), because they better predict hard-to-classify instances when

compared to LR, RF, and GradB (which work better for the easy-to-predict instances). Since we have already found powerful and diverse RF and GradB models from the prior analyses in (a) and (b), we now focus the MLP algorithm. For (c), we were unable to find any impactful models, i.e., only a tiny amount of instances are green-colored in Figure 3(d.2). Models originating from (a) and (b) sufficiently cover the need for diversity, including MLP models. We then pick the models shown in Figure 3(e) for our voting ensemble, and the remaining models are updated by the evolutionary optimization (see next subsection). This action concludes S_1 .

4.3. Process Tracker and Algorithms/Models Selector

After the initial generation of hyperparameter settings with the use of random search (S_0), we manually evaluated the models and forwarded the remaining unselected models for crossover and mutation [CK05]. As explained in Section 1, crossover blends randomly different models from the same algorithms (or else it is impossible due to differentiation in hyperparameters), and mutation captures the primary hyperparameter (previously mentioned in Section 4) and randomly mutates it with new values, which were previously not explored. This procedure repeats for every algorithm separately.

In the Sankey diagram (see Figure 4(a)), the user tracks the progress of the evolutionary process and is able to limit the number of models that will be generated through crossover and mutation

for each algorithm (Step 4 in Figure 2). The default here is defined as *user-selected random search value* / 2 for each algorithm, to sustain the vertical symmetry in the Sankey diagram, as shown in Figure 4(a), left. For S_1 , we choose to keep the default values for crossover and mutation, but an analyst with prior knowledge and experiences could fine-tune this process. While moving toward S_2 , we notice from Figure 4(b) that KNN and MLP perform similarly. The output of S_1 in Figure 4(a) becomes the input for S_2 , assisting us in the selection of appropriate numbers for model generation for S_2 . When we hover over a path of the Sankey diagram, we see how many models perform better or worse than the already-explored models for each particular algorithm. The color-encoding is the same as Figure 3(d.2–d.4)), and it is measured as the *number of overperforming models compared to the initial models / total crossover or mutate models* for each algorithm. If there are no overperforming models, then we show *number of underperforming models compared to the initial models / total crossover or mutate models* for each algorithm. This approach primarily allows the user to identify how many models are improved based on each transformation (crossover or mutation), but it also highlights cases with very bad results from crossover or mutation, where no better ML models could be found. In our example, KNN mutation produced bad results, hence, we set the subsequent KNN and MLP mutations (due to the previously-discussed similarity in Figure 4(a)) to lower values than the default (10 vs. the default of 25). The visualization reduces the width of each path line in the Sankey diagram accordingly when the values are smaller than the maximum permitted. Next, we apply an equivalent procedure for all the algorithms. Finally, an analysis is conducted in a similar way to previous sections, with the selection of points in Figure 4(c) as an outcome.

4.4. Majority-Voting Ensemble

From Figure 5(a), right, we see that only a few KNN, LR, and MLP models were better than the previous stages. Thus, we conclude that there is no further improvement, and it is hard to find better hyperparameter tuples. We skip the addition of models from S_2 to the final ensemble because RF and GradB seem to perform better overall. In Figure 5(b), we switch the embedding to an MDS projection which favors the global structure, and compare clusters of models until we discover the active ensemble that contains M1–M4 models (using the *Compute performance for active ensemble* button present in Figure 1(e)). Figure 5(c) suggests that M2+M3 are better for the *Healthy* class, while M1 is better for the *Diseased* class. M4 is somewhere in-between but very powerful overall. By keeping the balance in this ensemble, we achieve the highest recorded performance for our analysis (cf. horizontal bar chart in Figure 5(d)). The symmetric horizontal bar chart is split vertically based on the different metrics. The left-side is about one target class and the right side for the other one. Blue is always used for the actively explored ensemble combination while red is for the best ensemble found yet. The comparison of both serve the purpose of identifying exceptionally performing majority-voting ensembles (Step 5 in Figure 2).

4.5. Performance of Majority-Voting Ensemble

Latha and Jeeva [LJ19] tried out various ensembles for this same data set, with or without (as in our case) feature selection. They

found that applying majority vote with the NB, BN, RF, and MLP algorithms was the best combination, achieving $\approx 82\%$ accuracy without feature selection. However, they do not state how many models were used in the composition of this ensemble. With VisEvol, we reached an accuracy of 87% with only 4 ML models (see Figure 5(d)), thus surpassing their majority-voting ensemble. If the user wants to utilize one model, our selection would have been M4:RF329 (see Figure 5(b), top-right), which has a combined predictive accuracy of 85%. This shows that our VA approach can be effective in searching for hyperparameters and building powerful, simple, and diverse voting ensembles.

5. Use Case

In this section, we describe how VisEvol can be used to improve the results of a study about the relationships between *chemical structure and biodegradation of molecules*, when compared to previous work from Mansouri et al. [MRB*13]. The *QSAR (Quantitative Structure Activity Relationships) biodegradation* data set represents a binary classification problem where molecules are assigned to either the *Biodegradable* or *Non-Biodegradable* classes. The class distribution is rather imbalanced, with 284 degradable and 553 non-degradable molecules for the training set that contains 41 diverse features. For their solution, the authors trained three ML models (KNN, PLSDA, and SVM) and then combined their results using two consensus. We contrast their first consensus with our majority-voting ensemble, and we compare our results with the same validation metrics for the unseen test and external validation data, which simulate a real-world situation.

Exploration and Selection of Algorithms and Models. Similar to the workflow described in Section 4, we start by setting the most appropriate validation metrics for the imbalanced data set (see Figure 1(a)). The projection in Figure 6(a) offers an overview of the high performing clusters that need further investigation (Ⓒ₁–Ⓒ₅). By looking at Figure 6(b.1), we infer that Ⓒ₃ contains KNN and GradB models that perform worse than the remaining models, in general. For Ⓒ₃, *MCC* and *log loss* are very high when compared to the *g-mean* and *ROC AUC* metrics, as shown in Figure 6(c.1). This can be further explained if we dig into this cluster's performance for each instance. Those models create an illusion of performing well for the hard-to-classify instances (Figure 6(d.1)). Nevertheless, the previously spotted low values in *g-mean* and *ROC AUC* suggest that those models reach high precision but with low recall, or vice versa; hence, they should be avoided. On the contrary, Figure 6(b.2) presents a blend of performant models, reaching an equilibrium state where all the values for the metrics are concurrently high (cf. Figure 6(c.2)). The explored GradB models, for example, improve the *Biodegradable* class and accomplish a balance between the two mean values of the bins for both classes (see Figure 6(d.2)). This is especially true when we observe that the distributions of the instances (based on 100 clusters generated by KNN) are long-tailed. That means most instances belong in the first sorted bins, which are predicted better than the following bins (as described in Section 4).

Tuning the Evolutionary Optimization Process. After S_1 's default execution with 100 models for each algorithm (50 due to crossover and 50 because of mutation), we continue with setting the next batch of crossover and mutation processes. We received useful

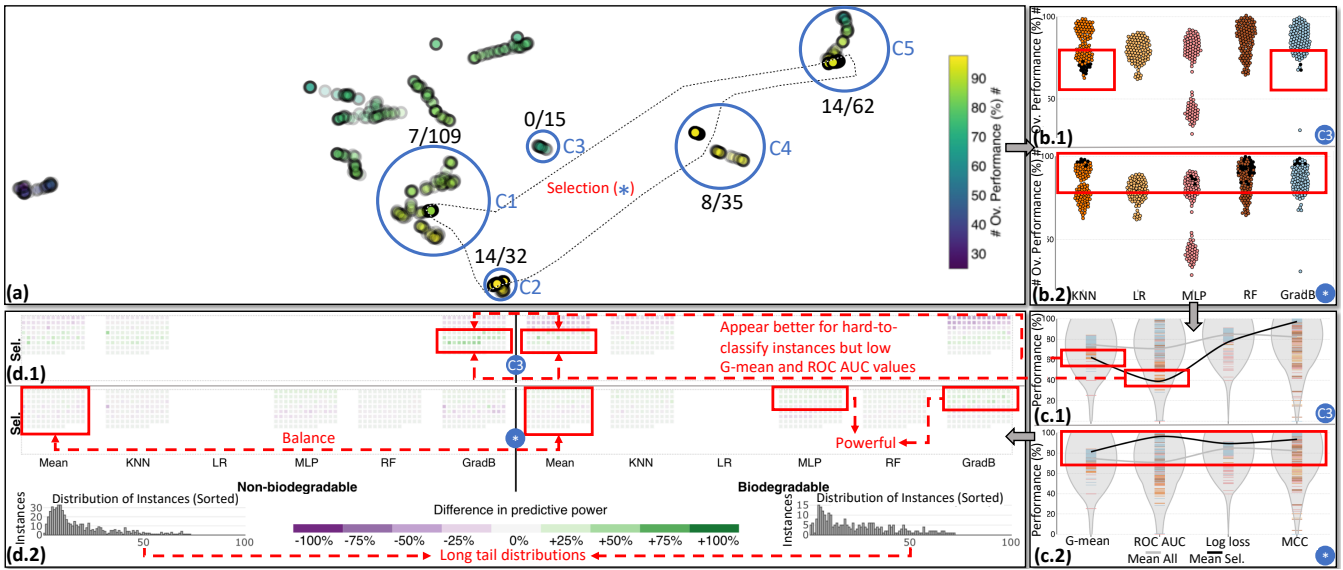


Figure 6: The exploration of clusters of interest that contain performant ML models. View (a) presents the user’s selection that drive the analyses performed in the remaining subfigures. (b.1) provides an overview of the performance, showing that © has underperforming KNN and GradB models. On the other hand, (b.2) shows that the user’s choice of models retains both performance and diversity. In (c.1), we observe that g-mean and ROC AUC scores are very low, which is a problem investigated further in view (d.1). Those models appear to perform better for the hard-to-classify instances; however, this is a misconception. (c.2) gives supporting evidence to the user’s selection, since all validation metrics are higher than the average values for all models, along with the in-depth visualization in (d.2).

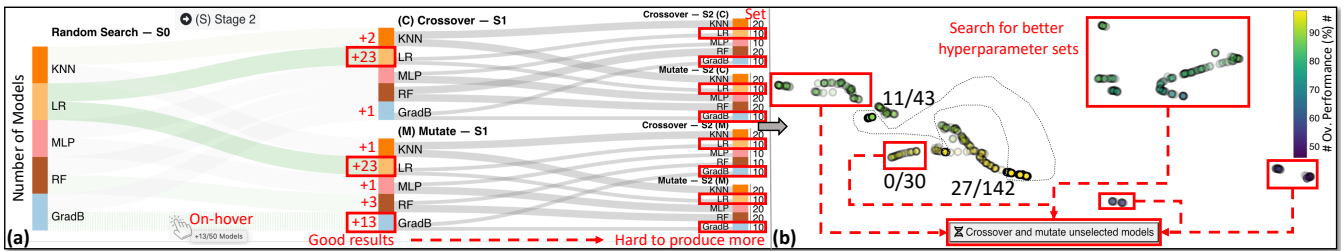


Figure 7: Setting crossover and mutation values for S_2 evolutionary optimization. The Sankey diagram’s feedback in (a) suggests that new hyperparameter sets that perform better than the current models will be hard to produce. Despite that, after user’s selection in (b), the leftover models crossover between each other and mutate over successive evolutions.

feedback from S_1 that supports us in creating new models for S_2 in Figure 7(a), left. When we hover over a mutation path for GradB, we see that 13 out of 50 models perform better than the initial ones from random search. This could mean that it is hard to produce new models that will outperform the previous production of stable and robust models. The same applies to the LR model, for both crossover and mutations paths (with 23 out of 50 models). Thus, we choose to set the production from 25 models to 10 for both LR and GradB algorithms. Similarly to the previous paragraph, we select well-performing and diverse models as shown in Figure 7(b), and the unselected models are being used in the crossover and mutation method based on the previously adjusted parameters.

Examining the Influence of Diversity. In Figure 1(b), we see that most paths fail to create extra powerful models, indicating that it will be hard to find better models. The GradB algorithm seems to have generated a few enhanced performance models from the

crossover path to mutation at S_2 (light-green color). After another hyperparameter space search (see Figure 1(d) with the help of supporter views (Figure 1(c, f, and g)), out of the 290 models generated in S_2 , we select 28 to add to the ensemble (cf. Figure 1(e)). Surprisingly, the best majority-voting ensemble for the test and validation data sets contains 1 RF and 3 GradB models, compared to the 110 models added from all stages in total. This currently active ensemble appears to perform worse in the 5-fold cross-validation results against the current best ML ensemble (Figure 1(h)). Though, this could imply that a larger k -fold value should have been used from the start. However, for comparison purposes, we have chosen the k value of 5 from Mansouri et al. [MRB*13].

Evaluation with the Test and External Validation Sets. To verify whether our findings were reliable, we applied the resulting majority-voting ensemble to the same test and external validation data sets as Mansouri et al. [MRB*13], see Table 1. For the test data

set, the reported accuracy was approximately 87%. In our case, we reached 89% for accuracy with the final voting ensemble (*macro-average*). Additionally, as an extra validation, we checked the results for the additional external data set. Using our approach, we managed to achieve the same accuracy as before, 89%, compared to 83% reported by Mansouri et al. [MRB*13].

Table 1: Summary of the test data and external validation data results for the QSAR biodegradation binary classification problem.

Validation Metric	VisEvol		Mansouri et al. [MRB*13]	
	Test Set	External Validation Set	Test Set	External Validation Set
Accuracy	89%	89%	87%	83%
Precision	89%	87%	92%	91%
Recall	86%	85%	82%	76%
F1-score	87%	86%	87%	83%

6. Evaluation

We conducted three online semi-structured interviews with ML experts to obtain qualitative feedback about our tool’s usefulness, as in prior works [MXLM20, XXM*19]. The first expert (**E1**) is a senior lecturer in mathematics working with reinforcement learning and has approximately 3 years of experience with ML. He recently acquired his PhD in mathematics and has basic knowledge regarding ensemble learning. The second expert (**E2**) is a senior researcher in software engineering and applied ML working in a government research institute and as an adjunct professor. He has worked with ML for the past 7.5 years, and 2.5 years with ensemble learning. The third expert (**E3**) is an ML engineer and manager in a large multinational company, working with recommendation systems. She has approximately 7.5 years of experience with ML, of which 2 years are associated with ensemble learning. The latter two experts have PhDs in computer science; none of our three experts reported colorblindness issues. The followed procedure was: (1) presentation of the key goals of VisEvol, (2) demonstration of the functionality of each view and interaction with the tool using the *heart disease* data set, and (3) explanation of the process of boosting the results in Section 4. Each interview took about one hour. We informed the participants of the main areas we expected feedback from, but they were free to comment on anything.

Workflow. **E1** and **E2** commented that the workflow of VisEvol is well designed. Although **E3** expected a more linear workflow, she agreed that the combined views are better positioned at the top, with the interactive projections in the middle and the shared views at the bottom. **E2** has recently worked with genetic algorithms for testing traffic-scenarios for autonomous vehicles. In that case, they had to set a strict budget before execution and perform multiple crossover and mutation stages which can take days to run. Nevertheless, he noticed that in evolutionary optimization, hundreds of stages might not be necessary since, with three stages, we could gather performant models that are hard to surpass in terms of predictive performance. Finally, **E1** mentioned that controlling the evolutionary process via the Sankey diagram can be time-saving.

Visualization and Interaction. **E1** and **E3** were delighted by the possibilities of the visual exploration of the hyperparameter space. **E1** was enthusiastic about the grid-based view and stated that it is a game-changer for finding performant and diverse models. **E2** was initially confused with the comparison of instances from various

clusters in this same view, but after some training period, he understood that he has to look for different patterns collectively (instead of individual instances). Afterwards, he agreed that, once he gradually explored a cluster, it was easier to gain insights from the comparison with the rest. Both **E1** and **E2** mentioned that even though interactions are mostly bounded in the projection-based views, this keeps the tool easy to interact with and removes additional complexity, which is an excellent practice to follow with people not accustomed to VA tools. “It is great to find that various combinations of models lead to different ensembles that are better for each class of the independent variable, which is visible from the two views [in Figure 1(g) and (h)]”, said **E2**. Although this extra information restricts the generalization of VisEvol in non-binary classification problems, modifying those views should be straightforward.

Limitations. **E1** and **E2** were worried about the *scalability* of the tool. Indeed, the excessive computational time required for producing new hyperparameters along with ensemble learning methods can be problematic. Despite that, one possible improvement for VisEvol is to utilize parallel processing on powerful cloud servers. Moreover, we believe that the advancements in high-performance hardware and progressive VA and data science workflows [SPG14, TPB*18] will be beneficial for VisEvol. The users can also avoid extra computations at certain steps of majority-voting ensemble construction, as discussed in Section 4.4. Another open issue is the *avoidance of hyperparameter tuning per se*, as noted by **E3**. The goal of the tool is not to explore or bring insights about the individual sets of hyperparameters of the models or algorithms, but instead we focus on the search for new powerful models and implicitly store their hyperparameters. The study of the impact of particular hyperparameters is considered as a future direction for VisEvol. Also, **E3** stated that we could allow the user to *specify the hyperparameters range* at every stage and test *alternative mutation strategies* [CK05]. **E1** expressed his interest in *checking combinations of evolutionary optimization* with the crossover and mutation process applied to the best-performing models (e.g., [YRK*15]). However, as the user usually adds—as few as possible—models to the ensembles, the hyperparameters’ evolution for the excluded algorithms will be infeasible. We plan to overcome such limitations.

7. Conclusion

In this paper, we presented VisEvol, a VA tool with the aim to support hyperparameter search through evolutionary optimization. With the utilization of multiple coordinated views, we allow users to generate new hyperparameter sets and store the already robust hyperparameters in a majority-voting ensemble. Exploring the impact of the addition and removal of algorithms and models in a majority-voting ensemble from different perspectives and tracking the crossover and mutation process enables users to be sure how to proceed with the selection of hyperparameters for a single model or complex ensembles that require a combination of the most performant and diverse models. The effectiveness of VisEvol was examined with use cases using real-world data that demonstrated the advancement of the methods behind achieving performance improvement. Our tool’s workflow and visual metaphors received positive feedback from three ML experts, who even identified limitations of VisEvol. These limitations pose future research directions for us.

References

- [ASY*19] AKIBA T., SANO S., YANASE T., OHTA T., KOYAMA M.: Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019), KDD '19, ACM, pp. 2623–2631. doi:10.1145/3292500.3330701. 3
- [Aut] AutoML — Google Cloud AutoML. Accessed February 26, 2021. URL: <https://cloud.google.com/automl/>. 3
- [Bay] BayesOpt — Bayesian optimization for science and engineering. Accessed February 26, 2021. URL: <https://bayesopt.github.io/>. 3
- [BB12] BERGSTRA J., BENGIO Y.: Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13 (Feb. 2012), 281–305. doi:10.5555/2188385.2188395. 2
- [BBBK11] BERGSTRA J., BARDENET R., BENGIO Y., KÉGL B.: Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems* (2011), NIPS '11, Curran Associates Inc., pp. 2546–2554. 2
- [BBKS13] BARDENET R., BRENDÉL M., KÉGL B., SEBAG M.: Collaborative hyperparameter tuning. In *Proceedings of Machine Learning Research* (2013), vol. 28(2), PMLR, pp. 199–207. 3
- [BCA*19] BELKINA A. C., CICCIOLELLA C. O., ANNO R., HALPERT R., SPIDLEN J., SNYDER-CAPPIONE J. E.: Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. *Nature Communications* 10, 5415 (2019). doi:10.1038/s41467-019-13055-y. 3
- [BDA13] BEKKAR M., DJEMAA H. K., ALITOUCHE T. A.: Evaluation measures for models assessment over imbalanced data sets. *Journal of Information Engineering and Applications* 3, 10 (2013). 5
- [BKE*15] BERGSTRA J., KOMER B., ELIASMITH C., YAMINS D., COX D. D.: Hyperopt: A Python library for model selection and hyperparameter optimization. *Computational Science & Discovery* 8, 1 (July 2015), 014008. doi:10.1088/1749-4699/8/1/014008. 3
- [Bre01] BREIMAN L.: Random forests. *Machine Learning* 45 (Oct. 2001), 5–32. doi:10.1023/A:1010933404324. 3
- [CAA*19] CHEN S., ANDRIENKO N., ANDRIENKO G., ADILOVA L., BARLET J., KINDERMANN J., NGUYEN P. H., THONNARD O., TURKAY C.: LDA ensembles for interactive exploration and categorization of behaviors. *IEEE Transactions on Visualization and Computer Graphics* (2019). doi:10.1109/TVCG.2019.2904069. 3
- [CDM15] CLAESEN M., DE MOOR B.: Hyperparameter search in machine learning. In *Proceedings of the 11th Metaheuristics International Conference* (2015), MIC '15. 3
- [CG16] CHEN T., GUESTRIN C.: XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), KDD '16, ACM, pp. 785–794. doi:10.1145/2939672.2939785. 3
- [CGW13] CAI J., GARNER J. L., WALKLING R. A.: A paper tiger? An empirical analysis of majority voting. *Journal of Corporate Finance* 21 (2013), 119–135. doi:10.1016/j.jcorpfin.2013.01.002. 2
- [CJ20] CHICCO D., JURMAN G.: The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics* 21 (Jan. 2020), 6. doi:10.1186/s12864-019-6413-7. 2
- [CK05] CANTU-PAZ E., KAMATH C.: An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 35, 5 (Oct. 2005), 915–927. doi:10.1109/TSMCB.2005.847740. 2, 7, 10
- [CMJ*20] CHATZIMPAMPAS A., MARTINS R. M., JUSUFI I., KUCHER K., ROSSI F., KERREN A.: The state of the art in enhancing trust in machine learning models with the use of visualizations. *Computer Graphics Forum* 39, 3 (June 2020), 713–756. doi:10.1111/cgf.14034. 4
- [CMJK20] CHATZIMPAMPAS A., MARTINS R. M., JUSUFI I., KERREN A.: A survey of surveys on the use of visualization for interpreting machine learning models. *Information Visualization* 19, 3 (July 2020), 207–233. doi:10.1177/1473871620904671. 4
- [CMK20] CHATZIMPAMPAS A., MARTINS R. M., KERREN A.: t-viSNE: Interactive assessment and interpretation of t-SNE projections. *IEEE Transactions on Visualization and Computer Graphics* 26, 8 (Aug. 2020), 2696–2714. doi:10.1109/TVCG.2020.2986996. 4
- [CMKK21] CHATZIMPAMPAS A., MARTINS R. M., KUCHER K., KERREN A.: StackGenVis: Alignment of data, algorithms, and models for stacking ensemble learning using performance metrics. *IEEE Transactions on Visualization and Computer Graphics* (2021). doi:10.1109/TVCG.2020.3030352. 3, 4
- [Com] Comet.ML — Build better models faster. Accessed February 26, 2021. URL: <https://comet.ml/>. 2
- [CSP*14] CLAESEN M., SIMM J., POPOVIC D., MOREAU Y., DE MOOR B.: Easy hyperparameter search using Optunity. *ArXiv e-prints* (Dec. 2014). arXiv:1412.1114. 3
- [D311] D3 — Data-driven documents, 2011. Accessed February 26, 2021. URL: <https://d3js.org/>. 5
- [Dat] DataRobot — Empowering the human heroes of the intelligence revolution. Accessed February 26, 2021. URL: <https://www.datarobot.com/>. 3
- [DCCE19] DAS S., CASHMAN D., CHANG R., ENDERT A.: BEAMES: Interactive multi-model steering, selection, and inspection for regression tasks. *IEEE Computer Graphics and Applications* 39, 9 (Sept. 2019). doi:10.1109/MCG.2019.2922592. 3
- [DDF*18] DI FRANCESCO MARINO C., DUMAS M., FEDERICI M., GHIDINI C., MAGGI F. M., RIZZI W., SIMONETTO L.: Genetic algorithms for hyperparameter optimization in predictive business process monitoring. *Information Systems* 74, Part 1 (May 2018), 67–83. doi:10.1016/j.is.2018.01.003. 2
- [DG06] DAVIS J., GOADRICH M.: The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning* (2006), ICML '06, ACM, pp. 233–240. doi:10.1145/1143844.1143874. 2
- [DG17] DUA D., GRAFF C.: UCI machine learning repository, 2017. Accessed February 26, 2021. URL: <http://archive.ics.uci.edu/ml/>. 5
- [FBO*07] FISZELEW A., BRITOS P., OCHOA A., MERLINO H., FERNÁNDEZ E., GARCÍA-MARTÍNEZ R.: Finding optimal neural network architecture using genetic algorithms. *Research in Computing Science* 27 (2007), 15–24. 3
- [FH19] FEURER M., HUTTER F.: Hyperparameter optimization. In *Automated Machine Learning: Methods, Systems, Challenges*. Springer International Publishing, 2019, pp. 3–33. doi:10.1007/978-3-030-05318-5_1. 2
- [FHOM09] FERRI C., HERNÁNDEZ-ORALLO J., MODROIU R.: An experimental comparison of performance measures for classification. *Pattern Recognition Letters* 30, 1 (Jan. 2009), 27–38. doi:10.1016/j.patrec.2008.08.010. 2
- [FKH18] FALKNER S., KLEIN A., HUTTER F.: BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of Machine Learning Research* (2018), vol. 80, PMLR, pp. 1437–1446. 2, 3
- [Fla10] Flask — A micro web framework written in Python, 2010. Accessed February 26, 2021. URL: <https://flask.palletsprojects.com/>. 5
- [FSA99] FREUND Y., SCHAPIRE R., ABE N.: A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence* 14, 5 (Sept. 1999), 771–780. 3
- [GSM*17] GOLOVIN D., SOLNIK B., MOITRA S., KOCHANSKI G., KARRO J., SCULLEY D.: Google Vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017), KDD '17, ACM, pp. 1487–1495. doi:10.1145/3097983.3098043. 3

- [HDK*19] HAMID S., DERSTROFF A., KLEMM S., NGO Q. Q., JIANG X., LINSSE L.: Visual ensemble analysis to study the influence of hyper-parameters on training deep neural networks. In *Proceedings of the EuroVis Workshop on Machine Learning Methods in Visualisation for Big Data* (2019), MLVis '19, The Eurographics Association. doi:10.2312/mlvis.20191160. 3
- [HHLB11] HUTTER F., HOOS H. H., LEYTON-BROWN K.: Sequential model-based optimization for general algorithm configuration. In *Proceedings of the International Conference on Learning and Intelligent Optimization* (2011), LION '11, Springer Berlin Heidelberg, pp. 507–523. doi:10.1007/978-3-642-25566-3_40. 3
- [HHLB13] HUTTER F., HOOS H. H., LEYTON-BROWN K.: An efficient approach for assessing parameter importance in bayesian optimization. In *Proceedings of the NIPS workshop on Bayesian Optimization in Theory and Practice* (2013), BayesOpt '13. 3
- [HHLB14] HUTTER F., HOOS H., LEYTON-BROWN K.: An efficient approach for assessing hyperparameter importance. In *Proceedings of Machine Learning Research* (2014), vol. 32(1), PMLR, pp. 754–762. 3
- [HHLBS09] HUTTER F., HOOS H. H., LEYTON-BROWN K., STÜTZLE T.: ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research* 36, 1 (Oct. 2009), 267–306. doi:10.1613/jair.2861. 3
- [ID87] INSELBERG A., DIMSDALE B.: Parallel coordinates for visualizing multi-dimensional geometry. In *Proceedings of the 5th International Conference on Computer Graphics* (1987), CG '87, Springer Japan, pp. 25–44. doi:10.1007/978-4-431-68057-4_3. 3
- [JDO*17] JADERBERG M., DALIBARD V., OSINDERO S., CZARNECKI W. M., DONAHUE J., RAZAVI A., VINYALS O., GREEN T., DUNNING I., SIMONYAN K., FERNANDO C., KAVUKCUOGLU K.: Population based training of neural networks. *ArXiv e-prints* (Nov. 2017). arXiv:1711.09846. 2, 3
- [JES*20] JÖNSSON D., EILERTSEN G., SHI H., ZHENG J., YNNERMAN A., UNGER J.: Visual analysis of the impact of neural network hyper-parameters. In *Proceedings of the EGEV International Workshop on Machine Learning Methods in Visualisation for Big Data* (2020), MLVis '20, The Eurographics Association. doi:10.2312/mlvis.20201101. 3
- [JWXY16] JIA D., WANG R., XU C., YU Z.: QIM: Quantifying hyperparameter importance for deep learning. In *Proceedings of the IFIP International Conference on Network and Parallel Computing* (2016), NPC 16, Springer International Publishing, pp. 180–188. doi:10.1007/978-3-319-47099-3_15. 3
- [KB19] KOBAK D., BERENS P.: The art of using t-SNE for single-cell transcriptomics. *Nature Communications* 10, 5416 (Nov. 2019). doi:10.1038/s41467-019-13056-x. 3
- [KE05] KERREN A., EGGER T.: EAVis: A visualization tool for evolutionary algorithms. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing* (2005), VL/HCC '05, IEEE, pp. 299–301. doi:10.1109/VLHCC.2005.33. 3
- [Ker06] KERREN A.: Improving strategy parameters of evolutionary computations with interactive coordinated views. In *Proceedings of the IASTED International Conference on Visualization, Imaging, and Image Processing* (2006), VIIP '06, ACTA Press, pp. 88–93. 3
- [KGG*18] KOCH P., GOLOVIDOV O., GARDNER S., WUJEK B., GRIFIN J., XU Y.: Autotune: A derivative-free optimization framework for hyperparameter tuning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2018), KDD '18, ACM, pp. 443–452. doi:10.1145/3219819.3219837. 3
- [KKP*18] KIM J., KIM M., PARK H., KUSDAVLETOV E., LEE D., KIM A., KIM J.-H., HA J.-W., SUNG N.: CHOPT: Automated hyperparameter optimization framework for cloud-based machine learning platforms. *ArXiv e-prints* (Oct. 2018). arXiv:1810.03527. 3
- [KMF*17] KE G., MENG Q., FINLEY T., WANG T., CHEN W., MA W., YE Q., LIU T.-Y.: LightGBM: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (2017), NIPS '17, Curran Associates Inc., pp. 3149–3157. 3
- [Kru64] KRUSKAL J. B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika* 29, 1 (Mar. 1964), 1–27. doi:10.1007/BF02289565. 6
- [LCW*18] LI T., CONVERTINO G., WANG W., MOST H., ZAJONC T., TSAI Y.: HyperTuner: Visual analytics for hyperparameter tuning by professionals. In *Proceedings of the IEEE VIS Workshop on Machine Learning from User Interaction for Visualization and Analytics* (2018), MLUI '18. 3, 4
- [LH18] LIU Y., HEER J.: Somewhere over the rainbow: An empirical assessment of quantitative colormaps. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (2018), CHI '18, ACM, pp. 598:1–598:12. doi:10.1145/3173574.3174172. 6
- [LJ19] LATHA C. B. C., JEEVA S. C.: Improving the accuracy of prediction of heart disease risk based on ensemble classification techniques. *Informatics in Medicine Unlocked* 16 (2019), 100203. doi:10.1016/j.imu.2019.100203. 5, 8
- [LJD*17] LI L., JAMIESON K., DESALVO G., ROSTAMIZADEH A., TALWALKAR A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research* 18, 1 (Jan. 2017), 6765–6816. 2, 3
- [LJVR08] LOBO J. M., JIMÉNEZ-VALVERDE A., REAL R.: AUC: A misleading measure of the performance of predictive distribution models. *Global Ecology and Biogeography* 17, 2 (Mar. 2008), 145–151. doi:10.1111/j.1466-8238.2007.00358.x. 2
- [LLN*18] LIAW R., LIANG E., NISHIHARA R., MORITZ P., GONZALEZ J. E., STOICA I.: Tune: A research platform for distributed model selection and training. In *Proceedings of the ICML/IJCAI-ECAI International Workshop on Automatic Machine Learning* (2018), AutoML '18. 3
- [LTKS19] LIU J., TRIPATHI S., KURUP U., SHAH M.: Auptimizer — An extensible, open-source framework for hyperparameter tuning. In *Proceedings of the IEEE International Conference on Big Data* (2019), Big Data '19, IEEE, pp. 339–348. doi:10.1109/BigData47090.2019.9006330. 3
- [LXL*18] LIU S., XIAO J., LIU J., WANG X., WU J., ZHU J.: Visual diagnosis of tree boosting methods. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan. 2018), 163–173. doi:10.1109/TVCG.2017.2744378. 3
- [MHM18] MCINNES L., HEALY J., MELVILLE J.: UMAP: Uniform manifold approximation and projection for dimension reduction. *ArXiv e-prints* 1802.03426 (Feb. 2018). arXiv:1802.03426. 6
- [MRB*13] MANSOURI K., RINGSTED T., BALLABIO D., TODESCHINI R., CONSONNI V.: Quantitative structure–activity relationship models for ready biodegradability of chemicals. *Journal of Chemical Information and Modeling* 53, 4 (2013), 867–878. doi:10.1021/ci4000213. 8, 9, 10
- [MRK06] MCNEE S. M., RIEDL J., KONSTAN J. A.: Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems* (2006), CHI EA '06, ACM, pp. 1097–1101. doi:10.1145/1125451.1125659. 2
- [MXLM20] MA Y., XIE T., LI J., MACIEJEWSKI R.: Explaining vulnerabilities to adversarial machine learning through visual analytics. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (Jan. 2020), 1075–1085. doi:10.1109/TVCG.2019.2934631. 10
- [NNI] NNI — Microsoft Neural Network Intelligence. Accessed February 26, 2021. URL: <https://github.com/microsoft/nni>. 2
- [NP20] NETO M. P., PAULOVIČ F. V.: Explainable Matrix — Visualization for global and local interpretability of random forest classification ensembles. *IEEE Transactions on Visualization and Computer Graphics* (2020). doi:10.1109/TVCG.2020.3030354. 3

- [PBB19] PROBST P., BOULESTEIX A.-L., BISCHL B.: Tunability: Importance of hyperparameters of machine learning algorithms. *Journal of Machine Learning Research* 20, 53 (2019), 1–32. 3
- [PKK*19] PARK H., KIM J., KIM M., KIM J., CHOO J., HA J., SUNG N.: VisualHyperTuner: Visual analytics for user-driven hyperparameter tuning of deep neural networks. In *Proceedings of the 2nd SysML Conference* (2019), SysML 19. 3
- [plo10] Plotly — JavaScript open source graphing library, 2010. Accessed February 26, 2021. URL: <https://plotly.com>. 5
- [PN17] PEREIRA L., NUNES N.: A comparison of performance metrics for event classification in non-intrusive load monitoring. In *Proceedings of the IEEE International Conference on Smart Grid Communications* (2017), SmartGridComm '17, IEEE, pp. 159–164. doi: 10.1109/SmartGridComm.2017.8340682. 2
- [PNKC21] PARK H., NAM Y., KIM J., CHOO J.: HyperTendril: Visual analytics for user-driven hyperparameter optimization of deep neural networks. *IEEE Transactions on Visualization & Computer Graphics* (2021). doi:10.1109/TVCG.2020.3030380. 3, 4
- [Pow11] POWERS D. M. W.: Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies* 2, 1 (2011), 37–63. 2
- [PVG*11] PEDREGOSA F., VAROQUAUX G., GRAMFORT A., MICHEL V., THIRION B., GRISEL O., BLONDEL M., PRETTEHOFER P., WEISS R., DUBOURG V., VANDERPLAS J., PASSOS A., COURNAPEAU D., BRUCHER M., PERROT M., DUCHESNAY E.: Scikit-Learn: Machine learning in Python. *Journal of Machine Learning Research* 12 (Nov. 2011), 2825–2830. doi:10.5555/1953048.2078195. 5
- [SDC*17] SWEARINGEN T., DREVO W., CYPHERS B., CUESTA-INFANTE A., ROSS A., VEERAMACHANENI K.: ATM: A distributed, collaborative, scalable system for automated machine learning. In *Proceedings of the IEEE International Conference on Big Data* (2017), Big Data '17, IEEE, pp. 151–162. doi:10.1109/BigData.2017.8257923. 3
- [SJS*18] SCHNEIDER B., JÄCKLE D., STOFFEL F., DIEHL A., FUCHS J., KEIM D. A.: Integrating data and model space in ensemble learning by visual analytics. *IEEE Transactions on Big Data* (2018). doi: 10.1109/TBDDATA.2018.2877350. 3
- [SKJ*17] SUNG N., KIM M., JO H., YANG Y., KIM J., LAUSEN L., KIM Y., LEE G., KWAK D., HA J.-W., KIM S.: NSML: A machine learning platform that enables you to focus on your models. In *Proceedings of the NIPS Workshop on ML Systems* (2017), ML-Sys '17. 3
- [SL09] SOKOLOVA M., LAPALME G.: A systematic analysis of performance measures for classification tasks. *Information Processing & Management* 45, 4 (July 2009), 427–437. doi:10.1016/j.ipm.2009.03.002. 2
- [SLA12] SNOEK J., LAROCHELLE H., ADAMS R. P.: Practical Bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems* (2012), vol. 2 of *NIPS '12*, Curran Associates Inc., pp. 2951–2959. 2
- [SPG14] STOLPER C. D., PERER A., GOTZ D.: Progressive visual analytics: User-driven visual exploration of in-progress analytics. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (Dec. 2014), 1653–1662. doi:10.1109/TVCG.2014.2346574. 10
- [SR15] SAITO T., REHMSMEIER M.: The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE* 10, 3 (Mar. 2015), e0118432. doi: 10.1371/journal.pone.0118432. 2
- [SR18] SAGI O., ROKACH L.: Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery* 8, 4 (July–Aug. 2018), e1249. doi:10.1002/widm.1249. 2
- [SSW*16] SHAHRIARI B., SWERSKY K., WANG Z., ADAMS R. P., DE FREITAS N.: Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE* 104, 1 (2016), 148–175. doi: 10.1109/JPROC.2015.2494218. 2, 3
- [Stu13] STURM B. L.: Classification accuracy is not enough. *Journal of Intelligent Information Systems* 41, 3 (Dec. 2013), 371–406. doi: 10.1007/s10844-013-0250-y. 2
- [Tak01] TAKAGI H.: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE* 89, 9 (Sept. 2001), 1275–1296. doi:10.1109/5.949485. 3
- [TBCT*18] TSIRIGOTIS C., BOUTHILLIER X., CORNEAU-TREMBLAY F., HENDERSON P., ASKARI R., LAVOIE-MARCHILDON S., DELEU T., SUHUBDY D., NOUKHOVITCH M., BASTIEN F., LAMBLIN P.: Orion: Experiment version control for efficient hyperparameter optimization. In *Proceedings of the ICML Workshop on Reproducibility in Machine Learning* (2018), RML '18. 3
- [Tha18] THARWAT A.: Classification assessment methods. *Applied Computing and Informatics* (2018). doi:10.1016/j.aci.2018.08.003. 2
- [THHLB13] THORNTON C., HUTTER F., HOOS H. H., LEYTON-BROWN K.: Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2013), KDD '13, ACM, pp. 847–855. doi:10.1145/2487575.2487629. 3
- [TLKT09] TALBOT J., LEE B., KAPOOR A., TAN D. S.: EnsembleMatrix: Interactive visualization to support machine learning with multiple classifiers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2009), CHI '09, ACM, pp. 1283–1292. doi:10.1145/1518701.1518895. 3
- [TMB*18] TSAY J., MUMMERT T., BOBROFF N., BRAZ A., WESTERINK P., HIRZEL M., HEIGHTS Y.: Runway: Machine learning model experiment management tool. In *Proceedings of the 1st SysML Conference* (2018), SysML '18. 3
- [TPB*18] TURKAY C., PEZZOTTI N., BINNIG C., STROBELT H., HAMMER B., KEIM D. A., FEKETE J., PALPANAS T., WANG Y., RUSU F.: Progressive data science: Potential and challenges. *CoRR abs/1812.08032* (2018). arXiv:1812.08032. 10
- [vdMH08] VAN DER MAATEN L., HINTON G.: Visualizing data using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605. 6
- [vRH17] VAN RIJN J. N., HUTTER F.: An empirical study of hyperparameter importance across datasets. In *Proceedings of the ECML-PKDD International Workshop on Automatic Machine Learning* (2017), AutoML '17. 2, 3
- [vRH18] VAN RIJN J. N., HUTTER F.: Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2018), KDD '18, ACM, pp. 2367–2376. doi:10.1145/3219819.3220058. 2, 3
- [vue14] Vue.js — The progressive JavaScript framework, 2014. Accessed February 26, 2021. URL: <https://vuejs.org/>. 5
- [WMJ*19] WANG Q., MING Y., JIN Z., SHEN Q., LIU D., SMITH M. J., VEERAMACHANENI K., QU H.: ATMSeer: Increasing transparency and controllability in automated machine learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (2019), CHI '19, ACM, pp. 681:1–681:12. doi:10.1145/3290605.3300911. 3, 4
- [Wol92] WOLPERT D. H.: Stacked generalization. *Neural Networks* 5, 2 (1992), 241–259. doi:10.1016/S0893-6080(05)80023-1. 3
- [WRW*20] WANG D., RAM P., WEIDLE D. K. I., LIU S., MULLER M., WEISZ J. D., VALENTE A., CHAUDHARY A., TORRES D., SAMULOWITZ H., AMINI L.: AutoAI: Automating the end-to-end ai lifecycle with humans-in-the-loop. In *Proceedings of the 25th International Conference on Intelligent User Interfaces Companion* (2020), IUI '20, ACM, pp. 77–78. doi:10.1145/3379336.3381474. 3
- [WWO*20] WEIDLE D. K. I., WEISZ J. D., ODUOR E., MULLER M., ANDRES J., GRAY A., WANG D.: AutoAIViz: Opening the black-box of automated artificial intelligence with conditional parallel coordinates. In *Proceedings of the 25th International Conference on In-*

- telligent User Interfaces* (2020), IUI '20, ACM, pp. 308–312. doi:
[10.1145/3377325.3377538](https://doi.org/10.1145/3377325.3377538). 3
- [XXM*19] XU K., XIA M., MU X., WANG Y., CAO N.: EnsembleLens: Ensemble-based visual exploration of anomaly detection algorithms with multidimensional data. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (Jan. 2019), 109–119. doi:
[10.1109/TVCG.2018.2864825](https://doi.org/10.1109/TVCG.2018.2864825). 3, 10
- [YM14] YOGATAMA D., MANN G.: Efficient transfer learning method for automatic hyperparameter tuning. In *Proceedings of Machine Learning Research* (Apr. 2014), vol. 33, PMLR, pp. 1077–1085. 3
- [YRK*15] YOUNG S. R., ROSE D. C., KARNOWSKI T. P., LIM S.-H., PATTON R. M.: Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments* (2015), MLHPC '15, ACM. doi:[10.1145/2834892.2834896](https://doi.org/10.1145/2834892.2834896). 2, 3, 10
- [ZWLC19] ZHAO X., WU Y., LEE D. L., CUI W.: iForest: Interpreting random forests via visual analytics. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (Jan. 2019), 407–416. doi:[10.1109/TVCG.2018.2864475](https://doi.org/10.1109/TVCG.2018.2864475). 3
- [ZWM*19] ZHANG J., WANG Y., MOLINO P., LI L., EBERT D. S.: Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (Jan. 2019), 364–373. doi:[10.1109/TVCG.2018.2864499](https://doi.org/10.1109/TVCG.2018.2864499). 3