OCEG: An interactive visualization system for classifier ensemble generation

Mauro Diaz
Universidad Católica San Pablo
Arequipa, Perú
dany.diaz@ucsp.edu.pe

Guillermo Camara-Chavez
Universidade Federal de Ouro Preto
Ouro Preto, Brazil
guillermo@ufop.edu.br

Erick Gomez-Nieto
Universidad Católica San Pablo
Arequipa, Perú
emgomez@ucsp.edu.pe

***Abstract*—Classification is a fundamental task in machine learning. However, individual models can struggle to deliver accurate results in various scenarios. Ensemble methods, which combine multiple classifiers, can enhance performance, but selecting and fine-tuning these models can be time-consuming. We introduce OCEG, a visual system designed to support the interactive exploration and composition of classifier ensembles. OCEG features a web-based interface that utilizes visual cues and pruning algorithms to highlight differences between models, automatically optimize ensemble parameters, and compare alternatives from a comprehensive perspective. We validate our system through quantitative experiments, a real-world case study, and user evaluations.**

I. Introduction

Classification models are widely used in predictive analysis and applied across various data types, *e.g.* , text [1], images [2], video [3], among others. However, a single classifier often falls short in accuracy, prompting data scientists to adopt ensemble methods, which are groups of classifiers working together, that tend to outperform individual models [4]. Still, building effective ensembles involves time-consuming trial and error to choose and combine classifiers effectively.

Meanwhile, interactive data visualization has emerged as a valuable tool for exploring patterns and relationships while enabling analysts to apply their domain knowledge. Different approaches in visualization for Explainable Artificial Intelligence (XAI) rely on multiple strategies for understanding results produced by automatic methods, such as counterfactual analysis [5]–[7] and model-agnostic methods [8], [9]. Nonetheless, they lack flexibility in creating weighted classifier ensembles or comparing multiple ensembles simultaneously.

To address these gaps, we propose OCEG, an interactive visualization system for generating classifier ensembles. Our approach blends visual mechanisms and automatic algorithms to help users build and export optimized ensembles more efficiently.

In summary, the main contributions of this paper are:

- A methodology that integrates visual and algorithmic tools to support the creation of classifier ensembles.
- An interactive system, named OCEG, that enables quick ensemble generation and export.
- A comprehensive validation including quantitative analysis, a case study, and a user evaluation.

II. Related Work

To properly frame our work, we describe the following state-of-the-art approaches, organized into two categories: *automatic approaches*, which utilize statistics for computing ensembles, and *interactive approaches*, which incorporate visual components as core elements. Our focus is primarily on the second group.

Automatic approaches: DEL [10] automatically configures neural network ensembles by optimizing their number and size. Moyano and Ventura [11] use genetic programming to construct adaptive, tree-shaped multi-label ensembles with minimal tuning. These methods typically lack analyst feedback unless parameters are manually tracked and reused. For a broader review, see [12].

Interactive approaches: Ribeiro *et. al* [13] combined visual tools for interpreting ensemble components and inspecting instances. Boxer [14] offers various coordinated views to compare classifier results, emphasizing ease of use. Other efforts focus on confusion matrix analysis, such as ConfusionFlow [15] and ConfusionVis [16]. StackGenVis [17] helps users construct stacked ensembles using base models and a metamodel. Although these studies address objectives similar to our proposal, they still have shortcomings. Many need to provide more flexibility to create ensembles with different weights for each member, or require a way to compare them simultaneously. In this work, we address these limitations for the design of our proposal.

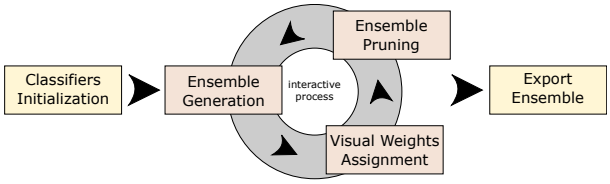

Fig. 1. Our five-step methodology contains an interactive process combining automatic and visual components to provide an ensemble that fulfills the analyst's requirements.

III. The OCEG system

OCEG is an interactive visualization system designed to assist in generating, comparing, and optimizing classifier ensembles. Following, we describe the methodology, visual components, and implementation details.

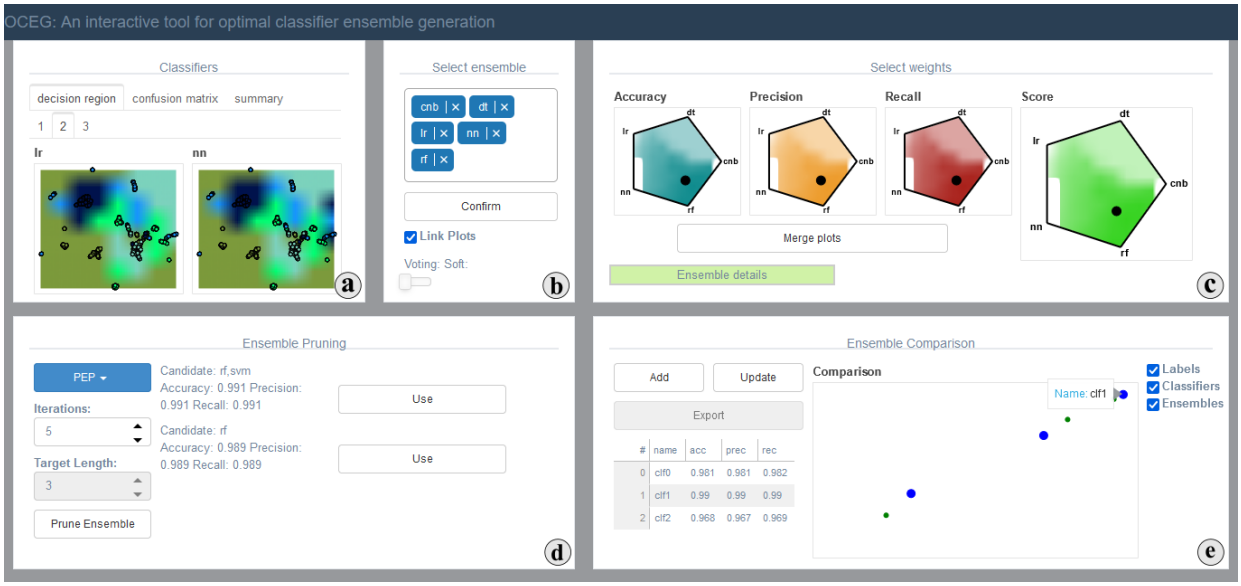

Fig. 2. Overview of the OCEG tool: (a) Classifier Members view, (b) Ensemble Generator view, (c) Visual Weights Assignment view, (d) Pruning Setup view, and (e) Global Ensemble Comparison view.

A. Methodology

Fig. 1 outlines the five primary steps in the OCEG methodology: (i) **Analysis of potential ensemble members**: this step starts by loading pre-trained classification models and preprocessed test/validation sets. Data is then projected into 2D space using a dimensionality reduction (DR) technique, strictly for visualization purposes; all subsequent analyses utilize the complete set of attributes. (ii) **Ensemble creation**: different ensemble configurations are generated using an interactive tool that provides preliminary results to expedite the process. (iii) **Ensemble pruning**: this involves the application of existing algorithms to refine the ensemble selection. (iv) **Visual comparison of ensembles**: ensembles are evaluated by mapping a 3D vector—comprising accuracy, precision, and recall metrics—into 2D space via a DR method. This visualization facilitates the easy distinction between the most and least effective ensembles. (v) **Selection and export of ensembles**: the analyst can select the most promising ensembles for export and further use.

B. Visual Components

OCEG is an interactive tool with fully-linked views. Fig. 2 shows our system's five main components, described as follows.

Classifier Members view (ⓐ): This view lets analysts upload pre-trained classification models and quickly inspect their characteristics via three plots: decision boundary, confusion matrix, and summary of metrics. For decision boundaries, we generate a grid of synthetic points in 2D, inverse-map them to the original space using UMAP [18], classify them, and color them based on the predicted labels. We then interpolate these values to form a continuous texture and overlay the original points. For confusion matrices, we follow the standard visualization approach. Additionally, the summary view displays key metrics (accuracy, precision, and recall) in a tabular format, which can be sorted by clicking any column name.

Ensemble Generator view (ⓑ): This view allows the analyst to select which classifiers will be used in the resulting ensemble. At least two classifiers must be selected in this view. It also provides the option to use either *Soft Voting* or *Hard Voting* as the consensus function. A useful review on classifier combination can be reached at [19]. Finally, the *Link Plots* checkbox links the weight-assigning plots in the subsequent view, synchronizing their interactions.

Visual Weights Assignment view (ⓒ): Building on [20], this view shows four weight-assignment polygons—three for individual metrics (*accuracy*, *precision*, *recall*) and one summarizing their normalized sum. Each vertex represents a classifier in the current ensemble (Fig. 3, left). The cursor's proximity to vertices defines classifier weights: closer means higher weight. The polygon area is discretized into points (Fig. 3, middle) and color-filled to highlight regions of interest, with darker hues indicating higher scores (Fig. 3, right). Initially, all cursors are linked—moving one moves all—but can be unlinked for independent adjustment by deselecting the *Link Plots* checkbox in the *Ensemble Generator view*. Clicking the "Merge Plots" button searches cursor neighborhoods for the best normalized sum score, selecting the optimal point. The resulting ensemble can then be added to the *Global Ensemble Comparison view* or *Pruning Setup view* for further refinement.

Pruning Setup view (ⓓ): Pruning reduces ensemble size to improve processing speed. Our tool supports two algorithms: Pareto Ensemble Pruning (PEP) [21] and Centralized Objection Maximization for Ensemble Pruning (COMEP) [22], allowing analysts to choose the one that best suits their specific needs. PEP requires setting the number of iterations, while COMEP requires the target ensemble size. After pruning, a detailed list of candidates is presented, along with accuracy,

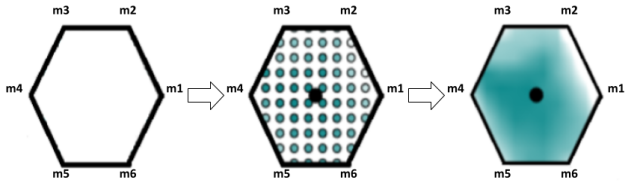

Fig. 3. Generation of metric-based polygons. (Left) We created a polygon where each corner represents a member of the ensemble. (Middle) We discretize the inner space and compute the score for each internal point. Each point represents a weight configuration for classifier models based on the distance to its corner. As it approaches its corner, it will have more weight, analogous to a spring-based scheme. (Right) Scores are smoothed into a texture, with a selector cursor initially centered to indicate equal weights.

precision, and recall metrics. These candidates can be further explored in the Visual Weights Assignment view to adjust weight distributions.

Global Ensemble Comparison view (ⓔ): This view visualizes classifier and ensemble performance in a 2D scatter plot. Each is represented by a 3D vector of accuracy, precision, and recall, then mapped to 2D using LAMP [23]. Two artificial vectors, representing the known best and worst metric values, serve as control points, helping to orient the layout. Points near the "Best" vector indicate better performance. Blue points represent generated ensembles, green points correspond to individual classifiers, and red points denote the control points. LAMP is chosen over UMAP (used in the Classifier Members view) for its ability to position control points, enabling users to customize the projection by adjusting these references.

C. Implementation Details

We implemented the backend in Python and built the user interface using Bokeh. Classifiers were created with Scikit-learn, but since its VotingClassifier requires retraining, we used Mlxtend for ensemble support with pre-trained models. The system currently supports only Sklearn models, as extending to heterogeneous models would require custom handling for output alignment and voting. For dimensionality reduction, we used UMAP* in the Classifier Members view and LAMP† in the Global Comparison view. For pruning, we used the public COMEP‡ implementation and implemented PEP from scratch.

IV. Results

We conducted two distinct types of assessments using OCEG across 16 datasets to evaluate both the benefits and limitations of our proposal. First, we compared its quantitative performance by measuring various metrics across 15 datasets (D3 to D17), aiming to demonstrate the advantages of using ensembles. Second, we examined OCEG's analytical capabilities by exploring a real-world dataset (DS1), focusing on the convenience of generating ensembles through our tool.

*https://github.com/lmcinnes/umap
†https://github.com/lgnonato/LAMP
‡https://github.com/eustomaqua/EPFD

A. Quantitative Evaluation

We performed our quantitative assessment on 15 datasets (DS3 to DS17) collected from UC Irvine Machine Learning Repository to assess the efficiency and utility of our tool. We followed Qian *et. al* [21] configuration, splitting each dataset into equal thirds for training, testing, and validation (for pruning). For our purposes, we trained six different classification models for each dataset to be used as ensemble members, *i.e.*, Decision Tree (**dt**), Random Forest (**rf**), Logistic Regression (**lr**), Neural Network (**nn**), Support Vector Machine (**svm**), and Complement Naive Bayes (**cnb**).

The workflow begins with inspecting individual classifiers using the *Classifier Members view*. Based on these insights, we explored different combinations of classifiers and weights through the *Ensemble Generator* and *Visual Weights Assignment views*. Promising ensembles were added to the *Global Ensemble Comparison view* and optimized using the *Pruning Setup view*. If pruning produced ensembles with comparable or improved scores, we fed them back into the *Ensemble Generator view*. This pruning loop was repeated until it no longer yielded better results. Finally, the optimized ensembles were added to the *Global Ensemble Comparison view*, and their accuracy, precision, and recall scores were recorded for qualitative assessment.

Table I summarizes our experiments. Each column corresponds to a dataset, and each row (at the leftmost level) corresponds to a classification model. We report accuracy (*acc*), precision (*prec*), and recall (*rec*) to assess output quality. The penultimate row shows a baseline ensemble using all classifiers with equal weights, included to illustrate that untuned ensembles can sometimes outperform individual models. At the bottom, we detail the best ensemble generated by OCEG, including its members, weights, and pruning method. Top scores for each dataset are highlighted in gray for clarity.

The performance evaluations of the ensemble classifiers reveal that OCEG consistently surpassed other methods across all three metrics. However, it's essential to note that constructing basic ensembles using only two components (*Classifier Members view* and *Ensemble Generator view*), without adjustments such as weight tuning and pruning, occasionally resulted in less effective models, particularly in the case of DS6.

B. Case Study

Our case study predicts whether a person will be diagnosed with COVID-19 based on their symptoms. The dataset, available on Kaggle §, contains publicly released information from the Mexican government. We performed a preprocessing step that involved removing instances with missing values in any attribute. We then applied the same dataset splitting strategy used in the quantitative evaluation.

To conduct our experiments, we first trained the individual classifiers. We used Decision Tree (*dt*), Random Forest (*rf*), Logistic Regression (*lr*), Neural Network (*nn*), Bernoulli Naive

§https://www.kaggle.com/datasets/tanmoyx/covid19-patient-precondition-dataset

TABLE I

RESULTS FROM SIX CLASSIFICATION METHODS AND THE BEST CLASSIFIER ENSEMBLE GENERATED BY OCEG IN 15 DATASETS (DS3-DS17).

ID		DS3	DS4	DS5	DS6	DS7	DS8	DS9	DS10	DS11	DS12	DS13	DS14	DS15	DS16	DS17
Name		*letter recognition*	*magic*	*breast cancer*	*heart* statlog	*ionosphere*	*kr vs kp*	*opt digits*	*landsat statlog*	*tic tac toe*	*dry bean*	*spambase*	*phishing*	*segment statlog*	*wireless*	*australian statlog*
dt	*acc*	95.50%	77.40%	92.40%	71.90%	86.20%	98.00%	84.90%	83.50%	87.70%	88.50%	86.40%	94.10%	94.90%	97.10%	77.20%
	prec	95.50%	77.40%	93.30%	73.20%	88.40%	98.10%	85.20%	79.60%	86.30%	90.10%	86.50%	94.00%	94.80%	97.00%	76.90%
	rec	95.50%	77.40%	90.70%	71.10%	84.20%	98.00%	85.20%	79.90%	87.30%	90.10%	86.50%	94.10%	94.90%	97.10%	77.20%
rf	*acc*	98.90%	85.20%	96.40%	79.80%	94.80%	97.60%	97.60%	89.60%	89.90%	**91.60%**	94.50%	95.70%	96.50%	97.70%	85.10%
	prec	98.90%	85.20%	96.20%	82.30%	95.90%	97.70%	97.60%	**88.60%**	89.10%	93.00%	94.50%	95.70%	96.50%	97.70%	84.90%
	rec	98.90%	85.20%	96.20%	79.00%	93.90%	97.60%	97.60%	84.80%	88.80%	**92.60%**	94.50%	95.60%	96.50%	97.70%	84.90%
lr	*acc*	94.10%	78.40%	94.70%	**87.60%**	82.80%	95.10%	96.20%	80.90%	69.60%	69.00%	92.80%	93.00%	89.80%	96.70%	86.80%
	prec	94.40%	78.70%	94.30%	**90.50%**	87.20%	95.30%	96.20%	77.60%	66.60%	71.20%	92.80%	92.90%	90.00%	96.50%	86.60%
	rec	93.90%	78.30%	94.30%	**86.90%**	79.90%	94.90%	96.20%	73.70%	63.80%	70.10%	92.80%	92.80%	89.80%	96.60%	87.00%
nn	*acc*	96.40%	82.40%	94.20%	78.70%	83.60%	98.80%	97.10%	70.80%	70.90%	35.70%	93.40%	94.80%	92.50%	97.00%	84.60%
	prec	96.50%	82.50%	94.40%	78.70%	89.00%	98.80%	97.30%	69.90%	68.10%	28.50%	93.30%	94.90%	92.40%	96.90%	84.40%
	rec	96.40%	82.40%	93.30%	78.40%	80.60%	98.80%	97.20%	65.70%	66.00%	28.00%	93.30%	94.60%	92.50%	96.90%	84.60%
svm	*acc*	98.40%	83.10%	95.10%	52.80%	92.20%	94.60%	98.70%	88.30%	74.40%	63.00%	69.60%	94.20%	93.20%	97.90%	86.80%
	prec	98.50%	83.20%	94.70%	26.40%	93.40%	94.90%	98.70%	86.60%	72.40%	55.70%	69.80%	94.30%	93.30%	97.80%	86.60%
	rec	98.40%	83.00%	94.90%	50.00%	91.10%	94.40%	98.70%	82.60%	69.80%	62.20%	69.70%	94.00%	93.00%	97.80%	87.00%
cnb	*acc*	69.80%	77.90%	88.40%	74.20%	75.90%	87.90%	81.60%	62.60%	52.50%	29.50%	76.80%	86.20%	48.00%	65.60%	76.30%
	prec	75.60%	78.30%	87.50%	74.10%	77.70%	87.80%	85.00%	44.30%	52.10%	7.10%	76.80%	86.00%	49.80%	54.90%	77.10%
	rec	70.10%	77.70%	88.40%	74.00%	73.10%	87.90%	82.10%	54.50%	52.30%	28.60%	76.80%	86.00%	49.20%	63.70%	77.30%
baseline	*acc*	99.10%	84.00%	95.60%	84.30%	89.70%	97.80%	98.20%	88.00%	84.80%	85.20%	93.40%	95.00%	95.80%	98.20%	85.10%
	prec	99.20%	84.10%	95.50%	86.30%	92.40%	97.90%	98.20%	85.90%	83.70%	90.30%	93.40%	95.00%	95.80%	98.10%	84.90%
	rec	99.10%	84.00%	95.00%	83.60%	87.80%	97.70%	98.10%	82.30%	82.90%	86.20%	93.40%	94.90%	95.80%	98.10%	85.40%
***Ensemble* OCEG**	*acc*	**99.30%**	**85.40%**	**97.30%**	86.50%	**96.60%**	**99.10%**	**99.00%**	**90.20%**	**91.10%**	91.50%	**94.80%**	**96.00%**	**96.90%**	**98.60%**	**87.70%**
	prec	**99.30%**	**85.40%**	**97.20%**	88.80%	**96.80%**	**99.10%**	**99.00%**	88.40%	**90.50%**	**93.10%**	**94.80%**	**96.00%**	**96.90%**	**98.60%**	**87.50%**
	rec	**99.30%**	**85.30%**	**97.20%**	85.80%	**96.20%**	**99.00%**	**99.00%**	**85.90%**	**90.20%**	92.40%	**94.80%**	**95.90%**	**96.90%**	**98.60%**	**87.80%**
	details	*Members:* dt, rf, lr,nn, svm, cnb *Weights:* 0.3, 0.17, 0.09,0.08, 0.12, 0.25 *Pruning:* *None*	*Members:* rf, svm *Weights:* 0.9, 0.1 *Pruning:* PEP	*Members:* dt, rf, lr, nn, svm, cnb *Weights:* 0.15, 0.06, 0.05, 0.09, 0.28, 0.38 *Pruning:* *None*	*Members:* cnb, dt, lr, nn, rf, svm *Weights:* 0.0, 0.04, 0.75, 0.2, 0.01, 0.0 *Pruning:* *None*	*Members:* cnb, svm *Weights:* 0.7, 0.3 *Pruning:* PEP	*Members:* dt, nn, rf *Weights:* 0.11, 0.83, 0.06 *Pruning:* PEP	*Members:* dt, nn, svm *Weights:* 0.11, 0.72, 0.17 *Pruning:* COMEP	*Members:* dt, nn, rf *Weights:* 0.11, 0.06, 0.83 *Pruning:* PEP	*Members:* dt, lr, rf *Weights:* 0.11, 0.06, 0.83 *Pruning:* COMEP	*Members:* cnb, dt, lr, nn, rf, svm *Weights:* 0.13, 0.02, 0.01, 0.02, 0.18, 0.63 *Pruning:* *None*	*Members:* dt, lr, nn, rf *Weights:* 0.05, 0.01, 0.16, 0.79 *Pruning:* PEP	*Members:* dt, nn, rf *Weights:* 0.22, 0.11, 0.67 *Pruning:* PEP	*Members:* cnb, dt, rf *Weights:* 0.33, 0.06, 0.61 *Pruning:* COMEP	*Members:* cnb, dt, lr, nn, rf, svm *Weights:* 0.3, 0.17, 0.09, 0.08, 0.12, 0.25 *Pruning:* *None*	*Members:* cnb, dt, lr, nn, rf, svm *Weights:* 0.08, 0.02 0.02, 0.05, 0.34, 0.49 *Pruning:* *None*

Bayes (*bnb*), and Complement Naive Bayes (*cnb*). After training, all models and test sets were merged into a single .zip file, which is then loaded into the system to initiate ensemble generation.

We summarize the whole procedure in Fig. 4. First, we start the analysis by uploading the zip file containing all the classifiers to the tool. Then, we had a quick overview of every classifier by looking at the *Classifier Members view*. This facilitates the generation of standard metrics that analysts commonly use.

After conducting an initial exploration, we determined that the best classifiers for the DS1 dataset were Logistic Regression and Random Forest. However, instead of solely relying on the best classifiers, we decided to use all available classifiers to generate our first ensemble. This approach would prevent us from overlooking a potentially superior ensemble. To accomplish this, we selected all individual classifiers in the Ensemble Generator view and opted for Soft Voting as the voting method.

This first ensemble outperformed every individual classifier, scoring 0.744 on all metrics. In contrast, Logistic Regression and Random Forest achieved the highest individual performances, scoring 0.719 and 0.72 respectively, for accuracy, precision, and recall. As a result, the point representing the first ensemble (which we named Clf0) within the *Ensemble Comparison* view is closer to the artificial "Best point" than the points associated with each classifier, representing its superior performance. We obtained a feasible solution, but our intuition is to keep searching for a better result.

After obtaining our first ensemble, we used the Weights Selection view to determine the optimal weights for each classifier within the ensemble. We inspected different weight distributions by moving the cursor inside the polygons. To perform a better exploration, we spread the different cursors into the opaque areas inside the polygons and clicked on 'Merge plots' to find the best distribution. The new ensemble resulting from this weight distribution (called Clf1) obtained 0.746 for accuracy, precision, and recall, reaching a minimal improvement over Clf0 (which assigned the same weight to all individual classifiers). The same procedure was repeated, but Hard Voting was used instead to generate ensemble Clf2. It yielded lower results than those produced by Clf0 and Clf1, so we decided to continue using Soft Voting instead.

The next step was to generate a faster-working ensemble by pruning the number of classifiers without significantly compromising performance. We used the Ensemble Pruning view to find candidates that could replace Clf1 to accomplish this task. The first method used was PEP. Our implementation of this algorithm requires a user-defined parameter: the Number of iterations. We set the number of iterations to three, as more iterations excessively reduce the number of classifiers, harming ensemble performance, while fewer iterations limit exploration. The algorithm selected a combination of five individual classifiers (Complement Naive Bayes, Decision Tree, Logistic Regression, Neural Network, and Random Forest) for our new ensemble (Clf3). Clf3 achieved an accuracy, precision,

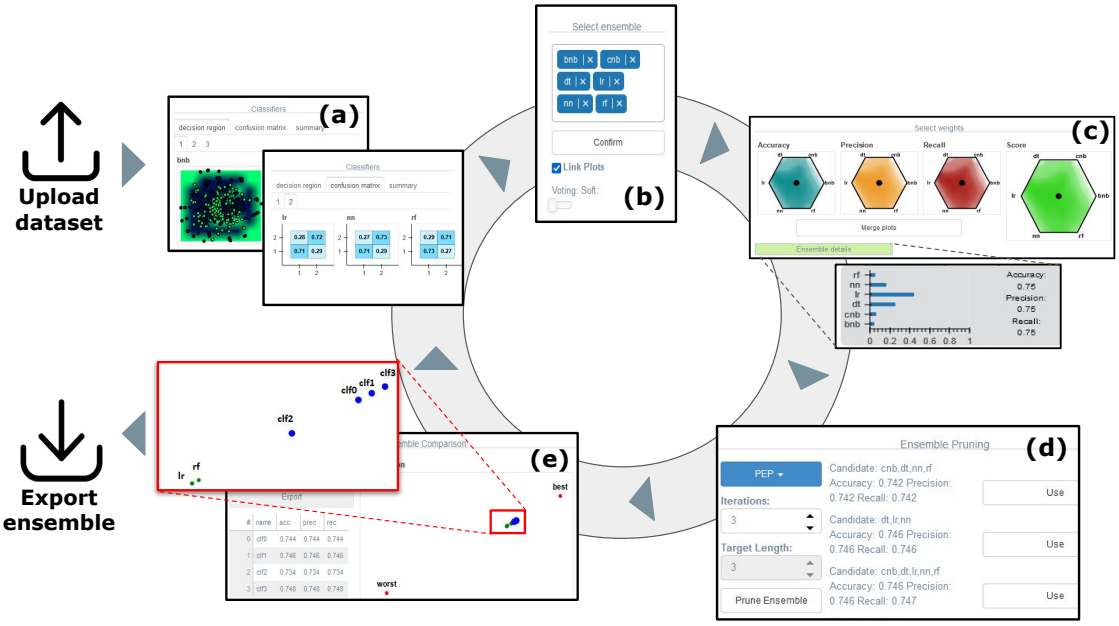

Fig. 4. Summarizing interactions in case study: (a) we load and inspect our classifier models, (b) build our first classifier ensemble, (c-e) weights assignment and optimization process for the built ensemble. Finally, we export the best classifier ensemble.

and recall of 0.747, slightly surpassing the results of Clf1, despite having one fewer classifier, which makes it faster.

We also used the COMEP algorithm to find candidates that could replace Clf1. This algorithm requires a user-defined parameter called Target Length (*TL*), which specifies the number of classifiers to include in the new ensemble. In this case, the *TL* parameter was set to three. The algorithm selected the classifiers Complement Naive Bayes, Decision Tree, and Logistic Regression for our new ensemble, which we named Clf4. Clf4 achieved an accuracy, precision, and recall of 0.737. After creating Clf4, we attempted to optimize it using the *Weights Selection view* to produce an optimal distribution of weights. The new ensemble (Clf5) achieved an accuracy, precision, and recall score of 0.744, outperforming Clf4 but performing worse than Clf1. Finally, we utilized the *Weights Selection view* to optimize Clf3, which was created using PEP. This final ensemble (Clf6) scored 0.748 for the same metrics. We concluded that our generated ensembles Clf3, Clf5 and Clf6 were faster than our initial Clf0 (built with fewer classifiers) while maintaining similar results (Clf3 and Clf6 slightly outperformed Clf0).

V. User Evaluation

We conducted a controlled user evaluation to assess whether OCEG enables the generation of useful classifier ensembles in a reasonable time. This section details our evaluation procedure, results, and user feedback.

TABLE II
Spent time (in minutes), accuracy (0-100%), and the number of ensembles generated by each participant.

	U1	U2	U3	U4	U5	U6	U7	Avg.
time	8:22	8:14	3:53	4:22	2:12	14:38	6:04	6:49
accuracy	94.80%	96.60%	95.70%	98.60%	97.40%	97.40%	96.60%	96.73%
ensembles	3	6	4	2	2	7	5	4

A. Procedure

We invited seven participants (users *U1* to *U7*) to perform our evaluation. They are computer or data scientists from industry and academia with three to ten years of experience in classification tasks. We employed DS7 (as shown in Table I) for our assessment. DS7 was chosen due to its quantitative results, but any dataset could have been used.

The evaluation follows a five-step procedure:

1) *Introduction*. We provided a brief explanation of the study's purpose to the participants.
2) *Tool exposure*. We show the participants the necessary functionalities of OCEG.
3) *User familiarization*. Participants had 10 minutes to explore the system, using a dataset different from DS7.
4) *Evaluation*. We invited the participants to perform the generation of the best ensembles using DS7.

B. Results

We summarize our results in Table II. For use as a baseline, we built an ensemble using all available classifiers, achieving an accuracy of 89.70%. As can be noticed, all participants outperform the baseline with values ranging from 2% to 9% with varying times. For instance, **U5** obtained a high-scoring ensemble in just two minutes but produced just two ensembles. This means that the user feels the task is accomplished with his/her first attempts. On the other hand, **U6** employs more than 10 minutes of searching for a better result; however, the user achieves similar results by generating more sets. Finally, in a time below the average, **U4** performs the best result of our evaluation (98.60%) by rapidly building two sets.

Our results show that OCEG can be handled by users with varying analytical skills. Its simplicity and effectiveness make it a potent alternative for classification tasks.

VI. Discussion

Our case study highlighted the effectiveness and user-friendliness of our tool. Initially, the approach might not have seemed conventional because it involved using all individual classifiers, regardless of their performance levels. However, our first ensemble, which included all available classifiers, outperformed its individual members, demonstrating the advantages of ensemble methods in classification tasks. Additionally, by applying the pruning methods available in our tool, we were able to enhance the ensemble in terms of both runtime and performance, demonstrating the practical value of our tool in real-world applications.

Although ensembles do not work for all classification problems (DS6 and DS12), using classifier ensembles with a suitable setup provides a general new model that potentially improves its predictive power. For instance, in DS4, our tool only produced an optimal ensemble employing two members. We also found that, in some cases, even the ensembles with no tuning can be better than individual classifiers (*i.e.* DS3).

The best improvements occurred in the DS7, DS11, and DS17 datasets compared to the baselines. In six of the experiments, it was necessary to use all classifiers to obtain the best performance. Although some classifiers achieve a low score, when combined with other classifiers, they contribute to the overall improvement.

Determining classifier weights is the main challenge when combining models. OCEG addresses this by letting analysts interactively test different values and create competitive ensembles. However, effective use still requires a base understanding of classification and ensemble methods.

VII. Conclusion

We introduced OCEG, an interactive visual tool designed to help create optimal classifier ensembles. It enables analysts to visualize the decision boundaries and confusion matrices of individual classifiers, facilitating the development of ensembles that outperform single classifiers. The tool also helps identify underperforming classifiers for improvement or removal. The effectiveness of our methodology is demonstrated through a case study and evaluation across 15 datasets, showing its efficiency in generating optimal ensembles.

Acknowledgments

The authors would like to thank the Department of Computing and the Research Directorate of Universidad Catolica San Pablo for their support in the development of this work.

References

[1] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.

[2] P. Dhruv and S. Naskar, "Image classification using convolutional neural network (cnn) and recurrent neural network (rnn): a review," *Machine Learning and Information Processing: Proceedings of ICMLIP 2019*, pp. 367–381, 2020.

[3] D. R. Beddiar, B. Nini, M. Sabokrou, and A. Hadid, "Vision-based human activity recognition: a survey," *Multimedia Tools and Applications*, vol. 79, pp. 30 509–30 555, 2020.

[4] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, no. 2, pp. 241–258, 2020.

[5] F. Cheng, Y. Ming, and H. Qu, "Dece: Decision explorer with counterfactual explanations for machine learning models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1438–1447, 2021.

[6] G. Garcia-Zanabria, D. A. Gutierrez-Pachas, G. Camara-Chavez, J. Poco, and E. Gomez-Nieto, "Sda-vis: A visualization system for student dropout analysis based on counterfactual exploration," *Applied Sciences*, vol. 12, no. 12, 2022.

[7] J. J. Ticona, L. G. Nonato, C. T. Silva, and E. Gomez-Nieto, "Sdr-explorer: A user-friendly visual tool to support preventing student dropouts in higher education," *Computers & Graphics*, vol. 132, p. 104375, 2025.

[8] A. Hinterreiter, P. Ruch, H. Stitz, M. Ennemoser, J. Bernard, H. Strobelt, and M. Streit, "Confusionflow: A model-agnostic visualization for temporal analysis of classifier confusion," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 2, pp. 1222–1236, 2022.

[9] J. Zhang, Y. Wang, P. Molino, L. Li, and D. S. Ebert, "Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 364–373, 2019.

[10] K. M. Alam, N. Siddique, H. Adeli *et al.*, "A dynamic ensemble learning algorithm for neural networks," *Neural Computing and Applications*, vol. 32, no. 12, pp. 8675–8690, 2020.

[11] J. M. Moyano and S. Ventura, "Auto-adaptive grammar-guided genetic programming algorithm to build ensembles of multi-label classifiers," *Information Fusion*, vol. 78, pp. 1–19, 2022.

[12] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[13] P. C. Ribeiro, G. G. Schardong, S. D. Barbosa, C. S. de Souza, and H. Lopes, "Visual exploration of an ensemble of classifiers," *Computers & Graphics*, vol. 85, pp. 23–41, 2019.

[14] M. Gleicher, A. Barve, X. Yu, and F. Heimerl, "Boxer: Interactive comparison of classifier results," in *Computer Graphics Forum*, vol. 39, no. 3. Wiley Online Library, 2020, pp. 181–193.

[15] A. Hinterreiter, P. Ruch, H. Stitz, M. Ennemoser, J. Bernard, H. Strobelt, and M. Streit, "Confusionflow: A model-agnostic visualization for temporal analysis of classifier confusion," *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[16] A. Theissler, M. Thomas, M. Burch, and F. Gerschner, "Confusionvis: Comparative evaluation and selection of multi-class classifiers based on confusion matrices," *Knowledge-Based Systems*, vol. 247, p. 108651, 2022.

[17] A. Chatzimparmpas, R. M. Martins, K. Kucher, and A. Kerren, "Stackgenvis: Alignment of data, algorithms, and models for stacking ensemble learning using performance metrics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1547–1557, 2021.

[18] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2018.

[19] M. Mohandes, M. Deriche, and S. O. Aliyu, "Classifiers combination techniques: A comprehensive review," *IEEE Access*, vol. 6, pp. 19 626–19 639, 2018.

[20] J. Castro-Ochante, G. Camara-Chavez, and E. Gomez-Nieto, "Ice: a visual analytic tool for interactive clustering ensembles generation," in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 2021, pp. 400–408.

[21] C. Qian, Y. Yu, and Z.-H. Zhou, "Pareto ensemble pruning," 2015.

[22] Y. Bian, Y. Wang, Y. Yao, and H. Chen, "Ensemble pruning based on objection maximization with a general distributed framework," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3766–3774, 2020.

[23] P. Joia, D. Coimbra, J. A. Cuminato, F. V. Paulovich, and L. G. Nonato, "Local affine multidimensional projection," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2563–2571, 2011.