

# MetaStackVis: Visually-Assisted Performance Evaluation of Metamodels

Ilya Ploshchik\*  
Linnaeus University

Angelos Chatzimpampas†  
Linnaeus University

Andreas Kerren‡  
Linnaeus University  
Linköping University



Figure 1: Comparing alternative metamodels' predictive performance with MetaStackVis: (a) a panel for the selection of UMAP hyperparameters and the active cluster; (b) a stacked bar chart for identifying the best-performing metamodel for each cluster of base models by analyzing validation metrics and confidence scores; (c) a UMAP plot that aggregates the results of the predicted probabilities and the metric-based performance for the base models and metamodels of the active cluster; and (d) a zone-based matrix that combines all pairs of metamodels to designate the possible benefits of an extra stacking layer due to the juxtaposition of the soft voting outcome (cf. grid of points) versus the optimal outcome from the merger between two metamodels (see gauge charts).

## ABSTRACT

Stacking (or stacked generalization) is an ensemble learning method with one main distinctiveness from the rest: even though several base models are trained on the original data set, their predictions are further used as input data for one or more metamodels arranged in at least one extra layer. Composing a stack of models can produce high-performance outcomes, but it usually involves a trial-and-error process. Therefore, our previously developed visual analytics system, StackGenVis, was mainly designed to assist users in choosing a set of top-performing and diverse models by measuring their predictive performance. However, it only employs a single logistic regression metamodel. In this paper, we investigate the impact of alternative metamodels on the performance of stacking ensembles using a novel visualization tool, called MetaStackVis. Our interactive tool helps users to visually explore different singular and pairs of metamodels according to their predictive probabilities and multiple validation metrics, as well as their ability to predict specific problematic data instances. MetaStackVis was evaluated with a usage scenario based on a medical data set and via expert interviews.

**Index Terms:** Human-centered computing—Visualization—Visualization systems and tools

\*e-mail: ip222gs@student.lnu.se

†e-mail: angelos.chatzimpampas@lnu.se

‡e-mail: andreas.kerren{@lnu.se, @liu.se}

## 1 INTRODUCTION

Stacking (also called *stacked generalization* [30]) is a machine learning (ML) paradigm that operates with heterogeneous ML models arranged in one or more layers, where each subsequent layer summarizes the previous ML models' predictions [24]. A model in such a context represents the output structure of the ML algorithm after fitting it to data and selecting a specific hyperparameter set. In its simplest form, the stacking ensemble is composed of two layers: layer 0, which comprises multiple base models, and layer 1, which contains one or more metamodels [6]. Stacking is a popular method that typically increases the predictive performance due to the deployment of several models. It can also attain a low bias and low variance simultaneously [13], especially when juxtaposed against a single ML model [29]. However, one should ensure getting the top-performing and diverse models by carefully choosing ML algorithms on the underlying and metamodel layers [21].

To eliminate—as much as possible—trial-and-error processes and manage the complete procedure of building impactful stacking ensembles, interactive visual analytics (VA) solutions have been demonstrated to be very effective [7]. Nevertheless, there has been little work to formally investigate further the influence of alternative metamodels on predictive performance (e.g., by visually comparing various metamodels). In a stacking ensemble scenario, Latha and Jeeva [14] found that the random forest algorithm is better in terms of predictive capability when used as a metamodel compared to random trees for the particular healthcare data set they examined. However, the choice of different metamodels depends on the given problem, and the applications of the same metamodels to local subsets of base models instead of all ML models could impact their prediction [32].

Furthermore, the current literature poorly covers the detection of behavioral differences between the most powerful base models’ predictions and the metamodels’ results. Finally, an indication that an additional layer would be beneficial when composing a stacking ensemble is another interesting but yet unexplored issue.

This short paper presents a follow-up analysis of the impact of different metamodels on the prediction and works as a stand-alone extension tool of our IEEE VAST’20 paper on StackGenVis [7], a *source-available VA system* [27] for constructing high-performance stacking ensembles. StackGenVis is the first VA system [5] specially designed to visually monitor and handle the stacking process from scratch with the selection of diverse algorithms and concrete models, including data wrangling support. Its current version was developed using logistic regression with default hyperparameters at the meta-model layer. In this work, we present MetaStackVis (see Fig. 1), an *interactive and open-source visualization tool* [19] for exploring alternative metamodels after extracting base models from StackGenVis. In addition to comparing various models’ configurations based on multiple validation metrics and predictive probabilities (or *confidence*) on the basic and metamodel levels, we also combined pairs of metamodels and contrasted their overall predictive power, as well as local performance for continuously misclassified data instances.

## 2 METASTACKVIS: SYSTEM OVERVIEW

MetaStackVis is a visualization tool implemented in Jupyter Notebook [12] with Plotly [23] as the visualization library and Scikit-Learn [22] for ML purposes. It is then deployed using Streamlit [28].

**Data Loading Tab.** In order to employ this visualization tool, users have to initially experiment with the most recent publicly available version of StackGenVis and extract the predicted probabilities for each data instance and the scores of all validation metrics for the ML models they prefer in the form of two separate CSV files. The latter file with the scores should also contain the hyperparameters used for every ML algorithm to produce the exported ML models. In this section, we explain MetaStackVis with the beginner-friendly Breast Cancer Wisconsin data set [10]. The data set includes records for 699 breast cancer cases, labeled as either benign or malignant depending on nine features. Through the StackGenVis system, we split the data into training and testing sets with an 80/20 ratio. For each of the 11 supported ML algorithms, StackGenVis uses pre-defined hyperparameter sets to generate concrete models. In all our experiments, we selected the top 5 base models in terms of overall performance per algorithm, leading to 55 base models in total.

**HDBSCAN Clustering Tab.** Afterwards, the base models are grouped into separate clusters using HDBSCAN [3] according to their predicted probabilities for all test instances. HDBSCAN is a state-of-the-art algorithm that focuses on creating groupings with base models that predict similar testing data subsets [?]. MetaStackVis allows users to test different hyperparameter combinations for this clustering algorithm (see the first three columns of Table 1) while optimizing the solution for the *density-based clustering validation* (DBC<sub>V</sub>) [20] and the *coverage* scores, concurrently. The former is calculated by computing the density within a cluster and the density between clusters to contribute to the weighted sum of “validity index” values of clusters, with higher density within a cluster and lower density between clusters indicating better results than the opposite case. The latter represents how many base models fall into specific clusters; thus, all base models that belong to an existing cluster, except for the *outliers/noisy data*, are divided by the 55 base models. The last three columns of Table 1 present the results for the top 5 cluster compositions with the best-found DBC<sub>V</sub>, Coverage, and a combination of both scores. This table is sorted in descending order based on the multiplication of DBC<sub>V</sub> and coverage. The goal is to minimize the number of outliers due to a high coverage score and keep the cluster formations—as precise as possible—with the DBC<sub>V</sub> heuristic. However, users can still manually choose the hy-

Table 1: Different hyperparameter sets are being used for the HDBSCAN clustering to find the top 5 similarly-performing cluster compositions. The table is sorted according to the *DBC<sub>V</sub>.Coverage* column.

min_cluster_size	min_samples	metric	n_clusters	DBC <sub>V</sub>	Coverage	DBC <sub>V</sub> .Coverage
3	5	manhattan	4	0.32	0.42	0.14
3	5	euclidean	4	0.10	0.71	0.07
5	5	chebyshev	2	0.06	1.00	0.06
4	5	manhattan	3	0.08	0.71	0.05
3	30	manhattan	3	0.00	0.24	0.00

perparameters from a pre-determined list of options. MetaStackVis facilitates users with either testing a new hypothesis or picking the *default* cluster division according to the highest *DBC<sub>V</sub>.Coverage*.

**Visualization Tab.** After exploring the desired cluster compositions, the *n\_clusters* column defines the number of clusters. The division of base models in them occurs from the HDBSCAN algorithm. We proceed with the defaults, resulting in: *cluster\_0* and *cluster\_1* containing 10 models each, *outliers* (i.e., unassigned base models by HDBSCAN) forming a cluster of 32 models, and *cluster\_2* including 3 models as illustrated in Fig. 1(b), in parentheses. Next, the 11 metamodels will be trained upon the base models of all the aforementioned setups. We distill the hyperparameters from every ML algorithm’s top-performing base model to use them for the metamodels originating from the same algorithms. However, a plethora of hyperparameter tuning alternatives can be found in the literature [25] (cf. Sect. 4 for such limitations). The Visualization tab in Fig. 1(a) incorporates options for selecting UMAP hyperparameters [18] and the cluster under investigation, as well as three different views: *stacked bar chart*, *UMAP plot*, and *zone-based matrix*.

The **stacked bar chart** in Fig. 1(b) presents the best-performing metamodel in each cluster, including all base models and the group of outliers for seven different validation metrics also supported by StackGenVis. This visualization provides an overview of performance (in percentage % format) for the best candidate from the 11 metamodels created in every cluster, using the following metrics: Accuracy, Precision, Recall, ROC AUC, Geometric Mean, Matthews Correlation Coefficient (CorrCoeff), F1 Score, and Confidence. The last metric is the *average predicted probability* for all test instances. Additionally, we convert Matthews CorrCoeff to an absolute value ranging from 0 to 100%. The average of all seven validation metrics plus the confidence is then divided by 2 in order to compute the Overall Performance that defines the ranking of the clusters from top to bottom in this visualization (i.e., from best to worst). Therefore, Confidence is multiplied seven times to capture the same space as all validation metrics because users should be able to compare the two main components of overall performance globally. The legend for this view maps the metrics to the different color encodings. If a user deems a metric useless for the given problem, they can deselect this metric and temporarily hide it. If we compare the total length of the stacked bars in Fig. 1(b), *cluster\_0* contains only 10 instead of 55 base models and reaches the highest overall performance with Linear Discriminant Analysis as the metamodel.

The **UMAP plot** in Fig. 1(c) enables the visual exploration of the base models belonging to the active cluster selected before and the 11 metamodels summarizing their predictions. Hence, offering a deeper behavioral analysis of all metamodels in contrast to the base models. Each point is one model, with base models being smaller in size, while the opposite is true for the metamodels. The UMAP projects the high-dimensional predicted probabilities calculated for the provided data set into two dimensions. In our example, groups of points represent clusters of models that perform similarly according to 140 test instances (which is the 20% testing set). A summary of the performance of each model according to the average value computed from the seven validation metrics is designated as Metric-Based Performance in Fig. 1(c) and is being color-encoded using

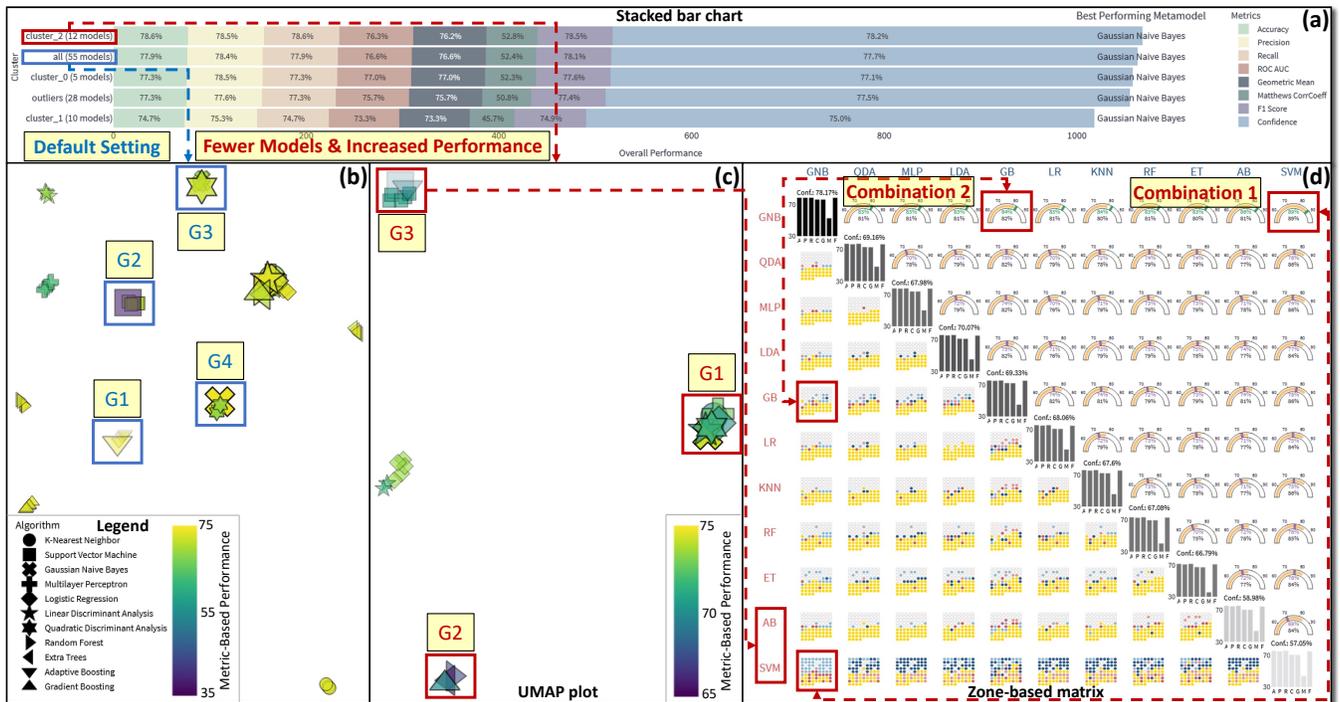


Figure 2: The investigation of *all* and *cluster\_2* comprising 12 base models. View (a) presents the performance of the best-performing metamodel for each cluster according to the seven validation metrics and confidence. The UMAP visible in (b) gathers base models and metamodels predicting similarly the same test instances in groups (Gs) such as G1–G4. On the other hand, (c) visualizes *cluster\_2*, with G1 showcasing that most of the metamodels perform identically, G2 solely with tree-based ML algorithms, and G3 with the two most unconfident metamodels. The unification of predictions from pairs of diverse metamodels is also possible as seen in (d), leading to two promising combinations.

the Viridis colormap [16]. The legend on the left-hand side of this visualization maps the different algorithms as 11 distinguishable symbols for each ML algorithm. For example, the right-pointing arrows are the models constructed from random forest and the left-pointing arrows from extra trees. The opacity of the models is used for the confidence previously introduced, with a higher value forcing the ML model to be more opaque and vice versa.

The **zone-based matrix** in Fig. 1(d) is inspired by the scatterplot matrix [4], and it provides a more comprehensive perspective of the metamodels’ performance. We designed three different zones: the matrix diagonal, the lower triangular part, and the upper triangular part. A bar chart in the matrix diagonal visualizes the metric-based performance of the validation metrics individually as a bar. Color and text convey the confidence (Conf.) of each metamodel, ranked from the best- to the worst-performing one, as already explained for view (b) in Fig. 1. Black denotes the highest confidence value, while light gray is the lowest possible. The remaining zones allow users to perform pairwise comparisons between all combinations of metamodels. The lower triangular part demonstrates the union of all misclassified test instances by at least one metamodel pair (20 in our example). The points in the grid are sorted according to the sum of predicted probabilities for all combinations, leading to the easiest-to-classify test samples always being on top (in white, if correctly classified by both metamodels) and the hardest-to-classify at the bottom (in yellow color, if wrongly classified by both metamodels). As a reference model, we apply the soft majority voting strategy [2] (i.e., predicted probabilities being used) with dark red when the row-wise metamodels are unable to overcome the wrong prediction of the blue metamodels and light red in case these metamodels are correct and their confidence surpasses the other metamodel. Thus, more prominent colors highlight the points and demonstrate the failure of metamodels to predict these points correctly. On the contrary, the upper triangular part is about the “theoretically achievable maximum” predictive performance if the optimal metamodel was selected for

all the test instances (140 in our case). The gauge charts represent the average of all validation metrics’ performance in orange (and in the black text below) and the higher or lower confidence value compared to this metric-based performance in green or purple colors, respectively. The exploration of metamodel pairs in Fig. 1(d) aims to indicate the available room for other schemas, such as establishing an extra stacking layer to aggregate the predictions of this layer.

### 3 USAGE SCENARIO

Supposedly Mia is a data scientist working in a hospital. She receives data about 268 positive and 500 negative patients with eight features related to Pima Indian Diabetes [26]. Her task is to improve predictive performance using stacking ensembles. After exploring the data, ML algorithms, and base models with StackGenVis, she deploys MetaStackVis to experiment with alternative metamodels.

**Overview.** First, Mia loads the exported data into MetaStackVis and chooses the setting for attaining the highest score for the HDBSCAN clustering algorithm. The hyperparameters are automatically set to 3, 5, and Chebyshev metric for the first three column headers visible in Table 1, respectively; thus, resulting in four clusters in total with a DBCV of 0.17 and Coverage of 0.49. All clusters are observable in Fig. 2(a), with *cluster\_2* consisting of fewer models (i.e., 12 base models instead of 55) and at the same time achieving better overall performance for the best-performing metamodel compared to *all*. Mia focuses on the UMAP plots for both settings, using the Manhattan metric because it is preferable for high-dimensionality [1].

**Detailed Exploration of Base Models and Metamodels.** An interesting finding when exploring *all* in Fig. 2(b) is that the base models are grouped depending on their origin and, in general, reach similar metric-based performance regardless of the hyperparameters chosen. The same effect can be observed for the metamodels: for example, Group 1 (G1) and G2 are formed because they predict similarly the same test instances. However, Mia is surprised by the fact that the metamodel in G3 and G4 have swapped places, with Gaus-

sian Naive Bayes (GNB) being closer to Quadratic Discriminant Analysis and vice versa. This pattern requires a deeper investigation.

**Pairwise Combination of Metamodels.** Mia continues her exploration with the more efficient and effective *cluster\_2*. From Fig. 2(c), she acknowledges that most of the metamodels belong to **G1**. An exception here is **G2**, which only contains tree-based ML algorithms such as Gradient Boosting (GB) and **G3** with Adaptive Boosting and Support Vector Machine (SVM). The last group's metamodels have the lowest confidence according to Fig. 2(d), but Mia sees the benefit of combining diverse metamodels. One candidate is the combination of GNB and SVM with 89% theoretical maximum prediction score and confidence (see gauge charts). When combined with a majority voting strategy, the GNB can correctly predict most easy test instances on top, as shown in the grid of points with light blue color, but SVM fails to overcome GNB for the difficult-to-classify test cases at the bottom (due to dark red colors). Another eye-catching combination is GNB and GB, with 82% hypothetical maximum metric-based performance and 84% confidence. According to the grid of points, except for the misclassified instances in yellow, four cases are correctly classified by GB and one by GNB but neither can reach higher confidence than its counterpart. Consequently, Mia understands that a potential extra layer summarizing this second layer's predictions could improve the combination of those metamodels.

## 4 EVALUATION

We performed online semi-structured interviews with four experts asynchronously to obtain qualitative feedback on the usefulness of MetaStackVis, using the same criteria from prior works [8, 9, 17, 31].

**Participants.** The first ML expert (**E1**) is a senior lecturer in mathematics with a PhD in this field and has 4.5 years of experience with ML. The second ML expert (**E2**) is an assistant professor focusing on ML and deep learning with 7.5 years of experience in ML. The third VA expert (**E3**) is a senior lecturer working with clustering and dimensionality reduction, and he has 6 years of experience with ML. The fourth VA expert (**E4**) is a senior lecturer focusing on natural language processing and applied ML with approximately 10 years of experience. The first three experts have reported no colorblindness issues. Although **E4** has a mild case of colorblindness, he affirmed having no difficulty accurately recognizing the specific color combinations we utilized in the MetaStackVis implementation.

**Methodology.** Each interview lasted about one hour, and the interviews were conducted as follows: (1) an introduction to our visualization tool's main goals; (2) a display of the functionality of each visualization and interaction with the tool using the Pima Indian Diabetes data set; and (3) a discussion of the processes followed to arrive at the findings in Sect. 3. We asked the participants to comment on anything. Their major points are summarized below.

**Workflow.** **E2** and **E4** mentioned that the overall proposed workflow makes sense and is appropriately reflected in the spatial arrangement of the views. In particular, **E2** commented on the progressive analysis of the different base model clusters and the produced metamodels as a positive aspect of MetaStackVis. However, **E1** and **E2** suggested a different approach for selecting clusters: concentrate first on the UMAP plot and let users pick clusters of interest with different base models, but they both admitted that using a user-controlled HDBSCAN clustering should be a practical starting point before the manual cluster exploration. **E1** stated that the MetaStackVis workflow requires a thorough understanding of stacking ensembles and access to the publicly-available StackGenVis source code [27]. Since it may also be advantageous to feed the generated human knowledge from MetaStackVis back to StackGenVis, as **E4** framed it, we intend to unify all features in a "single tool" solution.

**Visualization and Interaction.** **E3** had an overall good impression regarding the choice of visual representations to map the computed data. **E1** was amazed by the details provided in MetaStackVis but admitted that it could be slightly overwhelming when one sees

the tool for the first time. Specifically, **E2** and **E3** mentioned that the stacked bar chart and the UMAP plot are easy to interpret, but the zone-based matrix can be challenging to grasp and certainly needs additional time to understand. An improvement here could be to highlight the left-hand side with all confusing data points for the combinations of the 11 metamodels and the right-hand side with the gauge charts when users hover over either one of them because it would become easier to perform pairwise comparisons of the fused metamodels, as **E2** pointed out. Nevertheless, **E3** supported the idea that ordering metamodels from the best- to the worst-performing one helps interpret the zone-based matrix. Here, **E3** and **E4** mentioned that the better-performing pairs of metamodels could be considered future candidates for input to a potential third layer of metamodels. **E3** was fascinated by the clusters visible in the UMAP plot, which translates to the prediction capability of each model in the test data set. **E3** mentioned that this visualization illustrates our successful extraction of useful information and patterns related to cluster structure. He then continued: "it would be complicated to transform that information into reliable insights about the performance without support from the other views". **E2** agreed that the UMAP plot could suggest how differently the base models perform compared to the metamodels. **E3** proposed to segmentize and vertically align each metric instead of the global sorting, but he understands that will affect the global ranking. This observation partially matches with **E1**'s comment to focus on one or a group of validation metrics at a time and not have all seven visualized simultaneously. Although MetaStackVis already allows users to hide irrelevant metrics, this feature should be implemented in the future (as with StackGenVis).

**Limitations.** *Efficiency and scalability* were two concerns raised by **E4**. The former refers to the required computation time to render all views. However, this does not threaten interactivity as long as everything gets parallelized and/or pre-computed beforehand [15]. For the latter case, he pointed out the tool's limitation to visualize a much larger data set with more difficult-to-predict instances due to the increased space demand for the zone-based matrix. A simple solution to this problem could be filtering, which applies to scenarios where some metamodel pairs are performing poorly. As **E2** stated, the tool works solely with *binary classification problems* and does not support *alternative hyperparameter optimization techniques* [11]. **E1** referred to the important role that metamodels' confidence plays in the data exposition, but instead of being aggregated as in our tool, it could be beneficial to use *individual visual representations of spread*. He continued to say that it is necessary to visualize the data distribution on demand to better relate to the underlying *explanation of why some instances are constantly misclassified*. In the future, we plan to improve MetaStackVis to overcome such limitations.

## 5 CONCLUSION

In this paper, we presented MetaStackVis, a visualization tool that enables users to visually assess the performance of metamodels in stacking ensemble learning. It allows users to tune HDBSCAN and apply metamodels to different cluster compositions of base models. Users can also compare the metamodels based on seven validation metrics and their average predicted probability, observe the performance similarities with the underlying base models, and check for powerful pairwise combinations of metamodels that hint at the possible benefit of introducing an extra stacking layer. The applicability and effectiveness of MetaStackVis were evaluated using a real-world healthcare data set and interviews with four experts, who suggested that the comparison of alternative metamodels with our tool is promising. Finally, they helped us recognize the current limits of MetaStackVis, which we will work on in the future.

## ACKNOWLEDGMENTS

This work was partially supported through the ELLIIT environment for strategic research in Sweden.

## REFERENCES

- [1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. In *Proceedings of the International Conference on Database Theory (ICDT)*, pp. 420–434. Springer Berlin Heidelberg, 2001.
- [2] J. Cai, J. L. Garner, and R. A. Walkling. A paper tiger? An empirical analysis of majority voting. *Journal of Corporate Finance*, 21:119–135, 2013. doi: 10.1016/j.jcorpfin.2013.01.002
- [3] R. J. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 160–172. Springer, 2013.
- [4] D. B. Carr, R. J. Littlefield, W. L. Nicholson, and J. S. Littlefield. Scatterplot matrix techniques for large N. *Journal of the American Statistical Association*, 82(398):424–436, 1987.
- [5] A. Chatzimpampas, R. M. Martins, I. Jusufi, K. Kucher, F. Rossi, and A. Kerren. The state of the art in enhancing trust in machine learning models with the use of visualizations. *Computer Graphics Forum*, 39(3):713–756, June 2020. doi: 10.1111/cgf.14034
- [6] A. Chatzimpampas, R. M. Martins, K. Kucher, and A. Kerren. Empirical study: Visual analytics for comparing stacking to blending ensemble learning. In *Proceedings of the 23rd International Conference on Control Systems and Computer Science (CSCS)*, pp. 1–8. IEEE, 2021.
- [7] A. Chatzimpampas, 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*, 27(2):1547–1557, Feb. 2021. doi: 10.1109/TVCG.2020.3030352
- [8] A. Chatzimpampas, R. M. Martins, K. Kucher, and A. Kerren. VisEvol: Visual analytics to support hyperparameter search through evolutionary optimization. *Computer Graphics Forum*, 40(3):201–214, June 2021. doi: 10.1111/cgf.14300
- [9] A. Chatzimpampas, R. M. Martins, K. Kucher, and A. Kerren. FeatureEnVi: Visual analytics for feature engineering using stepwise selection and semi-automatic extraction approaches. *IEEE Transactions on Visualization and Computer Graphics*, 28(4):1773–1791, 2022. doi: 10.1109/TVCG.2022.3141040
- [10] D. Dua and C. Graff. UCI Machine Learning Repository, 2017.
- [11] M. Feurer and F. Hutter. Hyperparameter optimization. In *Automated Machine Learning: Methods, Systems, Challenges*, pp. 3–33. Springer International Publishing, 2019. doi: 10.1007/978-3-030-05318-5\_1
- [12] Jupyter — A web-based interactive computing platform. <https://jupyter.org>, 2014. Accessed December 10, 2022.
- [13] R. Kohavi and D. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the International Conference on Machine Learning, ICML '96*, pp. 275–283. Morgan Kaufmann Publishers Inc., 1996.
- [14] C. B. C. Latha and S. C. Jeeva. Improving the accuracy of prediction of heart disease risk based on ensemble classification techniques. *Informatics in Medicine Unlocked*, 16:100203, 2019. doi: 10.1016/j.imu.2019.100203
- [15] J. K. Li and K.-L. Ma. P4: Portable parallel processing pipelines for interactive information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 26(3):1548–1561, 2020. doi: 10.1109/TVCG.2018.2871139
- [16] Y. Liu and J. Heer. Somewhere over the rainbow: An empirical assessment of quantitative colormaps. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI '18*, pp. 598:1–598:12. ACM, 2018. doi: 10.1145/3173574.3174172
- [17] Y. Ma, T. Xie, J. Li, and R. Maciejewski. Explaining vulnerabilities to adversarial machine learning through visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1075–1085, Jan. 2020. doi: 10.1109/TVCG.2019.2934631
- [18] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *ArXiv e-prints*, 1802.03426, Feb. 2018.
- [19] MetaStackVis code. <http://bit.ly/MetaStackVis-code>, 2022. Accessed December 10, 2022.
- [20] D. Moulavi, P. A. Jaskowiak, R. J. G. B. Campello, A. Zimek, and J. Sander. Density-based clustering validation, pp. 839–847. 2014. doi: 10.1137/1.9781611973440.96
- [21] A. I. Naimi and L. B. Balzer. Stacked generalization: An introduction to super learning. *European Journal of Epidemiology*, 33(5):459–464, 2018.
- [22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, Nov. 2011. doi: 10.5555/1953048.2078195
- [23] Plotly — Python open source graphing library. <https://plot.ly>, 2013. Accessed December 10, 2022.
- [24] O. Sagi and L. Rokach. Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4):e1249, July–Aug. 2018. doi: 10.1002/widm.1249
- [25] M. Shahhosseini, G. Hu, and H. Pham. Optimizing ensemble weights and hyperparameters of machine learning models for regression problems. *Machine Learning with Applications*, 7:100251, 2022. doi: 10.1016/j.mlwa.2022.100251
- [26] J. Smith, J. Everhart, W. Dickson, W. Knowler, and R. Johannes. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium Computer Application in Medical Care*, pp. 261–265. American Medical Informatics Association, 1988.
- [27] StackGenVis code. <http://bit.ly/StackGenVis-code>, 2021. Accessed December 10, 2022.
- [28] Streamlit — The fastest way to build and share data apps. <https://streamlit.io>, 2020. Accessed December 10, 2022.
- [29] K. M. Ting and I. H. Witten. Stacked generalization: When does it work? In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence — Volume 2, IJCAI '97*, pp. 866–871. Morgan Kaufmann Publishers Inc., 1997.
- [30] D. H. Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.
- [31] K. Xu, M. Xia, X. Mu, Y. Wang, and N. Cao. EnsembleLens: Ensemble-based visual exploration of anomaly detection algorithms with multidimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):109–119, Jan. 2019. doi: 10.1109/TVCG.2018.2864825
- [32] X. Zhang, J. P. Ono, H. Song, L. Gou, K.-L. Ma, and L. Ren. SliceTeller: A data slice-driven approach for machine learning model validation. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–11, 2022. doi: 10.1109/TVCG.2022.3209465